Dublin Institute of Technology
ARROW@DIT

Conference papers

School of Hospitality Management and Tourism

2006-01-01

Requirements Gathering for Simulation

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

Cathal Heavey
University of Limerick

Follow this and additional works at: http://arrow.dit.ie/tfschmtcon

Part of the Industrial Engineering Commons, Other Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Engineering Commons

Recommended Citation
Ryan, J., Heavey, C.: Requirements Gathering for Simulation. United Kingdon Operational Research Society, Simulation Study Group, 3rd Simulation Workshop, 2006

This Conference Paper is brought to you for free and open access by the School of Hospitality Management and Tourism at ARROW@DIT. It has been accepted for inclusion in Conference papers by an authorized administrator of ARROW@DIT. For more information, please contact yvonne.desmond@dit.ie, arrow.admin@dit.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

School of Hospitality Management and Tourism

Books / Book chapters

Dublin Institute of Technology *Year* 2006

Requirements Gathering for Simulation

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

Cathal Heavey Dr.
UL

This paper is posted at ARROW@DIT.
http://arrow.dit.ie/tfschhmtbook/1

— 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.

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit:

- URL (human-readable summary):
  http://creativecommons.org/licenses/by-nc-sa/1.0/
- URL (legal code):
  http://creativecommons.org/worldwide/uk/translated-license

Proceedings of the 2006 OR Society Simulation Workshop
S. Robinson, S. Taylor, S. Brailsford and J.Garnett, eds.

REQUIREMENTS GATHERING FOR SIMULATION

Dr. John Ryan
School of Hospitality management and Tourism
Faculty of Tourism and Food
Dublin Institute of Technology
Cathal Brugha Street
Dublin 1.
John.ryan@dit.ie

Dr. Cathal Heavey
Department of Manufacturing and Operations Engineering
Schrodinger Building
University of Limerick
Limerick
Cathal.heavey@ul.ie

ABSTRACT:
This paper discusses the current shortfalls in support for the pre-coding phases of a discrete event simulation project. The paper then presents a process modelling technique, Simulation Activity Diagrams (SADs), developed to specifically support the initial requirements gathering phases of a simulation project. The paper concludes with an outline of proposed future developments to the modelling technique.

Keywords: Simulation, Requirements Gathering, Process Modelling

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 quoted in simulation texts. The rule states that, in developing a model, an analyst's time should be divided as follows [1]:

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

It is rare for these phases to be totally independent. For example, in the pre-coding phases 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 modelling with the shaded tasks depicting the application area of the proposed modelling technique within the overall modelling process [2].

As can be seen many 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. These pre-coding or conceptual modelling phases are not unimportant within the overall structure of a simulation project [3]. It has been argued that such conceptual process models may even lead to the discovery of a solution to a problem without the necessity of simulating the process [3]. Therefore, the process of developing an accurate process model of a discrete system prior to the development of a simulation model is an extremely important one. However there is a severe lack of publications on the overall subject of conceptual modelling [3], [4], [5]. Many developments have taken place around supporting the "model coding or translation task" of a simulation model with highly developed modelling tools such as EM Plant [6], Arena [7] and Flexsim [8]. But there have been very few techniques or tools developed to explicitly support the tasks prior to coding a simulation model. The problem definition, requirements gathering and conceptual model formulation process is often a time-consuming one, as is the process of collecting detailed information on the operation of a system [2].
Hollocks [9] recognised that such pre modelling and post experimentation phases of a simulation project together represent as much or more effort

than the modelling section of such projects and that software support for these phases of the wider simulation process would be valuable. Some of the particular areas of potential support highlighted by Hollocks included documentation, communication, and administration. Such areas are also discussed by Sargent [10] in terms of model documentation, and model validity. This lack of support for documentation in preference for rapid model production was further highlighted by Cornwell et al. [11], who claimed that only 2% of software systems such as modelling and simulation are usable upon delivery. This they ascribe points to the lack of development, documentation, maintenance and management practices for software development, which if in place can result in systems that can provide greater returns on investment and that can be used and evaluated for suitability without the need for costly rework. The difficulties of establishing model credibility due to the lack of good development practices and documentation are also discussed. Nethe and Stahlmann [12] discuss the practice of developing high level process models prior to the development of a simulation model. Such a method they feel would greatly aid in the collection of relevant information on system operations (i.e. data collection) and therefore reduce the effort and time consumed to develop a simulation model. Such a process modelling method for simulation could be used as a knowledge acquisition method for simulation studies. The above highlight both the importance of and lack of pre-coding support for simulation. The research outlined in this paper was undertaken in the development of a process modelling technique to aid a simulation model developer during the pre-coding/requirements gathering phases of a discrete event simulation project in an attempt to overcome some of these shortfalls.

