Exception Handling in Petri-Net-based Workflow Management[1]

Gert Faustmann, Dietmar Wikarski
Fraunhofer-Institut Software- und Systemtechnik ISST
Kurstr. 33, D - 10117 Berlin, Germany
Tel./Fax.: ++49-30-20224-850/-847/-799
e-mail: [Gert.Faustmann|Dietmar.Wikarski]@isst.fhg.de

Abstract

Various classes of Petri nets have been established as conceptual models for the modeling and control of workflows. Usually, these nets are enacted by administrators and there is no way to change the plan once enacted by the start of a computer-based interpretation of the net. In particular, necessary repetitions of wrong previous actions, changes of the business rules requiring changes of the process structure and changes of the organizational structure cannot be realized without restart of the whole net. In the context of a general definition of exceptions and using an example ("Planning permission procedure" in Berlin) it is shown how an appropriate combination of the concepts of *general task embedding*, *commitment and causality trees* and *modular process net*s has been used to implement a very flexible and adaptable workflow management concept.

Keywords

adaptability, exception handling, modular process nets, Petri nets, workflow management

1 Introduction

In recent years, various classes of Petri nets have been established as conceptual models for the modeling and control of workflows (cf. [Deiters 92] and [Reinwald 93]). As a rule, these nets are enacted by administrators and the activities of an enacted net are set as tasks to persons who have appropriate roles. Though Petri nets applied in this way ensure correctness and stability of the predefined workflows, they reveal several drawbacks concerning adaptation to unexpected situations (exceptions) including new organizational and technical infrastructures.
Given a classification of exceptions and their handling, exceptions could also be dealt with using appropriately defined Petri nets. In this case, each possible exception would have to be predicted and appropriately modelled as a partial net which is started from a certain place by branching (the net discussed in Section 2 can be considered as a typical example). As a result of such a procedure, the Petri nets constructed for real applications will be very complex and difficult to handle due to the increased overheads generated by the exception description parts.

1. *Published in: "Practical Aspects of Knowledge Management", Proceedings of the First International Conference, October 30-31, 1996, Basel, Switzerland*

In our concept, the treatment of exceptional cases can be predefined by nets, too, but as one consequence of our approach it will be possible to make these nets and the nets describing the "normal" behaviour considerably smaller and less complex.

For our general definition of exceptions, we need the notion of a *fixed intention* which includes both *commitments* (between persons of a common organizational context) and *plans* (which assign *tasks* to persons involved in a common workflow[1]). The essence of a "fixed intention", i.e. of both plans and commitments, is that at some point in time "in the past" (seen from present) a future procedure has been fixed in oral, written or computerized form which prescribes rules for the future cooperation between the persons involved in a set of tasks. We define an *exception* (with respect to a fixed intention) as the occurrence of any event which requires a deviation from the fixed intention.

In the context of workflow management, typical examples for exceptions are the detection of wrong previous actions (which may require their repetition), changes in the business rules (which may require changes in the structure of the business process) and changes in the organizational structure (which may lead to new responsible persons for given tasks).

Given the basic concept of *task*, commitments and plans differ in two respects - the time scale they cover and the strength of obligation to perform a given task. Commitments are *incremental* and *strong*, i.e. every single commitment concerns one task which the responsible person is committed to perform. If - due to an exception - the task cannot or should not be performed, an explicit communication (speech act, dialogue) between the involved persons is necessary which leads to the cancellation of the task. One can say that the set of (committed) tasks forms the current *strong working context*. In contrast to commitments, plans are *long-term* and *weak* with respect to the obligation to perform all their constituent tasks. In particular, there is usually a set of tasks on which work has not yet started. The cancellation of these tasks does not lead to any consequences for the persons who will be set these tasks.

The simple but powerful key point of our concept is that relatively small, fixed plans - causal structures of the planned activities called *process modules* - are combined with each other in such a manner that the resulting overall process in each particular workflow is the result of the combined use of predefined process modules and of situation-dependent actions made by the responsible persons. The general design and implementation frame used to achieve this is the concept of *task.* Tasks provide a unified interface both for interaction among people (undertake commitments) and between the workflow system and people (set and get tasks).

Given the framework of tasks, two other basic concepts ensure the memory about incremental (committed) and planned (intended) future - namely *commitment and causality trees* on one hand and *modular process nets* on the other.

If we take a look at the relations between workers in an office environment, we can see a network of commitments between them. These commitments are initiated by the current work and can be structured as *(commitment) trees* by filtering out those commitments that belong to the same workflow. Later on, this view will be extended by also incorporating activities and events resulting from the use of plans. Because of the causal relations between activities and events in these data structures we call them *causality trees*. They allow a good overview of what is currently going on and what is planned in the organization and thus improve awareness about the progress of work.

1. The term "plan" is used in this paper as a synonym for "prescriptive process model". Because no confusion is possible - the task of analysing (descriptive) process models will not be dealt with in this paper - we use also the term "process model" as a synonym for plan.

Modular process nets define discrete behaviour structures[1] for the processing of the planned tasks. Generally, only those who have started such nets know about the past, current and prospective states of the tasks associated with these nets. Moreover, these persons must have the (administrative) right to set tasks to the persons who will be set them in the course the interpretation of these process nets. Note that the persons who will be set these tasks in the future (by the workflow management system) are not committed to work on - or even know about - these tasks. Because these tasks are only “planned”, they can be withdrawn, unlike the tasks which have been set either as a result of dialogues (see Section 3.2) or directly set by a competent person or as result of a running process net.

