SEPP/OT as a Management Concept for the Modeling of Workflows

Technical University of Ilmenau
Faculty of Computer Science and Automation
Inst. of Theoretical and Technical Computer Science
Department of Computer Architecture
W. Fengler, A. Karg, A. Mühlpfordt
email: {Wolfgang.Fengler|Andrea.Karg}@TheoInf.TU-Ilmenau.DE

M. Wolf
OWiS Software GmbH. Lindenstr. 28, D-98693 Ilmenau
email: mwolf@otw.de

ABSTRACT In this contribution the SEPP/OT management concept for software development projects is introduced with all its elements like activities, processes and views. Some hints are given about the relation of SEPP/OT to diagrams defined by the UML standard. It is assumed that the reader is at least a little bit familiar with the UML standard. The main aim of this article is to show how this management concept for software projects also can be used for the modeling of workflows. It will be shown which parts of the concept are relevant for workflow modeling and how it has to be interpreted. A section about tool support completes this paper.

1. INTRODUCTION

In the last two years the UML (Unified Modeling Language) developed by Booch, Jacobson and Rumbaugh became the new standard for modeling in the software design process. It includes several diagrams for visualizing the behavior and the context of the system which is being developed. The utilization of the UML only makes sense with support of appropriate tools to ensure the consistency between the several diagrams.

The introduction of a management concept for the organization of a software development project as an extension to the UML would be very helpful. It should also be applicable to the design of workflows or business processes to benefit from the advantages of UML modeling in as many areas as possible. An attempt to implement such a concept is the "Unified Process" by Rational Software Corporation which can be used as an extension to the tool "Rational Rose". It "organizes projects in terms of workflows and phases, each of them consisting of one or more iterations. With the iterative approach, the emphasis of each workflow will vary throughout the lifecycle. Milestones enable the management to assess progress" [RUP].

Our approach to the integration of such a concept is SEPP/OT realized as an add-on to the OTW®2 (Object Technology Workbench) of OWiS Software GmbH. In addition to the standard diagrams of the UML the OTW®2 contains an additional diagram for representing the system dynamics, the so called Object-Process-Net (OPN), which can be simulated.

SEPP/OT represents a software development concept for professionally managed projects using object-oriented analysis and development technology, especially the UML. The substantial advantage of SEPP/OT is that it begins with use cases which represent the dynamical aspects of the system. Therefore, in SEPP/OT the processes are the main parts of a system. Due to the emphasis on processes SEPP/OT is predestined to use it for workflow modeling as well as for software design [OTW98].

2. SEPP/OT AS A UML BASED MODEL FOR THE SOFTWARE DEVELOPMENT PROCESS

SEPP/OT - the software development concept for professionally managed projects using object-oriented design and development technology - is a framework which consists of tools and procedures using pre-defined structures and relationships. For an individual application, it is modified by the user and allows sufficient room for modifications to meet the user's specific project or business needs. SEPP/OT is therefore not an instance of a software development model, but rather provides the foundation for instantiation of a specific software development model.