2. DEVELOPMENT OBJECTIVES

The objective of the work reported in this paper was to develop a process modelling technique known as Simulation Activity Diagrams (SADs) to aid a simulation model developer during the pre-coding/requirements gathering phases of a discrete-event simulation project.

The more detailed goals emanating from the primary objective above are the development of a technique that:

- Could capture a detailed description of the various aspects of a DES for the purposes of a simulation project, those being;
    - 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 the execution of the activities associated with both work and information within a discrete event system;
- Has a low modelling burden and therefore can be used by non-specialists; aspects that may facilitate this include:
    - The modelling 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 modelling 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.
- Presents modelling information in terms of concepts that are meaningful to system personnel such as resources and activities, as opposed to abstract terms,
- Facilitates understanding and communication.
- Has a good visualisation capability to facilitate communication between a model developer and system personnel.

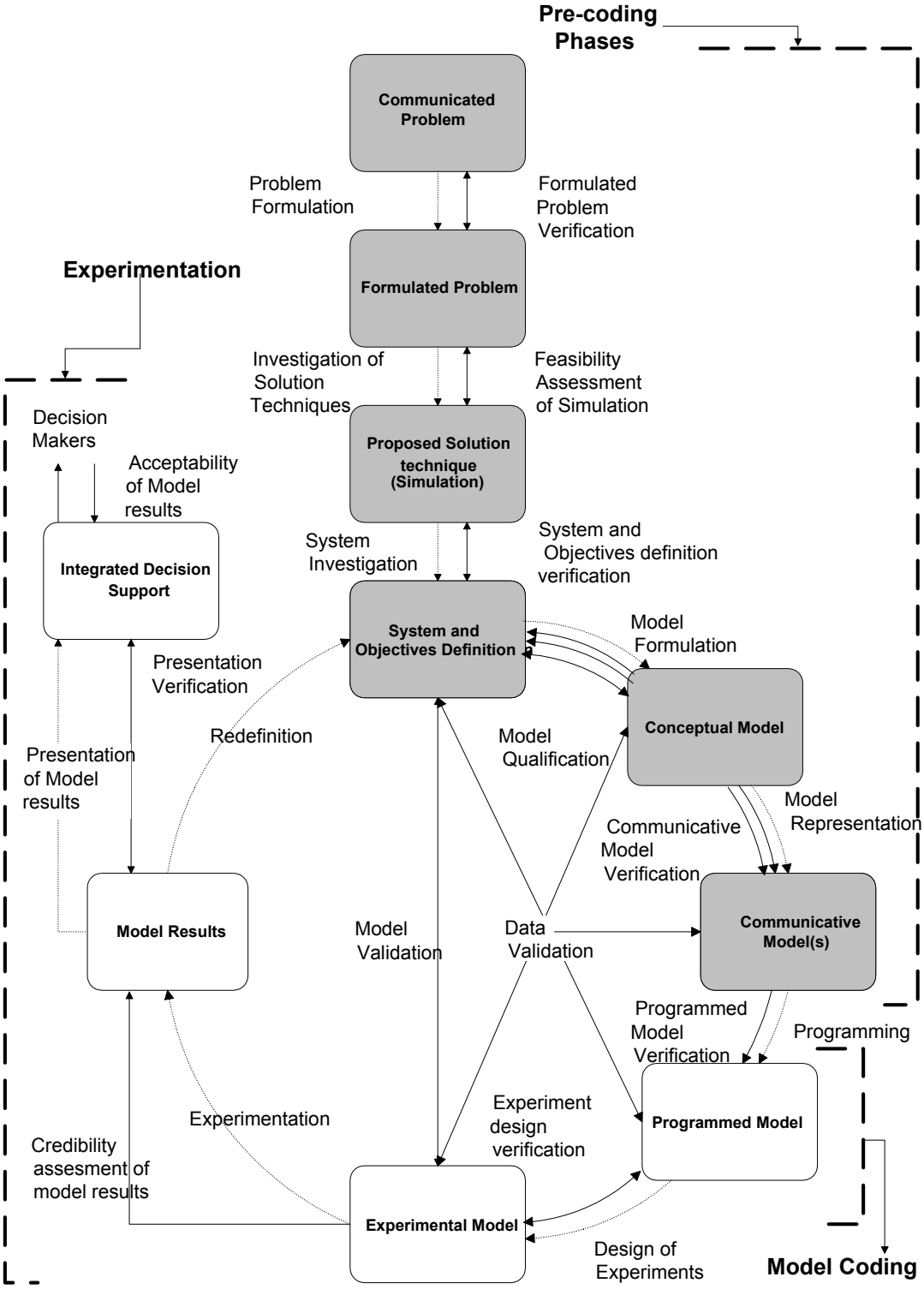

Figure 1 The Life cycle of a simulation Study [2]

In summary, the above requirements were developed to allow for the development of a process modelling technique that was capable of facilitating communication and understanding between a simulation model developer and system personnel, while simultaneously being capable of aiding in the pre-coding/requirements gathering phases of such a project.

3. SAD DEVELOPMENT PROCESS

The SAD development process initially involved a detailed review of process modelling techniques developed and used to support the pre-coding phases of a simulation project. This initial review highlighted the lack of research in this area. No techniques specifically developed to support these pre-coding phases of a simulation project were found. Noted authors in the field of simulation modelling such as Law and Kelton [13], give little more than an introduction to the field. Robinson [3] also highlights the lack of research in this area. This lack of research points to what may be viewed as a traditional narrow focus on simulation modelling support that fails to account for the broader modelling considerations as highlighted by a number of authors [14], [15]. As a result of this gap in the literature in relation to this specific area, the focus of the literature review