Survivability Architecture for Workflow Management Systems

Jorge Cardoso, Zongwei Luo, John Miller, Amit Sheth and Krys Kochut
LSDIS Lab, Computer Science Department
University of Georgia

Abstract

The survivability of critical infrastructure systems has been gaining increasing concern from the industry. The survivability research area addresses the issue of infrastructure systems that continues to provide pre–established service levels to users in the face of disorders and react to changes in the surrounding environment. Workflow management systems need to be survivable since they are used to support critical and sensitive business processes. They require a high level of dependability and should not allow process instances to be interrupted or aborted due to failures. Moreover, due to their sensitivity, business process should reflect any change in the environment. In this paper we describe the work on increasing the survivability of the METEOR workflow management system. We define an architecture describing the main technologies that enable survivability in WfMS. Using the developed architecture we describe two modules that have been implemented: dynamic change and adaptation module.

1 Introduction

The dependence of infrastructure systems on fragile information systems puts organizations at risk of disastrous failure. Threats that may compromise a systems may be originated from several sources; human error, application fault, security problems, network failure, natural catastrophe, etc. In reaction of such disruptions, diagnostic, corrective procedures and reconfiguration must be taken to ensure that the infrastructures continue to meet the original requirements. Disruptions that can occur are well illustrated by many incidents that have already been reported. Just in the security area, 4,299 security–related incidents on the Internet have been reported to CERT between 1989 and 1995 [Howard, 97].

Categorized as an information system, workflow management systems (WfMS) are used in a broad range of distinct applications. Applications can be more oriented to support or enhance existing processes, to increase competitive advantage, to reduce costs, and also to manage critical infrastructure systems. The applications that are managed by the WfMS have a vital significance to the organizations that govern them. In most cases disruptions of the services provided by the WfMS will incapacitate the completion of the running process instances. Additionally since the business logic is captured by the workflow system and may not be available in any other form, the organization faces the possibility to completely stop the activities represented by the damaged business processes. It is therefore clear that mechanisms must enable the reliability and decrease the risk of disruption that will lead to system breakdown and organization malfunction.

In order to cope with the disruptions that critical systems face new research areas need to be explored. The survivability research area, in the context of information systems, is one of them and it has started to concern an increasing number of people. This is primarily because our society is becoming more and more dependent on computer systems. The survivability keyword describes a class of systems that is able to "complete its mission in a timely manner, even if significant portions are incapacitated by attack or accident" [Barbacci, 96]. [Ellison et al., 97] refines the initial description requiring a survivable system to be able to protect against and react to any kind of attack, failure or accident that, alone or in combination, threatens the ability of a system to fulfill its mission in a timely fashion. Based on this definition we describe workflow survivability as the capability of a workflow management system to maintain a pre–established acceptable running mode and behavior after the occurrence of unexpected errors, accidents, failures or attacks, in a timely manner and to allow the adaptation and evolution of the supported processes in response to its surrounding environment. In our definition we include the need for adaptation and evolution since business processes and their environment are dynamic by nature. In order to respond to the emergent needs expressed by today's systems, the workflow management systems must follow the new requirements and allow the survivability of the entire system.

In this paper we describe a survivability architecture for WfMS. The developed architecture is based on the diverse functional modules that compose a WfMS. Therefore we start by functionally dividing the several components involved in the runtime environment of a workflow management system in four categories: *instance level*, *schema level*, *workflow level* and *infrastructure level.* For each functional level we briefly mention some solutions that may be implemented to increase and guarantee survivability. Finally we present two survivability units implemented for the METEOR WfMS. The first module allows the specification of dynamic changes to running instance of workflow schemas, which is a fundamental feature to allow adaptation and evolution. The second module, an adaptation module, allows the handling of exception generated during the execution of workflow instances based on knowledge acquired about past experiences.

2 METEOR - Workflow Management System

A Workflow Management System (WfMS) is a system or set of tools that completely defines, manages and executes processes schema ("workflows") through the execution of software whose order of execution is driven by a computer representation of the workflow logic [Hollingsworth, 95]. The idea behind the introduction of a WfMS in an organization consists in mapping processes that were executed manually into a workflow scheme that has a binary representation that will be executed and supervised by a computerized system. Such systems clearly give a competitive advantage to an organization, allowing re–engineering practices and streamline, control and automation of existing processes.