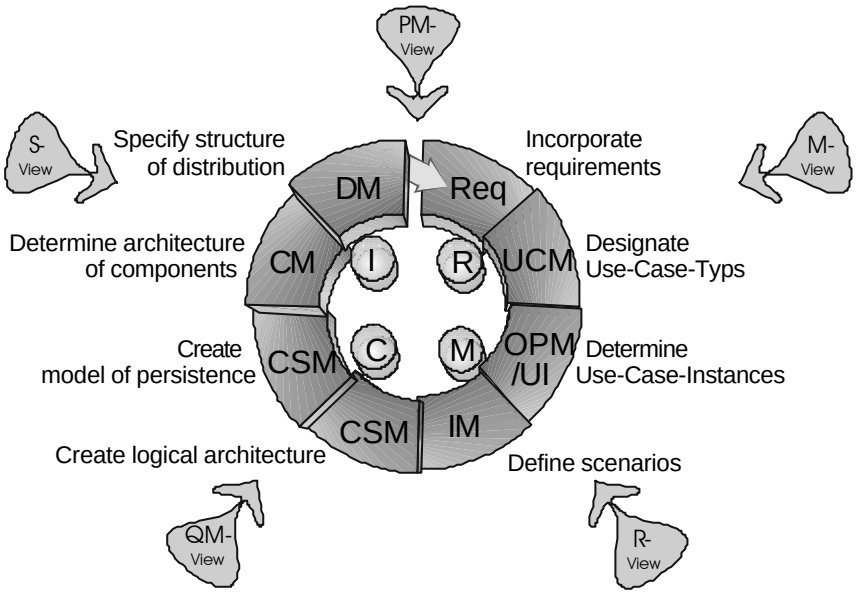

Figure 1: SEPP/OT - Framework

The concept of SEPP/OT can be seen as a spiral staircase in a tower of unknown height which represents the whole software development project (see Figure 1). Each floor contains eight steps of the staircase, which represent the main activities at each stage. These activities are "Specification of Requirements", "Denotation of the UML Use-Case Types", "Instantiation of the Use-Cases", "Definition of Scenarios", "Specification of the Logical Architecture", "Creation of the Persistence Model", "Definition of the Component Architecture" and "Definition of the Deployment Structure".

The eight steps of each floor of the staircase are core activities and assigned to the following relevant models: "Use Case Model", "Object Process Model" (a dynamical diagram as addition to the normal UML diagrams; it is a special model of the OTW®2, see chapter 5), "Interaction Model", "Class Structure Model", "Component Model" and "Deployment

Model". If the modeling process shows that an iteration is necessary, it passes through all eight steps of one floor. This assures that all aspects persist consistent after changes.

The staircase is supported by four columns, the four parallel processes "R", "M", "G" and "I". Thereby "R" stands for the specification of requirements. Paramount in this process is the collection of information from the relevant department regarding its business processes and requirements. But it also includes assistance in the establishment of technical requirements or the definition of Use-Cases.

"M" means the modeling process which covers the analyzing (what must be done) and the design (how should it be done) of an application. The activities performed here occur using the UML standard notation.

"G" represents the code generation process and "I" the implementation of the software itself. Since we intend to put the focus on workflow modeling only the first two columns are relevant to us while the two other columns extend the functionality of SEPP/OT for purposes of software development projects.

These four processes are executed by activities linked by pre- and post-conditions. These activities are represented by the already described steps of the staircase, but these steps are only the superset of activities, in SEPP/OT there are about 150 activities implemented.

From the outside the software development project tower can be viewed from five different perspectives "PM View", "M View", "A View", "QM View" and "S-View". These different point of views cause that from each point of view the different processes described above have different meanings and importance.

"PM View" represents the Project Manager View. The viewpoint of project management is mainly that of the project manager or other persons who are directly or indirectly part of the project management team. This view deals essentially with the management perspective. The PM-View focuses primarily on the requirement specification and implementation processes, whereas modeling and code generation are partially obscured by these first two processes.

"M View" stands for the Methodologist's View which registers which method-specific activities are occurring. This view concentrates on activities which answer the question "How is it done?" and encompasses the majority of the UML related activities. Requirement specification and modeling are at the center of this view, superimposed, but still visible. The code generation and implementation processes are also displayed. However, from this viewpoint all processes converge, because often many processes have to be monitored simultaneously.

"A View" means Adaptability View. This perspective focuses on the reusability and adaptability of the model. All activities observed from this perspective are activities that enhance adaptability or contain subactivities with reusable features. The modeling process is best seen from this perspective, because here one finds the greatest potential for adaptation and also the greatest need for it. This perspective also offers a good view of code generation. Implementation cannot be monitored as readily here, even though experience gained from previous projects can be incorporated throughout.

"QM View" is the Quality Management View. From this perspective quality control issues are in the foreground. The activities seen here are either pure QM activities or are subactivities with QM features. Quality management sees to it that all tasks are planned, implemented, and controlled in accordance with quality demands. It includes the assurance of quality standards in the development process and in the end result. Configuration management is also incorporated at this point. In the QM-View, code generation is in the foreground, but modeling is also clearly visible. Specifying requirements and implementation processes are less visible and only observable in segments.

