A STUDY OF DEVELOPING A PLANT DID RISK MONITOR FOR RESILIENT SEVERE ACCIDENT MANAGEMENT

Hidekazu Yoshikawa [1,2], Takashi Nakagawa [3],
Ming Yang[2],Hong Xia[2]

[1] Symbio Community Forum: C/o. Research Institute of Applied Sciences,
Tanaka Ohi-cho 49, Sakyo-ku, Kyoto, 606-8202 Japan
E-mail: yosikawa@kib.biglobe.ne.jp
[2] Harbin Engineering University:No.145 Nantong Street, Harbin, China, 150001
E-mail:{yangming,xiahong}@hrbeu.edu.cn
[3] Prime System Laboratory Co., Ltd.,Suimeidai 4-4-13, Kawanishi, 666-0116 Japan
E-mail:Nakagawa@prime-system.jp

A comprehensive risk analysis system has been under development to introduce a unique method for effective management of severe accident in nuclear power plant by resilient organization. In this study, the software system of the plant DiD risk monitor is developed for the plant personnel to design and evaluate the effective procedure by computer-assisted team learning. The developed system is validated by a case study for severe accident management in PWR.

I. INTRODUCTION

A comprehensive risk analysis system has been under development to introduce a unique method for effective management of severe accident in nuclear power plant by resilient organization.(Refs.1-5) The plant DiD risk monitor is an independent element of the whole risk analysis system, and it will be a useful tool for the plant personnel to design and evaluate plant emergency procedure by computer-assisted team learning In this paper, the details of the software methods developed for the plant DiD risk monitor will be first introduced, where also described are its objective, methods and how to describe complex man-machine interaction in the nuclear power plant. Then the result of applying the developed software will be introduced for validating it by a case study for severe accident management in PWR.

II. METHOD OF PLANT DID RISK MONITOR

The basic ideas to compose the plant DiD risk monitor are summarized as follows in the first place.

(i)The whole plant system is modelled by the combination of functional models for machine elements and human elements (operators, supervisors and other peoples who deal with the plant operation). These models are called as “actors” in this paper.

(ii)The dynamic behaviour of the whole plant system can be simulated by the interaction among these actors.

(iii)The individual actors have their own “scenario data” to behave in accordance with the given scenario. In order to create those scenario data easily and intuitively, the authors applied "State Chart Diagram" which has been extensively used in the field of systems engineering to model the behaviour of the computer systems.

(iv)The DiD risk monitor provides functions to investigate whether or not the plant situation is desirable, whether or not the mutual interaction between actors is suitable, and if inappropriate, what will be the causes of it.

The DiD risk monitor developed to realize the above idea consists of three subsystems: (i) Knowledge-base editor, (ii) Interaction simulator, and (iii) Interaction analyser. Knowledge-base editor provides editing functions to create scenario data in the form of the "State Chart Diagram". The Interaction simulator drives all actors based on their scenario data and simulates the whole plant situation as a result of these behaviours of the actors. The Interaction analyser shows how actors behave with one another and in what order in the form of "Sequence Diagram". The "Sequence diagram" is also widely used in the field of the system engineering to show the interactions such as event exchange and operation among system modules in time sequence.

These subsystems are developed as a plug-in of Integrated Development Environment "Eclipse"(Ref.6) with the use of Graphical Editing Framework "GEF"(Ref.7). Those software modules and the libraries only depend on Java, an object

oriented programing language which does not depend on any platforms, and therefore software system of DiD risk monitor can be installed on any Windows-PC or Macintosh-PC.

II.A Knowledge-base Editor

In order to express scenario date for each actor, the authors applied "State Chart Diagram". The "State Chart Diagram" is defined in Unified Modelling Language (UML) Ver.2.0 (Ref. 8). The UML has been succeeded in modelling the software modules until now in the software engineering, and especially, "State Chart Diagram" has been widely used to describe dynamic behaviours of software modules. By applying "State Chart Diagram" to model all knowledge-base information for the all kinds of actors, the following merits can be expected.

A. High capability to model dynamic behaviour: "State Chart Diagram" can model dynamic behaviours of the modelling target in different levels of abstraction (from abstract/outline to concrete/detailed). The model is easy to understand intuitively by users.

B. Simple modelling of interaction: "State Chart Diagrams" can model behaviours of the target by using states and transitions between the states and event handler, which causes the state transition. The interactions among actors are simply described by sending events among "State Chart Diagrams" and handling the event and making state transition.

