The Use of an Object Oriented Technique for Fault Diagnosis in Nuclear Reactors

by

N. A. Jalel and H. Nicholson

Department of Control Engineering
University of Sheffield
Mappin Street
Sheffield S1 3JD

Research Report No. 414
November 1990

1 Abstract

In recent years there has been an increased growth of interest in object oriented programming which is a new approach to software construction having wide application. The possibility of using object oriented programming to build a software package that will assist the nuclear reactor operator in diagnosing any faults or alarms in the Loss Of Fluid Test (LOFT), a small scale pressurised water reactor, reactor is investigated in this work.

2 Introduction

Object oriented programming is a powerful programming concept that is used widely in expert system packages, graphics, systems programming, database management systems and for many other demanding applications, including the analysis of fault trees and overall system reliability [17].

Smalltalk-80 is an object oriented language used to build the Troubleshooting Assistant System Prototype which is an expert system developed to provide technicians with advice and assistance in repairing faults in a steam generator [2].

Object oriented programming can be used for the simulation of continuous or discrete processes since it is based on a hierarchical classification of procedures and data that allows modeling of the systems in terms of objects and actions on objects. Flavors is an object oriented language used to simulate a nuclear reactor [3].

An interesting application of object oriented programming has been the building of a fault diagnosis expert system for a personal computer. The process of diagnosis is divided into fault classifications to determine which parts of causal knowledge are valid and causal diagnosis where the selected parts are used to identify the cause of the fault [4].

A survey of some of the applications which highlight the use of the Smalltalk language is given by Alexander [1]. The applications include, symbolic algebra knowledge collection and expert system shells, Prolog interfaces and integrated circuit design.

Some of the applications of object oriented programming involve implementing a general window system which includes menus, check boxes and bottoms, building a computer assisted navigation system, developing a computer aided design system and knowledge base and expert system applications [6].

Some work has been done to link object oriented programming and the theory of Petri nets. It involves modeling, simulating and the evaluation of Petri nets using an object oriented style of programming. The applications include designing a system within the Smalltalk environment which transforms programs to Petri nets and subsequently checks the nets for concurrence and invariants. The Petri net environment implemented in an object oriented programming language has been used for process control and for building distributed systems [7,8,9,10].

Object oriented programming incorporates a powerful set of concepts that can be applied in any field in the same way as structured programming. There are some applications for which object oriented programming particularly stands out, including exploratory programming environments where the program development is carried out by making modification to what already exists, frameworks and toolkits, user interface, for building integrated systems and in the field of artificial intelligence and knowledge representation [11].

In this paper, the object oriented programming concept is illustrated, a brief description of Smalltalk which is used to build the system is given, a general description of the system is explained and finally the system performance is examined using a hypothetical accident simulated in the LOFT reactor model.

3 Object Oriented Programming

The main idea behind the development of object oriented programming stemmed from the Smalltalk project at the Xerox Palo Alto Research Centre (PARC) in the early 1970's. The output of the research is the development and release of the programming environment Smalltalk-80. In the Smalltalk project the technical influences were from the SIMULA-67 language with its class mechanism and inheritance and the LISP language with its dynamic binding and interactive environment.

There are two groups for object oriented language, the Smalltalk group which includes Smalltalk-80 and Smalltalk/V which run on an IBM PC, while the other group is the C group which involves C++ and objective C. In addition, a number of other object oriented languages have recently become available such as Loops, Flavors, and Common LISP which are LISP-based systems. There are also Eiffle, Object Logo and Object Pascal.

Object oriented programming can be defined generally as a new way of thinking about data, procedures and the relationships among them and emphasises reusability and sharing. It can be implemented in any language in which it's concepts are provided using different constructs and to differing degrees [12,13,14,15,5,16].

3.1 Elements of Object Oriented Programming

A language must exhibit four characteristics to support object oriented programming:

1. Encapsulation
   This is the ability to pack a structure so that it is accessed in terms of its properties rather than its implementation, and accessed in a way that is carefully controlled and defined. It simply means that the two essential elements of processing, data and code, are packaged together inseparably in a capsule called an object.

2. Polymorphism
   Polymorphism is a unique characteristic of object oriented programming where different objects respond to the same message with their own unique behaviour. It is the ability to simplify code that deals uniformly with many different structures. The program is permitted to refer to objects of more than one class, and operations are permitted to have different realizations in different classes.

3. Inheritance
   Inheritance is the concept that is used to define objects that are almost like other objects. This technique is important because it is possible to declare that certain specifications are shared by multiple parts of a program. Inheritance allows us to reuse the code while knowing only its properties and without altering it. It enables programmers to create classes and, therefore, objects that are specializations of other objects. Creating a specialization of an existing class is called subclassing where the subclass inherits the variables and methods from it's superclass.

4. Storage management
   The new style of programming encourages the creation of internal structures corresponding to things in the problem domain. Object oriented languages provide aspects for the structured allocation and release of memory to do this safely and efficiently.

3.2 Mechanisms of Object Oriented Languages

The key concepts for object oriented programming techniques are:

1. Objects

   An object is central to the object oriented program and it represents something, often from the application domain such as a car or a computer. Objects are analogous to data structures such as numbers, arrays and records in other conventional languages. They combine the properties of procedures and data since they perform computations and save the local state. The data within the object can be accessed only by the code surrounding the object where the code is private in that it can't be modified by other objects.

