Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

DOCUMENTATION OF DISCRETE EVENT SIMULATION MODELS FOR MANUFACTURING SYSTEM LIFE CYCLE SIMULATION

Jan Oscarsson
Matías Urenda Moris

Centre for Intelligent Automation
University of Skövde
PO-BOX 408
S-541 28 Skövde, SWEDEN

ABSTRACT

The concept of life cycle simulation appeals to most production engineers. There is a problem with simulation models of manufacturing systems; they may be highly complex and time consuming to develop. They embrace a considerable experience, which is gained through the development process of the simulation model. It is therefore not enough to develop an accurate simulation model. The model must be understood, updated, re-used and inhered by others. One way to achieve these goals could be through use of a standardised documentation, which in turn explains the model and how it has been developed. This paper presents a method for how simulation models can be documented by standardised notations. The documentation is adapted to different users of the simulation model.

1 INTRODUCTION

The concept of life cycle simulation appeals to most production engineers. A manufacturing system has a life cycle, precisely as a product has its lifecycle. Normally the life cycle of the manufacturing system is much longer than the product life cycle. During its life the manufacturing system will go through a number of changes. The changes can be driven by various reasons:

- Introduction of a new product
- Rationalization of the production
- Changed legal provisions
- …

The manufacturing system is normally not 'new born' each time a change is made. Often, a move from one configuration to another is made by keeping the basic structure of machines and layout intact. Changes are made to accessories such as tools, fixtures and pallets. Other equipment, for instance conveyors and AGV's used for material handling may also be re-designed or changed in accordance with new product or production demands. By regular and careful maintenance a manufacturing system, e.g. a machining line, may have a technical life of 20 years or more. During this life span the system probably will undergo numerous changes, amendments or major re-configurations.

Product designers use CAD models and digital mock-ups as point of departure for work during the product life cycle. Similarly, the production engineer can use a simulation model over the manufacturing system as his starting point for work during the manufacturing system life cycle. This simulation model can be a 'digital factory', matching the 'digital mock-up'.

Simulation models of a manufacturing system can be made using different simulation technologies. This paper is limited to discrete event simulation technologies, mainly used for material flow analysis. There are two basic approaches to build such a model, either to use a simulation language or a simulation package. The two basic approaches can be categorized as (Randell 1999):

- **Languages** e.g. SLAM, GPSS/H, SIMAN. These are high-level languages that offer the programmer more flexibility and a more powerful language than the simulation packages can give. However this approach is much more time consuming than using a simulation package.
- **Simulators** e.g. Witness, ProModel, TaylorII. These are data driven systems with little or no programming required. This approach is fast and easy but more limited in application.

There is a third approach that is derived from these two:

- **Hybrid systems** e.g. Arena and QUEST which combines the flexibility of a simulation language i.e. SIMAN and SCL respectively, with the user-friendliness of a data driven system. The aim is to

exploit the speed of the simulator package and still have the flexibility of the simulation language.

This paper is concerned mainly with simulation packages of the hybrid type.

2 BACKGROUND

One particular aspect on simulation of a manufacturing system is that it is not continuously ongoing; it is done only when required. Compared to the product life cycle the manufacturing system life cycle is longer and normally changes are made less frequently. The system may be in operation for a year's time with no changes being made at all. Another aspect on life cycle simulation is the change of staff. The people working with the manufacturing system are continuously exchanged. People moves to new positions, retires etc. This means that the knowledge about the manufacturing system continuously needs to be up-dated, including the knowledge about the simulation model.

There is a problem with simulation models of manufacturing systems; they may be highly complex and time consuming to develop. They embrace a considerable experience, which is gained through the development process of the simulation model. It is therefore not enough to develop an accurate simulation model. The model must be understood, updated, re-used and inhered by others. One way to achieve these goals could be through use of a standardised documentation, which in turn explains the model and how it has been developed. It may also give careful notes about the syntax used and the model structure. The task of documenting systems or processes, e.g. data systems, businesses process, etc. has been subject to much work in order to develop notations providing a common language to describe such systems.

Today several notations are used in different engineering applications e.g. Unified Modelling Language (UML), Integration DEFinition language 0 (IDEF0) and Jackson Structured Programming (JSP). Some of these notations are specific for a special task and have been accepted in their respective fields, UML in software development, IDEF0 in business processes and JSP in procedural programming. On the other hand, simulation models rarely are documented by such recognised and accepted notations. There are a lot of suggestions regarding the importance of documenting the development process for simulation models, but only few suggestions about how to document the model itself (Richter and März 2000, Banks, Carson and Nelson 1996) The lack of a more uniform and accepted way of documenting simulation models leads to poor or scare documentation of the logic and structure of the model. In the software development area many systems are very difficult to change because of poor documentation. Often, simulation models suffer from the same kind of problems. This makes the re-use of experience and of the model itself very difficult.