2 Example: Application for Planning Permission (“Bauantrag”)

The following is a typical problem for the city of Berlin today: Investors wish to construct new buildings or to reconstruct old ones, but they have to observe many guidelines in order to receive permission from the local authorities to start their business. Although they have to apply formally only to one of them (e.g. to the Bau- und Wohnungsaufsichtsamt[2]), a number of other local authorities (e.g. Stadtplanungsamt, Hochbauamt, Tiefbauamt, Vermessungsamt, Wohnungsaufsicht, Wohnungsamt, Naturschutz- und Grünflächenamt, Denkmalschutzbehörde[3]) may have to be involved in the process of approval.

Bearing in mind this number of potential objectors, it is not surprising that the time interval before planning permission is granted is often discouragingly long - a delay of more than two years is no exception.

In the search for possibilities to speed up this process, two main problems have been uncovered:

First, the *applicants’ lack of knowledge of* (investors, architects) about the necessary steps. Usually, it takes too long before the rules have been learned and sometimes applications are unsuccessful due to this lack of knowledge.

Second, and even more importantly, the long delay between submission and approval of an application is due to sequential processing and communication overheads between authorities.

Though the second problem (long processing times) is also due to the chronic shortage of human processors in local authorities and to “communication difficulties” between investors and authorities, it is expected that the use of an advanced workflow management system could considerably speed up the evaluation procedures for applications.[4] In addition to more precise accordance of the process with the prescribed rules and an acceleration of the communication between those involved in the evaluation process, the main advantage is expected by paral-

1. *I.e. a partial order relation between tasks as well as explicit time restrictions like deadlines*
2. *Building and Housing Supervisory Authority*
3. *Town Planning Authority, Building Surveyor’s Office, Civil Engineering Authority, Public Surveyor’s Office, Public Housing Office, Housing Department, Parks and Conservation Authority, Architectural Conservation Authority*
4. *Workflow modelling and management can also help to solve the first key problem: The applicants can get more insight into the necessary procedures and documents and, in the case of frequent occurrence of similar procedures, even use an appropriately designed workflow management system. But this case will not be considered in this paper.*

lelization of as many activities as possible. Because some of the activities can be done in parallel and others cannot, it is clear that Petri nets are probably the best choice for a precise and graphical representation of such a mixed alternative and parallel branching structure.

Fig. 1 shows a typical “traditional” net description of a planning permission procedure. The shaded boxes denote transition-bordered subnets with fixed refinements, the roles of actors are assumed to be fixedly associated to the activities and are (with the exception of the “Reviews...”) not mentioned. A net of this type, together with the refinements, would have to be enacted by the administrator of the planning permission procedure. For simplicity of presentation, we have also considered only one part where parallelism is exploited - the reviews done by the different authorities. Clearly, a further refinement will lead to the parallel execution of further activities.

Once enacted, several exceptions can arise in the net-controlled processing at any stage of the evaluation process. For example:

- a previous activity was faulty - e.g. a form-filling error - so that the processing has to be repeated from that point and parts of other work may also have to be repeated
- one or more of the authorities are reorganized, e.g. redistribution of responsibilities
- guidelines, rules or laws have been changed so that the net has to be restructured, but some of the completed activities need not be repeated.

In all of these cases the traditional approaches require a restart of the whole net and often complicated measures are required in order not to lose the work already done.

Another problem closely related to that of exceptions is the potentially different degree of parallelism depending on the case in question. For example, investigations concerning a particular application may require questions to and permission from different sets of authorities, depending on the investment volume, the prospective height of the building and the kind of prospective use. Traditionally, each of these cases involves starting another kind of net.

