Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

GUIDING PRINCIPLES FOR CONCEPTUAL MODEL CREATION IN MANUFACTURING SIMULATION

Durk-Jouke Van der Zee

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

Jack G.A.J. Van der Vorst

Operations Research and Logistics Group
Wageningen UR, P.O. Box 8130
6700 EW Wageningen
THE NETHERLANDS

ABSTRACT

Conceptual models serve as abstractions of user's perceptions of a system. The choice and detailing of these abstractions are key to model use and understanding for analyst and project stakeholders. In this article we consider guidance for the analyst in his creative job of conceptual modeling. More in particular, we discuss guidance offered by diagramming techniques and simulation tools. Therefore we "unhide" their underlying engineering principles. The notion of such principles is helpful in judging techniques and tools for their conceptual richness and completeness, in educating simulation engineers and as a basis for a more structured approach towards conceptual model engineering.

1 INTRODUCTION

Conceptual modeling is almost certainly the most important aspect of a simulation project (Robinson 2007). Conceptual models serve as abstractions of user's perceptions of a system – being either real or proposed. Important purposes of conceptual models are: (1) to foster analyst's understanding of a domain (2) to help developers and users to communicate, (3) to serve as a basis for design, and (4) to serve as documentation of the original requirements of the system (Kung and Solvberg 1986). Clearly, quality of conceptual modeling does influence project management (time scales, organization, iterations), project contents (problem definition, model development, experimenting, validation & verification), and – last but not least – project success in terms of credibility and acceptance of solutions. This makes the creation of a conceptual model one of the crucial jobs of the analyst.

Conceptual model creation entails two activities: (1) the identification of the phenomena to be modeled and (2) the mapping of the phenomena into grammar's constructs (Wand and Weber 2002). In the context of simulation, grammar may refer to diagramming techniques, such as, for example, Petri Nets, activity cycle diagrams, flow charts, and – of course – (graphical) representations following the analyst's own definitions and logic. In this article we will mainly address the first phase of conceptual model creation. Our primary focus will be on guidance for the analyst in the development of the conceptual model.

In his survey on conceptual modeling for simulation Robinson (2007) distinguishes between three basic approaches in guiding the analyst: principles of modeling, methods of simplification, and modeling frameworks. *Principles of modeling* advocate an evolutionary development of models – start small and simple, and adapt and extend the model incrementally. On the other hand, *methods of simplification* work the other way around by suggesting ways for model pruning. While both approaches offer relevant assistance for conceptual modeling they do not a-priori address the creation of the conceptual model, i.e., the identification of elementary model components appealing to a domain and to project's stakeholders. *Modeling frameworks* distinguish themselves from the aforementioned approaches by specifying a procedure for detailing a model in terms of its elements, their attributes and their relationships. Examples include the general case of systems representation and domain related cases. The general case of systems representation anticipates conceptualization building on elementary system elements, i.e., components, including their variables and parameters, and mutual relationships, see, for example Shannon (1975). Domain related cases refer primarily to the military field, see for example Nance (1994). Outside this domain, examples are scarce. Guru and Savory (2004) address a framework for modeling physical security systems. Also our previous work on a modeling framework for manufacturing systems (Van der Zee and Van der Vorst 2005, Van der Zee 2006) may be included in this category.

With the exception of the aforementioned modeling frameworks for specific fields of interest, guidance for the analyst's in his creative job of conceptual modeling seems scarce. Not surprisingly, conceptual modeling is often typified as being more of an art than a science. This seems not only true for the simulation field (Shannon 1975), but also for the wider context of systems analysis (Weber 1997, Lieberman 2003). Analyst's "works of art" tend to be guided by their own implicit reference models, concepts underlying diagramming techniques, and the libraries offered by simulation tools. Obviously, next to the skills of the analyst and the tool developer, also the characteristics of the project at hand influence conceptual model quality in terms of an explicit and insightful representation of key decision variables. Here relevant project characteristics include, for example, the available resources, budget restrictions, time horizon, client's interest in modeling, and modeling efforts as they follow from problem complexity.

