

understanding. We studied the relevance and use of the modelling framework for simulation studies on manufacturing planning and control, and on supply chain design, see [15,16].

In this article, we generalize our focus by considering the role and meaning of the modelling framework for the process of simulation model construction. Therefore we first discuss the basics of simulation modelling in terms of elementary principles for simulation model decomposition. Next, we show how the modelling framework is built up, starting from the combination of these principles and linking them to concepts that appeal to the manufacturing field. Finally, we illustrate its implementation by means of a case example, and discuss its relevance for model construction and use.

A key new decomposition principle that is implemented in the modelling framework is: “All activities in a manufacturing system have a common denominator: the job.” Here a job refers to those activities that are associated with a specific transformation of goods and/or data. This includes those activities associated with decision makers responsible for manufacturing planning and control (e.g. schedulers, operators) and humans or systems supporting them (e.g. forecasters), i.e., decision logistics. Making all transformations explicit in a uniform way helps in avoiding the “hiding” of potential design variables. At the same time it leads to a more natural modelling from a business perspective, as value-adding activities are clearly identified [17], and model dynamics corresponds with the execution of jobs. As far as model building is concerned adherence to this “job oriented approach” enables a generic definition of entities in terms of agents that share a basic internal structure and behaviour.

The remainder of the paper is organized as follows. First we review elementary decomposition principles for simulation modelling (Section 2). Subsequently, in Section 3, we define the modelling framework. In Section 4 we consider its implementation in an object-oriented simulation language by means of a small case example. Starting from the case example, we discuss contributions of the modelling framework for model construction and use (Section 5). Finally, in Section 6 we summarize our main conclusions and highlight directions for future research.

2. DECOMPOSITION PRINCIPLES FOR SIMULATION MODELLING

“The process by which a systems engineer or management scientist derives a model of a system he is studying can best be described as an intuitive art. Any set of rules for developing models has limited usefulness at best and can only serve as a suggested framework or approach” [18]. We recognize the limitations of guidelines for simulation modelling – their relevance primarily lies in model structuring – not in detailing model elements. Typically, model components have to be tailored towards the specifics of the system under study. However, it is especially the notion of the model structure – in terms of its basic elements and their workings – which may make a difference in model understanding. This makes the identification and studying of the underlying rules for model construction, i.e., decomposition principles, worthwhile. In this section we will do so by reviewing a number of basic principles as they have been introduced in simulation literature. Next, we will argue the potential for improving model quality by adding a new principle – in terms of our job oriented approach towards manufacturing simulation.

Model and experimental frame

As a first step in modelling Zeigler [19] suggests to distinguish between the “experimental frame” and the “model”. Where the model corresponds to a (limited) representation of a real world system, the experimental frame specifies those circumstances under which the real system is to be observed or experimented with. The separation between model and experimental frame has been implemented in the language SIMANTM [20,21] Here the model describes the physical elements of the system (machines, workers, goods, etc.) and their logical interrelationships. The experimental frame specifies the experimental conditions under which the model is to run, like initial conditions, type of statistics gathered etc.

The structure for a model concerns two types of elements [19]:

- Component models that make up the overall model. They specify the static characteristics for the system. Their attributes specify states, inputs and outputs.
- Rules of interaction among component models – they specify dynamics for the system.

Let us now first consider principles for defining model components.

External/internal entities

The distinction between external and internal entities follows from the concept of a system boundary. Typically, system design involves the choice, configuration and operation of “internal” entities. “External” entities are only modelled as far as their behaviour is relevant for the system. Rather, they act as “sources” or “sinks” for physical flows (goods, resources), or data.

Movable and non-movable entities

Entities like workstations, information systems, and managers make up the physical and logical infrastructure for a manufacturing system. As such they are considered non-movable. They communicate by exchanging movable items like goods and messages.

