A PROCESS MODEL SIMULATION TOOL - MAPPING FROM PROCESS MODELS TO DISCRETE EVENT SIMULATION

Ana I. Martínez García*
Informatics Process Group, Room 2.110
Department of Computer Science
University of Manchester M13 9PL
e-mail: martinea@cs.man.ac.uk

ABSTRACT

This paper presents an overview of how business process modelling and simulation can be used together. In particular the development of a prototype tool that automatically generates simulation models from process models is illustrated. We describe the process of the development of the prototype, the different models involved, Role Activity Diagrams and Discrete Event Simulation and the mapping of the elements of one to the other.

INTRODUCTION

Process models are mainly used for representing and understanding the processes followed in organisations. Unfortunately the models currently available need more dynamic properties if they are going to be useful in helping managers to predict and understand the implications of change. Simulation of such complex systems is a way of promoting the understanding of current processes, and of proposed changes to improve their performance. Now we can see an increase in the application of simulation to the analysis and design of organisational processes and "we will continue to see the development of specialised tools that are easy to use and focused on this application domain" (Pegden 1997).

Process models are used to document organisational procedures in a consistent and uniform manner. They aim to aid the analysis of business processes, help in the finding of fundamental problems and aiding to address their solution by either a small evolution or radical redesign. When a redesign is applied, the best way to see if the change will actually work in the real system is by means of simulation, predicting the process performance and in helping to understand the impact of change. Thus, if existing process models are interfaced with a simulation tool, it will lead into a position to give a more complete and accurate approach of analysing processes. In this work some results arising from the development of a prototype tool with this characteristics is described. It is concerned with the mapping of a process captured on Role Activity Diagrams (RADs) to generate the input to a Discrete Event (DE) simulator.

Some possible explanations of why simulation is not being widely used as a tool for business process modelling simulation are: 1) There are just a few capabilities of interface or connection between simulation software and other process modelling tools (Gladwin and Tummay 1994), 2) The need to use both, the domain expert and the simulation analyst in the modelling process and 3) The long time and special skills needed for the effective use of simulation modelling techniques (Lingineni et al. 1995).

The aim of this work is to present the development of a prototype tool which seeks to address some of the above problems by facilitating the generation of simulation models from models already existing in many organisations in the form of process models. For this the differences and common features between RADs and Discrete Event Simulation (DES) are studied, mapping from one to the other aiming to address some of the communication problems between the domain expert and simulation analyst by facilitating the use of similar conceptualisations. Thus encouraging the reusability of process mapping efforts carried out previously in the organisations. This, in turn promoting a common ground for the communication and understanding of the basic process aspects: functional, organisational, behavioural and informational (Curtis et al. 1992).

In the following section a brief introduction to process model simulation and some of the quantitative measures that can be taken by means of simulation are described. This is followed by the description of our work in the development of a prototype tool for the automatic generation of simulation models from RADs.

*The author would like to thank CONACyT (México) for its financial support to carry out her PhD studies

PROCESS MODEL SIMULATION

"An organisational process consists of a group of logically related tasks that use the resources of an organisation to provide defined results in support of the organisation's objectives" (Gladwin and Tummay, 1994). Over the past few years, new tools have been developed for modelling business processes. Most of the tools capture the description of the process with a graphical representation (boxes, arrows, lines, etc.), showing the activities and goals of the organisational process. Also many of these tools allow some kind of analysis of the process depending on the technique (RADs, IDEF, Workflow, etc.) that the tool is using for the process representation. However few of them have the feature of generating consistent simulation models. An example of a tool that does, is described in (Lingineni et al. 1995; Thompson 1995), where this tool maps from process models captured in IDEF3 to DES in the WITNESS[1] simulation package.

On the other hand, some of the type of questions that can be addressed by simulation are: What is the total process cycle time?, How long do customers have to wait before being served?, What is the best way to schedule personnel?, Bottlenecks analysis (location and timing for processing them), etc. (Gladwin and Tummay 1994).

Process modelling and simulation are concerned with similar issues. However, the type of information that is gathered for the analysis and representation of their models is different. RAD is not time based, therefore it does not capture some of the basic information needed for the statistical analysis of a process.

OUR PROTOTYPE

In this work, we are building a process mapping prototype tool to support the capture, description, analysis of process models and then also to facilitate the generation of simulation models. This might help in the communication between managers and analysts and will enable the generation of quicker and more accurate simulation models based in the same process baseline.

