Aalto University

Development of a probabilistic risk analysis example using a computer tool for dynamic event tree modeling

Seminar on Case Studies in Operations Research MS-E2177

Lauri Nyman (Project manager) and Sakke Rantala
25.5.2016

Content

1 Introduction

Risk is a particular unwanted outcome with some probability and magnitude of harm. Risk analysis is a structured methodology to specify such hazards and identifying and qualifying their initiating events or sequences of them. Its high-level aim is to reduce both probability and effects of the unwanted outcomes. Risk analysis provides qualitative – and often quantitative – information for decision makers. It does not answer to the question what level of risk is reasonable or worth of cost. On the other hand it can try to answer to the question of how much risk costs increase if the design of a system is altered. Mathematical modeling of risks is used in such quantitative part of the analysis.

Risk analysis is applied in many different situations and therefore the analyses have different approaches as well. Example the sequences of events causing hazards are different in case of a nuclear power plant and an aircraft. More over the nuclear plants are different from each other and there are thousands of different types of aircrafts as well. Still, the both systems can be physically modeled as they are mainly technical constructions, though the operation often incorporates human factors. To go further with different applications, in large projects risk analysis is used to determine the most probable problems beforehand in order to overcome them without futile costs. But how about the government making a large scale decision? That is far away from technical system and the premises behind the visions and the decisions are often political. Still, risks loom and they should be, and hopefully are, considered beforehand.

The focus of this project is to study the quantitative risk analysis in the context of a nuclear power plant. The mathematical approach used is called *PRA* (Probabilistic Risk Analysis), also known as *PSA* (Probabilistic Safety Assessment). The method uses probability distributions and logic models of different events connected to their magnitude or cost and the theory is generally applicable. In the methology the risk analysis is divided into three levels: (**1**) technical event sequences leading to core damage, (**2**) the consequent radioactive releases after certain plant damage states and (**3**) the wider consequences on the environment. In addition, the risk analysis involves the analysis of risk reduction operations. In practice, the aim of the project was to re-construct the *level 2* model described in Okkonen (1995) with a given software program *FinPSA* and link the given *level 1* plant damage states to the *level 2*. The *level 2* model itself is validated and discussed in Okkonen (1995), which was excluded from the scope of the project. The model can be used to demonstrate the PSA-method in educational purposes. Due to restrictions set by the demo version of *FinPSA,* the model constructed in the report does not correspond completely to the earlier model. Therefore, the verification of the two models' equality is restricted.

In Chapter 2 we shortly discuss some of the basic principles of classical risk analysis and explain how risk analysis can be used to qualify different kind of risks. In Chapter 3 we study one risk analysis methodology, *PSA*, more in-depth. Chapter 4 is the practical part of the project, including short descriptions of the plant, re-construction of the model and results gained. In Chapter 5 we share our experiences on the software used.

2 Risk analysis

2.1 Risk definition

In exact terms, *risk* is some *event* S added with its negative *outcome* R and *probability* p as a triplet (S, R, p). Risk analysis helps to map the unknown events S_i linked to *risks* R_i and *probabilities* p_i so that the wide picture of the system risks is revealed. Furthermore, risk analysis should give the best opportunities to reduce risks with certain costs.

2.2 Probability and uncertainty

In technical systems the probability distribution of lifetimes for many components can be determined using existing statistics. That is, one cannot know the lifetime of a component before it breaks, but the probability distribution of the lifetime is known. This sort of uncertainty is called *aleatoric*, or, random. Another type of uncertainty is called *epistemic*. Drouin (2009) divides the epistemic uncertainties into three following categories: uncertainty regarding to the parameters, completeness of the model and model itself. That is: are all the relevant risks, their causes and consequences identified? Do the physical premises hold? Is the probability and the magnitude of the impact estimated correctly? What if another, qualitatively equally good model, gives quantitatively different results?

2.3 Probability estimation

There are situations where *epistemic* uncertainty is very high. For example yet there exist no statistical data for how often a computer-controlled car is having a serious accident. To obtain the best estimate for the frequency (accidents/year) one must use expert judgements to map the risks. This simply means asking different experts “What is the probability for computer-controlled car to have a serious accident per kilometer”. It is also possible that the experts are asked to give a probability distribution for the probability. When the expert judgements are gathered, one uses some model to define the uncertainty of the expert opinions.

In mathematical words, the described idea means that experts give an initial probability for the event (so called *a priori probability*). This could also be other way around: if the *a priori probability* is known to be roughly p, then we can calculate the conditional probability p' (so called *a posteriori probability*) for the event using experts judgements. The mathematical formula for the *posteriori probability* p' is given as $p' = \mathbb{E}[p|\text{expert judgements}]$. (O’Hagan, 2006) For some situations, we might have an estimate for the frequency. If one knows nothing about the *a priori probability,* then it should be estimated somehow. One approach is to use the maximum entropy principle. For example if we have an uneven coin, we obtain the *a priori probability* p for “the result is tail” with the principle of maximum entropy so that $p = 0.5$. (Campenhout et al., 1981) However it is a very exceptional case that we do not have any information from the probability. Almost in every situation, at least some rough estimate is known.

It depends on the situation if experts are asked the *a priori* or *a posteriori- probability*. For estimating risks in computer-controlled car the *a priori* approach for estimation is required. The posteriori probability for the risk could be used when some sufficient statistical data from computer-controlled car accidents are obtained. From other way around, when the initial probability is known but it is assumed that experts could have some additional information about the scenario then one can use the expert’s opinions for calculating *a posteriori*.

2.4 Risk-informed decision-making

The general aim of the risk analysis is to support decision-making process so that the possible unpleasant outcomes and their effects could be taken into account. Drouin (p. 10, 2009) depicts the phases of risk-informed decision-making process (Figure 1). The first way is to define the decision that, in the nuclear power plant context, could be for example:

- How to design the nuclear power plant?
- What is the maximum safe production rate?
- When to have the next maintenance breaks?
- Can two safety systems be repaired during operation?
- Are the emergency operating procedures adequate?

