THE 4TH INTERNATIONAL TOPICAL MEETING ON
NUCLEAR THERMAL HYDRAULICS, OPERATIONS AND SAFETY
April 6-8, 1994, Taipei, Taiwan

AN INTELLIGENT ENVIRONMENT FOR DYNAMIC SIMULATION PROGRAM GENERATION OF NUCLEAR REACTOR SYSTEMS

Hiroaki Ishizaka, Akio Gofuku and Hidekazu Yoshikawa
Institute of Atomic Energy, Kyoto University
Gokasho, Uji-shi, Kyoto-fu 611, Japan
Fax : +81-774-33-3407

ABSTRACT

A graphical user interface system was developed for the two dynamic simulation systems based on modular programming methods:MSS and DSNP. The following works were made in conjunction with the system development: (1)conversion of the module libraries of both DSNP and MSS, (2)extension of DSNP-pre-compiler,(3)graphical interface for module integration, and (4)automatic converter of simple language descriptions for DSNP, where (1) and (2) were made on an engineering work station, while the rest (3) and (4), on Macintosh HyperCard™. By using the graphical interface, a user can specify the structure of a simulation model, geometrical data, initial values of variables, etc. only by handling modules as icon on the pallet fields. The use of extended DSNP pre-compiler then generates the final product of dynamic simulation program automatically. The capability and effectiveness of the system was confirmed by a sample simulation of PWR SBLOCA transient in PORV stuck open event.

I. INTRODUCTION

The characteristic features of dynamic simulations of nuclear reactor systems are large-scale scientific calculations and necessity of long-time contribution by many specialists such as nuclear engineering, mechanical engineering, computer science, and so on. These features are ascribed to the scale and complexity of nuclear reactor systems. Therefore, it is important to develop a flexible support environment to make a dynamic simulation program consistent with simulation purpose, to improve software productivity, and to avoid human errors in software development. Until now, there are many researches to improve software productivity and reliability of dynamic simulations of nuclear reactor systems by applying software engineering techniques, i.e. modular programming technique and object-oriented approach[1-8].

As far as the applications of modular programming in the field of nuclear engineering are concerned, D.Saphier, et al. have developed Dynamic Simulation for Nuclear Power plant (DSNP), and H.Yoshikawa, et al. developed Module-based Simulation System (MSS) mainly for the dynamic simulation of LMFBR. In the both systems, they aimed at improving software productivity by enhancing the re-usability of the software resources by the way that the commonly used subprograms are pre-programmed by easy re-usable form and stored in module library. On the other hand of modular programming, there have been some researches by employing object-oriented approach such as by D.J.Nypaver, et al.[7], M.A.Abdalla[8], C.A.JONE[9], etc, mainly noticing favorable feature of inheritance of attributes and information hide to cope with large-scale software complexity. Also one of the authors of this paper(Yoshikawa) applied object-oriented programming by Smalltalk-80 for the user support of MSS, to implement automatic diagnosis of the module integration and automatic input data generation[3].

Concerning the problems of modular programming methods employed in both of DSNP and MSS, there are following two problems when we proceed to rather large-scale complex simulation practice such as encountered in nuclear power plant simulation.

1. Even if language description employed in the both systems is simple , it is rather specific and hence it becomes rather difficult to build up consistent simulation scheme when the problem size becomes large, and

2. The reason for the above stems from the difficulty of acquiring necessary information on different modules to assemble into an integrated simulation model, because those information are distributed diversely among the different modules.

The authors' point in this paper is that they propose to introduce graphical interface (GIF) for building up simulation scheme by object-oriented approach. By the introduction of GIF, user can construct the simulation model by visual manipulation of modules which is the central part of human computer interaction in case of building models of large-scale complex system simulation. The authors have applied the above idea for expanding the functions of modular programming system DSNP to materialize an effective environment for dynamic simulation of nuclear power plant.

In this study, a GIF system is developed for the modular programming environment of DSNP, where the capability of DSNP system developed on IBM mainframe computer is expanded so that it may include the handling of module library of MSS and that it can work on UNIX EWS(Engineering Work Station) to share the benefit of its versatile functions. The converter is developed to convert automatically from user-specified simulation information by GIF to a DSNP description.