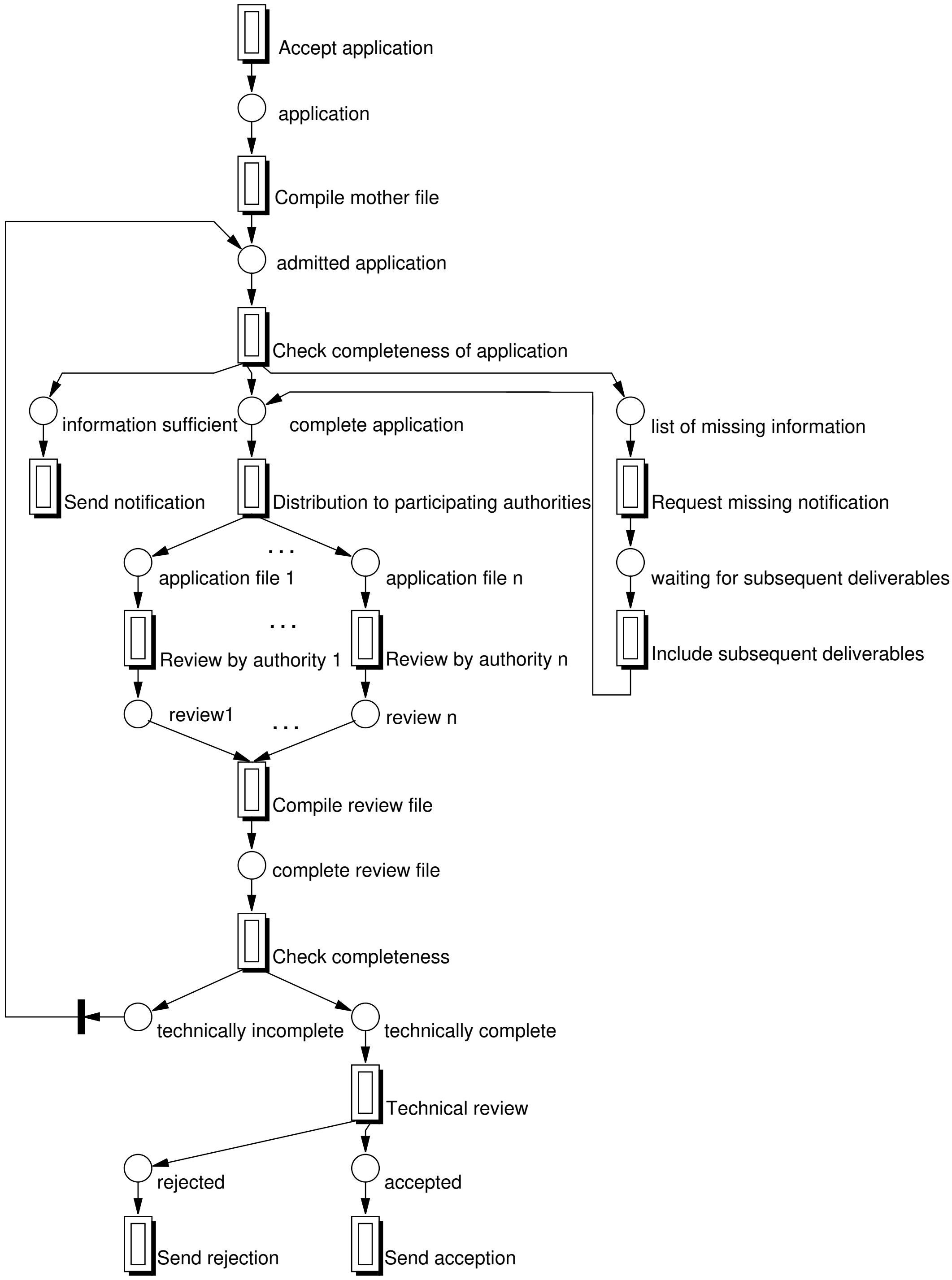

Fig. 1: Traditional net representation of the planning permission procedure

3 The WAM Workflow Management Concept

This chapter gives an overview of the main features of a new hybrid[1] concept for workflow management which has been implemented in a prototype system named after the corresponding project WAM (Wide Area Multimedia group interaction). The system is implemented as a CORBA-based client-server architecture with the main components named by their function as *task management, dialogue management, process management, event handling* and *desktop* components [Adametz et al. 95]. Whereas the desktop components -which function as typical graphical office communication interfaces - exist in as many instances as there are users of the system, all other components are unique for every single (workflow management) system instance. Since this paper aims to make clear the concepts rather than the implementation of the prototype, the sections of this chapter describe the concepts of task, dialogue and commitment, process net, and causality tree as realized in the four single components listed above.

3.1 Tasks

In our concept, work is organized according to tasks. Workers get tasks, they work on these tasks and they end work on certain tasks.
Tasks are parts of workflows[2]. A worker decides to initiate a workflow if an event occurs to which the organization must react and in whose context no action has been taken so far. When a workflow is started, the initiator can assign initial tasks to the new workflow. Tasks consist of a task description and a folder is assigned to every task. With this information, a person is able to work on an assigned task. There are now three options for work on a task:

1. The worker can work directly on the task. The task is performed immediately by doing things according to the task description. This can be done by using tools that are available in the system (e.g. text system, database system, communication system) or the task can be performed without using any computer tools, for example making a telephone call to an applicant.
   The system does not control what the worker is doing in this phase. It just knows that the task is set to the worker and that s/he is working at this moment.
2. The worker can divide the task into sub-tasks and delegate these tasks to other workers within the organization. The sub-tasks can be delegated separately. This means that only part of the work needs to be delegated while the rest is done by the worker him/herself. The delegating takes place in two ways depending on the person who is to get the work: A superior worker can instantly delegate work to others, while others must use the communication tool to negotiate who will work on the task.
3. The worker can start a process model which defines a plan for the processing of the task. A process model defines which tasks, which resources for these tasks and which capabilities to do the tasks are needed and also in which order the tasks are processed. The structure of the (causal) order is defined by a process net model.

1. *In the sense that it is both communication-oriented and workflow-oriented*

2. *Here we use the term “workflow” in the administrative sense of the German term “Vorgang”. In administrations, “Vorgang” usually has two meanings: all the documents belonging to one workflow process and the process itself. In this paper, “workflow” means the ongoing process together with its documents.*

There is no need to choose exclusively between these three possibilities, a combination of the methods is allowed. For example, it is possible to start a process model, delegate two sub-tasks and also do work on the task oneself. Note that a process model started by a particular worker can also set tasks to the initiator him/herself.

Figure 2 shows the connection of tasks to other objects in the WAM concept. Every task has exactly one folder containing the documents needed to work on this task, whereas a workflow consists of many tasks.