Clearly, the "implicit" practice of conceptual modeling has an impact on model quality in terms of its transparency and completeness with respect to key decision variables. Hence, model effectiveness may be impaired. However, also modeling efficiency may be harmed as this implicit practice does not directly contribute to, for example, education, i.e., analysts' or stakeholders' learning from each other experiences, model re-use, and the setup of robust simulation libraries. Moreover, the interest in innovative chain configurations in business, such as for example, supply chains, health care chains, and transportation networks, stress the importance of high quality conceptual modeling. This follows both from systems' complexity, and the presence of multiple autonomous stakeholders. System complexity of chain configurations sets higher demands on an insightful representation of model elements, both for the analyst and the stakeholders – in their strengthened role as domain experts. At the same time the presence of multiple autonomous stakeholders puts pressure on a joint understanding and creation of solutions. This implies the need for a shift from *communicative* models (Balci 1986) to *participative* models (Van der Zee 2007).

In this article we attempt to evaluate the current practice of conceptual modeling for the manufacturing field to some extent. We do so by assessing the choice of model components for diagramming techniques, and simulation tools. We relate this choice of components to their underlying model decomposition principles. Typically, diagramming techniques, and simulation tool libraries are built starting from these principles, which should relate to a specific field of interest. Unhiding these principles is assumed to:

- Create awareness on the way conceptual modeling may be supported and steered by diagramming techniques and simulation tools.
- Assist in understanding applicability and shortcomings of diagramming techniques and simulation tools with respect to conceptual modeling.
- Provide an initial set of decomposition principles which may be communicated and applied for practical or educational use.
- Establish a preliminary basis for creating, improving and extending conceptual frameworks underlying simulation tools and diagramming techniques.

As a first step in our evaluation we categorize engineering principles underlying conceptual modeling (Section 2). Next, in Section 3, we consider a few well-known diagramming techniques and illustrate their support for conceptual modeling by highlighting their underlying decomposition principles. In a similar way we consider two simulation packages (Section 4). Starting from the observations in Section 3 and 4, in Section 5, we review literature as it critisized tools and models for conceptual richness, and completeness. This results in the identification of an other set of decomposition principles, that is somewhat tailored to the manufacturing field. On the other hand, an explanation is offered for the conceptual model quality as it results from the dominating implicit approach. Section 6 introduces an alternative, structured, approach towards conceptual model engineering. Finally, main conclusions are summarized in Section 7.

2 CONCEPTUAL MODEL CREATION – CATEGORIZING ENGINEERING PRINCIPLES

In this section we briefly categorize principles underlying conceptual model creation. Two cognitive approaches are suggested as being fundamental in the process of identifying the phenomena to be modeled (Burton-Jones and Meso 2006):

- Decomposition: *breakdown* of a system into semi-independent components corresponding to its many functional parts (Simon 1996).
- Classification: *grouping* of similar types of objects (Parsons 2002).

Basically, decomposition starts from a top down approach towards modeling, while classification works the other way around. Essentially, classification boils down to an abstraction of similarities among individual things for supporting cognitive economy, inference, and communication, see, for example, Smith and Medin (1981).

Starting from these fundamental approaches other approaches may be adopted to refine model engineering. We mention the principles of modeling, and methods of simplification introduced in Section 1. They may be helpful, in choosing the right amount of detail, and to tailor model scope. Further Liebermann (2003) mentions the possibility of obscuring or emphasizing model components. Obviously, the application of suchlike principles assumes a thorough notion of the stakeholders' objectives, both with respect to system analysis and visualization. In this article we restrict ourselves to the evaluation of the fundamental principles for simulation model engineering.

3 BASIC PRINCIPLES – DIAGRAMMING TECHNIQUES

Diagramming techniques are often used as the initial (home made) grammar for representing and visualizing a system. Several techniques exist, see Pooley (1991) for a first introduction. In this section we will consider just three of these techniques: Activity Cycle Diagrams, Petri Nets and Flow charts. Reasons for choosing just these techniques lie in the fact that they are well-known, and address the general systems case. With this choice of techniques we aim to contribute to simple and insightful reasoning. For all three techniques multiple implementations exist – either formalized in literature or not. Hence, they should be considered as families of techniques, rather than being characterized as stand-alone. In this section we will discuss one specific implementation for each technique.

As a starting point for our discussion we consider a small case example – a simple job shop. The job shop is modeled by applying each of the three diagramming techniques, and illustrates model component use and meaning, see Figure 1. Next, we consider each technique in somewhat more detail, and unhide their underlying decomposition principles, by inference from their choice of components.

The job shop may be characterized by three entities: (1) products, arriving according to an irregular pattern, (2) a single machine, capable of processing all types of products, and (3) an operator, being responsible for machine setups. The machine may require maintenance. This need is to be established every time processing for a product has been completed. The machine has to be setup for each individual product. However, machine setups are not sequence dependent, i.e., do not depend on the type of product to be processed. The operator also has specific tasks at other departments.

