WorkFlow Systems: a few definitions and a few suggestions

Paulo Barthelmess and Jacques Wainer

Relatório Técnico DCC–95-26

Dezembro de 1995

WorkFlow Systems: a few definitions and a few suggestions[†]

Paulo Barthelmess and Jacques Wainer

Abstract

This paper hopes to make a contribution on three aspects of workflow systems: we stress the fact that there is a broken symetry between the level of the specification of the procedures and the level of their enactment; we propose some ways of classifying activities and exceptions; and we propose some run-time functionalities to help users deal with exceptions.

1 INTRODUCTION

Much work has been devoted to the study of document flow in organizations, yet much remains to be understood about the nature of activities. The focus on the flow (expressed even in the term Workflow) stems from the origins of present systems, derived from the early Office Automation Systems of the 80's. Questions related to the nature of activities, locus of the effective work in the system, as well as those about the flow of responsibilities, temporal concerns and other such important issues are rarely dealt with. The same can be said about exception handling. Some papers discuss the theme, but hardly any implementation actually supporting exception handling exists [Saa94]. Yet, an important share of office work can be charged to exception handling chores [SM89, EN93].

On the other hand, efficiency and rapid response are becoming an imperative. In order to survive organizations should more than ever be prepared to provide rapid responses to an ever increasing rate of changes, in the most efficient way. Reengineering introduces drastic changes that alter not only the flow but also the essence of organizational work itself. If new levels of efficiency are to be reached, it is mandatory that more be known about each of the steps involved in the workflow. This article aims at presenting some contributions on the less studied but nonetheless important issues related to the flow of work in organizations.

This work will first propose a few definitions, based on the distinction of description and execution level of workflow management systems (WFMS), and then we will center our analysis on two components of these two levels: the process modeling and the enactment level, and discuss their relationship specially under exceptional conditions. An important component of the process modeling is the activities that must be carried out during the execution of the process. We will classify activities activities according to a few criteria. The next section will propose some concepts related to exceptions in office procedures. Like activities, we believe that exceptions are a central aspect of office work and they need further study. Finally we discuss how a workflow system can be of help in dealing with exceptions.

[†]This work was published at the Proceedings of the ACM Conference on Organizational Computing Systems (COOCS'95), pages 138-147, August 13-16, 1995, Milpitas, CA

2 DESCRIPTION LEVEL and EXECUTION LEVEL

The first important concept on workflow management systems (WFMS) is the distinction between two distinct levels: description and execution.

The description level deal with describing (or modeling) the important aspects of a workflow system, that is, it describes things that determine what the WFMS will do: which procedures will it implement/monitor, what data will it store, which organization is the WFMS a part of, what user-interface will it present to its enactment-level users, and so on. The description level is made up of some components: the PROCESS DESCRIPTION which describes the procedures that the system will implement/monitor; the DATA DESCRIPTION which describes the data that will be stored/used/checked by the system; the ORGANIZATIONAL DESCRIPTION which describes the people and roles and their relationship in the organization; and the USER-INTERFACE DESCRIPTION which describes the user interface for users performing the activities and possibly the interface with other computational tools.

Within the description level, one can distinguish between the procedure and the organizational aspects of the description, which has a more model-oriented approach, and the data and user-interface aspects of the description, which has a more implementation-oriented approach. A description level user of a WFMS that is interested in designing or re-designing the business procedures of the organization would deal mainly with the organizational and procedure aspects of the description level, whereas users in the process of installing a WFMS would be dealing mainly with the data and user interface aspects of the description level. In this paper we will concentrate on these modeling aspects of the description level, and we will refer to them as the model level.

The runtime level deals with the execution of the WFMS. It is further decomposed into two aspects: ENACTMENT and MANAGEMENT. The enactment aspect deals with the effective execution of the activities specified at process description. That involves keeping track of what activities (or instance of activities, as we will call them later) are being performed which documents, and who is performing that activity, providing support for the execution of those activities, and so on. The enactment-level user of a workflow system is a office worker that fill the forms, approve documents, make decisions and so on. The management aspect of the execution level involves privileged access to audit information about current and past workcases, statistical information about users, processes, activities and so on .

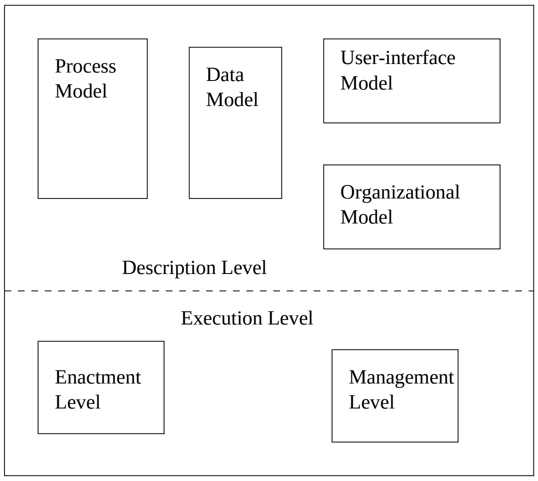

Figure 1: The Description and Execution levels of Workflow Systems

In this paper we will concentrate on elaborating the process description level and the enactment level, and their relations. The workflow literature has been mainly on process description, proposing languages or methodologies to describe the processes [MWFF92, EN93, SR92, Zis77, Joo94]. We will extend a bit that focus by including the enactment component and its relation to the process description.