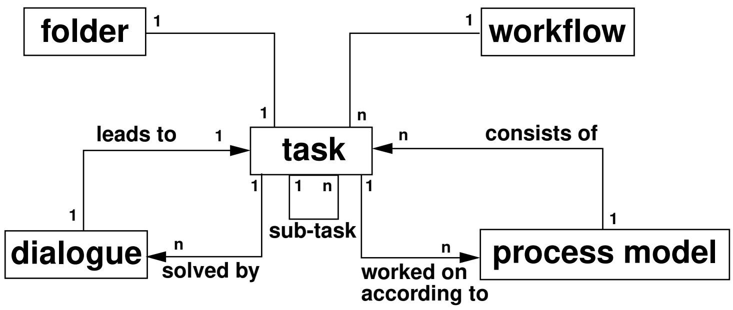

Fig. 2: The task as a central concept

3.2 Commitments

Like many previous systems, the WAM prototype system bases its communication concepts on speech act theory [Austin 62] [Searle 69]. Communication here consists of dialogues. Dialogues consist of utterances of the involved persons according to a special protocol [Winograd, Flores 87]. Such an utterance at a special location within a dialogue is a speech act. There are some types of dialogue that can be characterized by their initial speech act[1]. An example of a type of dialogue is requesting something (answers, work etc.) from the listener.
Types of dialogue are now further refined. This aims at making the use of dialogue structures more practicable and the dialogues easier to adapt to the concrete situation. Depending on the refinement of the dialogue type, the corresponding protocol changes slightly. An example is given for the dialogue type “request”.

Request to
- make a document/folder available
- remove a commitment
- have a consultation
- do a task
- do something (to be refined)

1. In the Coordinator “conversation for action” and “conversation for possibility”

By using dialogues of type “request”, commitments are created between workers if they come to an agreement within the dialogue[1]. By using the refined dialogue type “request to remove commitment” an existing commitment can be removed. This corresponds to the Action-Workflow loops in the ActionWorkflow approach [Medina-Mora et al 92]. If many commitments are created, one can imagine a net of commitments in which the members of the organisation correspond to the nodes of the net. In our approach this concept is extended by forcing everyone who tries to create a commitment to relate it to an earlier commitment. A new commitment can also be declared as the beginning of a workflow. It is now possible to filter commitment trees out of the commitment net between the workers (Fig. 3). These trees show causal relations between existing commitments and the different affiliation of commitments to workflows.

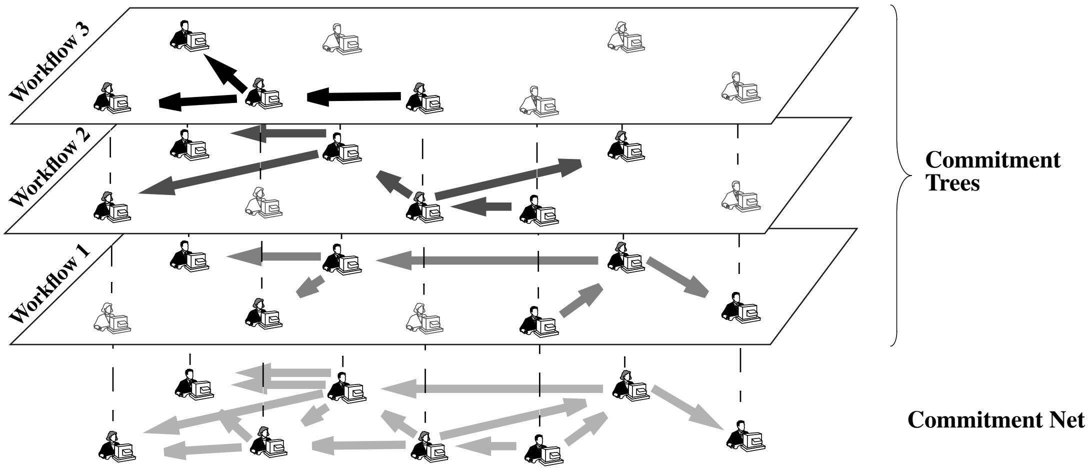

Fig. 3: Commitment trees filtered out of the commitment net

For identifying new tasks and commitments the concept of *events* is used.[2] For example, the setting of a task, the start of a process model, the end of a task and the end of a process model are events. If a new event is generated by a worker or an enacted process model, the event has to be announced to the system together with a context event identification. This means that a task can only be set in the context of another event. Every task has a causal relation to another activity, its “origin”.

3.3 Modular Process Nets

The process models used in our concept are based on *modular process nets* [Wikarski 96] which provide a formal, graphical notation for the representation of the procedural aspects of process models. They can be characterized as a generalization of safe elementary net systems [Reisig 82], i.e. they have untyped tokens with a maximum of one token on any one place.

1. *The dialogue sort “request” means that only the first utterance is a speech act with the illocutionary point of request. If the last utterance of the dialogue is a promise the commitment is established.*

2. *Up to this point, events have been implemented as messages, though in the modular process net definition the concepts of (broadcasted and non-stored) events is well distinguished from that of (uni- or multicast and storagable) messages.*

Additionally, modular process nets provide constructs for the creation and destruction of net instances and for communication of interpreted nets with each other and with the environment. There are two types of communication: *Message passing* (multicast of messages) modelled by *channel places* and *event broadcasting* modelled by *trigger transitions* and *sensor transitions*. Delays, deadlines and other time-related features can be can be modelled using special trigger transitions which generate events with prescribed delays.

Another essential feature of modular process nets is the *module concept*. This is a general concept for a well-structured hierarchical representation of nets. *Modules* are partial nets with well-defined input and output interface transitions called *ports*. Thus, “and” and “or” semantics of the processes to be modelled can be explicitly represented graphically.

