Branching Rules and Quantification in Dynamic Probabilistic Safety Assessment : Development of DICE (Dynamic Integrated Consequence Evaluation)

Sejin Baek[a], Taewan Kim[b], Jonghyun Kim[c], Gyunyoung Heo[a*]
[a]*Kyung Hee University, 1732 Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do, Republic of Korea, 17104*
[b]*Inchoen National University, 119, Academy-ro, Yeonsu-gu, Incheon, Republic of Korea, 22012*
[c]*Chosun University, 309, Pilmun-daero, Dong-gu, Gwangju, Republic of Korea, 61452*
[*]*Corresponding author: gheo@khu.ac.kr*

1. Introduction

The conventional PSA (Probabilistic Safety Assessment) methodology pre-specifies branching conditions and timing associated with automatic or manual tasks in an event tree. This has substantially contributed to improve the safety of NPPs (Nuclear Power Plants) by pursuing ease of calculation and securing conservative calculation results.

In the previous study, however, the potential shortcomings of the conventional PSA were suggested and D-PSA (Dynamic Probabilistic Safety Assessment) methodology that could supplement them was discussed [1, 2, 3].

The main characteristic of D-PSA methodology is that the event tree and plant physical model (e.g. thermo-hydraulic safety analysis code) are built up interactively. Therefore, the issue of task allocation between the PSA model and the plant physical model is important, and the scheduler module is responsible for this [4]. The PSA model also needs to be modified such that they can be separated with automatic tasks, manual tasks, and equipment malfunction including signal failures. In order to dynamically branch an event scenario in conjunction with a plant physical model, it is necessary to define branching rules by automatic or manual tasks. In each branch, the methodology for appropriately quantifying the branching probability is also required.

As a part of the research project in developing the D-PSA supporting tool (called DICE: Dynamic Integrated Consequence Evaluation), this paper describes how to branch event trees in a real-time simulation environment using the plant physical module and how to quantify each branch. In addition, data structure and algorithm for implementing branching rule and quantification methodology are presented.

Since the overview of D-PSA has already been presented in previous studies, the structure of DICE and the concept of DDET (Dynamic Discrete Event Tree) will be described only to the extent necessary for the context of this paper [4, 5].

Branching rules and quantification are defined differently according to the purpose of D-PSA. In this study, the necessary algorithm is set such that the coverage of the operational procedures can be evaluated.

2. Methods and Results

2.1 Structure of DICE

DICE has been designed on the basis of the structures and functions of other D-PSA tools. For instance, a DDET is used for scheduling, the fault tree of the conventional PSA model is applied to the equipment module. The equipment module is driven by an auto task module and a manual task module. In particular, the manual task module copes with an HRA (Human Reliability Analysis) model that reflects the decision algorithms of operator's action is taken.

The main module to support the structure of DDET has been already presented in the previous study [3]. The detailed structure is somewhat improved as shown in Fig. 1.

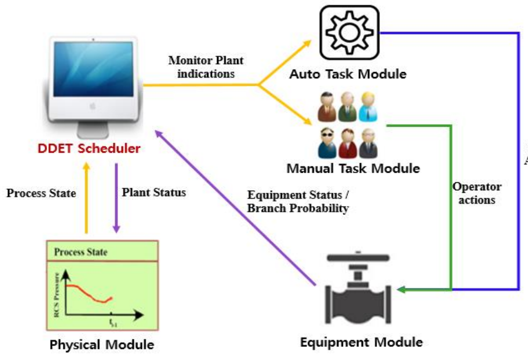


Fig. 1. Schematic diagram of DDET and dynamic interactions in NPPs

The scheduler runs based on a DDET and calls the physical and equipment module at the specified time interval to enable dynamic interworking. In other words, the scheduler acquits value of the plant variables from the physical module and distributes it to automatic and manual task module. And when if there are changes of equipment status in the equipment module due to the task modules, this information is sent back to the scheduler to change the plant status of physical module. This process continues until the simulation for all branches are completed.