Figure 1a represents the job shop by an activity cycle diagram. Activity cycle diagrams make use of two symbols: the dead state, i.e., waiting for some condition to be fulfilled, and the active state, i.e., some activity other than waiting. Activities for each entity are described in terms of a cycle of these states. Relationships between entities follow from their cooperation in active states. For example, the execution of an operation requires the availability of both a product and a machine. Use of Activity cycle diagrams is advocated in case of a system with a strong queueing structure (Pidd 1998).

An alternative way of modeling of the job shop is shown in Figure 1b. Here a Petri Nets formalism is used for modeling the job shop. Petri Nets consist of four types of components: (1) movable entities called “tokens”, (2) transitions, concerning transformations of tokens, (3) places, for temporarily storing tokens, and (4) arcs, for connecting places and transitions. In Figure 1b, tokens depict products, the machine and the operator, while transitions model shop activities. Further, Petri Nets foresee in an explicit notion of dynamics, i.e., transitions are allowed to execute (“fire”) if tokens are available at all its input places. For example, the distribution of tokens in Figure 1b allows for a start of the “operation”. Note how this rule also implies the need for an additional transition for modeling machine setup. Many alternative choices are described in literature as far as the definition of Petri Nets’ components and their dynamics are concerned, see Murata (1989) for an introduction. Here we consider timed Petri Nets, where times are associated with the duration of a transition.

Flow charts, or logic flow diagrams, model the logic of the system rather than the process flow (Robinson 2004). Figure 1c models shop behavior in case of a product arrival, as a sequence of decisions and processes. Note that this is a partial model. Similar diagrams may be developed for the case a machine completes service, a machine has been repaired or setup, and the arrival of an operator.

In Table 1 we give an overview of model components for the three diagramming techniques. We use this overview to infer seven underlying decomposition principles,

Table 1: Choice of model components for diagramming techniques

Model Components / Diagramming Technique	Entities	Activities	Relationships	Dynamics
Activity cycle diagrams	Entity – cycles between dead state and life state	Dead state – waiting Active state – processing	Arcs - joint active states	
Petri Nets	Token – input and output of a transition Transition – logically fixed operation; Place – storage for tokens	Place – waiting Transition – processing	Arcs – joint transitions	Firing – transitions start if tokes are present in all its input places
Flow charts		Decision making Processing		

