Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

MODELING CONTROL IN MANUFACTURING SIMULATION

Durk-Jouke van der Zee

Production Systems Design Group
Faculty of Management and Organization
University of Groningen
P.O. Box 800
9700 AV, Groningen, THE NETHERLANDS

ABSTRACT

A significant shortcoming of traditional simulation languages is the lack of attention paid to the modeling of control structures, i.e., the humans or systems responsible for manufacturing planning and control, their activities and the mutual tuning of their activities. Mostly they are hard coded and dispersed throughout the model. Consequently, not only realism but also modeling flexibility and modularity is harmed. In recognition of this fact we consider a framework for simulation modeling that explicitly represents control structures. The framework is meant to serve as a conceptual basis for extending capabilities of simulation models, tools and libraries in analyzing manufacturing systems. It does so by capturing key-abstractions of the manufacturing field in terms of classes and their relationships. To study the practical relevance of the framework its concepts were implemented in an object-oriented simulation language and applied to a case example.

1 INTRODUCTION

A fundamental challenge in simulation modeling of manufacturing systems is to produce models that can be understood by the problem owner. A clear notion of the model not only increases his confidence in problem solutions, it also helps in model verification and validation. Moreover, better or other solutions may be suggested as a joint effort of analyst and problem owner (Bell and O'Keefe, 1987). In this respect the introduction of visual interactive simulation by Hurrion (1976, 1989) meant a significant support for the analyst, as the problem owner may now be involved easier in problem solution (Bell et al. 1999). However, while the availability of a graphical language may help to realize a "communicative" model (Balci 1986, Robinson 1994), still languages "spoken" by the analyst and problem owner may be quite different from eachother. Often a pictorial display boils down to a mapping of programming constructs to their graphical counterparts - it does not offer a common language. Stated in an other way: the availability of a common means does not guarantee a common understanding. This is not only true for the analyst versus the problem owner, also different (competent, even expert) analysts may come up with quite different simulation models for a single scenario using the same language. A solution for this problem would be the introduction of simulation languages, which are closer to natural languages. The renewed interest in object-oriented simulation languages in the 1990s, which may be traced back to the simulation language Simula (Dahl 1966), contributes in this direction. Apart from the modeling efficiencies in terms of re-use, readability, and maintainability object orientation facilitates a natural one-to-one mapping of real world concepts to modeling constructs (see e.g. Adiga and Glassey 1991, Kreutzer 1993, Roberts and Dessouky 1998). Not surprisingly, this also means that real world concepts, which had to be modeled rather implicitly in traditional simulation languages, are now explicitly included. Decision-making is an important example of such a concept. Decision makers, control rules and their interactions are often "hidden" in these traditional languages. Mostly they are hard coded and dispersed throughout the model (Pratt et al. 1994). Consequently, not only realism but also modeling flexibility and modularity is harmed (Karacal and Mize 1996). Moreover, alternative and possibly better solutions to control problems may be overlooked.

In this article we review object-oriented simulation methodologies that recognize the need for the explicit representation of decision-making. After highlighting shortcomings in current approaches we consider a *new modeling framework for manufacturing simulation* (van der Zee 1997, van der Zee and van der Vorst 2002). The framework is meant to serve as a conceptual basis for extending capabilities of simulation models, tools and libraries in analyzing manufacturing systems. Control problems are specifically addressed by the framework through *the explicit notion of control structures*, i.e., the managers or systems responsible for control, their activities and their mutual tuning of these activities. Key concepts in the framework for modeling con-

trol structures are agents, jobs and flows. Agents model the relevant entities in a manufacturing system, e.g. work stations, storages, planners and information systems. Each agent is assigned decision-making intelligence. Jobs model activities of agents. Next to physical and data transformations control activities are explicitly recognized as jobs. Finally, flows constitute the movable objects being subject to jobs, e.g. goods, tools, personel, and data.

The remainder of the paper is organized as follows. First we review object-oriented simulation methodologies that explicitly allow for modeling control (Section 2). Subsequently, in Section 3, we define the modeling framework. In Section 4 we consider its implementation in an object-oriented simulation language by means of a small case example. Finally, in Section 5 we summarize our main conclusions and highlight directions for future research.

2 LITERATURE REVIEW

In a recent overview Narayanan et al. (1998) identify six large-scale, persistent object-oriented simulation methodologies. Among the methodologies identified they mention three methodologies that allow for explicit representations of the decision-making process itself. They are addressed as Laval, OSU-CIM and OOSIM. Below we will shortly characterize the approaches. Please note that a more elaborate description of the methodologies, including several references, can be found in the article of Narayanan et al. (1998).

