2006-01-01

Process Modeling for Simulation

John Ryan
Technological University Dublin, john.ryan@tudublin.ie

Cathal Heavey
University of Limerick

Recommended Citation

Ryan, J., Heavey, C.:Process Modeling for Simulation. Computers in Industry, Volume 57, Issue 5 (2006)pp. 437-450. doi:10.1016/j.compind.2006.02.002

Process Modeling for Simulation

John Ryan Dr.* Cathal Heavey Dr.†

*DIT, john.ryan@dit.ie

†University of Limerick, cathal.heavey@ul.ie

— Use Licence —

Attribution-NonCommercial-ShareAlike 1.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

- Attribution.
  You must give the original author credit.
- Non-Commercial.
  You may not use this work for commercial purposes.
- Share Alike.
  If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the author.

Your fair use and other rights are in no way affected by the above.

Process Modeling for Simulation

John Ryan [b] Cathal Heavey [a,*]

[a] *Department of Manufacturing and Operations Engineering, University of Limerick, Limerick, Ireland. Email: cathal.heavey@ul.ie*

[b] *School of Hospitality Management and Tourism, Faculty of Tourism and Food, Dublin Institute of Technology, Cathal Brugha Street, Dublin 1, Ireland. Email: john.ryan@dit.ie*

Abstract

This paper discusses shortfalls in relation to the requirements gathering phases of simulation. While many developments have taken place around supporting the model coding task of simulation, there are few tools available to assist in the requirements gathering phase. This is surprising as it has been reported by several researchers that the requirements phase can absorb twice as much resources as the coding phase. There are numerous process modeling tools available (over 100) that can and have been used to support the requirements phase of simulation. This paper provides a selective review of some of the most important in relation to simulation. A conclusion from this review is that none of the tools available adequately supports the requirements gathering phase of simulation. It is proposed that a process modeling tool be developed specifically to support simulation requirements gathering. The design objectives in the development of the tool are: (1) It should be capable of capturing a detailed description of a Discrete Event System; (2) It should have a low modeling burden and therefore be capable of being used by non specialists; (3) It should present modeling information at a high semantic level so that manufacturing personnel can rationalize with it; (4) It should have good visualization capabilities; (5) It should support project teamwork. Based on these design objectives a proposed simulation process modeling tool called Simulation Activity Diagrams (SAD) is presented.

Key words: Process Modeling, Simulation.

* Corresponding author.

1 Introduction

Most systems can be viewed as Discrete Event Systems (DES) e.g. manufacturing systems, business processes, supply chains. These systems are complex and difficult to both understand and operate efficiently. Because of its great versatility, flexibility, and power, simulation is one of the most widely used operations research techniques (Shannon et al. 1980). While simulation, in theory, has great potential to assist in the understanding and efficient operation of these systems, several studies show that there is a low usage of simulation by industry. An extensive study of the penetration and use of discrete event simulation in the UK manufacturing industry identified only 11% of sites out of a sample of 431 which were currently utilizing simulation as a decision support tool (Hollocks 1992). The ESPRIT working group on Simulation in Europe (SiE) subsequently expressed the view that this general picture of proliferation was reflected across Europe (Kerckhoffs et al. 1995). This view of the penetration of simulation into industry is also supported by a more recent survey covering non-academic members of INFORMS (Abdel-Malek et al. 1999).

One possible difficulty is the current way simulation is implemented, with simulation modeling very often becoming a heavy programming task with the inner workings of a system being lost in the detailed programming code and only visible to those intimately involved in the programming task. There are several problems with this current situation:

(1) Very valuable information concerning the operation of a system is lost in the detailed simulation code. While simulation will provide quantitative information a lot of insight can be obtained by studying in-depth how a DES operates;
(2) Managing a DES is a team activity. Simulation modeling is a very poor tool for communication and supporting teamwork. While it provides quantitative information and provides a certain level of visualization through animation, it is too specialized an activity to facilitate modeling by teams, consensus building, group understanding and visualization.

This paper presents initial research into developing support for the pre-coding activities of a simulation project and argues for increased research into this area.

2 The Process of Simulation

When conducting a simulation project it is recommended that a structured systematic approach be carefully planned and rigidly adhered to. The "40-20-40" rule is a widely quoted rule in simulation texts. The rule states that, in developing a model, an analyst's time should be divided as follows (Sheppard 1983):

(1) 40% to requirements gathering such as problem definition, project planning, system definition, conceptual model formulation, preliminary experiment design and input data preparation;
(2) 20% to model translation;
(3) 40% to experimentation such as model validation and verification, final experimental design, experimentation, analysis, interpretation, implementation and documentation.

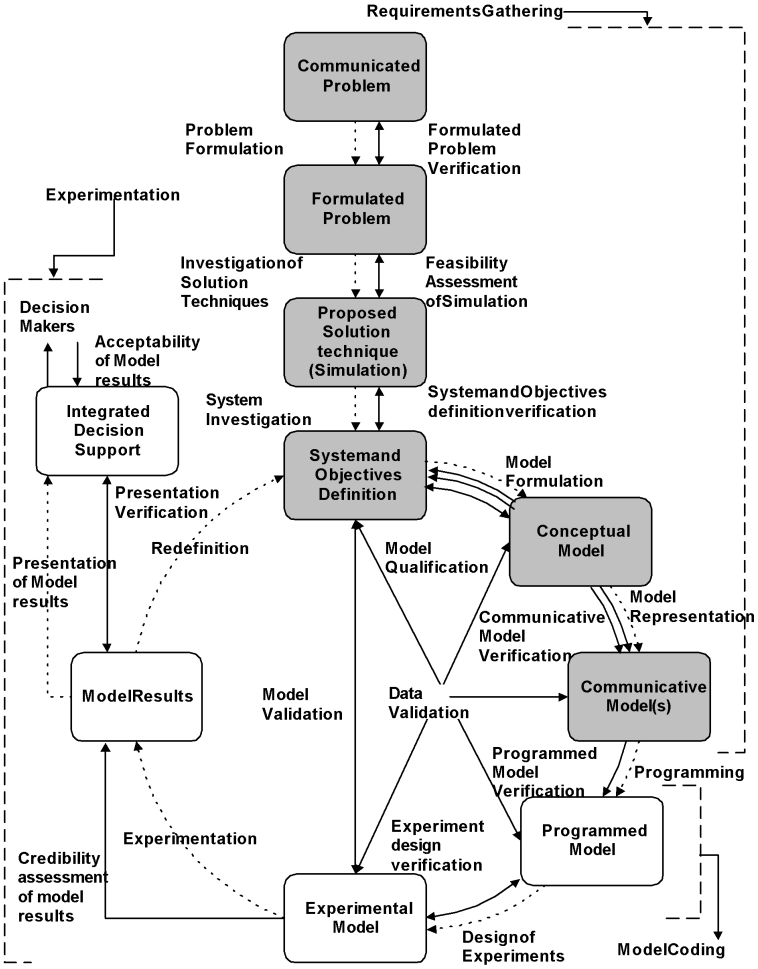


Fig. 1. The Life cycle of a simulation Study (Pidd 1989).

It is rare for these phases to be totally independent. For example, in the requirements gathering phase one would consider programming implications. The model developer would also make an effort to program the simulation model in such a way as to allow for easy and accurate experimentation. Figure 1 shows in more detail the tasks involved in simulation modeling. A lot of these tasks take place prior to the coding phase of a project and may be repeated at different stages of the project depending on model revisions. Many developments have taken place around supporting the "model coding or translation task" of a simulation model with highly developed modeling environments now available. But there has been very little research into develop support for tasks prior to coding (Robinson 2004).

Process modeling tools have been used to support tasks prior to coding in a simulation project. However, none of these tools have been developed specifically to support simulation. The next section briefly reviews a number of these tools highlighting their limitations in supporting simulation.

3 A Brief Review of Process modeling for Simulation Support

There are numerous process modeling tools available to aid in the modeling of a system. Kettinger et al. (1997) listed over 100 in a survey that was not exhaustive. These tools are capable of modeling many different aspects of a system to varying levels of detail. Some of these tools allow simulation of process models developed within the tool (i.e., Scheer (1998), Mayer et al. (1995)) and INCOME Process Designer (2005)) and a number have been used to support simulation.