In the following, Section 2 describes the outline of DSNP and its transfer to an EWS. The GIF developed in this study is described in Section 3. An example of simulation program generation by GIF is discussed in Section 4. Finally, the

applicability of the GIF and future problems are summarized in concluding remarks.

II. DSNP:Dynamic Simulator of Nuclear Power Plants

A.Outline of DSNP[1]

The DSNP is a thermal-hydraulic modular simulation language dedicated to the simulation of nuclear power plant system or a part of the whole system at different levels of sophistication. The DSNP was originally developed at Argonne National Laboratory and is now supported by Soreq Nuclear Research Center, Israel.

The DSNP is composed of i) module libraries of three different levels of sophistication, ii) a function library for various material properties and various correlations of mass and heat transfer and other auxiliary functions, and iii) a pre-compiler. The libraries cover the necessary modules for the simulation of PWR, LMFBR, HTGR, and so on. The modules based on lumped parameter models are included in the library of level 1. The libraries of level 2 and 3 contain modules based on few nodes and 1/2D distribution models,respectively The pre-compiler is the central part of the DSNP system. The pre-compiler reads and interprets a simple simulation language described by a special format (DSNP description). Then, it generates automatically a FORTRAN program. The generated program is put into FORTRAN compiler and calculation is executed.

B . Conversion of DSNP to EWS and its Extension

The DSNP is converted on a UNIX EWS and several modifications are made. The plotting function of calculation results in original DSNP is made by the form of printer plots with any configuration of main frame computer. This plotting function is simply deleted because the calculation results can be easily plotted by using a conventional graphic software on EWS. Because FORTRAN compiler installed on EWS does not use JCL (Job Control Language) against common usage in main frame computer, the necessary files in execution is allocated by 'OPEN' statement instead of JCL specification.

There are two problems to use DSNP in various simulations of nuclear reactor systems which are related with the usage and development of a new module. There are two types of module connection: one is data coupling by common variables and the other by arguments. Although the data communication between modules is made by common variables in DSNP, the data communication by arguments as adopted by MSS is better from the view point of software engineering[10]. The first problem is therfore how to include modules of MSS library into DSNP system. The second one which comes from the inclusion of MSS modules is the time integration problem. In DSNP, time integration is made for a group of several DSNP modules by selecting either one of preset integration schemes such as trapezoidal rule, Milne's method, Gear's method, etc, while MSS modules have internal time integration scheme by themselves.Therefore, a MSS module mixed in a group of several DSNP modules will be time integrated twice if there is no special attention to DSNP pre-compiler. Against the both problems, the first one is solved by using 'CALL statement' for MSS modules and including the related argument lists in the main common blocks , because DSNP pre-compiler accepts FORTRAN statements. The second problem is avoided by using the special time integration scheme 'NOINT' for MSS modules, which specifies a dummy time integration scheme with no execution of time integration.

III . INTELLIGENT SIMULATION ENVIRONMENT FOR NUCLEAR POWER PLANT

A. Whole Architecture of Intelligent Simulation Environment

The authors have been developing an intelligent simulation environment for nuclear reactor systems by the combination of an EWS (HP apollo 9000 series) and a personal computer (Macintosh, Apple Computer Inc.). Figure 1 shows the whole elements of the environment: module library, extended DSNP pre-compiler, and two support sub-systems for suitable module selection and simulation model construction.

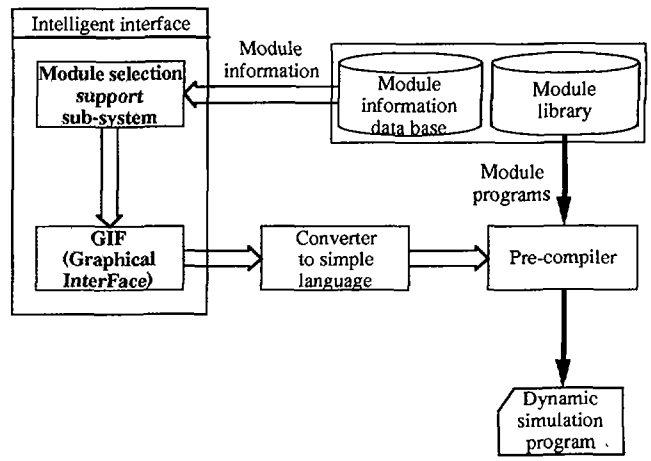

Figure 1 - Whole architecture of intelligent environment for dynamic simulation of nuclear reactor systems.