Discrete-event simulation procedures are different to software development in the sense that the developer does not necessarily need to write his own code, instead may predefined functions and features provided inside the simulation package be used. Even if the user can write user defined functions, the structure and platform of the software is predefined, particularly for hybrid simulation systems.

3 CRITERIA

A suitable notation for documentation has to be defined if a simulation model shall be documented in the perspective of life cycle simulation. This is of major importance since documentation is not for free. According to (Pressman 1992) 20 to 30 percent of the cost in software development are spent on documentation. This could be one reason for why there is a difference between how textbooks treat the need for documentation and how it is done in reality. In many cases documentation is done sparsely.

Despite the cost and the inconvenience, which some may argue that documentation cause, it is essential for the use, credibility and re-use of the model. Not having enough documentation of a simulation study may be more costly and cause more inconvenience than the opposite. This paper does not aim to consider all the documentation proposed to a simulation study. The aim is to concentrate on the documentation that is necessary to document the model itself, and not the entire project documentation.

Lehman (Lehman 1977) expresses the importance of directing program documentation in a research simulation study to different audiences. He identifies three audiences:

- The programmer itself.
- The researcher interested in the program and its model.
- The reader with no desire or need to understand the inner detail of the program.

These three audiences can easily be mapped to a non-research simulation study with the equivalence of:

- The programmer, or the original simulationist.
- The new engineer/simulationist who is going to either study the model or re-use it
- Others who want to understand the main frame, but not necessary all details in the model.

Lehman also differentiates between internal documentation within the programme and external documentation in a separate document. The internal documentation refers to explicit code documentation, in the heading of each procedure/routine and in the code itself giving comments. The programmer should follow the general guidelines/rules given in programming conventions (Eklund, Mellin and Brimark 1998). This documentation is understood to be of

value for the first and second audience, but Lehman suggests that a detailed specification of the conceptual model is even of greater value for the second audience. Finally he makes the point that the third audience should get some kind of user manual, a non-technical external description.

It is clear that simulation model documentation has to be developed for all different types of audience. There is also a need for documentation at different levels of abstraction, offering a way for communication and learning among a project group. There is a need for describing complex models from different views, providing a holistic view of the model.

3.1 Criterions to Consider in the Selection of Notations

From literature (Eklund, Mellin and Brimark 1998, NIST 1993) and experiences made in industry a number of criterions to be fulfilled by a notation have been identified. The following criterions have been identified:

- **Neutral notation**. The need for a neutral notation, not limited to any specific languages, software or systems. The main reason for this is that there is no language or software, which is a standard in simulation engineering, and there will probably never be. Because of this, the notation(s) should be able to support the development in e.g. QUEST, Arena or a simulation language. A neutral notation offers a possibility to e.g. first document a QUEST model and at the same time translate that documentation into an Arena model for the purpose of re-use.
- **Generic notation.** The need for a generic notation that can describe different systems of various purpose, complexity and scope. A simulation department may work with different types of simulation. The simulation task may be simulation of a production system, ergonomics in an assembly line or a robot cell. A generic notation would support the standardisation in documentation procedures.
- **A recognised notation**. The importance of using a known and well-recognised notation is beneficial. If there are any interpretation difficulties because of the use of an "in-house" solution, the notation may need an additional documentation in order to be understood and that is not the point. Using a notation that to some degree is considered as a standard or at least well recognised will improve the communication and support the model maintenance during its life cycle. It is as talking the same language, it helps when working together!
- **User friendly**. Notations help the reader to overcome the differences between the abstract code level and the natural language (Eklund, Mellin and Brimark 1998). To fulfil that purpose they need to visualise what is abstract, in other words be intuitive and descriptive. There is always a balance between the descriptive and intuitive approach and the extendibility of the notation (Sinan 1998). If the notation is too simple and intuitive it may lack the possibility to describe the simulation model in detail.
- **Descriptive in several levels.** A notation with the possibility to modulate and describe different levels of abstraction gives the user the possibility to study the system from a top-down or bottom-up point of view. The user may choose to use a top-level view in order to describe the system for an audience of category three, or a bottom-level for a co-programmer.
- **An in-house competence.** The documentation has, mostly, as a target customer the client's organisation. This means that the client should be included in the selection of notation. Some companies have already standards they use in documentation. In these standards one company might have chosen to use the complete IDEF family, another UML etc.. The consultant should not be surprised if he is the one with the need to adapt. In that case he should remember that the successful use, credibility and re-use of the model might lie in this adaptation.