Figure 1 shows a snapshot of the Knowledge-base editor. A canvas in the centre of the screen shows "State Chart Diagram" which is operator's knowledge to confirm a machine status. The users can drag and drop a state, a label and so on, by selecting it from the right side area named "Components" and dragging into the canvas. A “transition line” between the states can be drawn by Connection tool in the upper-right area named “Palette”.

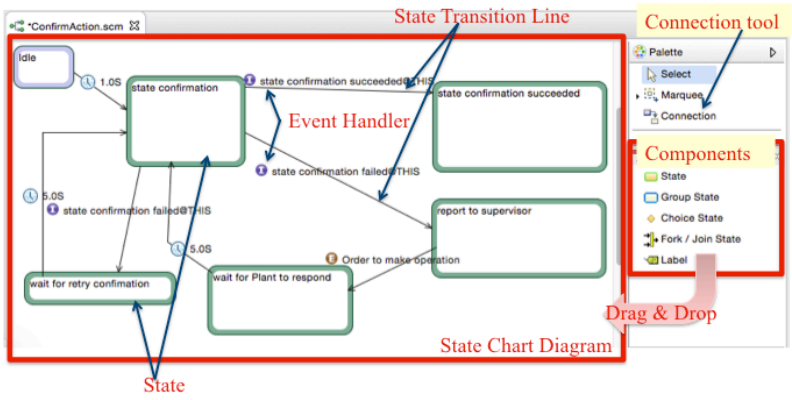

Fig.1 Snapshot of Knowledge-base editor

The role of transition line is to connect from a source state to the target state by an arrow line. It also holds several event handlers to make this state transition. When a certain event is received by the event handler, then the handler for this event makes the state transition and execute the command sequence which is defined as the action of this event handler. The users can write these command sequences in Java style program. In plant DiD Risk Monitor, the following 4 types of events and its handlers can be defined in "State Chart Diagram":

(A)Actor External Event: Actor External Events are transmitted among actors. For example, the plant actor sending an alarm as an "Actor External Event", other actors such as an operator and a supervisor, which have the corresponding event handler, receive and react the event of the occurred alarm. Therefore, the interaction among actors is simulated by sending and handling the "Actor External Events".

(B)Actor Internal Event: Actor Internal Events are used to communicate among the State Chart Diagrams within one actor.

(C)Primary Event: When states become active or inactive, the state generates a primary event such as "OnEntry" or "OnExit" on that time automatically. The corresponding event handler can be defined to execute some process in that timing.

(D)Timer Event: Timer event is generated after its pre-defined duration time.

The "State Chart Diagram" as shown in Fig.2 represents a task model of "Confirmation task" to confirm whether or not a certain machine state becomes a certain required status. Because operator's task in plant contains many confirmation tasks for various machines, these common tasks such as confirmation task should be modelled as software components and these components should be used repeatedly to make whole knowledge-base data efficiently.

To componentize task model in forms of "State Chart Diagram", target dependent information (*i.e.* machine name, desirable status and so on) should be separated from "State Chart Diagram" model. To achieve the requirement, our "State Chart Diagram" model has the parameters area to handle these target dependent information. The machine name and its desirable status are given as variables through the parameter area. So, our "State Chart Diagram" can be used as software component, its role seems like a subroutine of the computer programming.

All required basic tasks are provided in the form of components using the mechanism of the componentization. So general users can make scenario data easily and rapidly only by choosing the component and placing it on state. The componentization hides the detailed and complex information like a programming technique from the general user, and make the usage of the plant DiD Risk monitor easier.

The list of components corresponding to the basic tasks assumed in the present software is shown in Fig.2, and the example of creating the scenario data by using these components in Fig.2 is illustrated as shown in Fig. 3.