As a particular kind of net modules, so-called *task modules* have been introduced for easy representation of activities in process models. Task modules are net modules with predefined refinements and are called *activities* for short. Depending on the number of output ports, activities are assigned a type which is a natural number. The refinement of two typical activities of type 1 (there is just one sensor transition forming the only output port) and of type 2 (there are just two sensor transitions forming the output ports) are graphically represented in Fig. 4. For an interpreted (running) process net, the events corresponding to the output ports are generated by a user via the user interface as the result of a decision.

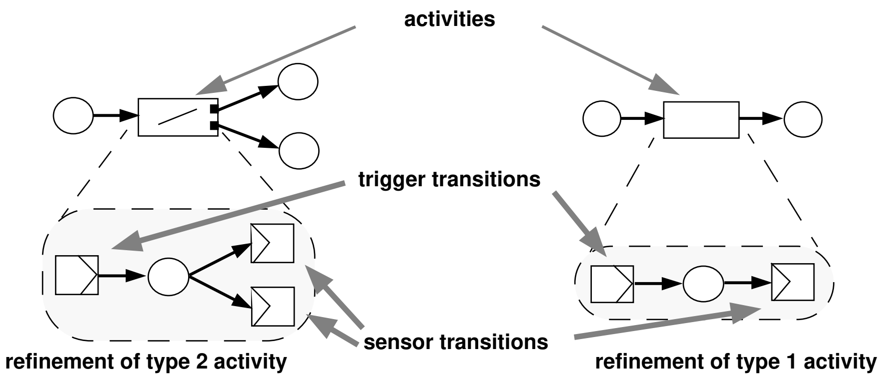

Fig. 4: Refinement of elementary task modules (activities)

Note that activities can be viewed both as a special kind of transitions (from a user’s viewpoint) and as net modules (subnets with special refinements, from the workflow management system’s viewpoint). According to the firing rules of elementary process nets, the firing behaviour of an activity is as follows: If all preplaces of an activity are marked, its input port transition (which is generally a trigger transition) signals an event to the system and a task will be set to the responsible person. When the processing of the task is complete, the net environment generates a (user-initiated) event which is received by one of the task’s sensor transitions and leads to the firing of that transition, resulting in the marking of all the postplaces of the activity. Decision activities (i.e. those with more than one sensor transition) require the person processing the corresponding task to choose between several alternative processing results.

The conceptual structure of modular process nets is modular in the following sense: They are based on *elementary process nets* where all the basic elements and their semantics (in the context of elementary Petri nets) are defined. These are places, transitions, arcs, fusion places, channels and event transitions (sensor, trigger, set-delay, clear-delay) which are graphically represented in a similar way as in Petri nets: Places are represented as circles and transitions as bars or boxes. In particular, event transitions are represented as square boxes with inscriptions and general module transitions, representing transition-bordered subnets, as double-line-bordered square boxes. The graphical representation of interfaces by graphical representation of port transitions can be seen as a completely new feature in Petri nets. Task transitions, as a particular kind of subnets, are represented as rectangular boxes. Besides the elementary ones with standard refinement as explained above, represented by a single line, complex task modules with arbitrary refinement can be graphically recognized by double lined shape. In order to be able to handle and combine complete process models, the notion of *process module* has been introduced. A visualization of this conceptual structure is given in Fig. 5.

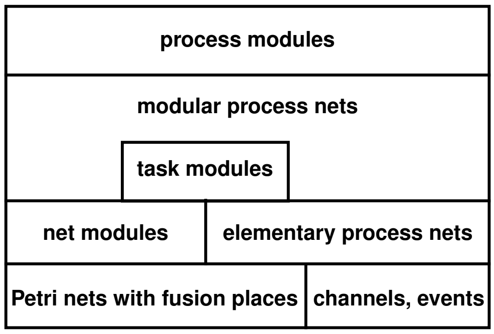

Fig. 5: Conceptual structure of modular process nets

3.4 Combining Tasks and Process Nets

To enable a uniform - and thus more understandable and easier-to-implement - treatment of tasks, commitments and plans, all these concepts are implemented in the framework of tasks. While the communication feature allows the ad-hoc creation of commitments (tasks) between workers and gives the possibility of working without the need to follow a plan, work which following process models (plans) are very efficient and secure. Enacted plans (formally defined as process modules) generate tasks that are distributed to members of the organisation. When work on a task is completed, the process model defines what tasks to do next. Some WFMS consider a process model to cover all activities within an organisation. Our approach aims at modelling parts of the work in an organisation that are routine and that do not change constantly depending on the situation. In this way the work in an organisation or a department is defined by some routine workflows represented by process modules. It is possible that within one workflow more than one process module is initiated because there may be different models for different parts of the work within the workflow. But how can these different process modules within one workflow be combined and how can the non-plannable parts of a workflow be dealt with? This can be achieved by allowing the worker to choose between different methods of working on a task as already mentioned. For example, s/he may use the communication tool to create commitments to others while they start new procedures (Fig. 6).