2.2 Mechanism of DDET

When an initiating event starts, the scheduler calls the physical and equipment modules for a time step, dt. If a particular branching condition for (1) automatic task and (2) manual task is satisfied, the equipment module calculates how many branches to make and what information to hold for each branch as shown in Fig. 2.

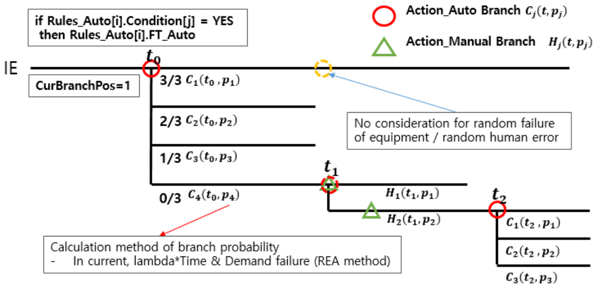

if Rules_Auto[i].Condition[j] = YES
then Rules_Auto[i].FT_Auto
Action_Auto Branch $C_j(t,p_j)$
Action_Manual Branch $H_j(t,p_j)$
IE
t_0
CurBranchPos=1
3/3 $C_1(t_0,p_1)$
2/3 $C_2(t_0,p_2)$
1/3 $C_3(t_0,p_3)$
0/3 $C_4(t_0,p_4)$
No consideration for random failure of equipment / random human error
t_1
$H_1(t_1,p_1)$
$H_2(t_1,p_2)$
t_2
$C_1(t_2,p_1)$
$C_2(t_2,p_2)$
$C_3(t_2,p_3)$
Calculation method of branch probability
- In current, lambda*Time & Demand failure (REA method)

Fig. 2. Execution Process of DICE

The information held by each branch includes (1) branching time, (2) branch probability, (3) equipment status which shows availability, (4) actuation status to operate components such as pump on/off, valve open/close/modulate, (5) information needed to properly operate the physical module. Except for (5), which is dependent on the physical module, other data structures consisted of the information required by the general DDET regardless of type of the physical module.

The scheduler contains information that can reconstruct a sequence of scenarios according to the combination of all branches. Furthermore, it can also identify the trends of variables calculated by the physical module for each scenario. Note that the DICE scheduler is implemented in a distributed computing environment with main/client server.

2.3 Branching Rules

While the scheduler runs a physical module, branching occurs when the plant variables meet a condition set by the user, and the conditions that apply are called branching rules. Since the main purpose of this study is to identify the coverage of the operator procedures, DICE applied branching rules to automatic and manual tasks to simulate the direct impact of the operator procedures. The automatic task means that the system is operated automatically performing the role at the setpoint already set in advance, while manual task is only operated by direction and execution of the operator.

The automatic task adopted by DICE are RPS (Reactor Protection System) and ESFs (Engineered Safety Features). The event tree in the conventional PSA is divided by predetermined success criteria, and the branch probability is calculated according to whether the success criteria is satisfied. However, D-PSA doesn't take into account the success criteria but reflects all possible combinations of safety system on a train scale. It also branches separately even if the trains of the safety system are not symmetrical as shown in Fig. 3.

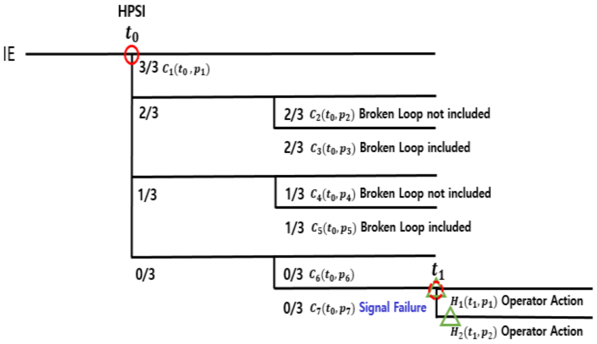

HPSI
t_0
IE
3/3 $C_1(t_0,p_1)$
2/3
2/3 $C_2(t_0,p_2)$ Broken Loop not included
2/3 $C_3(t_0,p_3)$ Broken Loop included
1/3
1/3 $C_4(t_0,p_4)$ Broken Loop not included
1/3 $C_5(t_0,p_5)$ Broken Loop included
0/3
0/3 $C_6(t_0,p_6)$
t_1
0/3 $C_7(t_0,p_7)$ Signal Failure
$H_1(t_1,p_1)$ Operator Action
$H_2(t_1,p_2)$ Operator Action

