CYTED-RITOS International Workshop on Groupware, Lisbon, Portugal, Sempember 1995.

Workflow Modeling

Paulo Barthelmess and Jacques Wainer
Departamento de Ciencia da Computação
Universidade Estadual de Campinas
13081-970 Campinas - SP
BRAZIL
{paulobar, wainer}@dcc.unicamp.br

Abstract

A discussion of workflow models and process description languages is presented. The relationship between data, function and coordination aspects of the process is discussed, and a claim is made that more than one model view (or representation) is needed in order to grasp the complexity of process modeling.

The basis of a new model is proposed, showing that more expressive models can be built by supporting asynchronous events and batch activities, matched by powerfull run-time support.

1 Introduction

Workflow management systems can be defined as "systems that completely define, manage and execute workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic" [WMC95] in a shared environment.

The expressive power of the model that drives the system's execution will ultimately determine the power of the system as a whole. The absence of rules for dealing with a situation generates an "uncertainty level" that demands extra information collection and processing [Gal77]. Therefore, more expressive models lessen processing effort.

An important part of organizations efforts is devoted to exception handling chores [Saa94, SM89]. If we could express more at model level, less effort would be spent treating those situations as exceptions. In fact, they would stop being exceptions, and become rule.

Most description languages are not able to depict asynchronous events and batch activities [BW95], forcing these events and activities to be treated as exceptions. We will propose a model that adds support to such events and activities. Because both asynchronous events and batch activities are commonplace, we feel that better models can be built by being able to support them.

Even very powerful models are not enough to cope with the dinamics of execution. We claim that in order to be useful, a Workflow system must offer better run-time exception handling support.

We will now discuss workflow modeling issues in general and follow with the specifics of a new model, that aim at building more expressive models.

2 Description languages

Business processes need to be abstracted, so that they can be automated by a workflow management system. This abstraction constitutes a model [EW94], built by analyzing the real world and depicting some of its aspects in a description language. The analysis process is heavily influenced by this language. Its constructs (the basic elements and a grammar to combine them) will shape the model, making it easier to express some of the concepts in a specific way, while barring others.

The semantics of the model will govern its workings, be it at simulation or enactment level. That is to say that the system will have an understanding only of those things that can be represented, making it hard to deal with concepts left off the model. The model by itself is not able to specify all the details required at enactment time. An additional semantic, that we will call RUN-TIME SUPPORT, also governs the model execution, providing additional tools and functionality without which the system does not work.

Workflow system models could therefore be analyzed within the following dimensions:

- Modeling
- Representation
- Semantics
- Run-time support

In the following sections, we'll analyze each of these items in turn.

2.1 Modeling

Robinson warns us that organizational modeling is no easy task. Organizations are normally comprised of many different semantic communities, each with its own specialized language and world views [RB91].

One must keep in mind that models are but maps and recipes, that in a similar way do not represent faithfully any reality, but can nevertheless be of help ("procedures are more like advice than algorithms" [Rob93]).

2.1.1 What needs to be modeled

Models produced by current system analysis normally comprise data and function models.Workflow system models must add to that, including a new aspect, that of coordination. Coordination does not have an objective of its own. It is a prerequisite for the organization to reach it's real objectives. Coordination establishes relationships between activities and their products, "binding tasks into larger, meaningful wholes" [Hol88]. This definition conforms with that of Bannon and Schmidt [BS91], that says that "Cooperative work is constituted by work processes that are related as to content, that is, processes pertaining to the production of a particular product or service". Coordination representation brings to light many new problems, as it corresponds to much more fluid, less structured concepts and practices.

Modeling has a long history, at least as long as that of computers. Many analysis methodologies exist (e.g. structured analysis [Yourdon], OOA [SM92]) and are used to try to capture reality aspects. Workflow modeling adds to this traditional process some new ingredients: roles and synchronism.

Roles In order to be able to distribute tasks to their performers, a workflow system must have organizational roles knowledge. This knowledge is not normally necessary when one is modeling using some traditional analysis method.

A system built according to traditional methods normally takes for granted that its operators will be trained to know what, when and how to access each of the system functions. In this case, the burden of determining which task pertains to which actor normally lay with the users.