2. Messages

   The action in object oriented programming comes from sending messages between objects. It is a request for an object to carry out one of its operations and is similar to function calls in conventional language. A message consists of three parts, a receiver object which is the object to which the message was sent and determines how to carry out the requested operation, a selector which tells the object what to do with the message and zero or more arguments. For example consider in Smalltalk the simple message

   ```
   'I hope I will get my PhD' size
   ```

   After evaluation the result should be the integer 24, however in this message the string is the receiver object, the message selector is size and there are no arguments.

3. Methods

   When an object receives a message, it must decide what to do and who will do it. Methods can be defined as the code associated with the object that takes care of handling the message, or in other words it is the algorithm that determines an object's behaviour and performance. Methods are like function definitions in other conventional languages and are evaluated with an object as an output result when a message is sent to an object. The following example shows that the object sent the message "at: put:"

   ```
   'pass the exam' at:1 put:P        [Pass the exam]
   ```

   In the example, the object recognized that it was a string and interpreted the message that a "P" should be placed at the first character position.

4. Classes

   Objects fall naturally into classes and subclasses based on their similarities, thus a class describes the implementation of a set of objects that all represent the same kind of system component. They define the behaviour of similar objects by specifying the variables they contain and the methods available for responding to messages sent to them. In most object oriented programming environments there is a hierarchy of classes and subclasses where every object is an instance or a member of a class. Thus for example in Smalltalk #(1 2 3) is instance of class Array, and all objects know which class they belong to.

4 Smalltalk

Smalltalk is a very pure object oriented language, and the concept of objects penetrates the whole Smalltalk system. It is a graphical interactive programming environment which provides many tools to aid the user to develop the programming code using windows, pop-up menus and a mouse in order to simplify computer usage.

Smalltalk is a tool for enhancing communication and creativity, encouraging rapid prototyping and the exploratory development of applications. It is a programming language for developing solutions to both simple and complex problems and has been used for simulation, expert systems, computerized typesetting and integrated programming environments.

Using Smalltalk, the program is built piece by piece and the result is seen immediately since with Smalltalk it is possible to edit and install small code parts without lengthy compile and link sessions. It also assists the programmer to examine in detail the execution of the code and to easily modify and refine the code.

In this work, Smalltalk/V is adopted to run on an IBM compatible with 512k RAM, monochrome or colour monitor and MS-DOS version 2.0 or later. Smalltalk/V is an advanced implementation of Smalltalk and is used for creating Smalltalk programs and an environment for a personal computer. It also provides its own components, including the Smalltalk source code as building blocks for the user to include in his applications. Reference [18] provides a full description and explanation of the Smalltalk/V system.

5 General Description of the System

The main purpose of the prototype system is to help the nuclear reactor operator in identifying and diagnosing any alarms or faults that could occur in the LOFT reactor. The LOFT reactor model is simulated using C language and runs on a Sun workstation, and the state variables of the simulated model are stored in a file on a floppy disk.

The system generally consists of two classes, one is the subclass for the main class object while the other is the subclass for the first one. The first one, Accident, consists of 19 'methods' and is intended to read the values of the simulated state variables from the floppy disk, assign the values for the right variables, store the values of each state variable in an array where the dimension of the array is equal to the number of times the state variables are read and for each state variable convert the variables from string to an integer. From the 19 'methods', one 'method' is for reading data from the floppy disk while the rest are for assigning the values for the state variables, where one 'method' is for each state variable. The 'method' for reading the data from the floppy disk is coded in the form of

```
|input output xValue|
xValue := Array new:6.
input := DiskA file: 'naj.dat'.
input lineDelimiter.
Output := ReadWriteStream on: Array new.
[input atEnd]
whileFalse:[output nextPut: input nextLine].
output reset.
1 to: 5 by: 2 do: [:i |
      xValue at: i put: output next.
      xValue at: i + 1 put: output peak.
      output skip: 1].
^xValue
```

The second one, Faulty, is a subclass of Accident and thus in class Faulty the variables and 'methods' are inherited from its super class in addition to containing the ones defined in its own. It involves 46 'methods' and it is responsible for monitoring the state variables for any increase or decrease from the normal operating limit, asking the user some questions concerning information about the whole power plant through using the Prompter which is a special kind of window allowing the system to ask questions, open a window to maintain all the output results and a menu to provide the user with the options available with the system and to diagnose any alarms or faults in the reactor where the symptoms for each alarm or fault are modelled using a single 'method'. Figure 1 illustrates the overall structure of the system.

