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

S A Journal of Industrial Engineering, Vol 7, No 2, December 1993, pp.36-46

The design of a simulation generator

Antonie van Rensburg
Department of Industrial and Systems Engineering
University of Pretoria
South Africa

ABSTRACT

Successful simulation studies depend on a variety of expertise such as statistics, computer programming, systems analysis and design techniques. Unfortunately these requirements limit the availability of competent simulation analysts, while simulation applications are in demand. This paper presents a possible solution to the shortage of expert simulation knowledge through the design of a simulation generator, a software tool that translates conceptual simulation models into computer code. The design process for the expert system and simulation components of the simulation generator uses an adapted software lifecycle methodology.

OPSOMMING

Suksesvolle simulasiestudies hang van verskeie kundigheidsonderwerpe af, soos byvoorbeeld statistiek, programmering, stelselanalise en ontwerptegnieke. Dié vereistes beperk die beskikbaarheid van geskikte simulasiekundiges, ten spyte dat simulasietoepassings in aanvraag is. Die artikel bied 'n moontlike oplossing aan vir die tekort aan simulasiekundigheid, deur die ontwikkelling van simulasiegenerators te bespreek, wat rekenaarprogrammatuur is wat konseptuele simulasiemodelle in rekenaarkode omskryf. Die ontwerp van die simulasiegenerator se ekspertstelsel- en simulasiekomponent is gebaseer op 'n aangepaste rekenaarprogrammatuurlewenssiklus metodologie.

1. INTRODUCTION TO SIMULATION GENERATORS

During World War II, British military leaders asked scientists and engineers to analyze several military problems: the deployment of radar and management of convoy, bombing, antisubmarine, and mining operations. The application of mathematics and the scientific method to military operations was called operations research. Today, the term operations research (or management science) means a scientific approach to decision making, which seeks to determine how best to design and operate a system, usually under conditions requiring the allocation of scare resources [12]. Two important steps in the operations research methodology are the problem formulation and mathematical model constructions. Constructions of these mathematical models rely on the application of mathematical techniques, with one such technique being simulation [12]. Decision makers use special purpose simulation software to analyse and study real-world systems as mathematical models, but successful simulation studies depend on a variety of expertise such as statistics, programming, systems analysis and design techniques. Unfortunately, these requirements limit the availability of competent simulation analysts, although simulation applications are in demand. This paper presents a possible solution to the shortage of expert simulation knowledge by introducing the concept of simulation generators as decision support tools.

A simulation generator is a software tool which develops and generates simulation models and simulation code, by using the decision maker's design specifications and turning it into a set of mathematical object relationships. The simulation generator translates these relationships into a special language syntax that are executed by a commercial software application (for example SIMAN from Systems Modelling Corporation) to present simulated real world results. The "intelligent" expert system that drives the simulation generator is a software application based on the working of artificial intelligence

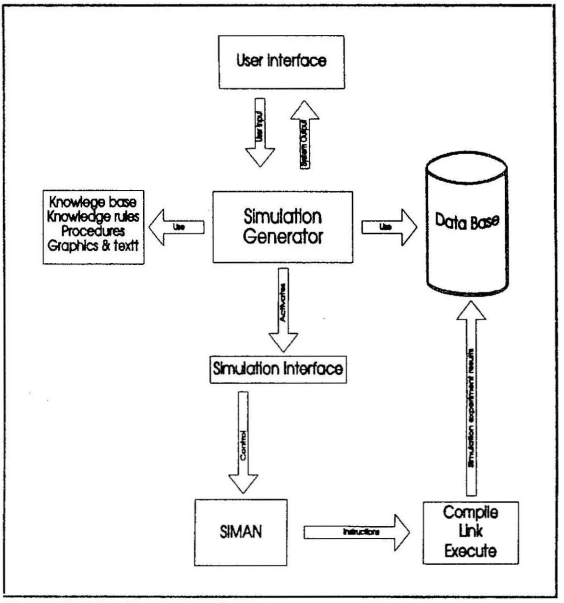

Figure 1 The Simulation Generator

techniques (see Figure 1). It uses expert statistical, programming, analysis and design knowledge to construct simulation models, freeing the decision maker from simulation model construction and coding.