The suitable module selection sub-system supports the user to select suitable modules consistent with the user's simulation purpose and conditions. The simulation model construction sub-system is composed of a) a graphical interface, b) an automatic converter to DSNP description, and c) a simulation model proposer by applying case-based reasoning[11,12]. By the graphical interface, user can visually generate a simulation model. By the automatic converter, the information on the simulation model generated by GIF is converted to the format of DSNP description language, and then the extended DSNP pre-compiler interprets it and generate FORTRAN source program for the simulation model.

B. Graphical Interface

The graphical interface is the user interface by which user can

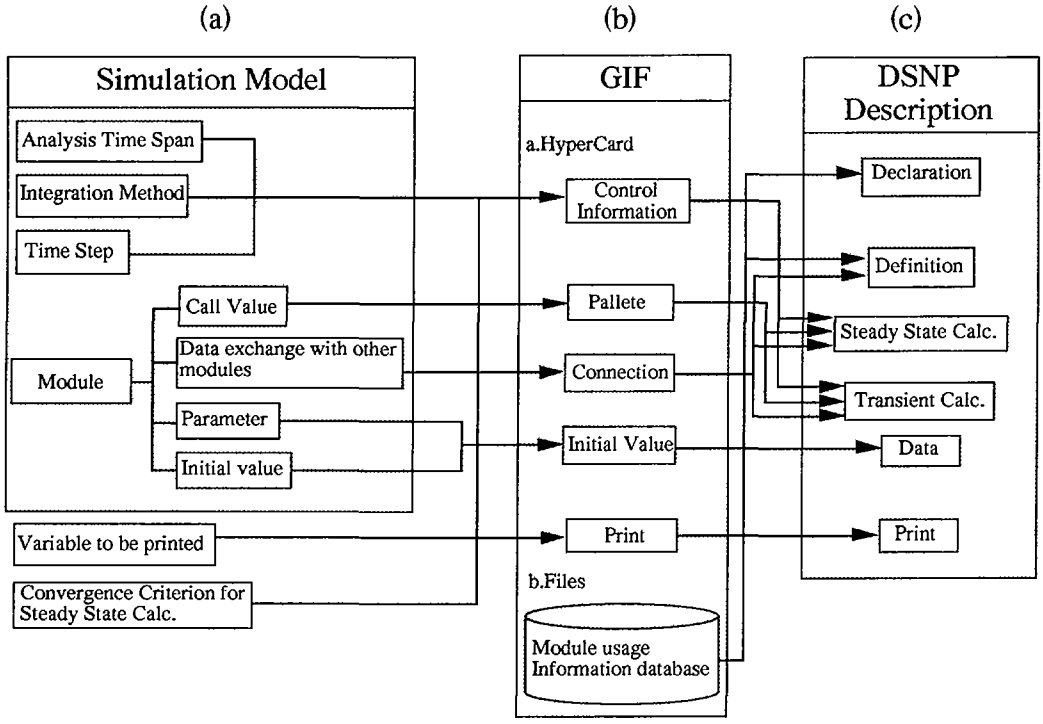

Figure 2- Data relationship between a dynamic simulation(a), its rearrangement by GIF(b) and the resultant structure of a DSNP description(c).

specify visually the structure of a simulation model, geometrical data, initial values of variables, etc on Macintosh. Figure 2 shows the relationship among simulation model,GIF and DSNP description language. In Figure 2, block (a) gives necessary information for DSNP-based simulation,while the block (c) the structure of DSNP description language. When a user would like to generate a simulation model, he will first imagine it rather macroscopiclly (macroscopic simulation model) how to arrange modules and what are interconnections between modules. Then, he goes on to specify detailed data such as setting parameters and initial values of modules, connecting variables between modules, time integration methods and print variables, etc. The block (b) shows how we rearrange the necessary simulation data by the form users think as mentioned above. Conclusively, Figure 2 shows how we designed graphical interface, in conjunction with the notion of "field" shown in Figure 3.