changed scope to a broader review of process modelling techniques that it was felt were capable of modelling a discrete event system. By taking such a broad approach to the literature review it became apparent that there were many process modelling techniques available, which were broadly capable of satisfying some of the required criteria. Kettinger et al. [16] quoted more than one hundred in a study that was not exhaustive. As a result it was deemed impractical to attempt to review every such technique. The focus of the literature review was then narrowed to process modelling techniques capable of or deemed to be suited to supporting the pre-coding phases of a simulation project even if such techniques had not been specifically developed for such a purpose. Again many techniques were examined which were proposed as being capable of modelling a discrete event system for the purposes of among others simulation. However due to their extremely broad scope and all encompassing nature a number of these techniques were deemed to be unsuitable to the specific nature of the problem area being examined. However a number of techniques were identified that were seen to be broadly focused on the problem area in question and also capable of somewhat representing complex discrete event logic. Figure 2 below gives a summary of each technique reviewed under the specific categories listed in the requirements. The grading under which each technique is listed is as follows:

- **High (H)** Highlights that the technique was very capable of fulfilling this requirement;
- **Medium (M)** Highlights that the technique was somewhat capable of fulfilling this requirement;
- **Low (L)** Highlights that the technique was not capable of fulfilling this requirement.

Technique	Good Communication / Visualisation medium	User Perspective	State flow modelling	Information flow modelling	Resource modelling	Activity Modelling	Complex branching logic	Decomposition	Elaboration Language
Petri Nets	Medium	Low	High	Low	Low	High	Low	Low	Low
ACDs	Medium	Low	High	Low	Low	High	Low	Low	Low
DEVS	Low	Low	High	Low	Medium	Medium	Medium	Low	Low
UML Activity Diagrams	High	Low	Low	Low	Low	High	Medium	Low	Low
UML Statecharts	High	Low	High	Low	Low	Low	Medium	High	Low
RADs	High	High	Low	Low	Low	High	Medium	Low	Low
GRAI	Medium	Low	Medium	High	Medium	High	Low	Medium	Low
IEM	High	Low	High	High	Medium	High	High	High	Low
EDPCs	High	Low	Low	Medium	Low	High	High	Low	Low
IDEF0	High	Low	Low	Medium	Medium	High	Low	High	Low
IDEF3	High	Low	High	Low	Medium	High	High	High	Medium

Figure 2 Requirements satisfaction attributed to reviewed techniques

.

