ISSN 1012-277X S.Afr.j.ind.eng.



S A Tydskrif vir Bedryfsingenieurswese, Vol 8, No 2, Desember 1994, pp25-36

# AN OBJECT ORIENTED APPROACH TOWARDS THE SPECIFICATION OF SIMULATION MODELS

Antonie van Rensburg
Department of Industrial Engineering
University of Pretoria

## ABSTRACT

To manage problems, is to try and cope with a flux of interacting events and ideas which unrolls through time - with the manager trying to improve situations seen as problematical, or at least as less than perfect. The ability of managing or solving these problems depends on the skills of the problem solver to analyse problems. This article introduces and discusses a proposed methodology for analysing real world problems in order to construct valid simulation models.

## OPSOMMING

Bestuurders probeer probleemsituasies bestuur, of verbeter deur 'n vloed van dinamiese interaktiewe gebeurtenisses te verstaan en te hanteer. Die sukses van die bestuur of oplos van dié probleme hang af van die kundigheid van die probleemoplosser om die probleme te kan analiseer. Die artikel bespreek 'n voorgestelde benadering tot die analise van probleme om sodoende daaruit, simulasiemodelle te kan opstel.

# 1. INTRODUCTION

Simulation modelling is applied to real world problems in the hope of either getting a clearer view of the problem or finding a solution to the problem. The real benefit of a simulation exercise is realised if the analyst succeeds in capturing descriptive real world relationships. Thus, the success of a simulation model is achieved if the problem's basic building blocks are identified and defined for use in the model specification.

Unfortunately, often, no methodology is used to capture meaningful relationships, or if a methodology is used, the fact that real world problems are "messy" is ignored. This means that a suitable simulation methodology should take in account that the modelling process includes rules or techniques that can cope and understand "messy" problems and thus bridge the gap between the real world problem and the problem specification.

# 2. THE REAL WORLD AND SYSTEMS THINKING

Scientists have discovered that in many natural phenomena, which an observer will describe as chaotic and random, a pattern exists which can be described by some kind of mathematical formula [7]. Many approaches or methodologies try to structure these chaotic real world problems in such a manner that certain techniques can be used to describe and solve the problem. It is argued that if these seemingly structured patterns exist in our environment they should be inherent to most real world problems. The result of this argument may be that many techniques found in current methodologies exhibit similar characteristics, as they try to describe the same patterns in problems. Studies have shown that these concepts have their roots in the systems thinking environment.

The definition of a system depends on the motives and values of those involved in describing the system. Difficulties will arise from the conflict of values of perception when one is dealing with a system that describes characteristics which are fuzzy-edged, messy and undefined. Where the system concerned is clearly defined and bounded, difficulties will be "hard" difficulties concerned with facts and techniques that can be dealt with by concentrating on structured and hard methodologies that say "we know what the problem is and we can establish a rationally based solution decision on that solution" [7]. The problem solver has to realise that the real world is a complex place, somewhat like a fractal, in the sense that the more closely you look, the more complexity you will see.

The human brain digests complex problems by building mental models which abstracts those features required for problem understanding by including carefully selected information and ignoring irrelevant features [14]. Thus, a mental model is a simplified view of how something works - this reasoning is necessary and crucial of our understanding of the world [16].

The systemic concept is important in the nature of systems thinking, as it refers to the system as a whole. This basic idea of systems thinking means that a complex whole may have properties which refer to the whole, but will be meaningless in terms of the parts that forms the whole. These emergent properties imply and define a view of reality as existing in layers in a system hierarchy, with the addition of communication and control activities, to complete the idea of a system [4].

## 3. OBJECT ORIENTED PRINCIPLES

Dijkstra suggests that the technique of mastering complexity has been known since ancient times "divide et impera (divide and rule)". This implies that a complex system is hierarchy dependent on small refined parts [6]. Taking in account the way the human brain solves problems, it seems that we only need to comprehend a few components of a system to understand the system complexity [14]. Currently, two types of decompositions exist to order chaos, the algorithmic decomposition and the object-oriented decomposition. The algorithmic decomposition decomposes each module in a system as part of a major step in the overall process, whereas object-oriented decomposition views the system as the environment of objects and observed behaviours. Algorithmic decomposition highlights the ordering of events while the object-oriented view emphasizes agents that either cause action or are being act upon themselves. Grady Brooch [2], describes the five attributes of a complex system as:

i) Hierarchy
Often elementary components form subsystems, which in turn form part of an interrelated hierarchy of subsystems comprising the complex system.