Workflow systems, on the contrary, are built to deal specifically with the distribution of work to the right persons, automatically or with the help of a human agent. In order to be able to distribute work, the system must have some knowledge of the organizational structure, hierarchical (line units), functional (project and committees), roles and actors. This organizational model will be used at enactment time to assign actors

to roles and for information and activity access control. Using this model, one can describe roles relatively to other roles, as for instance "the secretary of the manager", "the manager of the initiator", "the project leader of the project", "the members of the committee" as done in the WooRKS system [ALP+94].

Synchronism Another issue that is in most systems left to be dealt with by the users is the synchronization of activities. When using traditional systems, users must find some way of coordinating their efforts, perhaps by establishing an additional (manual) protocol. This extra protocol will then be used to warn other participants that some activity must be executed.

Synchronism establishes activity dependencies, and specifies which tasks can be executed in parallel and which must necessarily be postponed until some others are completed.

Traditional system analysis is normally not concerned with synchronism, as attested by the lack of ways of specifying it using ordinary diagrams (DFDs, for instance). In workflow systems, synchronism is usually the central focus, and the necessary protocol is automatically supported by the system. In a distributed shared environment such as offices, coordination of actions of the many players is essential if business objectives are to be reached.

2.1.2 Model scope

Workflow systems scope is normally greater than that of traditional systems, as they must (to a greater or lesser degree) support not only structured automatable activities, but also unstructured and manual activities, those that lack an algorithmic solution.

The focus of workflow systems are the humans responsible for activities, that will eventually make use of tools (automated or not) in order to expedite their work. These tools could be spreadsheets, text editors, database managers, custom made programs, or forms (electronic or not), that will be used to modify information stored in possibly many different documents that carry data concerning a case.

One must realize that workflow systems will normally have to deal with all elements of traditional systems plus the coordination aspects. Data modeling, function and event analysis and other activities involved in traditional analysis will continue to be relevant.

Workflow systems are extensions of traditional systems. The work of modeling such a system will always be harder than that of building a traditional system, as the coordination aspects have to be added to the description of the work itself. Events appear to be the binders of the two views, as they abstract real world incidents "that tell us that something is moving to a new state" [SM92]. These movement indications are important both for data transformation functions (the work itself) as for the coordination of work.

An analysis of existing workflow systems shows that most of them do not try to include data representation in their models, not because data is not important in workflow systems, but because the emphasis is on the coordination aspects.

We feel that each of the different aspects (data, functions, coordination) should be modeled using different kinds of representations, that when taken together would give a picture of the process as a whole. Traditional system analysis often produces many and simultaneous forms of representation (for example diagrams and data dictionaries). It is therefore unlikely that workflow systems could be depicted with just one (simple) diagram.

2.1.3 Meta-models

Each workflow system is based upon a meta-model, be it explicit or implicit [EW94, EN93]. This meta-model is a reflection of the system creators views of how specific models should be built. The meta-model offers a methodological backbone, upon which specific models are built. Description languages reflect the meta-model, offering constructions that make it simple to create models that follow meta-model guidelines.

Some systems have an explicit meta-model and are based in some well grounded theory, as for instance those based in a "language/action perspective" [Win86], based in the Speech/Act theory [Sea69], like The Coordinator [Win88] and ActionWorkflow [Med92]. Other systems do not have an explicit model, but have an implicit one nevertheless, as their functionality is based in assumptions on how user actions should be conducted [EW94].

Models that over specify, determining in a too rigorous way the temporal sequencing of activities tend to become to rigid, being incompatible with the form work is really done [Such87].

2.2 Representation

Visual languages are employed by most workflow systems for process description, probably because they offer a way to shield users from details of the formalism it is based upon. We'll assume from now on that the descriptions will always be visual.

2.2.1 Basic requirements

Look Schlaer and Mellor offer us a list of appearance requirements for a description language [SM92]:

- Hand-drawable / No delicate placements needed - one should be able to build models with nothing more than pencil and paper. This is essential as many tentative versions must be developed before all the details are in place. The sketches are most of the time produced in the field, away from CASE tools and the like.
- No needless differences (from existing languages) - if an appropriate language already exists, a new one ought not to be designed.
- Multiple Views / Density of information - Diagrams should be worth producing, but not unreadable.

We share the notion that multiple views are necessary to depict such a complex reality as that of organizational processes. If we only have a single model, either it will be too complex or else important details will be missing.

In the present paper we are interested in discussing only workflow specific aspects, i.e., the synchronization model. One must be forewarned though, that other complementing diagrams would have eventually to be produced, encompassing data and function modeling.