In this context, at the LSDIS Lab and with collaboration with the Naval Research Laboratory, we have developed the METEOR workflow management model and system. METEOR's architecture includes design, monitor, workflow repository, and the enactment system. Due to different needs in organizations we have developed two enactment service: ORBWork [Kochut et al., 99] and WEBWork [Miller et al., 97]. ORBWork is a CORBA based system oriented to support mission–critical enterprise applications requiring high scalability and robustness. It is fully distributed and scalable. Since we have used Java as the language for its development the system is portable across platforms. It supports interoperability standards such as JFLOW [jFLOW, 98] and SWAP [Swenson, 98]. The use of open standards such as CORBA makes it a good candidate to interoperate with existing systems in disparate distributed and heterogeneous computing environments. With the recently added modules it also includes dynamic changes at the instance level and an exception handling mechanism that is part of the adaptation module. The concepts used in WEBWork architecture are very similar to the one used in ORBWork system. WEBWork implementation relies solely on Web technology as the infrastructure for the enactment system. It is more suitable for static business processes that involve limited data exchange. The main goal is based on the easy development of workflow application, installation, use and maintenance.

3 Survivability architecture for workflow management systems

As we have mentioned previously the survivability of systems is a complex issue. And it is even more delicate in distributed systems because of the existence of dependability problems that are not frequently encountered in more traditional centralized systems. To develop successful survivability solutions for systems it is necessary to have a clear understanding and precise global vision of their architecture. Survivability purposes impact critical early decisions in system development, it is both cost–effective and efficient to conduct survivability analyses at the architecture level, before substantial resources have been committed to development [Bass et al., 98][Kazman et al., 98].

3.1 Four level architecture for WfMS

Depending on the type of problem that may affect the behavior of a workflow system, different strategies can be used to restore its correct activity. In [Casati, 98] a division and classification of sources of failures in workflow systems is made. The classification identifies two categories of failures: basic and application failures. Basic failures correspond to failures of the WfMS or of its underlying infrastructure, such as hardware failures, network failures, or failures of the DBMS supporting the WfMS. Application failures are related with malfunctions of instances invoked by the WfMS. This classification is not sufficient, and needs to be expanded in order to satisfy the survivability requirements. We classify architecturally failures in workflow management systems in four layers (Figure 1): instance level, schema level, workflow level and infrastructure level. In each of the layers we can identify a distinct classes of problems that a workflow system may encounter and that may jeopardize its survivability. This functional division gives four main architectural areas that need to be addressed. Each one has a specific class of problems that need to be handled properly.