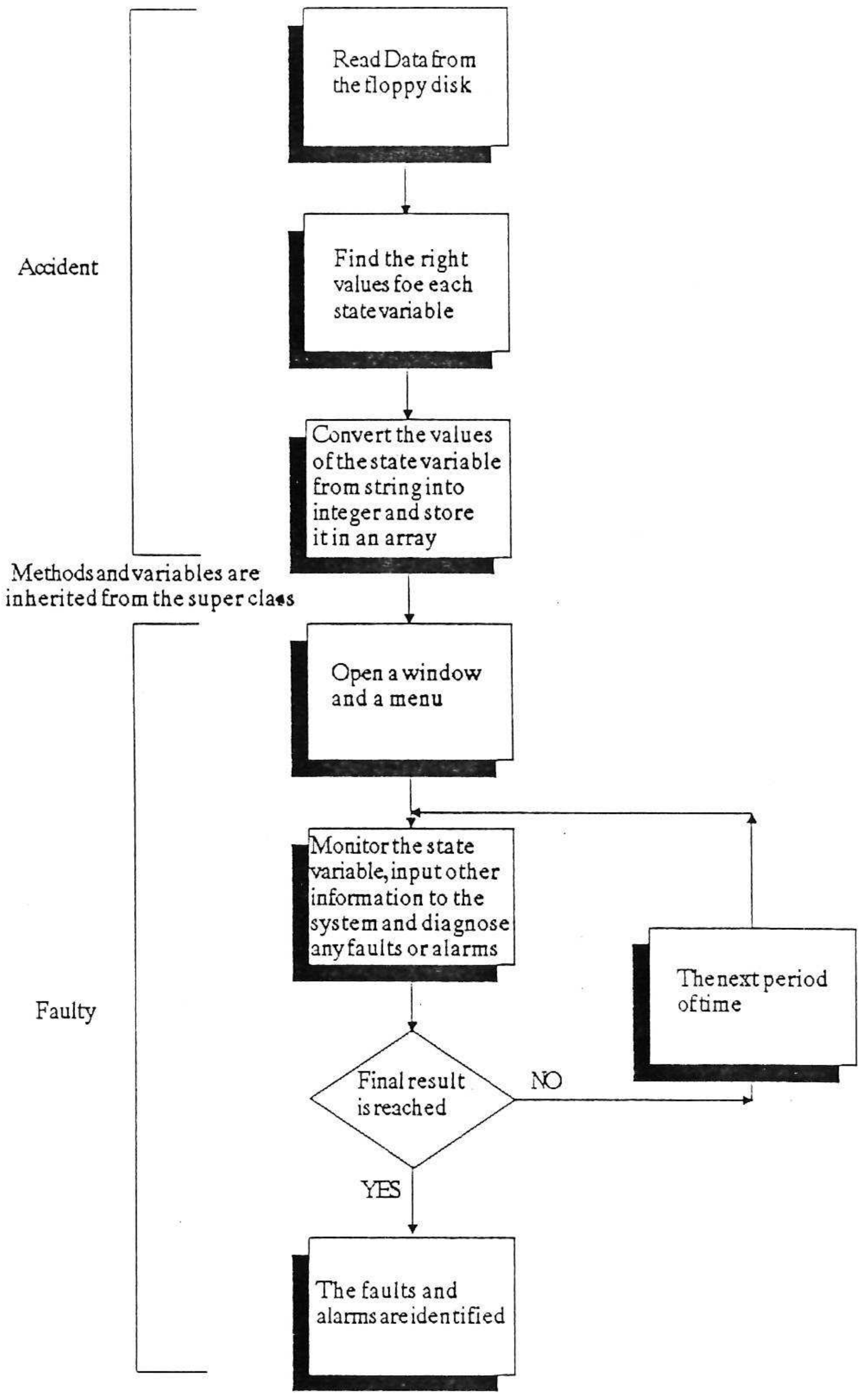

Read Data from the floppy disk
Find the right values foe each state variable
Convert the values of the state variable from string into integer and store it in an array
Accident
Methods and variables are inherited from the super class
Open a window and a menu
Monitor the state variable, input other information to the system and diagnose any faults or alarms
The next period of time
Final result is reached
NO
YES
The faults and alarms are identified
Faulty

Figure 1: The structure of the system

6 System Performance

The prototype system is built to assist the operator in identifying any faults or alarms in the reactor. The ability of the system is tested through the use of a hypothetical accident which is assumed to be a failure in the reactor control rod.

The accident is initiated at a time greater than 2.0 sec by a drop in the control rod causing a change in the rod reactivity from 0.0 to -0.5 ($). As shown in figure 2, the symptoms for this type of accident are a decrease in the reactor power, in the average coolant temperature and in the pressuriser pressure, and the indicator for the control rod bottom position should be on.

The system output environment is illustrated in figure 3, and consists of two windows and a menu, the top window contains a fixed message concerning the purpose of the system and advices the user to use the mouse to see the system menu and the options available. In the bottom window the final conclusion which includes; the fault diagnosis, the alarm analysis and the values of the state variables are shown while the system menu provides the user with the following options, the State variable which provides the user with the values of the important state variables of the LOFT reactor at a fixed period of time, Alarm analysis to identify the different alarms that might occur, Accident analysis to diagnose the different faults that might happen in the reactor and Continue to carry on the calculation for the next period of time.

The accident starts at a time greater than 2.0 sec, so at a time 0.0 sec the reactor is at normal operation. After choosing the State variable option from the menu, the values of the state variables at a time 0.0 sec are shown in figure 4 while figures 5 and 6 illustrate the system response after choosing the Alarm and Accident analysis options, respectively. The system indicates that there are no alarms or faults in the reactor at this period of time.

The values of the state variables at time 10 sec are illustrated in figure 7 where it is possible to indicate the decrease in the reactor power due to rod reactivity. Figure 8 indicates the occurrence of the first alarm for failure in the rod drop, loss of coolant and steam generator tube rupture. The system diagnoses the accident at this time and indicates the occurrence of failure in the rod drop as shown in figure 9.

If the user asks the system for further calculation at the end of the availabe set of data, the system will respond with a message indicating the end of data as illustrated in figure 10.

7 Conclusion

The object oriented style of programming is very promising and attractive and hopefully will be used more extensively in the future. The system results are very encouraging and are able to diagnose the fault in the reactor as illustrated with the hypothetical accident.

Programming in Smalltalk can be improved dramatically by practice since the program development is carried out entirely by making modification to what already exists in the Smalltalk environment. The main advantage of programming with Smalltalk/V is the possibility of dividing the program into different levels and each level can be built independently from the others, thus each level can be easily modified and refined, in the other hand the main difficulty with Smalltalk/V is the problem of using it espacially for a person who is used to the ordinary style of programming.