- Laval: The Laval approach is associated with a research group at Laval University, Canada, see e.g. Lefrancois and Montreuil (1994). The Laval approach distinguishes between intelligent agents and non-intelligent objects. Where the non-intelligent objects model e.g. workstations and work orders, agents are used to implement all decision support functions. As such agents may represent decision makers, analysts, evaluators etc. Intelligence of agents is associated with decision functions. Decision functions concern the allocation and release of jobs to resources, job sequencing and the monitoring of the status of resource it controls. To realize decision functions an agent has a memory, facilities for communication with other agents and reasoning capabilities.
- OSU-CIM: The OSU-CIM approach is associated with a research group at the Oklahoma State University, USA, see e.g. Pratt et al. (1994). They advocate the separation of physical, information and control elements in simulation models. Entities in manufacturing systems like machines and AGVs are modeled by a set of primitives including a controller primitive that represents decision-making aspects. Higher-level controllers are foreseen for coordinating activities of several entities. Controllers communicate by sending messages.
- OOSIM: The OSU-CIM approach is associated with a research group at the Georgia Institute of Technology, USA, see e.g. Narayanan et al. (2000). Their object-based modeling architecture builds on four fundamental software abstractions: material, locations, controllers, and process plans. Locations can either process material or store material. Process plans specify the required operations for producing specific parts. Controllers perform decision-making and control within the simulation. Controllers are event-driven entities that respond to state changes within their domain. Also they may communicate with other controllers.

All approaches mentioned recognize the need for separating activities associated with decision making from other activities related to physical transformations or data processing. Differences are found in the choice of primitive classes for modeling manufacturing entities and their relationships. Also the level of detail with which classes and class hierarchies are defined differs. Here we will only consider the modeling of control structures, i.e., the managers or systems responsible for control, their activities and their mutual tuning of these activities. All three approaches allow for specifying *decision logic* in terms of control rules for steering activities within a certain domain. Also the need for modeling control concepts like hierarchy and coordination is recognized. What is striking, however, is the lack of attention for *decision logistics*, i.e., the timing of control activities. Clearly, the timing of decisions may have a large impact on manufacturing performance. Dynamics in simulation models is mainly related to physical transformations and not to decision-making activities.

In this article we present an alternative modeling framework where activities associated with decision-making are modeled as decision jobs. Just like jobs associated with physical transformations decision jobs allow for a time related behavior. As will be shown in the next section the symmetry in addressing both types of activities allows for a common definition of manufacturing entities and a natural denominator for manufacturing dynamics – the job.

3 DEFINITION OF THE MODELING FRAMEWORK

In this section we define the constituent parts of the proposed modeling framework (MF). First the basic classes for building simulation models of manufacturing systems are identified, i.e., agent, flow item and job (Subsection 3.1). Next, in Subsection 3.2, their internal structure and coupling are discussed. Finally, in Subsection 3.3, we describe model dynamics. In section 4 we will show how class definitions may be formalized by implementing them in an object-oriented simulation language. Please note that a more elaborate description of our modeling framework can be found in van der Zee (1997) and van der Zee and van der Vorst (2002).

3.1 Class Hierarchies – A Job Oriented World View

To represent entities in the manufacturing domain we define three main classes in our modeling framework: agents, flow items and jobs. They are presented in Figure 1a,b,c. Classes are defined using the notation supplied by Booch (1994).

Agents represent the infrastructural elements of a manufacturing system such as workstations, information systems and managers. They are assumed to be intelligent to a certain extent. Their decision-making capabilities relate to transformations of goods or data. A boundary is recognized between the system under study and its related environment. This is reflected by distinguishing between internal and external agents. The latter types of entities are modeled as primary producers ("sources") and consumers ("sinks"). For internal agents such as machines, warehouses, AGV systems, planners etc. it is distinguished between processors and storages. This is a natural separation represented in many diagramming techniques, like e.g. activity cycle diagrams and Petri nets, cf. Pooley (1991). It isolates "servers" from "queues". To reflect the level of detail required in the simulation study internal agents may be used to model relevant entities within a company at different aggregation levels.

Flow items constitute the movable objects within manufacturing systems. We include four types of flow items in the modeling framework: *goods, resources* (like e.g. manpower, tools vehicles), *data* and *job definitions*. Goods, resources or data seldom flow spontaneously from one location to another - mostly some form of control is exercised. Typically, the activities of agents are directed by messages. We address this type of messages as *job definitions*. Job definitions specify a job in terms of e.g. its input, processing conditions and the agents to whom the resulting output should be sent.