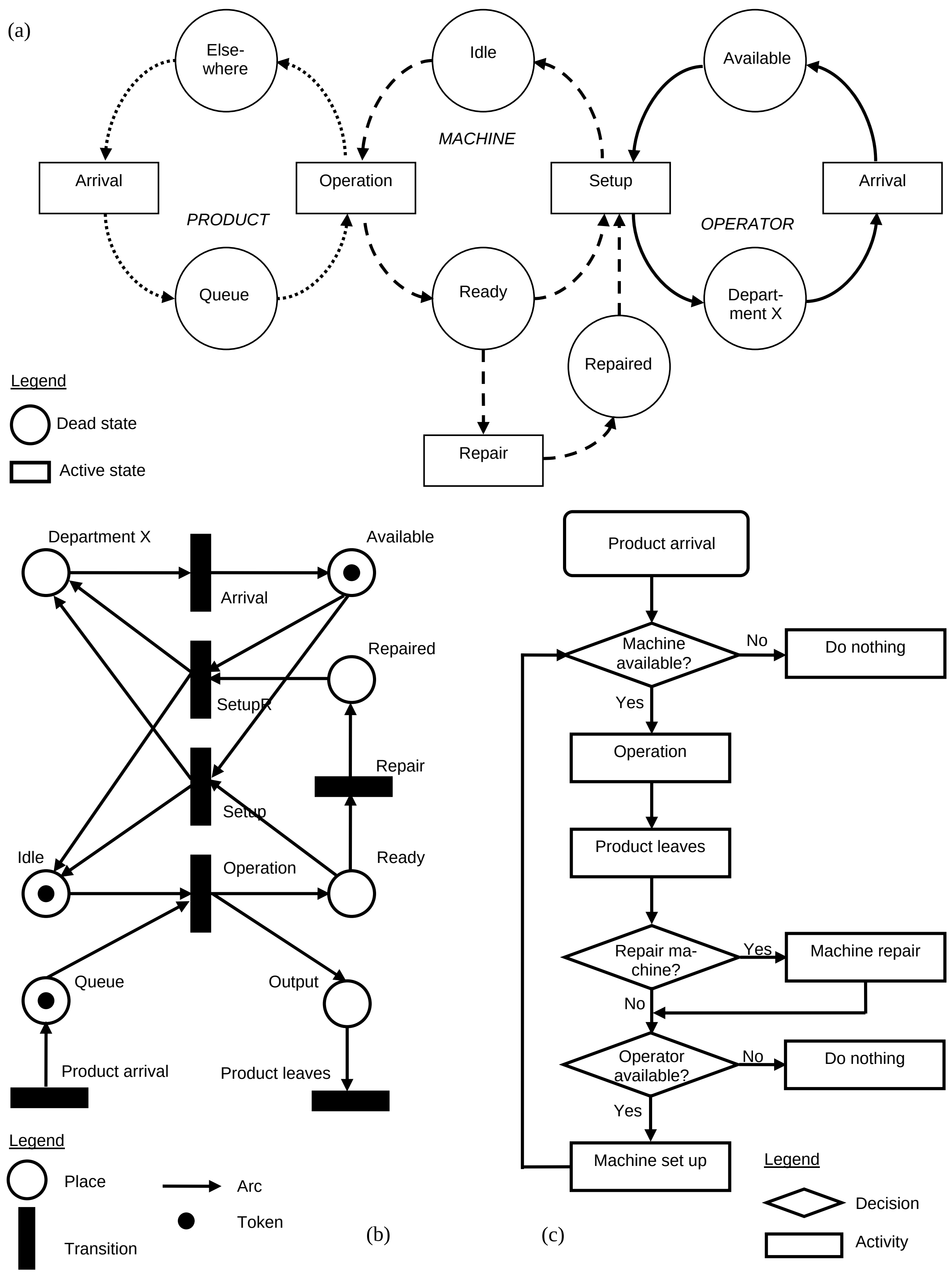

Figure 1: Diagrams for job shop example

Table 2: Relating model components to decomposition principles underlying diagramming techniques

Decomposition principles		Activity Cycle Diagrams	Petri Nets	Flow Charts
I	External and internal entities		X	
II	Movable and non-movable entities		X	
III	Active and passive entities		X	
IV	Waiting and processing	X	X	
V	Decision making and processing			X
VI	Relationships build on joint activities	X	X	
VII	Dynamics follows from presence of input for processes		X	

see Table 2. Both Tables 1 and 2 start from the explicit (visual) notion of model components.

Let us now discuss the decomposition principles in somewhat more detail. For Petri Nets, the concept of a *system boundary* (I) is made explicit. For example, the transition Product arrival acts as a *source* of products, while the transition Product leaves represents an external entity of type *sink*. Further, a distinction can be made between *movable and non-movable entities* (II). Where tokens in Petri Nets depict movable entities, transitions may present infrastructural entities. Remark, that infrastructure is related to the setup of the diagram, not a-priori to the physical or logic underlying the system. Both movable entities and non-movable entities may be further classified as being *active*, i.e., initiating activities, or *passive*, i.e., being acted upon (III). Here transitions may be considered active, while places and tokens are passive.

In both activity cycle diagrams and Petri Nets activities *waiting and processing* (IV) are distinguished. Recall, how this makes them fit for modeling queueing systems. Alternatively, for flow charts, it is distinguished between *decision making and processing* (V).

Relationships between entities follow from their *joint participation in activities* (VI). They are expressed by arcs. Finally, Petri Nets foresee in an explicit mechanism for representing *dynamics, through the "firing" of transitions* (VII).

Table 2 makes clear how each diagramming technique builds on a subset of decomposition principles. The inclusion of specific decomposition principles may make it more or less fit for specific applications. For example, the explicit notion of decisions may make a flow chart a suitable vehicle for modeling a decision algorithm, but less suitable for modeling process flow. On the other hand, at first sight, Petri Nets may offer advantages in case of a dynamics representation of the system. A more general observation on the use of these diagramming techniques concerns their contextual richness. As they build on a limited set of simple components their size may easily explode for real-life case settings. Consequently, diagrams often have to be enriched with supplemental texts or by modifications, following from the application of additional decomposition or classification principles. Next to model size, also model complexity may hinder practical use. For example, Kamper (1991), and Ryan and Heavey (2006) indicate that for realistic systems Petri Nets make it difficult for non-experts to reason with the logic contained within the model.