Nethe and Stahlmann (1999) suggest that the development of a high-level process model of an actual production system prior to the development of the simulation model could greatly help in the collection of relevant information on the operation of the system (i.e. data collection) and, therefore, reduce the effort and time consumed to develop a simulation model. A number of researchers have shown that methods from the IDEF approach could be used to support simulation. For instance, Jeong (2000) used both IDEF0 and IDEF3 to develop an optimized simulation-based scheduling system, while Perera and Liyanage (2000) used IDEF0 and IDEF1X to address the rapid collection of input information for the simulation of manufacturing systems. Also, other researchers such as van Rensburg and Zwemstra (1995) and Al-Ahmari and Ridgway (1999) have demonstrated the use of IDEF0, IDEF1X and IDEF3 to support simulation for manufacturing and system design. Furthermore, it has been suggested by van Rensburg and Zwemstra (1995) that the use of a processing modeling tool in simulation modeling enhanced the quality of simulation models and helped to reduce the time needed to generate simulation

models.

This section provides a selective review of process modeling tools. We focus on methods/tools that have been used to support simulation and that exhibit characteristics desirable in a dedicated process modeling tool for simulation. For example, we include Role Activity Diagrams (RAD) because they model the interaction of users within a system and the GRAI method as it models decision making. There are numerous ways of categorizing the methods/tools. We use the categorization:

Formal Methods: These are methods that have a formal basis and there are numerous software implementations of these methods;

Descriptive Methods: Methods that have none or very little formal basis and are primarily software implementations.

3.1 Formal Methods

3.1.1 Petri Nets

Petri nets are a mathematical formalism based on a few simple objects, relations and rules capable of representing very complex systems. Petri nets have been used to support simulation, (Ratzer et al. 2003) and (Ceric and Paul 1992). The formalism also forms the basis of a number of process modeling tools, (Prabhu 2003) and has been used in the area of workflow / business process management (van der Aalst 2002, 2003). Petri nets are capable of very accurately modeling and representing a real system. However, Petri nets of realistic systems are very often large and complex making it difficult for a non-expert to reason with the logic contained within a model. Therefore, while Petri nets can accurately represent a complex discrete event system, the technique does not have the ability to communicate the detailed information in a manner that could allow both a model developer and system users to use it as a communication medium to reach a common understanding with regard to system operational issues.

3.1.2 Discrete Event System Specification (DEVS)

The DEVS formalism described by Zeigler (1984) is a means of specifying a mathematical object called a system, with a time base, inputs, states, outputs, and functions for determining next states and outputs from current states and inputs. Zeigler (1984) proposed that discrete event systems represent certain collections of such parameters just as continuous systems do. He also proposed that there should be a separation between a model that describes a system and the mechanism used to simulate that system. This formalism has been

used to support the design and simulation of computer architectures, communications networks, and manufacturing systems (Rozenblit et al. 1990), (Thomasma and Ulgen 1988). It provides a formal representation of discrete event systems. However, the proposed mathematical representation is difficult to comprehend without a detailed knowledge of the formalism. So while the DEVS formalism is capable of accurately modeling a complex discrete event system the technique does not lend itself to communicating such complex information in a manner that facilitates its use as a means for model developers and non specialists to build a consensus view of system issues.

3.1.3 State Charts

Statecharts are based on the notation introduced by Harel (1987). A statechart diagram is made up of a number of basic elements, states and transitions. These statechart diagrams are used to show the flow of control or sequences of states that a system can proceed through as a result of discrete events (Borger et al. 2000). Such statecharts are used in the specification of dynamic systems and provide a means of mapping the various states through which a discrete system can transition and have been used in system simulation (Richter and Marz 2000, Hu and Shatz 2004). However the statechart diagram does not allow for the capture or modeling of either resource interactions or the activities that cause the change of states within a discrete system. Therefore, statecharts do not fully lend themselves to the visual representation of all detailed interactions that may occur within a complex discrete event system and as a result do not have the ability to communicate all such interactions and system issues in a visual manner.

3.1.4 Activity Cycle Diagrams

The Activity Cycle Diagram (ACD) is a technique for representing the interaction of entities within a system and is based on stochastic gearwheels as presented by (Tocher 1963). In an ACD, entities cycle through alternating states of activity and waiting (Chwif et al. 1999).

ACDs have been used in the development of STROBOSCOPE, a simulation language that can be used to express the logic of complex simulation models for construction (Martinez and Ioannou 1995). ACDs have also been proposed as a method for simplifying the modeling process of construction simulation (Shi 1997). While ACDs are capable of being used to model information a number of weaknesses have been noted including difficulties in capturing complex logic, along with models becoming cumbersome when modeling a complex system (Chwif et al. 1999).

3.1.5 Event Driven Process Chains

Event driven process chains (EPCs) of MERISE (Tardieu et al. 1983) as used in the ARIS ToolSet (Scheer 1992, 1998) and which form the basis of a number of process modeling tools (Kettinger et al. 1997), is a graphical business process description language. EPCs consist of sequences of functions and events. A function, being the basic building block of an event driven process chain, corresponds to an activity that needs to be executed, while an event describes the situations both before and after a function is executed. In this way an EPC consists of the capturing, representation and sequencing of activities that are to be executed in the progressing of a process.

EPCs are capable of accurately representing the flow of activities that are associated with the execution of tasks within a discrete event system. However, the technique does not allow for the modeling of the change of state of a discrete event system or the control of discrete systems. Therefore, while the EPC technique is capable of accurately representing certain areas of a complex discrete event system, it lacks the ability to capture and represent all of the aspects that would allow it to function as a communicative and representative technique for use by a model developer and system personnel during the initial requirements gathering or conceptual modeling phases of a simulation project.

3.2 Descriptive Methods

3.2.1 IDEF

As described earlier in this section IDEF has been used to support simulation development and was specifically developed to capture process knowledge.

The IDEF0 functional modeling method was developed from the SADT (Structured Analysis and Design Technique), to allow the analysis and communication of the functional aspect of a system (NIST 1993). The approach adopted in IDEF0 is to describe each process (or activity) as a combination of processes, inputs, controls and mechanisms in a hierarchical model. At the highest level the representation may be of an entire process. This representation may then be subdivided down into several more activity boxes or sub-processes. In such a fashion, the breakdown continues until the point is reached where sufficient detail is at hand to make the changes that might be needed. IDEF0 allows for the visual modeling of the decisions and activities in a system. However, the technique again lacks the ability to model the various other aspects of a complex discrete event system, such as the workflow and control flow, that are necessary to capture and communicate during the conceptual modeling or requirements gathering phase of a simulation project. The technique also lacks the capability to graphically represent the division of a system into multiple

processes.

The IDEF3 technique forms the basis of the ProSim modeling tool (Whitman et al. 1997). There are two IDEF3 description modes: process flow and object state transition network. A process flow description captures knowledge of "how things work" in an organisation, e.g. the description of what happens to a part as it flows through a sequence of manufacturing processes. The object state transition network description summarises the allowable transitions an object may undergo throughout a particular process. Both the process flow and object state transition descriptions contain units of information that make up the system description. These model entities, as they are called, form the basic units of an IDEF3 description (Mayer et al. 1995).

The IDEF3 process modeling technique allows for the capture and graphical representation of both the transition of states through a discrete event system and the activities associated with such state transitions. However, the modeling of the control of a discrete system is not graphically represented. This technique also does not allow for the graphical representation of resources within either the process flow description or the Object State Transition Network (OSTN) views. Such resources are often very important in the modeling and simulation of a discrete event system, as are queues, which again are not graphically represented. The IDEF3 elaboration language does allow for the capture and representation of resource interactions and queuing situations. However, the language is abstract in nature and does not lend itself to the communication of information to untrained users. As a result while the IDEF3 technique is capable of capturing certain aspects of a complex discrete event system, it lacks the ability to represent a number of important issues such as resource interactions and queuing. Therefore, the technique is not fully suited to the capture, representation and communication of all discrete system issues between a process model developer and system personnel in the early phases of a simulation project.

