Flexible Handling of Work Processes by Situation-dependent Support Strategies

Gert Faustmann
Fraunhofer Institut Software- und Systemtechnik ISST
Kurstr. 33, D-10117 Berlin, Germany
Gert.Faustmann@isst.fhg.de

Abstract

Although work processes, like software processes, include a number of process aspects such as defined phases and deadlines, they are not plannable in detail. However, the advantages of today's process management, such as effective document routing and timeliness, can only be achieved with detailed models of work processes.

This paper suggests a concept that uses detailed process models in conjunction with the possibility of defining the way a process model determines the work of individuals. Based on the WAM *approach[1], which allows workers to choose methods for their tasks according to the situation, we describe features to carry out planned parts of a process with workers always being able to start exceptional mechanisms. These mechanisms are based on the modelling paradigm of linked abstraction workflows (*LAWs*) that describe workflows at different levels of abstraction and classify refinements of tasks by the way lower tasks can be used.*

1. Introduction

Process management or workflow systems give the opportunity to support document-based work processes with the help of a pre-planned model of the process. Workers are informed about current tasks and the documents and tools they need are made available. By strictly mapping the model onto the real world, certain quality requirements of a real world process can be obtained. In this way, the performing of all specified activities can be guaranteed, the time for routing documents and information between workers can be minimized and one can keep a general overview of the whole workflow. These advantages are based on the assumption that a real world process can be modeled in detail in advance.

Research in CSCW, workflow management and software process management has shown that real world processes are composed both of well structured parts and parts that cannot be foreseen ([18],[13],[4]). An additional view of work routines is given here which we call processual penetration. The processual penetration of a workflow describes which parts of the workflow can be assigned specific methods and which granularity the descriptions of these methods have. We distinguish between global and local methods and between high and low granularity of these methods.

		Methods	
		global	local
Granularity	high	Administrational work	Assistance work
	low	Project Planning	Rules of work

While most administrative work is plannable in great detail, software projects in particular can be planned only to a limited degree. Assistance work stands for a detailed method that perfoms a small part of a larger task. An example of assistance work is the preparation and assessment of meetings with time scheduling plus the taking and distibution of minutes ([16]). Rules of work only appear at a certain place within a workflow and are often only of low granularity. Quality management aspects of a process are one example of this.

As a result, when planning processes in detail, the workflow paradigm of controlling real world processes with process models has to be extended by additional features. One approach is giving workers the possibility to use exceptional functions and breaking out of the model-

1. *This paper partly describes results from the project* WAM *- Wide Area Multimedia Group Interaction.* WAM *is funded by the Telekom subsidiary DeTeBerkom.*

based control cycle ([3], [10], [12]). Another approach is to leave parts of a workflow open and specify these open parts when the process arrives at these places ([11]). This paper describes an approach that uses very detailed process models as the basis for process support. With the help of a flexible enactment strategy which is realized in the WAM approach plus additional features to strengthen model correspondence and cope with exceptional situations, workflows of different process penetration degrees can be supported.

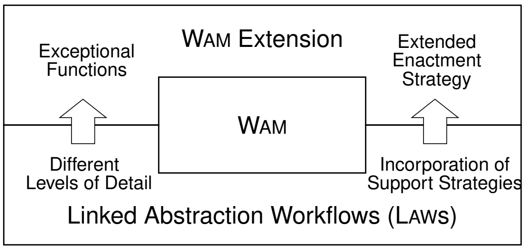

Fig. 1: Logical architecture

The following chapter describes the core WAM approach. The third chapter then introduces the modelling paradigm of LAWs, while the fourth chapter shows how to use LAWs to support processes with different structures. The closing chapters give a short insight into comparable approaches and an outlook to future work.

2. WAM Approach

The main objective of the approach is to combine workflow technology with general groupware approaches. The concept allows the construction of processes by workers during process enactment and gives the opportunity to integrate groupware tools. Particular attention is paid to integration of processes with communicational work ([7], [16]).

2.1. Workflow-Metamodel

Work in WAM 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. 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 (see also Fig. 2):

1. Direct work
   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 by 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. Delegation
   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 makes it possible to delegate just part of the work while the rest is done by the worker him/herself. Delegation is realized using a communication tool.
3. Starting sub-processes
   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) sequence is defined by a process net model.

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.