It could be argued that there is no need to interface a process modelling tool to simulation, and that everything could be done directly with a simulation tool. But abstract process capturing without too much detail, cannot be performed with most simulation tools, instead all the process details have to be modelled before implementing the process in the simulation tool. One advantage of combining process capture and simulation, is the automatic generation of a simulation model from a process model. This provides the capability for initially generating a model of the general view

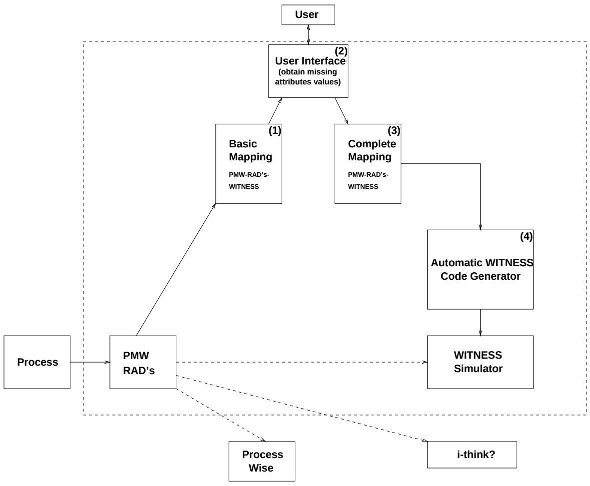

Figure 1: Proposed design of an automatic simulation model generator from process models.

of the process and subsequently to introduce details as more information about the process is being defined or obtained (Silver 1997).

In this prototype, as outlined earlier, the process models are assumed to be captured using RADs (a modelling technique where the process descriptions are based on Activities, Roles and their Interactions) (Ould, 1995). From this representation DES models are generated for the WITNESS simulation package. This is a DE visual interactive simulation tool, which supports modelling in terms of parts (entities), machines (servers), labours (resources), buffers (queues), etc. (Thompson 1995). In particular the basic process information is captured in the Process Modeller Workcentre[2] (PMW) as RADs, and then the first mapping from its elements (activities, roles, etc.) to the WITNESS simulation elements (machines, buffers, parts, etc) is performed. Because RADs are not time-based, the technique does not identify the dynamic properties or attributes of the activities that are needed for the simulation. Thus, the missing information (some simulation attributes such as cycle and arrival times, etc.) for the experimentation is obtained via the user interface. Then, with this information, the mapping is finished and the final simulation model is generated in the WITNESS Input Command Language. The model is generated to maintain consistency with the way in which the process was captured in RADs. The program is automatically loaded into the WITNESS environment and then very basic simulation (and animation) of the process can be performed.

The overall structure of the system is represented in the diagram shown in fig. 1, where the tool has the following elements: 1. **Basic Mapping.** A first mapping of the existing attributes on the RAD model to the WIT-

[1] WITNESS AT&T ISTEL is a discrete event visual interactive simulation tool, commercialized by Lanner Group

[2] PMW, developed by IPG of the University of Manchester

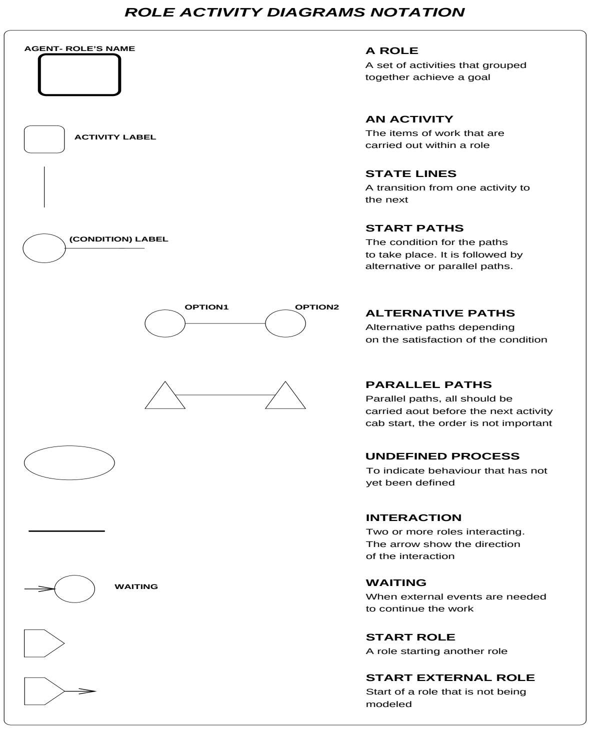

Figure 2: RADs elements and a brief description.