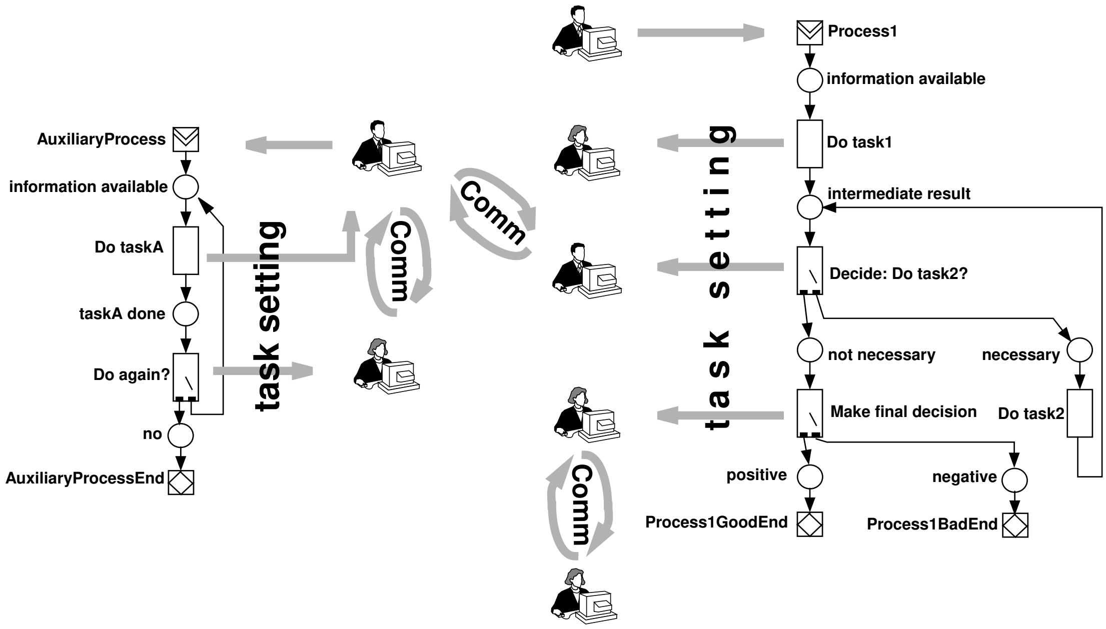

Fig. 6: Combination of process nets and communication facilities

3.5 Causality Trees

One of the most important basic functions of WFMS is the structuring of group work. A good means of doing this is the modelling of causal connections between tasks which can be realized on the basis of *events* as introduced and used in the previous sections. Two tasks T_0 und T_1 are in a *causal relation* to one another if T_0 is the cause of T_1. Each sub-task T_j of the task T_i is dependent on T_i in the sense that if T_i is withdrawn then the subtasks T_j may become unnecessary. In order to enable the representation of parallel work, so-called *causality trees* [Faustmann 96] have been introduced and are used for keeping track of these causal dependencies. Each sub-tree is causally dependent on its root, i.e. if the task at the root of a sub-tree is cancelled then all the tasks of that sub-tree are called into question.

time

event	from	to
applicants' call	customer	b&h superv. auth.
inquiry to plann. auth.	b&h superv. auth.	planning auth.
start process model	planning auth.	planning auth.
start process model	b&h superv. auth.	b&h superv. auth.
task: conserv. query	b&h superv. auth.	storage dept.
task: earlier queries?	planning auth.	planning auth.

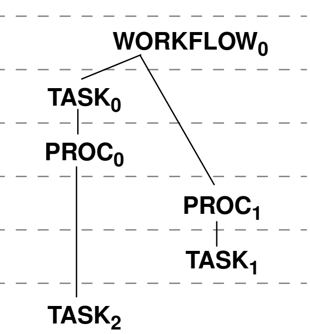

Fig. 7: A sequence of events builds a causality tree

Figure 7 shows an example of a causality tree for a specific workflow. Because every new event is sent to the system together with a context event identification, a causality tree can be composed from this information. The sequence of events here only consists of events composing the tree. The next state of the workflow could be the finishing of the task ‘earlier queries?’ by the employee in the town planning authority. In this case, a decomposing event would be generated and the event $TASK_2$ would be deleted from the causality tree. As a consequence, the process model enacted in the town planning authority would be informed of the finished task.

The use of causality trees offers some advantages. Firstly, causality trees give an overview of the running tasks within a workflow. In the WAM system one can see directly from a causality tree what tasks were set to what persons at what time, what process models were started at what time by whom and what causal relations exist. Causal relations show the context in which a task was delegated or a process model started. Whereas there is certainty about the causal relations between current activities, this cannot be said about those relations where future activities and events are included. These future parts of a workflow are defined by enacted process models and the setting of their tasks depends on future decisions about other tasks within the process model. The graphical representation of these dependencies helps to deal with these difficulties.

Secondly, causality trees allow an adaptive management of a workflow with regard to current activities within the workflow. Adaptive management of a workflow means changing the state of a workflow by activities other than those normally included in the workflow. This must be done if there is no way to adapt the state of the workflow to reality. Adaptation to reality is defined in our concept as sending an event that changes the state of a causality tree and thus initiating some system actions. In normal cases, a causality tree is composed of expected events. As the work proceeds, the leaves of the tree are deleted as decomposing events occur (arrive[1]). But sometimes this might not be the case because there are events that change the structure of a causality tree essentially. For example, such a situation occurs when an event is to be deleted that still has children in the causality tree, e.g. when a worker wishes to abort delegated sub-tasks. Further possible reasons for such changes are changes of the processing times and of deadlines.

4 Application of the WAM Concept to the Planning Permission Example

This chapter gives a brief explanation of the advantages to be expected from the application of our concept to the example given in Section 2. Seen from the graphical representation of the process net, the difference may not seem so striking, but we hope that the reader will bear in mind the ideas and details of the concepts explained above.