The second phase is to ensure that the current regulations affecting the decision are understood and identified. The third phase is to perform the analysis itself, including deterministic and probabilistic analysis. Deterministic analyses provide boundary conditions and success criteria for the probabilistic model. For example determining the minimum configuration to manage a disturbance or how much time operators have to recover a situation belongs to the deterministic phase. Probabilistic analyses provide the risk metrics and associated event importance information for decision making. The fourth phase is to implement the program and monitoring its performance. The fifth phase is the final decision and its essential element is the assessment of uncertainties.

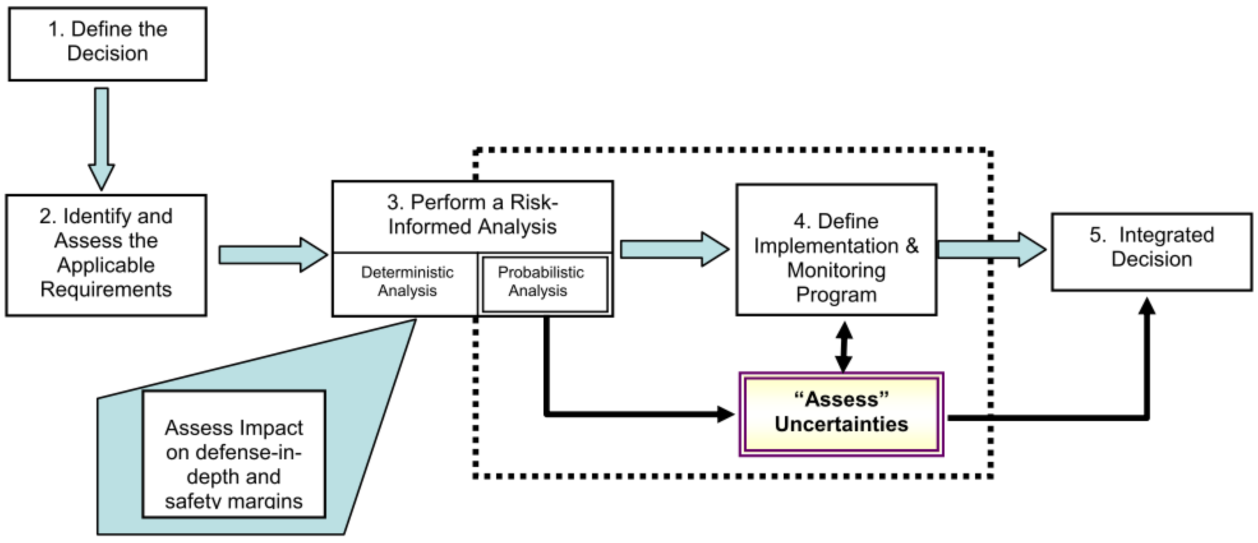
1. Define the Decision
2. Identify and Assess the Applicable Requirements
3. Perform a Risk-Informed Analysis
Deterministic Analysis
Probabilistic Analysis
4. Define Implementation & Monitoring Program
5. Integrated Decision
"Assess" Uncertainties
Assess Impact on defense-in-depth and safety margins

Figure 1: Phases of risk-informed decision-making process. Image source: Drouin (p.13, 2009).

3 Probabilistic Safety Assessment (*PSA*)

In this project, we perform risk analysis with a method called *PSA* (Probabilistic Safety Assessment). The method is one of the mathematical methods of risk analysis designed to define risks and quantify them in mathematical terms. *PSA* is based on *event tree-fault tree* approach to model accident scenarios. *Fault trees* are used to connect *basic events* to each other using logic gates in order to represent the failure logic of the system. *Event trees* define the probabilities of the different scenarios (outcomes/different failure states). In the following chapters, we define and discuss usage and utility of *fault trees* and *event trees*.

3.1 Fault trees

Fault trees are models or of the system that represents the failure logic of the system, including defining which failure combinations can cause the system failure, which is called the top event of *fault tree.* The logic in the *fault trees* is based on *Boolean* variables that have a binary value of *true* or *false*. The binary value indicates the state of some system part or event. If the event occurs, the value of the Boolean variable related to the event is *true* or if it does not then the value is *false*. (Barlow et al., 1975) The event could be *failure* of a system part. The *fault tree* consists from *basic events* that are connected mainly with *OR*- and *AND*-gates that are used to construct the logic the tree presents. There exist many other ports like *XOR* (One input and one only) that can be used to simplify complex logic.

Cut sets are all the sets of *basic events* that cause the system failure. From *cut sets* can be defined so called *minimal cut sets* that are useful in risk analysis. *Minimal cut set* is *a cut set* that contains minimal amount of failures to cause system failure. If any event is removed from *minimal cut set,* the *top event* does not occur anymore. If probabilities for the *basic events* in the *event tree* are known precisely, the upper boundary for the failure probability can be defined using *minimal cut sets*. In addition minimal cut sets can be used for example calculating risk important measurements.

The figure 2 demonstrates an example of a *fault tree* of a gasoline-to-electricity generator. Generator works if it gets fuel. To get fuel, there must be fuel in the fuel tank, fuel should be pumped to fuel injector and finally the injector should be working. The engine needs to get fresh air to operate. The fuel is injected with fresh air to cylinder. This is the situation when spark plug takes action (ignition). If both of the spark plugs fail then the engine fails. However, it is adequate that only the other spark plug operates (*AND*-gate*)*. In long run, the engine should also be cooled to avoid overheating. The fault tree in the figure 2 presents all just described *basic events* that can cause the generator failure.

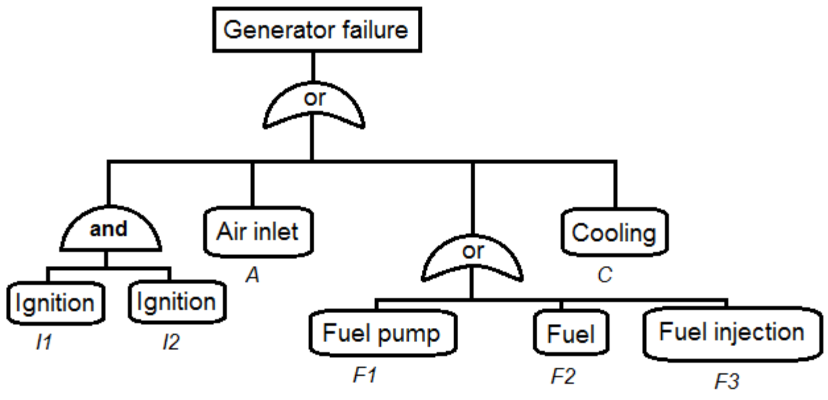

Figure 2: Fault tree for generator failure

Minimal cut sets can be obtained using *Boolean algebra*. For the figure 2 we get:

$$TOP = (I_1 \cap I_2) \cup A \cup (F_1 \cup F_2 \cup F_3) \cup C.$$

So the *minimal cut sets* are:

$\{I_1, I_2\}, \{A\}, \{F_1\}, \{F_2\}, \{F_3\}$ and $\{C\}$.