It is common practice to think of *agents* in terms of the type of flow items that are the subject of their jobs. In line with practice it is possible to define more specific classes of internal agents, where the type of flow item serves as a parameter. For example, a workstation may be considered an internal agent of a processor type handling goods. In a similar way control systems and *decision-makers may be defined as internal agents producing job definitions*.

In a manufacturing system agents and flows are linked by *jobs*, i.e., business activities. In our *job oriented world-view* we assume that *each manufacturing activity corresponds to a job that is specified by its controller*, i.e., an other agent responsible for issuing or planning activities.

From a business point of view it is very natural to give jobs a central place in a model, because a companies depend on the execution of these value-adding activities for their existence, cf. Porter (1985).

As far as the choice of classes is concerned similarities may be found between classes defined above and abstrations defined as a result of earlier research efforts, see Section 2. As a historic precursor one may consider the the notation proposed by Forrester (1961). He also assumes an explicit notion of activities related to decision making in

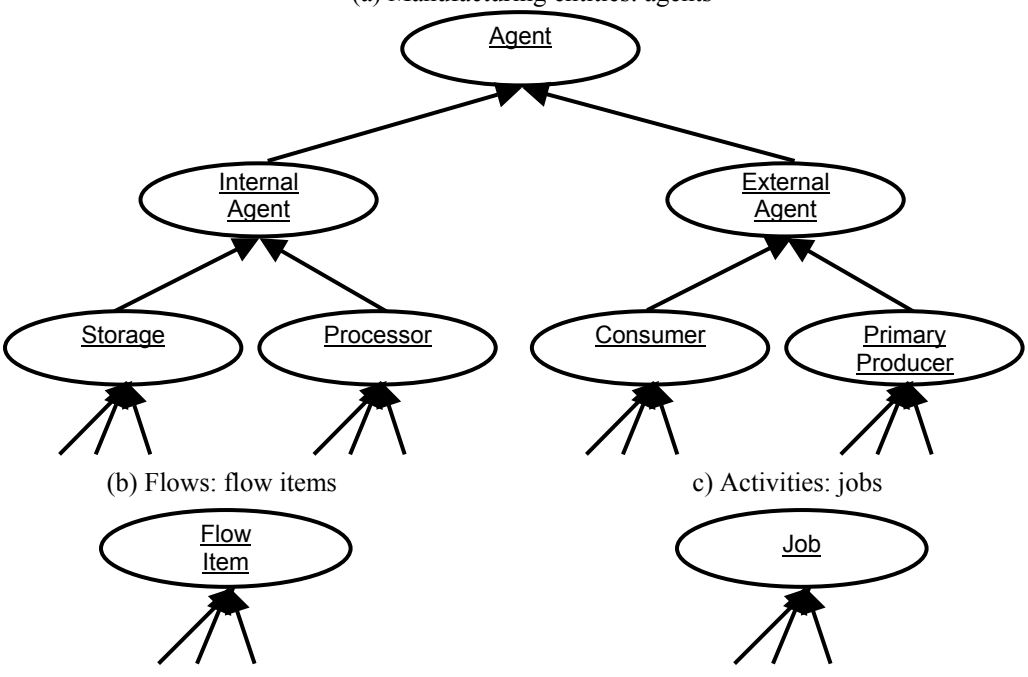

Figure 1: Main Classes in the Modeling Framework

terms of decision functions. Also a distinction is made in several types of flows. Next to those flow types mentioned above he considers also money and capital equipment. We consider money as a specialization of the class data, while capital equipment is modeled in terms of agents.

3.2 Internal Structure of Agents

In this subsection we consider the internal structure for agents, building on the class definitions supplied in the previous subsection. Let us start by considering the structure for internal agents (Figure 2). The definition of a structure for an internal agent was inspired by the atomic model as defined by Zeigler (1990). Starting from a general view on simulation modeling an atomic model captures basic elements and functions of an entity in a formal way. In Figure 2 the input and output 'ports' for an internal agent are denoted by the lines crossing the oval. Typically, ports (interfaces) are related to a type of flow item. In the figure only two input ports are shown: one through which job definitions are received and one through which other types of flow items may be received. However, more ports may be distinguished, e.g. in order to make a distinction between resources, information and goods. The same applies to output ports.