Feel Some more subtle requirements for process description languages (PDL) are established by Rein in [Rei92]:

- formal in its syntax and semantics,
- based on intuitive concepts,
- visual,
- enactable (executable),
- able to express a variety of processes that can be any combination of formal to informal and automatic to manual processes,
- able to express both concurrent and sequential temporal relations,
- able to support dynamic iteration of process steps,
- able to support dynamic change of the process descriptions.

2.3 Semantics

Petri-nets [Rei82] seem to be the basic formalism upon which variations are introduced to express model semantics. Many models describe their semantics by showing standard conversions between their representation and that of Petri-nets.

Petri-nets offer the basic means to express the essential synchronization, that specific aspect that take workflow systems apart from other systems.

What most of the representation languages do is therefore to offer meta-level constructions over Petri-nets or one of its variations. One can allegedly express more with less symbols, once that each basic symbol corresponds to a sub-net.

2.4 Run-time support

By itself, representations are not able to express all the necessary details for the use of a system. Extra functionality must be added in order for the system to be useful, due to workflow dynamic behavior. We will now examine some of these functionalities.

Exception handling actions At enactment time, many special situations arise, for which no prescribed processing exist. We call these special situations EXCEPTIONS.

Exceptions are much more common than the (ill-chosen) name implies [EN93, Saa94, SM89]. Efficient exception handling is therefore of the utmost importance.

According to Strong and Miller [SM89], two main kinds of exceptions exist: 1) information exceptions and 2) process exceptions. Information exceptions occur when part of the information is incorrect or missing. In some cases, information is altered after it has been used (to make a decision, for instance). Process exceptions occur when special operations are required to produce an acceptable outcome. This may happen for instance when an urgent request has to be put through in haste. Process exceptions may also be caused by the lack of usually available resources, a broken computer system, or an assigned actor on vacations, for instance.

The necessary steps to bring the exceptional case to an acceptable outcome will be called EXCEPTION HANDLING ACTIONS (ehas). Exception handling normally involves a redirection. These redirections could aim at one of many objectives:

- Looping back to 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.
- Skipping over some activities as may happen when an special urgent case has to be put through in haste.
- Rerouting to include activities not present in the prescribed flow. One could add such activities to the ones in the flow or those in the flow could be abandoned in favor of new ones, that are more apt at dealing with the exceptional case.

The execution of non conventional actions will imply in many occasions in referring the workcase upward in the hierarchy for further analysis. This is not a redirection in the strict sense of the term, but implies in changes on the responsibility level. Hierarchies function in this case as additional conflict resolution tools [Gal77].

Actor scheduling At enactment time, roles will have to be filled with real actors, that will perform the activities. This scheduling may be done automatically using some algorithm (round-robin, work load distribution) or manually by an actor, when subjective judgment must be applied. Some of the chosen actors may not be available at some moment, in which case an enactment time reallocation may need to take place.

The acceptance of a task by an assigned actor may involve negotiation between this actor and the requester. This negotiation may be conducted to iron out task details and is made before the acceptance or the refusal of the task. The system must provide functionality to assist in this negotiation process, as done in the Regatta system [SMM+94], that automatically support this kind of negotiation prior to the start of any stage.

Time related issues Organizations work with deadlines. Deadline tracking must therefore be offered by the system: to warn users of approaching deadlines, to warn supervisors of overdue work, and so on.

Access control Security issues surface in most systems, and workflow systems are no exception. It is not desirable that some role/actors might have access to information or could execute actions beyond their authority level.

Systems must offer ways of assigning people authority levels and of enforcing that no one may access unauthorized items. The seeming simplicity of authorization might offer some traps, for instance when there is an inconsistency between authority levels for task execution and for document access [AS94]. The agent may have authority enough to execute the task, but not to access the necessary documents. A difference between the responsible actor and the executor surfaces here [Joo94].

Tool integration Activities work is best conducted with the aid of tools. These tools may be unstructured, like spreadsheets and word processors or highly structured, like forms. Workflow related data is usually presented by means of forms. Database access is done with the aid of forms, that mimic their paper counterparts. The tools, as well as database structure are determined by specific analysis, complementary to the coordination analysis.

A system is greatly enhanced by a flexible integration of tools, as done by the WooRKS system [ALP+94] that offers external application integration through its operator and information models.