These six criterions should guide the selection of notation, they are probably not the only ones, but these are among the most important ones and could serve as a base in future consideration of notations for simulation studies.

4 MODEL DOCUMENTATION NOTATIONS

A survey over a number of potential notations used in different areas has been made. For each of the notations an evaluation has been made regarding the criterions identified. The sixth criterion, an in house competence, cannot be evaluated for obvious reasons. It is up to each individual company or organisation to find out their competence. Nevertheless this is something, which at the end of the day may be of a much higher priority than all other criterions.

For documentation of discrete event simulation models for manufacturing industry the following recommendations is made:

- For low-level documentation intended for the programmer use flowcharts and comments in the code. If there is a need to be more precise in the description of the structure and behaviour of the system use UML diagrams. Concentrate on structure and behavioural diagrams.

- For conceptual documentation, use IDEF0 to describe the flow in the system. Decompose to a more detailed level when it is necessary to describe the system flow more accurately. Combine the IDEF0 models with Flowcharts to visualise a more physical structure of the system.
- When describing the system for people with no interest in details, choose the simulation. An animation of the simulation, perhaps in 3D with fairly realistic graphics is by far the clearest documentation of system behaviour. It supports communication in a project group, and thanks to its dynamic behaviour illustrates more than many diagrams can do together.

4.1 IDEF0

IDEF (Icam DEFinition) was developed within the US Air Force program for Integrated Computer Aided Manufacturing (ICAM) during the early 80s. Originally the IDEF package contained three different techniques for modelling (NIST 1993):

1. IDEF0, used to produce a "function model".
2. IDEF1, used to produce an "information model".
3. IDEF2, used to produce a "dynamic model".

Today several other modelling techniques have been implemented in the IDEF family, but the consideration that follows will focus on IDEF0.

IDEF0 (Integration DEFinition language 0) is a method designed to model the decisions, actions, and activities of a system or organisation. The methodology focus on describing a system in terms of their activities and information flows instead of its (data) structure and by doing so it is different to the JSP notation. IDEF0 is based on SADT (structured Analysis and Design Technique) developed for software engineering projects.

The methodology combines both graphics and text, and presents them in an organised and systematic way making it easy to understand and use. It can be used in both top-down and bottom-up approaches and one of its major advantages is the built-in decomposition methodology it possesses. Each activity or function can be decomposed and be easily tracked without loosing the overall picture. This gives IDEF0 the ability of reaching the level of detail necessary for the analysis, and at the same time give the project team the possibility to choose, in there communication, the adequate level of abstraction increasing the consensus and understanding among different project members.

- **Neutral notation.** IDEF0 is neutral, but it does not give much support in programming tasks. The notation syntax gives the user a possibility to decompose activities to very small fractions, but it is still not suitable for describing simulation programme logic.
- **Generic notation**. IDEF0 is a generic notation for system analysis. The notation is good at representing activities and processes in a structured way. It is ideal to describe the flows in a simulation model.
- **A recognised notation.** IDEF0 is well recognised among people working with engineering. Others working with software engineering background would probably recognise it if the name of SADT were used since it is the same methodology. This supports the communication between different groups, which is an advantage when presenting a simulation model.
- **User friendly**. IDEF0 is a logical and descriptive notation that does not require too much work to understand and use. The semantic and syntax are simple and the textual descriptions adds user friendliness to the notation. Some difficulties can be experienced when the user have to decide a flow to either be an input, mechanism or control flow. Another difficulty is to find suitable names to the more abstract activities. But looking at the whole picture, IDEF0 is user friendly.
- **Descriptive in several levels.** This is one of the strongest features of IDEF0. The notation gives the user the opportunity to either decomposed or abstract activities. The user can choose to either follow a bottom-up or a top-down methodology. And he can choose to describe parts of the model flows at a conceptual level, and others at a more detail level. This flexibility is of great value.

IDEF0 is well suited for documenting the flows in a simulation model. It does not cover all the aspects of the model documentation and do not say much about low-level logic, structure or dynamics. It cannot handle parallel processes well, but it plays an important role in giving an overall understanding of the simulation model. The quality of being comprehensive and broadly used among engineers makes the IDEF0 model documentation a good choice.