"S-View" is the Security View. Software security features are in the forefront in this perspective. The visible activities are either security activities or subactivities which contain security features. One sees all processes (requirement specification, modeling, code generation and implementation) clearly. However, implementation is at the center significantly effect the security of the finished program. This also affects modeling, especially when one extensively uses the adaptability feature. The implementation process, which includes user training, should also ensure that future system operators are sensitive to security issues [Burk97], [Muehl98].

The spiral staircase of SEPP/OT is not rigid but can be changed by the user. The user skips activities or adds some as his specific needs require. It is also possible to enter and to leave the staircase on every single step. So the concept of SEPP/OT is flexible enough to be utilized in many different ways.

3. THE OBJECT PROCESS NET (OPN)

The OPN carries on the idea of the Object Process Model (OPM) [Burk94], which was especially developed for software engineering starting by analyzing the dynamic aspects of systems [Schm97]. The goal is the creation of a system model, which will be directly transformed into source code of the destination language [OTW98]. When using the OPN for modeling business processes this feature is not the most important one but the ability of describing the dynamic behavior of a system.

According to the object oriented paradigm the abstract objects represent the classes and the processes represent the methods defined for the classes. The graphical notation of an OPN is a bipartite, directed graph the nodes of which are the objects and processes. The state of the modeled system is described by the values of attributes belonging to the objects. Therefore, a change of these values describes a variation of the systems state. Attribute values can be changed by activating and running processes.

On the one hand, the OPN is considered as an additional diagram within the family of diagrams forming the UML. Therefore it is possible to enhance an OPN by OCL-statements (Object Constraint Language) [UML]. This makes the models more understandable and clear. On the other hand, the OPN can be understood as an object oriented Petri Net. It is possible to transform an OPN into high level Petri Nets conform to the standard [Conc97]. This results in the ability to find analysis techniques directly for the OPN.

Objects are described exclusively as abstract objects or as classes respectively. Hence, the properties modeled within an OPN hold for all instantiated objects of this class and of derived classes. In this way, the inheritance relationships are captured by an OPN.

The graphical representation of an object is a circle, which is divided into three parts. In the upper part the name of the abstract object has to be denoted mandatory. In the lower part a set of attributes can be specified. It is only necessary to declare such attributes which are essential within the context of this OPN, i.e. the same object can be depicted with different subsets of attributes within various OPN belonging to a system of OPN's. According to the object-oriented paradigm the attributes are instances of four pre-defined color classes or of a user-defined structured data type, which is a class of itself. These color classes are:

ENUM – comparable to an enumeration type. The finite set of values has to be defined by enumeration of all elements.

$$< attr > = ENUM\ \{value\_1, value\_2, \ldots, value\_n\}, n \in N$$
$$\text{with } \mathrm{val}(\mathrm{attr}_{\mathrm{ENUM}}) = \mathrm{value}_i \vee \mathrm{undef}\ ; \quad \mathrm{value}_i \in \mathrm{ENUM}$$

INT – comparable to integers in programming languages.

$$val(attr_{INT}) = n \in N \vee undef$$

SET – comparable to a container class, which can only contain one copy of each element (or comparable to a mathematical set)

$$val\left(attr_{SET(ENUM)}\right) = VAL \vee undef, \quad with \quad VAL \subseteq ENUM$$

The value of a SET-attribute is a subset of the related ENUM. Defining a SET-attribute includes the declaration of an underlying ENUM as its domain.