As is shown above the literature review concluded that no technique examined was adequately equipped to fully support the requirements outlined in section 2. As a result the development of the Simulation Activity Diagrams (SADs) was undertaken. The initial development process focused primarily on the state or entity flows through a discrete event system. This was primarily examined as the majority of process modelling techniques concentrated on representing this element of a discrete event system and through an iterative series of discussions with a number of simulation experts the technique was further developed to include the various aspects of a modern discrete event system for the purposes of requirements gathering. The technique developed will be briefly presented in the following section.

4. SIMULATION ACTIVITY DIAGRAMS (SADS)

The technique presented here (Simulation Activity Diagrams (SADs)) 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. A brief overview of the Simulation Activity Diagram (SAD) is now presented.

4.1 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 3 shows an activity consisting of three actions, which are executed as follows.

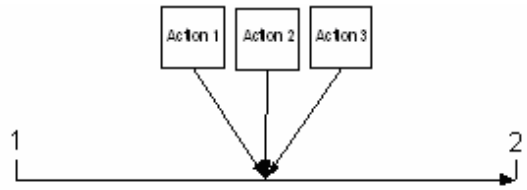

Figure. 3. SAD Actions.

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

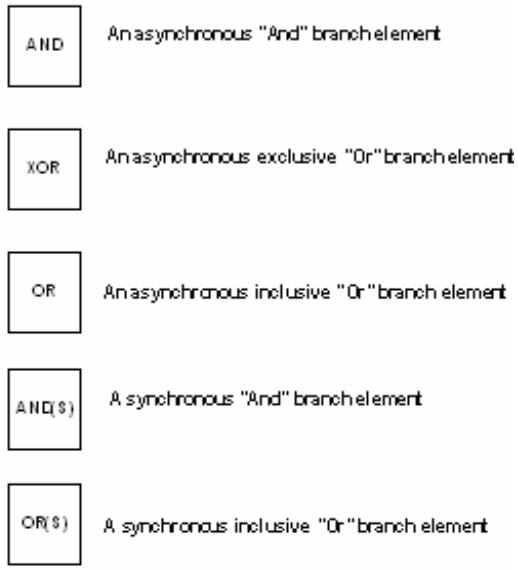

Figure. 4 SAD Branching elements.

4.2 SAD MODELLING 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 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;

An informational state element 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 modelling technique.

These branching elements are used to eliminate ambiguous instances that may occur in complex models. For example on examination of the elements in Figure 5 a number of semantic ambiguities become apparent. Firstly the links between auxiliary resources, "actor" and "supporter", and the actions shown are ambiguous. In this instance the meaning of the links are unclear, either one or both of the auxiliary resources may be necessary for the execution of each action or any number of the actions. A similar ambiguity may arise within the graphical representation of the various phases of execution within a system.

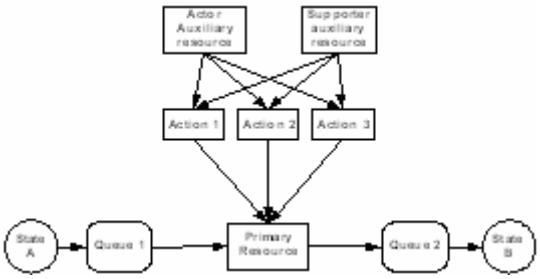

Figure 5 SAD model without branching elements

To overcome such ambiguous situations, the branch elements can be used as shown in Figure 6. In this diagram the branching elements are used to model the divergence of the links into multiple paths by means of an asynchronous "AND" branch in each case. This graphically represents the fact that each of the auxiliary resources are used in the execution of the three actions. The convergence of these links back into a single path is also represented by a branch element in this instance a synchronous, "AND(S)" branch. This graphically represents the fact that each of the two links converging at this branch should be present simultaneously for the execution of the exiting link. In other words both the actor and supporter auxiliary resources have to be present at the same time for the execution of each of the actions 1,2 and 3. Finally the use of the and asynchronous branch, "AND", to link actions 1, 2 and 3 with the primary resource element indicates that the actions 1, 2 and 3 have to be executed prior to the SAD model advancing past the primary resource element. In other words the three actions have to be executed prior to any transformation of an entity taking place.