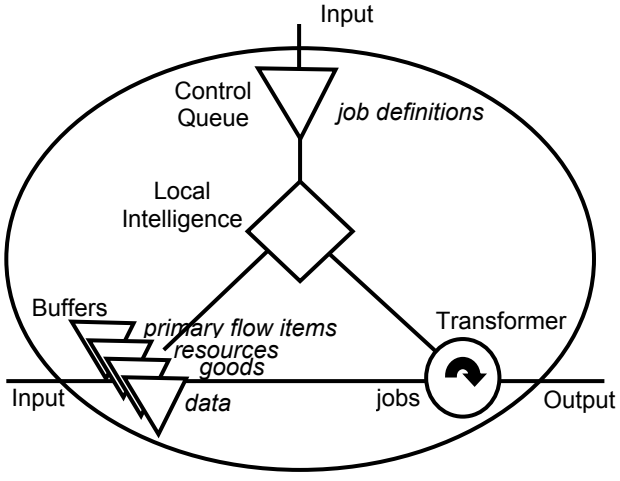

Figure 2: Structure for an Internal Agent

The state of an agent relates to its attributes. Attributes concern *buffers* and *transformers*. Buffers model the temporary storage of those flow items which are the subject of a future job or which facilitate a job (resources, information). The first category of flow items is addressed in Figure 2 as primary flow items. Except for the buffer that handles the job definitions for an agent, i.e., the *control queue*, buffers for facilitative flow items are optional. The transformer contains the jobs in execution and those flow items that are subject to a job in execution.

The handling of incoming flow items is dealt with by one or more *input* operations. An input operation places flow items in the right buffers. The figure shows two such input operations: one that puts job definitions in the control queue and one that updates buffers.

The initiation of a job is enabled by rules comprised in the *local* intelligence. As a first rule in initiating a job, the job with the highest priority in the control queue is investigated for execution. Before a job may be started, two requirements (preconditions) have to be fulfilled:

- The availability of a job definition
- The availability of the required input for a job

In accordance with our job-oriented approach each job has to be pre-specified. This is reflected in the requirement that a job definition should be present in the control queue. The job definition encompasses the required input to be withdrawn from the buffers, capacity needed, processing conditions and the identifiers of agents to which the job's output has to be sent. In accordance with our emphasis on an explicit notion of control structures, the set of rules comprised in the local intelligence should typically be small for those agents who are not realizing control functions. Otherwise they might interfere too much with the decision freedom of the controller. On the other hand decision logic for controllers may be comprehensive, involving data handling and the calling of decision jobs, i.e., control strategies, which map information on system status to job definitions.

After completion of the job the output operations take care of sending the resulting items to the respective output addresses (agents) by calling the respective input operations.

Let us now consider external agents, i.e., the clients and suppliers that make up the environment for a business system (Figure 3). In Figure 3 a distinction is made between three types (classes) of elements. Besides the element local intelligence, which is also found for internal agents, *generators* and *annihilators* are distinguished. Generators represent 'sources' of flow items, while annihilators model 'sinks' in which flow items disappear. Local intelligence may be used to link activities of generator and annihilator. For example, local intelligence may comprise a rule that states that a new order may only be issued if the goods corresponding to the last order have been received.

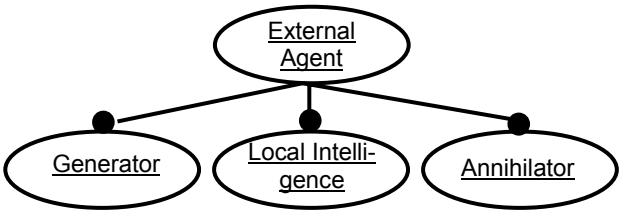

Figure 3: Elements of External Agents

Above we discussed the structure for the different types of agents. In the next subsection we will consider their coupling (class relationships).

3.3 Relationships between Agents

In this subsection we will consider two categories of relationships between agents:

- The relationship between an internal agent and its controller
- Relationships between external and internal agents

These relationships may be considered as specializations of the basic type of relationship between agents. This basic type of relationship foresees in a restricted set of flow items being exchanged between two agents.

Figure 4 depicts the control relationship between the class controller and the class internal agent. Control is assumed to be effectuated by the sending of job definitions from a controller object to an internal agent, denoted as Int. Each agent (subordinate) refers to exactly one controller (manager) from which it receives its job definitions, denoted as F(C). Reversely, a subordinate can send information (F(I|D)) about its status to its controller. Such a status report acts as a request for control, as it is one of the activities of the controller to interpret this type of message. Note how mechanisms like hierarchical control and coordinated control are embedded in this class structure. Both mechanisms may be considered as important building blocks in manufacturing control systems.