4 TAILORED PRINCIPLES – SIMULATION PACKAGES

Next to diagramming techniques also simulation tools may guide conceptual modeling. This may happen in an indirect way by relating the development and definition of initial diagrams to the foreseen tool for model coding, as well as in a direct way - in case simulation tools are used as a first basis for setting up preliminary (conceptual) models. In this section we assess underlying decomposition principles for simulation tools. We do so not with the idea of a full coverage of all tools, or even of a single tool. Clearly, the abundance of available simulation packages (Swain 2005), and their individual world view and level of detail hinders this. Rather, we seek to illustrate how simulation tools further detail system decomposition, using an additional set of principles, which is tailored to the field of interest. Therefore we studied just two of these packages: EM-Plant^{TM} and Enterprise Dynamics^{TM}. Both packages are object oriented and are widely applied in manufacturing practice. As findings for both packages are rather similar we condense our discussion here by only highlighting observations on EM-Plant^{TM}.

To infer basic decomposition principles underlying decomposition principles for EM-Plant^{TM} we studied the components of its so-called basic library, its introductory manual (Tecnomatix 2003) and its help function. The basic library is the set of building blocks that is used as a start for all model development, including dedicated libraries – tailored to a specific field of interest. Table 3 shows how different types (classes) of entities are foreseen. In turn, each of these classes my be further decomposed in subclasses. Entity descriptions are derived from the definitions supplied by the tool builders. Original definitions are somewhat condensed in order to meet space requirements, and clarity in typifying entities. Note how we do not distinguish between classes and their graphical representation. In principle, class definitions and graphical representations, may concern alternative grammars for conceptualization, for which a one to one mapping is not always guaranteed.

Table 3: Decomposition of entities for EM-Plant^{TM}

EM-Plant^{TM} Entity decomposition			
Material flow: objects representing, processing, transportation or storage of materials	Moving units (MUs): physical or logical parts being transported or processed by material flow objects	Active units: move on its own	Transporter: object with a propulsion system allowing it to move on its own on tracks. It also has loading capacity for transporting MUs
		Passive units : parts being acted upon	Entity: parts being produced and transported, but not transporting other work pieces Container: object for transporting other Mus
	Fixed units: work stations in a factory	Active units: transport or process parts	Source: produces MUs in a single station Drain: a single processing station for destroying MUs SingleProc: a single station for processing an MU. ParallelProc: concerns several stations for processing MUs in parallel Assembly: adds mounting parts to a main part DismantleStation: removes mounting parts from the main MU or it creates new MUs Buffer, PlaceBuffer: facility for temporary starage of MUs. Sorter: sorts the MUs located on it according to sort criteria Line: represents a conveyor system or a part thereof
		Passive units: store, route or display parts	Store: stores any number of MUs you define Track: represents part of a transport line, with or without automatic routing FlowControl: allows to model common strategies for diverging and converging the flow of materials Cycle: synchronizes the transfer of MUs from station to station
Information flow: objects for providing the simulation run with information	Controls: Objects for programming controls and for commenting them	Methods: program controls that other objects start Variable: global variable that other objects and methods can access during a simulation run Comment: explanatory notes to a simulation model	
	Lists and tables: data storage	TableFile, CardFile, StackFile, QueueFile: store data in alternative formats TimeSequence: controls and administers time-related values, such as shift plans or machine maintenance	
	Time patterns: when and how moving objects are created	Generator: time based activitation of methods ShiftCalendar: model shift patterns in systems Trigger: changes values of attributes and global variables according to a custom pattern	
	Interfaces: Objects that allow to exchange data with other applications	XMLinterface, FileInterface, FileLink: alternative types of interface	
Resources: objects for representing worker and worker activities	Moving units (MUs): workers being idle or at work	Worker: a movable working person who performs a job on a work place	
	Fixed units: work stations in a factory	Footpath: models walking distances between the WorkerPool and work stations or between work stations Worker pool: contains idle workers Work place: actual place at the station, where the worker performs his job Exporter, Broker: assign workers to stations	

As a first step in model decomposition EM-Plant^{TM} distinguishes between three elementary modalities, i.e., types of flow, for manufacturing systems: material flow information flow, and resources, see Table 3. For the material flow, further decomposition is realized by applying ba sic principles I-III, see Table 2. This is illustrated in Figure 2, which relates choice of entities to decomposition principles (in italics). Next, applying the concept of classification, results in a number of subclasses per functional class. For resources a similar set of principles is used. Note how resources are restricted to workers, and their associated infrastructure, and control.