Unstructured communication Communication is not always electronic and intra-organizational. Phone, fax, mail and other media is extensively used, both inside and outside of organizations borders. The system must offer a way of recording the meaningful contents of these communications. Even the conduction of electronic interchange could benefit from the existence of a tool (maybe based on speech/act concepts) that would help add semantic to these interchanges.

Models must be in constant evolution, in order to cope with the changes in the real world [EK93]. Once older case histories must be kept, some versioning means must be supported, as process, document and even activity descriptions will probably be continuously changed. An older case may make no meaning under new rules.

3 Model proposal

Existing description languages normally support only the specification of the organizational main line, i.e., of "office procedures for the most predictable normal cases" [Saa94]. We aim at creating a more powerful description, able to express more than strictly the main line. This goal can be reached by better integration of exception handling.

We would like to state clearly that our intention is not of creating a model that tries to predict every possible exception beforehand. Such a goal would be destined to failure, as some authors forewarn ([EW94, AS94], for instance). Our objective is to enhance the model so that the handling of routine exceptions could be incorporated at specification level, and supported at enactment level.

We discuss next how we intend to reach the fore mentioned goals, through better exception handling and component reuse.

3.1 Modeling

3.1.1 Definitions

We'll base our model proposal on the definitions of its conceptual components. The definitions follow when possible those proposed by the Workflow Management Coalition [WMC95].

A PROCESS DEFINITION models the solution for a business objective. It is comprised of plans, objects, and activities. A PLAN specifies states and the transitions between them, establishing an ordering. It models exclusively coordination related aspects. In a way, the plan specifies a possible life cycle [SM92] of a workflow instance. A workflow may have one or more plans, corresponding to the main line and important variations.

The OBJECTS (application data) encapsulate the data needed during workflow execution. ACTIVITIES specify object transformation functions, the work itself. Activities are comprised of WORK-ITEMS, representation of work to be processed. Work-items are atomic (indivisible) and present trivial sequencing requirements. Activities can be either structured or unstructured, manual or automatic.

Each activity has an associated role, that at execution time will be filled by an actor that will actually execute the work-items. A ROLE is a synergistic collection of defined attributes, qualifications and/or skills that can be assumed and performed by an actor, for the purpose of achieving organizational objectives. An ACTOR (workflow participant) can be a human, a program or an equipment (e.g. a printer). Activities are atomic in the sense that they are performed by a sole actor and that the work items have trivial sequencing requirements, therefore involving no coordination aspects. When more than one role is involved, or some special sequencing must be guaranteed, a sub-plan must be specified.

SUB-PLANS correspond to a hierarchical decomposition of a plan, and like a plan, specify transitions between states. Each STATE indicates a stage in the life cycle of a workflow instance. Each state can be

associated with an activity or a sub-plan, to be activated at execution time (like a procedure call) when/if the state is ever reached. States are reached through TRANSITIONS, triggered by the occurrence of events. EVENTS are abstractions of real world incidents that signal state changes [SM92]. Events can be generated outside the system (for instance the arrival of ordered goods) or internally, generated inside one of the executing activities (for instance when the ending of one activity triggers the start of the next).

WORKCASES (workflow process instance) are the instances of a process description created at enactment time. Workcases are initially comprised of instances of the prescribed objects and follow the prescribed plan, executing the prescribed activities.

During execution, exceptional situations may cause changes in either the plan, the types of required objects or the activities. These changes will reshape the workcase to conform it to the exceptional situation. The prescribed plan, objects and activities can, and probably will, be changed dynamically, during the lifetime of the workcase. The need to deviate from the prescribed plan, objects and activities can arise many times during workcase's lifetime. As the workcase execution proceeds, a HISTORY is formed, describing the steps taken so far.

