2006-12-03

Process Modelling Support for the Conceptual Modelling Phase of a Simulation Project

Cathal Heavey
University of Limerick

John Ryan
Dublin Institute of Technology, john.ryan@dit.ie

Process modelling support for the conceptual modelling phase of a simulation project

John Ryan Dr.
DIT, john.ryan@dit.ie

Cathal Heavey Dr
UL

— 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.

Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

PROCESS MODELLING SUPPORT FOR THE CONCEPTUAL MODELLING PHASE OF A SIMULATION PROJECT

Cathal Heavey

Enterprise Research Centre
University of Limerick
Limerick, Ireland.

John Ryan

School of Hospitality Management and Tourism
Faculty of Tourism and Food
Dublin Institute of Technology, Cathal Brugha Street
Dublin 1, Ireland.

ABSTRACT

While many developments have taken place around supporting the model coding task of simulation, there are few tools available to assist in the conceptual modelling phase. Several authors have reported the advantages of using process modelling tools in the early phases of a simulation project. This paper provides an overview of process modelling tools in relation to their support for simulation, categorizing the tools into formal method and descriptive methods. A conclusion from this review is that none of the tools available adequately support the requirements gathering phase of simulation. This is not surprising as none of the process modelling tools were developed for explicit support of simulation. The paper then presents results of research into developing a new process modelling method for simulation.

1 INTRODUCTION

In 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 analysts time should be divided as follows (Harrell and Tumay 1995):

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.

Many developments have taken place around supporting the "model coding or translation task" of a simulation model with highly developed modelling environments now available. But there have been very few tools developed to support the tasks prior to coding. It is in the development of a tool to aid in the capture and communication of knowledge within these phases that this research is undertaken.

2 SUPPORT FOR CONCEPTUAL MODELLING

During the initial stages of developing a simulation model, a means of presenting the current system and proposed simulation (or conceptual) model is typically required. This may be simply documentation of system description with diagrams or in some cases a process modelling tool may be used. A number of researchers have documented the benefits of using process modelling tools to support the initial stages of a simulation project (Nethe and Stahlmann 1999, Jeong 2000, Perera and Liyanage 2000, van Rensburg and Zwemstra 1995). There are numerous process modelling tools available to aid in the modelling of a system. Kettinger, Teng, and Guha (1997) listed over 100 in a survey that was not exhaustive. These tools are capable of modelling 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 (2003) and a number have been used to support simulation i.e., van Rensburg and Zwemstra (1995) and Al-Ahmari and Ridgway (1999). To ascertain the level of support given by current process modelling tools a selective review of a number of methods/tools was carried out (Ryan and Heavey 2006). The review focused on methods/tools that have been used to support simulation and/or exhibit characteristics desirable in a dedicated process modelling tool for simulation. The methods/tools were categorized into:

Formal Methods: These are methods that have a formal basis and there are numerous software implementations of these methods. Methods reviewed under this category were: **(i)** Petri Nets (Ratzer et al. 2003); **(ii)** Discrete Event System Specifi-

Table 1: Main Characteristics for Evaluation

Characteristic	Description
Communication	The ability of the method to communication system information, especially to non-experts.
State	The ability of the method to model state changes in a system.
Information	The ability of the method to model information flow in a system.
Resources	The ability of a method to model resources used in a system.
Branching	The ability of the method to model complex branching logic.
Elaboration	The ability of the method to allow elaboration of system descriptions.

cation (DEVS) (Zeigler 1984); **(iii)** State Charts (Harel 1987); **(iv)** Activity Cycle Diagrams (ACD) (Tocher 1963) and **(v)** Event Driven Process Chains (EDPC) (Tardieu, Rochfeld, and Colletti 1983).