2. THE SIMULATION GENERATOR COMPONENTS

Mathewson (1984) defines a simulation generator as: "*an interactive software tool that translates the logic of a model described in a relatively general symbolism into the code of a simulation language and so enables a computer to mimic model behaviour*" [2]. The first step for the simulation analyst is to analyse the real world system and then to formulate assumptions regarding system operations. These assumptions translate into a mathematical model that is executed by SIMAN, generating representative system performance results. During this process, the analyst applies expert statistical, programming, system analysis and design techniques. This expert knowledge, is build into the simulation generator allowing the simulation generator to mimic the simulation analyst's actions and to satisfy the need for rapid development, analysis and reuse of simulation models. The Simulation generator consists of two components, the simulation software part and the expert system, with a possible definition for expert systems according to BG Buchanan as "*An expert system is: A knowledge-based system that contains expert knowledge rather than general knowledge. It is a problem solving program that solves substantial problems generally conceded as being difficult and requiring expertise*". Expert systems are usually concerned with tasks that are normally performed by specially trained people [1,4]. There are in general three components to an expert system, the knowledge base, production rules and inference engine, with the knowledge base containing the expert knowledge needed for problem solving or decision support (Figure 1). The production rules are the set of heuristics the expert uses to reach solutions or decisions, and the inference engine, using these production rules, searches the knowledge base for possible answers. The simulation component produces simulation models that are stochastic, dynamic and discrete, which means that the simulation generator tries to capture the effect of system randomness over time by simulating real world objects as discrete entities (simulation

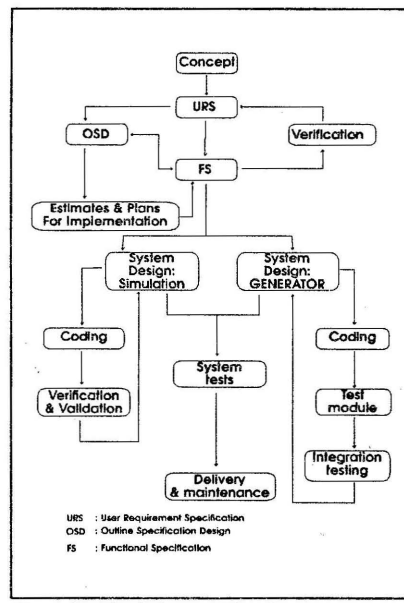

Figure 2 The Software Lifecycle

may be defined as the imitation of real world operations as it evolves over time [12]).

3. SOFTWARE ENGINEERING PRINCIPLES

As the builder needs a building plan to complete a house, one needs a design method for developing software applications. The simulation generator (as a software application) is designed according to such a method, software engineering. Macro and Buxten[1987] give a formal definition for software engineering as "*The establishment and use of sound engineering principles and good management practice, and the evolution of applicable tools and methods and their use as appropriate, in order to obtain-within known and adequate resource provisions - software that is of high quality in an explicitly defined sense*" [6]. Software engineering is the set of activities that produces high quality software applications within the known limitations of available resources. These activities include specification, feasibility and programming documentation, featuring as the User Requirement Specification (URS), Functional Specification (FS), Outline System Design (OSD) and Software Design (Figure 2). The URS, FS, OSD and Software Design are developed accordingly to a software lifecycle methodology, representing the lifecycle as a cyclic process that starts with the project idea and finishes with a working software application. The aim of this process is to establish a basic shared vocabulary that makes the software development process visible and comprehensible between the parties involved [6].

