Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

AN INTEGRATED OBJECT MODEL FOR ACTIVITY NETWORK BASED SIMULATION

Gert Zülch
Jörg Fischer

ifab-Institute of Human and Industrial Engineering
University of Karlsruhe
D-76128 Karlsruhe, Kaiserstrasse 12, GERMANY

Uwe Jonsson

AXION GmbH
D-50968 Köln, Goltsteinstraße 89, GERMANY.

ABSTRACT

This paper describes an object-orientated simulation approach towards an integrated planning of production systems. The main obstacle for an integrated use of simulation over different planning areas and stages are the different views on a production system. Therefore, an object model is developed, which enables the co-existence of different views and levels of detail in the same simulation model while maintaining its consistency. This is achieved by combining object-orientated technology with a network based simulation approach. The prevailing idea is to offer the opportunity to re-use existing models for the investigation of different aspects of a production system. The approach is abstractly described as a conceptual object model and is thus, independent from a concrete simulation language, tool or environment. The last part of this paper introduces the simulation tool *OSim*, that implements this object model and demonstrates its usage through an example.

1 RESERVATIONS ON APPLYING SIMULATION IN PRACTICE

For the design and control of production systems, simulation has proven to be a powerful tool. However, investigations of the market situation have shown that many companies are not willing to use simulation as a permanent planning tool (Schmittbetz 1998). One reason for this is the expenditure associated with modeling a production system (Rabe 1999). The problem is that most simulation tools which are available on the market today, do not support the re-use or exchange of models e.g. exploiting them for different planning aspects. Thus, modeling has to be done for every field of application, each time from scratch.

Following Rabe (1999) it is in fact the openness of the interface that helped many software tools to establish themselves in practical use. In order to realize the exchange and the re-use of models there are two fundamental possibilities. Either one defines a standard interface between different tools or one combines the different views of a production system, e.g. the view of material flow, the view of manufacturing or the view of personnel planning in one simulation application.

During the development of such a simulation concept the problem of semantic compatibility is likely to occur. Each view of a production system shows different parts of it (Zülch and Brinkmeier 1998). If one wants to combine these views, the generic model, on which the simulation application is based, has to determine the main elements of all these views.

If an additional task of a simulation tool is to reach a flexibility similar to that of other planning tools, e.g. operation plans or plain text descriptions, the modeling methodology has to support a step by step development of the simulation models. In this case, it should be possible to simulate the model has to work in every development phase. Thus, the simulation concept has to support various levels of detail and also the addition and removal of model parts.

2 THE IDEA OF AN INTEGRATED OBJECT MODEL OF A PRODUCTION SYSTEM

In order to take a step towards a solution of the described problems, an object-orientated simulation concept is under development at the ifab-Institute of the University of Karlsruhe. The basic idea is the development of a generic model of a production system including the simulation functionality of the model's elements. In this generic model, different application views shall be combined. Thus, the derived models may be used in different planning steps.

For the development of the basic generic model object-orientated modeling principles were chosen. With the help of the object-orientated technique of specialization the functionality of an object model can be structured hierarchically. In this manner, abstract object classes may be defined with more universal functionality compared to traditional approaches. This universal functionality can be adjusted to the requirements of a specific application field, with the help of specialization.

In this way, common features and even variations in different application views can be brought into a defined context. Separate objects can be combined using the principle of composition. Thus, more complex functionalities can be realized (Jonsson 1998).

Experience has shown that the hard part of representing a production system is to depict the processes and their respective control because these aspects are quite abstract and it is rather difficult to get an intuitive correspondence between model and reality (Rabe 1999). One possibility is to depict the processes by modeling them based on activity networks. This technique is found mainly in business process modeling (Schmid 1998). It makes intuitive handling of the activities together with their releases and dynamics to a great extend possible. For the basic generic model

- object-orientated modeling and draft techniques combined with an
- activity network based simulation approach have been used.

Additionally, the generic object model will realize the following properties:

- The object model should support the co-existence of different views and levels of detail.
- The object model should allow hierarchical structuring of simulation models.
- The step by step modeling of simulation models should be made possible while preserving the consistency of the model.
- The most important monetary and logistical key data should be consistently available throughout all modeling views and hierarchical levels.

3 THE OBJECT MODEL FOR THE SIMULATION OF PRODUCTION SYSTEMS

3.1 Description of the Development Steps

The development of the generic object model was performed in five steps (Figure 1). In every step a new view was grasped and added to the former one. The different views were chosen because they represent the typical planning functions that appear in the planning of production systems.

The process-orientated view looks at the process flow of a company. In this phase, the modeling of activity networks are in the limelight. The resources-orientated view focuses on the behaviour of the modeled resources. One determines whether the resources behave passively towards the modeled processes or intervene actively in the processing. In the fourth step the product- or material-orientated view is grasped. Here, the modeling of the products and their movements are regarded. However, for reasons of simplification, no attempt at inserting a complete material-orientated view into the generic model was made. The last step is characterized by a hierarchical view. It runs vertically to the views that have been named so far. Through this step the four phases that had already been developed were widened by a hierarchical aspect.