ii) Components
Systems can be divided into some kind of components, with the choice of these components, primarily arbitrary to the observer.

iii) Inter and intracomponent linkages
Inter and intracomponent linkages exist within and between components respectively. Intracomponent linkages involve the internal structure of the components and intercomponent linkages the interaction between components, separating the high frequency and low frequency dynamics of the system.

iv) Sub-systems
A few subsystems describe a hierarchic system in various combinations and arrangements.

v) Evolution
It is usually found that a complex system which evolved from a simple system works. According to Anne Arbor [1], a complex design from scratch never works and cannot be patched up to make it work.

Object oriented approaches model systems from an object view rather than a functional viewpoint. The object viewpoint can be defined as the abstraction of "things" in the problem domain that we want to keep information about (attributes), and interact with (services) [9]. The more important characteristics exhibited by any object oriented approach is *abstraction*, *encapsulation*, *inheritance* and *hierarchy*. Abstraction is achieved by the conceptual boundary that makes objects distinguishable, while encapsulation is the hiding of unneeded object detail. Inheritance acknowledges the fact that objects can inherit object relationships which in turn forms hierarchical structures [3,4].

The concept of human motivation (anthropomorphism) describes the object characteristics such as encapsulation and object responsibilities. With this "human like" approach the procedure that an object will carry out, is defined as its services and the guarantee that the services will be carried out becomes a contract of the object. Thus the contract involves an agreement between objects whereby a service provider promises to deliver the expected results (collaborations) if the service requester makes a request in prearranged ways [5].

## OBJECT ORIENTED METHODOLOGY

The following steps propose the iterative process that should be followed to construct a valid simulation specification, based on the concepts found in the object oriented analysis and design methodologies:

i) Problem Scope
ii) System Definition
   - General system description
   - Main system components
   - System boundaries
iii) Superclass and Class definitions
iv) System and Class hierarchy graphs
v) Object Design
   - Responsibilities
   - Information Encapsulation
   - Collaborations
vi) Communication Protocol Graphs

The application of the methodology is demonstrated by means of an example that features the layout of a generic manufacturing plant (Figure 1). The plant process workpieces (which can either be of part type 1 or part type 2), at two workstations, the *process workstation*, and the *inspection workstation*. Workpieces arrive at the *process waiting area*, waiting for the *process workstation* to become available. After process completion at the *process workstation*, the workpieces move to the *inspection waiting area* before being inspected at the *inspection workstation*. At the *inspection workstation* workpieces are inspected and classified into three categories, good, repairable or bad. Workpieces marked as good will leave the system after inspection, while repairable workpieces cycle back to the *process workstation*. The workpieces marked as bad are sent to salvage. The manufacturing plant has to be simulated and the following recorded:

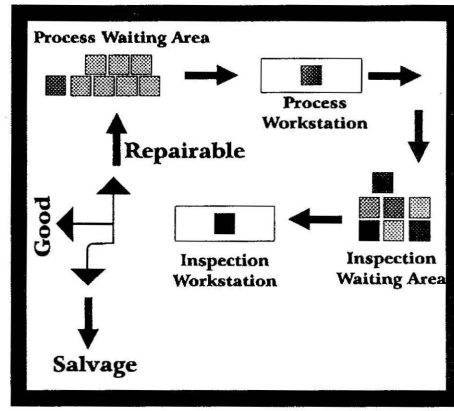


**Figure 1 Manufacturing System**

i) The number of completed workpieces (good).
ii) The number of reworked workpieces (repairable).
iii) The number of rejected workpieces (bad).

## 4.1 Problem Scope

The *Problem Scope* defines a certain view of the problem that aims to capture the essence of the problem objective. This forms the basis of the language in which the problem is seen and described by the problem solving team, taking in account multiple human perceptions from team members [14]. For this example the problem objective will be to simulate the manufacturing system with the following in mind:

- Record the number of workpieces completed (good, repairable, bad).