2.1 Process Description

At the process description level (or process modeling as we will call it) we define a PROCEDURE as a specification on how a particular business process should be performed: which activities should be performed, by whom, and in what order. Thus an organization may have the customer order procedure, which specifies what is supposed to be done to each customer order; or a purchase procedure, which specifies what steps have to be taken if a member of the organization wants some product to be bought from outside sources, and so on.

An ACTIVITY is one of the steps in a procedure. It is a set of operations to be performed by a person filling a particular ROLE onto some objects (or documents), in order to achieve a particular goal. Thus, in the customer order procedure, one can talk about the *checking the customer's credit* activity, whose goal is to check whether the customer has credit to pay for the purchase ordered. This activity is performed by a person in the role of a *credit specialist.* The carrying out of the activity is do checking organizational data-bases for past information on this customer, verifying outside sources, like credit services, for the credit history of the customer, and so on. All this information may be placed in the customer's order documents, or it may be placed in temporary documents, and only the final decision on whether to approve the credit or not is placed in the order documents.

A procedure has two aspects: its activities and the ordering among them. We will call this ordering the FLOW. The flow is interested in specifying what activity will follow another, what can be done in parallel, and what must be sequentialized, and so on.

The focus of the process description literature has been centered mainly in describing the flow of a process: there are many languages to describe the flow, for example Information Control Languages [Ell79], Role Interaction Networks [SR92], Action Technologies flow language [MWFF92, MM92], and many forms of Petri Net based languages.

Another component of the process model is the ACTOR ASSIGNMENT, that is, the specification on how to assign actors to activities. The specification of an activity includes the definition of the role that can carry out that activity. But the real executing of the activity (at the enactment level), has to be carried out by people (or actors) filling that particular role. Thus there must be a specification on how to choose (at the enactment level) an actor to perform that activity. Standard specifications include: select one of the actors that can fill that role at random; distribute the activities to a pool of workers in a round-robin fashion; assigning the activity to the least loaded worker in the pool of workers capable of filling that role, and so on.

The user of the model level of a workflow system is interested in designing office procedures for an organization, and verifying whether the implemented procedures are efficient. Thus, a model-level user is typically a manager, and he is interested in tools that makes the task of designing and redesigning [Ham90] procedures more efficient. For example: graphic language to specify procedures, simulation and animation of procedures [EN93], methodologies that guide the design of procedures [MWFF92], statistical data about existing procedures and activities within those procedures, the ability to dynamically change a procedure even when there are instances of that procedure still being processed [EK93], and so on.

2.2 Enactment level

The model level of a workflow deals with modeling the application, the enactment level deals with "running" it. At the model level one defines a purchase order procedure, and instances of it are carried out at the enactment level. We will call an instance of a procedure as a WORKCASE. Workcases can be active, if they are still being carried out, or may be inactive, if they have been all carried out. Thus the system may have many active and inactive workcases that are instance of the same procedure.

The corresponding concept to activities at the enactment level are INSTANCES OF ACTIVITIES. Unless there is the danger of ambiguity, in this paper we will call instances of activities just as activities. The instance(s) of activity that is(are) being carried out in a workcase at the moment are called the active activity(ies).

There is no simple correspondent to the flow at the enactment level. The flow is the specification of the sequence of activities a workcase must or should go through as a whole. But on the enactment level, this temporal dimension is not represented but lived through. Thus the whole of the flow of a procedure should be divided for each workcase, at the enactment level, into three components: the current instances of activities that are active; the activities instances that have already terminated; and the activities that will or should still be done. The first component is the STAGE of the workcase: what activities instances are being performed on the workcase, and by who. The second component will be called HISTORY of a workcase: which instances of activities it has been through, when those activities started and ended, and who performed them. And the third component, the PRESCRIBED FUTURE STEPS of the workcase.

For a workcase, a deviation from the prescribed flow will be called a REDIRECTION. For example, the flow may specify that after receiving a purchase order, the purchase department will invite bids from suppliers to fill that order, choose the best bid, place the order, and so on. But it could happen that in a workcase the suppliers returns the order because some specification is incorrect. Thus this workcase will return to whoever initiated the purchase order to correct the specifications. There was a redirection of this workcase, which deviated from its normal sequence of activities.