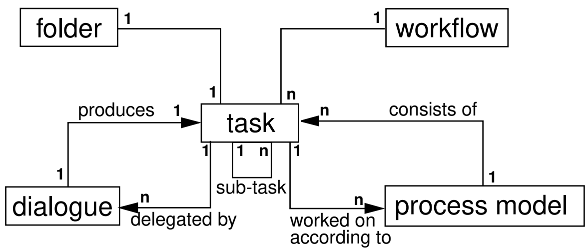

Fig. 2: WAM Workflow-Metamodel

Because WAM offers different methods of processing tasks, it is possible to combine predetermined work with self-defined, spontaneous work. This approach aims at using workflow models only in contexts which clearly involve processual work. It is independent of the processual penetration of work.

2.2. Enactment in WAM

To demonstrate the WAM approach, we describe an extract from a work process showing the enactment of a small process model. Fig. 3 gives an overview of the example in which processes, communcational parts and basic activities are connected.

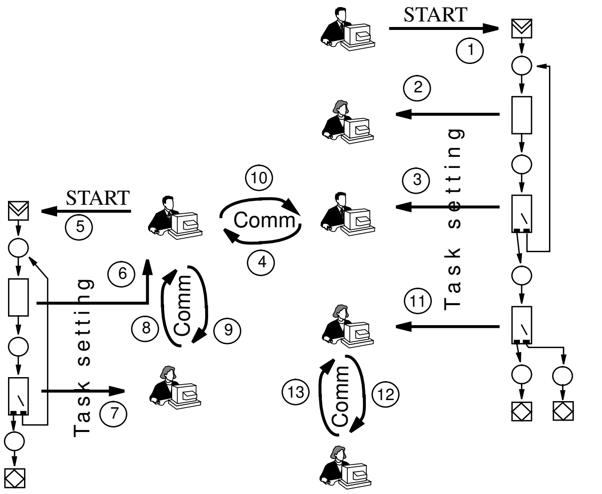

Fig. 3: Task distribution in WAM

First, a work process is initiated by starting a process model (1). The first task is set to a worker (2) who immediately performs this task. The following task involves a decision (3) and is carried out via the communication tool (4). The contacted colleague starts a process on his own (5) which sets a first task to the worker himself (6) and a second task to another worker (7). The latter asks the initiator for further information (8)(9). When the process ends, it provides a result which is returned via the dialogue (10). The final task of the main process (11) is also performed via a dialogue (12). A solution is sent back immediately (13). The results of the main process are delivered, as in the case of the sub-process, to the initiator.

2.3. Drawbacks of the WAM approach

Although the WAM approach offers a flexible way of supporting processual work, there are some missing features of support:

- WAM uses abstract models of processes that are refined by the workers while doing a process. It is not clear at the beginning of a process precisely which activities will take place. At best, there may be recommendations for how to refine a given task.
- Having chosen a process model as a method for a task it is difficult to change this decision or parts of the process model when enactment has started. In core WAM, a worker may use an extra task to contact the initiator or workers within the process to influence further work activities.
- After some time of supporting work with WAM a situation-dependent hierarchy of processes and tasks appears. WAM offers no other features to coordinate group work than the availability of contacts to other workers within the whole process. Internally, WAM provides a data structure called causality trees for storing the (causal) relations between activities of all abstraction levels. This could be used as the basis for further coordination mechansims ([5]).

3. Linked Abstraction Workflows (LAWs)

LAWs provide aspects which could form the basis for overcoming the drawbacks of the WAM approach. They introduce different levels of a process model that represent different abstractions of work. On the lowest level they provide very detailed work descriptions. The refinements (the connections between two neighbouring abstraction levels) are classified by refinement degrees that influence the way a worker can use the methods of a deeper abstraction level. In exceptional situations, the different levels of abstraction allow smooth changes of method ([8]).

3.1. Different levels of abstraction

Here, in contrast to WAM, a hierarchy of tasks is modelled *before* process enactment. The main objective of the process forms the root of this hierarchy (Fig. 4).

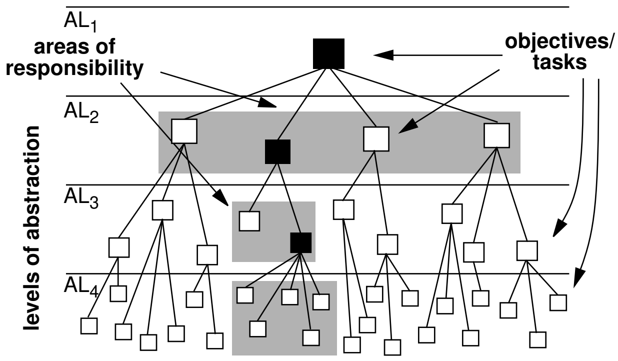

Fig. 4: Hierarchy of objectives

On the lowest abstraction level there are very concrete objectives that can be reached by certain activities. Within this hierarchy, leaves may appear at different abstraction levels. A leaf is symply an objective which is not refined any further. This is independent of the level of abstraction a leaf is located at. A LAW can thus contain objectives which represent very abstract tasks without being assigned to a concrete method.