Queues and servers

Non-movable entities may be classified according to their associated activities. Following the general notion of manufacturing systems being queueing systems a basic distinction can be made between queues (items waiting) and servers (items being processed).

The above principles share a general focus on manufacturing systems modelling. In recent years the relevance of two additional principles for classifying model components has been recognized:

Physical, information and control elements

The separation of physical, information and control elements is assumed to facilitate a higher degree of model reusability and a more “natural” model building environment [11]. Mize et al. point at the fact that traditional languages do not provide natural constructs for separately and distinctly modelling the three types of basic functions, i.e., physical, information and control/decision. In addition, the constructs provided for information and control are often hard coded and dispersed in the model [12]. Clearly, this hinders modification and re-use of code. A policy that adheres rather strictly to the “one component-one function” doctrine would suffer less from these drawbacks. In turn this provides a more natural modelling environment as the modeller is forced to think about model elements independently.

Intelligent and non-intelligent entities

According to Lefrancois and Montreuil [22], and Lefrancois et al. [23] a distinction between intelligent and non-intelligent entities permits a more natural and richer presentation and implementation of systems modelled. In such a

context, intelligent beings are modelled as agents. Agents are used to implement decision rules inherent to manufacturing system planning and control. Non-intelligent objects model e.g. workstations and work orders.

The above decomposition principles focus on a “natural” one-to-one mapping of real world entities on model components, starting from a categorization that appeals to the manufacturing field. Clearly, this contributes to model transparency and completeness. However, the rules offer less guidance where it comes to the workings of the model, i.e., the specification of activities and their execution. Note how the term “workings” relates structure of model components (static characteristics) to model dynamics. Therefore we introduce a new principle: “*all activities in the manufacturing system have a common denominator: the job*” [15,16]. The job concept is meant to bring two important advantages. First, the use of this common denominator for all activities will provide a clear and natural mechanism for event scheduling, where events are related to the start and the completion of jobs. Second, an explicit notion and allocation of company activities, increases visibility and traceability of decision variables. This is especially true where it comes to modelling decision logistics, in terms of entities who are responsible for the control of manufacturing operations and the mutual attuning and timing of their activities.

Above we discussed basic decomposition principles for simulation modelling. We consider these principles to be the building blocks underlying our modelling framework for manufacturing simulation.

3. DEFINITION OF THE MODELLING FRAMEWORK

In our focus a *modelling framework* concerns a well-defined conceptual view on manufacturing systems. Typically, it would capture essential elements and relationships as well as their dynamics. As such it would be supportive to simulation model construction as well as the mental modelling of the analyst and stakeholders, see Figure 1.

In this section we define the constituent parts of the modelling framework (MF) building on: (1) The decomposition principles for manufacturing systems modelling introduced in the previous section, and (2) The object model.

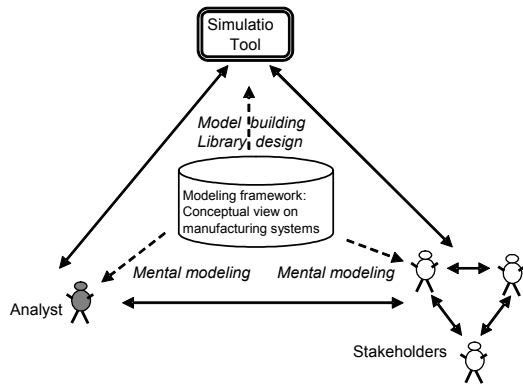


Figure 1: Modelling Framework

The object model [5] underlies the object oriented approach, see Section 1. Its elements specify a number of non system specific decomposition principles:

- In object-oriented design, *abstractions* are used to refer to the essential characteristics of an object which distinguish it from all other objects. Abstractions are denoted as classes and objects. A class defines the structure and behaviour for a group of similar objects. Objects resemble entities in a certain field of interest. An object has a state, behaviour and identity.
- *Encapsulation* serves to separate the interface of an abstraction from its implementation. Its essential benefit is that it is possible to change the implementation of an abstraction without disturbing the relationships with any of its clients.
- *Modularity* offers a way to deal with the complexity of large systems. It considers the decomposition of a system into cohesive and loosely coupled modules. The cohesiveness of a module is improved by grouping logically related abstractions.
- The two most important hierarchies in complex systems are its *class hierarchy* and its *object hierarchy*. The class hierarchy points out the *inheritance* relationships between classes. It shows how classes may share (re-use) the structure and behaviour defined in other classes. The object hierarchy shows in which way objects are part of a larger whole.