MULTISET – comparable to a container class, which can contain more than one copy of each element (or comparable to a mathematical bag)

$$MULTISET: ENUM \rightarrow \mathrm{N}$$

The multiplicity of an element $attr \in$ ENUM within the bag is denoted by *m(attr)*. The value of such an attribute is a bag itself and can be denoted as a weighted sum:

$$val(attr_{MULTISET(ENUM)}) = \sum_{attr \in ENUM} m(attr)\ val(attr)$$

Using these four elementary color classes it is possible to declare structured types, called CLASS.

$$CLASS = (e_1, e_2, \ldots, e_k) \quad with\ e_i \in (ENUM, INT, SET, MULTISET)$$

A CLASS-attribute describes the merger of some elements belonging to various elementary color classes and can be called by the name of the attribute. However, access to the individual elements is possible too. The value of such a CLASS-attribute is the combination of the individual element values.

$$val(attr_{CLASS}) = (val(e_1), val(e_2), \ldots, val(e_k)) \quad and \quad val(attr_{CLASS}.e_i) = val(e_i)$$

Every abstract object possesses an instance list, which can be considered as a special attribute and which contains all the instantiated objects. This list is a finite set of object identifiers, whereas each of them refers one-to-one to an instantiated object. This list can – particularly in the initial state – be empty. Every instantiated object possesses all the attributes, which are defined for the abstract object to which it belongs. That is, the instance list of abstract objects having derived objects is the union of all instance lists of the derived abstract objects.

The middle part of a pattern for an object can be used for showing role-associations between various objects. This is a useful feature for modeling.

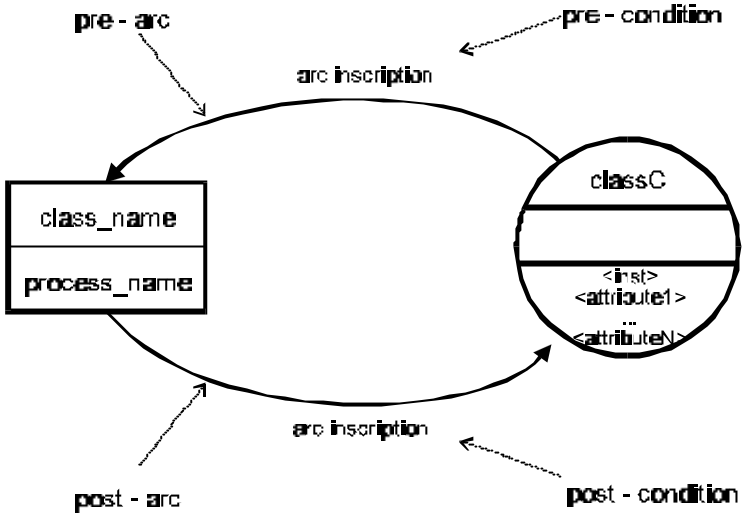

Figure 2 : Graph of an OPN

Processes represent the instances of methods of single classes and describe the dynamic behavior of objects. When they are activated they can change the values of attributes of the corresponding objects and thus, depict state modifications within the system. A process always belongs to a class and can only be activated and run with objects of this class. For objects of other classes processes may only cause changes to the attributes by sending messages. Hereby, it is ensured that an object can only call a different object or communicate with it out of a method.

The graphical representation of a process in an OPN is a divided rectangle. In the upper region the name of the class to which the process belongs has to be inscribed and in the lower part the name of the process itself is denoted. None of these inscriptions can be omitted. A process can be drawn in a system of OPN's only once (in opposition to objects, which can appear arbitrarily often).

A process can be activated only under certain circumstances: preconditions determine the individual system state, which is necessary for starting the process. Therefore, preconditions check certain values of the attributes of objects. Postconditions determine the state of the system after termination of the process. They usually change the attribute's values, i.e. they describe the changes of attribute values by running a process. Before starting a process it has to be checked whether these changes are possible.

Arcs connect objects and processes and vice versa, but never net elements of the same type. Pre-and postconditions of processes are represented as arc inscriptions. Arc inscriptions are logical expressions built with attributes of related objects using suited operators. These expressions result in Boolean values. A condition is fulfilled if the related term has the value TRUE. Related to their directions arcs denote pre- or postconditions of processes: an arc directed from an object to a process is called pre-arc, an arc from a process to an object is a post-arc.