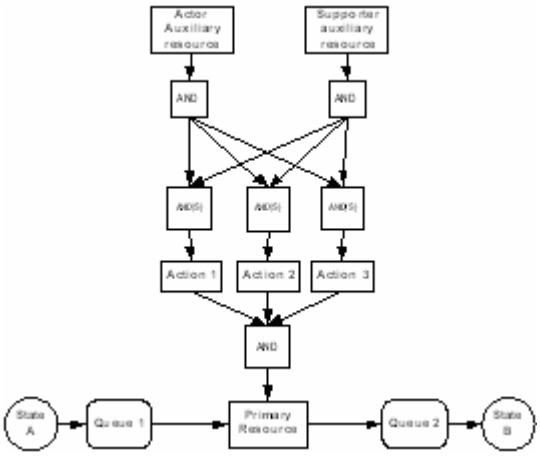

Figure 6 SAD model with branching elements

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 7.

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.

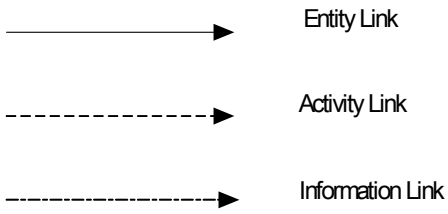

Figure. 7. SAD Link Types.

4.3 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, Figure 8. 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 9, 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.

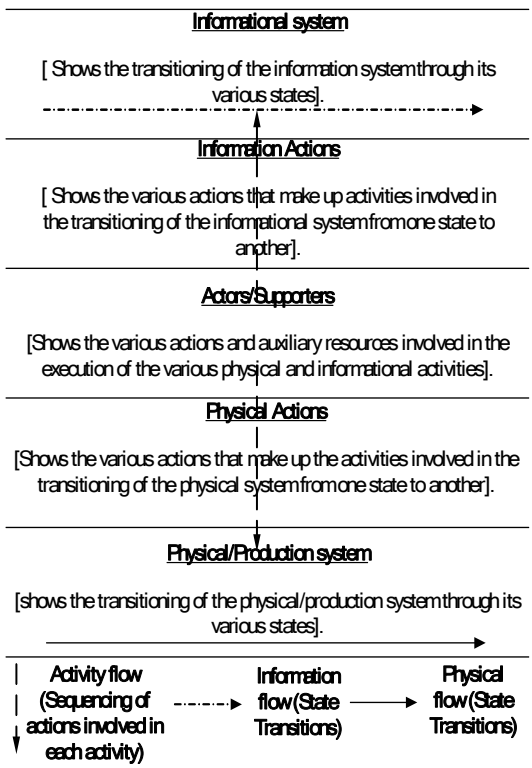

Figure. 8 SAD Model structure.

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.

4.4 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, Table 1. These words are used to describe the various interactions that take place in a SAD diagram.

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

Table 1 Structured language

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,AND(S), OR, OR(S) and XOR.