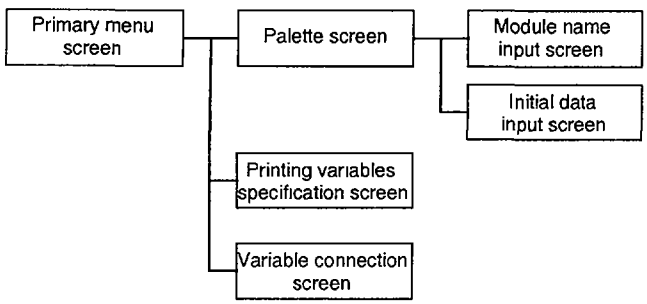

Figure 3 - Hierarchical structure of fields.

There are seven fields in the graphical interface. 1) primary menu field, 2) palette field, 3) module name input field, 4) variable connection field, 5) simulation administration information input field, 6) initial value input field, and 7) print variables specification field. Figure 3 shows the hierarchical structure of those fields. In addition to the fields, there are two data files (module usage information data base) for each module: i) argument list and ii) list of parameters and variables to set initial values.

Figure 4 shows an example of the variable connection field by which users will specify the connection of the selected two modules.

C. Automatic Converter to DSNP Description Language

The input simulation information by the graphical interface will be then automatically converted to DSNP description. In DSNP, a module has unique calling name and unique definition statements for parameters such as geometrical data, input/output variables of the module from other modules, etc. Moreover, there are special formats which will be used for the specification of integration methods, variable convergence condition for steady state calculation, initial values, and printing variables. The sequence is as follows: 1) template statements as shown in Figure 5 are prepared for module usage information data base beforehand for all modules and different integration methods, and so on, 2) the input from the graphical interface is arranged in the suitable field of the template statements, and after completing template statements, 3) re-ordering of template statement into the order of DSNP description language.

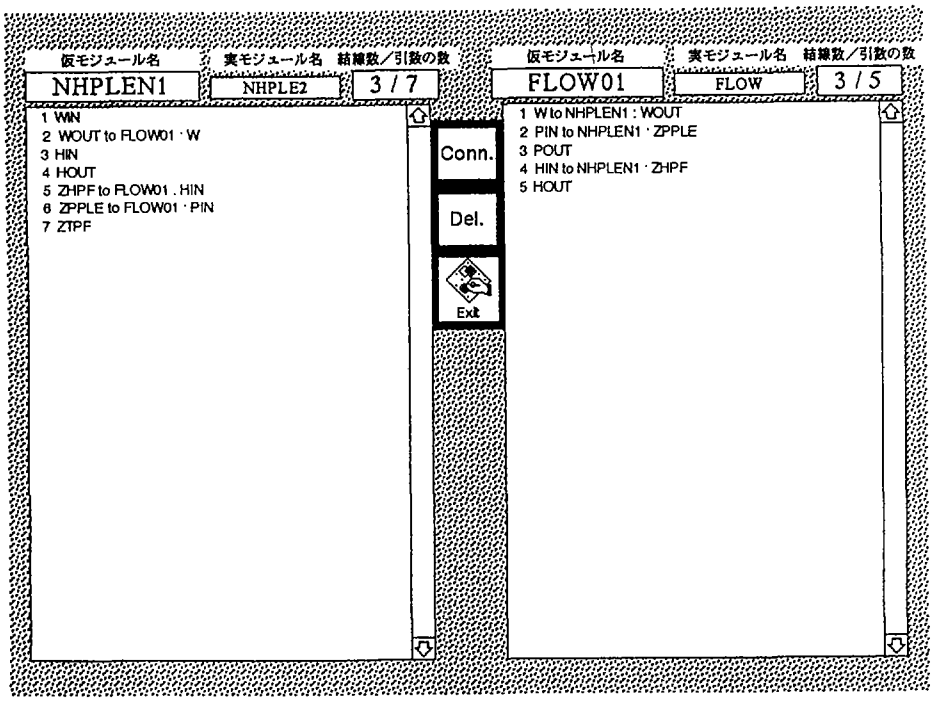

Figure 4 - Example of variable connection field.

```
. DFNHPLEN #N1 ( #C8 , #C9 , #C10 , #C11 , #C12 ,
.       INFLO( #V1 , #V3 ),EXFLO( #V3 , #V5 ),
.       0.2D0 , #C13 , #C14 , MAINP5);
```

Figure 5 - Examples of template statements for automatic conversion to DSNP description.

Ⅳ.EXAMPLE: PWR SBLOCA ACCIDENT SIMULATION

This section shows the results of a practice of simulation program generation by the graphical interface, automatic converter, and extended DSNP pre-compiler, followed by simulation run.