The objectives of one abstraction level that have the same “parent” objective are denoted as an area of responsibility, since when enacting a LAW, a worker is responsible for all sub-objectives of a task that was set to him.

In the initial state of a LAW, all objectives of one abstraction level that belong to the same parent objective can be ordered to process descriptions. A process description may contain the following aspects:

- A set of tasks (obviously, because children objectives are always identified as tasks),
- a set of documents or information pieces produced by the process
- an ordering relation that combines tasks with documents and/or pieces of information and relates tasks to one another.

In order to describe a process, at least the set of tasks or the set of documents are required. If an objective is not subdivided (refined) into other objectives, it may nevertheless be refined to give a process that is only described by the documents to be produced. The ordering relation can only be applied when a process is described by tasks.

Additionally, a process description must also contain an interface definition for input and output of the process. This can be done by specifiying which documents are needed for starting the process and which documents are produced by the process. This gives the opportunity to link all processes of an abstraction level together, resulting in a single overall process at each abstraction level (Fig. 5).

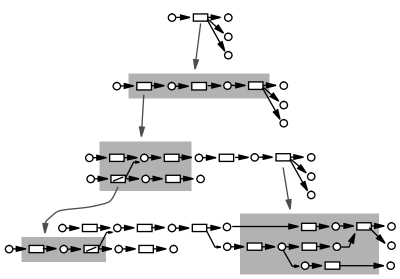

Fig. 5: Linked Abstraction Workflow (LAW)

3.2. Modelling support strategies

As stated above, an objective can generally be refined by a process description. If this process description contains tasks, these tasks can be interpreted as objectives and be further refined in the LAW. So the LAW models tasks and assigned methods. The way tasks and assigned methods are set to a worker depends on the support strategy that is realized in the process management system. Many systems set tasks to workers and strictly assign a method by presenting documents and corresponding tools. Some even set tasks of only the lowest abstraction level to workers. The WAM approach just recommends a method for a given task and leaves the final decision on how to refine (and thus how to do) work to the worker.

LAWs contain refinement degrees for all refinement relations between an objective and all parts of its corresponding process description. At the moment a refinement degree can be one of the following marks:

- **CAN** is the default mark and is like a recommendation to use the proposed method.
- **MUST** forces the worker to use the given method. S/he may use further methods, but is not allowed to leave out the specified method.
- **STRICT** is like MUST without the possibility to use other methods.

4. Work support with LAWs

Work support with LAWs is based on the managment of tasks in the WAM approach. If a LAW is to be enacted it can be initiated on any abstraction level. Every level contains one process model (of that certain abstraction level) that can be interpreted.

If a task is to be done, the task is set to a worker (there may be mechanisms using roles and organizational models that are not discussed here). Just like in WAM, the worker now has various options for proceeding with work. But here, these possibilities are restricted by the specific refinement degree of that task in the LAW. Consider, for example, a process that is specified by some tasks, documents and an ordering relation between tasks and documents. If the refinement to this underlying process is marked with CAN, the worker may use the process as well as any other method. S/he can work as s/he wants to. If the refinement is marked with MUST, the child process has to be used by the worker although s/he may also use additional methods. In the case of a STRICT, the worker must use the process and may not use any additional method.

Because process descriptions consist of different parts, refinement degrees can differ between these parts. It is therefore possible to refine an objective to give a process that is described by tasks and an ordering relation on these tasks, but with a STRICT tag on sub-tasks and a CAN tag on the ordering relation. This would mean that all specified tasks have to be done, but that the specified ordering of these tasks is just a recommendation. Note that the overall use of STRICT tags produces process models in the common sense of workflow management where no flexible process adaptation is possible.

4.1. Exception Handling with LAWs

The advantage of the WAM approach was the possibility to react flexibly to changes in the external situation. Refinement degrees now return some kind of strictness to model deployment. But the formal structure of LAWs offers a transactional-like approach to exception handling.

If a worker encounters a problem when working on a task, s/he should be given the possibility to contact the contractor of this task. In a LAW, the contractor of a task is the owner of the superior task in that LAW. The contractor now has to decide how to solve an existing problem. S/he may have already done some other work within the context of her/his own task. If s/he is not able to cope with the problem within her/his own area of responsibility, s/he also has the possibility to give a task back to the contractor. Fig. 6 shows the skipping of abstraction levels in the case of an exception at a deeper level.