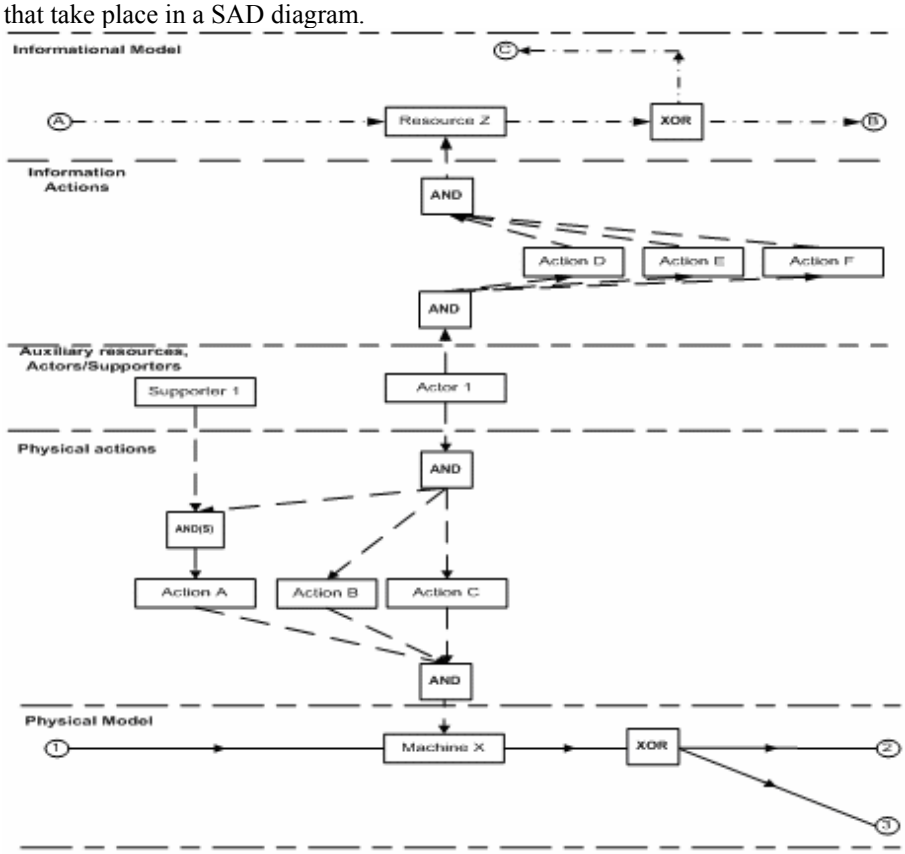

Figure. 9. A Simple SAD.

5. PROCESS MODELLING FOR SIMULATION SOFTWARE

A prototype software application called the PMS (Process Modelling for Simulation) 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 modelled with the aim of validating the SAD technique. Systems modelled 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.

6. PROPOSED USAGE AND FUTURE DEVELOPMENT OF THE SAD TECHNIQUE/PMS TOOL

The SAD technique and PMS tool can currently be used to support a simulation model developer during the requirements gathering phase of a simulation project. As can be seen from Figure 10 such a phase would involve discussions with systems personnel on the requirements and the model being developed. To this end the PMS tool combines the high level semantics of the SAD technique with the automatic generation of a high level textual language to support communication and understanding between the model developer and systems personnel.

A further enhancement to this will be the step through facility, which will explicitly link the textual language and the SAD model to further support communication and understanding. However as can be seen from Figure 10 while the requirements gathering phase of a simulation project is supported currently the conceptual modelling phase, which is the next phase in the progressing of a simulation project is not. To facilitate the support of this phase of a simulation project it is proposed to develop a versioning module within the PMS tool. Such a versioning module would allow for the requirements model to be reduced or versioned within a separate screen thus allowing for the conceptual model to be developed, while still being explicitly linked to the requirements model. The explicit linking of the requirements model and conceptual model in this way would further support communication and understanding of the overall simulation model being developed as the conceptual model developed would be used to form the basis of the simulation model as shown in Figure 10.