Descriptive Methods: Methods that have little formal basis and are primarily software implementations. Methods reviewed here were: **(i)** IDEF (NIST 1993); **(ii)** CIMOSA (Vernadat 1998); **(iii)** Integrated Enterprise Modelling (IEM) (Mertins, Jochem, and Jakel 1997); **(iv)** Role Activity Diagrams (RAD) (Ould 1995); **(v)** GRAI Method (Doumeingts 1985) and **(vi)** UML State Charts and Activity Diagrams (Muller 1997).

The main characteristics used for evaluation are listed in Table 1.

In summary this review concluded that Petri nets are to a certain extent capable of visually representing and communicating discrete event system logic, however such Petri net models are not capable of visually accounting for complex branching logic or hierarchically decomposing complex models into sub models and as a result become very cumbersome as system complexity increases. The technique also does not account for a user's viewpoint, resources, information flows or a means of elaborating the graphical model in a textual manner. However the technique is capable of accurately representing state flows and the activities associated with the execution of such flows.

ACDs are again somewhat capable of visually representing and communicating certain discrete event system logic. It achieves this by means of modelling state flows and the activities that cause such state flows to be executed. However the technique fails to account for a user's perspective, resources, information modelling, branching logic or a means of textually elaborating graphical models.

The DEVS formalism is capable of accurately representing the various changes in state of a discrete event system along with being somewhat capable of representing resources, activities and branching within its mathematical representation. However the formalism is not visual in nature and does not account for the user's interactions with the system, information flows or a user friendly elaboration language.

UML activity diagrams are designed to represent a discrete event system as a series of activities linked together to show the various phases of activity within a discrete event system. The technique is highly visual and communicative and also has to a certain extent a means of visually representing the logical flow of activities. However the system does not account for a user's perspective, state flows, information modelling, resource modelling or a means of elaborating the graphical models. UML statecharts are a highly visual and communicative modelling technique that are used represent a discrete event system as a series of interrelated state flows. This technique also has a means of graphically representing the logical flow of states and hierarchically decomposing a model into sub models. However the system does not account for information flows, resources, activities, and an inclusion of a user's interaction with the system or a means of textually elaborating the graphical model.

RADs are a highly visual modelling technique that accounts for the user's perspective in the development of a process model of a discrete event system. The technique is to a certain extent also capable of representing the logical branching of such activities within a model. The technique however does not have the means of representing state flows, information flows, resource interactions or a means of either hierarchically decomposing or textually elaborating graphical models.

The GRAI model offers a means of modelling the detailed information and control interactions within a discrete event system. This information model is also capable of representing discrete activities and model decomposition along with to a lesser extent both state flows and resources. However the model does not account explicitly for the user's perspective, branching logic or an elaboration language.

The IEM technique presents a highly visual and communicative model of a discrete event system, which is capable of graphically representing state flows, information and resource elements. The technique is also capable of hierarchically decomposing a model into sub models along with having a detailed branching logic associated with it. However the technique does not account for a user's viewpoint or have an associated elaboration language.

EDPCs are a highly graphical process modelling technique which are capable of representing a discrete event system as a series of activities. The technique is capable of representing branching logic and to a lesser extent information interactions within the system. Drawbacks of the

system however include its lack of a representation of the user's perspective, state flows, and resource interactions. The technique also does not have the capability to hierarchically decompose a model into sub models or have access to an associated elaboration language.

IDEF0 is a graphical modelling technique capable of representing a discrete event system as a series of interrelated activities. The technique is capable of hierarchically decomposing a model into sub models and is also to a certain extent capable of accounting for both information and resource interactions. However the technique does not account for system branching, the elaboration of graphical models, state flows or the modelling of a user's perspective. The IDEF3 process modelling technique is capable of graphically representing the various states through which a discrete event system can transition along with the various activities associated with each change of state. This technique also offers a means of representing complex system branching logic along with a means of hierarchically decomposing a model into related sub models. The technique is also capable of textually representing the graphical models, however this representation language is abstract in nature. This representation language also offers a means of representing resources associated with the graphical models. However the technique does not account for information flows or modelling from a user's perspective.