Fig 3. Branching according to the HPSI branching rule in case of not symmetrical trains

For example, if there is a HPSIS (High Pressure Safety Injection System) which has three injection trains, it can be split from a branch that are not injected into every trains to a branch that are injected into every trains. In addition, even if each train is injected, if a pressurizer is installed or a specific train pipe is broken, the branch is divided separately. In each branch, it is also important to distinguish between signal and mechanical failure for next manual tasks.

Once a branching rule is met, the combination of failure status of the equipment module and the operational status for the plant physical module can be determined by the equipment module, and transferred to the physical module to perform simulations for each branch. In this manner, the equipment module of DICE can control all of the operational status of the physical module while the plant physical module, in this study, MARS-KS does not have any actions at all.

This study assumes that all manual tasks are performed based on the operator procedures. Thus, the branching rules are set to error of omission or commission that stand for the actions are taken or not at a particular time. The execution of a single action is also discretely branched, and considered to be 'action executed' or 'action not executed' at any branching point as shown in Fig. 4.

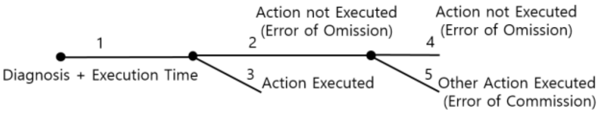


Fig 4. An example of branching for manual task

The current manual tasks reflect the operators' recovering the signal or manually operating the injection for the case where all the trains of the automatic task branch described above are not injected due to the failure of the signal. In addition, it covers a manual control of the flow or the operation of specific valves and pumps. The branching time in the manual task is calculated by setting the diagnostic time and execution time as one action in the operator crew module that supporting the manual task module, and is defined as a time that actually affects the operation of the equipment. However, an error of commission that is not based on the operator procedure is not taken into account due to the number of combinations are too large as same with the random failure of the equipment, which will be reflected in certain time for specific purpose.

2.4 Quantification for Branches

Branches occur in both automatic and manual tasks, but the calculations of probabilities for each branch are done in completely different ways. This paper only deals with a method for automatic tasks, and manual tasks will be covered in a separate paper [5]. The advantage of D-PSA is that it is possible to represent all the different possibilities in the branching conditions and to identify their sequences and impacts. However, there is also a downside to this which could result in the explosion of computations.

Even with a simple system, implementing the PSA model through fault trees results in a large number of cutsets. While previous paper has suggested the reasonable ways to minimize these cutsets, the quantification method has been improved as described in this paper [6]. This section describes the resources to be prepared in advance in the equipment module before DICE runs, and the detailed linkage algorithm for each resource is discussed in section 3.2.

Fig. 5 shows an example of fault tree of each branch for HPSIS. Quantifying each top event of the fault trees yields in a number of cutsets depending on the failure combination of the equipment, which becomes reference cutsets. The reference cutsets of each branch which can be generated in every system should be set in advance so that could be loaded with DICE runs (KooN).