4. THE USER REQUIREMENT SPECIFICATION

A user requirement specification document is a document that states user requirements, proposed solutions and recommendations at a high definition level. The document focuses on the general problem description, user requirements and a proposed problem solution. These steps are demonstrated be using the Post Office distribution network as an example [10]:

i) General Problem Description

The Post Office operates a country wide parcel distribution network, with the mission to deliver parcels, on the right time at the right place. Two critical areas are identified for decision support, the development of optimal transport strategies and the strategic placing of central distribution locations (central hubs).

ii) User Requirements

The user wants a decision support tool to provide decision support on any sector of the distribution network, including payoffs between various alternatives.

iii) Proposed Problem Solution

The dynamic nature and size of the distribution network makes it impossible to determine optimal strategies at a static level, and it is proposed to develop

and implement a decision support tool that allows the decision maker to model the real world system in a dynamic way. The software tool models any given part of the network (or as a whole), to provide decision support.

The important technique of prototyping is introduced during the user requirement phase with the principle purpose of using a simulation model to inform the analysts with how and what to do information [5]. Prototypes at this level are used to clarify requirements and educate the user about simulation as a decision support tool.

5. THE FUNCTIONAL SPECIFICATION

The goal of the Functional Specification is to deliver a software system with the required functions, using an IOP (Input, Output and Process) classification, hardware and software specifications and a preliminary user interface design phase. Information gathered during the user requirement specification phase is unambiguously expressed in the Functional Specification, allowing the Functional Specification to act as an indisputable baseline for implementation [6]. The Functional Specification consists of the functional specification objective, functional analysis, IOP classification, system properties and the user interface.

i) The Functional Specification Objectives

The functional specification objective for the Post Office network comprises of: a) *deliver a decision support tool capable of, modelling dynamic distribution network interactions on a personal computer*, b) *allowing rapid analysis and design of simulation models on specified sectors of the distribution network*, c) *performing tradeoff analysis based on the number and type of transport vehicles, central hub locations, parcel services and alternative routes*, and c) *design simulation models in an intuitive manner with extensive expert support from the simulation generator.*

ii) The Functional Analysis

The Functional Analysis uses the functional objectives to determine main system functions. The IPO Classification then uses these functions to develop a meaningful functional structure for the proposed software with the input (I) and output (O) classification refering to data traffic across the human interface, comprising of screen and keyboard formats [6]. The process (P) represents a definite system action needed to transform input to output. The IPO technique tracks logical representations (trace-abilities) between the input and output categories through actions or processes. These actions define a set of system properties for the simulation generator. The classification is demonstrated by the example presented in Tabel 1.

iii) The Preliminary User Interface Design Phase

The user interface is the system function that allows communication between the simulation generator and the decision maker. The functional decomposition

processes suggest that the simulation generator user interface be divided into general interaction screens (the simulation generator's menu, input, output and instruction screens), and help support screens (hypertext text and graphical screens).

FUNCTION *Network Modelling*		
INPUT	**PROCESS**	**OUTPUT**
Route information.	Execute set-up of network model	Route information
	Write network model program	Network model programs
	Execute simulation connection	Simulation results

Tabel 1 The IPO Classification

The concluding phase of the Functional Specification deals with the system platform requirements, which splits into hardware and software specifications. Hardware requirements describe the target computer platform for the decision support tool and software requirements the decision support tool's application language. The definition of the system's functional objectives by means of the functional analysis, IPO, system properties, user interface, hardware and software specifications, extend the user requirements into agreeable detail, both from the user and developer viewpoints.

6. THE OUTLINE SYSTEM DESIGN

The Outline System Design's (OSD) objective is to check implementation considerations against functional specification unfeasibilities by using a hardware, software and "tools" taxonomy (taxonomy is the classification of items with specialised terminologies) [6]. The hardware taxonomy establishes hardware alternatives, the software taxonomy determines software modules and the "tools" taxonomy applies development tools in the software coding process. Cohesion and coupling criteria determine software modules and module code sizes for implementation where cohesion is associated with the maximisation of the internal module strength, and coupling the minimizing and simplifying module interfaces [6,7,8,11]. The software taxonomies produce results that are used to calculate effort and timescale estimates by means of activities-planning techniques, such as CPM and PERT (both methods use graphical approaches to depict "resource constraining" activities in an ordered sequence). In the final form, the Outline System Design supplies implementable information to the Functional Specification by means of a hardware, software and tools taxonomy.