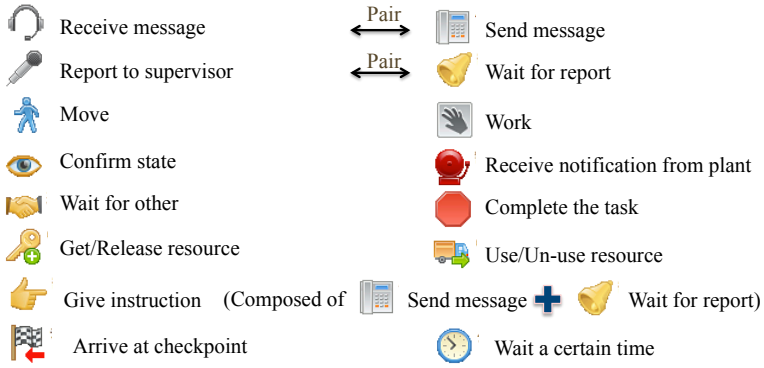

Fig.2 Component list for all basic tasks

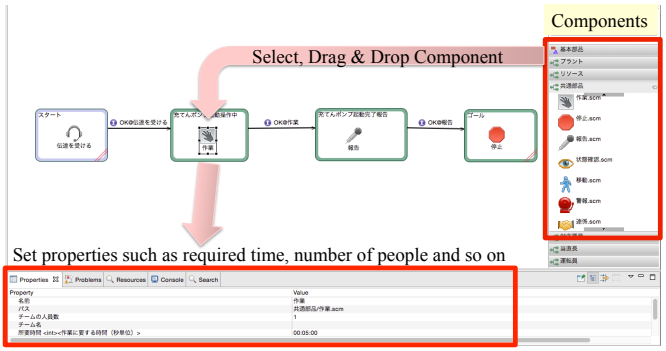

Fig.3 Example of creating a scenario data by using Components

In Fig. 3, the user places "State" on the canvas 4 times and connects them from left to right by transition lines. Selecting a suitable component from "Components" in the right side of the window, the user can drag and drop the selected component on the state respectively. Selecting the component on the state, the user can define required properties of the component such as required time and required number of members to complete the task modelled by the component.

Then the user should set event handlers for these transition lines. Figure 4 shows the direction of the setting event handlers. State-A holds a working component modelling working-task of the operator, the name of the task and required time to fulfil it are also set into the component as properties. All components for the basic task are developed to generate Actor Internal Event named "OK" when the task is completed. Therefore, the event handler for the Actor Internal Event should be

placed on the transition line from State-A to State-B. Because Actor Internal Event handler should be expressed as "EventName"@"Generate Place Name of the Event" in this system, the internal event handler named OK@OpenValve should be place on the transition line.

When the interaction simulator drives the "State Chart Diagram", completing the working-task at State-A, "State Chart Diagram" will make transition from State-A to State-B by the Actor Internal event handler named "OK@OpenValve" and start to do the moving-task placed on State-B.

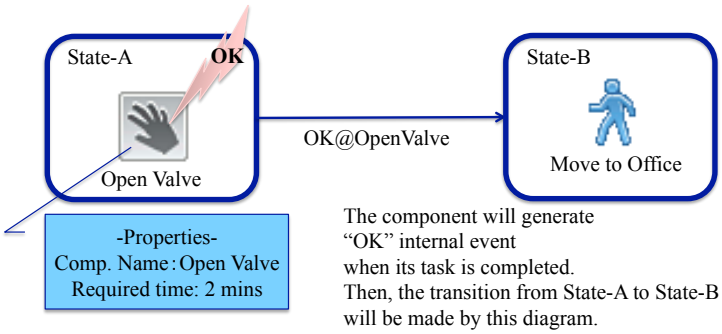

Fig.4 Directions of setting event handler

II.B Interaction Simulation

Once users converts all knowledge-base information for a given accident scenario into a set of "State Chart Diagrams", the users can execute the interaction simulation among actors by activating the interaction simulator. The execution time of each basic task will be affected by the following factors;

a. Component property for required time: Each component for the basic task has a variable for required time. The variable will be used to simulate the execution time to complete the task.

b. Traveling time: When people move to other place, the traveling time will be added, which is calculated by the distance and the way of getting there (by walk or by car).

c. Required number of people: Each "State Chart Diagram" needs required number of people to complete it. The interaction simulator tries to get the required number of people to execute the "State Chart Diagram". If it failed, the execution will be postponed until it gets enough people.

d. Waiting time for others: If the scenario needs someone's help or accomplishment of other task, a person needs to wait someone.

e. Required time to get resources: If people fail to get a certain resources such as cars, tools and materials to complete a task, he/she needs to wait until getting them.

II.C Interaction Analyzer

The results of the interaction simulation and the goal of the investigation by the interaction analyser are summarized as follows;

1) The result of the simulation is out of expectations: In the case that the simulation result is out of expectations or the simulation stops in the middle of the scenario, it is considered that the scenario data might have some errors or incorrectness. If these scenario data were entered by the actual operators or supervisors based on their understandings, their understandings might have errors or incorrectness. By correcting the problems in their understandings and the scenario data, then simulating them again, this iterative process will make their understandings for the emergency situation, their roles and the scenario deeper and wider.