NESS simulator code. 2. **User Interface.** The different models do not map directly one to the other and the extra attributes needed for the simulation model are obtained here. 3. **Complete Mapping.** After adding the missing information a complete mapping is obtained and therefore also a complete model. 4. **Automatic WITNESS Code Generator.** This is the final system. The prototype has been tested with simple examples to check consistency in the mapping and then in a real case study with more complex models.

Role Activity Diagram Model

RAD is a structured technique for modelling processes, it focuses on roles, their interactions and their activities, as well as the connection with external events (Ould 1995). This technique is considered by some to be the most complete to represent most of the features of a process (goals, roles, decisions, etc.) (Miers 1996). The main graphic notation and a brief description of the RAD elements are given in fig. 2. The roles are represented by a bold rounded square with a label at the top specifying the role's name and the agent (a person or machine) responsible for it, the activities with small squares and its corresponding name label at the right side and, the roles' interactions are marked with bold lines going from the activity of one role where the interaction is initiated to the activity of other. A simple RAD example that will be used to demonstrate some concepts is illustrated in fig. 3. This is a very simplistic view of a request to an insurance process, which consists of two **roles**: the *customer* and the *clerk*, where each one has, as part of their roles, a set of **activities** such as a *request of insurance* (customer) or to *create a customer record* (clerk). Two **interactions** (contacts) are carried out, one for the request for insurance and the second to inform the customer of a decision. The idea of this example is only to demonstrate the concepts, our tool is generic independent of this application domain and works for more complex process models.

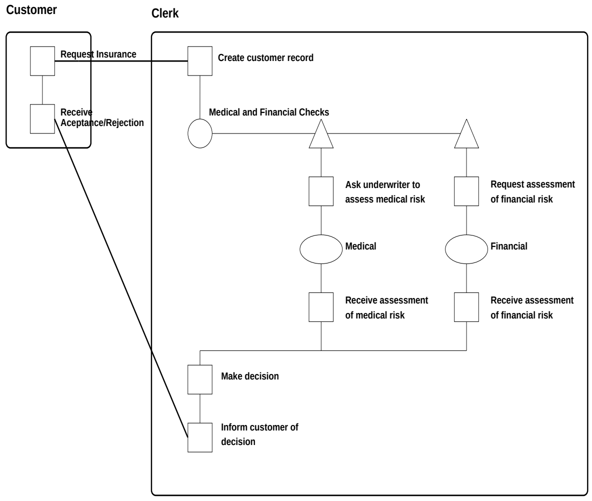

Figure 3: Simple Insurance Request Model in RADs.

Discrete Event Simulation Model in WITNESS

These types of simulation are composed of objects called entities that have some properties or attributes. The state of the system is a collection of attributes or state variables that represent the entities of the system at a given time. At these times (different discrete points in time) some event occurs. The system state may be changed when an event happens (the change in the properties or attributes of the entities in a specific time). This event starts an activity that occupies a length of time, during which the entities of the system perform some operations. The entities, attributes, events, activities and the relationships between them are defined in the model of the system (Law and Kelton 1991). The WITNESS simulator is a DE, visual interactive simulator, and because of this it supports this kind of modelling, it also allows the visual representation and animation of the elements of the system making it easier to analyse their behaviour.

WITNESS has 22 pre-defined elements (Thompson 1995) and these can be divided in 2 different types: PHYSICAL and LOGICAL Elements.

1.Physical Elements. (These are displayed as dynamic icons and represent tangible entities in the real life situation being studied): (a)**Parts (*entities*):** Flow through the model (products, telephone calls, etc.); (b)**Buffers (*queues*):** In which parts can be held waiting to be processed; (c)**Machines (***servers, activities***):** Anything that takes parts from somewhere and processes them. Models factors such as: (d)**Labour (***resources***):** This is a resource (for example tools or human operators) which may be required by other elements for processing; (f)**Conveyors**, **Vehicles**, **Tracks**, **Processors**, **Fluids**, **Tanks**, **Pipes**, **Modules**.

2.Logical Elements. (These represent the data and reporting aspects of the model, that is the control of the model.): (a)**Attributes**, **Variables**; (b)**Distributions:** Used to build variability; (c)**Functions:** To build intelligence into the logic of the model, (d)**Time series, Histograms, Pie Charts:** Presents results on the screen as graphs, (e)**Shifts**, **Files**, **Part Files**.