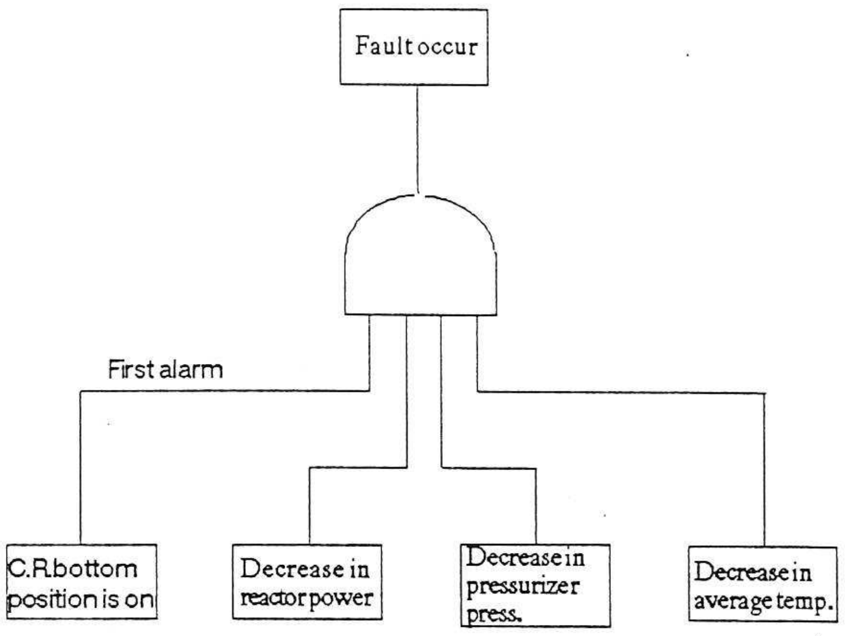
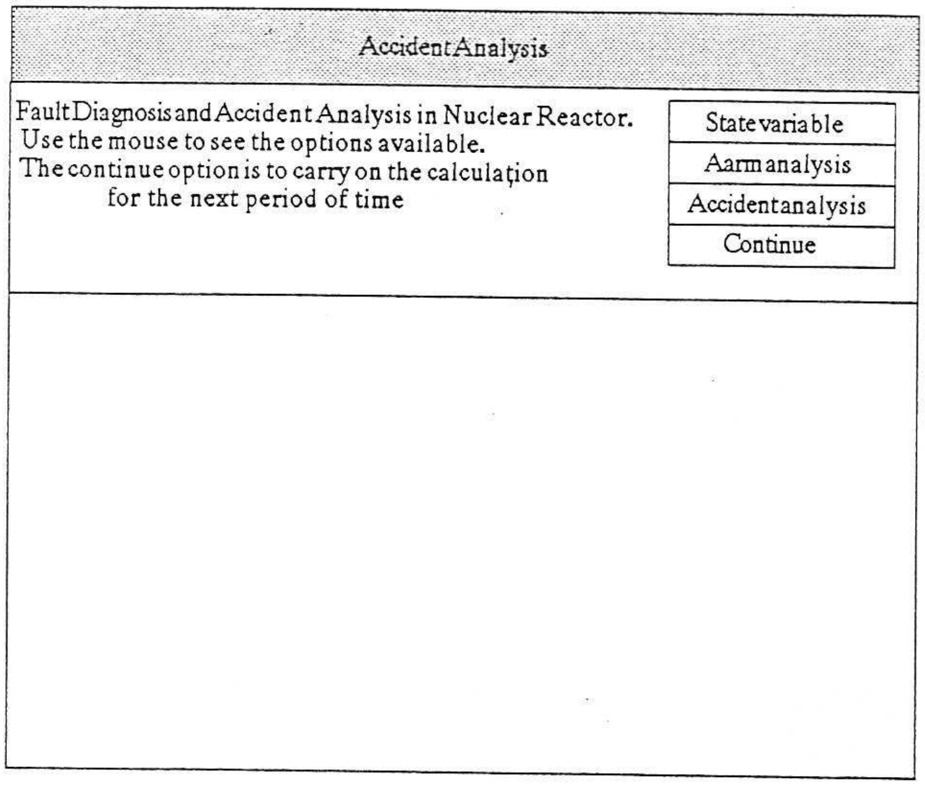
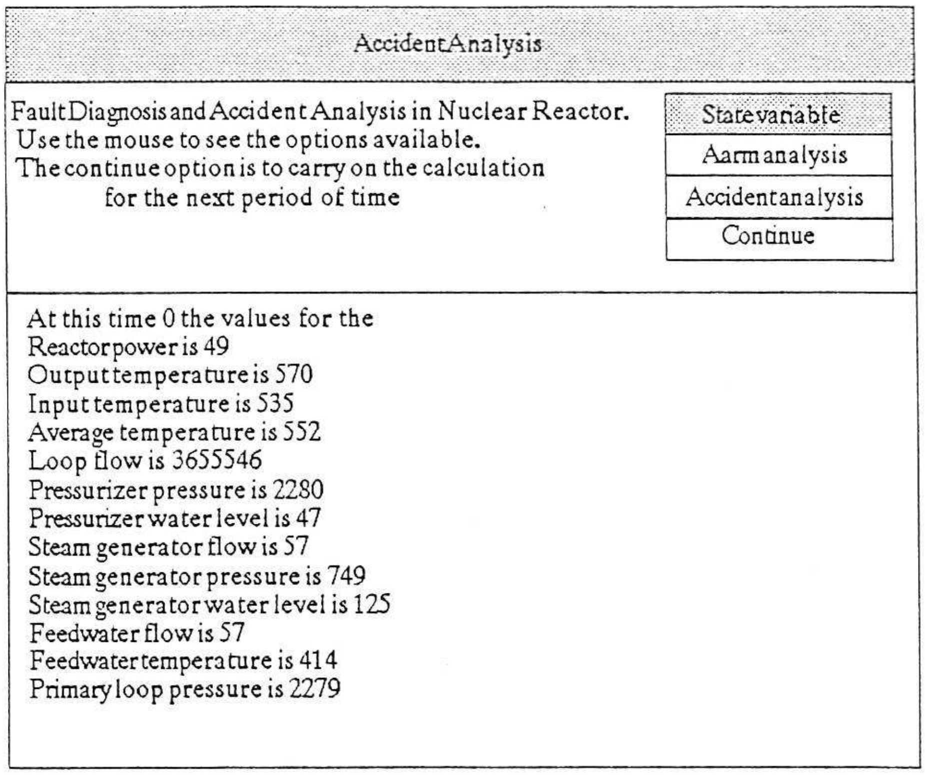