For the information flow, alternative decomposition principles are applied. Where material flow entities appeal to the physical elements of a manufacturing system, the choice of information flow entities seems to be largely motivated by a general systems focus. This follows from entities, that may be characterized as programming facilities, rather than building blocks tailored to the manufacturing field. Next to the direct access to the internal language, i.e., SimTalk^{TM}, via Methods, there is the explicit notion of data storage in alternative formats. Further, direct manipulation of the event list is foreseen for activating entities, and code, for example, to model shift patterns. Finally, the library contains interfaces with external software.

Objects representing alternative modalities

- Material flow
  - *Movable and non-movable entities (II)*
  - Movable entities
    - *Active and passive entities (III)*
    - Active entities
      - *Objects with own propulsion*
    - Passive entities
      - *Object being produced and transported*
      - *Objects for transporting other objects*
  - Non-movable entities
    - *Active and passive entities (III)*
    - Active entities
      - *External and internal entities (I)*
      - External entities
        - *Sources and sinks*
      - Internal entities
        - *Single station and parallel stations*
        - *Type of dominant operation: Processing, (dis)assembly, storage, transport, sorting*
    - Passive entities
      - *Type of function*: store, display, route
- Information flow
  - *Control logic, list and tables, time patterns, and interfaces*
  - Control logic
    - *Elements for model control: Methods, variables and comments*
  - Lists and tables
    - *Alternative means for data storage within model context*
  - Time patterns
    - *Alternative types of time patterns*
  - Interfaces
    - *Alternative types of interface with external software*
- Resources
  - *Movable and non-movable entities (II)*
  - Movable entities
    - *Workers, movable working persons*
  - Non-movable entities
    - *Elements for modeling worker infrastructure: Foot path, worker pool, work place, and assignment controls*

Figure 2: Relating choice of entity for EM-Plant^{TM} to decomposition principles

For material flow EM-Plant^{TM} distinguishes between three basic types of activities: production, transport and storage. In principle, most of the fixed units, see Table 3, may undertake each of these activities – as made explicit in performance registration. However, initialization for each entity allows for a specific choice of activity. For example, a SingleProc is associated with production, while a Store is related to storage. This implies an initial entity classification. For resources similar types of activities can be distinguished. However, they are restricted to a specific type of resource, i.e., the worker.

Decision making activities may be directly related to each fixed material flow entity through entrance and exit controls. These controls allow for calling on a method, i.e., code based on the internal language. Also standard or user-defined priority rules for dispatching may be embedded in material flow entities, for example, to sort or route movable entities.

For information flow there is no clear notion of activities. This follows from the building blocks being programmers' facilities rather than manufacturing systems' elements, also see above.

Entity relationships may be specified for fixed material flow units. Relationships are founded on the exchange of movable units. Here types of movable units exchanged determine the relationship.

Basically, model dynamics is related to the input of movable material flow elements at fixed material flow entities. Typically, this starts an activity. Note how activities include the possibility of waiting for a next activity. Other means to start activities are time patterns, for example worker shifts. While time patterns have an explicit notion, being part of the library, also "implicit" access to the event list is possible through model code.

5 AN INITIAL ASSESSMENT OF CONCEPTUAL MODEL CREATION

In previous sections we discussed a few examples of diagramming techniques and their underlying decomposition principles. In this section we relate these initial observations to criticism in literature on manufacturing simulation models' conceptual richness and completeness. We have two objectives in doing so: (1) explain criticism building on a more thorough notion of the creative principles underlying conceptual modeling (2) highlight additional principles that address the criticism.

Mize et al. (1992) point at the fact that traditional languages do not provide natural constructs for separately and distinctly modeling the three types of basic functions, i.e., physical, information and control/decision functions. In addition, the constructs provided for information and control are often hard coded and dispersed in the model (Pratt et al., 1994). 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 modeling environment as the modeler is forced to think about model elements independently. It is interesting to see how EM-Plant^{TM} distinguishes between modalities. However, the notion of the information flow – and especially – the decision flow is weak – leaving conceptual modeling to the model builder rather than offering guidance.