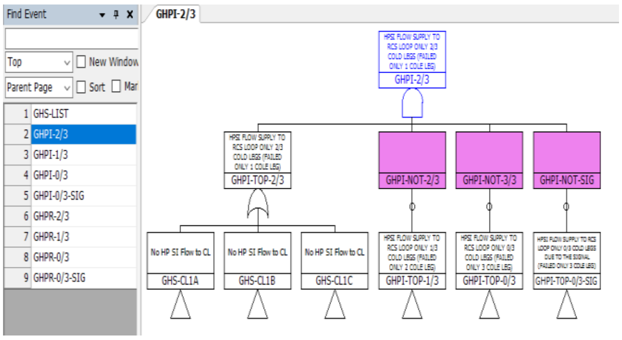

Find Event
GHPI-2/3
Top
□ New Window
Parent Page
□ Sort
□ Mar
1 GHS-LIST
2 GHPI-2/3
3 GHPI-1/3
4 GHPI-0/3
5 GHPI-0/3-SIG
6 GHPR-2/3
7 GHPR-1/3
8 GHPR-0/3
9 GHPR-0/3-SIG
HPSI FLOW SUPPLY TO RCS LOOP ONLY 2/3 COLD LEGS (FAILED ONLY 1 COLE LEG)
GHPI-2/3
HPSI FLOW SUPPLY TO RCS LOOP ONLY 2/3 COLD LEGS (FAILED ONLY 1 COLE LEG)
GHPI-TOP-2/3
GHPI-NOT-2/3
GHPI-NOT-3/3
GHPI-NOT-SIG
No HP SI Flow to CL
GHS-CL1A
No HP SI Flow to CL
GHS-CL1B
No HP SI Flow to CL
GHS-CL1C
HPSI FLOW SUPPLY TO RCS LOOP ONLY 1/3 COLD LEGS (FAILED ONLY 2 COLE LEG)
GHPI-TOP-1/3
HPSI FLOW SUPPLY TO RCS LOOP ONLY 0/3 COLD LEGS (FAILED ONLY 3 COLE LEG)
GHPI-TOP-0/3
HPSI FLOW SUPPLY TO RCS LOOP ONLY 0/3 COLD LEGS DUE TO THE SIGNAL (FAILED ONLY 3 COLE LEG)
GHPI-TOP-0/3-SIG

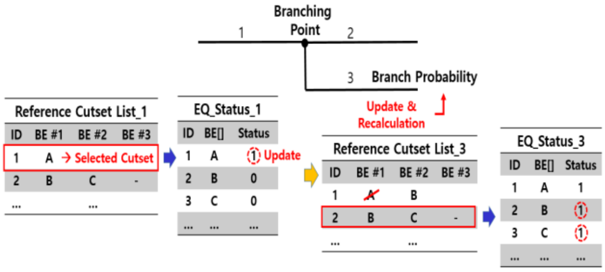
Fig 5. Fault tree for 2 out of 3 HPSIS injection branches

Since a top event only represents the one mode of the plant, the plant physical module is irrelevant to implement the plant model along with the number of cutsets. However, the quantification of branch probability in the equipment module requires all cutsets because it is worked out by summating each cutset vale.

Therefore, each branch should (1) reflect failure status of the equipment module and plant operation status of the plant physical module considering the branches previous performed, and (2) determine which equipment has failed in current branch to quantify the branch probability. In the case of (1), it can be carried out by listing the basic events in the fault trees and updating the failure status whenever the branches are made (e.g. using an array named as EQ_Status), and the other case (2) can be conducted by selecting a specific cutset as following methods.

1) Select a specific cutset for each branch's reference cutset before starting DICE
2) The user selects the cutset at each branching point
3) The cutset is randomly selected if no user is involved

Fig. 6 shows the quantification process in each branching point. If a cutset is selected, the components included in this cutset are updated with a failure in EQ_Status, and are excluded from the reference cutset of next branch. This means that the reference cutset is also updated according to EQ_Stauts. Finally, the summation of each cutset value in the updated reference cutset is calculated as the branch probability [7].



Reference Cutset List_1

ID	BE #1	BE #2	BE #3
1	A		
2	B	C	-
...		...	

EQ_Status_1

ID	BE[]	Status
1	A	1
2	B	0
3	C	0
...	...	...

Reference Cutset List_3

ID	BE #1	BE #2	BE #3
1	~~A~~	B	
2	B	C	-
...		...	