Resources are a major issue in many simulation projects. Techniques such as IEM and EDPCs are capable of accurately representing such resources within a discrete event system. To a lesser extent IDEF0, IDEF3, GRAI, RADs and DEVS can represent aspects of resources within a discrete event system. However techniques such as Petri Nest, ACDs, UML activity diagrams and UML statecharts do not have such a means of representing such resources. Activities are also well represented within many techniques such as Petri nets, ACDs, UML activity diagrams, RADs, GRAI, IEM, EDPCs, IDEF0 and IDEF3. While the DEVS technique is capable of representing activities to a lesser extent. Certain techniques such as UML statecharts are not designed to represent such activities.

Complex branching logic is well represented with techniques such as UML activity diagrams, UML statecharts, EDPCs and IDEF3 by means of the branch types used in each. Techniques such as Petri Nets, DEVS, RADs and IEM have the ability to represent such branching to a lesser extent. While techniques such as IDEF0, GRAI and ACDs lack the capability to display such branching logic. Finally no technique examined apart from the IDEF3 technique was capable of presenting the user with an elaboration language to further explain the graphical model produced. While the IDEF3 technique did have this capability the elaboration language was abstract in nature and not easy to reason over.

From the analysis above it is concluded that while there are many process modelling techniques and software tools available that may be used to support the requirements gathering phases of a simulation project, none of the techniques reviewed fully support the conceptual modelling phase of a simulation project. As a result of this review research has been carried out into developing a process modelling specifically tailored to support the conceptual modelling phase of a simulation project. The following design objectives were used in developing the process modelling method:

- The technique has to be capable of capturing a detailed description of a discrete event system;
- The technique should have a low modelling burden and therefore be capable of being used by non specialists;
- The technique should present modelling 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.

The resulting process modelling tool is called Simulation Activity Diagrams (SAD) and is briefly described in the next section.

3 SIMULATION ACTIVITY DIAGRAMS (SAD)

A brief overview of the Simulation Activity Diagram (SAD) is presented in this section.

3.1 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 1 shows

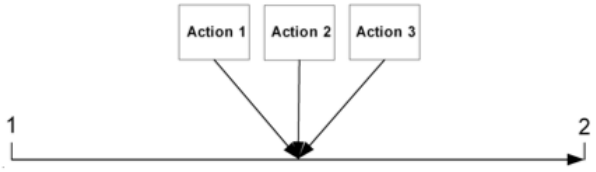

Figure 1: SAD Actions

an activity consisting of three actions, which are executed as follows.

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 modelling primitives that may be used in the detailing of such an activity.

3.2 SAD Modeling Primitives

Within most systems, actions such as those in Figure 1 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 modelling 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. For example, within a system being simulated a primary resource, such as a machine may be used in the transformation of an entity from state A to state B. However this primary resource may require an operator and a number of other tools that an operator may use to operate the machine.

- **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. Standard branching elements such as, Asynchronous AND, Synchronous AND, Asynchronous Exclusive OR, Asynchronous Inclusive OR and Synchronous Inclusive OR are available in SAD.

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 2.

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.

Figure 2: SAD Link Types

3.3 SAD Model Structure

A SAD model is executed in time sequenced ordering from left to right and from the centre auxiliary resource area to the extremities of the model and is structured as follows, see Figure 3. At the centre 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 SAD modelling 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 9 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 4, only the actor auxiliary resource element "Actor 1" 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,

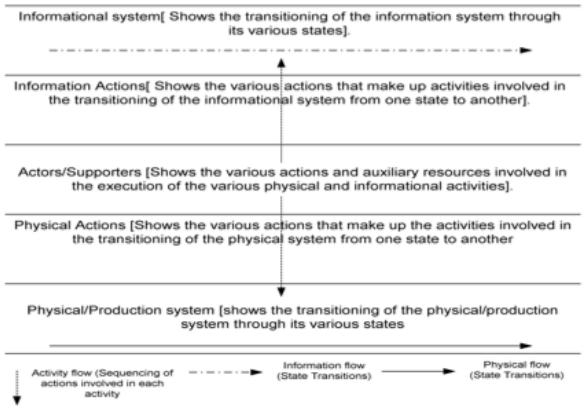

Figure 3: SAD Model structure

"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". 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.