Let us now discuss the definition of the constituent parts of the modelling 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 Subsections 3.2,3.3 their internal structure and coupling are discussed. Finally, in Subsection 3.4, we describe model dynamics.

3.1 CLASS HIERARCHIES – A JOB ORIENTED WORLD VIEW

To represent entities in the manufacturing domain we define three main classes in our modelling framework: *agents*, *flow items* and *jobs*. They are presented in Figure 2a,b,c. Classes are defined using the object-oriented notation supplied by Booch [5].

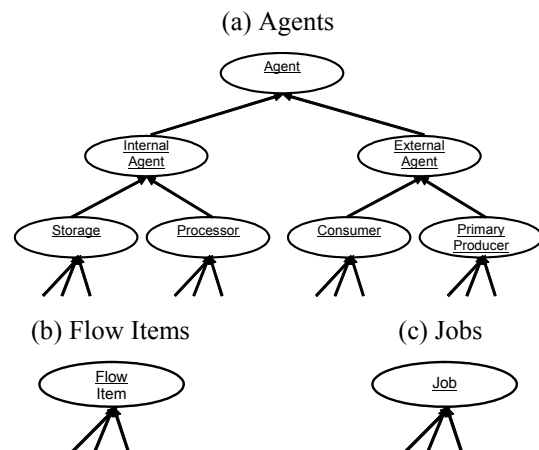


Figure 2: Main Classes in the Modelling Framework

Agents represent the infrastructural – *non-movable* - 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 modelled 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.

Flow items constitute the *movable* objects within manufacturing systems. We include four types of flow items in the modelling 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 worldview* we assume that *each business activity corresponds to a job.*

3.2 INTERNAL STRUCTURE OF AGENTS

In this subsection we consider the structure for agents, building on the class definitions supplied in the previous subsection. Let us start by considering the structure for internal agents (Figure 3). The definition of a structure for an internal agent was inspired by the atomic model as defined by Zeigler [19,24]. Starting from a general view on simulation modelling an atomic model *encapsulates* basic elements and functions of an entity in a formal way.

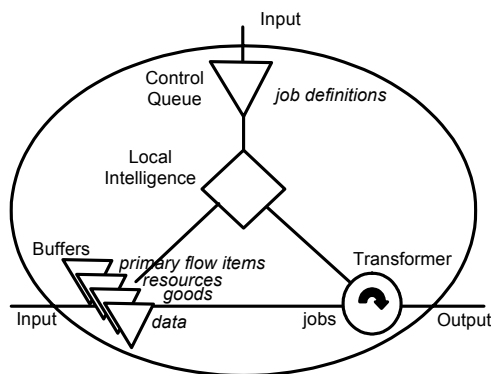


Figure 3: Structure for an Internal Agent

In Figure 3 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.

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 3 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. Within the context of manufacturing planning and control these strategies may refer to a wide range of rules that support e.g. capacity planning, material planning, scheduling and dispatching. Note how the output of the one decision job (e.g. capacity planning) may set (part of the) input for the other decision job (e.g. scheduling). In this respect a hierarchy of controllers and their jobs may be distinguished (see Subsection 3.3).

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. In Figure 4 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.