2) Steps cannot complete in the limitation time: Some steps in an emergency procedure generally have limitation time to do them. If such a step is done over the limitation time by the interaction simulation, the investigation for the cause of the late should be conducted. The cause might be the potential problems of the scenario, lack of people and/or lack of resources.

3) The simulation result is not desirable: In the case that the simulation is executed on the given scenario, but the result is not desirable, for example, it comes core melt accident, radiological hazard as the result, the investigation should be conducted to find turning points which can exit the current scenario and the countermeasures should be considered to exit the scenario at the turning points.

To support these goals of the investigation, the interaction analyser shows the result of the interaction simulation as in form of the "Sequence Diagram". The "Sequence diagram" is also defined in UML and widely used in the field of the system engineering to show the interactions such as event exchange and operation among system modules in time sequence (Fig.5).

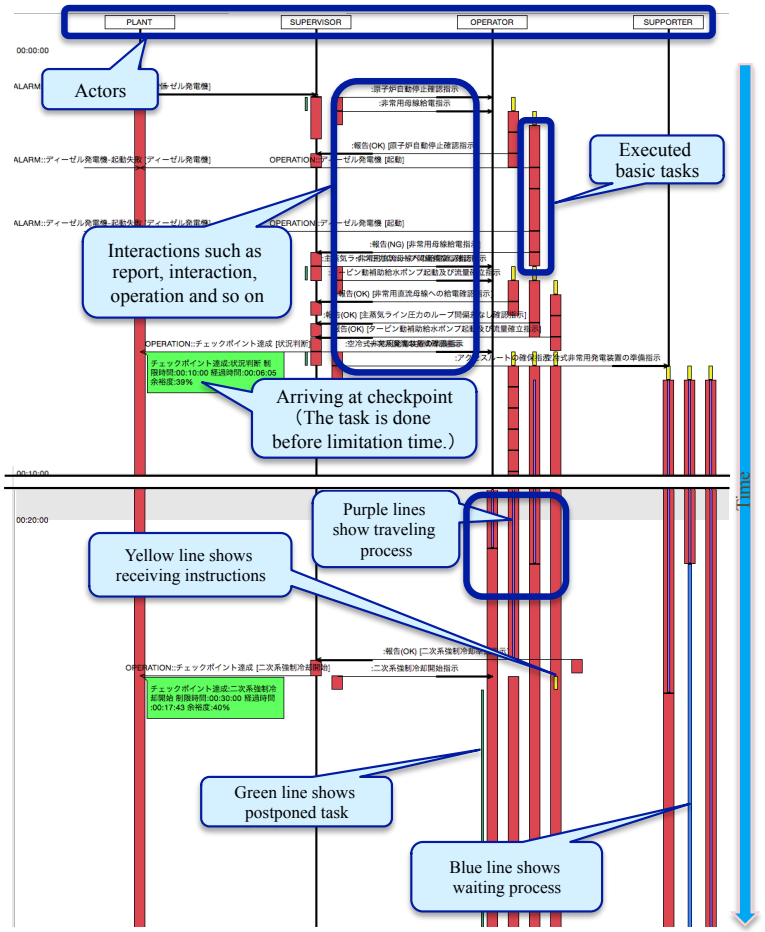

Fig.5 Example of Sequence Diagram

In Fig.5, all actors are laid out horizontally and the interaction among actors such as event exchange and the execution of the task are arranged vertically in time sequence. The arrow line between actors shows sending and receiving Actor External Events. Thin arrow lines are for sending the event. Thick arrow lines are for receiving and handling the event. One red box shows executions of one basic task. Plural red boxes arranged vertically shows the execution plural steps defined in a "State Chart Diagram". The green label is placed on the left side in Fig.5 means that a certain step can be executed before its time limit. Purple lines in red boxes shows the traveling process. Yellow lines mean receiving the instruction from other actors. After the receiving the instruction, required number of the people is tried to get. If it's succeeded, because the next task is executed immediately and the red box of the task is placed just below the yellow line. If it is not succeeded, a green line is drawn to show the task was postponed. Blue lines mean waiting time for others.

When user finds some problem such as long postponed task (green line), long waiting time (blue line), no reaction (no thick arrow line) and so on, he/she will do by right-clicking a line, a red box or an arrow line. Then the corresponding state or event handler of the "State Chart Diagram" is promptly displayed by the Knowledge-base editor and the user can check the scenario data to correct it rapidly.