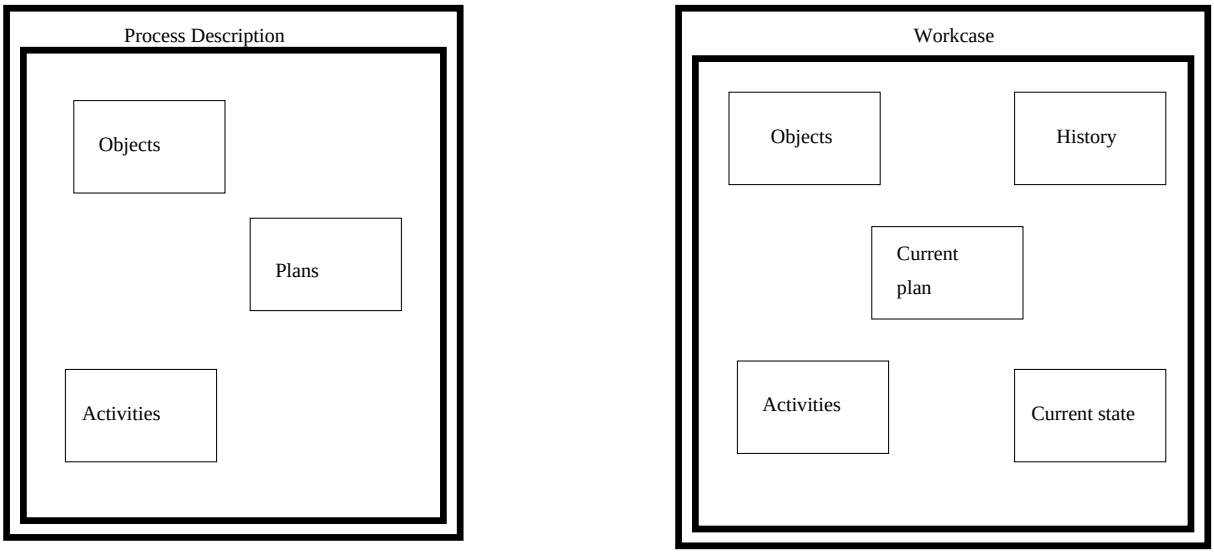

Figure 1: Process description x Workcase components

Workflow systems pose the very interesting problem of dynamically changing requirements. For some processes, analysis will only reveal its goals and the indication of how they are most of the time reached. Exceptions will cause changes in the requirements, bringing sometimes the need for a run-time re-analysis and re-planning. Workflow systems must offer its users tools that would only normally be used by analysts in traditional systems.

In our model, each of the workflow description components (plans, objects, activities) is addressed separately and will eventually lead to different separate diagrams and/or descriptions. We consider that each representation should focus on the specific aspect it is trying to portray. A single diagram will most certainly not be able to contain all the relevant information: either it will be too complex or it will not address all the relevant issues.

3.1.2 Asynchronous events modeling

Workflow systems are reactive systems, i.e., systems that "are event driven, continuously having to react to external and internal stimuli" [Har88]. Some of these events that must be dealt with are asynchronous, i.e., one can not anticipate the exact moment of their occurrence. As will later be discussed, depending on the moment of occurrence of an event, a radically different response must have to be generated.

Most description languages are not able to depict asynchronous events. This will force the events to be treated as exceptions, even if they are commonplace. In this case, the adequate response will depend on user's knowledge in dealing with the situation. More knowledgeable users will probably deal with it successfully, but that may not be true for less trained ones. We intend to fill this gap, supporting asynchronous events both at model and enactment level.

We envision a system that would let us specify asynchronous events handlers in a hierarchical way, i.e., that generic system level handlers could be specified only once, and that more specific handlers could be specified at the levels where a special response is required.

Events would therefore receive a polymorphic treatment. They would be treated either at activity, process or system level, depending on the existence of a specific handler or not.

Even with more expressive models, a great deal of unanticipated exceptions will continue to have to be dealt with at execution time. We will discuss its handling later on.

3.2 Representation

We will examine in this section not a specific representation, but some representation problems we would like to solve. The pictures presented are not to be considered as proposed forms of representation. They are just illustrative.

3.2.1 Asynchronous events representation

The arrival of a signal connected to an asynchronous event, a cancel request, for instance, can impact a workcase in very different ways, depending on the moment of its arrival.

A possible order processing life cycle is depicted in figure 2. The actual responses would naturally depend on organization policies.