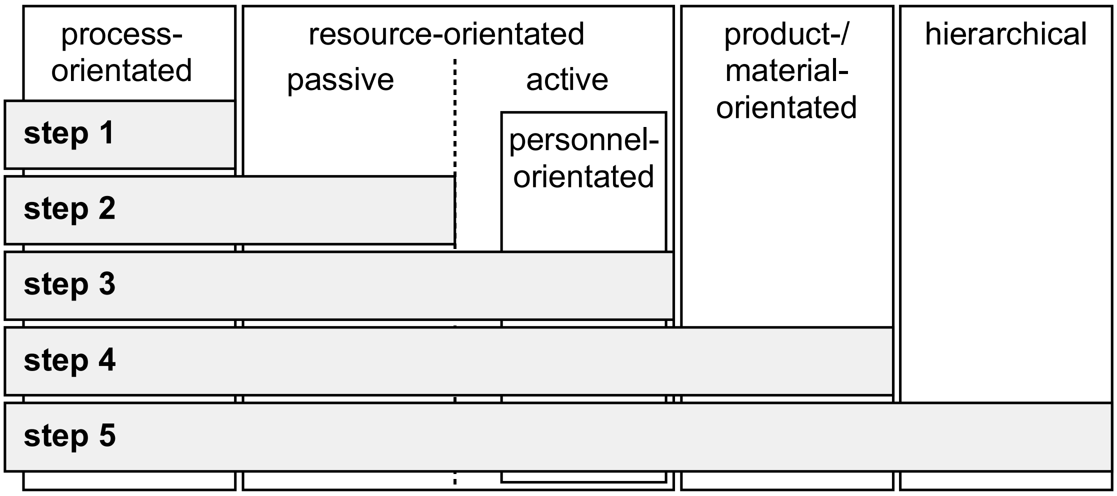

Figure 1: Development Steps of the Object-orientated Model of a Production System (Jonsson 2000)

The single phases are built upon each other and can be combined. In the following description the names of the object classes (*NName*) are given in italics and brackets.

3.2 Step 1: Activity Network-Orientated Modeling

The starting point of the object modeling is the dynamic course of activities in the production system which is modeled by activity networks. Activity networks are directed graphs with a logical sequence of activities for the winding-up of orders. A modeled production system can be seen as a collection of activity networks, in which all kinds of orders appearing in the system are modeled (Brinkmeier 1998; Grobel 1992). Activity networks can be released by external or internal events such as customer orders, servicing orders, or internal requirement orders. In this way, it is also possible to model indirect activities.

The winding-up of the activity networks during the simulation run is realized with the help of so-called process objects (*PProcess*). In the frame of the object model that is introduced, the activity networks form a meta-level. During the simulation run processes are generated or initiated from this meta-level.

This occurs with the help of process objects, which are created from the activity network after its start. In this case, a process object represents the currently treated activity. The triggering of the activity network's processing during the simulation run is done by a special object named *PInit*. This object type represents all kinds of orders in the generic model.

Following the object-orientated design technique, activity networks are modeled as sequences or networks of objects. To enhance the objects with a kind of simulation intelligence exploiting the object methodology, is one of the main ideas behind the concept. This can be subdivided into the following aims:

- The objects will contain a method of controlling the simulation run.
- The objects own methods to evaluate the simulation experiment. In this manner, a consistency concept for the evaluation over different hierarchical levels should be realized.
- The activity network objects own methods of supporting the development of simulation models. This aspect does not cover the described object model, but for the realization of the simulation tool it was most important.

Like every graph an activity network contains nodes and arcs. In the generic object model nodes and arcs are represented by objects. The nodes represent the activities of the production system, whereas arcs represent the interference between the activities. A node is always connected with a previous and a following arc. An arc can have between 0 and n previous and following nodes. In Figure 2 a typical activity network is shown.

With the help of the object-orientated design technique, it is possible to create an object hierarchy. The different object classes of this hierarchy can be inserted in the same activity networks. Thus, the behaviour of the different node types can vary. In the generic model there exists a class of network nodes which represent the operation time as a constant. Another class of network node represents this as a statistical distribution.

The concept of heredity is also used for the hierarchical aspects (cf. chapter 3.6). For example, an activity network object (*PAcnt*) is specialized from an activity network node object (*PAnod*). Thus, an activity network can contain an activity network as a node object. This kind of encapsulated object is possible at all level of recursion (cf. chapter 3.6).

In Figure 2 two further node classes, a re-perform node (*PAnrp*) and an alternative node (*PAnal*), are shown. These nodes are specializations of the base class (*PAnod*). They present the possibilities of modeling alternative paths and representing the repeating of activities.

3.3 Step 2: Resource-Orientated Simulation with Passive Resources

In this step, the generic object model is extended to resources. The behaviour of the resources is, at first, purely a passive one. It is controlled by the assigned activity network objects, mainly the node objects.