III. CASE STUDY OF PWR STATION BLACKOUT ACCIDENT

The purpose of this case study is to confirm functionality and ability of the developed plant DiD risk monitor. The outline of the scenario is followings; after station blackout accident is occurred in conventional PWR plant in Japan, because emergency diesel generator is failed to start, people coping with the accident try to set up alternative generators and fire engine tracks to pump seawater in order to activate core-cooling functions, which is seriously considered in Japan to reflect the lessons from Fukushima Daiichi NPP accidents in March, 2011. The scenario is originally confronted by 2 supervisors, 8 operators and 17 supporters. This scenario contains following checkpoints and limitation times; judge the accident (10 mins.), start forced cooling of secondary system (30 mins.), supply electricity from alternative generator (1 hour), start alternative water injection into core by charging pump(2 hours 20 mins), hot shut down status(4 hours), able to supply seawater to aux feed water tank(11 hours), able to supply seawater to cv recirculation unit and high pressure injection system(51 hours). The case study was conducted by the following four steps, and the authors' intentions and the resultant observations are added for each step;

(1) Input scenario data in the form of "State Chart Diagram"

The total time needed to input this scenario data resulted in 1 - 2 hours. If the details of the procedure are clear, input work is rather simple and easy task because the required components for the basic tasks are already provided. The input work also made user's understandings for the procedure clear. Knowledge-base editor supported to find careless mistakes such as name mismatch between component and event handler, lack of event handler, lack of component and so on.

(2) Run through the whole scenario - analysis phase 1 -

The total time 1-2 hours was required to improve the scenario data until the user was able to run through the scenario from start of the accident until the end of the scenario. In this improvement phase, the user could find problems, in which the interaction had stopped unexpectedly from "Sequence Diagram", identify and correct the cause of them in "State Chart Diagram" easily and rapidly. Because the interaction simulation could run by 100 times faster in speed, the process of simulating, analysing and editing can be repeated many times.

(3) Focus on the limitation time and long waiting time - analysis phase 2-

During the process of finding the late execution of the checkpoint task, long postponed task, long waiting time and long traveling time from "Sequence Diagram", the user could investigate the causes from the "State Chart Diagram". Then the user could change the scenario data, assigned number of person and resources and confirm the result by running the interaction simulation. Through the analysis, the user could find potential problems of the scenario or better assignment of person /resources.

(4) Consider failure of machine and trouble in work - analysis phase 3-

The interaction simulation could conduct on the changed conditions such as a certain machine is failure and/or a trouble occurred in a certain work and it consumes long time to complete. Simulating and analysing the interactions under altered conditions could introduce more resilient procedures or person against the accident.

The result of this example study is shown in Table 1for the analysis of the above phase 2 and 3. In this table, case 1 is the original condition of how the emergency response team is composed: There are 2 supervisors located in the main control room, 8 operators located within the reactor building in which the main control room is included, and 17 supporters located outside of the reactor building. There seven checkpoints A, B, C, D, E, F, and G to see whether or not the emergency response team works successfully to finish each action until its specified time limit.

At this point, how the checkpoint A (Judge accident) is judged to reach in the interaction simulation is explained. The judgment of the supervisor as "loss of all AC power accident" will be made by getting the five reports from the operators: (i)confirmed automatic shutdown of nuclear reactor, (ii)tried several times to start emergency diesel generator but all failed, (iii)no discrepancies observed of main steam lines between loops, (iv)no problem in emergency DC batteries, and (v)no problem of feed water by turbine-driven auxiliary feed water pump. Just after this event diagnosis the supervisor and

operators will set to the works corresponding to this "loss of all AC power accident" asking the supporters such works as "maintaining access route by mending land slide and repairing road" and "starting up emergency power".

Table 1. Result of analysis for this case study

Checkpoint task	Time limit to finish the task (Hr.:Min.:Sec.)	Case 1 Supervisors: 2 Operators:8 Standard procedure	Case 2 Supervisors: 1 Operators:9 One supervisor is moved to operator	Case 3 Supervisors: 2 Operators:8 Procedure is changed with the original staff organization
A. Judge accident	0:10:00	0:06:10	0:07:29	0:06:07
B. Start forced cooling of 2ry system	0:30:00	0:18:04	0:19:43	0:18:00
C. Supply electric power from alternative generator	1:00:00	0:37:12	0:38:51	0:37:09
D. Start alternative water injection into core	2:20:00	2:44:24	2:00:17	1:38:11
E. Reach hot shutdown state	4:00:00	2:44:24	2.33:34	2:33:41
F. Able to supply seawater to aux. feed water tank	11:00:00	5:00:18	5:01:58	5:00:17
G. Able to supply seawater to CV recirculation unit	51:00:00	6:28:39	6:30:19	6:28:38