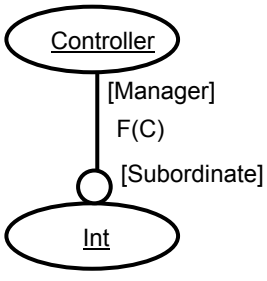

Figure 4: Relationship between an Internal Agent and its Controller

The effectuation of control relationships steers the behavior of the manufacturing system within its boundaries. Let us now consider relationships of the manufacturing system with the outside world, i.e., external agents. For external agents we distinguish between consumers and primary producers. Figure 5 shows how a consumer (Con) issues an order by sending a demand signal (F(I|D)) to an internal agent of the type controller. The controller in its turn specifies a job definition (F(C)) for an internal agent (Int) who is responsible for the deliverance of the requested items (F(M)), where M refers to the modality. In the case of a primary producer roles have changed: the controller sends an order to a primary producer (PP), who has to take care of delivery of the requested items.

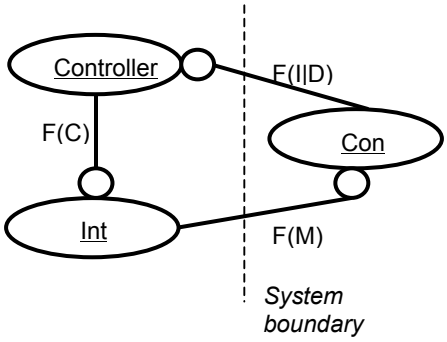

Figure 5: Relationships between Internal Agents and External Agents (Consumers)

3.4 Dynamics Structure

In line with our job-oriented view we assume the execution of jobs by agents as the driving force of business dynamics (see also Subsection 3.2). Job execution is related to a procedural three-phase description, cf. Pidd (1998). In the A Phase it is determined which job is completed next (Figure 6). Once this job has been found, time is advanced to the corresponding moment in time. Subsequently, the job is 'completed' (in the B-Phase), i.e., the resulting output is sent from the agent that carries out the job to other agents. In the C-Phase it is tested if the flow items that are received by these agents enable the initiation of new jobs (conditional activities). The three phases are repeatedly run through until the (simulation) time is up.

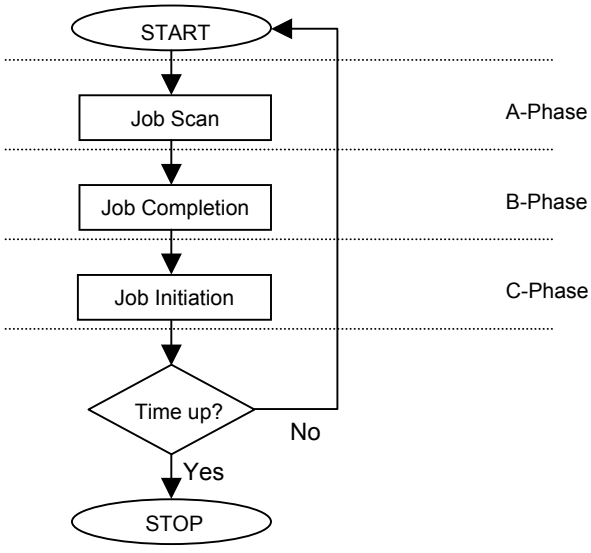

Figure 6: Simulator Definition

4 IMPLEMENTATION AND APPLICATION OF THE MODELING FRAMEWORK

To illustrate the implementation and use of our modeling framework we discuss a simulation model of a small ficti-

tious repair shop. As a modeling tool we used the simulation language EM-Plant™ (Technomatix). EM-Plant™ is one of the few true object-oriented simulation packages that is commercially available, cf. Law and Kelton (2000). It proved to be a flexible tool in implementing the concepts included in our modeling framework. Note that an alternative choice of a tool is very well possible due to the general nature of the framework.

For modeling the repair shop a class library is built which comprises the class definitions for the three main classes in the modeling framework, i.e., flow items, logistic agents and jobs (Figure 7). These classes are the essential building blocks for the aggegrate class RepairShop. Please note that in more complex shops it may be worthwhile to introduce more aggregate classes representing hierarchical levels in modeling. Such classes help to improve model overview. All classes are built starting from the basic class library of EM-Plant™ that covers the class definitions contained in the folders MaterialFlow, InformationFlow, UserInterface and MUs.

In this subsection we will address the implementation of the three main classes FlowItems, Agents, and Jobs by giving a number of examples. Subsequently, we will relate the classes and model dynamics by considering the internal structure and workings of an agent. We will use the class RepairShop as a starting point for our discussion (Figure 7).

Flow items are represented in EM-Plant by "movable units". We distinguish between four classes of flow items in this model: Engine, StatusUpdate, JobDefinition and Scheduler. Engines model the physical flows between the logistic agents. StatusUpdate and JobDefinition are used to model feedback and control among agents. The Scheduler is used to represent the availability of a person capable of scheduling jobs for the RepairStation (see below). Each flow item has multiple attributes. Next to "header data" needed for identification or routing, they represent the logical or physical contents of the associated object.