One can see that the only *minimal cut set* containing more than two events is the set linked to ignition system. This is because there is a redundancy in the ignition system. There are many *minimal cut sets* containing only one element because the system fails from almost any failure of one part of it.

Defining the upper bound for the generator failure probability is done with *minimal cut sets* using upper boundary for the union probability for small probabilities $P(A)$ and $P(B)$. The approximation is done so that $P(A \cup B) = P(A) + P(B) - P(A \cap B) \approx P(A) + P(B)$. From the fact that $P(A \cap B) \geq 0$ if follows that $P(A \cup B) \leq P(A) + P(B)$. This ensures that the risk analysis gives upper bound for the risk from this point of view. However, this does not ensure that the risk analysis gives upper bound for the risks: What if the probabilities for the events are underestimated, or what if not the all events have been included in the model?

Using same kind of approach for the figure 2 we obtain

$$P[(I_1 \cap I_2) \cap A \cap (F_1 \cup F_2 \cup F_3) \cap C] \approx P[(I_1 \cap I_2)] + P[A] + P[(F_1 \cup F_2 \cup F_3)] + P[C] \approx$$

$$P[I_1]P[I_2] + P[A] + P[F_1] + P[F_2] + P[F_3] + P[C]\ .$$

3.2 Event trees

An *event tree* represents the progression of accident scenarios from an initiating event to consequences. The tree is a tool to calculate probabilities of different scenarios. For example in case of a nuclear power plant accident, there are different outcomes that are defined from the state of the system. Outcomes depend on how the accident takes place: did the core melt or not, did all the safety precautions fail.

In the figure 3 we have an *event tree* containing chain of events. It demonstrates the situation when the primary generator fails. There are three devices to ensure that the system still gives electricity in this kind of malfunction situation. The three parts are flywheel carrying rotational energy, secondary generator that can produce electricity and lithium batteries that can quickly respond to cover a quick shortage of energy.

If the flywheel operates, there is enough time to start the secondary generator to produce electricity and no problem occurs. If the secondary generator fails then one has still 100 seconds to do something before the flywheel has no energy to rotate and lithium batteries are empty.

If the flywheel fails then the lithium batteries can still cover the shortage of energy for the time of starting secondary generator. However, the generator needs time to start, so the system will immediately fail if the batteries fail with the flywheel. In the case that only batteries work, there is 90 seconds time to do something for the situation.

What is done in the timeframe of t = 10s, 90s, 100s is not involved in the model. It could be starting a third generator, gas turbine or something else. If no action is taken, the system will fail after that time.

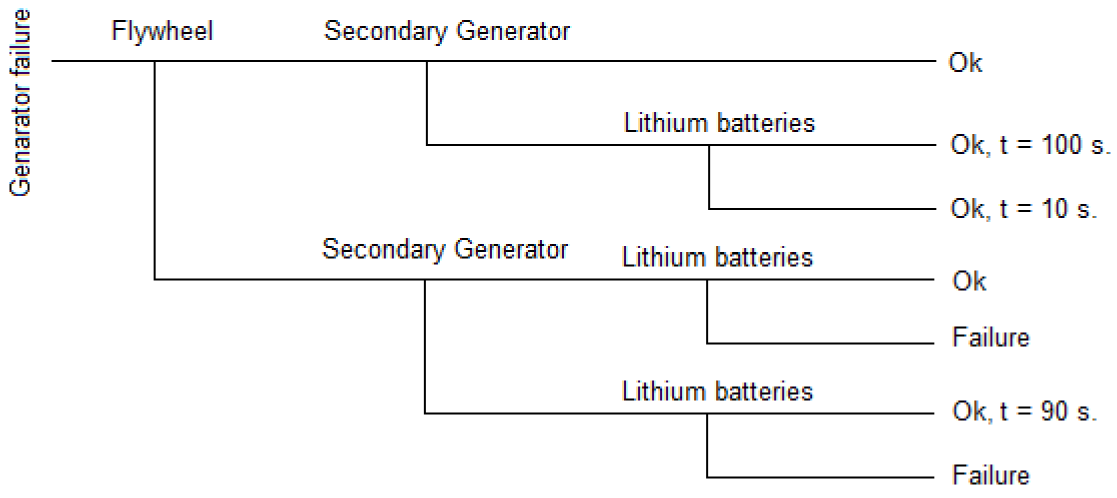

Figure 3: A simple event tree for a power generation problem

3.3 General structure of a nuclear power plant PSA

We use a nuclear power plant as an example demonstrating the 3-level *PSA*-model, whose main idea is presented in the Figure 4[1]. The levels are modeled distinctively but the sub-models are dependent on each other. Their scopes and aims are the following:

- Level 1: core damages and their frequency
- Level 2: fission releases and their frequencies
- Level 3: wider consequences in the environment