3.2.2 UML

Within UML, (Unified Modeling Language) statecharts (see subsection 3.1.3) are commonly used with UML activity diagrams. Such a diagram represents the execution of a process as a sequence of steps grouped sequentially as parallel control flow branches. An activity diagram consists of a series of activities (represented by rounded rectangles), decision points (represented by a diamond), synchronisation bars (represented by bars), and transitions (represented by lines). These diagrams may also be split into swim lanes to show the various responsibilities within an organisation (Muller 1997). These elements give the user a notation, which can be used to model both data and workflow. Activity diagrams have been used in this regard to model or assist in

the modeling and simulation of business systems (Barjis and Shishkov 2001). UML activity diagrams have been proposed as a pre-simulation technique (Barjis and Shishkov 2001) and have also been used as part of the FUJABA environment which has been used to test and simulate production control systems (Niere and Zundorf 1999). While UML activity diagrams are capable of representing workflow and dataflow within a discrete process they do not visually account for detailed interactions or the complex use of resources within a detailed simulation model. A technique that can visually represent the interactions between resources, system activities and the flow of work would, it is felt, be more capable of communicating detailed simulation logic to a non simulation expert.

3.2.3 Integrated Enterprise modeling (IEM)

The business process and relevant information, which is represented in one integral model, forms the core element of the Integrated Enterprise modeling (IEM) (Mertins et al. 1997, Mertins and Jochem 1999). To this core model, the organisational structure, quality management system, cost structure, control system and information system are represented by means of user views, which are directly related to the core elements in the model. The IEM method uses the object-oriented modeling approach and is based around three generic object classes, Product, Resource and Order.

The IEM technique is capable of modeling discrete processes. The technique also accounts for the interaction of both control and resource elements in the execution of activities. However, it is limited in its three modeling constructs and lacks the inclusion of a queue element which would be vital to the modeling of a discrete event system when gathering requirements or building a conceptual model for the purposes of a simulation project. As a result, the technique, while being capable of modeling discrete systems is not capable of capturing and representing such detailed interactions as those inherent in complex discrete event systems. Therefore, it is not ideally suited to the purpose of communicating system issues between model developers and system personnel involved in a simulation project.

3.2.4 Role Activity Diagrams

The technique of Role Activity Diagrams (RADs) as introduced by Ould (1995), attempts to model a process in terms of the roles present within the process, their component activities and their interactions, together with external events and the logic that determines what activities are carried out when and by whom. Although RADs have been used in software engineering they are not primarily directed at modeling the information flows within an organ-

isation, a feature that distinguishes them from many other notations in the field (Murdoch and McDermid 2000). As a result, RADs can, and have been used, to express the organisation of design activities, communication between various groups involved and the links between these and the evolving project (Murdoch and McDermid 2000). RADs have also been proposed as an aid to the modeling of a safety process for the purposes of building a safety case for new systems (Dawkins 1998)

While RADs lack the ability to model the change of state of a discrete event system they do attempt to model a process in terms of roles that have to be carried out within that process. This modeling approach, while not explicit in terms of the logical execution of tasks, as is the case with simulation, does place the interactions or roles of a person within a process more to the fore (Heavey and Ryan 2002). Such an approach lessens the cognitive jump that a user has to make to visualise their interactions within the model and in turn the real process. Conseqently, the user's ability to understand information contained within the model improves.

3.2.5 GRAI Method

GRAI borrows concepts from control and systems theory allowing both the structure and control of a system (primarily manufacturing systems) to be captured (see (Doumeingts 1985)), (Chen et al. 1997) and (Zülch et al. 2001)). This GRAI model, along with five other elements, constitutes the GRAI Integrated Methodology (GIM) (Chen et al. 1997). The GRAI model, which is relevant to the discussion here, is made up four subsystems. These are the physical, operational, decisional, and information systems. The physical system is used to model the process of transforming input objects into output or finished objects by means of a flow of these objects through a model of the physical elements of the system. The physical model contains the resources which are needed to fulfill the operations represented in the operational model, which is on the next level of the GRAI approach ((Doumeingts et al. 1998)). Over the physical and the operational models, the decisional model is layered, and this is split into two levels: (i) The higher level which represents the general decisional structure and is modeled by a decisional matrix (GRAI-grid); (ii) The lower or operational level describes in detail the single-decision centres of the GRAI-net. GRAI has been used in the modeling and simulation of production systems (Ducq et al. 2001), (Al-Ahmari and Ridgway 1999). The GRAI method primarily focuses on the decisional structure of a manufacturing system. It does not adequately model the flow of work for the purposes of capturing and aiding in the communication of system issues in the pre-coding stages of a simulation project. The modeling of the decisional structure of a discrete event system is, however, important as modern systems rely heavily on such decisional systems for control and regulation. As a result, it is felt that

the graphical representation of such a decisional system and its interactions with the flow of work through a discrete event system would be vital to aid in the communication of system issues between a model developer and system personnel.

3.3 Discussion

Section 3 presented a review of different process modeling techniques applicable to pre-coding activities within a simulation project. This review was carried out with a view to ascertain each technique's ability to capture, represent and communicate the various aspects of a complex discrete event system. While there are many process modeling techniques and software tools available that may be used to support the requirements gathering phases of a simulation project, none of the techniques reviewed are capable of capturing, representing and communicating the various aspects of discrete event systems and their interactions within a complex process in such a way as to fully support the transmission of detailed simulation information to a non simulation expert. No technique reviewed was capable of representing all of the following aspects of a discrete event system within a single process model:

- The flow of work, or change of state of a discrete event system;
- The flow of information associated with the control of a discrete event system;
- The activities that are associated with the execution of the flow of work and information within a discrete event system;
- The resources necessary and their usage in carrying out the activities associated with the execution of both work and information within a discrete event system;
- The modeling of a discrete event system from the perspective of the user and their interactions with the system in the execution of activities within the system;
- The separation between the process modeling tool and the simulation engine to allow for the capture, representation and communication of detailed interactions at a high level during the requirements gathering phase as opposed to purely at the low level code stage of a project;
- The access to a means of elaborating graphical models to facilitate the communication of detailed information associated with such graphical representations.

To overcome these shortfalls the following sections outline a process modeling technique, Simulation Activity Diagrams (SADs), which has been developed to specifically support the pre coding phases of a simulation project. This technique has been developed with a view to overcoming the shortfalls outlined

above and in doing so, it is argued, is well placed to support a model developer in the requirements gathering phases and conceptual model development within the process of a simulation project.

4 Simulation Activity Diagrams (SAD)

The technique proposed here (Simulation Activity Diagrams (SAD)) aims to overcome the shortfalls discussed in the previous section. The technique aims to be highly visual and aid in the process of communication between the model developer and system users, while still aiding the model developer in the gathering of data for the creation of a simulation model.

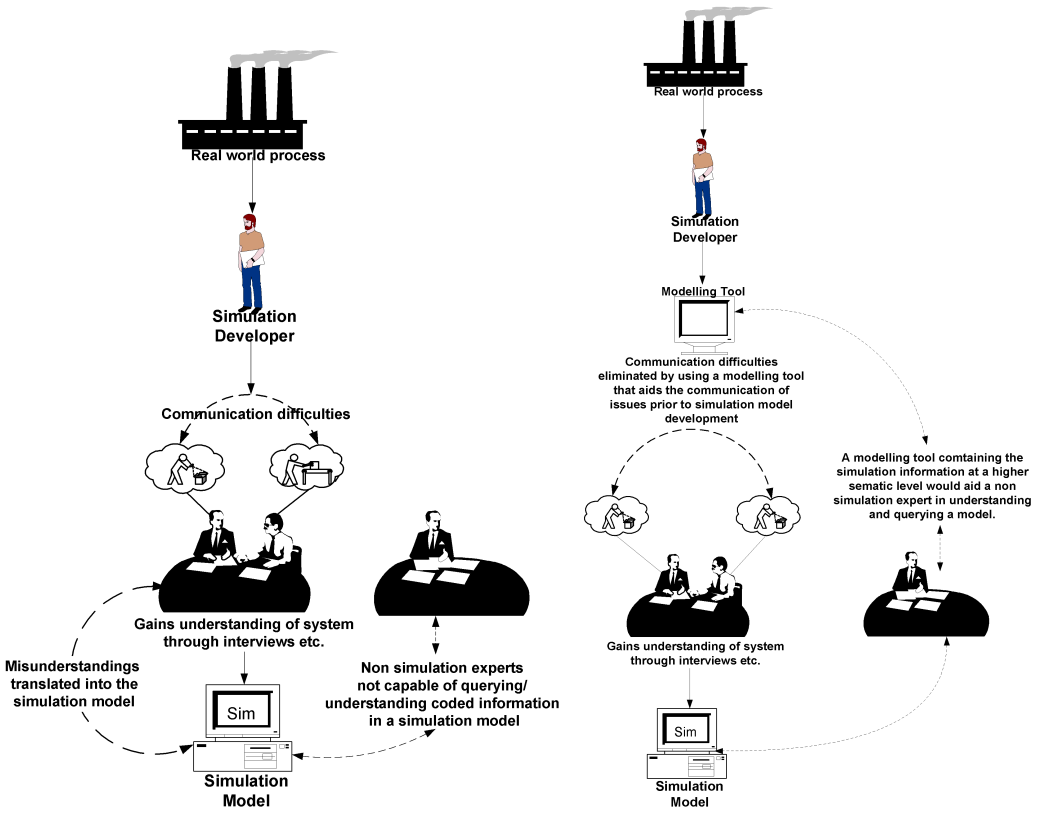
As well as supporting the requirements gathering phase of a simulation project, another important function of the technique proposed here is to act as a knowledge repository. Figure 2 depicts the situation where no technique is used for requirements gathering in a simulation project and where one is used. In Figure 2(a) system information contained in a simulation model cannot be easily queried by non-specialists. Simulation, while providing quantitative information, is a poor communication tool. Essentially, valuable system knowledge is lost. In Figure 2(b) requirements collection is supported but more importantly system knowledge that is gathered can be stored. If it is stored in a format that is easily accessed by a range of personnel it should be a valuable resource for system design tasks, i.e. provide support for continuous improvement programs.