7. SOFTWARE DESIGN

Software Design follows the specification phases and produces implementable software applications with the required user requirements. Different methodologies and design approaches, such as modular specification and pseudocode are used in the Software Design process. Three basic structure applications define module specifications, sequence statements (*Begin End*), conditional statements (*IF Then*) and repetition statements (*While Do*) [5]. The basic principles of structured programming provide the following pseudocode structures:

i) Constructs are all single-exit control flow statements,
ii) structures are hierarchical; that is, within any box a low-level construct of any three programming types may be introduced, and
iii) the three constructs are sufficient to program any sequential problem [6,11].

The following example shows the pseudocode for module 03-03-0, (done in KnowledgePro syntax), with the *Repeat...While* statement (line three) an example of the While...Do statement, the *IF...Then* statement example in line seven, and a *Begin...End* statement in line eleven.

PSEUDOCODE EXAMPLE

```
Line 1    SELECT picture
Line 2    DISPLAY picture
Line 3    REPEAT
Line 4        IF topic = active screen buttonLine 5
THEN DISPLAY helpscreen
Line 6        UNTIL user QUIT
Line 7    IF topic = active button_2
Line 8        THEN DO SUBROUTINE_1
Line 9    END
Line 10
```

```
Line 11   SUBROUTINE_1
Line 12   BEGIN
Line 13       SELECT picture
Line 14       DISPLAY picture
Line 15       REPEAT
Line 16               IF topic = active button
Line 17               THEN DISPLAY helpscreen
Line 18       UNTIL user QUIT
Line 19   END
```

The final software design follows the pseudocode descriptions, by converting modular pseudocode into programming code. The conversion steps for the expert system module, *Simulation Interface* (03-02-03), illustrates this step:

Pseudocode For The Set-up Part Of The Simulation Interface

MODULE NUMBER: 03-02-03

```
ENTER setup information
        DETERMINE current system information
WRITE batch file
END
```

The pseudocode program determines current system information and then proceeds to write the SIMAN control batch file.

Code For The Set-up Part Of The Simulation Interface

SOFTWARE MODULE: 03-02-03

ENTER setup information

```
(*SETUP PROCEDURE*)
TOPIC setup.
h_d = current_directory().
H_1 = STRING_REPLACE(?HUIDIG,'C:','CD').
window('SYSTEM SETUP',white,red,white,5,3,65,10).
write(con:,#x1,#y2,'TO RUN THE SYSTEM THE
FOLLOWING INFORMATION IS REQUIRED',
#x1,#y4,'Location of SIMAN files ',
#x1,#y5,' * On which hard drive (i.e. d )',
#x1,#y6,' * Sub-directory name ( i.e. siman)',
#x1,#y8,'Name of model file',#x1,#y9,
'Name of experimental file ').
```

DETERMINE current system information

```
(*DETERMINE PATH FOR SIMAN FILES*)
read_response([#x40,#y5],loc_1).
read_response([#x40,#y6],loc_2).
c_1 = concat(?loc_1,':\').
c_2 = ?loc_2.
loc = concat(?c_1,?c_2).
pad = concat('cd\',?loc_2).
(*NAMING OF MODEL AND EXPERIMENTAL FILES*)
write(con:,#x45,#y8,'SNEL_FIN.MOD').
read_response([#x40,#y8],mod).
write(con:,#x45,#y9,'SNEL_FIN.EXP').
read_response([#x40,#y9],exp).
mod = ('SNEL_FIN').
exp = ('SNEL_FIN').
```