Fault occur
First alarm
C.R.bottom position is on
Decrease in reactor power
Decrease in pressurizer press.
Decrease in average temp.

Figure 2: Fault tree for the rod drop fault



Fault Diagnosis and Accident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

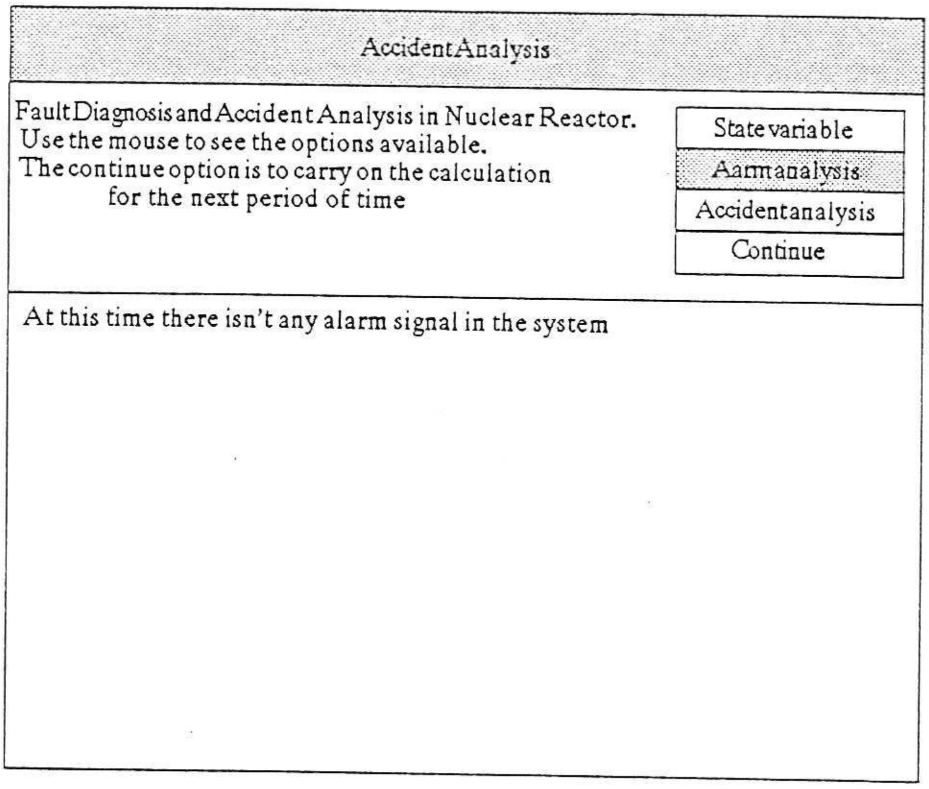
Figure 3: System output environment



FaultDiagnosisandAccident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

```
At this time 0 the values for the
Reactorpower is 49
Outputtemperature is 570
Input temperature is 535
Average temperature is 552
Loop flow is 3655546
Pressurizer pressure is 2280
Pressurizer water level is 47
Steam generator flow is 57
Steam generator pressure is 749
Steam generator water level is 125
Feedwater flow is 57
Feedwater temperature is 414
Primaryloop pressure is 2279
```