EQ_Status_3

ID	BE[]	Status
1	A	1
2	B	1
3	C	1
...	...	...

Fig 6. Quantification process in each branching point

3. Implementation

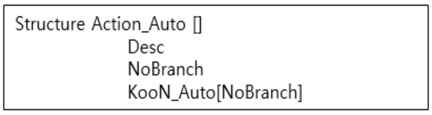
3.1 Data Structure

This section introduces the detailed data structures for branching in DICE. The structure 'Rules_Auto' means the branching rule data structure for automatic task as shown in Fig. 7. The array, 'Condi[NoCondition]' stands for the logic to compare the plant variables from the plant physical module with the set values. The 'Running' checks whether the branching rule has occurred before and prevents re-branching or simultaneously branching.

```
Structure Rules_Auto []
                Desc
                NoCondition
                Condi[NoCondition]
                Running
                ActionID_Auto
```

Fig 7. Branching rule data structure for automatic task

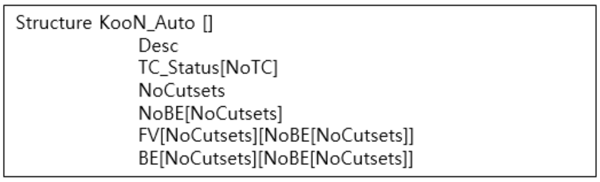
When the structure 'Rules_Auto' meets a specific branching rule, the 'Action_Auto' data structure determines which branch information to load as shown in Fig. 8.



```
Structure Action_Auto []
                Desc
                NoBranch
                KooN_Auto[NoBranch]
```

Fig 8. Intermediation data structure between branching rule and branch information

The branch information allocated by the ‘Action_Auto’ is stored in the 'KooN_Auto’ data structure shown in Fig. 9. It has the ‘TC_Status’ to assign trip card operational status of the plant physical module and 'NoCutsets’ that supports the calculation of branch probability with reference cutset information including the Fussell-Vesely importance and the basic events for each branch.



```
Structure KooN_Auto []
                Desc
                TC_Status[NoTC]
                NoCutsets
                NoBE[NoCutsets]
                FV[NoCutsets][NoBE[NoCutsets]]
                BE[NoCutsets][NoBE[NoCutsets]]
```

Fig 9. Data structure for the information of the reference cutset and operational status of the plant physical module for each branch

3.2 Algorithm

Each branch generated has information as described in section 2.2 and will be stored in a structure, ‘Branch_Split’ The pseudo-code that constitutes the algorithm of the equipment module and the automatic task module is shown in Fig. 10. The function, 'Main_A’ is responsible for diagnosing branching rules and preventing reoccurrence of it. When if the execution of the action is confirmed thorough the diagnosis, the branching rule is changed to the active state, and the following actions are taken from a function, ‘Action_A’ to return the structure, ‘Branch_Split’ for each branch to the Scheduler.

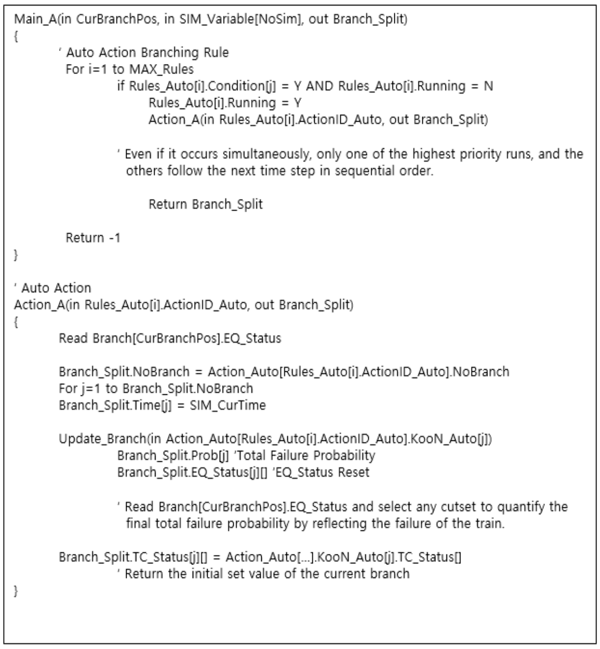
4. Conclusions

In this study, a branching rule and a branch probability quantification methodology to be applied to the DDET are presented, and the data structures and algorithms that can be actually coded are described. Branching rules and quantification can be customized according to the purpose of the study.