The model (see Figure 7) distinguishes between *external agents and internal agents*. External agents considered are the customers asking for repair of their engines (CustomersIn, CustomersOut). Internal agents model the parties involved in the shop. They are associated with the physical handling of goods (InspectionStation, RepairStation), i.e., engines; data processing (InspectionStation) and control in terms of the scheduling and release of jobs (Planning). The agents communicate by flow items as introduced in the previous section.

In order to make the shop work jobs have been allocated to agents. To reflect the different nature of jobs we distinguished between several classes of jobs. For example the agent Planning is associated with two classes of jobs:

- Release – release repair jobs
- Schedule – set up a new schedule of repair jobs

All job classes are implemented in EM-Plant™ Methods, i.e., code (see Figure 8).

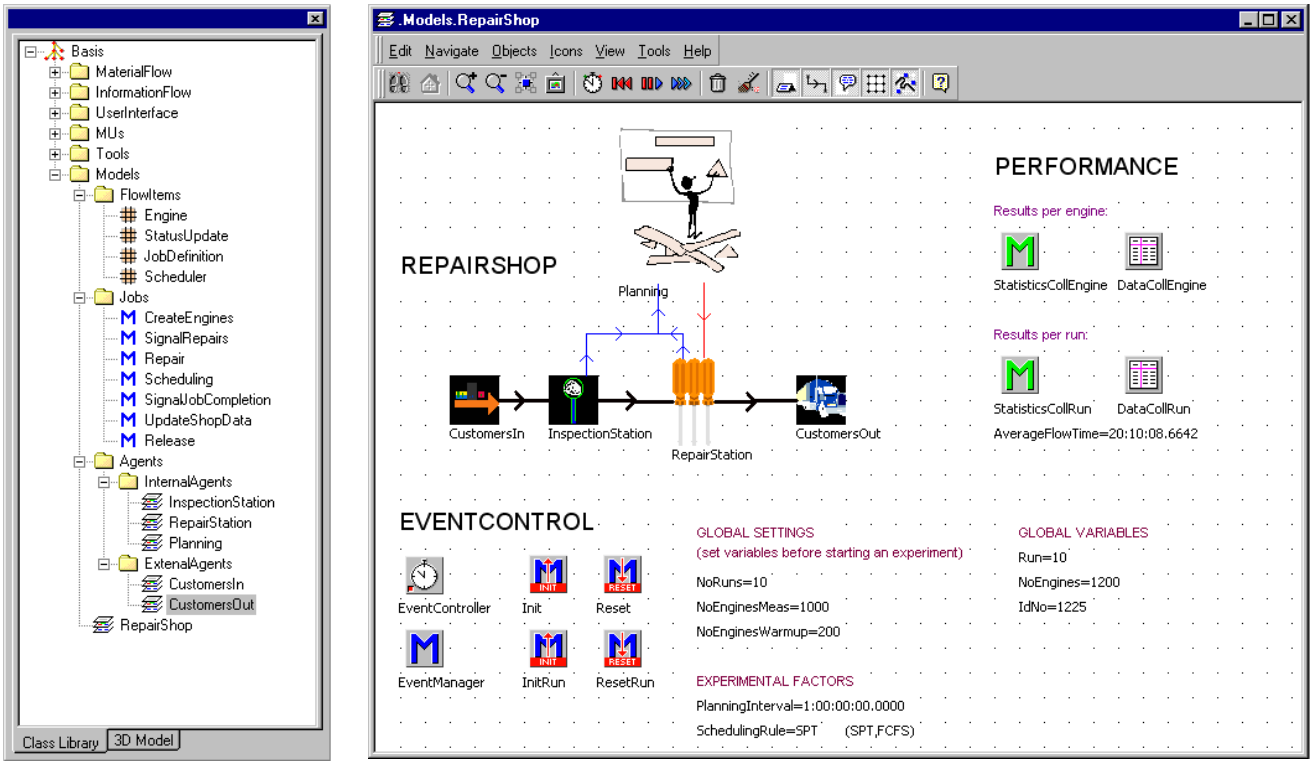

Figure 7: Class Library and Class Repairshop

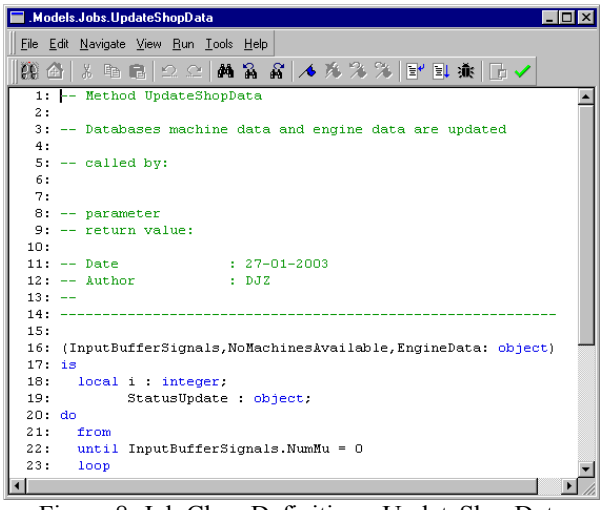

Figure 8: Job Class Definition - UpdateShopData

To show how classes may be linked we will now look at the internal structure and working of the agent RepairStation (Figure 9). Basically, the class definition covers the functionalities introduced in Figure 2, where we give a general class definition for an internal agent. Buffers considered are: InputBufferGoods and JobQueue. Transformers are TransformerGoods and TransformerSignals. Both are linked by local intelligence (JobExecutionProc). The local intelligence takes care of calling on the right jobs for realizing the required transformations. The local intelligence is activated by the arrival of job definitions and/or goods, i.e., engines.

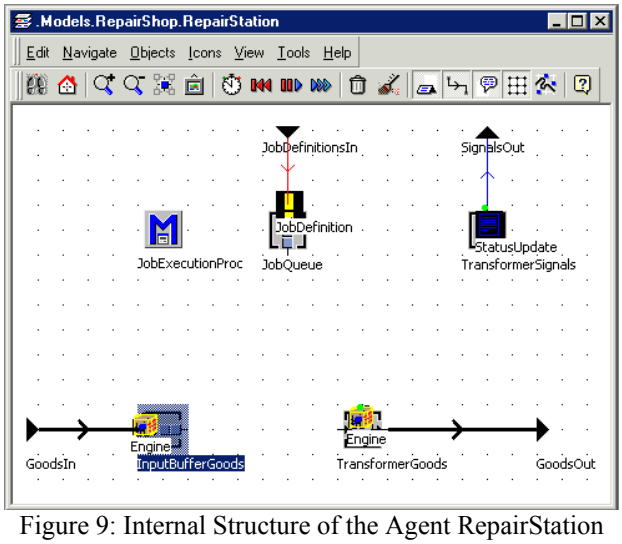

Figure 9: Internal Structure of the Agent RepairStation

Above we discussed the modeling of the default shop configuration in rough detail. The modeling framework offers a flexible response for modeling alternative control structures, for example:

- Alternative control rules can be modeled by adapting the definition of job classes associated with the planning department.
- The choice of another planning period is realized by considering the time related behavior of the planning job.
- The quality of information on the required repair time for engines as supplied by the inspection station can be adapted by the jobs associated with this station.