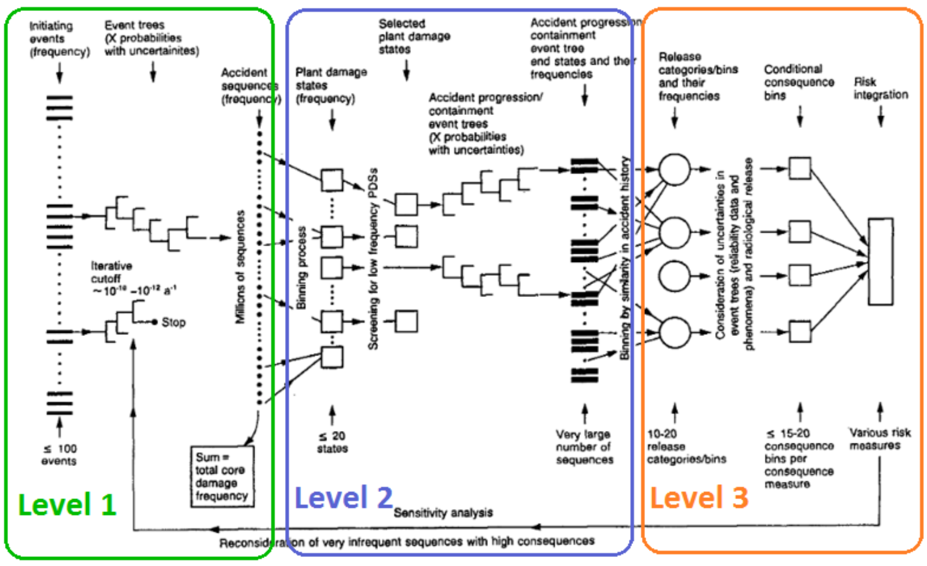
Initiating events (frequency)
Event trees (X probabilities with uncertainites)
Accident sequences (frequency)
Iterative cutoff ~10⁻¹⁰ –10⁻¹² a⁻¹
Stop
Millions of sequences
≤ 100 events
Sum = total core damage frequency
Level 1
Plant damage states (frequency)
Selected plant damage states
Accident progression/ containment event tree end states and their frequencies
Accident progression/ containment event trees (X probabilities with uncertainties)
Binning process
Screening for low frequency PDSs
Binning by similarity in accident history
≤ 20 states
Very large number of sequences
Level 2
Sensitivity analysis
Release categories/bins and their frequencies
Conditional consequence bins
Risk integration
Consideration of uncertainties in event trees (reliability data and phenomena) and radiological release
10-20 release categories/bins
≤ 15-20 consequence bins per consequence measure
Various risk measures
Level 3
Reconsideration of very infrequent sequences with high consequences

Figure 4: General 3-level structure of PSA (Source of the original picture: IAEA Safety Series N. 50-P-8)

Level 1 starts from an initiating event such as loss of offsite power and the following events are modeled using an event tree. The output of the *level 1* is a series of core damages with corresponding frequencies. The numerous core damage sequences are grouped to form certain *plant damage states* that are the input

[1] This is only one use of *PSA*: *PSA*-models could be implemented in other contexts as well.

for the level 2. To link certain plant damage states and *level 2 event trees* together, we used *interface trees* which implement the binning rules for plant damage states. This is important as *level 1* events might affect the *level 2* functionality.

Level 2 output calculation uses *event trees*. Output is the frequency distribution for different radioactive releases from the nuclear power plant accident. Those releases involve radioactive substances for example: Xenon, Cesium, Iodine and Tellurium. *Level 2* output is divided to different scenarios depending on the amount of emissions and seriousness. *Level 2* outputs are inputs for *level 3*. At this time, *level 2* output is divided quite roughly to different release categories. This is done because *level 2* produces numerous outputs. The binning is a good approach for modeling the system and the accident in this stage.

Level 3 defines how radioactive emissions spread out to environment. *Level 3* can also take into account countermeasures such as evacuation, sheltering, land-use restrictions, which have an important contribution to the population dose.

3.4 Monte-Carlo simulation

In *level 1 PSA* and in many *level 2 PSA* application, the modeling is based on *event tree – fault tree* approach and solving *minimal cut sets*. The *FinPSA*-tool for *level 2 PSA* is based on use of so called *Monte Carlo simulation,* due to limitations of *fault tree* modeling to handle complex combinations of uncertainty distributions for a large number of parameters, which is the case when modeling severe reactor accident phenomena.

In large-scale systems there could exist more than 1000 components and enormous amount of physical states with probability distributions. The system is so complex that simulating every possible input combination is not a realistic goal or not possible at all. That is why *Monte-Carlo simulation* is used. *Monte-Carlo simulation* means that the outcome of the system is simulated many times with different system parameters. The values of the parameters are drawn from probability distributions of the system parameters. When this simulation is executed many times, a good estimation for the distribution of the system outcome is obtained. Determining the risk of a system is often performed by numerous *Monte-Carlo simulations*.

The probabilities and system states are raffled from different probability distributions. The distributions are variable-specific. If one uses for example 10 *Monte Carlo simulations* simulating model, different run time gives almost certainly significantly different results and some rare events that should be taking into account does not occur at all. Using very large number of samples (of class $> 10^4$ depending on the nature of the system) ensures that rare events are taken into account and the probabilities are sufficiently accurate.

4 Developing the level 2 model

4.1 The demo of the nuclear power plant *level 2 PSA*

The aim of the project was to reconstruct a nuclear power plant model – similar to the model described by Okkonen (1995) – in new program environment. The probability equations and physical models were to be maintained the same, with minor modifications for syntactical or other relevant reasons. However, the demo-tool restricted the size of the model so that the model constructed was not identical with the previous one.

4.2 Boiling water reactor (BWR)

Schematic diagram of a boiling water reactor is presented in the Figure 5. Inside the reactor vessel, the energy released in the fission process boils the water into steam. The energy of the moving steam flow is transformed to rotational energy in turbine, which is later transformed to electrical energy. The steam is condensed back to water, and is returned into the core. The fuel of the BWR is radioactive and isolated physically by multiple barriers: fuel cladding, reactor pressure vessel and primary circuit, reactor containment. Safety systems take care of keeping the barriers intact in case of process disturbances such as leakages in primary circuit. The safety systems include i.e. quick reactor shutdown and halting of the fission process, reactor coolant systems and severe accident management (SAM) to prevent the radioactive releases.