4.2 UML

UML stands for Unified Modelling Language and it is a "modelling language for specifying, visualizing, constructing and documenting the artefacts of a system-intensive process." (Sinan 1998) This means that UML can be used in software engineering, in business modelling and in others non-software systems. UML was developed by Rational Software Corporation and three methodologists in the information system and technology industry, Grady Booch, James Rumbaugh and Ivar Jacobson. The language have gained significant industry support from various or-

ganisations via the UML Partners Consortium and has been submitted to and approved by the Object Management Group (OMG) as a standard.

Compared IDEF0, UML includes a wide range of diagrams and complementary notations based on the same metamodel. The metamodel is based on the object-oriented paradigm and is divided in three different packages, the Foundation package, the Behavioural Elements package and the Model Management package. All these packages are related, but within each package the elements have a stronger relation to each other's than to other packages. The Metamodel elements are organised into diagrams. Different diagrams are used for different purposes depending on the angle from which the system is viewed, see figure 1.

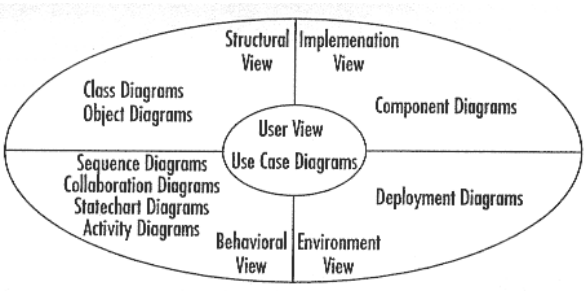

Figure 1: Model View and Diagrams (Sinan 98)

- **Neutral notation.** UML is considered to be a language, with a syntax that is neutral and can be used to design and document any kind of programmes. It is based on the object–oriented paradigm, which makes it very suitable to document or even design simulation models. The UML notation supports all the features in QUEST's logic. This means that it is up to the producer of the model to decide how much to document, the notation places no limits.
- **Generic notation**. UML is a generic language. It is used for specifying, designing and documenting software systems, as well as for business modelling and other non-software systems. From a simulation modelling point of view it covers more of the software documenting and constructing aspects of the model, but not the process flow as well as IDEF0.
- **A recognised notation**. It is a relatively new notation even if much of its contents come from older notations. In the software engineering arena it is well recognised. Engineers on the other hand normally have herd of the language, but they probably do not understand it and much less us it.
- **User friendly.** UML feels abstract and difficult to embrace at first sight. With use and practice this surely changes, but the question is if people adopting the model will have the knowledge- or interest in learning UML. Perhaps the UML model will be more difficult to understand than the simulation model itself.
- **Descriptive in several levels**. UML can describe systems at many different levels, from very high-level description to low-level diagrams. Case tools can even generate program code directly from UML. But it has not the flexible ability experienced in IDEF0, and it lacks the same visual strengths.

UML surely can describe the structure and behaviour of a simulation model. The question is if it would add any value to the documentation? It would, if the personnel working with the model have UML experience, otherwise it would probably not. There is always the possibility to choose some of the diagrams in UML to describe smaller parts of the model e.g. class diagrams to visualise the structure of elements and their associations, or sequence diagrams to give a more accurate description of the element behaviour. But unless the model is of the magnitude that it is completely impossible to embrace it, UML feels unnecessary complex to add value to the documentation.

4.3 Flow Charts

A flow chart is defined as a pictorial representation describing a process being studied or even used to plan stages of a project. This pictorial attribute facilitates the communication among people. Despite peoples different backgrounds they tend to understand the symbols and can easily visualise the main features of the process. But even if it is a widely used modelling technique it is just as often abused. The lack of use of formal procedures sometimes makes the interpretation ambiguous or misleading. (Pressman 1992)

Flow charts are easy to use and customised to different processes. Several software products makes a whole set of symbols available, more or less descriptive for visualisation and analysis of a process. Flow charts exist for different disciplines; ranging from flow charts describing programmes to flow chart used to describe business processes at high abstract level. Despites the different scenarios flowcharts are based on some basic constructs, plus some added constructs intended for special areas.