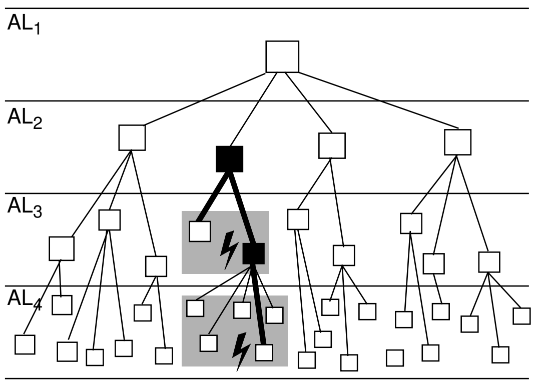

Fig. 6: Handling exceptions over abstraction levels

If a worker wants to break a refinement tag because the external situation demands methods other than those specified, s/he has to contact the contractor of the task in question. In this way, it is even possible to break STRICT tags and adapt to external changes.

4.2. Example: project management

The following example will show the special application of exception handling in project management. Fig. 7 illustrates one main process that is being handled by a supervisor, two processes of the second abstraction level (worker 1 and 3) and a process of the third abstraction level. Tasks are depicted as bars similiar to the graphical presentation of projects in planning tools. An inner bar of a task shows the progress of a task.

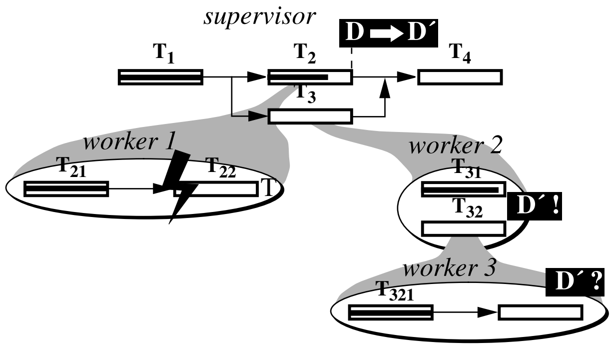

Fig. 7: Deadline propagation

When worker 1 begins work on task T_{22}, s/he realizes that s/he cannot finish within the deadline D specified by the supervisor. S/he starts an exceptional function that leads the supervisor to change the deadline D to D´. The system automatically propagates this deadline to all dependent sup-processes of the next level. So worker 2 gets the new deadline D´ and can decide if the system should propagate the new deadline to subordinated processes too.

5. Related Work

First approaches to flexible handling of workflows were integrated into workflow systems based on the strict workflow paradigm. The MELMAC system proposed changeable process areas that had to be specified before process enactment ([2]). ProMInanD extends the possibilities of a worker and allows the return of tasks (electronic cirulation folder) and changes to the specified path of a folder ([12]).

Approaches exist which use hierarchical views of processes. The OBM approach allows the gradual refinement of a process with the objective of supporting different stages of process modelling ([17]). The PEACE system goes one step further ([1]). It integrates a hierarchy of goals with sets of activities. Additionally, these activities can be ordered. So one can distinguish between strategic (goal hierarchy), tactical (goals associated to activities) and operational process modelling (ordering of activities). The process management system ProcessWEAVER uses a hierarchy of activity types ([9]). These activity types are bound to process fragments that again can be integrated as methods into other process fragments. But these refinement decisions have to be made when modelling a process. Furthermore, a strict approach of support strategy was chosen.

Another view on work processes is given by the Action-Workflow approach that models processes by sequences of communcative actions ([19]). The flexibility of conversations is the basis for process enactment. With its flexible way of building situation-dependent processes this approach is comparable to the WAM approach. In the same direction but with its main focus on resource management, the COSNA approach allows the incorporation of subprocesses into processes during enaction ([11]).

Finally the CoMo-Kit approach allows the alternation of process planning and process execution with special emphasis on design processes ([14], [15]). Tasks can be decomposed by workers and delegated to other workers. Exception handling is realized on the one hand by looking at physical (input/output) relations between tasks and using organizational models, and on the other hand by using a truth maintenance system allowing management of previous decisions and adaptation to changed decisions.

6. Outlook

Some extensions to the concept will be made in the near future. Besides the storage of causal dependencies of tasks and activities, the storage of "physical" dependencies (characterized by input/output-relations) extends the possibilities for reacting to exceptional situations. A first step in this direction was taken in [6], in correspondence with [14] and [15].