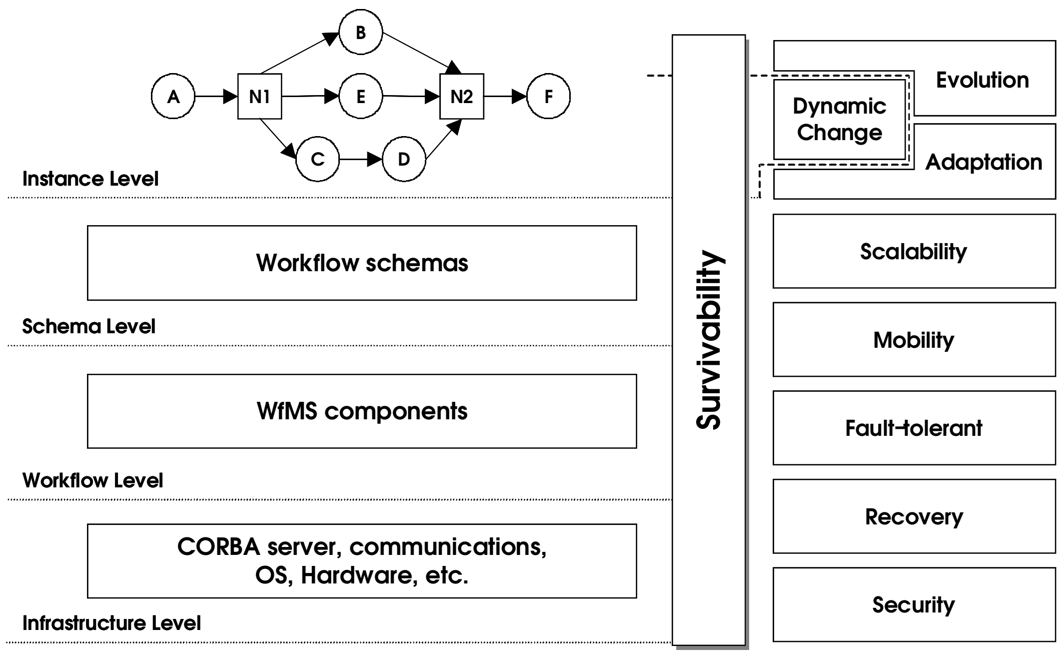

Figure 1 – Survivability architecture for workflow management systems

Instance level

At the instance level layer we find all the issues that are closely related to instances or application execution. In this layer, failures may occur when the design of a workflow schema did not anticipate a possible error related to the execution of a workflow task. For example, a workflow instance is running correctly when a specific task is unable to complete properly. This may be due to the inability to connect to a resource (e.g. DBMS) because of an invalid address or port number, insufficiency of requested resources (e.g. disk space or memory) or unauthorized access to a resource (e.g. ftp server). This anomalous situation is captured, suitably represented and delivered to be handle by the survivability module. Several possible actions may be taken to ensure that the system still continues in a stable and acceptable state: skip the faulty task, retry the task realization, abort current faulty instance, start another workflow instance, request user involvement, dynamically reconfigure the workflow or raise and handle an exception.

Schema level

As the name suggests the schema level regroups the workflow schema definitions. A workflow schema is business process representation that is suitable to be interpreted by a WfMS. A workflow schema is the static view of a workflow instance. Workflow schemas are generally stored in a repository and subsequently used by the WfMS. At this level we need to guarantee that the existing schema exhibit a valid structure according to the organizational context. Furthermore, if an organization adopts mechanisms that allow the evolution or adaptation of schemas, we need to verify the correctness of the generated schemas under the current environment.

Workflow level

At the workflow level we find a layer that represents the modules that compose a workflow management system. Those workflow modules, depending in the workflow system, will typically include the enactment module, the monitor and repository. In this layer, failures may occur when any of the modules is unable to maintain an acceptable behavior. For example, the workflow server managing task *a*, task *b* and task *c* entered an invalid state where no more useful processing is done regarding task *a*. This may be due, for example, to a buffer overflow in the manager of task *a*. In this situation we may possibly restart task manager *a*, abort instances involving task *a*, dynamically reconfigure the instances to exclude task *a,* or request a user involvement.

Infrastructure level

The infrastructure level regroups all the elements that compose the underlying infrastructure that supports the WfMS. It includes CORBA servers, operating systems, communication protocols, hardware, etc. A malfunctioning of one of those components may reveal to be quite complex to recover from. For example, the underlying operating system, where a workflow scheduler was running, suddenly ceases its activity due to a fault in the physical memory. This type of situation is the most serious one since it requires dealing; not only with a problem that happen at the infrastructure level, but also with the recovery of the workflow scheduler that has crashed, that is an workflow level error, and also with the recovery of all the instances that were running, an instance level problem.

3.2 Survivability components

Survivability is a multidisciplinary research area. In the architecture proposed we have identified seven main areas that need a special attention: evolution, adaptation, scalability, mobility [Zukunft, 97], fault–tolerance [Alonso et al., 00][Sabnis et al., 99] [Hiltunen and Schlichting, 96], recovery [Elnozahy et al., 99] and security [Regis et al., 98] [Sullivan et al., 99]. We believe that all this domains need to explored to archive survivable systems. In addition, when dealing with WfMS, we need to consider the above domains for each layer of a WfMS.