Processes can subjoin new objects to the list of instances of an object via post-conditions (create functionality). It is possible to instantiate new objects from the class to which the process belongs or from other classes. The values of the attributes of this instantiated objects can be undefined or have defaults or certain values. According to the create mechanism there exists a destroy functionality, i.e. instantiated objects can be destroyed. Furthermore, objects can be displaced by processes within the static class hierarchy.

Processes can carry a priority in order to solve conflicts during simulation.

Processes can be refined. The refinement of a process is an OPN as well, which is called subnet. Using this mechanism of hierarchy realizes inheritance and polymorphism within the system model. The preconditions of a process having a refinement will be inherited to all the incoming processes on the refined level. Subnets cannot be connected directly. Only indirect links between different subnets are possible because processes belonging to different subnets can affect the same object.

Arc inscriptions are terms built from attributes by using the defined set of operations for each color class. According to the actual value of the appropriate attributes the term results in a Boolean value. If the value of a term is TRUE, than the related condition is fulfilled.

There are two types of operations/operators which are defined for the several color classes. Detailed description of each operation is not possible here because of the limited scope of this article.

1. *Value-changing operations* modify the values of an attribute. The resulting value belongs to the same color class as the attribute.

$$op: C_i \times C_i \rightarrow C_i \quad \textbf{or} \quad op: C_i \times N \rightarrow C_i$$
$$with \quad C_i \in (ENUM, INT, SET, MULTISET)$$

Examples for such operations are increment or decrement of INT-attributes.

These operations are mostly used for building post-conditions.

2. *Testing operations* check the value of an attribute and result in a Boolean value.

$$op: \; C_i \times C_i \rightarrow BOOLEAN$$

Typical examples of such operations are tests for equality or inequality.

Testing operations are used within preconditions exclusively.

The set of terms which can be pre- or postconditions is denoted by TERM and defined inductively as follows:

1. $< attr > \in TERM \quad with \quad val(TERM) = \begin{cases} false & val(attr) = undef \\ true & otherwise \end{cases}$

2. $T_1, T_2 \in TERM \rightarrow T_1 \,\&\,\& T_2 \in TERM \;,\; T_1 \parallel T_2 \in TERM$

$\&\&$ *means the logical AND*

$\parallel$ *means the logical OR*

Simulating an OPN is comparable to playing the token game in Petri Nets: if all the pre- and post-conditions of a process are fulfilled it will run. All the objects contained in the instance list of an object can be considered like the structured tokens in Petri Nets. Therefore, a process can be activated several times if there is more than one object fulfilling the conditions.

If conflicts occur they can be solved by interpreting the priorities given to the processes.

4. SEPP/OT FOR THE MODELING OF WORKFLOWS

4.1 Adaption to Modeling of Workflows

Regarding the steps of the spiral staircase in detail one can notice that for the modeling of workflows not all the steps of the staircase are important. The adaption of SEPP/OT to workflow modeling is therefore in general nothing but a more detailed description of the steps three to five. However, including the main objective of a particular project, the weighting of the steps may change and other steps of the spiral staircase may become more relevant than.

With respect to our four main processes depicted by the four columns in the center of the spiral staircase for the purpose of workflow modeling only the two processes "Specification of Requirements" and "Modeling" need to be regarded more specifically. The other two processes are irrelevant here, because they deal with the implementation of the software itself.

Figure 3 illustrates our method for creating business process models. The spiral arrow symbolizes that the modeling process is iterative and incremental. During the cyclic passing through the several phases the model will be completed and enhanced.

The "Specification of Requirements" stands at the very beginning of a project about workflow modeling. The methods used in this part of the project are informal ones. The most important part at this time is the discussion with experts to learn as much as possible about the workflow to be modeled. To sort this information and to get a straight transition to object oriented modeling, the Use Case Diagrams of the UML are recommended. This diagrams are very well suited for the comprehension of the functionality of the system. The aim is to record the outside view of an enterprise as well as a clear interface between the enterprise and the outside world.