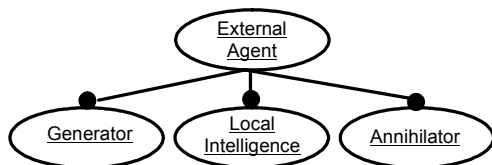


Figure 4: Elements of External Agents

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 a restricted set of flow items being exchanged between two agents. Figure 5 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 a manufacturing planning and control system (mpcs). For

example, a mpcs framework (hierarchy) may be considered as consisting of three levels [25]:

- Level 1: Set of activities and systems for overall direction setting, like demand management, resource planning, sales and operations planning and master production scheduling.
- Level 2: Detailed material and capacity planning.
- Level 3: Execution systems for shop floor control and communication with suppliers.

Typically, the actual framework for a company may be set up in numerous ways by making choices with respect to the design and use of the systems making up each level. Essential choices concern the timing and contents of decision jobs (see Subsection 3.2). These choices are influenced by more general planning and control concepts like Manufacturing Resource Planning (MRP), Just in Time (JIT), CONstant Work In Process (CONWIP), see [26].

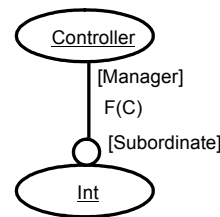


Figure 5: Relationship between an Internal Agent and its Controller

For external agents we distinguish between consumers and primary producers. Figure 6 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.

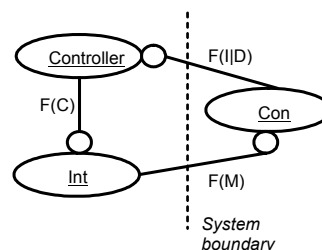


Figure 6: Relationships between Internal Agents and External Agents (Consumers)

3.4 DYNAMICS STRUCTURE – AGENT BEHAVIOUR

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 [21]. In the A Phase it is determined which job is completed next. 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.

4. A CASE EXAMPLE

To illustrate the implementation and use of our modelling framework for manufacturing systems we discuss a simulation model of a small fictitious shop that repairs engines [14]. As a modelling 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 [27]. It proved to be a flexible tool in implementing the concepts included in our modelling framework. Note that an alternative choice of a tool is very well possible due to the general nature of the framework. However, the choice for a tool that is not object-oriented may restrict modeling flexibility - lacking the availability of concepts such as, for example, inheritance.

For modelling the repair shop a class library is built which comprises the class definitions for the three main classes in the modelling framework, i.e., flow items, logistic agents and jobs (Figure 7). These classes are the essential building blocks for the aggregate class RepairShop. Please note that in more complex shops it may be worthwhile to introduce more aggregate classes representing hierarchical levels in modelling. 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.

Let us now discuss 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 8).

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.