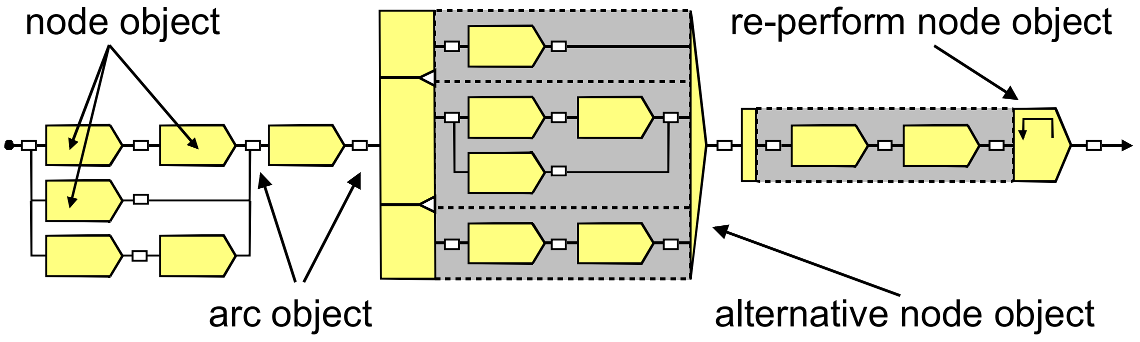

Figure 2: Activity Network

3.3.1 Passive Resources

Resources may take different rolls in the assigned processes. In the generic model there are the states

- to consume,
- to produce,
- to exist, and
- to occupy.

Incoming parts e.g. from an assembly activity are consumed. Components or products are produced. Information forms have to exist. Resources such as machines, employees and tools are occupied. In Figure 3 the concept of the resource integration into the generic model is shown.

In the generic model the relationship between the resources and the activities is assigned with the help of association objects (*PAssc*). The association object determines the role of a resource towards an activity network node during the simulation run. Thus, for every role a resource can be assigned using an association object which has to be implemented. Any association object can be assigned to an activity network node. In turn, resources can be assigned to the association objects.

Figure 3 shows a situation which may occur during the simulation run. Resources (*PResc*) R1, R2 and R3 are assigned to the activity node N1 with the help of the association object A1. A simulation state is shown in which a process object P1 for activity N1 already exists. From the three available resources, resource R1 is chosen. This fact is shown through the existence of the relation object E1 (*PRelt*). This relation object manages the communication between the association object and the chosen resource.

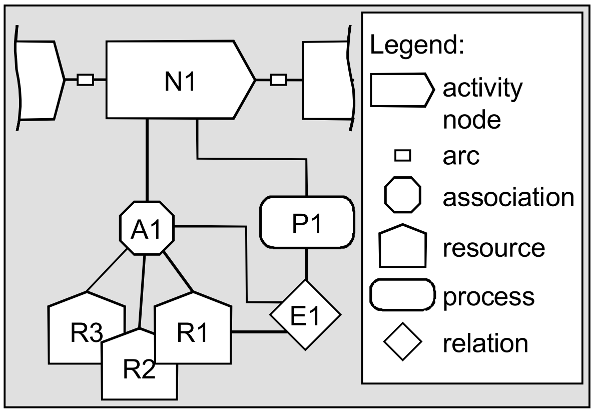

Figure 3: Passive Resources

From the standpoint of an activity network, it does not matter what kind of role a resource takes. Only the association object's answer is important, whether a resource is available or not. If there is no resource available, the process has to wait until one required resource is available.

3.3.2 Processors

The first type of resources are the processors (*PRprc*). As a processor, one can model e.g. operational units, manufacturing tools, computers, workplaces or employees. The important feature of a processor is the fact that it contains a state, whether it is occupied or not. If a processor is in the occupied state during a simulation run it can not be occupied by other processes. In this case, the other process has to choose another resource or has to wait until the resource becomes free.

Processors contain several methods. Typically, calculation of key data e.g. of working to capacity or capacity stock are implemented (Jonsson 2000).

3.3.3 Modeling of Personnel Structures

Only in this step it is possible to model personnel structures. The principle of this feature is to subdivide the dimensions activity, personnel type and operational unit.

A personnel structure can be modeled by assigning a definite person to a definite operational unit and to an activity (Zülch, Jonsson and Rinn 1998). From the standpoint of the object model there are two resources (personnel type/operational unit) which can only be assigned in pairs.

In order to achieve this, an object class for personal types (*PPers*) and an object class for operational units (*POpun*) are specialized from the class *PRprc*, without any special behaviour. The missing link is an association object class to assign tuples of personnel and operational units to an activity network node. In order to achieve this, an association object class (*PApon*) has been created to assign such tuple elements. During the simulation run an inquiry to the association whether a resource is free or not is in the case of *PApon* only returned as true if one of the resource tuples is free (Jonsson 2000).

3.3.4 Consumable Resources

Another important type of resources is that which are consumed or produced by the processes. An example of resources with the role "produce" or "consume" is the material resources in manufacturing.

A consumable resource (*PRcon*) is specialized from the base class of resources (*PResc*). It contain an attribute for its current amount and its own methodology to check the amount, to increase or decrease it and to calculate relevant key data.

For the definition of an abstract interface between consumable resources and activities, a special association class (*PAcon*) for consumable resources is derived from the base association class (*PAres*). From this class four association classes for the roles of the consumable resources in face of the activity network nodes are specialized. There are three, the roles produce (*PAprd*), consume

(*PACon*) and one to check the existence of a resource (*PAqch*). These are completed by a fourth association that represents a gozinto list (*PAgol*), which is a special kind of a bill of material. With the help of an association object of type *PAprd* the amount of the assigned resource is increased after finishing a connected process. By using an association object of type *Pacon,* the consumed amount of the assigned resource is decreased. If there are not enough parts available, the process has to wait. In the case of association *Paqch,* only the existence of a part is checked.