Another example of redirection may not involve repeating some activity but changing the actor assignment of a workcase: a worker calls in sick or leave for vacations and the system reassigns the activities that had been attributed to that worker to other people in the organization. Redirection may also involve delegation of work: a worker asks a colleague to deal with an activity because he must leave earlier today, or he believes that the colleague is more capable of dealing with that case, or for other reasons.

At the enactment level, the system should provide its users (usually an office worker) tools and functionalities that will help that user to carry out the activities. Thus, at the enactment level, one would like the workflow system to have access the organization's data and knowledge bases; to easily integrate other computational tools normally used by the enactment level user in his daily work; to provide help in case something exceptional or out of the ordinary happens; to help a office worker manage his time and tasks; to help a new worker learn his new duties and how to carry them out, and so on.

2.3 Comparison with other works

The separation between a description level and a execution level for workflow systems is present in many works [AK94, EK93, Wor]. But these works take as basic the distinction of modeling and enactment, which we believe is only a component of the basic distinction between description and execution levels.

The idea of other components in the description level, besides the process description is present in the description of two implemented workflow management systems [IBM95, A+94], although their definitions for these other components do not agree totally with ours. Those two documents made it clear to us that the modeling aspects of the description level (that is, process description and organization description) does not contain information enough to describe a working system (but [Wor] seems to take a different view in the sense that data and user-interface descriptions can be seen as concerning only applications that are triggered by a workflow enactment engine). The need for a static (data) description, a dynamic (process) description and a user interface description in order to fully specify a workflow system agrees with the ideas described in [EW94].

The definitions in this paper in general agree with the ones proposed by the Workflow Coalition [Wor]. The most relevant difference is the fact that the Workflow Coalition does not define concepts that would correspond to our definition of flow and its correspondent definitions at the enactment level. The table below lists the correspondences.

Our Definition	Workflow Coalition Term
Procedure	Process
Activity	Process Activity
Role	Organizational Role
Enactement-level User	Workflow Participant
Flow	-
Workcase	Process Instance
Instance of Activity	Process Activity Instance
History	Audit Trail
Stage	-
Prescribed Future	-

Next section will analyze the concept of activities which we believe is a somewhat forgotten aspect of the process modeling.

3 ACTIVITIES

Activities are the locus of processing work, defined at process modeling level and effectively performed at enactment level. We believe that a more specific classification of the types of activities is necessary for many reasons. From a description point of view, a specification of the activity (as oppose for example to treating activities as "generic" black boxes as they are treated in some of the process description languages) may help users to better understand the procedure being described. More importantly, from the WFMS point of view, knowing the type of activity may allow it to provide the actors of that activity with better tools do perform that activity. Finally, we will see that a better understanding of activities themselves will point us to some short comings of current process description languages.

There are many dimensions by which one can classify activities. We will discuss some of these dimensions but we make no claim that the list is exhaustive.

3.1 Batch x Workcase-Based Activities

WORKCASE-BASED activities are activities that deal exclusively with one workcase. The information needed to perform the instance of activity is available in that particular workcase (and possible data-bases and knowledge bases of the organization), and the product of the activity instance is added to that workcase. Thus filling a purchase order is typical workcase-based activity: one need to fill the appropriate fields for that workcase, and maybe consult organizational data-bases to retrieve information about cost and code of items. No access to other workcases was necessary, and the result of the activity is added to just that workcase.

A BATCH activity is an activity that can only be completed if a set of workcases are brought together, and the product of the activity is added to a subset of those workcases. A typical batch activity is ranking, for example ranking candidates for a position. One can only rank the candidates if all the workcases for each applicant are brought together, and are compared to each other. A product of such activity could be to fill a field in each workcase with the rank the candidate received (1st, 2nd and so on).

Another examples of batch activity are some forms of actor scheduling and time scheduling (see below). For example, a phone company may have a scheduling activity (both actor and time) that determines what repair activities will be performed by what repairperson and when will those repairs be made. This assignment must take into consideration that the company wants to optimize the traveling of the repairpersons, that one repairperson may be more knowledgeable than others on different kinds of problems, that a repair order should be fulfilled as soon as possible, and so on. Again in this case, all repair orders (or at least a lot of them) must be brought together in order to perform this activity.

In general batch activities are either ranking activities (in which the workcases in set are ranked according to some preference criteria) or optimization activities (in which some decision about a subset of the workcases is made, and this decision in some way optimizes some cost function).

Batch activities cannot be fully described in flow languages such as ActionFlow [MWFF92, MM92] or ICN [Ell79]. These flow languages are inherently based on workcases, that is, they describe the activities and the ordering of these activities that a workcase must go through. Thus a batch activity, that deals with more than one workcase at the same time, cannot be fully represented: only the "workcase projection" of a batch activity, that is its results in a workcase, can be represented. In the first example above, the ranking activity would be represented in a ICN as a *classify* or *attribute classification* activity, which describes the effect of the ranking in that workcase, but not its details.