4.1 Design Objectives

In developing SAD a number of design objectives were adhered to. These were:

- The technique has to be capable of capturing a detailed description of a discrete event system;
- The technique should have a low modeling burden and therefore be capable of being used by non specialists;
- The technique should present modeling information at a high semantic level so that personnel can rationalize with it;
- The technique should have good visualization capabilities;
- The technique should support project teamwork.

In addressing these design objectives the technique developed uses a set of modeling elements that allows both a simulation model developer and a non-expert to reach a common understanding of the system being modeled. The



(a) Difficulties with simulation models as a communicative tool.

(b) Proposed use of the SAD technique.

Fig. 2. Present situation and proposed use of the SAD technique.

technique allows the construction of a detailed and highly visual model of a system. This model can then be used as a common representation and a focal point for discussion through which both system personnel and the developer can reach a common understanding of the operation of the system and the data requirements. In this way the technique allows the user high level access to knowledge contained in the simulation code that would otherwise be lost due to its low level nature.

To achieve this effectively there are a number of functional requirements that are central to the proper development of a SAD model:

- In instances where a system's complexity makes reasoning difficult the method is capable of breaking the system into partial models and hierarchically structuring these models. These models allow users to comprehend complex data;
- The user is presented with a core set of high-level elements that allow the construction of a fully detailed model;

A brief overview of the Simulation Activity Diagram (SAD) is now presented.

4.2 SAD Action List

A discrete event system consists of a series of discrete events, the outcomes of which when grouped together ultimately decide the progress of a particular system. In a simulation engine these events are stored in an event list and executed in order of their time of occurrence. The SAD technique graphically represents every event in a simulation model of an activity.

> An activity is any event that causes the change of state of a discrete event system.

However an event in a simulation model can often represent more than one event or task. Often model developers group such events together to lessen the programming burden. This can often lead to difficulties in relation to non simulation personnel understanding simulation models. To overcome this an activity can be subdivided into a series of what are defined as actions.

> An action element represents the individual task or tasks that have to be performed to execute an activity.

This approach allows an activity or event to be further subdivided into its various individual elements or tasks. In other words an activity in a SAD model can be considered to be a list of actions that have to be executed in order for the activity to be fully completed. Figure 3 shows an activity consisting of three actions, which are executed as follows.

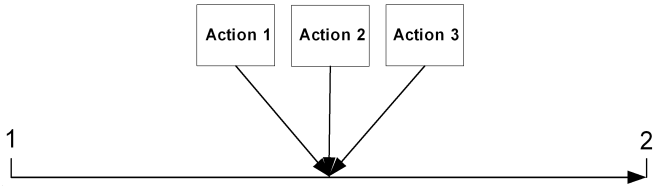


Fig. 3. SAD Actions.

The system is in state 1. Before it can transition to state 2, all actions, 1,2 and 3 must be executed. In this way an individual activity is considered a separate mini event list or action list within the SAD model. These actions are executed in a time ordered sequence from top to bottom and from left to right ensuring that each criterion is satisfied. Only when each action has been executed, can the full activity be executed and the system transition successfully to state 2. Taking this approach a SAD becomes a graphical representation of the various events in a simulation model. Each event is represented in a SAD by an activity. This activity is then further graphically represented by an action

list. This will be further developed in the following section by the introduction of a series of modeling primitives that may be used in the detailing of such an activity.

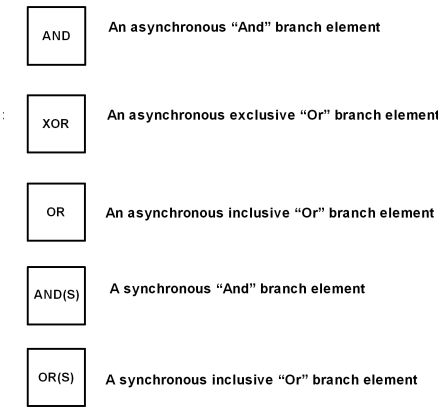


Fig. 4. SAD Branching elements.

4.3 SAD Modeling Primitives

Within most systems, actions such as those in Figure 3 are rarely executed without a number of other types of resources being used. These resources are briefly introduced below:

Primary resource element: A primary resource element represents any resource within a discrete event system which facilitates the transformation of a product, physical or virtual, from one state of transition to another;

Queue resource element: A queue modeling element represents any phase of a discrete event system where a product, virtual or physical, is not in an active state of transformation within the system;

Entity element: An entity element represents any product, physical or virtual, that is transformed as the result of transitioning through a discrete event system;

- **Entity state element:** An entity state represents any of the various states that a physical object or component explicitly represented within a system transitions through during physical transformation

Informational element: An informational element represents any information that is used in the control or operation of the process of transition by a product through a discrete event system.

- **Informational state element:** An informational state represents any of the various states that information used in the operation or control of a discrete event system transitions through during the support of the operation of the physical transformation

Auxiliary resource element: An auxiliary resource represents any resource used in the support of a Primary Resource.

Actor auxiliary resource: An actor auxiliary resource represents any auxiliary resource used in the direct support of the execution of an action or actions within the process of transitioning a system from one state to another.

Supporter auxiliary resource: A supporter auxiliary resource represents any auxiliary resource used in the direct support of an actor auxiliary resource in the execution of an action or actions within the process of transitioning a system from one state to another.

Branching Elements: Most discrete event systems are complex in nature and are rarely, if ever, linear. To account for the representation of such situations the SAD technique uses a number of branching elements. Figure 4 shows the various types of branching elements used in the SAD modeling technique.

Link Types: Links are the glue that connects the various elements of a SAD model together to form complete processes. Within the SAD technique there are three link types introduced known as entity links, information links and activity links. The symbols that represent each type are shown in Figure 5.

SAD Frame Element: The SAD frame element provides a mechanism for the hierarchical structuring of detailed interactions within a discrete event system into their component elements, while also showing how such elements interact within the overall discrete event system.

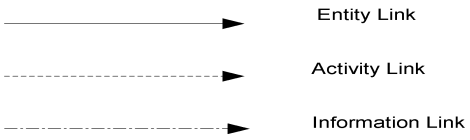


Fig. 5. SAD Link Types.

4.4 SAD Model Structure

A SAD model is executed in time sequenced ordering from left to right and from the center auxiliary resource area to the extremities of the model and is structured as follows (see Figure 6). At the center of the model are located the actors and supporters also known as auxiliary resources. These are the supporters for both the information and physical models. This is advantageous for the purposes of communication during the requirements gathering phase of a simulation project as the persons with whom the simulation model developer will be communicating will generally be a supporter within the process. Therefore, each SAD model will be developed from the perspective of the persons interacting with the system.