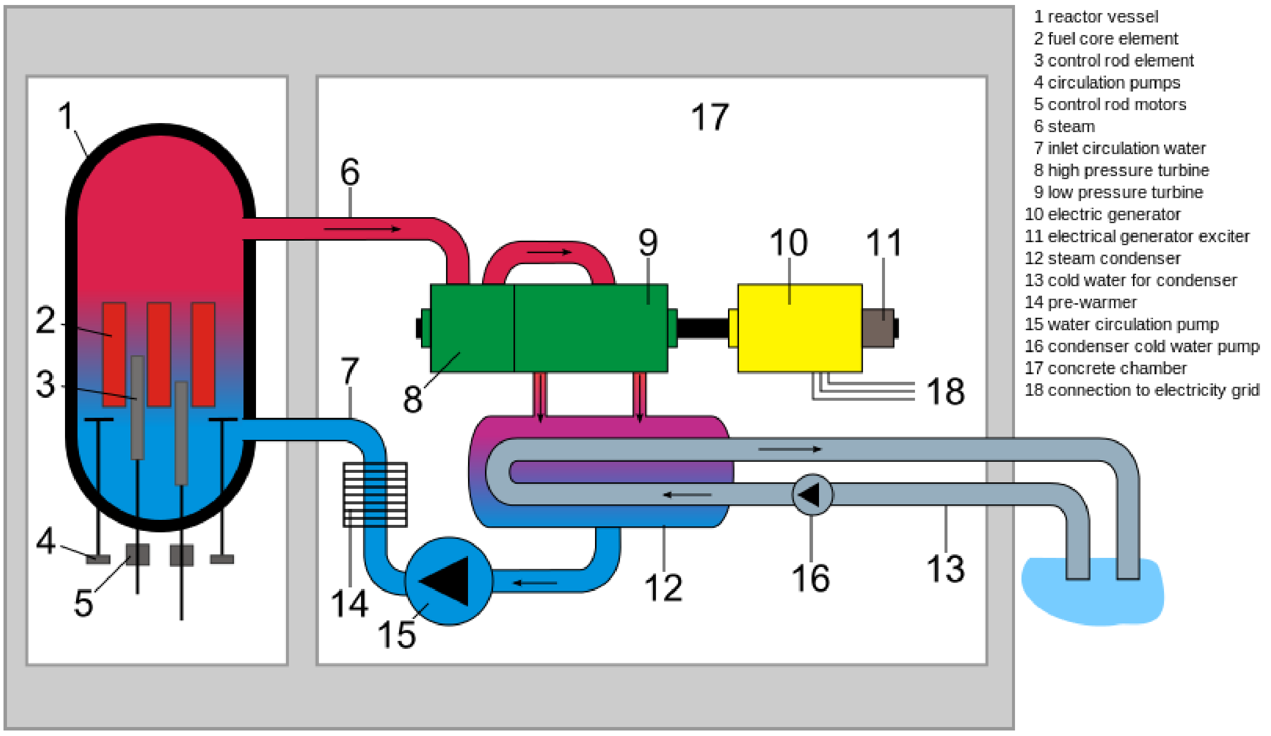

Figure 5: Schematic figure of a BWR. By Robert Steffens (alias RobbyBer 8 November 2004), SVG: Marlus_Gancher, Antonsusi (talk) using a file from Marlus_Gancher. https://commons.wikimedia.org/w/index.php?curid=14617356

4.3 The computer tool (*FinPSA*) and the model

Our goal is to import the old model to the new computer program working in newer *Windows operating systems*. The old program (*SPSA*) is based on old MS-*DOS* (Microsoft Disk Operating system). Modern day operating systems (2016) like *Microsoft Windows 7/8/8.1/10* differs significantly from this. The new program for *PSA*-model implementation is called *FinPSA* which has been developed since 2001. Compared to old *DOS*-version based one, it involves more user-friendly graphical user interface (*GUI*). Computation algorithms were first inherited from SPSA, but they have been improved later, including parallel computation. *FinPSA* contains some features and properties that *SPSA* does not and vice versa.

FinPSA Level 2 incorporates a programming language *CETL* (Containment Event Tree Language) that is functional programming language and is syntactically close to i.e. Pascal. The language is designed for building different kind of logical and mathematical modeling of systems and all the code handled during the project was written with it. The code can be written into CET branches describing the physical event, or into common section that is common to all CET:s in the model. The branch-functions (an example of a CET with such branches is shown in the Figure 6) determine the probability of the event, and its complement's, probability. The common section may contain global variables or frequently used functions that can be called from CET specific scripts. For each CET one has to provide initialization section containing initial and finalization routines, in addition i.e. rules how the different events are binned[2] to form certain *release category*.

4.3.1 Model importing

Importing the model can be divided roughly into three phases:

I. Construction of three *containment event trees* (CET)
II. Integrating the *level 1* and *level 2*. *Level 1* accident sequences are linked to different plant damage states. Each PDS is linked to its CET via an interface tree
III. Importing functionality
    - Common section where the general variables and functions are defined
    - Functions associated with branches
    - Involving source code modification to the new system, minor changes at notations

The first phase is thoroughly reported in Okkonen (1995). One CET describes progression of the events in the containment and reactor vessel after certain severe accident, or, plant damage states (PDS) that are results from the *level 1*. The *FinPSA* demo version restricted the number of the CETs to three. We were advised to select the trees HPM, LPM-LM and LPM-TR, explained shortly in the Table 1. NOS, ROP and COP have typically a very low frequency and are thus less interesting for PSA modelling. On the other hand, LPM is the most important element in SAM as the pre-core meltdown conditions are "expected" and can thus be guarded against. However, one version of it, LPM-SB, was left outside the model.

[2] This is similar to that of binning level 1 outputs to certain plant damage states.

Table 1: Plant Damage States of the original model

Abbreviation	Explanation
NOS	NO Scram[3]. I.e. control rods not into the core due to hydraulic problem. Severe releases into the environment.
ROP	Reactor Over-Pressure. Caused by loss of reactor coolant system (RCS) pressure control and would lead RCS blow-down.
COP	Containment Over-Pressure. RCS blow-down might continue to this catastrophic consequence.
HPM	High-Pressure Melting. The core starts melting at high RCS pressure. Many probably causes, including technical failures and operating errors with the unsuccessful depressurization of the coolant system.
LPM	Low-Pressure Melting. Sub-critical core, an intact containment and a low RCS pressure (“expected” pre-core-meltdown conditions).
LPM-TR	Transient[4].
LPM-SB	Small-Break LOCA[5] (Loss Of Coolant Accident).
LPM-LM	Large/Medium-break LOCA.
VLL	Very Late melting at Low RCS pressure.