We plan to realize a prototype implementation to confirm the applicability of the introduced concepts. This implementation will be based on the existing WAM prototype.

7. References

[1] S. Arbaoui, F. Oquendo, Goal Oriented vs. Activity Oriented Process Modelling and Enactment: Issues and Perspectives, in.: Brian C. Warboys (ed.): Software Process Technology, EWSPT‘94 Proceedings, LNCS 772, Springer, Berlin, Heidelberg, 1994, pp. 171-176.

[2] W. Deiters, A View Based Software Process Modeling Language, Dissertation, Technische Universität Berlin, 1992.

[3] G. De Michelis, M. A. Grasso. Routines and Conversations. In: *Structured Programming* 14, 1993, pp.110-118.

[4] M. Dowson, Ch. Fernström, Towards Requirements for Enactment Mechanisms, in: Software Process Technology, Third European Workshop, EWSPT‘94, Villard de Lans, France, February 7-9, 1994, Proceedings, LNCS 772, Springer, Berlin, u.a., 1994, pp. 90-106.

[5] G. Faustmann, Workflow Management and Causality Trees, COOP’96, Second international conference on the design of cooperative systems, Juan-les-Pins, France, June 12-14, 1996.

[6] G. Faustmann, A Design Rationale based on Situative Process Evolution, Abstract, Workshop "Coordinating Work Processes", available through "http://wwwagr.informatik.uni-kl.de/~dellen/Beitraege.html#Faustmann", University of Kaiserslautern, August 22-23, 1996.

[7] G. Faustmann, D. Wikarski, Exception handling in Petri-Net-based Workflow Management, International Conference on Practical Aspects of Knowledge Management, Workshop on Adaptive Workflow, Basel, October 30-31, 1996.

[8] G. Faustmann, Cooperative Exception Handling with Hierarchical Processes, COOP’98, Third international conference on the design of cooperative systems, Cannes, France, May 26-29, 1998 (to appear).

[9] C. Fernström, ProcessWEAVER: Adding Process Support to UNIX, Proceedings of the Second International Conference on the Software Process, Berlin, February 25-26, 1993.

[10] G. Cugola, E. Di Nitto, C. Ghezzi, M. Mantione, How to deal with deviations during process model enactment, in: Proceedings of the 17th International Conference on Software Engineering, Seattle, Washington, April 1995.

[11] Y. Han, J. Himmighöfer, Th. Schaaf, D. Wikarski, Management of Workflow Resources to Support Runtime Adaptability and System Evolution, International Conference on Practical Aspects of Knowledge Management, Workshop on Adaptive Workflow, Basel, 1996.

[12] B. H. Karbe, N. G. Ramsperger, Influence of Exception Handling on the Support of Cooperative Office Work, in: S. Gibbs, A.A. Verrijn-Stuart (Eds.), Multi-User Interfaces and Applications. Elsevier Science Publishers, North-Holland, 1990.

[13] M. Löwe, D. Wikarski, Y. Han, Higher-Order Object Nets and Their Application to Workflow Modeling, Forschungsbericht der Techn. Universität Berlin, Berlin, 1995.

[14] F. Maurer, Project Coordination in Design Processes, Proceedings of WET ICE 96, Stanford, California, June 19-21, 1996.

[15] F. Maurer, G. Pews, Supporting Cooperative Work in Urban Land-use Planning, COOP’96, Second international conference on the design of cooperative systems, Juan-les-Pins, France, June 12-14, 1996.

[16] B. Messer, G. Faustmann, Efficient Videoconferencing with Workflow Management Systems (in german), in: K. Hammer, D. Schmolke, F. Stuchlik (Hrsg.): Synergie durch Netze (Tagungsband), Otto-von-Guericke-Universität Magdeburg, 5.-6. Oktober 1995.

[17] J. Sa, B.C. Warboys, Modelling Processes Using a Stepwise Refinement Technique, in.: Brian C. Warboys (ed.): Software Process Technology, EWSPT‘94 Proceedings, LNCS 772, Springer, Berlin, Heidelberg, 1994, pp. 40-58.

[18] K. Schmidt, Riding a Tiger, or Computer Supported Cooperative Work, in: L. Bannon, M. Robinson, K. Schmidt, Proceedings of the Second European Conference on Computer Supported Cooperative Work, Amsterdam, The Netherlands, September 25-27, 1991, pp. 1-16.

[19] T. Winograd, A Language/Action Perspective on the Design of Cooperative Work, Computer-Supported Cooperative Work: A Book of Readings, Morgan Kaufmann, San Mateo, 1988, pp. 623-653.