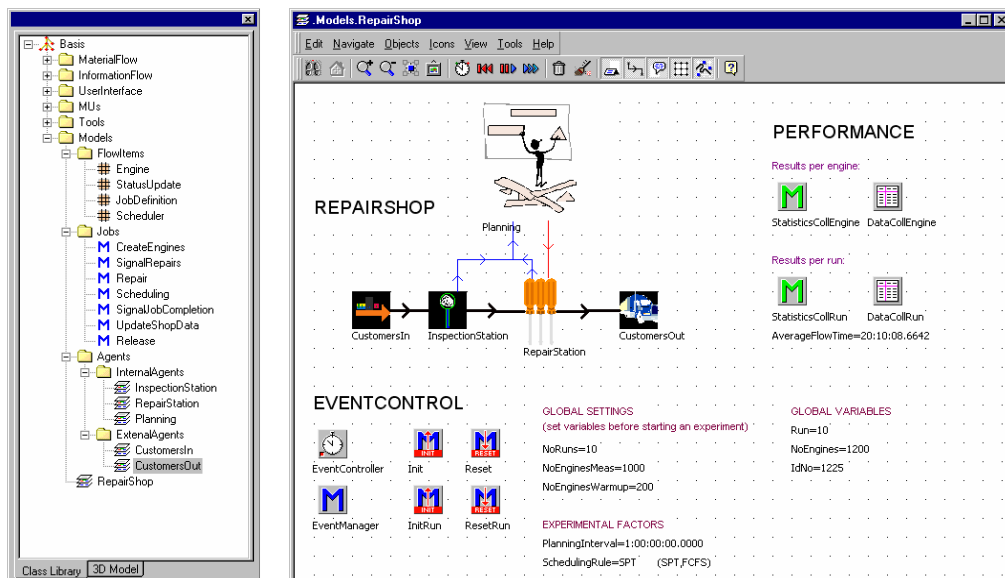


Figure 7: Class Library and Class Repairshop

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: (1) Release – release repair jobs, and (2) Schedule – set up a new schedule of repair jobs on a daily basis. All job classes are implemented in EM-Plant™ Methods, i.e., code.

To show how classes may be linked we will now look at the internal structure and workings of the agent RepairStation (Figure 8). Basically, the class definition covers the functionalities introduced in Figure 3, 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.

5. DISCUSSION

In the previous section we illustrated the implementation of our modelling framework for manufacturing simulation. Let us now discuss its added value for model construction and use.

Guidance in modelling

The modelling framework offers an aggregate approach towards model construction in terms of a well defined conceptual view on the field – building on elementary decomposition principles. Instead of having to create – an implicit – view on a manufacturing system of his own the analyst may start from a clear point of reference. This may result in important time and cost savings in model construction. Also the efforts to be put in remodelling, following from stakeholders' requirements, may be less as the definition of model elements is closer to their mental models of the system under study. Furthermore, we

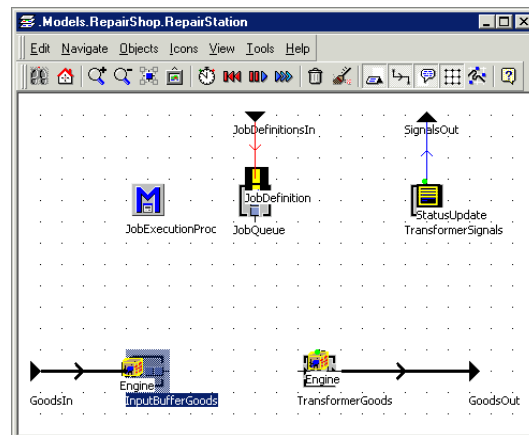


Figure 8: Internal Structure of the Agent RepairStation

found that adapting/re-using models to deal with alternative scenarios is relatively easy. For example, consider the case example described in Section 4:

- Inspection policies; they can be changed by adapting the associated job class. A change may be reflected in terms of the quality of information on repair times.
- The number of repair stations; the reduction or increase of the number of stations is realized by adapting attributes for repair station as well as for the planner.
- Alternative control rules; they can be modelled 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 behaviour of the planning job.

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. For a major part model flexibility as indicated above is the net result of a natural mapping of concepts – knowing where to look and to make the change.

Model completeness and transparency

In order to facilitate communication between analyst and stakeholders model understanding among all parties involved is crucial. Joint understanding helps in model validation/verification and in the creation of alternative solutions. Moreover, it is a

prerequisite for the acceptance of solutions. We found that the importance of these points may be even stronger for supply chains – as understanding may help to build trust among autonomous parties [15]. Cornerstones for building understanding are model transparency and model completeness. Model transparency should result from the limited set of elementary manufacturing concepts offered by the modelling framework – together they should assist in building model structures that appeal to the imagination of all parties involved in the study. On the other hand, model completeness is related to the explicitation of relevant manufacturing objects and their workings. In this respect it is important to note that our modelling framework is an aggregate approach – being founded on the combination of several decomposition principles. It is the combined application of these principles that lies at the basis of more complete models.

6. CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

In this article we discussed a modelling framework for manufacturing simulation as a means for building more communicative models. Here the communicative value is related to the choice of model concepts, and their logic. They should appeal to all parties involved, and should facilitate a joint understanding of the problem at hand. In turn, a joint understanding may contribute to better quality solutions.

The framework presents a software independent view on manufacturing systems. Key concepts in this view are agents and jobs. The framework assumes a uniform definition of jobs. As a net result the concepts of planning and control are naturally embedded by considering decision makers as agents who carry out control jobs similar to machines carrying out physical jobs. The foreseen added value of the modelling framework relates to its role for:

- Model construction: Instead of having to rely on/build up their own implicit reference models, analysts may start from an explicit point of reference. This may reduce efforts (time, money) to be put into modelling.
- Model use: A well-defined, common and appealing jargon helps in building a joint understanding of the model. Moreover, it contributes to quality of solutions.

Obviously, validation of the above benefits requires application of the modelling framework in practical case settings. Some of our recent

experiences with respect to the added value of the framework for model construction are reported in [28]. Here, modelling of food supply chain networks is being considered. An explicit notion of control structures is found to be highly relevant in designing this type of systems. This is not surprising, as control is intrinsic to the supply chain concept. We intend to learn about the communicative value of models built based on our modelling framework – and ways of improving it – by simulation games used for training and educational purposes, see e.g. [29]. Typically, such models assume autonomous user interaction, and may therefore be an adequate testbed.

Some interesting directions for future research include specialization of the modelling framework towards specific industries, and its use for creating manufacturing games, see for example [29,30].

REFERENCES

- [1] R.D. Hurrión: The design, use and required facilities of an interactive visual computer simulation language to explore production planning problems, PhD thesis, University of London, England 1976.
- [2] R.D. Hurrión: “Graphics and interaction”, in: M. Pidd, ed., Computer modelling for discrete simulation, Wiley, Chichester 1989.
- [3] P.C. Bell, and R.M. O’Keefe: “Visual Interactive Simulation - History, recent developments, and major issues” Simulation Vol. 49(3), pp. 109-116, 1987.
- [4] P.C. Bell, C.K. Anderson, D.S. Staples, and M. Elder: “Decision-makers’ perceptions of the value and impact of visual interactive modelling” Omega – The International Journal of Management Science, Vol. 27, pp. 155-165, 1999.
- [5] G. Booch: Object-oriented Analysis and Design with applications, Benjamin Cummings, Redwood City 1994.
- [6] O. Dahl, and K. Nygaard: “SIMULA - an Algol-based simulation language” Communications of the ACM, Vol. 9(9), pp. 671-678, 1966.
- [7] C.R. Glassey, and S. Adiga: “Berkeley Library of Objects for Control and Simulation of Manufacturing (BLOCS/M)”, in: L.J. Pinson, and R.S. Wiener, eds., Applications of Object-Oriented Programming, Addison-Wesley, Reading, Massachusetts, pp. 1-27, 1990.
- [8] W. Kreutzer: “The Role of Complexity Reduction in the Development of Simulation Programming Tools, An Advanced