The interconnecting areas between both models contain the actions to be executed. A series of these actions and the associated interactions with other

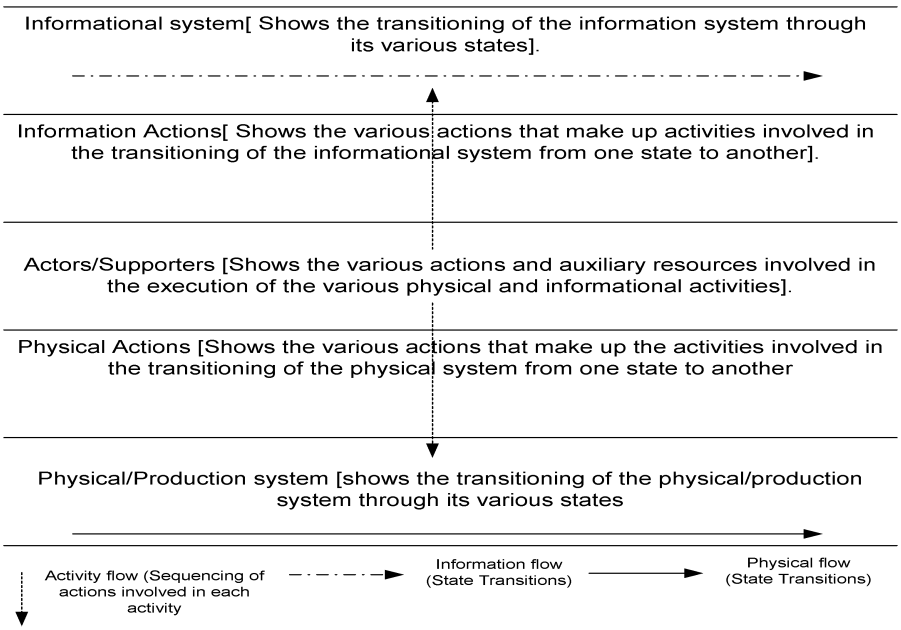


Fig. 6. SAD Model structure.

SAD modeling elements make up an action list. A series of these activities in turn make up a sequence of transition for physical or information entity. Figure 7 shows a simple SAD model for both a physical and informational system.

In this simple example there are two auxiliary resource elements, namely, supporter auxiliary resource element, "Supporter 1" and the actor auxiliary resource element "Actor 1". In the case of the information model (top of Figure 7) only the actor auxiliary resource element "Actor1" is used. This aspect of the model captures the flow of information required to operate a system.

The physical model, shown at the lower extremity of the extended SAD, shows the possible physical states that the system can transition through. Such transitions only take place as a result of the execution of all necessary actions, which are executed from left to right within the SAD model. In this case the physical system can transition from state 1 to either state 2 or state 3 as a result of the actions carried out on the primary resource element, "Machine X". The auxiliary resources section again details what resources are used in the execution or in the support of the execution of each of the actions. In this case the supporter auxiliary resource, "Actor 1" is used in the execution of each of the three actions A, B and C. However, again, in this case, the supporter auxiliary resource, "Supporter 1", is used only in the execution of action A. Therefore, both of the auxiliary resources "Actor 1" and "Supporter 1", denoted by the synchronous And, "AND(S)" fan in branch element, have to be present at the same instance for the successful execution of "Action A". All three actions are executed on the primary resource element "Machine X".

As a result of the execution of these three actions the physical system can undergo a transition from state 1 to either state 2 or state 3.

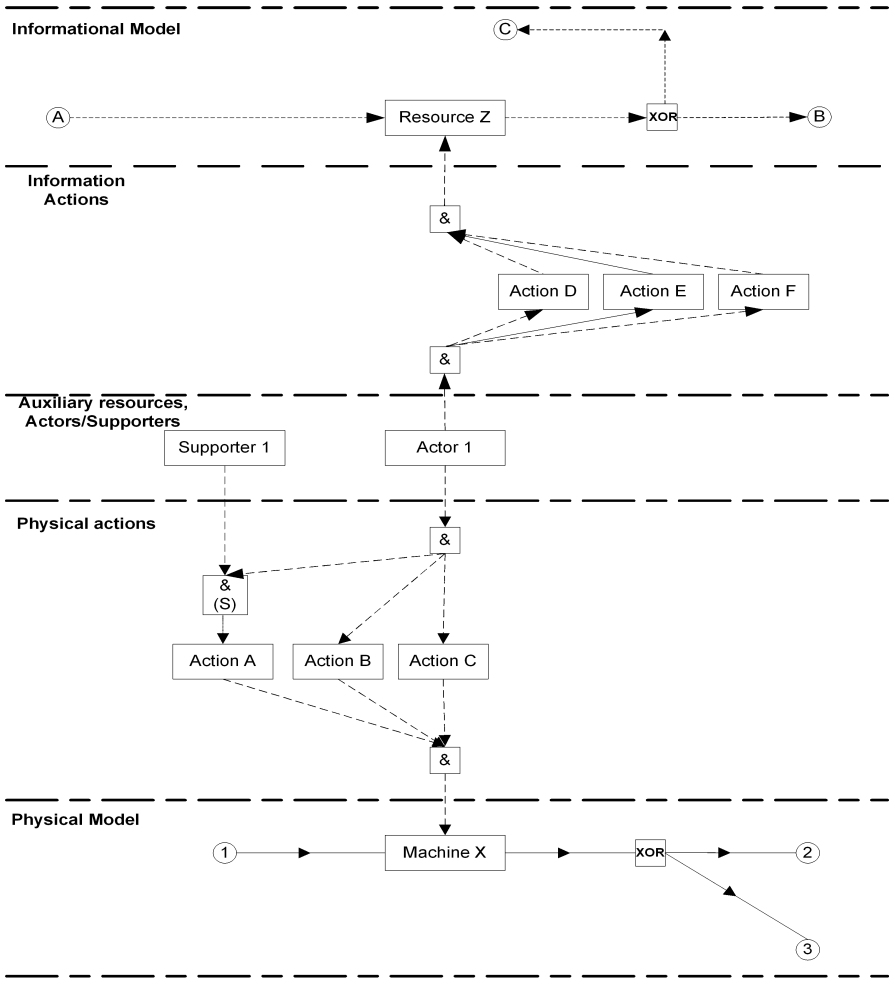


Fig. 7. A Simple SAD.

4.5 Elaboration of SAD Models

Thus far, the modeling elements used to develop a SAD model have been introduced to provide a means of visually modeling discrete event systems. However, such graphical models are capable of only representing a certain amount of detailed information and knowledge. Often, complex discrete event systems contain detailed information and knowledge related to process interactions that cannot be captured well by such graphical representations. To provide a means of making such information available to a model user the SAD technique also makes use of an elaboration language with which each individual SAD diagram can be described in greater detail. This structured language makes use of a number of different reserved words to allow the description of SADs (see Table 1).

Keyword	Description
USES	The supporter resource may at times make use of auxiliary resources to execute an action or actions, in other words a supporter **USES** auxiliary resources.
TO	Details the action or actions that are executed by use of an auxiliary resource by a supporter resource.
AT	Specifies the Locations where the action or actions are executed.
ON	Specifies the primary resources that are used to transform entity states.
TRANSITIONS TO	Specifies the change of state of entity or information from one state to another

Table 1
Structured language

These words are used to describe the various interactions that take place in a SAD diagram. While such interactions are represented by various branches, which show the convergence or divergence of a system at certain points within the visual model, such branches may have a different semantic meaning to a user based on where within the model they are used. Branch statements are also used in the structured language, e.g., **AND**.

4.6 Process modeling for Simulation Software

A prototype software application called the PMS (Process Modeling Software) has been developed using Microsoft Visual C++ to implement the SAD methodology. The focus of the application has been to represent the SAD technique and to demonstrate the technique's ability to capture and visually communicate detailed system information in a user-friendly manner. Using this software several systems have been modeled using PMS with the aim of validating the SAD technique. Systems modeled were: (i) A Small Medium Enterprise (SME) that produce precision components; (ii) A manufacturing system that implements Kanban production control; (iii) A batch flow-shop; (iv) A production line. To further illustrate the PMS software and the SAD technique, part of the SAD model for (i) above is described in the next section.