- **Neutral notation.** Flow charts are extremely neutral. Their basic constructs can be used without consideration of the programming language. The notation is widely used to document programmes.
- **Generic notation.** Flow charts are very generic. They are used in the most different imaginable situations, and can be used to describe all kinds of manufacturing systems.
- **A recognised notation.** Flow charts are well recognised. The user may not always recognise all the symbols, but they usually understand the

meaning from the context of the diagram. In engineering contexts flow charts have been used for decades, in operation descriptions and system analysis. This is a major advantage when a simulation model is described, people of different positions will understand the diagram and is able to contribute to the overall discussion.

- **User friendly**. Mostly flowcharts are easy to understand. When they become too large they lose a bit of its intuitive communication, and become more complicated to track and understand. But to describe material flows, or logic in not to large diagrams, they fill a clear descriptive purpose. They are more intuitive than IDEF0 diagrams, but lack the same flexibility in decomposition that IDEF has. A possible problem could be that they can cause ambiguous interpretations when no precise syntax has been applied.
- **Descriptive in several levels.** Flow charts can be used at different levels. They also share the decomposition characteristic that IDEF0 posses, but not to the same magnitude and not with the same documentation methodology.

Flow charts are well suited to support the documentation of a simulation model. They can be used at different levels and are well understood by different categories of people. Their strengths lie in their descriptive and intuitive syntax. Their weakness is that they can be difficult to track in large diagrams with a lot of branches.

5 CONCLUSIONS

An evaluation is always coloured by the authors experience and expectations. Especially when the evaluation is made on subjective matters and not mainly facts e.g. when people evaluate the performance of a car. The interpretation of what is a good notation for simulation is based on some fundamental guidelines, experience and personal intuition. Being totally objective is impossible even when effort is put to it.

Documentation deals with storage of experience and with communicating or making this experience accessible to others. This means that it is not always the technically perfect language that is the best medium for documentation. The language people understand and use is probably the best. If we choose to write all reports in Latin, the experience and knowledge they contain would be lost, or at least obscure for a majority of people. Documentation has to reach people, and hopefully help them to act according to what they may have learned.

From the point of view of documenting a simulation model, we want people to understand and believe in the models credibility. Good documentation will do that job, long after the consultant had a convincing presentation and the first newborn interest in simulation cooled. It will make all the experience embedded in the model accessible and the model ready to re-use.

The proposed method for documentation of simulation models for manufacturing system simulation has been evaluated in a number of case studies. This far it has proven to work, providing a mean to communicate and store knowledge over the manufacturing system life cycle.

REFERENCES

Banks J, Carson J S and Nelson B L 1996 *Discrete Event System Simulation*, 2nd Ed. Prentice-Hall, pp 3-18

Eklund A, Mellin J and Brimark B 1998 Programmeringsmetodik 1- föreläsningsmaterial, University of Skövde, Sweden, pp 5-107. In swedish

Lehman R S 1977 *Computer Simulation and Modelling,* Lawrence Erlbaum Associates, Inc., pp 215-223

NIST- National Institute of Standards and Technology 1993 "Integration definition for function modeling (IDEF0)", Processing standard publication 183, Federal Information Processing Standards Publication, pp ii- ix, 1-115

Pressman R S 1992 *Software Engineering: Practitioner's approach, European 3rd* Ed McGraw Hill, pp 277-286, 344-349, 732

Randell L 1999 *Discrete-Event Simulation Workshop*, Revision: 1.8, The Lund University, Sweden, pp 1-47

Richter H and März L 2000 Toward a standard process: The use of UML for designing simulation models. In *Proceedings of the 2000 Winter Simulation Conference*, Ed. J. A. Joines, R. R. Barton, K. Kang, P. A. Fishwick, pp 394-398

Sinan S A 1998 *UML in a Nutshell*, O'reilly & Associates, Inc., pp 1-126

AUTHOR BIOGRAPHIES

JAN OSCARSSON is a senior lecturer at the Centre for Intelligent Automation at the University of Skövde, Sweden. He received his Ph.D. 2000 at De Montfort University, Leicester, UK. Currently he is the project manager of the dAISy project, in which the current paper have been prepared. His main research interests includes integrated product development, the 'digital factory' concept and issues related to systems engineering. His email and web addresses are <jan.oscarsson@ite.his.se> and <http://www.ite.his.se/~jan/>.

MATÍAS URENDA MORIS is a Ph.D.-student at the Centre for Intelligent Automation at the University of Skövde, Sweden. He received his MSc in Automation Engineering 2001 at the University of Skövde in collaboration with Loughborough University, UK. His main research area is DES for manufacturing, with an emphasis on integration of simulation and logistics in a demand – supply chain perspective.