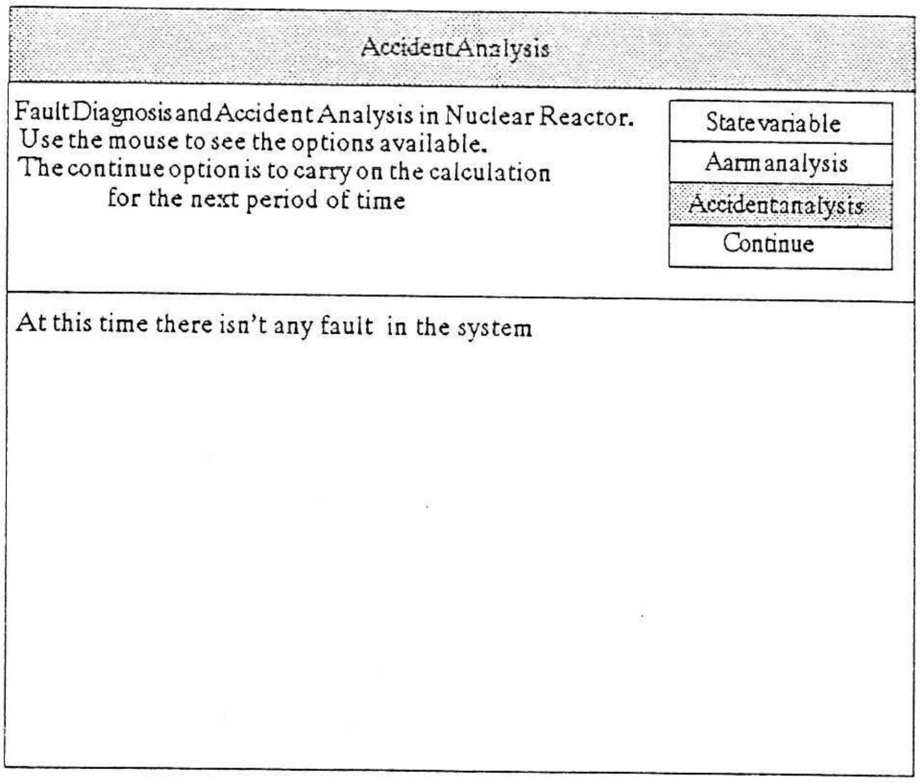
Figure 4: Values of the state variables at time 0 sec



FaultDiagnosisandAccident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

```
At this time there isn't any alarm signal in the system
```

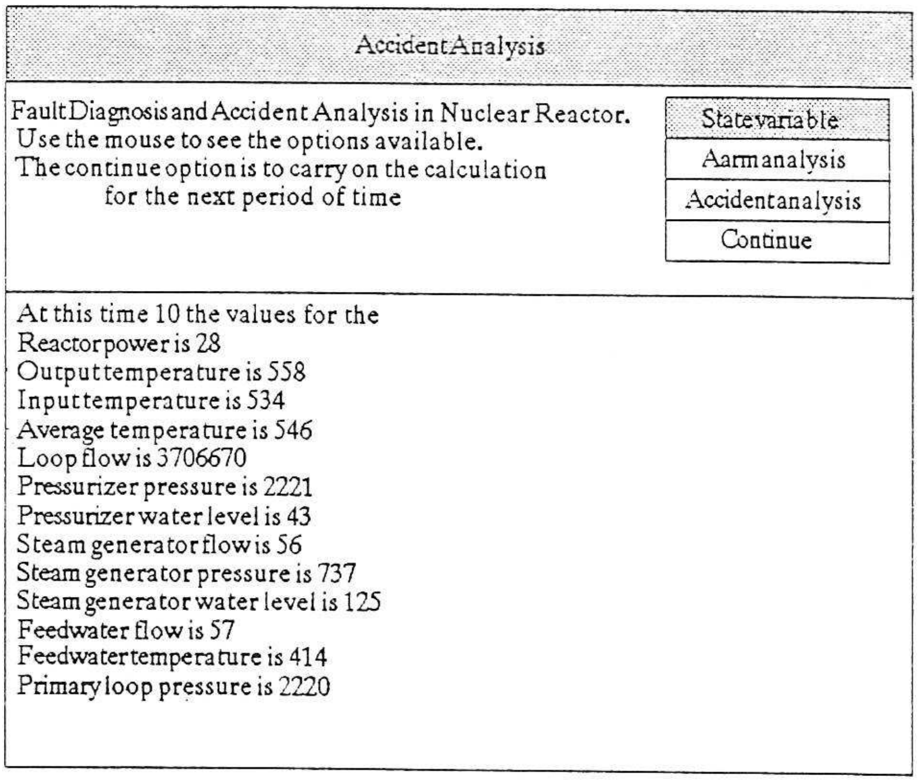
Figure 5: Selecting the Alarm analysis option at time 0 sec



FaultDiagnosis and Accident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

At this time there isn't any fault in the system

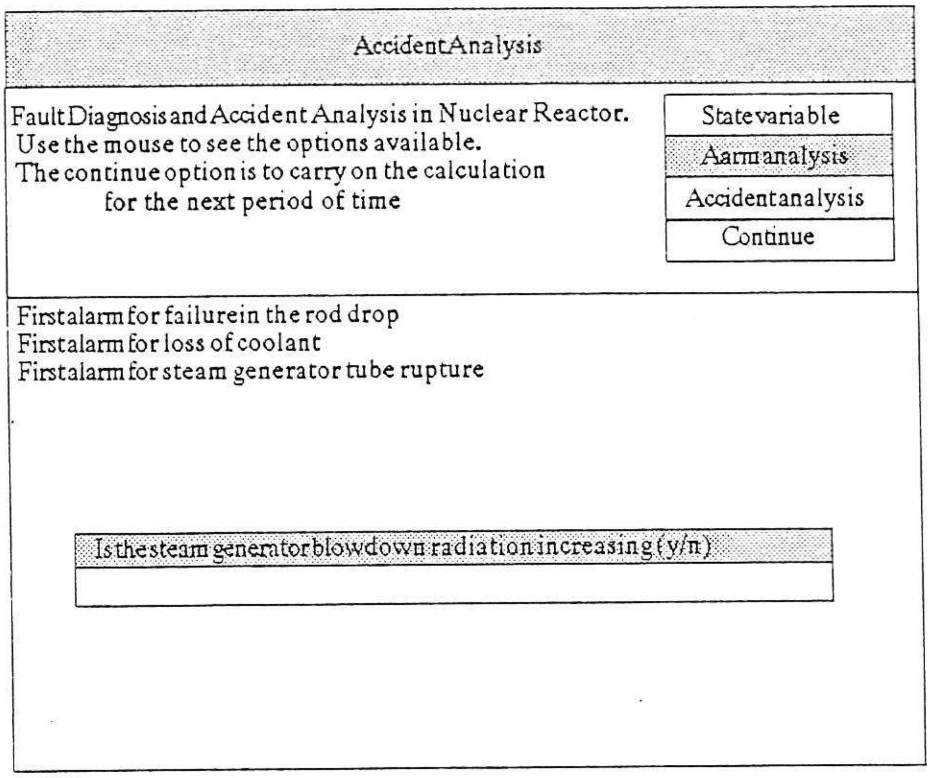
Figure 6: Selecting the Accident analysis option at time 0 sec