The checkpoint D is the critical time point to start alternative water injection into the reactor core by activating charging pump, and it should be finished within the time limit (2 hours 20 mins). But in the case 1 it was delayed until 2 hours 44 mins. This is the failed case. The reason of delay is because 7 operators had to move to the field and only one operator was left in the control room, although 8 operators had allocated in the main control room at the beginning of this scenario. Then the remaining one operator had to do with all the assigned tasks one by one so that the operator had to postpone the delayed task until completion of the previous tasks such as starting forced cooling of secondary system.

To avoid such problem, one supervisor is changed to an operator role in the case 2, where the simulation was conducted under the assignment of 1 supervisor and 9 operators and 17 supporters. In this case, all the checkpoints were performed before the time limit, but the initial checkpoint task A of judging the accident was a bit delayed to be compared with the case 1. The reason is because there are so many tasks have to be done by one supervisor immediately after the accident so that the completion to judge the accident should be delayed.

In the case 3, the procedure was modified so that the task of starting alternative water injection into core by charging pump was assigned to the supervisor, although all the other assignment conditions are the same as the case 1. The result of the case 3 shows all checkpoints are successfully conducted before the time limit and these are no delays to be compared with the other cases.

To sum up, the investigations was conducted easily and rapidly by the plant DiD risk monitor. The process for the investigation is very effective to understand the procedure, and to find proper person assignment to overcome potential problems among actors.

IV. CONCLUSION

The progress of the author's developmental study on a new risk monitor system was introduced, which can be applied not only to severe accident prevention in daily operation but also to serve as to mitigate the radiological hazard just after severe accident happens and long term management of post-severe accident consequences. Then, the fundamental method was summarized on how to configure "Knowledge-base data" in the plant DiD risk monitor by "State Chart Diagram" and how to describe the interaction among actors by "Sequence Diagram", these diagrams are extensively used in the field of systems engineering.

In this paper, the authors show that by applying the componentize mechanism of the "State Chart Diagram", whole knowledge-base information essential to simulate human-machine interactions can be modeled easily and efficiently. And the "Sequence Diagram" can describe the interactions among actors and express the problems of the interactions well. The investigation method for the integrations is also proposed by searching the problems on the "Sequence Diagram", finding and correcting the cause of the problems from "State Chart Diagram" and conducting the interaction simulation for the corrected data iteratively. The authors confirmed the method could be done easily and rapidly, and the process for the investigation was very effective to understand the procedure, person assignment, and potential problems among actors.

As the future work of the authors of this paper, the plant DiD risk monitor will apply to other case studies in order to improve total usability for all the process of creating data, conducting interaction simulation and investigating the interactions.

REFERENCES

1. Yoshikawa, H., *et al.,,Nuclear Safety and Simulation*, **3**, 2, 140(2012).
2. Yoshikawa, H., *et al*. *Nuclear Safety and Simulation,* **4**, 3, 192 (2013)..
3. Yoshikawa, H., *et al.*, "Integrated functional modeling method for configuring NPP plant DiD risk monitor and its application for AP1000, (ICONE22-30987) ", *Proc. the 22nd International Conference on Nuclear Engineering (ICONE22),* Prague, Czech, July 7-11, 2014,ASME.
4. Yoshikawa, H., *et al,* "Integrated functional modeling method for configuring NPP plant DiD risk monitor and its application for conventional PWR", *Proc. ISOFIC/ISSNP2014*, Jeju Island, Korea, Aug.24-26, 2014, KNS.
5. Yoshikawa, H. and Nakagawa, T, "Software system development of NPP plant DiD risk monitor -basic design of software configuration-, (ICONE23-1312)", *Proc. the 23th International Conference of Nuclear Engineering (ICONE23*), Chiba, Japan, May 17-21, 2015,ASME.
6. Eclipse foundation. Eclipse IDE for Java Developer Version: Luna (4.4.1) http://www.eclipse.org, (As of 2014)
7. Eclipse foundation. GEF(Graphical Editing Framework) Release 3.9.101, https:// eclipse.org/gef,, (As of 2014)
8. Object Management Group. UML Version 2.4 Specification, http://www.omg.org/spec/UML/2.4 (As of 2011),