With the fourth type of association it is possible to realize a bill of material. It contains a methodology to collect association objects of the type *PACon*. In Figure 4 the activity N1 represents an assembly of a component. The association on the left side is a gozinto list (*PAgol*). With the help of the association objects of type *PAcon*, which is contained in this list, the amounts in which the parts P11, P12 and P32 go into the assembly activity are fixed. The produced component P12 is specified with the help of an association object of type *PAprd*.

3.4 Step 3: Object Model for Simulation of Active Resources

So far, the behaviour of the resources in the resource model is generally a passive one. The execution of the processes and the occupation of resources is initiated exclusively by the activity networks.

In addition to passive resources, there are, in reality, structures with actively acting resources, so-called actors (Alhir 1998, p. 160) e.g. persons or equipment controlled by persons. These actors choose one out of a set of activities on their own. In this case, the decision which one to choose, is based on the current situation, the available information or from personal preferences.

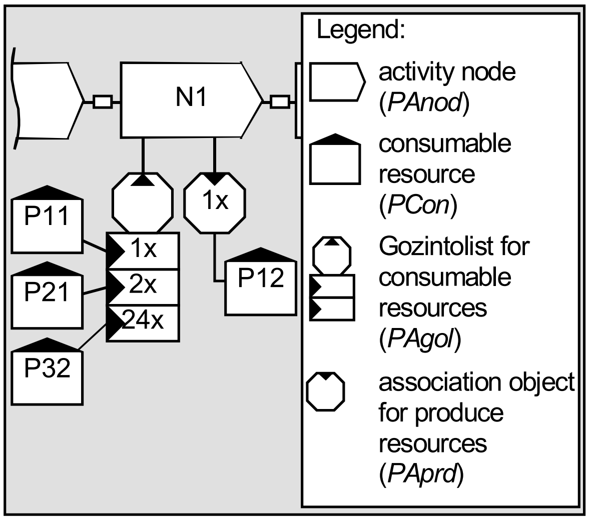
Legend:
activity node (PAnod)
consumable resource (PCon)
Gozintolist for consumable resources (PAgol)
association object for produce resources (PAprd)
N1
1x
P11
P21
P32
P12
1x
2x
24x

Figure 4: Example for Modeling Consumable Resources

To insert active resources into the generic object model, different object classes have been developed. These classes will be described in this chapter. After this description, the interaction of the active objects will be shown.

3.4.1 Process Stores

Whenever an instance of a process object is created during the simulation run, it has to be defined where the object is to wait until its processing. So far, the process objects have usually been added to a central queuing line.

In order to simulate active resources it is necessary to model explicit locations for the waiting process objects. Thus, the active resources can be assigned to process stores, from were they will be processed. For this, a process buffer object class (*PStore*) has been created. The main task of this object is of course to offer a method for the storage of processes during the simulation run.

3.4.2 Actors

During the design of the active resource concept one handicap was the smooth integration into the basic generic object model. Because of this, an approach for the development of the actors, which made the use of a resource as a passive or active entity possible, was chosen. It is realized as an interface class named *PActor* for all actors.

The processor class *PRprc* described in chapter 3.3.1, is now inherited from the interface class *PActor* with the help of the multiple specialization concept of the object-orientated design method. Hence, the class *PRprc* can be used both as an actor and as a passive resource. An actor is notified at the moment a new process enters one of its assigned process stores. As a result, it can independently start winding-up its processes.

If the processing is successful, the active resource takes another process out of its assigned process stores to wind-up. It decides on its own in which kind of sequence it will wind-up its processes. In the following, a description of the interaction between the described objects is given.

In Figure 5 a possible situation of resources is shown. The association objects A1 and two process stores P1 and P2 are assigned to the activity node N1. In these process stores the processes created from node N1 may be added. The actors AR1 and AR2 take processes from process store P1 to perform. Actor AR3 only takes processes from process store P2. The passive resources R1, R2 and R3 are assigned, with the help of the association object A2, to the node N1.

If the simulation run arrives at node N1, this activity creates a process object. The special association object A1 inserts the process object into one of the process stores, P1 or P2. The standard implementation of the association object A1 always inserts a process into the least occupied store. After the process is inserted, all actors will be notified. If an actor is free, it tries to start performing the process. In the case of Figure 5, this is only possible if one of the resources R1, R2 or R3 is free.