3.4 Elaboration of SAD Models

Thus far, the modelling elements used to develop a SAD model have been introduced to provide a means of visually modelling 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 2.

4 TESTING OF SAD

A paper-based testing of the SAD technique was carried out on a number of real systems. Each system was chosen to highlight the ability of the technique to visually model and communicate a different aspect of a discrete event system. The first model developed represented a simple Kanban system. This was successfully used to validate the technique's ability to represent divergent production and information/control systems within the same model. The second paper-based model highlighted the SAD's ability to accurately represent different levels/views of the same

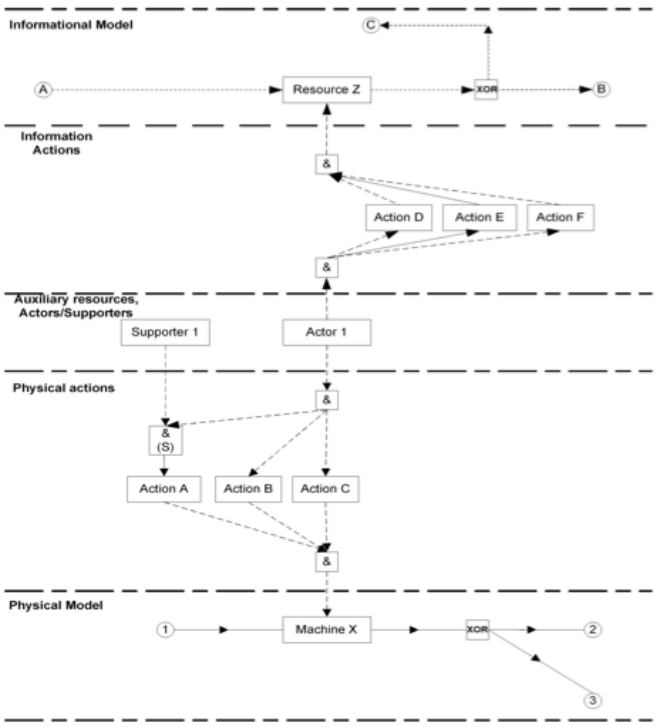

Figure 4: A Simple SAD Example

Table 2: Structured Language

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.
TRANSITIONS TO	Specifies the change of state of entity or information from one state to another

information within a model and capture interactions between various sub-systems. The third paper-based test example modeled a furnace area within a batch production flow-shop. This example model was used to highlight the SAD technique's ability to visually represent and communicate detailed operator-system interactions.

Following from this paper based testing a prototype software application called the PMS (Process Modeling Software) has been developed using Microsoft Visual C++ to implement the SAD methodology (Ryan 2005). 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. This system is currently undergoing a further set of validation tests on real world production systems in a further effort to validate the technique's ability to capture and visually communicate detailed system information. An example screen from the PMS software is shown in Figure 5.