Starting from the use case diagrams there are some possibilities for upgrading the model. The use case model itself can be refined during the next modeling steps as well. However, in most cases it only results in a first informal approach to the workflow model. The dynamic behavior is represented in more detail in activity diagrams or in OPN's. Activity diagrams are well suited for describing chronological and logical order of business activities. In [MUEHL98] a set of modeling rules for creating workflow models is established and a method for checking activity diagrams is given to ascertain that they comply with the rules.

Within the Class Diagram the structural makeups of an enterprise can be specified. Additionally, this diagram can be used for declaring the attributes of the objects and the role associations between objects.

The application of the OPN for describing the dynamic aspects was circumstantiated in detail above. Especially, it should be used for verifying the model by simulation in each modeling phase.

4.2 Meta-Model for business process modeling

In connection with SEPP/OT a meta-model was developed which gives a recommendation for profoundly structuring the problem of modeling. It can be seen in the following figure 3.

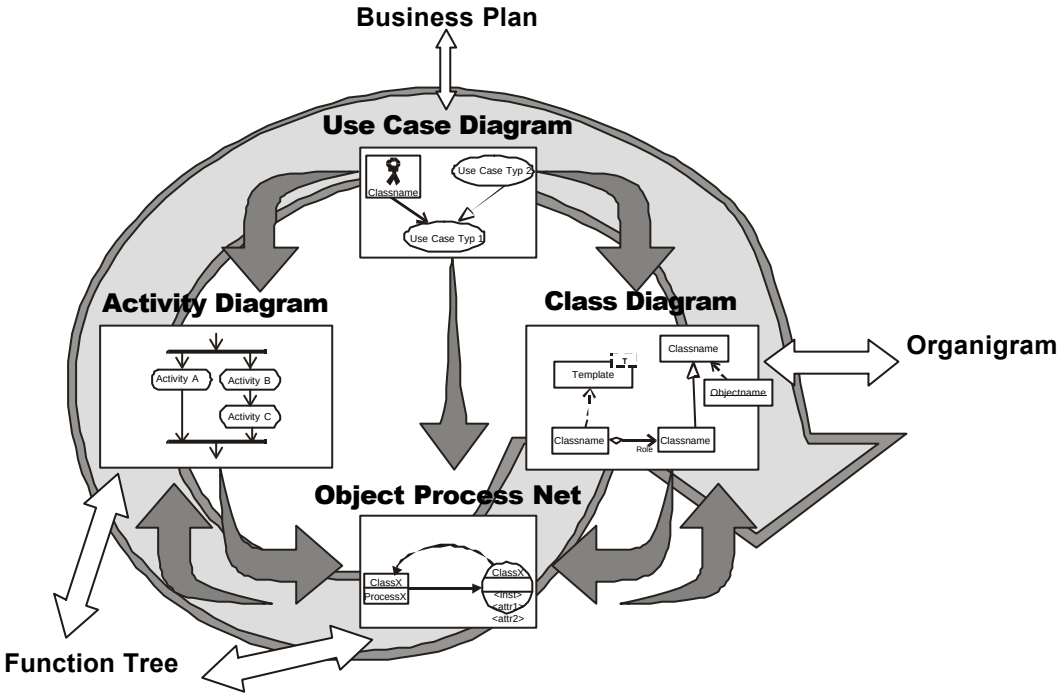

Figure 4: Connection between several diagrams

According to the figure, a business process consists of a set of business activities of different levels of detail.

A business activity has a well-defined result. It uses resources and/or information and changes the state of business objects. An activity itself can consist of several activities or partial activities. For performing an activity an organizational field of work is responsible. The structure of an activity, i.e. how it is composed from partial activities, can often be found in companies as classical function-trees. Such functional descriptions of activities are useful for modeling. Within such a functional structure of activities, there are no chronological or logical relations between the individual partial activities.

Business objects are objects of the business reality which are to be modeled. They are changed and/or used with the help of the activities.

Fields of work are those organizational units which are responsible for carrying out an activity and are occupied by persons. Hierarchical relations between different fields of work can be shown by a class diagram. An instance of such an organizational class diagram shows the organizational structure of a company.