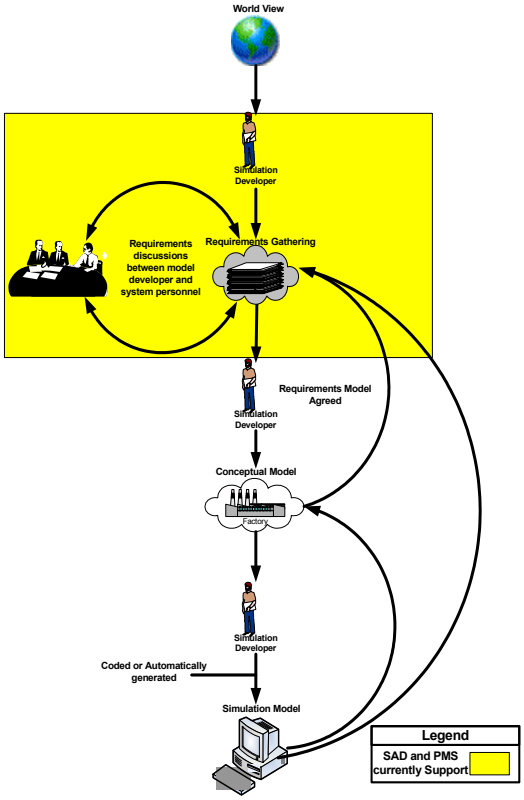

Figure 10 SAD and PMS Current sphere of usage

The SAD technique is currently being used on a pilot basis in a simulation project within a major electronics manufacturer with a view to further validating and developing the technique.

7. CONCLUSIONS

The pre-coding phases of a simulation project are important in relation to the overall success of a simulation project. This paper highlighted the fact that there is inadequate support currently available for this phase of a simulation project. While numerous process modelling techniques are available and several have been used to support the pre-coding/requirements gathering of a simulation project, the paper argues that the techniques available do not provide adequate support. The paper presented an overview of a process modelling technique, Simulation Activity Diagrams (SAD) developed to endeavour to overcome some of the current shortfalls highlighted. 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 modelling method uses the various SAD modelling 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 technique also allows for the modelling 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 technique is proposed to allow a user to understand and validate a SAD model. Currently, the technique is being further developed and validated.

8. REFERENCES

[1] Pegden, C.D., Sadowski, R.P. and Shannon R.E. (1995) Introduction to Simulation using Siman, McGraw-Hill Education.

[2] Balci, O. (1986) "Credibility Assessment of Simulation Results", in Proceedings of the 1986 Winter Simulation Conference, Piscataway, NJ, pp 38-43

[3] Robinson, S. (2004) Simulation: The Practice of Model Development and Use, Wiley.

[4] Ryan, J. and C. Heavey. Simulation Activity Diagrams. In 32nd International Conference on Computers and Industrial Engineering. 2003. Limerick, Ireland, Editor, C. Heavey.

[5] Ryan, J. and C. Heavey. Process Modelling for Simulation. in 19th International Manufacturing Conference. 2002. Belfast, Northern Ireland.

[6] EMPlant http://www.tecnomatix.com. Date Accessed 25/10/2005

[7] Kelton, D.W., Sadowski, R.P. and Sadowski, D.A. (1998) Simulation with Arena, McGraw-Hill.

[8] Flexsim http://www.flexsim.com/ Date Accessed 25/10/2005

[9] Brian W. Hollocks: (2001) "Discrete-event simulation: an inquiry into user practice" in Simulation. Practice and Theory, Vol. 8(6-7): pp 451-471

[10] Robert G. Sargent: (1999) "Validation and verification of simulation models" In proceedings of the 1999 Winter Simulation Conference: pp 39-48

[11] Conwell, C.L., R. Enright, and M.A. Stutzman. (2000) "Capability Maturity Models Support of Modelling and Simulation Verification, Validation, and Accreditation" In Proceedings of the 2000 Winter Simulation Conference, Orlando, Society for Computer Simulation International.

[12] Nethe, A., and Stahlmann H.D. (1999) "Survey of a General Theory of Process Modelling", in International Conference on Process Modelling, Cottbus, 22-24. February 1999.

[13] Law, A M. and Kelton, W.D. (2000). Simulation Modelling and Analysis. McGrawHill, 3rd edition.

[14] Sagasti, F. R. & Mitroff, I. I. 1973. Operations research from the viewpoint of general systems theory. OMEGA, 1:695-709

[15] Landry, M., J-L. Malouin, M. Oral, (1983), "Model Validation in Operations Research," European Journal of Operational Research, Vol.14, pp.207-220 (Invited Paper).

[16] Kettinger, W.J., Teng, J.T.C. and Guha, S. (1997) "Business process change: A study of methodologies, techniques and tools", MIS Quarterly, Vol. 21(1) pp 55-80.

9. AUTHOR BIOGRAPHIES

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 Ryan's research interests include Process modelling, Operations improvement, Business process reengineering, Service operations management and operations research.

CATHAL HEAVEY

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.