A simulation program is generated for a four-loop Westinghouse-type PWR primary thermal-hydraulics in the case of a small-break loss-of-coolant accident. Figure 6 shows the whole model of dynamic simulation. The major conditions and assumptions of the simulation are as follows :

1) the primary loops are represented by a lumped single loop,
2) the neutronics calculation is omitted and total heat power of fuel rods is given as a function of time,
3) the primary pipes are represented by the modules based on the homogeneous two-phase flow model,
4) only one PORV and one safety valve of pressurizer are modeled,
5) the bypass of reactor vessel is neglected,
6) the accident by PORV stuck open without scram is assumed to be calculated, and
7) only the safety functions of main coolant pump trip and turbine trip are considered.

By the graphical interface, the macroscopic simulation model is specified as shown in Fig. 7. Then, a user proceeds (1) to specify necessary connection of variables between modules, (2) to input geometrical data and initial values of variables, (3) to select printing variables, and (4) to specify calling order of modules, time integration method, etc. In the case where the functions that are not modeled by modules in the module library, user can add to the additional FORTRAN statements in the appropriate places of the converted DSNP description. A simulation program is generated by the extended DSNP pre-compiler after converting the simulation information to the corresponding DSNP description automatically. It takes only four minutes to generate simulation program of 8000 steps of FORTRAN statement for the whole work mentioned above. The simulation result of the dynamic behavior of the plant is depicted in Figure 8 ,with respect to time histories of 200 seconds into transient on pressure and water level in pressurizer, flowrate through PORV, coolant temperature and pressure at the bottom of reactor core, flowrate through SG safety valve, water level of SG secondary side and reactor power, respectively.

V. CONCLUDING REMARKS

In this paper, the authors presented on the development of our intelligent environment for dynamic simulation program generation of nuclear power plant. The salient feature of the authors' approach is the use of graphical user interface designed by the concept of object oriented approach with modular programming. The underlying concept was applied for the realization of an environment for the two simulation systems MSS and DSNP based on modular program. The effectiveness of the developed system was demonstrated by a simulation practice of PWR SBLOCA accident.

Currently, the authors have been developing various support

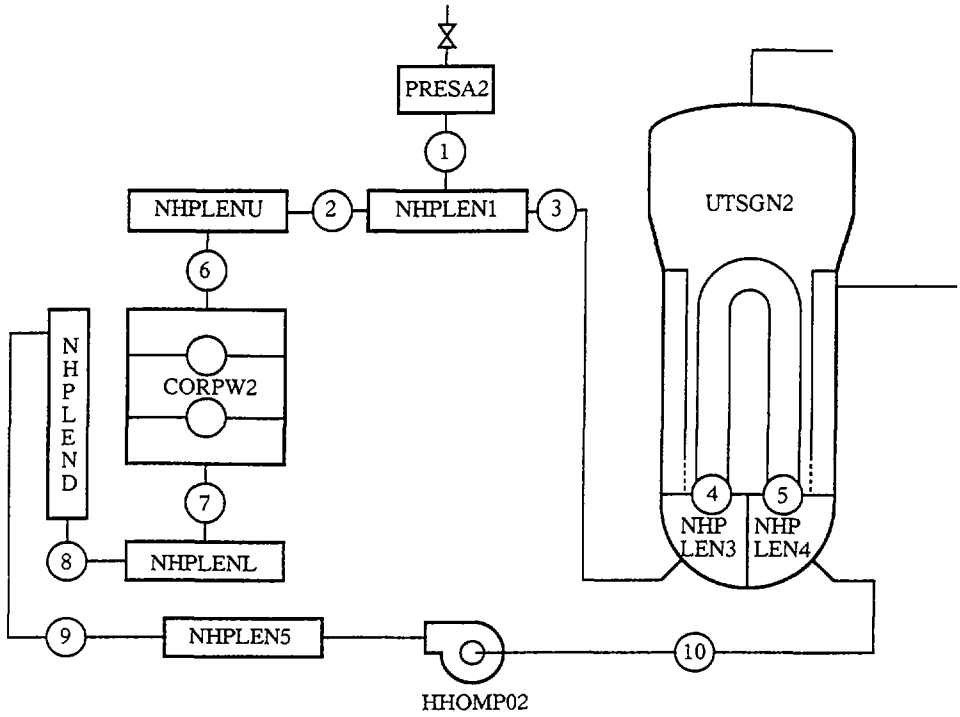

Figure 6 - Whole model of a dynamic simulation for a four-loop Westinghouse-type PWR primary thermal-hydraulics in the case of a small-break loss-of-coolant accident.