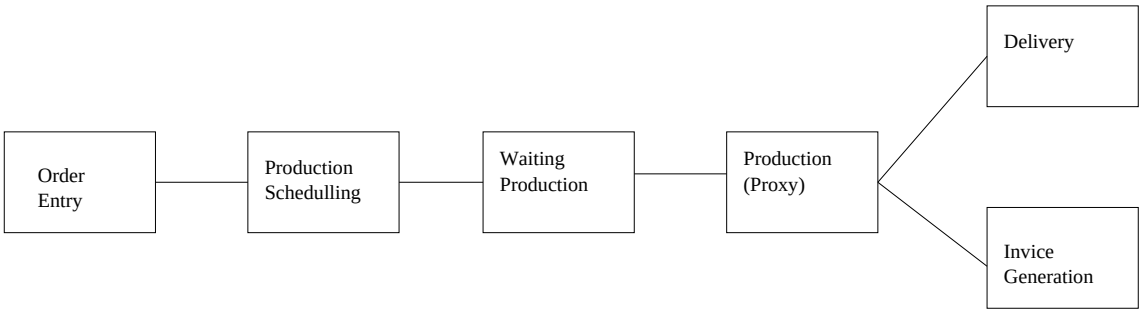

Figure 2: Order Processing

It could easily be seen that different courses of action are taken, depending on the stage the workcase is in at the moment of signal's arrival.

Let's examine some of the representation problems exhibited by the example:

- A fair amount of handling is involved. Its direct integration in a diagram may cause the diagram to become too complex, specially if many different events are being dealt with.
- The same response applies to more than one state. For instance Production scheduling and Waiting for production both elicit the same reactions.
- On occasion, some complex action arises, for instance "Take legal action ". This will most probably involve many expert decisions and on the whole this action can span months or years. For all effects, this action corresponds to a completely new workcase.

State	Response at signals arrival
Order entry	Abort
Production scheduling Waiting for production	Undo any scheduling actions Redo scheduling to fill eventual gaps Abort
Production	Decide if production would be stopped or continued If decided to continue - store the goods Take legal action Abort
Delivery Invoice generation	Store the goods Take legal action Abort

3.2.2 Batch activities

Batch activities [BW95] also pose some special representation problems. In this kind of activities, many individual workcases are brought together and suffer a collective action. This collective action usually involves grouping or ordering the workcases according to some criteria that take all workcases in consideration. The outcome of this action can be in one or more different dimensions:

- Establishing an execution precedence - when some kind of priority is established. Some workcases will suffer further processing before any others.
- Establishing an outcome - might happen for instance when positions are filled. The workcases that graded best will be approved and others will not. Different further processing will be applied in each case.
- Actor scheduling - workcases might be grouped according to some needed expertise, or to some geographic criteria. The groups will then be assigned to an actor that holds the needed expertise, or that services the specified locations.

Individual workcases have to be put in wait state till the batch's starting condition is met. This condition also varies:

- A point in time is reached - the workcases will wait till a deadline is met. This deadline can be set once (e.g., 3rd. of January, 2019), or be repetitive (e.g., every friday, or every morning at 8 o'clock).
- A quota is filled - when a preset number of workcases are waiting (e.g., process batches of 10 or more)
- By manual intervention - the workcases wait till an actor activates the batch.

3.3 Run-time support

3.3.1 Exception handling actions

Even more important than being able to represent exception handling at the model level is the ability to deal with them at execution time. This is important because many exceptions are unpredictable. We envision a system that would offer standard responses to be applied when dealing with exceptions. These exception handling actions (ehas) would range from the most specific (and restricted) to the most general (and powerful).

We will now tentatively group some possible ehas in some different dimensions.

Responsibility related actions All the actions in this group cause the workcase to be sent to some other actor, that will either do some work or forward it once again.

- Back to sender - return the workcase to its sender. This eha can be used to establish a conversation between a requester and a prospective executor of an activity, so that they may iron out some details.
- Back to the information provider - sends the workcase to the actor that provided some piece of information. It might be used to request data correction.
- Forward to responsible - sends the workcase to whoever is responsible for it. It will normally cause the workcase to be referred upward in the hierarchy. This may happen when an actor does not have enough authority to handle a case.
- Forward to - sends the workcase to some other actor (supposedly more knowledgeable), to be taken care of.

Activity related actions The actions in this group command that one or more activities be re enacted or that their work be undone.

- Redo state related activity - commands that the activity related to a state be re done. The state has to be specified because a process might employ a standard activity in more than once, in different states.
- Loop back to state - causes the workcase to be sent back to a previous state, to be re done from there on.
- Undo state related activity - commands that the actions applied in an activity be undone. This action should be automatic, i.e. the system should keep track of the changes, so that they could be undone. Can be used to undo the effects of an activity that should not have been done in the first place.
- Rollback to state - commands that all the work done from the mentioned state on be undone. Again, this action should be automatic.