According to Lefrancois and Montreuil (1994), and Lefrancois et al. (1996) a *distinction between intelligent and non-intelligent entities* permits a more natural and richer presentation and implementation of systems modeled. In such a context, intelligent beings are modeled as agents. Agents are used to implement decision rules inherent to manufacturing system planning and control. Examples include routines for scheduling, dispatching and releasing jobs for a machine or department. According to Lefrancois and Montreuil (1994), and Lefrancois et al.

(1996) workstations and work orders, are assumed to be non-intelligent. Note how their work assumes a further detailing of the concept of passive and active entities with respect to decision making. Such detail is not met by the diagramming techniques and simulation tools studied here.

Van der Zee and Van der Vorst (2005) stress the separate and explicit notion of activities in simulation modeling. They advocate to *denominate all activities – including information processing and decision-making – in the manufacturing system as "jobs"* The job concept is meant to bring two important advantages. Firstly, 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. Secondly, an explicit notion and allocation of company activities, increases visibility and traceability of decision variables. Note how the modeling of decision logistics and supportive information processing is left largely to the analyst. Only basic support is offered for common dispatching rules.

6 DISCUSSION – OFFERING STRUCTURE IN CONCEPTUAL MODEL CREATION

The evaluation of diagramming techniques and simulation tools with respect to their conceptual backing brings us: (1) an initial insight in principles for model decomposition and classification, and (2) the logic of their application in terms of model completeness, and richness. Clearly, the explicit notion of such principles has a direct value for educating simulation engineers in setting up conceptual models. Moreover, it points at the need for a structured approach towards conceptual model creation, see Figure 3.

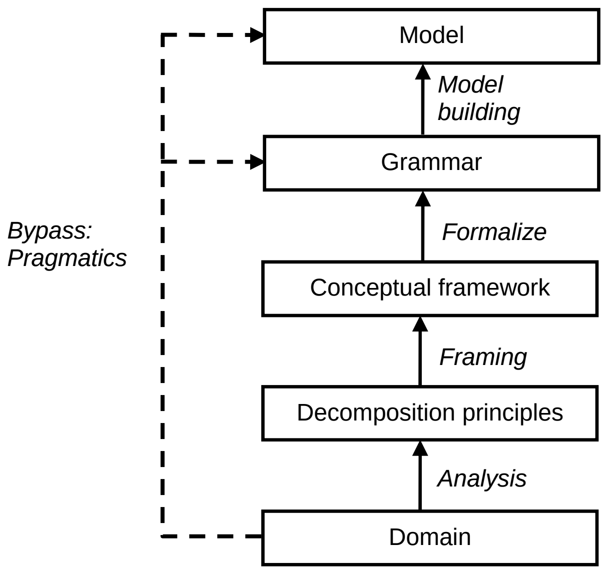

Figure 3: Conceptual model engineering

Figure 3 shows how conceptual model creation may be subdivided in 4 engineering steps:

1. Domain analysis, resulting in definitions of principles for model decomposition and classification.
2. Framing, i.e., combining principles to develop a conceptual framework for defining model entities, activities, relationships, and their dynamics.
3. The conceptual framework should be the basis for defining a grammar, i.e., a technique supporting an explicit (graphic) model notion.
4. Model building, i.e., the application of the grammar for a specific case example.

At the same time, the figure clarifies the current implicit practice in terms of a pragmatic bypass of the aforementioned steps.

7 CONCLUDING REMARKS

In this article we considered guidance for the analyst in his job of conceptual model creation. Starting from the observation that domain specific guidance in this respect is scarce, we evaluated a few basic diagramming techniques and simulation tools for their underlying principles for model decomposition and classification. Where literature critisizes manufacturing simulation models for conceptual richness and completeness, unhiding these principles helps in an in-depth explanation and adjustment of their shortcomings. Moreover, the explicit notion of principles for conceptual creation may be helpful in simulation engineering education. It is suggested that such an education should include a more structured approach towards conceptual model engineering. Such an approach distinguishes itself from many of the current implicit approaches, which may be characterized by a "pragmatic bypass" of key phases in conceptual model structuring, i.e., the isolation of relevant principles for conceptual model engineering, and their framing (combination) to make up a complete and rich conceptual framework underlying model grammar.

REFERENCES

Balci O (1986). Credibility assessment of simulation results. In *Proceedings of the 1986 Winter Simulation Conference*, 209-222. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Burton-Jones, A., and P.N. Meso. 2006. An Empirical Test of Decomposition Principles in Object-Oriented Analysis. *Information Systems Research* 17(1): 38–60.