Alternative scenarios may e.g. concern the distribution of job classes over the agents or the number of agents involved. For example, where the default shop model assumes one agent to be responsible for both release and scheduling of the repair station, in an alternative setting there may be two specialized agents each being responsible for one task. Note how this separation of tasks resembles different levels of shop control.

We found that adapting our model to deal with scenarios like those mentioned above is relatively easy. For a major part this is due to the object-orientedness of the approach. For another important part, however, the ease of adaptation comes forth from the natural mapping of concepts – knowing where to look and to make the change.

5 CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

In this article a modeling framework for manufacturing simulation has been proposed. The approach stresses an object-oriented choice of concepts that is transparent from a logistic point of view. The framework is meant to serve as a conceptual basis for extending capabilities of simulation models, tools and libraries in analyzing manufacturing systems.

The framework specifically pays attention to decision logistics in terms of decision makers, their activities and the mutual tuning of their activities. They are naturally embedded in our approach by considering decision makers as agents who carry out control jobs, just like machines carry out physical jobs. Where physical jobs result in goods, control jobs result in job definitions for logistic agents in the controllers' domain of control. A clear internal structure for agents has been defined that specifies how jobs are being processed. Model dynamics is linked to job dynamics as activities only start if both a job definition and its required input are present.

The authors see the natural way in which control is embedded in our modeling framework as a means for more efficient and effective simulation. Especially, in case simulation modeling is used for testing systems for manufacturing planning en control systems, benefits of the new approach are expected in model building, verification and the generation of alternative solutions. Some indications of the expected benefits are illustrated by a case example. Clearly, many more case examples should be studied to improve insight in these advantages.

Some interesting directions for future research include specialization of the modeling framework towards specific industries, and its use for creating manufacturing games.