In this paper we restrict our study to adaptation at the instance layer, describing the work done in this domain. Two major issues have been addressed in the context of survivability (Figure 2): dynamic change and adaptation. The dynamic change interface allows the modification of running workflow instances. This interface directly creates a necessary and indispensable basic building block to the support of adaptation and evolution at the instance level. We also provide a user interface to the dynamic change interface allowing an administrator to manually modify workflow instances in execution. Supporting dynamic changes significantly increases the flexibility and robustness for a workflow management system to cope with all kind of unplanned events during the execution of the business process. At the adaptation level we have implemented a sophisticated exception handling mechanism that allows the system to adapt, automatically or via human involvement, in response to changes in the environment [Luo et al., 00].

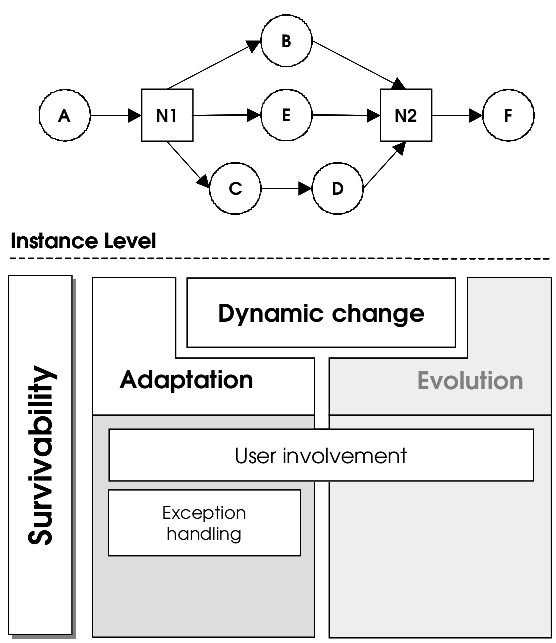

Figure 2 – Survivability architecture (adaptation and evolution) at the instance level

3.2.1 Dynamic changes

Traditional WfMS are adequate to support business processes with a defined structure and with no need for *ad hoc* deviations or dynamic extensions at run–time [Reichert and Dadam, 98]. But, recently there has been an increasing demand in developing WfMS with dynamic capabilities, with a special emphasis to dynamic changes at the workflow instance level. This makes sense since there are in reality very few business processes that are static (i.e. without a need to change their business practices over time). As workflow processes are instantiated, changes in the environment or previous activities may invalidate the current workflow instances, requiring adaptation or evolution. It is therefore required to continuously repair or improve the execution of a workflow process [Berry and Myers, 98]. A good example of the need of dynamic reconfiguration can be found in [Horn and Jablonski, 98]. Additionally, long running heterogeneous autonomous distributed applications, like ORBWork, require support for dynamic reconfiguration since machines fail, services are moved or withdrawn and user requirements change. In such an environment, it is essential that the structure of applications can be modified to reflect such changes [Shrivastava and Wheater, 98]. Therefore one of the objectives of dynamism consists in allow the structural change, both control and data flow, of instances at run–time without interfering with the other instances not implicated in the modification and without loss of run–time performance.

In ORBWork system we have implemented a layer that permits the realization of dynamic change of instances in a consistent manner [Chen, 00]. The implemented module guarantee that all consistency constraints that have been ensured prior to a dynamic change are also ensured after the workflow instances have been modified [Reichert and Dadam, 98].

Classification of dynamic changes

Before implementing the dynamic change extensions to the ORBWork system we made a classification of different types of changes that can be applied to a workflow instance (Table 1). We classify the different types of changes in two main categories: *primitive change* and *composite change*.

Primitive changes are composed of "atomic" changes that can only be applied to process definition totally or not applied at all (e.g., adding a synchronous transition between two tasks). Primitive changes can be further divided into *immediate changes* and *incremental changes*. Immediate changes can be introduced in one step without losing the correctness and consistency of the workflow enactment system. Incremental changes, on the other hand, deal with situations where we cannot apply the changes to a particular instance in a one step procedure. For example, if a set of instances are waiting for the completion of a task *t*, and we dynamically change *t* specifications, the waiting instances may enter an inconsistency state since the information that they have relatively to the previous task *t* does not reflect any more current state of the system. In our work, most of the primitive changes implemented in ORBWork are incremental changes.