5 CONCLUSIONS

The conceptual modelling phase of a simulation project is important in the overall context of a simulation project. From a review of this area it can be seen that little research has been carried out in this area. However, examples can be found in the literature documenting the benefits of using process modelling tools in the conceptual modelling phase of a simulation project. From a review of process modelling tools that are available gaps can be found in the support

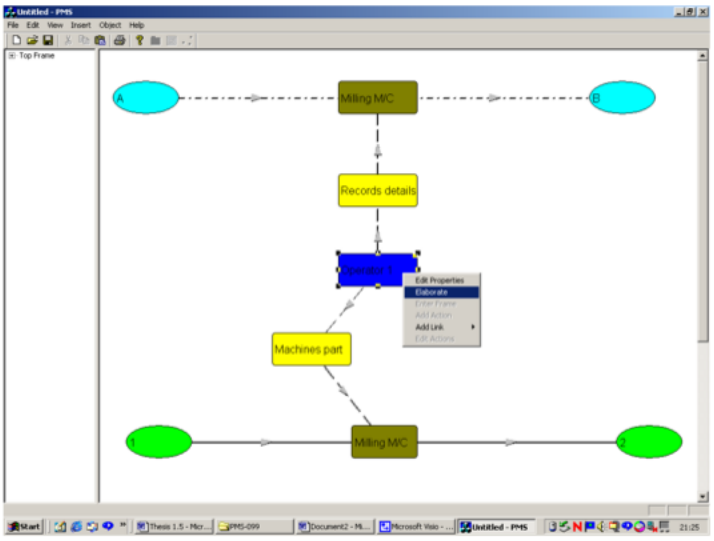

Figure 5: Example Screen from the PMS Application

they provide for simulation. This is not surprising as none of them were developed specifically to support simulation. Research has been carried out into developing a process modelling tool that provides better support for simulation than currently available. The result of this is a method called SAD which has been implemented in a software application called PMS. The efficacy of this tool has been tested on a small test set with further testing currently underway.

REFERENCES

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

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

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

Harrell, C., and C. Tumay. 1995. *Simulation made easy - a manager's guide*. Norcross, Georgia, USA: Industrial Engineering and Management press.

INCOME Process Designer 2003.

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

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

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

Mertins, K., R. Jochem, and F. W. Jakel. 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.

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

NIST 1993, 21/12/1993. 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 K. Liyanage. 2000. Methodology for rapid identification and collection of input data in the simulator of manufacturing systems. *Simulation Practice and Theory*:645–656.

Ratzer, A. V., L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. 2003. CPN tools for editing, simulating, and analysing coloured Petri nets. In *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003*, ed. W. van der Aalst and E. Best, Lecture Notes in Computer Science, 450–462. Eindhoven, The Netherlands: Springer-Verlag Heidelberg.

Ryan, J. 2005. *Development of a process modelling system for simulation*. Ph. D. thesis, University of Limerick.

Ryan, J., and C. Heavey. 2006. Process modeling for simulation. *Computers in Industry* 57:437450.

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

Tardieu, H., A. Rochfeld, and R. Colletti. 1983. La mthode merise. In *Principes et outils, les ditions d'organisation*.

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

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

Vernadat, F. 1998. *Handbook on architectures of information systems*, Chapter The CIMOSA Languages, 243–264. Springer-Verlag.

Zeigler, B. P. 1984. *Multifacetted modelling and discrete event simulation*. London: Academic Press.

AUTHOR BIOGRAPHIES

CATHAL HEAVEY is a Senior Lecturer of Operations Management in the Department of Manufacturing and Operations Engineering at the University of Limerick. He is an Industrial Engineering graduate of the National University of Ireland (University College Galway) and holds a M. Eng. Sc. and Ph.D. from the same University. He has published in the areas of queuing and simulation modelling. Research interests are: Simulation Modelling of Discrete Event Systems; Modelling and Analysis of Supply Chains and Manufacturing Systems; Process modelling; Component-based simulation; Decision support systems. His e-mail address is <cathal.heavey@ul.ie>.

JOHN RYAN received a BEng (Hons) Industrial Engineering from the University of Limerick in 1997. He completed his PhD at the same University in 2005. He is currently a lecturer in operations management in the school of hospitality management and tourism, at the Dublin Institute of Technology. Dr Ryans research interests include Process modelling, Operations improvement, Business process reengineering, Service operations management and operations research. His e-mail address is <john.ryan@dit.ie>.