FaultDiagnosis and Accident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

```
At this time 10 the values for the
Reactorpower is 28
Outputtemperature is 558
Inputtemperature is 534
Average temperature is 546
Loop flow is 3706670
Pressurizer pressure is 2221
Pressurizer water level is 43
Steam generatorflow is 56
Steam generator pressure is 737
Steam generator water level is 125
Feedwater flow is 57
Feedwatertemperature is 414
Primary loop pressure is 2220
```

Figure 7: Values of the state variables at time 10 sec

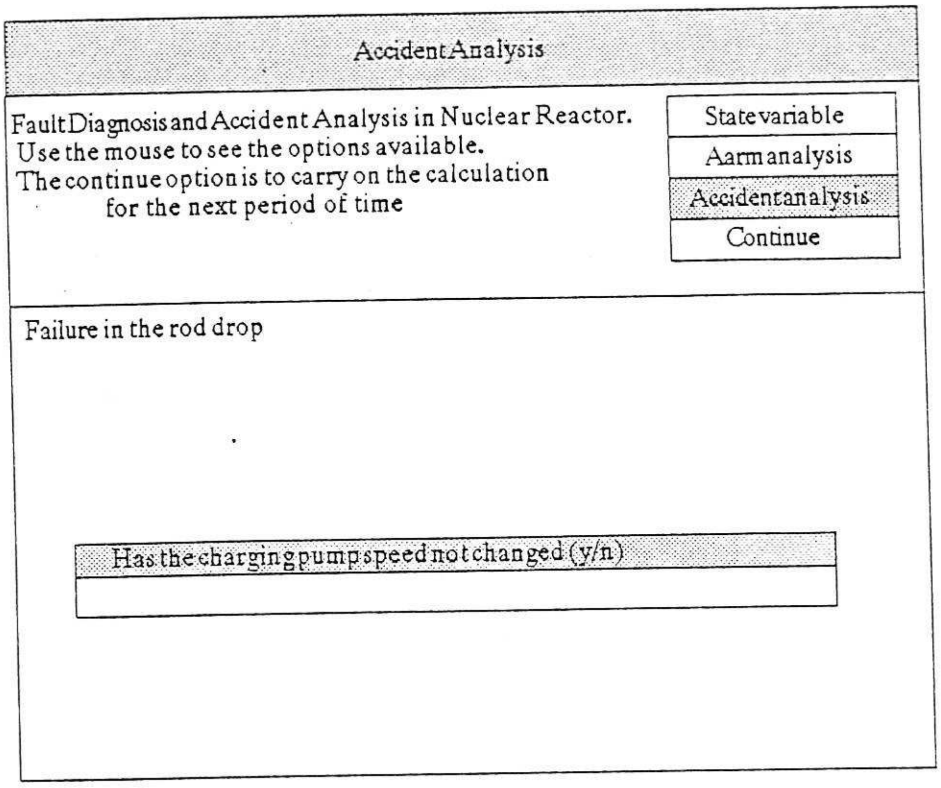


FaultDiagnosisandAccidentAnalysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

Firstalarm for failurein the rod drop
Firstalarm for loss of coolant
Firstalarm for steam generator tube rupture

Is the steam generator blowdown radiation increasing (y/n)