Composite changes are composed of a sequence of primitive changes that describe a elaborated process definition change (e.g., adding a task between two existing tasks is the result of applying a sequence of primitive changes).

Dynamic Change	Change Type
AND to OR Join Change	Incremental
OR to AND Join Change	Incremental
Split Change	Incremental
Addition of an AND Transition	Incremental
Addition of an OR Transition	Incremental
Deletion of a Transition	Incremental
Data Object Transfer Addition	Incremental
Data Object Transfer Deletion	Incremental
Parameter Mapping Change	Incremental
Parameter Type Change	Incremental
Task Type Change	Incremental
Task Invocation Change	Composite change
Insertion of a Task	Composite change
Deletion of a Task	Composite change

Table 1 – Dynamic changes classification

Status of implementation

From the classification of different types of changes that can be applied to a workflow instance exposed in Table 1 we have implemented the ones showing in Table 2.

Dynamic Change	Status
AND to OR Join Change	Implemented
OR to AND Join Change	Implemented
Split Change	Implemented
Addition of an AND Transition	Implemented
Addition of an OR Transition	Implemented
Deletion of a Transition	Implemented
Data Object Transfer Addition	Implemented
Data Object Transfer Deletion	Implemented
Parameter Mapping Change	Implemented
Insertion of a Task between Two Tasks	Implemented

Table 2 – Status of implementation

Following the architectural implementation of ORBWork, the dynamic change interface was built on top of the CORBA ORB infrastructure and using IIOP as the underlying communication protocol. Additional functions have been added to the IDL interface of the CORBA object responsible for managing tasks.

3.2.2 Adaptation

As Charles Darwin mentioned – "It is not the strongest species that survive, or the most intelligent, but the one most responsive to change". Adaptation characterizes the ability of a system to adjust to environmental conditions. It is the modification of a system or its parts that makes it more fit for existence under the conditions of its environment. The importance of adaptation has been recognized in several areas, that include software [Margaret, 95][Heineman, 98], database systems and mobile systems [Zukunft, 97], and fault–tolerant systems [Hiltunen and Schlichting, 96].

To better understand the concept of adaptation lets consider a very simple example [Hiltunen and Schlichting, 96]: the Ethernet protocol. It may be not completely obvious but the Ethernet protocol is considered to be an adaptive algorithm. Analyzing its behavior we verify that the protocol increase or decrease the interval after which it tries to resend the message based on the collisions on the broadcast medium. Thus the algorithm changes its behavior in response to changes and events in the environment making it adaptable.

In the domain of workflow management systems we also desire to obtain adaptable features. This permits workflow systems to be prepared to adapt themselves to a range of different business and organization settings and also to a changing context [Han el at., 98]. This requirement is a direct consequence of the highly changeable environment existing around business processes. The environment can be characterize has heterogeneous and is affected, in a global perspective, by events like political decisions, new company polices, new laws and regulations, and changes in global markets. In a more fine grain analysis we may find that the environment also include simple elements, like the people involved in the execution of a business process, or the resources used to archive the goals of the process. Let's consider the following scenario: a workflow instance is running when a task cannot be completed due to the inability to access a DBMS. At this point a change in the internal environment is identified. In consequence an event or exception is generated describing the change in the environment. A competent module subsequently processes the event, with the objective of restoring the environment to a stable state.

Having this scenario in mind we have developed a module that allows the METEOR workflow management systems to be an adaptable system. The module deals with exceptions, a well–defined class of events that may occur during the realization of a process instance.

3.2.2.1 Exception handling a case of adaptation

Although research in workflow management has been very active for several years, and the need for modeling exceptions in information systems has been widely recognized only recently the workflow community has tackled the problem of exception handling [Casati, 98]. An exception refers to facts, situations, or abnormal events no modeled by the underlying workflow management system or deviations between what we plan and what actually happen [Luo et al., 00].