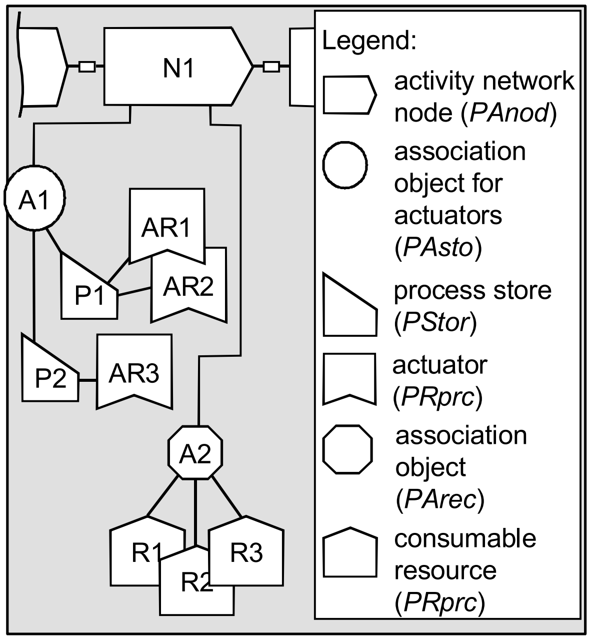
N1
A1
AR1
P1
AR2
P2
AR3
A2
R1
R2
R3
Legend:
activity network node (*PAnod*)
association object for actuators (*PAsto*)
process store (*PStor*)
actuator (*PRprc*)
association object (*PArec*)
consumable resource (*PRprc*)

Figure 5: Simulation of Active Resources

3.5 Step 4: Process Entities

In development step 4, the generic object model has been extended to a material-orientated view. This is, in general, possible with the help of consumable resource objects, but there is no entity which could represent the material during the entire processing of an activity network. That is the reason why the idea of a general entity, which accompanies the process flow from start to finish, has been added to the concept. During a simulation run such an entity may assume different states, collects information, represents the material or influences the alternative path within the activity network.

Other kinds of processing facts such as information or order papers, can be modeled with the help of these entities. In order to represent an entity, the object class *PEnty* has been defined. This object class exists in different forms of specialization of the generic model.

A process entity is created during the simulation run at the release time of an activity network. The activity network assigns an entity to each related process object and network node which is in process.

In order to integrate the concept of process entities, it is necessary to extend the dynamic model of simulation with a passing mechanism for entities. In Figure 6 the passing of a process entity along an activity network in different phases is shown. At the beginning of the simulation run the entity is assigned to an activity network trigger of type *PInit*. The entity is then passed to the first activity network arc. This arc passes it further to the first node.

The process of the second node, in position 2, is already finished. The process object is destroyed and only a connection between the node and the entity object is left.

Because of the branch in the activity network at position 3 the process entity is passed onto both network nodes simultaneously. In this case, it is assumed that both branches of the activity network can be processed simultaneously.

In the final position the process entity is shown assigned to the last activity node. The process object has already been created and the process is ready to be performed. At the end of processing the entity will not be destroyed, as it is the standard in the case of a process. It

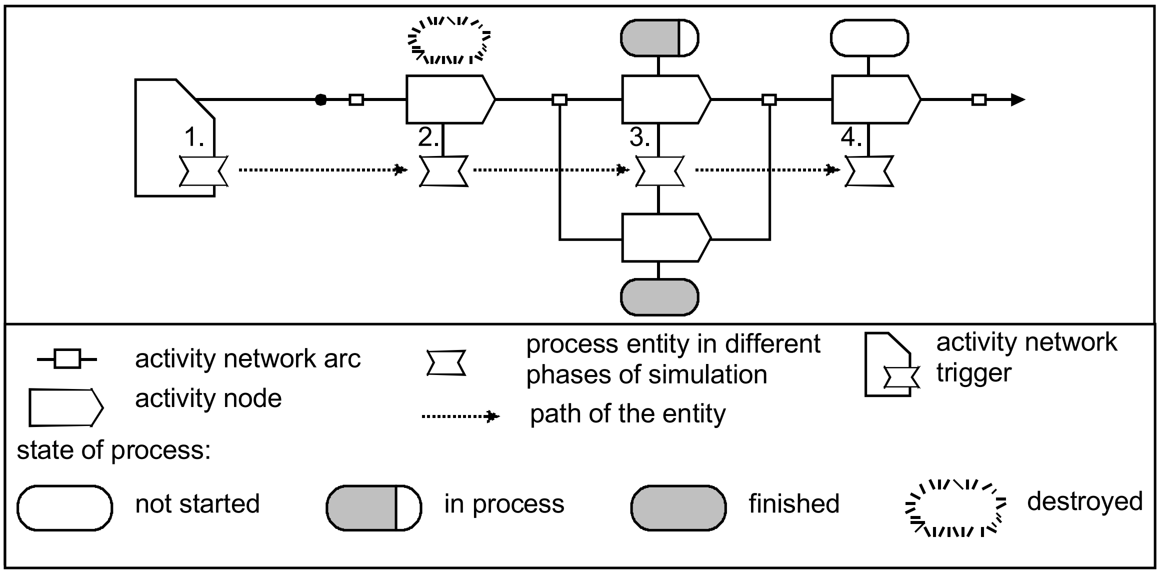

Figure 6: Simulation of an Activity Network with Related Process Entity

remains assigned to the trigger object and can be used for later evaluation of the simulation run.

3.6 Step 5: Adding Hierarchical Aspects

In the last development step of the generic model, hierarchical aspects were added. The hierarchical modeling should not only be limited to self-sufficient subsystems at this phase. Instead, it is the intention of the concept to incorporate structural information as well.

In the case of step by step modeling, the refining or enhancing of the model must be supported. The addition of hierarchical aspects is divided into three different steps, namely adding hierarchical aspects of

- the resources,
- the processes, and
- the evaluation.