Main events in all the selected CETs are the same and the branch structure for LPM-LM is given in the Figure 6. For example, the branching section RECO (Emergency Core Cooling Systems Recovery) determines whether the system can recover the (emergency) core cooling systems before the complete core meltdown. This, naturally, has a considerable impact on the later events. In practice, many global variables are changed in all of these functions as the events proceed and it makes the model *dynamic*.

In the Figure 6 one can see a simple definition for the events linked to branches. For example the event MD means Core melting down and at that branch evaluates probability for core melting down, reflecting the earlier events in the sequence. For a more comprehensive definition of the events, see appendix A (PSA2-Documentation) and the report Okkonen (1995).

[3] DEFINITION AT www.nrc.gov: Scram = The sudden shutting down of a nuclear reactor, usually by rapid insertion of control rods, either automatically or manually by the reactor operator. [Accessed 2.5.2016]
4 DEFINITION AT www.nrc.gov: Transient = A change in the reactor coolant system temperature, pressure, or both, attributed to a change in the reactor's power output. Transients can be caused by (1) adding or removing neutron poisons, (2) increasing or decreasing electrical load on the turbine generator, or (3) accident conditions. [Accessed 29.4.2016]
[5] DEFINITION AT www.nrc.gov: LOCA = Loss of coolant accident. Accidents that result in a loss of reactor coolant at a rate in excess of the capability of the reactor makeup system from breaks in the reactor coolant pressure boundary, up to and including a break equivalent in size to the double-ended rupture of the largest pipe of the reactor coolant system. [Accessed 8.5.2016]

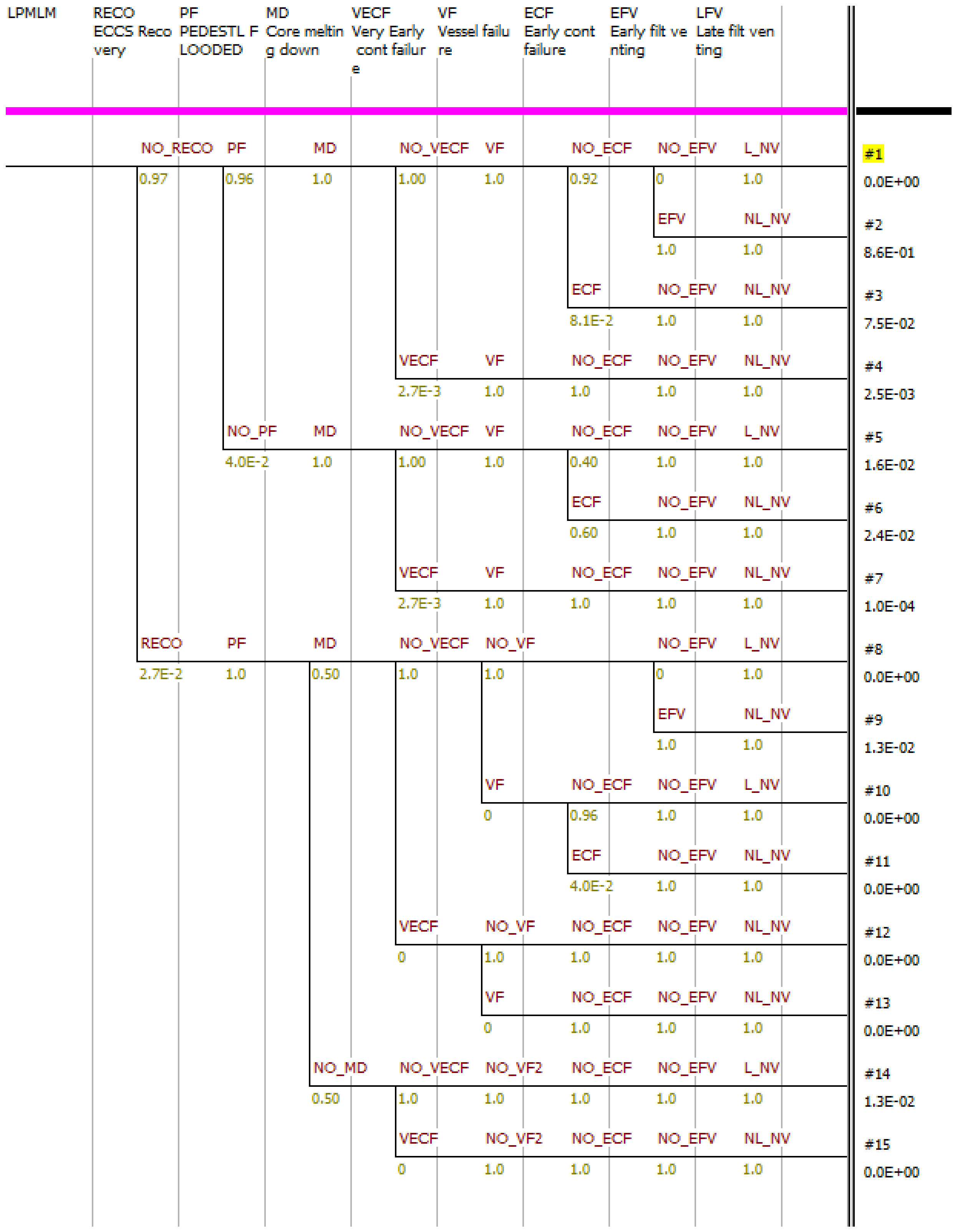
LPMLM
RECO ECCS Recovery
PF PEDESTL FLOODED
MD Core melting down
VECF Very Early cont failure
VF Vessel failure
ECF Early cont failure
EFV Early filt venting
LFV Late filt venting
NO_RECO 0.97 PF 0.96 MD 1.0 NO_VECF 1.00 VF 1.0 NO_ECF 0.92 NO_EFV 0 L_NV 1.0 #1 0.0E+00
EFV 1.0 NL_NV 1.0 #2 8.6E-01
ECF 8.1E-2 NO_EFV 1.0 NL_NV 1.0 #3 7.5E-02
VECF 2.7E-3 VF 1.0 NO_ECF 1.0 NO_EFV 1.0 NL_NV 1.0 #4 2.5E-03
NO_PF 4.0E-2 MD 1.0 NO_VECF 1.00 VF 1.0 NO_ECF 0.40 NO_EFV 1.0 L_NV 1.0 #5 1.6E-02
ECF 0.60 NO_EFV 1.0 NL_NV 1.0 #6 2.4E-02
VECF 2.7E-3 VF 1.0 NO_ECF 1.0 NO_EFV 1.0 NL_NV 1.0 #7 1.0E-04
RECO 2.7E-2 PF 1.0 MD 0.50 NO_VECF 1.0 NO_VF 1.0 NO_EFV 0 L_NV 1.0 #8 0.0E+00
EFV 1.0 NL_NV 1.0 #9 1.3E-02
VF 0 NO_ECF 0.96 NO_EFV 1.0 L_NV 1.0 #10 0.0E+00
ECF 4.0E-2 NO_EFV 1.0 NL_NV 1.0 #11 0.0E+00
VECF 0 NO_VF 1.0 NO_ECF 1.0 NO_EFV 1.0 NL_NV 1.0 #12 0.0E+00
VF 0 NO_ECF 1.0 NO_EFV 1.0 NL_NV 1.0 #13 0.0E+00
NO_MD 0.50 NO_VECF 1.0 NO_VF2 1.0 NO_ECF 1.0 NO_EFV 1.0 L_NV 1.0 #14 1.3E-02
VECF 0 NO_VF2 1.0 NO_ECF 1.0 NO_EFV 1.0 NL_NV 1.0 #15 0.0E+00