With all these elements in mind we can proceed to the translation process from RADs to the elements of a DES model in particular in WITNESS. This is described in more detail in the next section.

RAD mapping to WITNESS

RAD element →	WITNESS element
ROLES	1 PART
	N LABOURs
ACTIVITY	1 BUFFER
	1 Single MACHINE
THE START OF ALTERNATIVE OR PARALLEL PATHS (CHOICE)	1 BUFFER
(when followed by alterantive paths)	1 Single MACHINE
(when followed by parallel paths)	1 Producction MACHINE
THE END OF PATHS	
(when terminating alterantive paths)	1 BUFFER
	1 Single MACHINE
(when terminating parallel paths)	N BUFFERs
	1 Assembly MACHINE
ALTERNATIVE PATH	As activity
PARALLEL PATH	As activity
START ROLE	As activity
START EXTERNAL ROLE	As activity

Figure 4: Mapping from RADs to WITNESS elements.

Mapping

From the analysis of RADs and DES concepts, we established a basic mapping between the elements of both models. We took the RAD model: roles, activities, decisions and interactions, and mapped them to entities (parts), resources (labours), servers (machines), and queues (buffers). The structure of the mapping that we used is shown in fig. 4. Next we describe briefly some aspects of this mapping relating them to the example in fig. 3.

Parts: The PMW-RAD tool has an attribute for the *Roles* to identify which will trigger the simulation. This attribute is called "*first role*" and its value can be *false* or *true*. When generating the parts for the simulation we take the *role* of the RAD model with the former attribute equal to "*true*", and from this we define the **part** (entity) that will be moving through the system. In the case of the example of fig. 3, the *role* customer is the one that will trigger the simulation and from this we generate a **part** for the request form.

Labours: Also for each *role* of the RAD we generate a number of **labours** which will perform the *activities* of that *role* in the simulation. The particular number of **labours** assigned is given by the user by a very simple user interface. For the example (fig. 3), we generate two **labours** (via the user interface) for the *role* of the clerk.

Machines: There are six different kinds of machines in WITNESS: (a)**Single**: Processes one part at a time; (b)**Production**: Takes in a single part and outputs a number of parts. When, after a simple activity we need to perform two or more different activities, we need to produce one or more copies of the entity; (c)**Assembly**: takes in a number of parts and outputs a single part. When in carrying out an activity we need to wait for the processing of two or more different activities and join the results together; (d)**Batch**, **General**, **Multi-station**.

The type of **machines** that we use in our mapping are **single**, **production** and **assembly**. Basically we generate a **single machine** for all the *activities* on the RAD model, for *alternative paths* and *parallel paths*, *undefined activities* and for *start paths* (the condition for a path to take place) only when they are followed by *alternative paths*.

In the particular case when a *start path* is followed by *parallel paths*, we map the *start path* to a **production machine**, thus the **part** feed into the **production machine** will be multiplied by the number of parallel activities that need to be performed. In PMW-RAD *end path*s are the elements that mark the end of the set of activities carried out within a *start path*. When the *end path* is the terminator for the activities coming from parallel activities, then we map the *end path* to an **assembly machine**. This means that we cannot continue with this activity of the process until all the threads of activities from the parallel paths have been completed. The time taken will be the path that takes the longest to finish its activities. A simple example of this kind of cases is shown in fig. 3, where we can see that, before making a decision about an insurance, the results about the medical and financial checks have to be completed.

Buffers: **are waiting or queueing elements**, therefore we generate a **buffer** for each **single** and **production machine** and **N buffers** for the **assembly machines**, according to the number of parallel activities that have to complete their work before the next *activity* can be carried out. **Buffers** are an important element of simulation, as they allow us to monitor waiting times, queues and so on.

An interesting question is why it is necessary to introduce **buffers** to all the **machines** if some may not need them? We define a **buffer** for each **machine**, because in a generic tool based on RADs, such as ours, it is not possible to know in advance which activities will need them. This is because we do not know which will take more time than others or which will not take any time at all. Nevertheless we need to specify some time for completeness of the model. This redundancy with dummy activities do not cause any problems, as long as their times are very small compared with the relevant activities. We should also remark that in this way we keep the structure of the original model, making it more easy for the domain experts in the organisations to understand the simulation model.