5 Sample SAD Application

This section takes the modeling constructs introduced previously in section 4 and uses them to develop a SAD model of a discrete event system. The

graphical representations of each SAD modeling element that were introduced in section 4 are shown in Figure 8. Along with these modeling elements the SAD elaboration for each diagram will be provided to demonstrate the SAD approach to communicating operational information at a semantically high level.

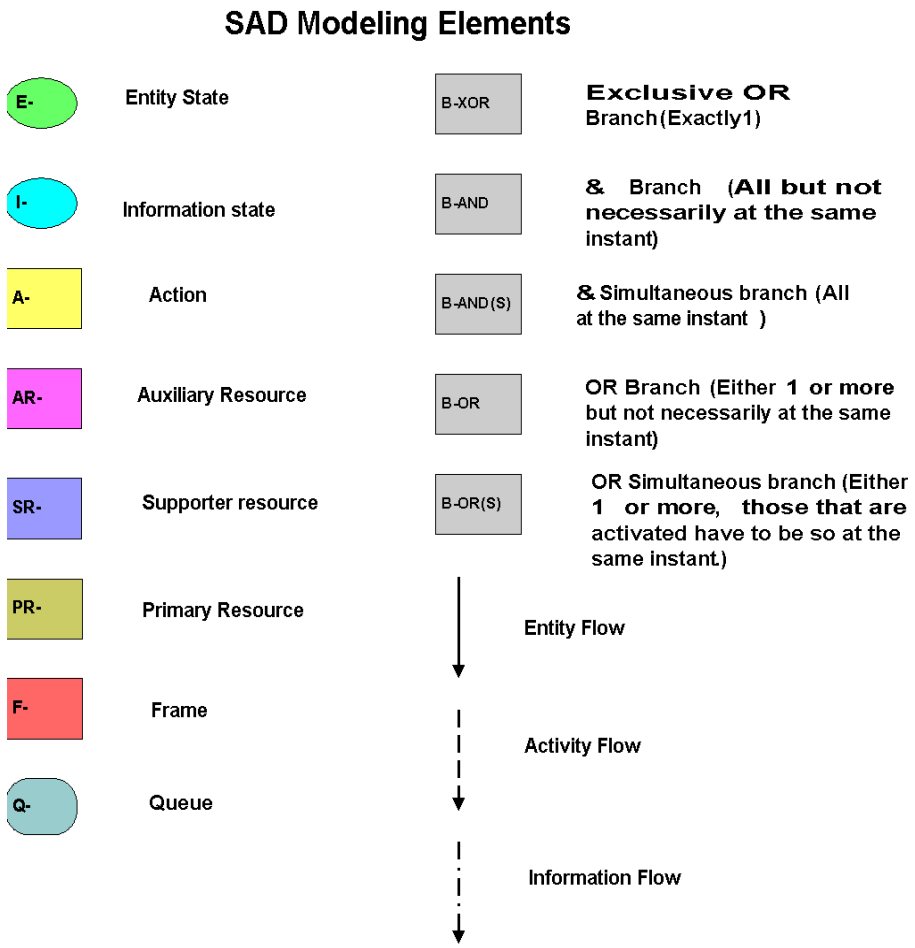


Fig. 8. SAD modeling elements

5.1 SAD model – Overall System

The system modeled in this section is based on the results of a series of system interviews conducted with a number of workers in the precision component manufacturing facility. In the early stages of any simulation project, indeed any project, it is necessary to gain a detailed understanding of the operation of the system being studied. The shop-floor layout of the manufacturing consists of six separate areas of processing. Each of these areas will be modeled using the SAD modeling technique. Figure 9 shows the highest level of this system modeled in this case. Here, the various actions carried out by the produc-

tion manager are shown, as are the various flows of information and entities through the manufacturing facility. The bottom portion of Figure 9 models the flow of the physical part through the system. The part is accompanied by documentation which needs to be updated as it flows through the system. The top portion of Figure 9 models this flow of information. In this instance it is an exact parallel flow of the physical part, but this may not always be the case. An elaboration language description of this highest level diagram is shown in Table 2.

Table 2
Elaboration description of overall system

A part and accompanying documentation enters the system in an entity state and information state, respectively. During its time in the system the following actions are executed.
Production Manager **USES** Computer **TO** Monitor Production **AT** Delivery area **AND** Drilling **AND** Milling **AND** Inspection **AND** Packaging **AND** Warehouse
AND Production Manager **USES** Computer **TO** Oversee orders **AND** Monitor quality **AT** Delivery area **AND** Drilling **AND** Milling **AND** Inspection **AND** Packaging **AND** Warehouse
THEN Delivered entity state **TRANSITIONS TO** Shipped entity state **AND** Delivered information state **TRANSITIONS TO** Shipped information state

The high level SAD presented in Figure 9 consists of a number of frame elements, which are used to allow for the hierarchical decomposition of a SAD diagram or particular system into more detailed SAD diagrams or subsystems. In this instance the frame elements are used to represent the following sub systems or work areas; Delivery area, Drilling, Milling, Inspection, Packaging and Warehousing. The following section presents the SAD diagram and elaboration associated with the Inspection frame element. In other words this SAD diagram is used to represent more detailed information associated with the Inspection subsystem of the system being modeled.