The process module shown in Fig. 8 is started by an authorized clerk as a particular instance of this application which is given a unique name and number. All task modules have the same refinements as explained above. The tasks are assigned to persons who have roles corresponding to role names as written in the brackets after the task names. Those with empty brackets “()” will be assigned to the starter of the net module him/herself, while the others are assigned to other clerks - in our example with the role “tc” (technical clerk). The start symbol

1. In terms of our message-passing implementation, events do in fact “arrive” means that in reaction to the event “Start Planning permission” emitted by the workstation of the clerk, a process module instance is created, whereas the process module’s end symbols denote the destruction of the process module instance and the signalling of the respective end event “PlanningPermissionRejected” or “PlanningPermissionGiven”. The shaded places “admitted application” and “complete application” are fusion places and have to be imagined as one place each.

The most visible difference to the Petri net given in Fig. 1 is the activity “Give other authorities a share”. In that net, the degree of parallelism had to be specified explicitly, whereas in the process net it can be defined “on the fly”. The clerk is informed of the necessity and degree of giving another authority a share in the given process by the corresponding task description. In accordance with this description s/he has to start a certain number of (possibly even different) process modules of the type “Review by authority n” and wait for their completion. After they have all been completed (or after some additional dialogues in the case of further exceptions) s/he confirms the completion of the task in question and the next task (“Compile review process”) is set. Note that in our context of exceptions, the variable degree of parallelism can be considered in a sense as “foreseen exceptions” (although this appears to contradict the definition given at the beginning of the paper).

Clearly, the following tasks „Check completeness...“ and „Technical review...“ will be set to other clerks with the role „tc“ (technical clerk) and during work on these tasks they can use further, appropriate process modules and/or communicate with other clerks using the dialogue facilities of the workflow management system.

As already mentioned, our concept allows the processing of the workflow to be stopped according to the process module at any point of time by the starter of this module. Then, any untouched tasks are not set, whereas the completed results of the completed tasks are available to the initiator in the context of the task in which the process net was started.

In this way, other exceptions such as fusions and splittings of responsibilities for tasks, reorganizations of authorities etc. can be handled in a systematic and aware way.

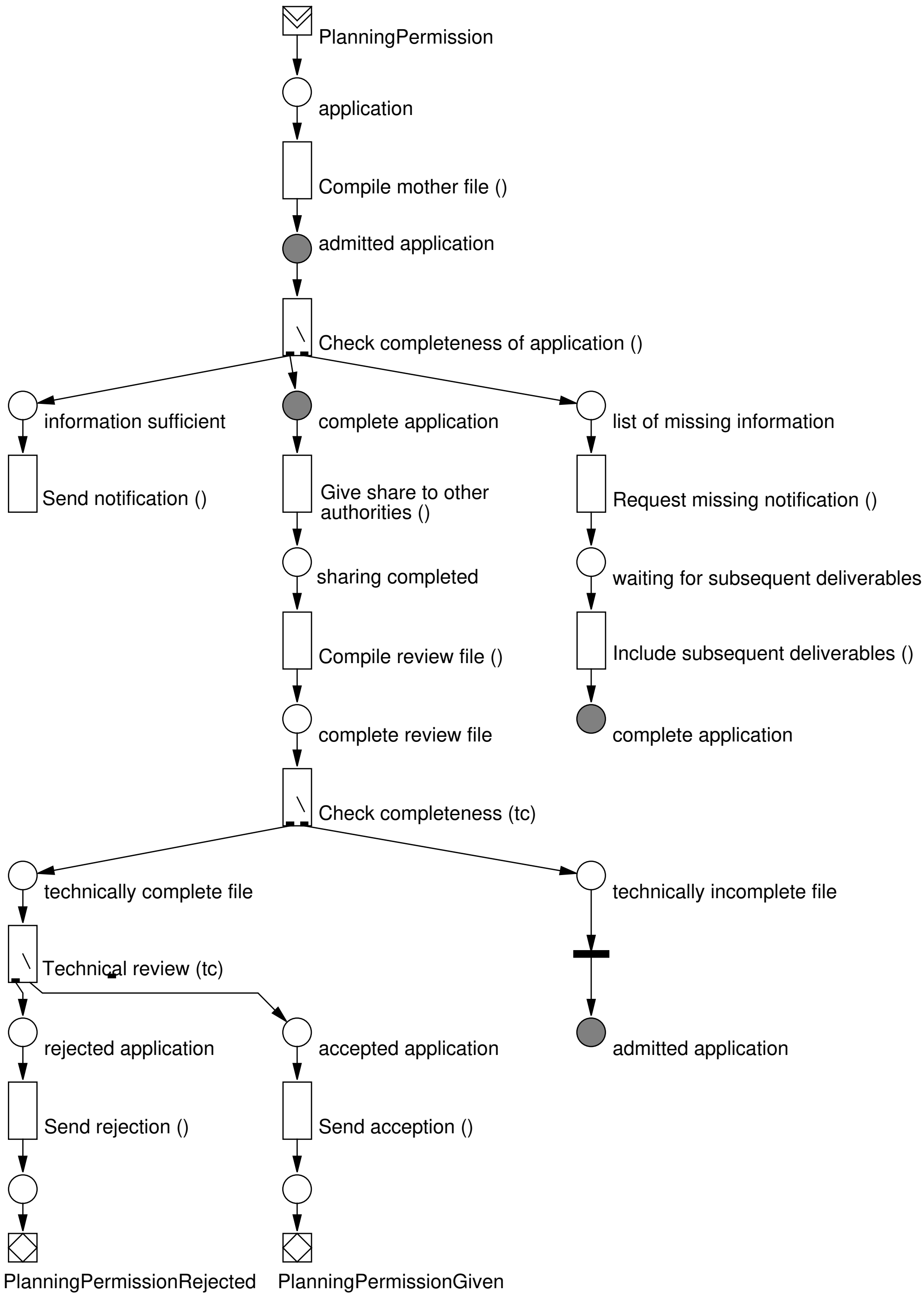

Fig. 8: Planning permission procedure represented as an executable modular process net