Figure 6: CET-model for the Plant Damage State LPM-LM

In the second phase, we constructed three *interface trees*, one for each selected PDS and the corresponding CET. The binning rules are given in the Table 2. For example, the initiating event that is not a large loss of coolant accident (ALOCA) eventually leads to plant damage state HPM (High Pressure Melting) and core damage category CD2, **if** the first event C (failure of reactor scram system) successes and the events Q (Main feed water fails), U (Emergency feed water system fails) and X (Automatic depressurization system fails) do fail. In the interface trees, then, we linked that *level 1* core damage to *level 2* CET HPM. So, the output of the *level 1* now is connected to *level 2*. Note that the interface tree structure in this case was very simple because there were even no branches in them. Often, the interface trees are more complex and include additional conditions that are helpful for the modeling in *level 2*.

Table 2: Binning rules to link the levels 1 and 2 together

Initiating event	**C**	**Q**	**U**	**X**	**V**	**W1**	**W3**	**Core damage category**	**Plant damage state**
any	fail	-	-	-	-	-	-	CD1	NOS
not ALOCA	success	fail	fail	fail	-	-	-	CD2	HPM
not ALOCA	success	fail	-	-	fail	-	-	CD2	LPM-TR
ALOCA	success	-	-	-	fail	-	-	CD2	LPM-LM
any	success	fail	?	?	?	fail	-	CD3	VLL

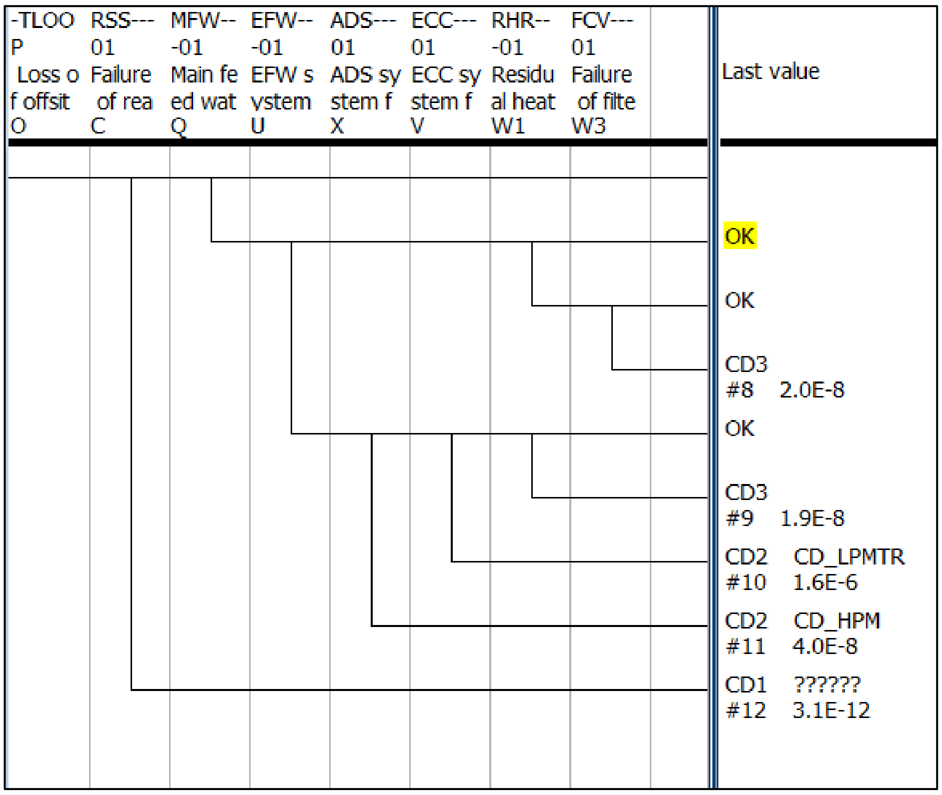

Figure 7: Event tree for loss of offsite power (LOOP) and its link to the *level 2* CETs LPMTR and HPM through interface trees CD_LPMTR and CD_HPM.

In the first phase, the model was built with a static demo code defining the branching probabilities. In the third phase that was replaced with correct code, copied from the source report (with slight syntactical changes, though).

4.3.2 Model validation

The validation of the physical model constructed was excluded from the scope of the project. The technical model and the failure probabilities of different components are direct copies from the existing model provided by Okkonen (1995). As the model is more than 20 years old, some updates to the modeled plant was made and therefore the model does not represent the current situation. Such validation is less important as this model will be used for demonstration and education for *level 2 PSA.* Also, the methodology is still valid and the modeled plant features are representative enough. Before actual use of such model, the model should be verified to make it correspond the real plant.

4.3.3 Current state of the model

The integrated model of *levels 1* and *2* works now in a restricted manner. The re-constructed model differs significantly from the original one. In the figures below the CET LPM-TR are shown for both models. The *FinPSA* demo version restricted the number of sections to 9 and number of sequences to 15. So, first three sections and the last were omitted, which resulted in 15 sequences.