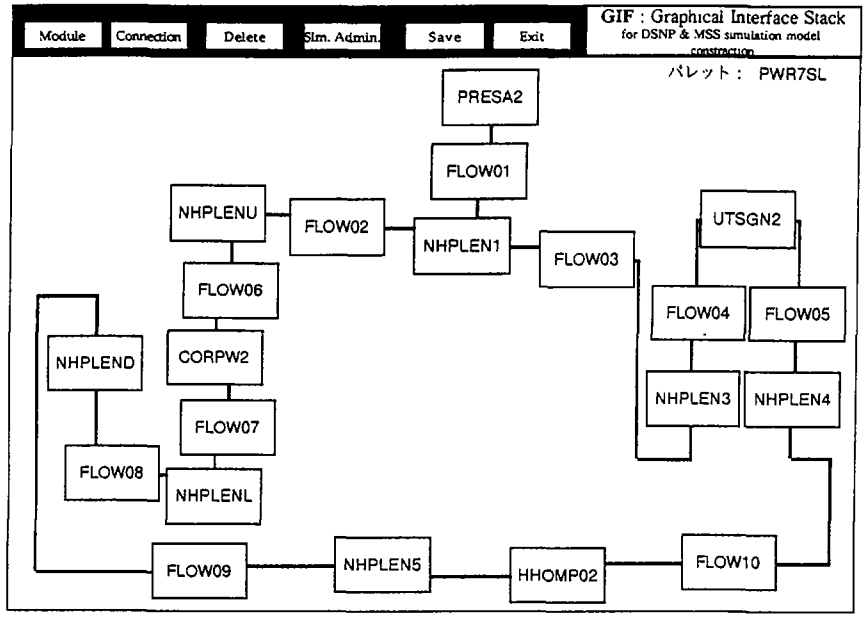

Figure 7 - Macroscopic simulation model specified by graphical interface.

systems on Macintosh mainly for the purpose of verifying the proposed object-oriented approach by rather rapid prototyping way. But it goes without saying that those user support systems should be re-integrated by the same way on EWS for the final stage of the system development.

We believe that with the use of GIF, the users can concentrate on the physical aspects of simulation models and can perform simulation practise more easily and rapidly than conventional programming methods. The proposed approach can be applied not only to the nuclear engineering domain but also to the other

engineering domains which require the dynamic simulation of the large scale and complex systems.