## 4.2 System Definition

"The essence of the modelling art is abstraction and simplification" [10]. The system definition phase tries to identify system features that will be sufficient to obtain the objectives (*Problem Scope*) of the study. According to Pegden, et al [10], this process entails itemizing system components that contribute to the effectiveness or ineffectiveness of the system operations - although the determination of the usefulness

or significance of each component may be difficult at this stage of the model development. However, the analyst must remember that an effective model should neither oversimplify the system to the point where it becomes trivial nor carry so much detail that it becomes clumsy and expensive to develop. The system definition of the problem is a general system description that references system boundaries, system components and the system interfaces involved.

**General System Description**

The general system description is a conceptual overview describing the real world problem from the *Problem Scope* view. The aim of this is to ensure that all the relevant parties involved, understand the nature of the problem in an unambiguous manner, consolidating the different perceptions involved. The general system description guides the problem solver to the identification of system components. For this example the general system description will read as follows, "*the manufacturing process includes two workstations with workpieces processed at the process workstation and inspected at the inspection workstation. The inspected workpieces can either leave the system as good, bad or rejected workpieces, with the rejected workpieces returned to the process workstation for rework.*"

**System Components**

From the above description the following components are identified:

| | |
|---|---|
| Workpieces | Process workstation |
| Inspection workstation | Waiting areas |

**System Interfaces**

Within the given problem description and scope, the following components interface with the "outside world":

| | |
|---|---|
| Process Waiting Area: | Receive workpieces from the "outside world". |
| Inspection Workstation: | Workpieces marked as good leaves the system to the "outside world". |

The "*System Interfaces*" phase, identifies relationships between the defined problem space and the outside environment it communicates with. The knowledge of these relationships is helpful when objects and object relationships are defined.

## 4.3 Superclass and Class Definitions

In the object oriented environment a class is a generic specification for a number of objects that share the same behaviour. They provide the analyst with the means to describe in one place the generic behaviour of a set of objects, which is then used to create other objects in the system. The concept of a class and an object is tightly interwoven because of the fact that an object cannot be referenced without regard for

its class [15,16]. Booch [2] explains a class and an object as follows:

*"A Class is a set of objects that share a common structure and a common behaviour. A single object is simply an instance of the class. An object is not a class, although a class may be an object. Whereas an individual object is a concrete entity that performs some role in the overall system, the class captures the structure and behaviour common to all related objects. Thus, a class serves as sort of binding contract between an abstraction and all of its clients."*

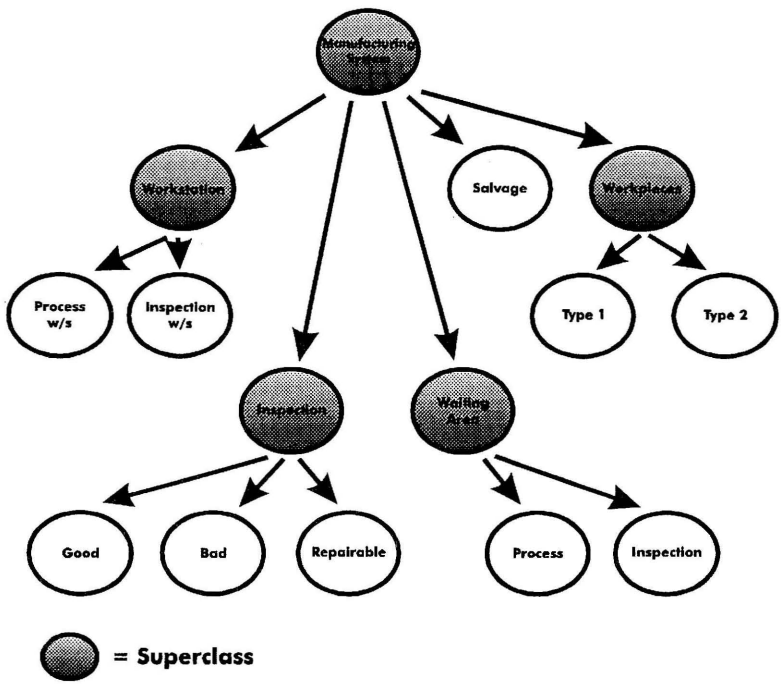


**Figure 2 System Hierarchy Graph**

A superclass is a class from which classes inherit specific behaviours (see Tabel 1 and Figure 2). A class in the object model may have only one superclass or it may have several - combining behaviour from several sources to produce a unique kind of an object [7]. The superclasses and classes are ranked, or ordered into a hierarchy to display the inheritance structures of the problem (Usually hierarchy graphs are used to explain the abstract structure of the problem).