The architecture developed [Luo et al., 00] and implemented in ORBWork runtime system includes a sophisticated exception handling mechanism with the crucial requirement to allow workflow management system to be deployed in cross–

organizational settings. During a workflow schema execution if an exception occurs and it is propagated to the case–based reasoning exception handling module, the CBR system is used to derive an acceptable exception handler [Luo et al., 98]. The system has the ability to adapt itself over time, based on knowledge acquired about past experiences that will help to solve new problems. As the CBR system collects more cases, the global WfMS becomes more resistant, preventing unwanted states, since it has a larger set of knowledge to handle future exception. A simple example is shown in Figure 3.

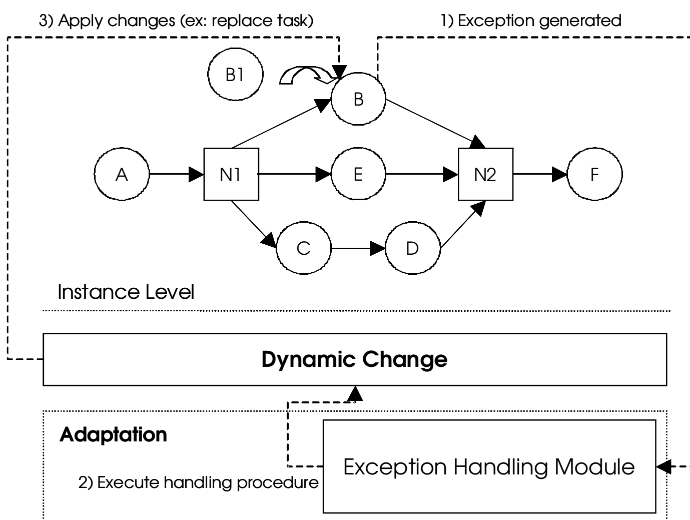

Figure 3 – Example of adaptation using Exception Handling mechanism

System description

A knowledge–based approach of managing the exception handling knowledge is used in our exception handling system. A case–based reasoning (CBR) mechanism is used to improve the exception handling capabilities [Luo et al., 00]. In this approach, information about previous problem solving cases is retrieved to help solve new problems [Luo et al., 00]. During the workflow execution, if an exception is propagated to the CBR based exception–handling component, the case–based reasoning process is used to derive an acceptable exception handler. Human involvement is needed when acceptable exception handlers cannot be automatically obtained. Solutions given by a person will be incorporated into the case repository. Effects of the exception handler candidates on the workflow system and applications will be evaluated. Thus, when the exception is handled necessary modifications to the workflow systems or applications may be made. The exception resolution process is actually the population process of CPR templates. The actual exception resolution performs the following tasks [Luo, 00]:

- The *coordination mode* of exception handling will be determined. The coordination mode will be determined according to the type of process interactions between business processes.
- The *contacting party* as well as interaction point will be determined. A contacting party is one of the entities that are responsible for handling exception in the processes in its organization. An interaction point is where the interactions can take place.
- The *compensation scheme* will be found if necessary. The nature of the processes will affect the compensation schemes. Human involvement is allowed in determining the compensation schemes.
- The *rework scheme* will be found if necessary. Rework scheme is the plan for the processes to make progress from the failure points. Human involvement is allowed in determining the rework schemes.

The retrieval procedure of similar previous cases is based on the similarity measure that takes into account both semantic and structural similarities and differences between the cases. A similarity measure is achieved by get the following [Luo, 00]:

- *Exception similarity*. Exception similarity is based on the is–a relationship in the exception hierarchy in METEOR model 3.
- *Workflow similarity*. It is the workflow structural similarity such as AND, OR building block similarity.
- *Context similarity*. It is obtained by computing nearest neighborhood function of the quantified degrees of semantic similarities over workflow application data. To do so, a concept tree should be built first [Luo, et al., 00]. The distances between concepts will be stored into the case repository.

We use a pattern guided case adaptation scheme [Luo, 00]. There are four steps in the adaptation process in this pattern guided adaptation scheme. The process is really the population process of the CPR handling template [Luo, 00].