.

5 Conclusion and Outlook

A framework was presented that contains concepts for increasing the flexibility of workflow management systems. Workflows can be carried out both by routines (using process models) and communicational activities. During the execution of a process model, the commitments of single workers are pushed into the background, but they can be made explicit in unplanned situations.

In the DeTeBerkom-funded project WAM (Wide Area Multimedia group interaction) a prototypical workflow management system has been developed that was presented first in autumn of 1995. Its concepts are primarily the result of research concerning workflows in the field of public administration. The prototype has been demonstrated using the example of a public administration showcase scenario which also included multimedia communication for synchronous exception handling.

Future research in this direction must concentrate on further concepts that support the worker in his/her daily work. When dealing with non-plannable and especially exceptional situations, still more extensions are needed, in particular for the integration of transactional concepts. This means that invalid data, which currently has to be handled manually, can be dealt with automatically.

Another concept for ensuring adaptability and flexibility in Petri nets is that of explicit *resource management* which is described in a companion paper [Han96]. Though not yet implemented, there is an intention to marry both concepts in the future in order to obtain a still more adaptable concept for net-based workflow management.

Acknowledgements

Many people were involved in the WAM project and contributed to the concepts presented in this paper. Burkhard Messer (as the initiator of the project), Beate Barthel and Helmut Adametz as members of the project group were the main contributors. We'd like to thank Nick Grindell for his valuable comments concerning language aspects.

6 References

[Adametz et al. 95]
Adametz, Helmut; Barthel, Beate; Faustmann, Gert; Messer, Burkhard; Wikarski, Dietmar; Fellien, Arne: The WAM Project - Final Report. Internal ISST Report, September 1995

[Adametz et al. 96]
Adametz, Helmut; Barthel, Beate; Faustmann, Gert; Messer, Burkhard; Wikarski, Dietmar: Konzepte flexibler Workflow-Management Systeme. In: Sandkuhl, Kurt; Weber, Herbert (Ed.): “Telekooperations-Systeme in dezentralen Organisationen”, Fraunhofer ISST, Februar 1996

[Austin 62]
J. L. Austin: How to do things with words. Oxford, Oxford University Press, 1962.

[Deiters 92]
Deiters, Wolfgang: A View Based Software Process Modeling Language. Dissertation, Technische Universität Berlin, December 1992

[Deiters et al. 95]
Deiters, Wolfgang; Lindert, Frank; Schiprowski, Reinhard: A Transaction Concept for Software Process management with FUNSOFT Nets. ISST-Bericht 30/95 der Fraunhofer Einrichtung Software- und Systemtechnik, Dortmund, Oktober 1995

[Faustmann 96]
Faustmann, Gert: Workflow Management and Causality Trees. Proc. of the 2nd Intl. Conference on the Design of Cooperative Systems (COOP‘96), June 12-14, 1996, Juans-les-Pins, France.

[Gruhn 91]
Gruhn, Volker: Validation and Verification of Software Process Models. Dissertation, Universität Dortmund, Bericht Nr. 394/91, 1991

[Han et al. 96]
Han, Yanbo; Himmighöfer, Jürgen; Schaaf, Thorsten, Wikarski, Dietmar:
Management of Workflow Resources to Support Runtime Adaptability and System Evolution. Submitted to the Workshop on Adaptive Workflow, October 30-31, 1996, Basel, Switzerland

[Medina-Mora et. al. 92]
Medina-Mora, Raul; Winograd, Terry; Flores, Rodrigo; Flores, Fernando: The Action Workflow Approach to Workflow Management Technology. In: Turner, Jon; Kraut, Robert (Eds.): CSCW‘92 Sharing Perspectives, Proceedings of the Conference on Computer-Supported Cooperative Work, Toronto, Canada, 31.10.-4.11.1992, ACM Press, New York, 1992, pp. 281-288

[Reinwald 93]
Workflow-Management in verteilten Systemen, Teubner-Texte zur Informatik, B.G. Teubner Verlagsgesellschaft, 1995

[Reisig 82]
Reisig, Wolfgang: Petri Nets. An Introduction. Berlin, Heidelberg, New York: Springer, 1982

[Searle 69]
John R. Searle: Speech Acts. Cambridge, Cambridge University Press, 1969.

[Wikarski 96]
Wikarski, Dietmar: An Introduction to Modular Process Nets, Report 96-019, International Computer Science Institute, Berkeley, CA, April 1996

[Winograd, Flores 87]
Winograd, Terry; Flores, Fernando: Understanding Computers and Cognition - A New Foundation for Design. Second Publishing. Addison Wesley, Inc. Reading et al., 1987