3.2 Structured x Non-Structured Activities

Some activities can not be put under direct supervision of the system. That is the case of activities that involve very elaborate creative work. For example, a service order for a construction company may involve a "project" activity which is performed by a group of engineers and architects. This project activity, most likely, will not be under supervision of the workflow system mainly because it is probably unreasonable to decompose it as a subprocedure (with its temporal scheduling of well defined subactivities).

Non-structured activities cannot be decomposed into subactivities, because they usually demand very creative forms of interactions, and one would not like to constrain the activities by prescribing how they should be carried out. But the products of these non-structured activities must be incorporated back into the system, so that the activities that follow the non-structured one can be carried out. In the example above, following activities may include: sending out a bidding proposal, getting the client to approve the project's specifications, and so on. Non-structured activities are usually performed by a group of people, working with tools outside the workflow system.

Structured activities, on the other hand, are better understood by the system and by the people that design the procedure itself. They can be achieved by a single person working "within the workflow system". In fact the standard way of partitioning a procedure into activities is by separating the operations that can be done by a single person alone.

3.3 Internal x External activities

Some activities are carried out "outside" the workflow system. That means that during the execution of these external activities the workflow system neither provides the tools to carry them out, nor is able to monitor the progress being made. The system has to be informed when these activities are done since it has no knowledge about them. Furthermore, the documents that are the result of the activity may have to be entered (maybe manually) back into the system so that they can be incorporated into the workcase.

Example of external activities include, usually, the non-structured activities. But other structured activities may also be external. For example it is possible that the workflow system is used only within subsets of the whole organization (for example, in a few departments) and the activities in a procedure that are performed by the other departments are external to the system. The relevant documents of the workcase must be printed, handled over to the other departments, and the results of those activities may have to be entered back into the system.

3.4 Prototypical activities

One can also classify an activity as how close it matches some prototypical activities. The list of prototypical activities below is not complete, but we believe it represent some major classes

External data gathering Data from external sources is obtained and recorded. The interface for such activities are usually implemented as a form whose fields must be filled either by the client (the outside source) or by a office worker that is in contact with the client.

Data Processing New information is derived from the external data, classifying it according to organization's internal criteria. It may involve batch activities like ranking, or workcase-based activities like determining whether the data provided is appropriate.

Decision A choice between alternatives that involves subjective judgment. The decision would determine which specific path of the flow will be taken. The interface for a decision activity is usually presented as a set of boxes and the decision maker must check the box that correspond to the outcome of the decision.

Authorization Authorization is a weaker form of decision where a person within the organization assumes responsibility for the state of a workcase. In paper-based workflows this is usually done by signing. On an computer based workflow, the interface for a authorizing activity is presented as a two choice boxes: authorize or don't authorize.

Waiting Waiting is a degenerated form of activity in which the workcase is waiting for some trigger to proceed for the next activity. In some cases, the waiting precedes a batch activity: the workcase is waiting for a batch activity to be activated. Wait activity also may occur before the synchronization of parallel branches in the flow: the workcase is waiting for the activity on the other branch to finish before it can proceed to the next step.

Proxy Activities Proxy activities are a kind of wait activity that represents the workflow counterpart of an external activity. The workflow system represents the fact that some external activity over which it has no control or monitoring capabilities by placing the workcase in a proxy activity.

Temporal Scheduling Determines the point in time or at least the ordering in which a set of activities will be performed. For example, a procedure may determine that three activities can be performed in parallel, but at the enactment level, one may have to sequentialize them due to the lack or resources. The temporal scheduling would determine which sequencialization will be followed. Temporal scheduling may also be enacted as a batch activity in which many workcases are scheduled in such a way as to optimize resource utilization, for example.

Actor Assignment Actor assignment is the activity of determining who will perform a particular activity instance. It can be performed by the system using some scheduling algorithm, or it may involve optimization (as a batch activity), or it may involve subjective judgment and complex knowledge. As an example of the later, a consulting company may need a non-structured actor assignment activity to decide who will take care of a case. That decision may involve workload, who has experience with that kind of problem, who has experience with that client, who needs to be exposed to new challenges and so on.

Knowing what type of activity should be performed will allow a workflow system to provide better tools for the actors performing those activities, and specially tolls for the actors to deal with exceptions that may happen during the execution of the activity. For example a common exception for data gathering activities happens when the external source does not have all the information requested. A well designed workflow system should allow for incomplete information to be entered, and allow for the missing information to be entered at a later time.

4 EXCEPTIONS

As a working definition in this paper we will call EXCEPTIONS the departures of the history of a workcase from its prescribed (or normal) flow. The first consequence of this working definition is that exceptions are things that appear at the enactment level, when activity instances are being executed. That does not mean that exceptions cannot be thought about at the model level, and in fact we will propose such approach further on, but one must realize that exceptions are mainly enactment-level phenomena.