WRITE batch file

```
(*WRITE BATCH FILE FOR EXECUTION OF
SIMULATION*)
(*CREATE NEW FILE*)
new_file('sim.bat').
write([['sim.bat'],
#l,
'copy snel_fin.mod',#s,?loc,#n,
'copy snel_fin.exp',#s,?loc,#n,
concat(?loc_1,':'),#n,
?pad,#n,
,'model #s',?mod,' ',?mod,#n
,'expmt #o',?exp,' ',?exp,#n
,'linker #o',?mod,' ',?exp,' ',?mod,#n
,'siman ',?mod,#n,
'outpt',#n,
?H_1,#n,
'kp BEGIN BEGIN.ckb -t').
close('sim.bat').
close_window().
model = concat(?mod,'.mod').
exper = concat(?exp,'.exp').
(*PATH FOR MODEL FILE*)
pth_1 = concat(?loc,?model).
(*PATH FOR EXPERIMENTAL FILE*)
pth_2 = concat(?loc,?exper).
close_window().
(*compiled version*)
load('begin').
close('snel_fin.ckb').
(*load('begin').
close('snel_fin.kb').*)
END.
```

8. THE SOFTWARE LIFECYCLE AND DESIGN METHODOLOGY

A design methodology is a collection of procedures, rules, criteria or guidelines, describing a way of carrying out the design process, from specification to detailed building descriptions [6]. Appropriate design approaches have to be chosen to ensure good design at low levels, accomplished by using customized problem solution methodologies (comprising of different procedures), rather than "off the shelve" methodologies. This paragraph introduces different design procedures for both the expert system and simulation part of the simulation generator to accommodate special simulation characteristics (Figure 2).

i) The Simulation Methodology

The goal of the simulation methodology is to develop a generic simulation model that predefines user simulation models. The expert system design depends on the effective design of the generic simulation model, as this model forms the basis for the expert

system design. The simulation methodology steps are the problem definition, system definition, process definition, data modelling and process modelling.

a) Problem Definition and System Definition

The aim of the problem definition is to motivate an explicit experimental goal for the simulation model from the user requirement objectives, as was stated in the User Requirements Specification (such an example can be: provide decision support to decide whether a new infrastructure has to be built to accommodate additional workloads) [9]. The System definition tries to identify a small subset of characteristics that are sufficient to serve the study objectives. This means simplifying reality to the point where there is no significant loss in accuracy of results.

b) Data Modelling

Data modelling activities include the identification of entities and entity relationships. Entities are tangible objects in the system definition, for example in the distribution network environment context it may be parcels, transport vehicles and sorting stations are defined as entities. A useful method for depicting conceptual relationships (or entity interactions) are the graphical representation formats used in information engineering, known as entity relationship diagrams. The entity relationship diagram presents entities as rectangles and relationships as straight lines, defining relationships both ways between entities.

c) Process Modelling.

Process modelling consists of the following steps, process definitions, process decompositions and process procedures. The goal of process modelling is to identify process in the system, map entities to these processes and then develop process procedures. The application of these steps produced the following example for regional stations in a distribution network, forming the basis for the simulation model development.

Local Station Procedure

1. *Destination shipment arrives.*
2. *Unload shipment.*
3. *Determine parcel origin.*
4. *If parcel origin equals same destination area then go to 9.*
5. *Determine applicable parcel service type.*
6. *Place parcel in service channel according to service type.*
7. *Load service channel shipments.*
8. *Sent transport resource to central hub.*
9. *Sort parcels for local distribution.*
10. *Deliver parcels.*

ii) The Generator Methodology

The generator methodology incorporates functional decomposition and process driven design techniques, as used in the Information Engineering discipline. A functional

decomposition of a business comprises of functional area groupings into business functions, which as a group of activities, support one aspect of furthering the enterprise mission [7]. This concept is extended to the development of the expert system, with the identification of functional specification objectives as business areas (user requirements partitions into "business" functions). The next step involves the division of functions into processes, relating processes to specific acts that have definable beginning and ending points. The functional decomposition indicates "what" has to be done to support user requirements, while processes are concerned with "how" the "what" is to be done.

a) Functional Decomposition

A functional decomposition is performed on the functional specification objectives resulting in partitioned functions as shown in Figure 3. The function, *Generator System* represents the simulation generator, incorporating two main functions, the *User Interface* and *System Support*. *System Support* includes functions such as mouse support and stand alone execution. The *User Interface* consists of four functions, *Database Information*, *Network Modelling*, *Help Support* and *Output*. *Database Information* handles network information needed for model development, with the *User Input* and *dBASEIII Interface* functions providing data requirements. The *Network Modelling* function, responsible for simulation model development, coding and model execution, includes *Network Model Development*, *Network Model Generation* and *Simulation Interface* functions. *Help support* aids the user in system use and provides support by means of hypertext text (*Text Support*) and hypertext graphics (*Graphical Support*). The last function, *Output* provides statistical analysis (*Statistical Analysis*), evaluation (*Evaluation*) and simulation output interpretation (*Results Interpretation*).