- *Classifying the exception pattern*. At this step, the exception pattern will be identified. If it is a new pattern, it will be added to exception pattern repository.
- *Searching the handling pattern*. Once the exception pattern is determined, a search will be conducted for the handling pattern. At this step, the exception handling coordination mode will be determined. Contacting party as well as interaction point is also determined by analyzing the interactions among business processes.
- *Selecting a handler pattern*. A handler pattern will be selected based on the search result from step 2. The compensation scheme as well as the rework scheme will be determined.

- *Initializing the handler*. The CPR handling template will be populated. An adapted case is created.

- **Conclusions**

The new requirements of modern systems in a highly technological society demand that critical systems to be survivable. Survivability addresses a set of characteristic that systems should have in order to be resistant to failures and changes in the surrounding environment. In order to archive those two goals, security, recovery, fault–tolerant, mobility, scalability, adaptation and evolution factors have to be considered. Our work focuses the survivability issues of workflow management systems (WfMS). We have defined a survivable architecture that functionally divide workflow WfMS in a four–layer architecture and include the seven fundamental survivable characteristics mentioned previously. This overall architecture sketches a global picture of the main issues that have to be solved. In this context two main modules have been developed for the METEOR system to increase its survivability: dynamic change and adaptation. Dynamic changes module gives an interface that permits the change of workflow instances. This module is indispensable to allow adaptation and evolution at the instance level. The adaptation module developed deals with exception that occur during instance realization and is based on case base reasoning algorithms.

5 References

[Alonso et al., 00] Alonso, G., Hagen, C., Agrawal, D., El Abbadi, A., Mohan, C. *"Enhancing the Fault Tolerance of Workflow Management Systems"*. IEEE Concurrency, Vol. 8, No.3, pp 74–81, Jul–Sep, 2000.

[Barbacci, 96] M. Barbacci. *"Survivability in the age of vulnerable systems"*. IEEE Computer, 29(11):8, Nov, 1996.

[Bass et al., 98] L. Bass, P. Clements and R. Kazman, *"Software Architecture in Practice"*. Addison Wesley, 1998.

[Berry and Myers, 98] Pauline M. Berry and Karen L. Myers. *"Adaptive Process Management: An AI Perspective"*. The 1998 ACM Conference on Computer Supported Cooperative Work. Seattle, Washington, 1998.

[Casati, 98] Fabio Casati, *"A Discussion on Approaches to Handling Exceptions in Workflows"*. Proceedings of 1998 Computer–Supported Cooperative Work (CSCW 1998), Towards Adaptive workflow Systems Workshop, Seattle, WA, 1998.

[Chen, 00] Yufeng Chen, "Design and Implementation of Dynamic Process Definition Modifications in OrbWork Enactment System". M.Sc. thesis. University of Georgia. July 2000.

[Ellison et al., 97] R.J. Ellison, D. Fisher, R.C. Linger, H. F. Lipson, T. A. Longstaff, and N.R. Mead, *"Survivable Network Systems: An Emerging Discipline"*. Software Engineering Institute Technical Report No. CMU/SEI–97–TR–013, Nov 1997.

[Elnozahy et al., 99] E. Elnozahy, L. Alvisi, Y.M. Wang, and D.B. Johnson. *"A Survey of Rollback–Recovery Protocols in Message–Passing Systems"*. Carnegie Mellon University, Tecnical Report CMU–CS–99–148, Jun 1999.

[Han el at., 98] Yanbo Han, Amit Sheth and Christoph Bussler. *"A Taxonomy of Adaptive Workflow Management"*. Proceedings of 1998 Computer–Supported Cooperative Work (CSCW 1998), Towards Adaptive workflow Systems Workshop, Seattle, WA, 1998.

[Heineman, 98] Heineman, George T. *"Adaptation and Software Architecture"*. ISAW3, Orlando, Florida, USA, check reference, Sept 1998.

[Hiltunen and Schlichting, 96] Hiltunen, Matti A. and Schlichting, Richard D. *"Adaptive Distributed Fault–Tolerant Systems"*. International Journal of Computer Systems Science and Engineering, vol.11, n.5, pp. 125–133, Sep 1996.