Plan related actions These actions cause changes in the plan. They prescribe the future steps of the workcase.

- Skip state activity - cause the mentioned state to immediately trigger the start of the next state, without executing its own related activity. If the activity has already been started, it should be interrupted. This action may be used to expedite the processing of a workcase, when an activity is taking to long and therefore compromising a deadline.
- Perform sub-plan - performs an exception handling sub-plan, continuing processing from the current state on as soon as the sub-plan finishes.
- Plan change - determines that the workcase should flow from now on based on a new plan. This new plan can be brought from a library or be designed from scratch. Partial or incomplete plans should be allowed [AS94, SMM+94], as future steps may not yet be clear.

Infrastructure related actions A system administrator must have ways for dealing with communication, server or software failures. Actions of this kind may for instance include:

- Store locally - in the event of communication or remote server failures, store the data locally, till the resources are restored.
- Dump out of the system - orders the transfer of data to other media (diskette, paper) so that it can be further processed by means other than the system's.

3.3.2 Component reuse

Plan construction would be much faster if one could build a new plan deriving it from already existing ones [MCL+92], specializing, modifying or doing composition. This plan library could then function as a common artifact [Rob93], serving as repository of common evolutive solutions, built and refined by the many involved in reaching organization business goals.

Once these components reach a widespread use, plans could be developed in a higher abstraction level, using the components as basic building blocks. This basic reusable components will then function as a meta-language. Each organization would then have a way of expressing their plans using its own tailor made constructs.

Systems are often used in ways its creators had not devised [EW94, Rob93]. Component reuse would in a way allow the system to be extended by its users, that would effectively build in their own specific (meta) language.

3.4 Differences to other trigger models

The main difference from our model and other trigger based models (e.g. Joosten's Trigger Model [Joo94], Regatta [SMM+94] and InConcert [AS94]) is that we are concerned with asynchronous event handling, both at the specification and execution levels.

Our plans show state transitions, and not activity transitions as most of the other models do. This allow us to depict waiting states and proxy (representing external events) in a uniform way, even though these states do not have any associated activity, i.e., they are pure synchronization devices.

Joosten provides in [Joo94] a very interesting example (of a batch activity) that shows no connection between following activities (fig. 3, p. 5). The "mail order" activity does not trigger the next one, "empty letter box". The latter is a time triggered batch activity. This leaves mail order hanging alone, with no explicit connection with the next activity.

We feel that by using states, the flow connections are made clearer, as shown in figure 3. By using states, one can more truly depict a workcase state, showing that it is "waiting for collection ", and not that it is hanging between "mail an order " (performed by a client) and "empty letter box (performed by a mail collector) in an indetermined state.

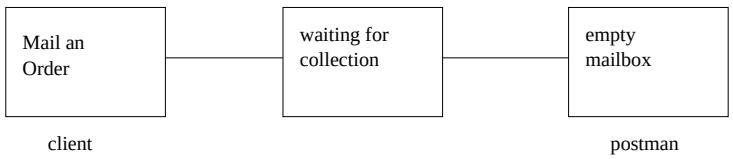

Figure 3: Waiting State in trigger model

4 Concluding Remarks and Future Work

This paper represents an intermediary step in our research on workflow modeling and the development of better suited representational languages and meta-models of workflows. Additional issues need to be investigated:

- A graphical representation must be developed. This representation must be able to express asynchronous events in an easy way.
- A methodological framework must be specified, encompassing all process aspects: data, functions and coordination. For each aspect, a representation must be devised. The methodology has to conductive of process analysis and model construction.

- Plan, object and activity reuse tools have to be devised. The browsing of existing components and construction of new derived ones must be made in an easy, flexible way.
- The executor/responsible relationships must be better studied.

References

[ALP+94] Ader,M., Lu, G., Pons, P., Monguio, J., Lopez, L., De Michelis, G., Grasso, M.A., Vlondakis, G. WooRKS, an Object Oriented Workflow System for Offices, Working paper.

[AS94] Abbot,K.R., Sarin, S.K. "Experiences with Workflow management: Issues for the Next Generation, " in CSCW'94, ACM, 1994.

[BB90] Bullen, C, Bennet, J. "Groupware in practice: An interpretation of work experience, " MIT Center for Information Systems Research, Cambridge, MA, 1990.