We believe that exceptions are important and costly phenomena in an organization. Important because they are common [Ham90, SM89], and costly because they demand a lot of effort to be dealt with.

There are very few works that discuss and classify exceptions [Saa94, SM89, SMS94] and even those describe exceptions from an organizational point of view, whereas we are interested in exceptions from a workflow system point of view: how can a workflow system be impaired by exceptions, or help in dealing with the many kinds of exceptions that will happen? Below, we discuss some dimensions in which to classify exceptions from a workflow point of view.

4.1 Information x Infrastructure Exceptions

The first important distinction on exceptions is whether the source of the exception is the workflow system itself, or the outside reality. There are exceptions that derive from the fact that the WFMS is in some way faulty: a software bug in the system, a problem in the computer network underlying the system, workstations off line, database servers off line, printer off line, and so on. All these forms of exceptions would not appear if the organization were not using a WFMS in the first place. We call them INFRASTRUCTURE exceptions.

The other class of exceptions, that we will call INFORMATION exceptions, has no relation with the workflow system itself. They refer to the reality, that is, there is something different in this workcase that makes it deviate from its prescribed flow.

The designer of a WFMS must prepare the system to deal with infrastructure exceptions, since they will affect the system itself. For example, since portions of the network can be unreachable, a well designed WFMS must allow for operations like: storing filled forms locally until connection to a central database are restored, printing the electronic forms into paper and accepting the data entered into these paper forms at a later stage, and so on.

4.2 Data exceptions x signal exceptions

Data exceptions appear when some of the data present in a workcase is incorrect or missing. For example, the budget classification of a purchase order was wrongly filled, or the shipping address of a customer is missing, and so on. Data exceptions are usually detected in the activity where that data is needed and are usually handled by redoing the activity that generated the incorrect data, or the activity that failed to collect the missing data.

Signal exceptions are exceptions that are bought up by external, asynchronous information that in some way changes the set of conditions about a workcase. For example a customer may cancel an order that is already being processed. This involves canceling all the prescribed following activities, undoing some activities that can be undone, starting activities that are not part of any procedure, like suing the customer for the costs, changing activities and attributions in workcases totally unrelated to this case (for example because of the cancellation, resources that where tied up with this case can be released to other workcases), and so on.

Signal exceptions may cause a lot of changes on the workcases, or they may be easily dealt with, depending on what kind of signal exception is received, and in what stage the workcase is in when the exception was received. For example, a customer realized he needs more goods than the number he actually ordered, and thus he changes his previous order. If production of this order has not been scheduled yet, then accepting the change may only involve redoing the credit checking on the customer to be sure that he can pay for the extra goods. On the other hand, if the production has already started then dealing with the change in the order may involve rescheduling other orders, or postponing the production of the extra goods for a later time, and so on.

4.3 EHAs – Exception handling actions

Exceptions are dealt with with possibly very complex sequence of operations, and ad hoc activities. But in some more standard exceptions (for example the not so crippling infrastructure exceptions, and data exceptions) the exception handling process starts with what we will call as EXCEPTION HANDLING ACTION, abbreviated as EHAs. EHAs are not the whole of the exception handling process, but are simple operations that are provided by the workflow system and that initiates the redirection process that would eventually lead to the resolution of the exception.

An example of EHA is sending a workcase back to whoever was responsible for filling a particular piece of incorrect information. The tt send to responsible EHA redirects the workcase so that a previously done activity must be redone in order to correct a wrong information in the workcase. We see EHAs as run-time functionalities that are provided by the workflow system independent of the procedures that are currently implemented. The EHAs are activated by enactment-level users when they realize that a workcase must deviate from its prescribed flow.

Since EHA are the primary way in which a enactment level user will start a exception handling situation, they should be part of the tools available to enactment-level user. We believe that WFMS that do not implement any EHA are incapable of dealing with exceptions at all.

There are a multitude of EHAs that should be made available in a workflow system. We can list some of the most common EHA.

- Redo instances of activities that have already been executed. This may happen when incomplete or incorrect data is detected at a later activity and must therefore be altered. This include *return to sender*, *return to responsible for data* and *return to customer* EHAs.

- Attach an explanation to a document or components of the document. Sometimes enatment-level users must explain the context in which one form or a field of form was filled to other users down the flow. That transfer of context could explain, for example, the meaning the first users attribute to some of fields in the form, or explain why there is some data exception in this workcase.