3.6.1 Hierarchical Aspects of Processes and Resources

Hierarchical aspects of resources and processing are realized by using the object-orientated technique of specialization. Out of the object class consumable resources (*PRprc*) as well as the object class activity node (*PAnod*) and collection classes, that inherit all characteristics of their base classes were specialized. An additional method for these collection classes is to provide collections of the objects of their base classes (Jonsson 2000).

This kind of functionality shall be clarified in Figure 7. On the left hand side a hierarchical activity network is shown. The top level is the main activity network. Within, there are the nodes N1, N2 and N3. N2 is an activity network object of type (*PAcnt*); which is a specialization of the node object class (*PAnod*). Thus, it is able to contain activity network nodes and to be inserted into another activity network (cf. chapter 3.2).

This fact is similar for the resources. On the top level there is the production system. On the lower level one can see different production segments, modeled with the help of resource collection objects. Thus, production segment R2 is further subdivided.

On the different levels the activity nodes are connected to the related resources with the help of association objects. The association objects on the different levels are completely independent. During the simulation run each association object independently chooses a resource. Thus, A2 does not replace the lower level association objects A4 and A7. It merely models a resource relation on a higher level. This relation is simulated in parallel to the relations on the lower levels.

It is interesting that there is only a slight increase in effort for communication to enable hierarchical simulation. The simulation dynamics can be transferred with almost no change to the simulation dynamics (Jonsson 2000).

3.6.2 Hierarchical Evaluation

One of the most important problems with hierarchical simulation approaches is the concept of evaluation. In this case, the consistency of calculating key data has to be ensured and available at all hierarchical levels.

To simulate on different hierarchical levels it has to be examined whether or not existing evaluation concepts are still valid. In order to ensure validity, the reports recorded during the simulation run and the methods for calculation of key data are designed in a way that considers the hierarchical structure.

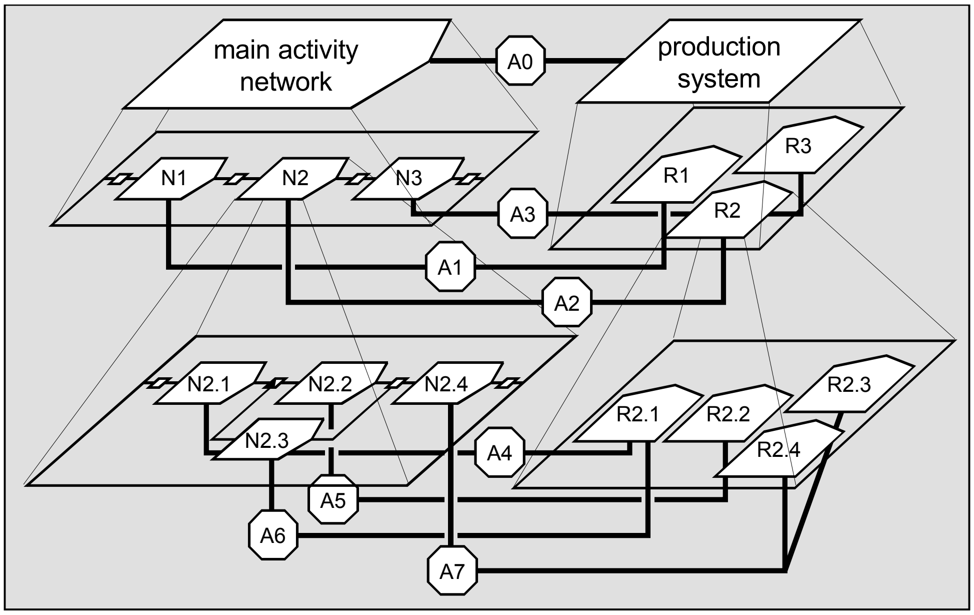

Figure 7: Example of a Hierarchical Simulation Model

Hence, the algorithms for calculating key data can be subdivided into local or delegated calculations. In the case of local calculation, the particular object is able to calculate the key figure only with the help of its attributes and recorded simulation reports. Such key figures are completely independent of the hierarchical structure.

An example of such a key figure is the simulated average lead time. The calculation of the average lead time is based only on the starting and finishing time of the process belonging to the activity network node, which incorporates a method to calculate this lead time. Therefore, it is independent of any hierarchical structure.

The minimal lead time is an example of a delegated or polymorphic calculation of a key figure. A higher level entity, e.g. an activity network, calculates its critical path with the help of the work content of its activity nodes. Methods for all delegated key figures have to be implemented for all new object classes which are to deliver this key figure.

4 EXAMPLE MODEL OF A GEAR BOX PRODUCTION

After describing the conceptual object model, the simulation tool ***OSim***, that implements this meta model is demonstrated. In general, any programmable object-orientated simulation tool is suitable as an implementation framework for the object model. However, in order to implement all object and network based concepts in a concise manner and to make these concepts directly available to the user via an object-orientated user interface, the simulation tool ***OSim*** (Object-orientated Simulator) has been newly developed.

The data for the explained model is taken from the Special Research Centre 346 at the University of Karlsruhe. This interdisciplinary institution is designated to "Computer Integrated Design and Manufacturing of Parts" and sponsored by the German Research Association. The model deals with the manufacturing of gear boxes and their respective sales and design activities. It is only a simple model and the given example does not claim to be a full simulation investigation.