Ryan J., and C. Heavey. 2006. Process modeling for simulation. *Computers in Industry* 57: 437-450.

Guru, A., and P. Savory. 2004. A Template-Based Conceptual Modeling Infrastructure for Simulation of Physical Security Systems. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R.G. Ingalls, M.D. Rossetti, J.S. Smith, and B.A. Peters, 866-873. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kamper, S. 1991. On the Appropriateness of Petri Nets in Model Building and Simulation. *Systems Analysis Modelling Simulation* 8(9): 689-714.

Kung, C. H., and A. Solvberg. 1986. Activity modelling and behavior modelling. In *Information Systems Design Methodologies: Improving the Practice*, ed. T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, 145-171, Amsterdam, The Netherlands: IFIP, North-Holland.

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

Lefrancois P., S. Harvey, B. Montreuil, and B. Moussa. 1996. Modelling and simulation of fabrication and assembly plants: an object-driven approach. *Journal of Intelligent Manufacturing* 7: 467-478.

Lieberman, B. 2003. The art of modeling: Part 1: Constructing an analytical framework. *Rational Edge* (August). Available via <http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/aug03/f_modeling_bl.pdf> [accessed March 26, 2007]

Mize, J.H., H.C. Bhuskute, D.B. Pratt, and M. Kamath. 1992. Modelling of Integrated Manufacturing Systems Using an Object-Oriented Approach. *IIE Transactions* 24(3): 14-26.

Murata, T. 1989. Petri Nets : Properties, Analysis and Applications. *Proceedings of the IEEE* 77(4): 541-580.

Nance, R.E. 1994. The Conical Methodology and the Evolution of Simulation Model Development. *Annals of Operations Research* 53: 1-45.

Parsons, J. 2002. Effects of local versus global schema diagrams on verification and communication in conceptual data modeling. *Journal of Management Information Systems* 19: 155–184.

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

Pidd, M. 1998. *Computer Simulation in Management Science*. 4th ed. Chichester: Wiley.

Pooley, R.J. 1991. Towards a Standard for Hierarchical Process Oriented Discrete Event Diagrams. *Transactions of the SCS* 8(1): 1-41.

Robinson, S. 2004. Simulation – The practice of model development and use. Chichester: Wiley.

Robinson, S. 2007. Conceptual modelling for simulation Part I: definition and requirements. *Journal of the Operational Research Society* (in press).

Shannon, R.E. 1975. *Systems Simulation – The Art and Science*. Englewood Cliffs :Prentice Hall.

Simon, H. A. 1996. *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

Smith, E., and D. Medin. 1981. *Categories and Concepts.* Cambridge: Cambridge University Press.

Swain J.J. 2005. 'Gaming' Reality- Biennial survey of discrete-event simulation software tools. *OR/MS Today* 32(6), 44-55.

Tecnomatix. 2003. *EM-Plant 7.0 – Step-by-Step*. Stuttgart: Tecnomatix.

Van der Zee, D.J. 2006. Modeling Decision Making and Control in Manufacturing Simulation. *International Journal of Production Economics* 100(1): 155-167.

Van der Zee, D.J. 2007. Developing participative simulation models – framing decomposition principles for joint understanding. Journal of Simulation (accepted for publication)

Van der Zee, D.J., and J.G.A.J. van der Vorst. 2005. A modeling framework for supply chain simulation – Opportunities for improved decision-making. *Decision Sciences* 36(1): 65-95.

Wand, Y., and R. Weber. 2002. Information systems and conceptual modeling—A research agenda. *Information Systems Research*. 13(4) 363–376.

Weber, R. 1997. Ontological Foundations of Information Systems. Melbourne, Australia: Coopers & Lybrand and Accounting Assoc. of Australia and New Zealand.

AUTHOR BIOGRAPHIES

DURK-JOUKE VAN DER ZEE is an Assistant Professor in the department of Production Systems Design at the Faculty of Management and Organization, University of Groningen, The Netherlands. He 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 and flexible manufacturing systems. He can be contacted by email at <d.j.van.der.zee@rug.nl>

JACK G.A.J. VAN DER VORST is Professor in Logistics and Operations Research at Wageningen University and Research center (WUR) in The Netherlands. His Ph.D. research concerned the assessment of food supply chain redesign strategies using simulation and pilot studies. His research interests include supply chain management, logistics, performance measurement systems, and simulation methodology and applications. He can be contacted by email at <jack.vanderVorst@wur.nl>