REFERENCES

Adiga, S., and C.R. Glassey. 1991. Object-oriented simulation to support research in manufacturing systems. *International Journal of Production Research*, 29(12): 2529-2542.

Balci, O. 1986, Credibility assessment of simulation results. In: *Proceedings of the 1986 Winter Simulation Conference*. 38-43, Piscataway, New Jersey: IEEE.

Bell, P.C., and R.M. O'Keefe. 1987. Visual Interactive Simulation - History, recent developments, and major issues. *Simulation*, 49(3): 109-116.

Bell, P.C., Anderson, C.K., Staples, D.S., and M. Elder. 1999. Decision-makers'perceptions of the value and impact of visual interactive modeling. *Omega – The International Journal of Management Science*, 27: 155-165.

Booch, G. 1994. *Object-oriented Analysis and Design with applications*. Redwood City: Benjamin Cummings.

Dahl, O., and K. Nygaard. 1966. SIMULA - an Algol-based simulation language. *Communications of the ACM*, 9(9): 671-678.

Forrester, J.W. 1961, *Industrial Dynamics*. New York: Wiley.

Hurrion, R.D. 1976. The design, use and required facilities of an interactive visual computer simulation language to explore production planning problems. PhD thesis, University of London, England.

Hurrion, R.D. 1989. Graphics and interaction. *Computer modeling for discrete simulation*, ed. M.Pidd. Chichester: Wiley.

Karacal, S.C., and J.H. Mize. 1996. A formal structure for discrete event simulation. Part I: Modeling multiple level systems. *IIE Transactions*, 28(9): 753-760.

Kreutzer, W. 1993. The Role of Complexity Reduction in the Development of Simulation Programming Tools, An Advanced Tutorial. In: *Proceedings of European Simulation Conference*. Delft: Society for Computer Simulation.

Law, A.M., and W.D. Kelton. 2000. *Simulation Modeling and Analysis*. (Singapore: McGraw-Hill.

Lefrancois, P., and B. Montreuil. 1994. An object-oriented knowledge representation for intelligent control of manufacturing workstations. *IIE Transactions*, 26(1): 11-26.

Narayanan, S., Bodner, D.A., Sreekanth, U., Govindaraj, T., McGinnis, L.F., and C.S. Mitchell. 1998. Research in object-oriented manufacturing simulations: an assessment of the state of the art. *IIE Transactions*, 30(9): 975-810.

Narayanan, S., Evans, J., Bodner, D.A., Sreekanth, U., Govindaraj, T., McGinnis, L.F., and C.S. Mitchell. 2000. Modeling a Printed Circuit Board Assembly Line Using Objects. *Simulation*, 75(5-6): 287-300.

Pidd, M. 1998. *Computer Simulation in Management Science*. Chichester: Wiley.

Pooley, R.J. 1991. Towards a Standard for Hierachical Process Oriented Discrete Event Simulation Diagrams, Part 1: A comparison of existing approaches. *Transactions of the Society for Computer Simulation International*, 8(1): 1-20.

Porter, M.E. 1985. *Competitive advantage: Creating and sustaining superior performance*. New York: The Free Press.

Pratt, D.B., Farrington, P.A., Basnet, C.B., Bhuskute, H.C., Kanath, M., and J.H. Mize. 1994. The separation of physical, information, and control elements for facilitating reusability in simulation modeling. *International Journal of Computer Simulation*, 4(3): 327-342.

Roberts, C.A., Y.M. Dessouky. 1998. An Overview of object-oriented simulation. *Simulation*,70(6): 359-368.

Robinson, S. 1994. *Successful Simulation – A practical approach to simulation projects*. London: McGraw-Hill.

Zee, D.J. van der. 1997. Simulation as a tool for logistics management. PhD thesis, University of Twente, The Netherlands.

Zee, D.J. van der, J. van der Vorst. 2002. A modeling framework for supply chain simulation, Technical Report No. 02A54. Groningen : Research School SOM.

Zeigler, B.P.. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models*, Intelligent Agents and Endomorphic Systems. London: Academic Press.

AUTHOR BIOGRAPHY

DURK-JOUKE VAN DER ZEE is Assistant Professor of Production Systems Design at the Faculty of Management and Organization, University of Groningen, The Netherlands. Dr. van der Zee received his M.Sc. and Ph.D. in Industrial Engineering at the University of Twente, The Netherlands. His research interests include simulation methodology and applications, shop floor control systems and decision making related to the introduction and use of flexible manufacturing systems. He can be contacted by e-mail at <d.j.van.der.zee@bdk.rug.nl>