Resources are such things which are necessary for fulfilling the business tasks. It depends on the level of detail of the model to be created to what extent these resources are included in the model.

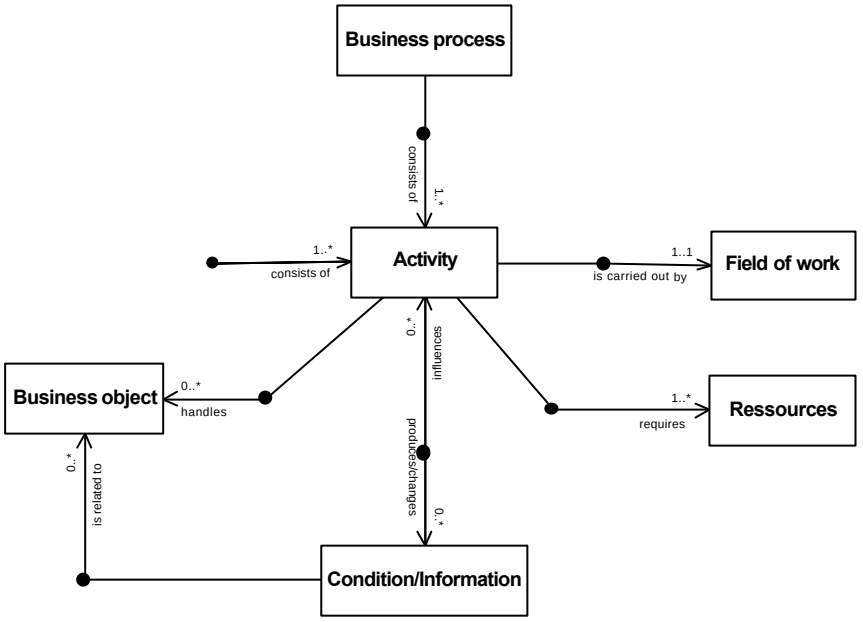

Figure 5 : Meta-model for business processes

Conditions and information are related to the state of the business objects and, therefore, they form a frame for performing individual business activities.

The identification of individual elements of the meta-model during the modeling process helps creating several diagrams. The following table shows where the elements of the meta-model can be found in the UML-diagrams.

Business reality	**Representation within the model**
Business process	Some diagrams forming a model: Use case diagram. Activity diagram and OPN describe the dynamic behavior. Static structure can be found within the class diagram.
Activity	Use case in use case diagram Activity in activity diagram Process in OPN Method in class diagram
Business objects	Objects in OPN Objects/classes in class diagram
Ressources	Objects in OPN Objects/classes in class diagramm

Field of work	Actor in the use case diagram Objects in OPN Objects/classes or roles in class diagram
Condition/Information	Conditions in activity diagram Pre- or post-conditions in OPN

5. TOOL SUPPORT

If you model real-world problems you cannot do this by the only use of pencil and paper. Such models will reach a complexity which can only be handled by a software tool. The main idea of UML-based modeling is to discuss a system from various points of view by using different types of diagram which requires permanent checks of consistency between the several diagrams forming a system's model. For instance, if a new attribute for an object in the OPN is introduced, this attribute will have to be visible in the class diagram as an attribute of the related class, too. For an automated code generation the use of an appropriate software tool is essential.

Another important point concerning the creation of models is documentation. Re-Engineering and re-use of models can only be done if a good description of the models is available. Modern tools should support the documentation process by using templates.

Utilization of stereotypes can increase the clarity of diagrams and explain the modeling intention. But the use of stereotypes without tool support makes no sense.

A feature of modern modeling tools is their ability to work with design patterns. Design patterns can be used to create reference models or meta-models. This modern modeling technique is well suited for building reusable models.

The Object Technology Workbench OTW®2 from OWiS Software GmbH, Ilmenau is a tool which offers most of the UML diagrams and supports the software development concept SEPP/OT described above. The OTW®2 was developed in close cooperation between OWiS Software GmbH and the Technical University of Ilmenau.