Input and Output Rules: These determine the flow of entities through the system. In the RAD model this is given by the state lines (links) between the *activities* and from *interactions* between *roles*. When we have alternative paths, RADs does not explicitly capture information about the probabilities of the entity flowing to any possible path. Hence, we have to make some assumptions. In this case, we use very simple **rules** to determine the behaviour of the process. We are using an **Output Rules** with a given percentage of possibility of taking any of the possible paths, but we can use any other type of rule or distribution.

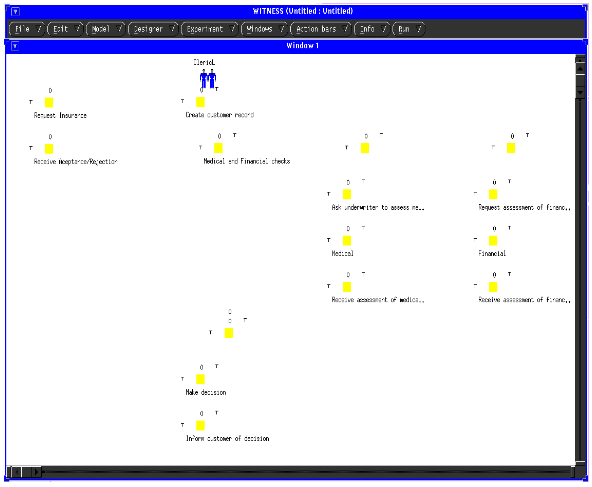

Figure 5: Insurance model in WITNESS keeping the structure from the original process.

Probability Distributions: We are using a theoretical probability distribution: a **Negative Exponential** with different service times to define the **machines' cycle time** and the mean of inter-arrival time for the **parts**. However this can be easily modified to introduce any theoretical or empirical probability distribution.

Mapping example

We mapped the RAD model of the insurance request process on fig. 3 to the corresponding WITNESS representation. By means of the simple user interface we defined two agents (**labours**) that will be responsible for the *role* of Clerk. The visual representation of the WITNESS model is as shown in fig. 5, where we can see how the tool keeps the original structure, making it easier to understand for those familiarised with the original process model represented in RADs. The model is ready to be simulated to gather the information required.

CONCLUSIONS

We have presented an experiment involving the development of a prototype tool for the automatic generation of Discrete Event simulation models from RADs. We expect this to be useful as a guide for other research in this field. Our aim is to continue with the development of this tool, to create a more interactive user interface, to gather the correct information given the model context, thus making results more useful. Finally, we are working on the production of a modified RAD that will contain more dynamic attributes to make the representation more appropriate for simulation and thus subsequent automatic generation will be more straightforward.

Thanks to my supervisor Prof. Brian C. Warboys for his useful advice and comments on this work.

REFERENCES

Curtis, B.; Kellner, M. I. and Over, J., 1992, "Process Modeling", *Communications of the ACM*, 35, no. 9, 75-90.

Gladwin, B. and Tummay, K., 1994, "Modelling Business Processes with Simulation Tools", *Proceedings of the 1994 Winter Simulation Conference*, 114-121, and references within.

Law, A.M. and Kelton, W.D., 1991, *Simulation Modeling and Analysis*, McGraw-Hill, Inc. Second edition.

Lingineni, M.; Caraway, B.; Benjamin, P.C. and Mayer, R.J., 1995, "A Tutorial on ProSim: A Knowledge-Based Simulation Model Design, *Proceedings of the 1995 Winter Simulation Conference*, 408-412.

Miers, D. 1996. "Use Of Tools and Technology Within A BPR Initiative" In *Business Processes Reengineering*, 1996 edited by Coulson-Thomas Colin, Kogan Page Limited, 142-165.

Ould, M. A., 1995, *Business Processes: Modelling and Analysis for Reengineering and Improvement*, John Wiley and Sons Inc.

Pegden, C.D., 1997, "Simulation: A Look to the Future", *11th European Simulation Multiconference (ESM'97)*, K17- K21.

Silver, Bruce Associates, consulted in September 1997, "BPR Tools Add Power to Workflow Automation", *Industry Insider Report URL: http://www.metasoftware.com/whiteppr.html*

Thompson, W.B., 1995, "Introduction to WITNESS and Linking to Process Mapping Tools", *Proceedings of the 1995 Winter Simulation Conference*, 462-466.