- Add or change information in the workcase. This may happen when a enactment-level user detects missing or incorrect data (data exceptions) and correct them himself, without looping back to previous activities in the workcase. This may happen if the office worker realizes the correction of the data can be easily done (a missing shipping address can be discovered by phoning the customer) even though the activity being performed by that office worker may not have the right to change or add that information.
- Skipping over some activities of the flow, as may happen when an special urgent case has to be put through in haste, or when some parts of the organization are unreachable due to an infrastructure exception.
- Reassigning an activity to some other actor. This EHA can be used to reassign the workload of a worker that called in sick, whose workstation is down, or it can be used by a worker that realizes that someone else is more capable to deal with this case and thus asks for help from that person.
- Infrastructure specific EHAs that have been implemented by the designer of the WFMS (if they are not automatic), like *store locally until network is ok*, or *merge local and global databases* and so on.
- Forwarding the workcase to people that have the rights to make broad decisions about them.
- Temporarily or permanently dumping the workcase out of the system. This involve printing all documents in the workcase, and for the system's record, start considering the case as temporarily inactive, permanently inactive or even remove all records about the workcase from the system. This EHA can be used to deal with some more severe infrastructure and signal exceptions.

5 WORKFLOWS AS EXCEPTION HANDLING TOOLS

We believe that it is very interesting to view workflow systems as tools that allow people in the organization to deal with exceptions. Exceptions are not only the rule (in the sense that although a particular exception is very unlikely, the probability of having an exception of any kind may be really high [Ham90, SM89]), but dealing with them is a task for what people in the organization could use some help.

To be able to help the actors dealing with exceptions, a WFMS should be conceived with that in mind, by providing tools, ideas, concepts, methodologies and so on, both at the enactment level and at the model level. We will discuss how to deal with data exceptions at the enactment-level and with signal exceptions at the model level.

5.1 Exception handling support at the enactment level

Exception handling at the enactment level involves two steps: detection of the exception, and handling of the exception [SM89]. For data exceptions, detections means the realization that some piece of data in the workcase is incorrect or missing, usually during the execution of an activity that need the data. A WFMS can help the user to detect data exceptions by, for example, bringing the user's attention to data that significantly deviate from stored examples, historical means, and so on.

Once the exception has been detected, it has to be dealt with. As we mentioned above, dealing with an exception begins with a the user performing an EHAs: obtaining and entering the missing data, or sending the workcase back to whoever was responsible for the wrong data, forwarding the workcase to whoever has the right to authorize the further processing of the workcase in the absence of the missing data, or any other of the many EHA discussed above. Thus providing the enactment-level user with as many EHAs as possible seems to be a good way of empowering them to deal with exceptions.

Another way a WFMS can help the handling of exceptions, is through its knowledge of the activities being performed. Certain types of exceptions are more common to certain types of activities. By knowing this, a WFMS can provide its enactment-level users with fine tuned EHA for particular classes of activities. For example, a common exception in external data gathering activities is incomplete information. A system should allow for incomplete information to be entered, and should also forecast that either a corresponding signal exception will occur (the customer will call to provide the missing information) or a data exception will happen (when the missing information becomes needed).

Data processing activities are a potential source of data exceptions: the workers may not know all the rules for generating new data, the meaning of the rules may change across different departments in the organization [Gal77], and so on. So it is important that a WFMS remembers who produced which information within a data processing activity, since some of the information may be incorrect and a further contact with whoever generated them will probably be necessary to correct them. By keeping track of who did what, the system would allow a worker further down the line to perform a *return to responsible* EHA.

External activities, since they are outside the control of the system, are a source of exceptions both data and signal: the system has no way of checking the consistency of the data generated within the external activity; nor it can forecast problems that will appear. Thus, internal activities that represent these external activities (proxy activities) and internal activities that follow external activities must be carefully conceived to deal with both data exceptions and signal exceptions generated during the execution of the external activity.

5.2 Exception handling at the model level

Exception handling at the model level amounts to planning on how to deal with expected signal exceptions. We believe that there are exceptions that can be expected to happen, and at the model level one can specify how to deal with those expected exceptions. There are signal exceptions that are very common, and as a matter of methodology, the designer of the organization's procedures should include dealing with them when designing the procedures.

Common signal exceptions are: inquire (what is the stage of a workcase); change data (some of the data provided by the initiator of the workcase has changed); urgency (the initiator wants the workcase to be processed with urgency); and cancellation (the initiator wants to cancel the request underlying the workcase).

In order to plan for signal exceptions at the model language, the procedure description language must be expressive enough for the model-level user to specify the effects of the exception at each stage of the process. We believe this is a major deficiency in most current procedure description languages. These languages are only expressive enough to describe the "normal" flow of a procedure, and provides no support to represent external events.

Dealing with signal exceptions at the model level is both a question of providing the necessary language tools so the model-users can express how those exceptions should be handled, and a question of providing the model-level users with a methodology that guides them into thinking about these kinds of exceptions. For example, the Action Technology workflow language [MWFF92, MM92] forces model-level users to think about what has to be done when a task is not accepted as completed by whoever asked for it, since satisfaction is one of the steps the language requires to be specified in describing a procedure or subprocedure. This kind of forcing the model user to think about exceptions should be extended to other kinds of predictable signal exceptions like the ones described above.