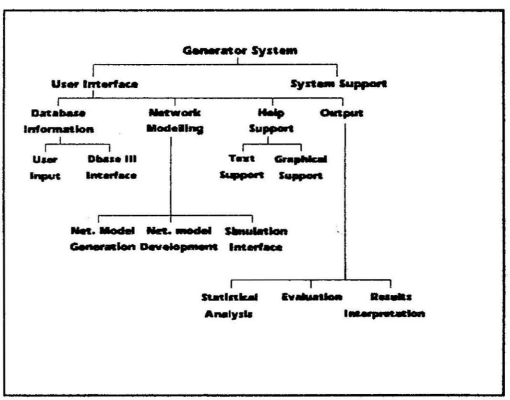

Figure 3 Functional Decomposition

b) Process Driven Design

Process Driven Design is a mixture of Transform Driven Design and Process-orientated Design. Transform Driven design's key component, data flow analysis combines with the Process-orientated Design's top-down process analysis approach to present data flows and transformations (processes) as functions in a functional decomposition manner. The process driven design notations make use of information flow diagrams, which are intuitively and simple to understand [6].

Functions (Figure 3) are subjected to an information flow analysis to ensure a valid process driven design, which proceeds into the coding phase.

9. CONCLUSION

The ability to customize simulation applications, will ensure that traditional simulation languages such as SIMAN will be the heart of simulation generators. Two factors contribute to the non-attractiveness of developing dedicated simulation software, the time factor involved in writing a new knowledge based simulation and the confinement to particular problem domains where as a general language like SIMAN, is quickly customised to suit particular needs for developing simulation applications. The development and application of a simulation generator do not guarantee a satisfied user but the user is satisfied with the software system if the system is of high quality and with explicitly defined user requested functions.

The use of different technologies is not important. The integration of technologies to enhance and provide solutions are.

10. REFERENCES

1. Haddock J and O'Keefe, "*Using Artificial Intelligence To Facilitate Manufacturing Systems Simulation*", Computers and Industrial Engineering, Vol. 18, No 3, 1990, pp. 275-283.
2. Mathewson S C, "*The application of program generator software and its extensions to discrete event simulation modelling*", IIE Transactions 16, No 1, 1984.
3. Moser J G, "*Integration of artificial intelligence and simulation in a comprehensive decision-support system*", Simulation, Vol. 47, No. 6 , December 1986, pp. 223-229
4. Shannon R E, Mayer R and Adelsberger H H, "*Expert systems and simulation*", Simulation, Vol. 44, No. 6, June 1985, pp. 275-284.
5. Connor D, "*Information System Specification and Design Road Map*", Prentice-Hall Inc., 1985.
6. Macro A, Buxton J, "*The Craft Of Software Engineering*", Addison-Wesley Publishing Company, 1987.
7. Martin J, "*Information Engineering, Book I, Book II and Book III*", Prentice-Hall Inc., 1990.
8. Mcdermid D C, "*Software Engineering for Information Systems*", Blackwell Scientific Publications,1989.
9. Pegden C D, Shannon R E and Sadowski R P, "*Introduction to Simulation Using SIMAN*", McGraw-Hill, New York, 1990.
10. Van Rensburg ACJ, "*The design of a simulation generator*", Department of Industrial Engineering, MENG dissertation, 1993.
11. Whitten, Bentley, Blow, "*Systems analysis and design methods*", Irwin Honewood, Second Edition, 1989.
12. Winston, W L, "*Operations Research, Applications and Algorithms*", PWS-Kent, Second Edition, 1992.