Figure 8: Selecting the Alarm analysis option at time 10 sec

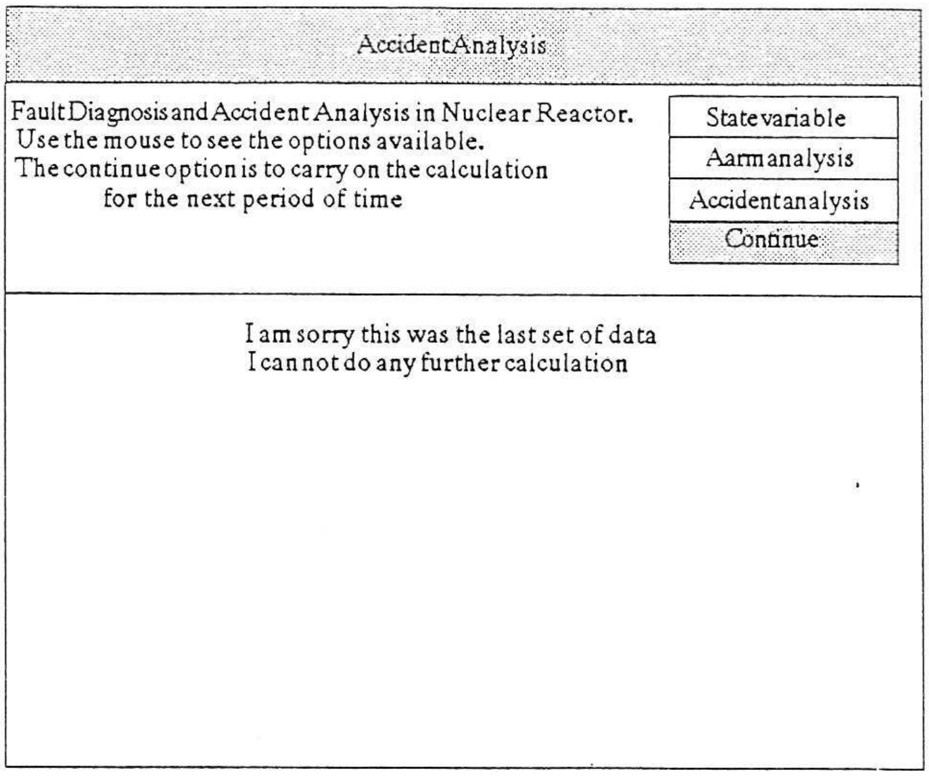


FaultDiagnosisandAccidentAnalysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

Failure in the rod drop

Has the charging pump speed not changed (y/n)

Figure 9: Selecting the Accident analysis option at time 10 sec


AccidentAnalysis

FaultDiagnosisandAccident Analysis in Nuclear Reactor.
Use the mouse to see the options available.
The continue option is to carry on the calculation
for the next period of time

I am sorry this was the last set of data
I can not do any further calculation

Figure 10: The response of the system at the last set of data

References

[1] Alexander, J. (1985). *Exploratory Application Development Using Smalltalk.* Computer Research Laboratory, Tektronix Inc., USA, Technical Report No. CR-85-16, pp. 1-11.

[2] Alexander, J. and Freiling, M. (1984). *Building an Expert System In Smalltalk-80.* Artificial Intelligence Department, Computer Research Laboratory, Technical Report No. CR-85-06, pp. 1-14.

[3] Robinson, J. and Otaduy, P. (1990). *An Object Oriented Simulation Package for Power Plants.* Artificial Intelligence and Simulation, The Diversity of the SCS Multiconference on Artificial Intelligence and Simulation, San Diego, pp. 55-58.

[4] Nishiuchi, N. (1988). *Concurrent Object Oriented Diagnostic System.* IEEE Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications, Japan, pp. 591-596.

[5] Pascoe, G. (1986). *Elements of Object Oriented Programming.* Byte, Vol. 11, No. 8, pp. 36-48.

[6] Tesler, L. (1986). *Object Oriented Languages Programming Experiences.* Byte, Vol. 11, No. 8, pp. 195-206.

[7] Bologna, S., Pisacane, F., Ghezzi, C. and Mandrioli, D. (1988). *An Environment for Requirements Specification and Analysis of Real Time Software Based on Timed Petri Nets.* Safety of Computer Control Systems, Proceedings of the IFAC Symposium, West Germany, pp. 7-10.

[8] Bruno, G. and Balsamo, A. (1986). *Petri Net-Based Object-Oriented Modelling of Distributed Systems.* SIGPLAN NOT. (USA), Vol. 21, No. 11, pp. 284-293. (OOPSCA'86. Object Oriented Programming Systems, Languages and Applications Conference Proceedings, Portland, USA).

[9] Christodoulakis, D. (1988). *Modelling the Semantics of Smalltalk-80 With Petri Nets.* SIGPLAN NOT, Vol. 24, No. 4, ACM SIGPLAN Workshop on Object Based Concurrent Programming, San Diego, USA, pp. 156-158.

[10] Mittelmann, R. (1990). *Object Oriented Implementation of Petri Nets Concepts.* Cybernetics and Systems'88 Proceedings of the 9th European Meeting on Cybernetics and Systems Research, Vienna, Austria, pp. 759-766.

[11] Cook, S. (1990). *The Object Oriented Programming and It's Applications.* Tutorials and Workshops on Applied Software Development OOPS (Object Oriented Programming Systems) Brunel Conference Centre.

[12] Anderson, B. (1987). *Object Oriented Programming.* Microprocessors and Microsystems, Vol. 12, No. 8, pp. 433-442.

[13] Cox, B. (1986). *Object Oriented Programming an Evaluationary Approach.* Addison-Wesley.

[14] Howard, G. (1988). *Object Oriented Programming an Evaluationary Approach.* Journal of Systems Management, Vol. 39, No. 7, pp. 13-19.

[15] Meyer, B. (1988). *Object Oriented Software Construction.* Prentice Hall International.

[16] Stefik, M. and Bobrow, D. (1984). *Object Oriented Programming: Themes and Variations.* AI Magazine, Vol. 6, No. 4, pp. 40-62.

[17] Sharif Heger, A., Patterson-Hine, F., Harringtton, R. and Koen, B. (1989). *Reliability Database Development for Use With an Object Oriented Fault Tree Evaluation Program.* Annual Reliability and Maintainability Symposium Proceedings, Atlanta, USA, pp. 283-287.

[18] Digitalk Inc. (1986). *Smalltalk/V Tutorial and Programming HandBook.* Digitalk Inc.

[19] Tylee, J. (1980). *Low Order Model of the Loss Of Fluid Test (LOFT) Reactor Plant for Use in Kalman Filter Based Optimal Simulation.* Proceedings of the 4th Power Plant Dynamics, Control and Testing Symposium, Gatlinburg, pp. 1-31.

[20] Tylee, J. (1980). *Low Order Model of the Loss Of Fluid Test (LOFT) Reactor Plant for Use in Kalman Filter Based Optimal Estimators.* EGG-2006, EG & G Idaho Falls, Idaho.