5.3 Exception handling when designing a workflow system

Finally there are exceptions should be dealt with by the designer of the WFMS itself: the infrastructure exceptions. The designer of a WFMS must prepare the system to deal with infrastructure exceptions since they are potentially very common ([SMS94] lists technical malfunction, which includes infrastructure, as the most common exception in that survey).

6 CONCLUSIONS and FUTURE DIRECTIONS

This paper tried to proposes some distinctions and definition about concepts related to workflow management systems. It expanded the model/enactment distinction into a description/execution distinction. The description level includes the following components: process, data, organization and user-interface description; and the execution level includes an enactment and a management component. The relation between the process modeling level and the enactment level was described: the two levels are somewhat symmetric but the presence of exceptions may force a workcase to deviate from the flow prescribed in the process description level. Furthermore, exactly because the enactment level user will have to deal with the exceptions, the system must provide her with extra functionalities that was called Exception Handling Activities (EHAs). These EHAs in some way transfer some amount of control of the workcase from the system to the user, and thus a WFMS should provide its enactment level users with a variety of EHAs, in the hope that one particular EHA would be exactly appropriate to deal with an exception.

One of the important components of the process description level is the activities. Process description languages have concentrated in describing mainly the flow of a procedure, and the description of the activities themselves has been less emphasized. This work attempted to describe activities according to many dimensions. A particularly important one are the batch versus workcase-based activities; we are not aware of any procedure description language that fully addresses the problems of having a batch activity in the flow.

Finally this paper also attempted to classify exceptions according to many dimensions. We hope that a better understanding of the nature of the exception and the nature of the activity in which it occurs would allow a designer of a WFMS to provide an EHA that would more appropriately deal with that situation.

There are many aspects in which this work is being extended. On of them is the development of a augmented procedure description language. Some of the flow languages available nowadays are inadequate to define the complexities of office procedures. These languages present three problems: they are unable to represent asynchronous events, they cannot fully represent batch activities, and they do not specify the types of activities present in the flow. The inability to represent asynchronous events (or signals) limits how much exception planning can be done at the description level: if one cannot represent a cancellation signal from the customer, how can one represent what should be done when that signal arrives in different stages of the procedure. We are currently developing a procedure description language combining the concept of triggers [Joo94] and state charts [Har88] that would allow representing the prescribed treatment of some signal exceptions.

Another direction of future research is the elaboration of the relationship of the other components of the description level and the enactment level. Intuition would suggest that data exceptions (in particular missing data) are the manifestation of a broken symmetry between the data model and the enactment level: the data model describes what the data should be, but at the enactment level it may not be possible to satisfy those requirements. Furthermore, that seems to be possible to classify some EHAs as procedure-centered, those that cause a redirection of a workcase like *redo an activity*, or *send workcase to other user*; and data-centered, those EHA that changes data but do not cause redirection like *add data to the workcase* or *attach comment to workcase*.

Finally there is the need to further elaborate the two other components of the description model: the organization model and the interface model.

6.1 Acknowledgments

The second author would like to thank Clarence Ellis and the Coordination Technology Research Group at the University of Colorado at Boulder for discussions on many topics developed in this paper. All errors and omissions are sole responsibility of the authors.

References

[A$^+$94] M. Ader et al. WooRKS, an object oriented workflowsystem for offices. Ithaca technical report, Bull S.A., T.A.O. S.A., Universitá di Milano, and Communication and Management Systems Unit, 1994. ftp://cui.unige.ch/OO-articles/ITHACA/WooRKS.

[AK94] K. R. Abbot and S. K. Karin. Experiences with workflow management: Issues for the next generation. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*. ACM, 1994.

[EK93] C. Ellis and K. Kedara. Dynamic change within workflow systems. Technical report, University of Colorado, Computer Science Dept., 1993.

[Ell79] C. Ellis. Information control nets: A mathematical model of office information flow. In *Proceedings of the 1979 ACM Conference on Simulation Measurement and Modelling of Computer Systems*, pages 225–239, 1979.

[EN93] C. Ellis and G. Nutt. Modelling and analysis of coordination systems. Technical report, CU-CS-639-93, Department of Computer Science, University of Colorado at Boulder, 1993.

[EW94] C. Ellis and J. Wainer. A conceptual model of groupware. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, pages 79–88, 1994.

[Gal77] J. R. Galbraith. *Organization Design*. Addison-Wesley, 1977.

[Ham90] M. Hammer. Reengineering at work: Don't automate, obliterate. *Harvard Business Review*, July, August 1990.

[Har88] D. Harel. On visual formalisms. *Communications of the ACM*, 31:514–530, 1988.

[IBM95] IBM. *IBM FlowMark for OS/2 Version 2*, 1995. http://www.torolab.ibm.com/workgroup/flowm018.html.