| SUPERCLASS | CLASS |
|---|---|
| Workpieces | *Type 1 workpiece*<br>*Type 2 workpiece* |
| Workstation | *Process Workstation*<br>*Inspection Workstation* |
| Inspection Check | *Good*<br>*Bad*<br>*Repairable* |
| Waiting Area | *Process Waiting Area*<br>*Inspection Waiting Area* |

**Table 1 Superclass and Class classification Descriptions**

## 4.4 Analysis and Design of Objects

During the analysis and design phase of the system objects, the "essence" of every object is recorded in terms of its place in the problem hierarchy, the services and contracts that it delivers and the relationships with other objects. Table 2 displays the proposed format used to describe objects in the system.

| **PUBLIC RESPONSIBILITIES** | |
|---|---|
| **Class Name :** Process Workstation<br>**Part of Superclass:** Workstation<br>**Public Interface with other objects:** Process Waiting<br>Inspection Waiting | |
| **INFORMATION ENCAPSULATION** | |
| **Static Attributes:**<br>Single Part Processing | **Dynamic Attributes:**<br>Processing times<br>Mean / Standard dev.<br>Type 1 5 / 2<br>Type 2 4 / 1 |
| **COLLABORATIONS** | |
| **Contracts:** Process workpieces<br>Client: *Inspection workstation*<br>Server: *Process Waiting Area* | |
| **External Messages Received:**<br>Message: *Receive parts*<br>Server: *Process Waiting Area*<br>Information Protocol: *Part types* | **External Messages Send:**<br>Message: *Workpieces completed*<br>Client: *Inspection Waiting Area*<br>Information Protocol: *Part types* |
| **Object Responses on contract:**<br>The workstation receives the different types of workpieces, which can either be of Type 1 or of Type 2. These workpieces are processed according to the status of the workpiece. | |

**Table 2 Object Table**

Explanatory Notes:

1. Public Interface
   *Public interface* identifies those objects that will have contact with the object *Process Workstation* during the modelling of the problem.

2. Static attributes
   Static attributes describe static characteristics of the object. In this example the process workstation is only able to process parts in a single queue.

3. Dynamic attributes
   Dynamic attributes are those object characteristics that may change in the course of modelling the system - the processing times of the Process workstation varies according to the different workpieces it receives.

4. External Messages received and External Messages Send
   During the object lifecycle, messages are received and send by the object - *External Messages Received* records all possible messages that the object receives and *External Messages Send* records those that are send by the object to other objects.

6. Information Protocol
   Describes the information carried by the object (the message).

7. Object Responses
   The reaction or procedure of actions that takes place within the object when the contract is carried out.

## 4.5 Communication Protocol Graphs

A Communication Protocol Graph displays object relationships and message paths between the objects to obtain a structure that will either be used for coding or experimental purposes. One of the inherent difficulties with semantics employed by current methodologies (to display object relationships) is the ability to represent relationships dynamically [13].

During 1962 Carl Petri developed a technique called Petri Nets which had the original purpose of specifying system information flows in a descriptive manner. The technique has since been adapted to suit the design of simulation models as JL Petersen [11] explains "*The simplicity and power of Petri nets make them excellent tools for working with asynchronous concurrent systems*". A Petri net graph models the static properties of a system much in the same way as a flowchart represents the static properties of a computer program, but in addition to that incorporates the dynamic responses by means of the position and movement of markers in the graph.

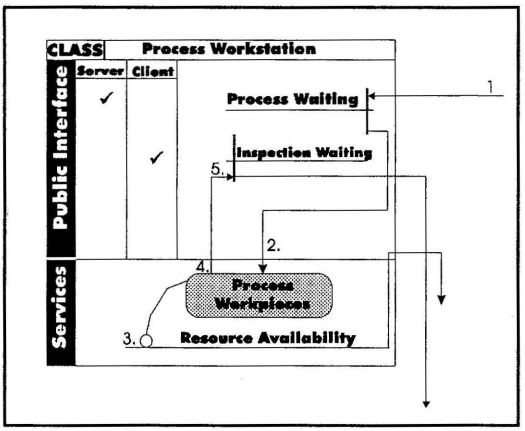


**Figure 3 Communication Protocol Graph**

**Example**

The following example explains Figure 3 which represents the object, *Process Workstation*.