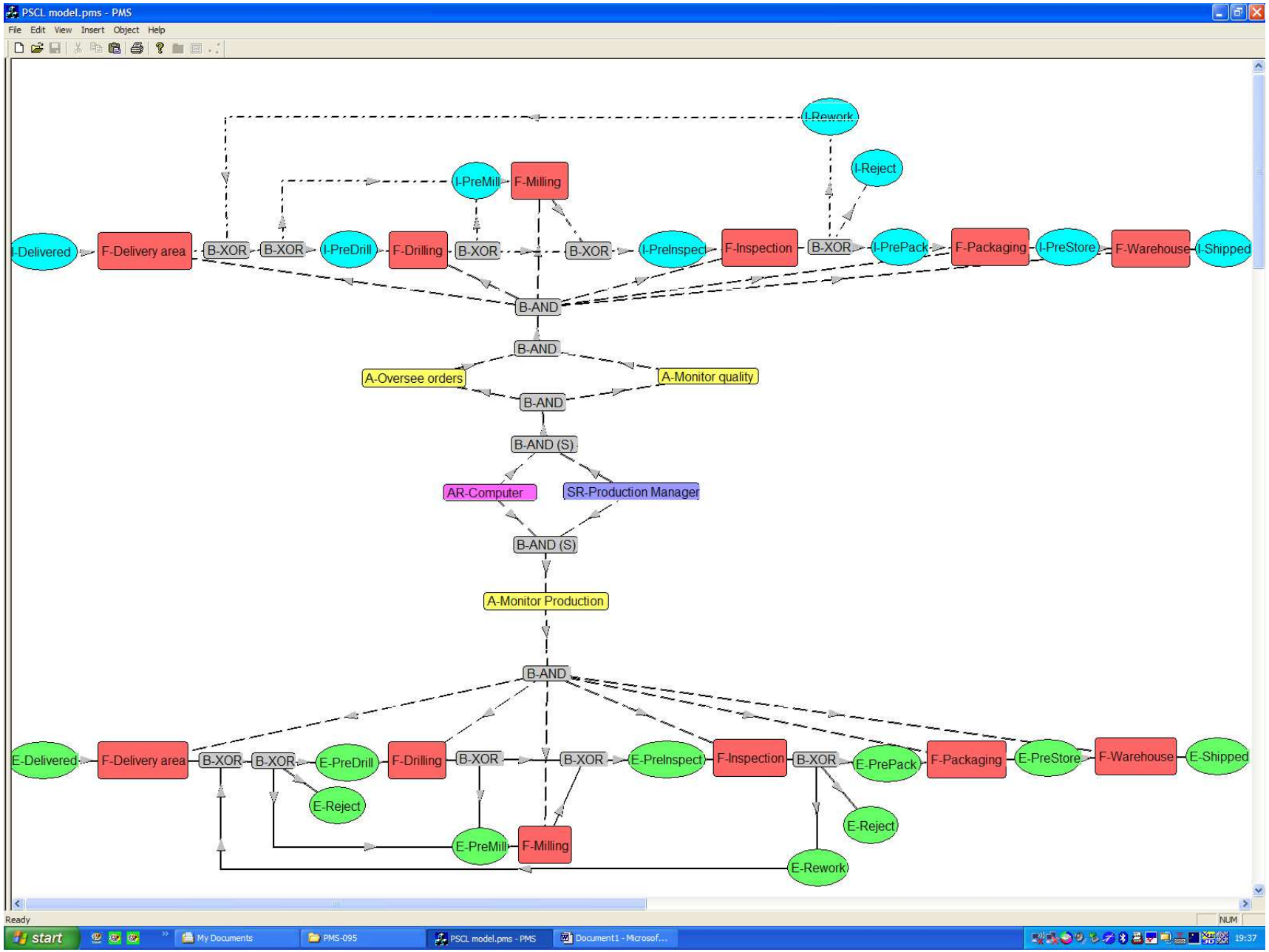


Fig. 9. SAD model overall system

The inspection area consists of an inspection table where one operator inspects every part passing through the station. If the parts pass the inspection of the operator they are placed directly on a pallet for transfer to the packaging area. If the parts are found to be oversized for drilling or undersized for milling they are placed on a pallet for disposal. If the parts are found to be under sized for drilling or oversized for milling they are placed on pallets for transfer to their respective rework sections of the delivery holding area. The inspection area is modeled as shown in Figure 10, with the elaboration language description of this area being contained in Table 3. Similar to the SAD shown in Figure 9 the bottom portion of Figure 10 models the phyical flow of the part and the top portion models the documentation that accompanies the part.

6 Conclusions

A very important task in a simulation project is requirements gathering and conceptual model development. This paper highlights the fact that there is inadequate support currently available for this task. While numerous process modeling techniques are available and several have been used to support the requirements gathering of a simulation project, the paper argues that the techniques available do not provide adequate support. Several deficiencies of current tools were highlighted. The design objectives of a modeling method that would overcome these deficiencies were presented. Results of a research effort into developing such a technique is reported.

The SAD technique endeavours to model complex interactions such as those that take place within an actual detailed simulation model of a real system. To achieve this the modeling method uses the various SAD modeling primitives to represent the events in a simulation model. To also represent more complex interactions the SAD method introduces the concept of an action list, which is used to represent detailed actions that collectively can make up any event within a simulation model. The SAD method also allows for the modeling of both a physical and informational system that may make up a discrete event system along with interactions between both. The use of elaborations using structured text within the SAD method is proposed to allow a user to understand and validate a SAD model. Currently, the method is being further developed and validated.

Requirements gathering and conceptual model development is a very important task in the simulation modeling process (Law 1991). It is claimed that 50% of the benefit is obtained in many simulation projects just from the re-

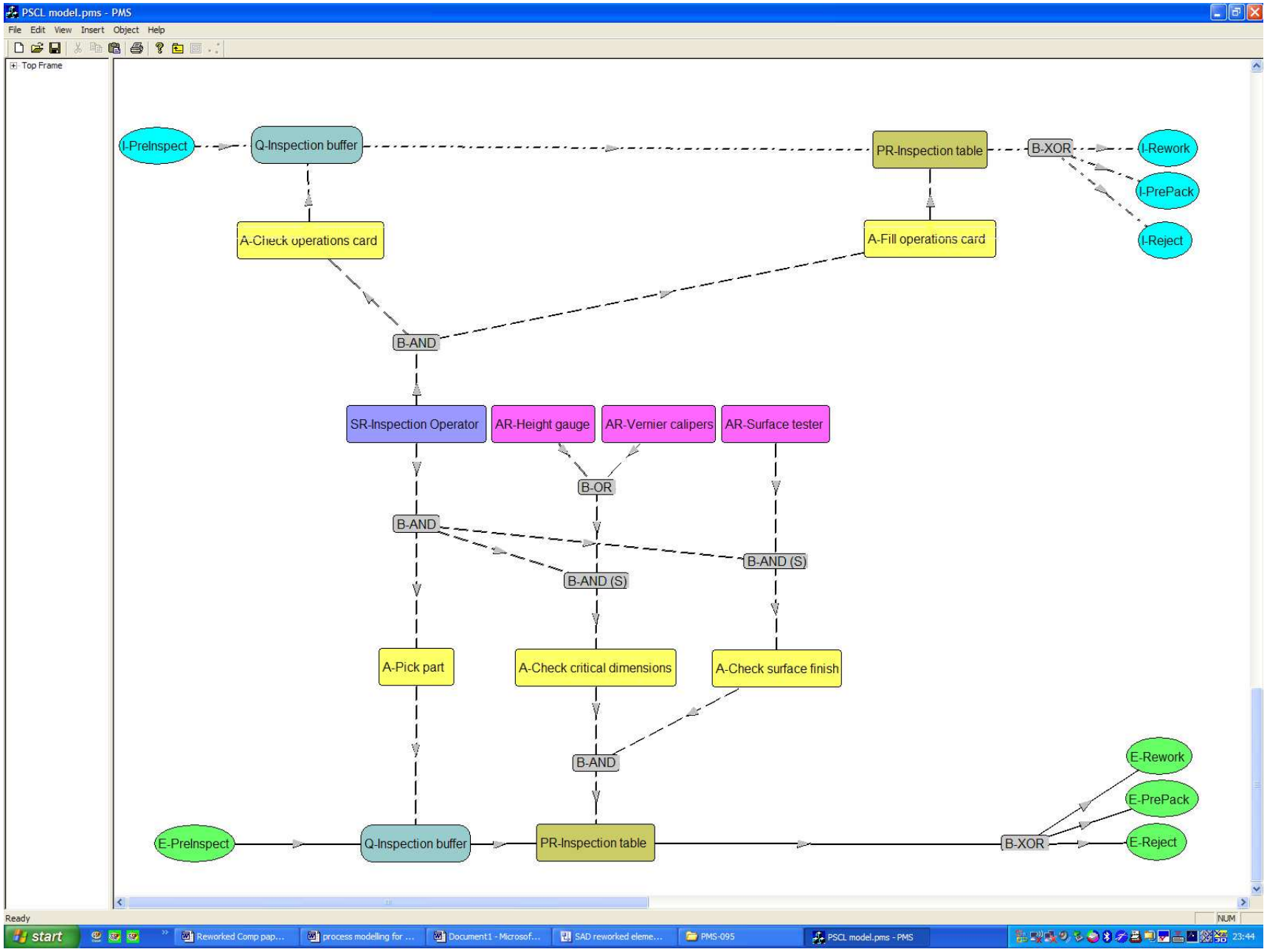


Fig. 10. SAD model inspection area

Table 3
Elaboration description of Inspection Area

A pre-inspect entity state enters the inspection area in batches of 100 accompanied by a pre inspect information state for the following actions to be executed.
Inspection Operator
Picks part
AT Inspection buffer
The Inspection buffer treats parts in a First In First Out (FIFO) manner.
AND
USES Height Gauge **OR** Vernier calipers
TO Check critical dimensions
The setup times for this operation average 1.36 mins and the details of this are recorded in the attached document(Dimension_test_setup.xls). The average time taken for this operation is 5.8 mins, with the details contained in the attached document (Dimension_Op_Times.xls)
AND
USES Surface tester
The details of the Surface finish tests performed on the parts in the Inspection area are contained in the attached document (Surface_tests.doc)
TO Check surface finish
The setup times for this operation average 2.56 mins and the details of this are recorded in the attached document(Surface_test_setup.xls). The average time taken for this operation is 3.2 mins, with the details contained in the attached document (surface_Test_Times.xls). The Mean Time to Failure (MTF) and the Mean Time to Repair (MTR) for this operation are attached in the following documents respectively(Surface_test_MTF.xls)
AT Inspection table
AND Inspection Operator
Check operations card
AT Inspection buffer
AND
Fill operations card
AT Inspection table
THEN
Pre-Inspection entity state **TRANSITIONS TO EITHER** Rework entity state **OR** Prepack entity state **OR** Reject entity state
This transition is based on the results of the tests carried out on the parts by the inspection operator.
AND
Pre Inspection information state **TRANSITIONS TO EITHER** Rework information state **OR** Reject information state
The transition here represents the transition of the operations card, which details each operation and in the case of the inspection operation, the outcome of the operation, which accompanies each batch of parts through the system.

quirements and conceptual model development task (Robinson 2004). Due to its importance the lack of research into this area is surprising. SAD is one possible solution to developing support for this area. Many avenues of future research exist in this area.