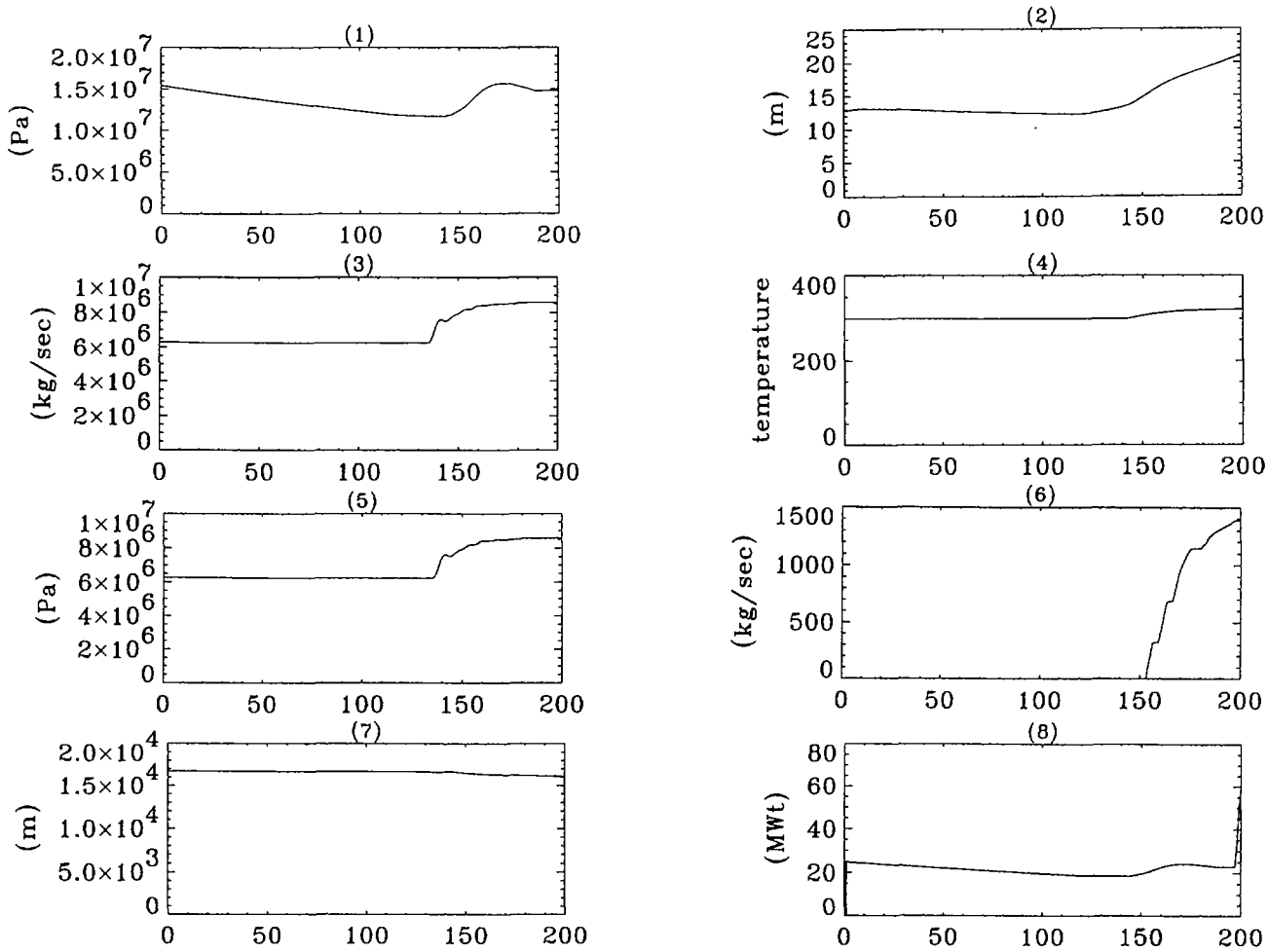

Figure 8 - Time histories of pressure(1) and water level(2) of pressurizer, PORV flowrate(3), temperature(4) and pressure(5) at the bottom of reactor core, flowrate(6) through SG safety valve, water level(7) of SG secondary side and reactor power(8) in case of a PWR SBLOCA simulation practise.

REFERENCES

1. D. Saphier, The DSNP Users' Manuals, Soreq Nuclear Research Center (1992).

2. H. Yoshikawa, et al., Proc. Int. Topical Mtg on Advances in Human Factors in Nuclear Power Systems, 311 - 319 (1986).

3. H. Yoshikawa, et al., J. of the Atomic Energy Society of Japan, Vol. 30, 699 - 713 (1988). (in Japanese).

4. A. Endo, et al., Proc. of ICHMT 2nd Int. Forum on Expert Systems and Computer Simulation in Energy Engineering, 13-3-1 - 13-3-6 (1992).

5. K.Gorai, Journal of Atomic Energy Society of Japan,Vol.32, No.1,49-55 (1990).

6. G. Oudot and A. Valembois, Proc. of Specialist Meeting on Simulations and Plant Analyzers, (1992).

7. D. J. Nypaver, et al., Proc. of 8th Power Plant Dynamics, Control & Testing Symp., Vol. 1, 12.01 - 12.12 (1992).

8. M. A. Abdalla, et al., ibid, Vol. 1, 13.01 - 13.06 (1992).

9. C. A. Jones, ibid, Vol. 1, 11.01 - 11.11 (1992).

10. Y. Kunitomo, Kokateki Puroguramu Kaihatsu Gihou (Efficient Program Development Techniques), Kindaikagaku-sha, (1983). (in Japanese)

11. J. L. Colodner, Proc. 4th Annual International Machine Learning Workshop, 167 - 178 (1987).

12. S. Kobayashi, et al., ICOT Technical Memorandum TM-0938, (1990). (in Japanese)