In Figure 8 the model is shown. On the left hand side the activities for design and sales (a) are shown and the activities for the manufacturing (b) on the right hand side. The model of sales/design contains in the alternative node PA1 the possibility for a new or adapted design. The structure of the alternative activity networks are mostly the same, but the operation times are different. Alternative node PA2 features a probability branch. Either the order is given (80% possibility) or it is not given (20% possibility).

Manufacturing is modeled within the alternative node PA3. There are three possible alternative paths due to different possible operational unit combinations which offer different capacities and qualities.

Next, passive resources are added (c). In the sales and design part the personnel resources are the released elements, and in the manufacturing part they are the machinery equipment. The modeled resources dominate the operation times in each case. In the shown state of the model, only passive resources are used.

During the simulation run, an entity object accompanies the processes. At alternative node PA3 it intervenes in the simulation process. At this point, it decides which alternative path the order has to take. In the described

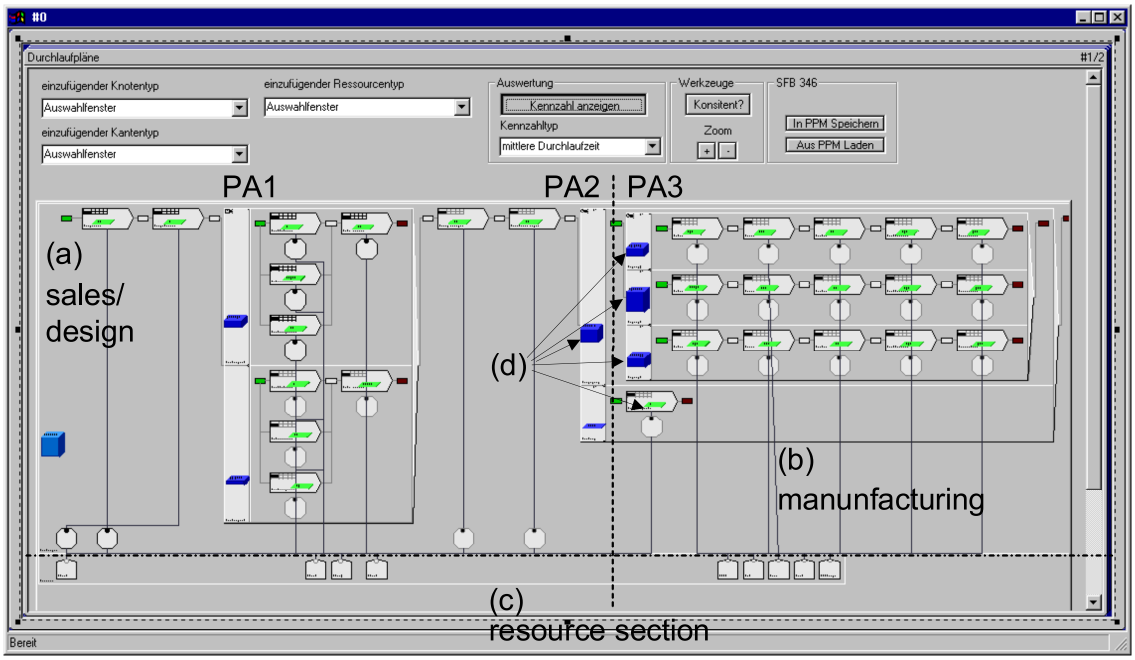

Figure 8: Modeling Tool of *OSim*

simulation run the entity object decides, first considering the available capacity at first. If this decision offers two different paths, it decides based on the quality level which an alternative path offers.

In Figure 8 the activity network is shown after the simulation run. At the node objects there are activated bars that portray the activity related average lead times (d).

In Figure 9 (a) the average lead time of the main activity network is depicted, and in Figure 9 (b) the lead times of the node objects of the main activity network are shown. Here the local aspect of the average lead time can be observed (cf. chapter 3.6.2). The entire result in (a) is not the sum of the node objects lead times in (b). This is due to the fact that the top level activity network calculates its average lead time with the help of its recorded process protocols and not with the help of its respective node objects.

In the given example the model is not very detailed. It does not use the whole functionality e.g. modeling of active resources or differentiated personnel structures.

As a next step, the model can be validated and adjusted using the real system for calibration. In a case where the results are in the same value range, the model can be used for further investigations, if not, the model should be improved, step by step, until the results correspond. The model can be simulated at every state of development.

5 RESULT AND OUTLOOK

The functionality of this object-orientated simulation approach is demonstrated using the model in chapter 4. With the help of object-orientated design methods, the generic model can easily be extended. Thus, it can be adjusted to concrete problems of practice.

As a summery, it can be stated that the combination of activity network based modeling principles and object-orientated design techniques are suitable as a base for a generic simulation model for production systems. With the developed object model the foundations are layed to use simulation as a permanent planning tool.

However, to reach this goal, further concepts are required. If the object model could be integrated into a production data base, it should be possible to configure simulation models out of it, perhaps with the help of drag and drop. Thus, the creation of a simulation model would be much easier than at the present moment.