1. The object *Process Workstation* receives the message "Receive Parts" (Table 2).
2. The rule is fired and the contract *Process Workpieces* is executed.
3. The inhibitor arc only fires if the contract *Process Workpieces* is not busy. When the contract is finished, the rule starts firing again. In this example it means that a signal is send to the object *Process Waiting Area* indicating that the process workstation is idle and ready to process another workpiece.
4. After completion of the contract the rule is fired and the workpiece is ready to move to the following object.
5. No conditions are set to be met by the processed part and the entity moves on to the client: Inspection Waiting Area.

The reader is referred to the article "Petri Nets" by JL Petersen [11] for a detailed explanation on Petri Net rules and structures.

## 5. CONCLUSION

Solving real world problems, which includes the formal definition of the problem and its boundaries, tends to be a complex problem. It is even more difficult for the simulation analyst as simulation models normally aims to represent real world problems with mathematical models. In the search for a methodology that can be used to define specifications for simulation models, experiments with current methodologies and techniques (such as process flows, entity relationship diagrams, etc.) proved unsuccessful. This is attributed to the fact that there is a difference between the design of simulation models and those models developed in the information engineering field.

This article suggests that when one seeks an applicable methodology or technique to specify simulation models, the organization of natural systems should be used as the basis to establish a worthwhile and applicable simulation modelling methodology. Research shows that systems thinking and object methods prove to be successful in capturing problem structures, while information engineering has some successes with the simulation model information flows. Thus, it seems that the combination of systems thinking principles, information engineering and object oriented analysis and design techniques could prove to be successful in aiding the design of a simulation methodology.

## 6. REFERENCES


1. Arbor, Anne, "*Systemantics: How systems really work and how the fail*", 2nd ed. MI: The General Systematics Press, p965.
2. Booch Grady, "*Object Oriented Design with Applications*", The Benjamin/Cummings Publishing Company Inc. 1991.
3. Carver DL, "*Promoting the use of an object-oriented software development methodology by merging structured and object-oriented analysis methods*", Proceedings of the IEEE Souteastcon '92, IEEE, 1992, p593-599.
4. Checkland P, ScholesWiley J, "*Soft Systems Methodology in Action*", John Wiley, 1992.
5. Cockburn AAR, "*The Impact of object-orientation on application development*", IBM Systems Journal, Vol 32, no 3, 1993, pp420-443.
6. Dijkstra E, "*Programming Considered as a Human Activity*", Classics in Software Engineering, Yourdon Press.
7. Harry M, "*Information and Management Systems: Concepts and Applications*", Pitman Publishing, 1990.
8. Monarchi DE and Puhr I, "*A Research Typology for Object-Oriented Analysis and Design*", Communications of the ACM, September 1992, Vol 35, No 9.
9. Norman RJ, "*Object-Oriented Systems Analysis: A methodology For The 1990s*", Journal of Systems Management, July 1991, p32-40.
10. Pegden CD, Shannon RE, Sadowski RP, "*Introduction to Simulation Using SIMAN*", Mcraw-Hill, 1990.
11. Peterson JL, "*Petri Nets*", ACM Computing Surveys, Vol 9, No 3, September

1977, pp 223-252.

12. Thesen A, Travis L E, "*Introduction to Simulation*", Proceedings of the 1991 Winter Simulation Conference, p5-14.
13. Törn AA, "*Simulation Graphs: A General Tool for Modeling Simulation Designs*", Simulation, December 1981, p187-194.
14. Shlaer S and Mellor SJ, "*Object-Oriented System Analysis: Modelling the World in Data*", Yourdon Press, 1998.
15. Wilson, B, "*Systems: Concepts, Methodologies and Applications*", John Wiley.
16. Wirfs-Brock, Wilkerson B, Wiener B, "*Designing Object-oriented Software*", Prentice-Hall, 1990.