[Hollingsworth, 95] David Hollingsworth. *"The Workflow Reference Model"*. Workflow Management Coalition, Document Number TC00–1003, 19 Jan 1995.

[Horn and Jablonski, 98] Stefan Horn, Stefan Jablonski, *"An Approach to Dynamic Instance Adaption in Workflow Management Applications"*. 1998.

[Howard, 97] John D. Howard. "An Analysis Of Security Incidents On The Internet 1989–1995". PhD thesis, Carnegie Mellon University, Pennsylvania USA, Apr 1997.

[jFLOW, 98] OMG BODTF RFP #2 Submission, Workflow Management Facility, Revised Submission, (ftp://ftp.omg.org/pub/docs/bom/98–06–07.pdf), 4 Jul 1998.

[Kazman et al., 98] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, *"The Architecture Tradeoff Analysis Method"*, Proceedings of ICECCS '98, Monterey CA, Aug 1998.

[Klein and Dellarocas, 98] Mark Klein and Chrysanthos Dellarocas, *"A Knowledge–Based Approach to Handling Exceptions in Workflow Systems"*. Proceedings of 1998 Computer–Supported Cooperative Work (CSCW 1998), Towards Adaptive workflow Systems Workshop, Seattle, WA, 1998.

[Luo et al., 00] Zongwei Luo, Amit Sheth, Krys Kochut, and John Miller, *"Exception handling in workflow systems"*, Applied Intelligence: the International Journal of AI, Neural Networks, and Complex Problem–Solving Technologies, Volume 13, Number 2, pp125–147, Sep–Oct 2000.

[Luo, 00] Zongwei Luo, *"Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross–Organizational Business Processes"*, Ph.D. dissertation, Department of Computer Science, University of Georgia, December, 2000.

[Luo et al., 98] Zongwei Luo, Amit Sheth, John Miller, Krys Kochut, *"Defeasible Workflow,its Computation, and Exception Handling"*. Proceedings of 1998 Computer–Supported Cooperative Work (CSCW 1998), Towards Adaptive workflow Systems Workshop, Seattle, WA, 1998.

[Margaret, 95] Davis, J. Margaret. *"Adaptable, Reusable Code"*. SSR, ACM, check reference, Seattle, WA, USA, 1995.

[Kochut at al., 98] Krys J. Kochut, Amit P. Sheth, John A. Miller. "*ORBWork: A CORBA–Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR*". Large Scale Distributed Information Systems Lab, Computer Science Department, University of Georgia, 1999.

[Miller et al., 97] J. Miller, D. Palaniswami, A. Sheth, K. Kochut, and H. Singh. *"WebWork: METEOR's Web–based Workflow Management System"*. Journal of Intelligence Information Management Systems, pp. 185–215, 1997.

[Reichert and Dadam, 98] Manfred Reichert, Peter Dadam, *"ADEPT$_{flex}$–Supporting Dynamic Changes of Workflows Without Loosing Control"*. Journal of Intelligent Information Systems (JIIS), Special Issue on Workflow Management Systems, Vol. 10, No. 2, pp. 93–129, 1998.

[Shrivastava and Wheater, 98] Santosh K. Shrivastava, Stuart M. Wheater, *"Architectural Support for Dynamic Reconfiguration of Distributed Workflow Applications"*. IEE Proceedings – Software Engineering, 145:5, pp. 155–162,1998

[Swenson, 98] Swenson, K., *"SWAP – Simple Workflow Access Protocol"*. 1998.

[Voas et al., 97] Jeffrey M. Voas, Gary E. McGraw, & Anup K. Ghosh, *"Reducing Uncertainty About Survivability"*. Information Survivability Workshop – ISW'97, San Diego, California, Feb 12–13, 1997.

[Zukunft, 97] Olaf Zukunft. *"Rule based Adaptation in Mobile Database Systems"*. Proc. 12th Symposium on Applied Computing, ACM SAC 97, pp. 310–317, Mar 1997.