[BC88] Bowers,J., Churcher, J. "Local and global structuring of computer mediated communication: developing linguistic perspectives on CSCW in Cosmos, " from CSCW'88, ACM, 1988.

[BW95] Barthelmess,P., Wainer, J. "Workflow Systems: a few definitions and a few suggestions, " to be published ACM Conference on Organizational Computing Systems (COOCS'95), San Jose, CA, 1995.

[Cle90] Clement, A. "Computer Support for Computer Work: A Social Perspective on the Empowering of End Users, " in CSCW'90 Proceedings, ACM, New York, 1990.

[EK93] Ellis, C.A., Kedara, K. "Dynamic Change within Workflow Systems, " Technical Report, Department of Computer Science, University of Colorado at Boulder, 1993.

[EN93] Ellis, C.A., Nutt, G.J. "Modelling and Analysis of Coordination Systems, " Technical Report CU-CS-639-93, Computer Science Dept., University of Colorado at Boulder, 1993.

[EW94] Ellis, C., Wainer, J. "Goal Based Models of Collaboration, " Collaborative Computing Journal, Vol. 1, No. 1. June 1994.

[Gal77] Galbraith, J. R. "Organization Design, " Addison-Wesley, Reading, MA, 1977.

[Har88] Harel, D. "On Visual Formalisms, " Communications of the ACM, 31 (1988), pp 514-530

[Joo94] Joosten, S. "Trigger Modelling for Workflow Analysis, " in: G. Chroust, A. Benczur, Proceedings CON '94: Workflow Management, Chalanges, Paradigms and Products, Oldenbourg, Vienna, Munich, pp. 236-247, ISBN 3-7029-039706, October 1994.

[MCL+92] Malone, T., Crowston, K., Lee, J., Pentland, B. "Tools for inventing organizations: Toward a handbook of organizational processes," in Proc. 2nd IEEE Workshop on Enabling Tchnologies: Infrastructure for collaborative Enterprises (

[Med92] Medina-Mora, R. "ActionWorkFlow Technology and Applications for Groupware," in GroupWare '92, D.D. Coleman, eds, Morgan Kaufmann Publishers, San Mateo, California, 165-167.

[Rei82] Reisig, W. "Petri-nets," Springer-Verlag, Berlin Heidelberg New York, 1982.

[Rei92] Rein, G.L., "Organization Design Viewed as a Group Process Using Coordination Technology," Dissertação de Ph.D., University of Texas at Austin, Austin, 1992.

[RB91] Robinson, M., Bannon, L., "Questioning representations," in Proceedings of the Second European Conference on Computer-Supported Cooperative Work ESCW'91, sept. 1991, Amsterdam, Netherlands, Bannon, L., Robinson, M. & Schmidt, K. (Editors).

[Rob93] Robinson, M. "Design for Unanticipated Use...," Proceedings of the Third European Conference on Computer-Supported Cooperative Work, sept. 1993, G. de Mi chelis, C. Simkone and K. Schmidt (Editors).

[Sea69] Searle, J.L, "Speech Acts," Cambridge University Press, Cambridge, 1969.

[SM89] Strong, D. M., Miller, S. M. "Exception Handling and Quality Control in Office Operations," Working Paper Number 89-16, Boston University, School of Mana gement, Boston, MA, 1989.

[SM92] Schlaer, S., Mellor, S.J. "Object Lifecycles: Modeling the World in States," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[SMM+94] Swenson, K.D., Maxwell, R.J., Matsumoto, T., Saghari, B., Irwin, K. "A business process environment supporting collaborative planning," in CSCW'94, ACM, 1994.

[Suc93] Suchman,L. "Do categories have politics? The language/action perspective reconsidered," Proceedings of the Third European Conference on Computer-Supported Cooperative Work, sept. 1993, G. de Michelis, C. Simkone and K. Schmidt (Editors)

[WMC95] Workflow Management Coalition, "Glossary - A Workflow Management Coalition Specification", http://www.aiai.ed.ac.uk:80/WfMC/glossary.html, 1995.

[Win86] Winograd, T., "A language/action perspective on the design of cooperative work," in CSCW'86 Proceedings, ACM, 1986.

[Win88] Winograd, T. "Where the Action Is," Byte, pp. 256A-258, December 1988.