- Tutorial”, in: Proceedings of European Simulation Conference, Society for Computer Simulation, Delft 1993.
- [9] C.A. Roberts, and Y.M. Dessouky: “An Overview of object-oriented simulation” *Simulation*, Vol. 70(6), 359-368, 1998.
- [10] O. Balci, O.: “Credibility assessment of simulation results”, in: Proceedings of the 1986 Winter Simulation Conference, (IEEE, Piscataway, New Jersey), pp. 38-43, 1986.
- [11] J.H. Mize, H.C. Bhuskute, D.B. Pratt, and M. Kamath: “Modelling of Integrated Manufacturing Systems Using an Object-Oriented Approach” *IIE Transactions*, Vol. 24(3), pp. 14-26, 1992.
- [12] D.B. Pratt, P.A. Farrington, C.B. Basnet, H.C. Bhuskute, M. Kanath, and J.H. Mize: “The separation of physical, information, and control elements for facilitating reusability in simulation modelling” *International Journal of Computer Simulation*, Vol. 4(3), pp. 327-342, 1994.
- [13] D.A. Bodner, and L.F. McGinnis: “A Structured Approach to Simulation Modelling of Manufacturing Systems”, in: Proceedings of the 2002 Industrial Engineering Research Conference, (IIE, Orlando, Florida), 2002.
- [14] S. Galland, F. Grimaud, P. Beaune, and J.P. Campagne: “M(A)MA-L: An introduction to a methodological approach for the simulation of distributed industrial systems” *International Journal of Production Economics*, Vol. 85(1), pp. 11-31, 2003.
- [15] D.J. van der Zee, and J.G.A.J. van der Vorst: “A modelling framework for supply chain simulation – Opportunities for improved decision-making” *Decision Sciences*, Vol. 36(1), pp. 65-95, 2005.
- [16] D.J. van der Zee: “Modelling Decision Making and Control in Manufacturing Simulation” *International Journal of Production Economics*, Vol. 100(1), pp. 155-167, 2006.
- [17] M.E. Porter: *Competitive advantage: Creating and sustaining superior performance*, The Free Press, New York, 1985.
- [18] R.E. Shannon: *Systems Simulation – The Art and Science*, Prentice Hall, Englewood Cliffs 1975.
- [19] T.I. Ören, and B.P. Zeigler: “Concepts for Advanced Simulation Methodologies” *Simulation*, Vol. 32(3), pp. 69-82, 1979.
- [20] C.S. Pegden, R.E. Shannon, and R.P. Sadowski: “Introduction to Simulation Using SIMAN”, McGrawHill, New York, 1990.
- [21] M. Pidd: *Computer Simulation in Management Science*, Wiley, Chichester, 1998.
- [22] P. Lefrancois, and B. Montreuil: “An object-oriented knowledge representation for intelligent control of manufacturing workstations” *IIE Transactions*, Vol. 26(1), pp. 11-26, 1994.
- [23] P. Lefrancois, S. Harvey, B. Montreuil, and B. Moussa: “Modelling and simulation of fabrication and assembly plants: an object-driven approach” *Journal of Intelligent Manufacturing*, Vol. 7, pp. 467-478, 1996.
- [24] B.P. Zeigler: *Object-Oriented Simulation with Hierarchical, Modular Models, Intelligent Agents and Endomorphic Systems*, Academic Press, London 1990.
- [25] T.E. Vollman, W.L. Berry, D.C. Whybark, and F.R. Jacobs: *Manufacturing Planning and Control for Supply Chain Management*, McGraw-Hill, New York 2005.
- [26] W.J. Hopp, and M.L. Spearman: *Factory Physics – Foundations of Manufacturing Management*, Irwin, Chigago 1996.
- [27] A.M. Law, and W.D. Kelton: *Simulation Modelling and Analysis*, McGraw-Hill, Singapore 2000.
- [28] Vorst, J. van der, Tromp, S., and Zee, D.J. van der: “A simulation environment for the redesign of food supply chain networks: integrating quality and logistics modeling”, in: Proceedings of the 2005 Winter Simulation Conference (IEEE, Piscataway, New Jersey), pp. 1658-1667, 2005.
- [29] Zee, D.J. van der, Slomp, J.: “Simulation and Gaming as a Support Tool for Lean Manufacturing Systems – A Case Example from Industry”, in: Proceedings of the 2005 Winter Simulation Conference (IEEE, Piscataway, New Jersey), pp. 2304-2313, 2005.
- [30] M. Holweg, and J. Bicheno: “Supply chain simulation- a tool for education, enhancement and endeavour” *International Journal of Production Economics*, Vol. 78(2), pp. 163-175, 2002.

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 design and use of flexible manufacturing systems.