References

Abdel-Malek, L., Johnson, F. and Spencer III, T.: 1999, Or practice: Survey results and reflections of practising informs members, *Journal of Operational Research Society* **50**(10), 994–1003.

Al-Ahmari, A. M. A. and Ridgway, K.: 1999, An integrated modelling method to support manufacturing systems analysis and design, *Computers in Industry* pp. 225–238.

Barjis, J. and Shishkov, B.: 2001, UML based business systems modeling and simulation, *4th International Eurosim 2001 Congress*, Delft, The Netherlands.

Borger, E., Cavarra, A. and Riccobene, E.: 2000, Modeling the dynamics of UML state machines, *International Workshop on Abstract State Machines (ASM'2000)*, Springer, pp. 223–241.

Ceric, V. and Paul, R.: 1992, Diagrammatic representations of the conceptual simulation model for discrete event systems, *Mathematics and Computers in Simulation* **34**, 317–324.

Chen, D., Vallespir, B. and Doumeingts, G.: 1997, GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology., *Computers in Industry* **33**, 387–394.

Chwif, L., Paul, R. J. and Barreto, M. R. P.: 1999, Combining the best of the two: An activity cycle diagram/condition specification approach, *Fourth National Conference of the UK Simulation Society*, Nottingham Trent University, Nottingham, UK., pp. 93–98.

Dawkins, S.: 1998, Role activity diagrams for safety process definition., *16th International System Sefety Conference.*, Seattle, WA, USA.

Doumeingts, G.: 1985, How to decentralize decisions through GRAI model in production management, *Computers in Industry* **6**(6), 501–514.

Doumeingts, G., Vallespir, B. and Chen, D.: 1998, Decisional modelling using the GRAI grid, *in* P. Bernus, K. Mertins and G. Schmidt (eds), *Handbook on Architectures of Information Systems*, Springer, Berlin, pp. 313–338.

Ducq, Y., Vallespir, B. and Doumeingts, G.: 2001, Coherence analysis methods for production systems by performance aggregation., *International Journal of Production Economics* **69**, 23–37.

Harel, D.: 1987, Statecharts: A visual formalism for complex systems, *Science of Computer Programming* **8**(3), 231–274.

Heavey, C. and Ryan, J.: 2002, Process modelling for simulation, *International Manufacturing Conference*, Queens University Belfast, pp. 509–518.

Hollocks, B.: 1992, A well kept secret? simulation in manufacturing industry reviewed, *OR Insight* **5**(4).

Hu, Z. and Shatz, S.: 2004, Mapping uml diagrams to a Petri net notation for system simulation, *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Banff, Canada.

INCOME Process Designer: 2005.
URL: *http://www.get-process.com/ - Last accessed 09/08/2005*

Jeong, K.-Y.: 2000, Conceptual frame for development of optimized simulation-based scheduling systems, *Expert Systems with Applications* **18**(4), 299–306.

Kerckhoffs, E., Vangheluwe, H. L. and Vansteenkiste, G.: 1995, Interim report of ESPRIT basic working group 8467, *Technical Report ESPRIT DG-XIII*.

Kettinger, W. J., Teng, J. T. C. and Guha, S.: 1997, Business process change: A study of methodologies, techniques, and tools, *MIS Quarterly* **21**(1), 55–80.

Law, A. M.: 1991, Simulation model's level of detail determines effectiveness, *Industrial Engineering* **23**(10), 16–18.

Martinez, C. J. and Ioannou, G. P.: 1995, Advantages of the activity scanning approach in the modeling of complex construction processes, *Winter Simulation Conference*, Arlington, Virginia, US.

Mayer, R. J., Menzel, C. P., deWitte, P. S., Blinn, T. and Perakath, B.: 1995, Information integration for concurrent engineering (IICE) IDEF3 process description capture method report., *Technical report*, Knowledge Based systems Incorprated (KBSI).

Mertins, K. and Jochem, R.: 1999, *Quality-oriented design of business processes.*, Kluwer Academic.

Mertins, K., Jochem, R. and Jakel, F. W.: 1997, A tool for object-oriented modelling and analysis of business processes., *Computers in Industry.* **33**, 345–356.

Muller, P. A.: 1997, *Instant UML*, Wrox Press.

Murdoch, J. and McDermid, J. A.: 2000, Modelling engineering design processes with role activity diagrams, *Transactions of the SDPS* **4**(2), 45–65.

Nethe, A. and Stahlmann, H. D.: 1999, Survey of a general theory of process modelling, *International Conference on Process Modelling*, Cottbus, Germany, pp. 2–16.

Niere, J. and Zundorf, A.: 1999, Testing and simulating production control systems using the fujaba environment, *AGTIVE 1999*, Kerkrade, The Netherlands.

NIST: 1993, Integration definition for function modeling (IDEF0), *Technical Report FIPS 183*, National Institute of Standards and Technology.

Ould, M. A.: 1995, *Business processes: Modeling and analysis for the re-engineering and improvement*, Wiley.

Perera, T. and Liyanage, K.: 2000, Methodology for rapid identification and collection of input data in the simulator of manufacturing systems, *Simulation Practice and Theory* pp. 645–656.

Pidd, M.: 1989, *Computer Modelling for Discrete Simulation*, John Wiley & Sons Ltd.

Prabhu, V.: 2003, *Scalable Enterprise Systems: An Introduction To Recent Advances*, Kluwer Academic.

Ratzer, A. V., Wells, L., Lassen, H. M., Laursen, M., Qvortrup, J. F., Stissing, M. S., Westergaard, M., Christensen, S. and Jensen, K.: 2003, CPN tools for editing, simulating, and analysing coloured Petri nets, *in* W. van der Aalst

and E. Best (eds), *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003*, Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Eindhoven, The Netherlands, pp. 450–462.

Richter, H. and Marz, L.: 2000, Toward a standard process: the use of UML for designing simulation models, *Winter Simulation Conference 2000*, Orlando, Florida, USA.

Robinson, S.: 2004, *Simulation: The practice of model development and use*, John Wiley & Sons.

Rozenblit, J. W., Hu, J. F., Kim, T. G. and Zeigler, B. P.: 1990, Knowledge based design and simulation environment (KBDSE): Foundation concepts and implementation., *Journal of Operational Research Society* **41**(6).

Scheer, A.-W.: 1992, *Architecture of Integrated Information Systems: Foundations of Enterprise Modelling*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Scheer, A. W.: 1998, ARIS, *in* P. Bemus, K. Mertins and G. Schmidt (eds), *Handbook on Architectures of Information systems*, Springer- Verlag, Berlin.

Shannon, R., Long, S. and Buckles, B.: 1980, Operations research methodologies in industrial engineering, *AIIE* .

Sheppard, S.: 1983, Applying software engineering to simulation, *Simulation Practice and Theory* **10**(1), 13–19.

Shi, J.: 1997, A conceptual activity cycle-based simulation modeling method, *Winter Simulation Conference*, Atlanta, GA, USA.

Tardieu, H., Rochfeld, A. and Colletti, R.: 1983, *La Méthode Merise, Principes et Outils*, Les Editions des Organisations.

Thomasma, T. and Ulgen, O.: 1988, Hierarchical, modular simulation modeling in icon-based simulation program generators for manufacturing, *Winter Simulation Conference*, San Diego, California, USA.

Tocher, K. D.: 1963, *The art of simulation*, English Universities Press.

van der Aalst, W. M. P.: 2002, Making work flow: On the application of Petri nets to business process management, *in* J. Esparza and C. Lakos (eds), *Application and Theory of Petri Nets 2002, volume 2360 of Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 1–22.

van der Aalst, W. M. P.: 2003, Challenges in business process management: Verification of business processes using Petri nets, *Bulletin of the EATCS* **80**, 174–198.

van Rensburg, A. and Zwemstra, N.: 1995, Implementing IDEF techniques as simulation modelling specifications, *Computers & Industrial Engineering* **29**(1-4), 467–471.

Whitman, L., Huff, B. and Presley, A.: 1997, Structured models and dynamic systems analysis : The integration of the IDEF0/IDEF3 modelling methods and discrete event simulation, *Winter simulation conference*, Atlanta, pp. 518–524.

Zeigler, B. P.: 1984, *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, London.

Zülch, G., Rinn, A. and Strate, O.: 2001, Dynamic analysis of changes in

decisional structures of production systems., *International Journal of Production Economics* **69**, 239–252.