DICE is supposed to be released as its first draft by the end of 2019, and case studies will be performed in 2020 with the revision particularly for post-processing part.

ACKNOWLEDGEMENT

This work was supported by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety(KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission(NSSC) of the Republic of Korea. (No. 1803008)



```
Main_A(in CurBranchPos, in SIM_Variable[NoSim], out Branch_Split)
{
        ‘ Auto Action Branching Rule
         For i=1 to MAX_Rules
                  if Rules_Auto[i].Condition[j] = Y AND Rules_Auto[i].Running = N
                        Rules_Auto[i].Running = Y
                        Action_A(in Rules_Auto[i].ActionID_Auto, out Branch_Split)

                  ‘ Even if it occurs simultaneously, only one of the highest priority runs, and the
                    others follow the next time step in sequential order.

                        Return Branch_Split

         Return -1
}

‘ Auto Action
Action_A(in Rules_Auto[i].ActionID_Auto, out Branch_Split)
{
        Read Branch[CurBranchPos].EQ_Status

        Branch_Split.NoBranch = Action_Auto[Rules_Auto[i].ActionID_Auto].NoBranch
        For j=1 to Branch_Split.NoBranch
        Branch_Split.Time[j] = SIM_CurTime

        Update_Branch(in Action_Auto[Rules_Auto[i].ActionID_Auto].KooN_Auto[j])
                  Branch_Split.Prob[j] ‘Total Failure Probability
                  Branch_Split.EQ_Status[j][] ‘EQ_Status Reset

                  ‘ Read Branch[CurBranchPos].EQ_Status and select any cutset to quantify the
                    final total failure probability by reflecting the failure of the train.

        Branch_Split.TC_Status[j][] = Action_Auto[...].KooN_Auto[j].TC_Status[]
                  ‘ Return the initial set value of the current branch
}
```

Fig 10. Pseudo-code and algorithm for the equipment module

REFERENCES

[1] Hansul Lee, Hyeonmin Kim, Taewan Kim, Gyunyoung Heo, Survey of Dynamic PSA Methodologies, Korean Nuclear Society Spring Meeting, Jeju, 2015

[2] Kyunghee university, Application of Dynamic PSA for Multi-unit Risk Evaluation, NSTAR-18N811-20, 2018

[3] Hansul Lee, Taewan Kim, Gyunyoung Heo, Application of Dynamic PSA Approach for Accident Sequence Precursor Analysis: Case Study for Steam Generator Tube Rupture, International Association of PSAM, Seoul, 2016

[4] Sangwha Lee, Sejin Baek, Taewan Kim, Jonghyun Kim, Gyunyoung Heo, Development of DICE(Dynamic Integrated Consequence Evaluation) for Procedure Coverability Studies: Conceptual Design Phase, Korean Nuclear Society Autumn Meeting, Yeosu, 2018

[5] Gyeongmin Yoon, Daeil Lee, Jonghyun Kim, Feasibility Study on the Application of Artificial Operator to Estimation of HEPs, Asian Symposium on Risk Assessment and Management 2019, Gyeongju, 2019

[6] Sejin Baek, Taewan Kim, Jonghyun Kim, Gyunyoung Heo, Introduction to DICE (Dynamic Integrated Consequence Evaluation) Toolbox for Checking Coverability of Operational Procedures in NPPs, Proceedings of the 29th European Safety and Reliability Conference, Hannover, 2019

[7] J. Collet, Some remarks on rare-event approximation, IEEE Transactions on Reliability, 45(1), 106-108, 1996