In addition to the UML-diagrams the OPN is supported by the OTW®2 which can be transformed into High Level Petri Nets and than simulation can be done. Core of the OTW®2 is an object-oriented repository which contains all information about the diagrams forming a model. Using the OTW®2 generation of documentation can be done automatically based on templates for documentation. Furthermore, the OTW®2 was the first tool supporting comfortable work with design patterns. It is possible to create new models by varying the parameters of former modeled patterns. A pattern can be instantiated again and again.

In addition to the modeling features the OTW®2 includes some tools for model checking and code generation. The software architecture of OTW®2 enables everybody to create new tools using the information from the repository.

6. SUMMARY

The UML is the first promising approach for standardization in the field of object oriented analysis techniques. The given diagrams permit the construction of system models in an incremental way, whereas the level of formalization can be staggered according to the progress of the modeling process. With the OPN an additional means of description for the dynamic aspects of a system is provided expanding the facilities for simulation and for model verification.

In onward development steps of our modeling technique a meta model for business process modeling was created which also further simplifies the work with the OPN for the inexperienced user. So far, the OPN supports the capture and check-up of the logical flow structures. The expansion to a business process cost accounting tool is conceivable.

For further improving the modeling power of the OPN especially for the use in business process modeling we will consider timed OPN; i.e. the processes will get an additional time condition. But this influences the possibilities of formal analysis. Without time conditions for processes the OPN can be transformed into high-level Petri Nets, which are conform with the standard [Conc97]. Furthermore, we are directed towards a tool-supported, automated transformation of OPN into high level Petri Nets in order to achieve the opportunity to apply formal analysis techniques.

The main advantage of the SEPP/OT concept is the ability to work with design patterns. This gives the possibility to work with reference models and therefore to instantiate these patterns again and again. This reduces cost and effort of modeling in each new software development project enormously.

The SEPP/OT concept appears to be better suited for modeling workflows than the "Unified Process" which uses the workflow concept to organize software development projects. SEPP/OT is more directed towards the modeling process than "Unified Process". With its different views on the system from the outside it is very convenient for different types of users like managers and developers. Therefore, it can be used for both: the organization of software projects and the modeling of workflows.

LITERATURE

[Burk94] Burkhardt, R.: **Modellierung dynamischer Aspekte mit dem Objekt-Prozeß-Modell**, Dissertation, 1994, TU Ilmenau, Ilmenau, Germany.

[Burk97] Burkhardt, R.: **Die Unified Modeling Language, Objektorientierte Modellierung für die Praxis**, 1997, Addison-Wesley, Bonn, Germany.

[Muehl98] Mühlpfordt, A.: **Objektorientierte Geschäftsprozeßmodellierung auf der Basis der Unified Modeling Language**, Dissertation, handed in at the TU Ilmenau 1998. (Manuscript).

[OTW98] **Objekttechnologie-Werkbank OTW® 2, Modellierungswerkzeug zur Modellierung mit der UML**, Handbuch, OWiS Software GmbH, 1998 (www.otw.de) (in German, to appear in English soon).

[RUP] http://www.rational.com/products/rup/prodinfo/index.jtmpl

[Möld96] Mölders, A., Fengler, W., Burkhardt, R., Philippow, I.: **Workflow Configuration Using the Object Process**, Proceedings of the 1. International Conference PAKM, Basel, Oktober 1996.

[PMK97] Petersen, U., Mölders, A., Köhler, T.: **Optimierung von Geschäftsprozessen durch objektorientierte Modellierung und Simulation**, Proceedings of "Workflow-Management in Geschäftsprozessen im Trend 2000", Schmalkalden, 15./16. Oct. 1997.

[Schm97] Schmutterer, A.: **Theoretische Formalisierungen verschiedener objektorientierter Workflow-Modellierungstechniken**, Diplomarbeit, 1997, TU Ilmenau, Ilmenau, Germany.

[Conc97] **High-level Petri Nets – Concepts, Definitions and Graphical Notation**, October 2, 1997, Committee Draft ISO7IEC 15909, Version 3.4.