[Joo94] S. Joosten. Trigger modelling for workflow analysis. In G. Chroust and A. Benczur, editors, *Proceedings CON '94: Workflow Management, Challenges, Paradigms and Products*, pages 236–247, Oldenbourg, Vienna, 1994.

[MM92] R. Medina-Mora. Action workflow technology and applications for groupware. In D. Coleman, editor, *GroupWare'92*, 1992.

[MWFF92] Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. The action workflow approach to workflow management technology. In *Proceedings of the ACM CSCW'92*, pages 281–288, 1992.

[Saa94] H. Saastamoinen. Exceptions: Three views and a taxonomy. Technical report, Department of Computer Science, University of Colorado at Boulder, 1994.

[SM89] D. M. Strong and S. M. Miller. Exception handling and quality control in office operations. Working Paper Number 89-16, Boston University, School of Management, Boston, MA, 1989.

[SMS94] H. Saastamoinen, M. Markkanen, and V. Savolainen. Survey of exceptions in office information systems. Technical report, CU-CS-712-94, Department of Computer Science, University of Colorado at Boulder, 1994.

[SR92] B. Singh and G. L. Rein. Role interaction nets (rins): A process description formalism. Technical report, MCC Technical Report CT-083-92, Austin, Texas, 1992.

[Wor] Workflow Management Coalition. *Glossary - A Workflow Management Coalition Specification.* http://www.aiai.ed.ac.uk:80/WfMC/glossary.html.

[Zis77] M. D. Zisman. *Representation, Specification and Automation of Office Procedures.* PhD thesis, 1977.

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ – **An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um** SGBD **Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

94-05 **Using Versions in** GIS, *Claudia Bauzer Medeiros and Geneviève Jomier*

94-06 **Times Assíncronos: Uma Nova Técnica para o** Flow Shop Problem, *Hélvio Pereira Peixoto e Pedro Sérgio de Souza*

94-07 **Interfaces Homem-Computador: Uma Primeira Introdução,** *Fábio Nogueira de Lucena e Hans K. E. Liesenberg*

94-08 **Reasoning about another agent through empathy,** *Jacques Wainer*

94-09 **A Prolog morphological analyser for Portuguese,** *Jacques Wainer, Alexandre Farcic*

94-10 **Introdução aos Estadogramas,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

94-11 **Matching Covered Graphs and Subdivisions of K_4 and $\overline{C_6}$,** *Marcelo H. de Carvalho and Cláudio L. Lucchesi*

94-12 **Uma Metodologia de Especificação de Times Assíncronos,** *Hélvio Pereira Peixoto, Pedro Sérgio de Souza*

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

95-10 **A Highly Reconfigurable Neighborhood Image Processor based on Functional Programming,** *Neucimar J. Leite, Marcelo A. de Barros*

95-11 **Processador de Vizinhança para Filtragem Morfológica,** *Ilka Marinho Barros, Roberto de Alencar Lotufo, Neucimar Jerônimo Leite*

95-12 **Modelos Computacionais para Processamento Digital de Imagens em Arquiteturas Paralelas,** *Neucimar Jerônimo Leite*

95-13 **Modelos de Computação Paralela e Projeto de Algoritmos,** *Ronaldo Parente de Menezes e João Carlos Setubal*

95-14 **Vertex Splitting and Tension-Free Layout,** *P. Eades, C. F. X. de Mendonça N.*

95-15 **NP-Hardness Results for Tension-Free Layout,** *C. F. X. de Mendonça N., P. Eades, C. L. Lucchesi, J. Meidanis*

95-16 **Agentes Replicantes e Algoritmos de Eco,** *Marcos J. C. Euzébio*

95-17 **Anais da II Oficina Nacional em Problemas Combinatórios: Teoria, Algoritmos e Aplicações,** *Editores: Marcus Vinicius S. Poggi de Aragão, Cid Carvalho de Souza*

95-18 **Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems,** *Rosiane de Freitas Rodrigues, Pedro Sérgio de Souza*

95-19 **wxWindows: Uma Introdução,** *Carlos Neves Júnior, Tallys Hoover Yunes, Fábio Nogueira de Lucena, Hans Kurt E. Liesenberg*

95-20 **John von Neumann: Suas Contribuições à Computação,** *Tomasz Kowaltowski*

95-21 **A Linear Time Algorithm for Binary Phylogeny using PQ-Trees,** *J. Meidanis and E. G. Munuera*

95-22 **Text Structure Aiming at Machine Translation,** *Horacio Saggion and Ariadne Carvalho*

95-23 **Cálculo de la Estructura de un Texto en un Sistema de Procesamiento de Lenguaje Natural,** *Horacio Saggion and Ariadne Carvalho*

95-24 **ATIFS: Um Ambiente de Testes baseado em Inje,c ao de Falhas por Software,** *Eliane Martins*

95-25 **Multiware Plataform: Some Issues About the Middleware Layer,** *Edmundo Roberto Mauro Madeira*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br