Further approaches, which are needed to use simulation for production planning and control, are concepts to set-up the simulation model to the current state of a production system. To release this, it should be possible to model the used lead times for real activities, alternative paths and the usage of resources.

With the help of these concepts, simulation models can sufficiently represent the current state of a production system. In this case, the results of the simulation model can also be used as input data for production control.

The described research tasks are now contained in a project of the Special Research Centre 346 at the University of Karlsruhe. This project is currently running at the ifab-Institute of the University of Karlsruhe.

REFERENCES

Alhir, S. S. 1998. *UML in a Nutshell*. Cambridge: O'Reilly.

Brinkmeier, B. 1998. *Prozessorientiertes Prototyping von Organisationsstrukturen im Produktionsbereich*. Aachen: Shaker Verlag. (ifab-Forschungsberichte, vol. 17)

Grobel, T. 1992. *Simulation der Organisation rechnerintegrierter Produktionssysteme*. Karlsruhe Uni: Institut für Arbeitswissenschaft und Betriebsorganisation. (ifab-Forschungsberichte, vol. 3)

Jonsson, U. 1998. Object-Oriented Modeling of Products and Production Systems. In *Design of Organisational Structures, Work Systems and Man-Machine-Interaction*, ed. G. Zülch, 33-45. Aachen: Shaker Verlag. (ifab-Forschungsberichte, vol. 16)

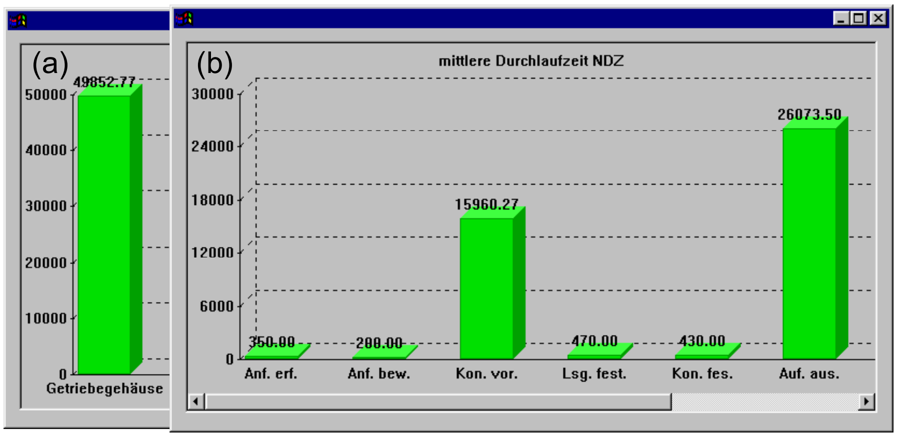

Figure 9: Average Lead Times of the Object Nodes and the Main Activity Network

Jonsson, U. 2000. *Ein integriertes Objektmodell zur durchlaufplanorientierten Simulation von Produktionssystemen*. Aachen: Shaker Verlag. (ifab-Forschungsberichte, vol. 21 - in print)

Rabe, M. 1999. Beginnt ein neues Zeitalter der Simulation? In *Simulation und Visualisierung '99*, ed. O. Deussen, V. Hinz and P. Lorenz, 3-18. San Diego (CA) et al.: The Society for Computer Simulation International.

Schmid, C. 1998. *Beitrag zur Modellierung und Analyse dynamischer Unternehmensmodelle*. Aachen: Shaker Verlag. (Forschungsberichte aus dem Institut für Rechneranwendung in Planung und Konstruktion der Universität Karlsruhe, vol. 98, 1)

Schmittbetz, M. 1998. Simulation wird zum Treibwerk für Innovation. In *VDI-Nachrichten*, 24:18.

Zülch, G., Brinkmeier, B. 1998. Object-Oriented Product/Production Model – Integration Concept and Application in Production Management. In *Strategic Management of the Manufacturing Value Chain*, ed. U.S. Bitici and A.S. Carrie, 577-584. Boston, Dordrecht, London: Kluwer Academic Publishers.

Zülch, G., Jonsson, U., Rinn, A. 1998. INSIGHTS – Integrated simulation game for a comprehensive redesign of production systems. In *Experimental Learning in Production Managment*, ed. R. Smeds and J.O. Riis, 105-117. London et al.: Chapman & Hall.

AUTHOR BIOGRAPHIES

GERT ZÜLCH, born in 1946, studied mechanical engineering at the University of Technology at Brunswick, and Industrial Management at the Superior School of Technology at Aachen, Germany. In 1985, after 10 years of experience in research and industry, he became professor and head of the then newly founded ifab-Institute of Human and Industrial Engineering at the University of Karlsruhe.

JÖRG FISCHER, is a research assistant at the ifab-Institute. He received a Dipl.-Ing. in mechanical engineering in 1999 at the University of Karlsruhe. He is active in researching time discrete simulation methods and object orientated-simulation models.

UWE JONSSON, received a Dipl.-Inform. in computer science at the University of Karlsruhe, in 1993. From 1993 until 1999 he was a research assistant at the ifab-Institute. In 1999 he graduated as Dr.-Ing. Since 1999 he is Senior Consultant at AXION GmbH in Cologne, Germany.