The simulation works only with some random seed numbers, which means that some errors either in the program code, in the physical model or in the random numbers exist.

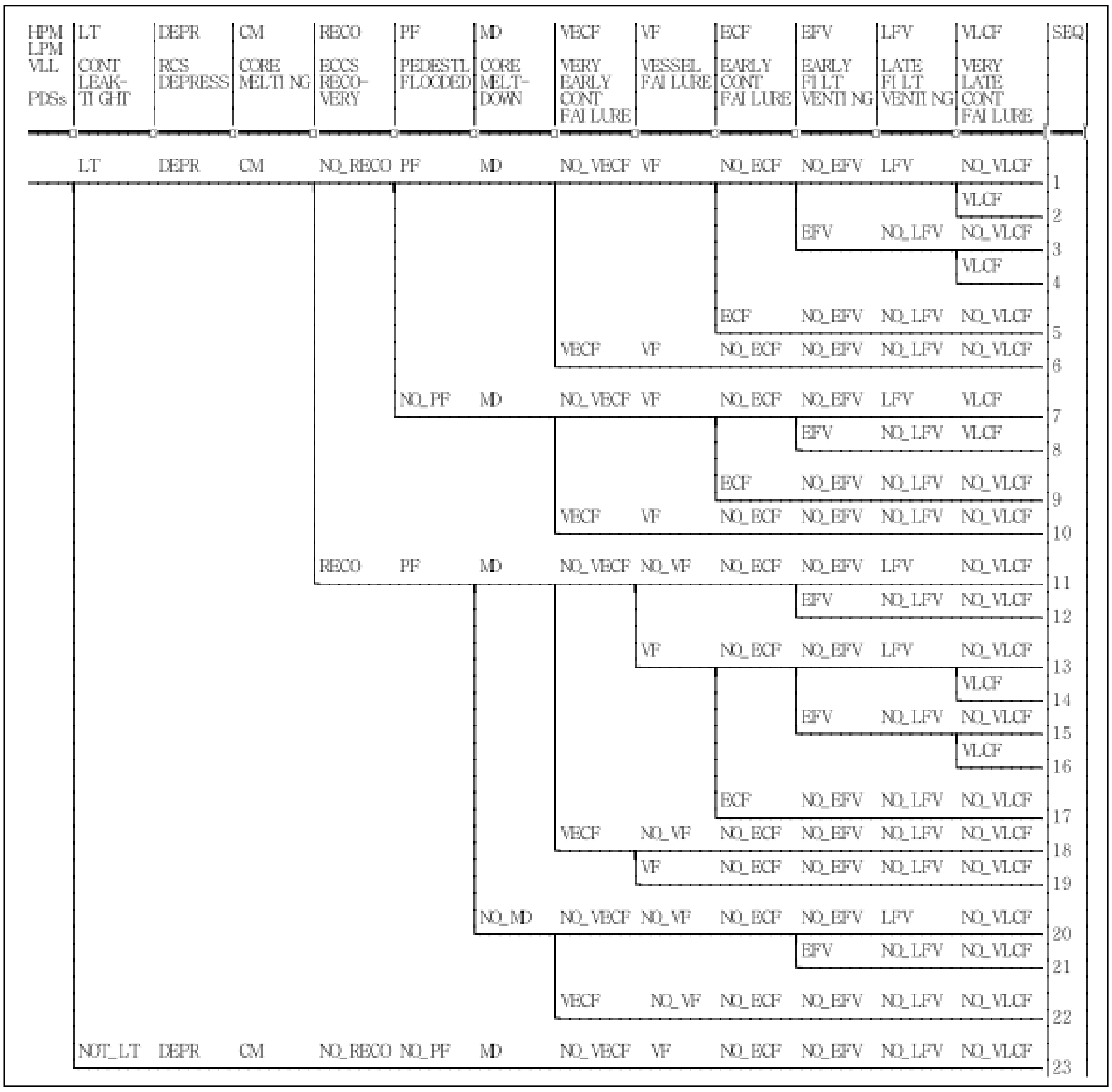
HPM LPM VLL PDSs
LT CONT LEAK-TIGHT
DEPR RCS DEPRESS
CM CORE MELTING
RECO ECCS RECO-VERY
PF PEDESTL FLOODED
MD CORE MELT-DOWN
VECF VERY EARLY CONT FAILURE
VF VESSEL FAILURE
ECF EARLY CONT FAILURE
EFV EARLY FILT VENTING
LFV LATE FILT VENTING
VLCF VERY LATE CONT FAILURE
SEQ
LT DEPR CM NO_RECO PF MD NO_VECF VF NO_ECF NO_EFV LFV NO_VLCF 1
VLCF 2
EFV NO_LFV NO_VLCF 3
VLCF 4
ECF NO_EFV NO_LFV NO_VLCF 5
VECF VF NO_ECF NO_EFV NO_LFV NO_VLCF 6
NO_PF MD NO_VECF VF NO_ECF NO_EFV LFV VLCF 7
EFV NO_LFV VLCF 8
ECF NO_EFV NO_LFV NO_VLCF 9
VECF VF NO_ECF NO_EFV NO_LFV NO_VLCF 10
RECO PF MD NO_VECF NO_VF NO_ECF NO_EFV LFV NO_VLCF 11
EFV NO_LFV NO_VLCF 12
VF NO_ECF NO_EFV LFV NO_VLCF 13
VLCF 14
EFV NO_LFV NO_VLCF 15
VLCF 16
ECF NO_EFV NO_LFV NO_VLCF 17
VECF NO_VF NO_ECF NO_EFV NO_LFV NO_VLCF 18
VF NO_ECF NO_EFV NO_LFV NO_VLCF 19
NO_MD NO_VECF NO_VF NO_ECF NO_EFV LFV NO_VLCF 20
EFV NO_LFV NO_VLCF 21
VECF NO_VF NO_ECF NO_EFV NO_LFV NO_VLCF 22
NOT_LT DEPR CM NO_RECO NO_PF MD NO_VECF VF NO_ECF NO_EFV NO_LFV NO_VLCF 23

Figure 8: LPM-TR CET in Okkonen (1995)

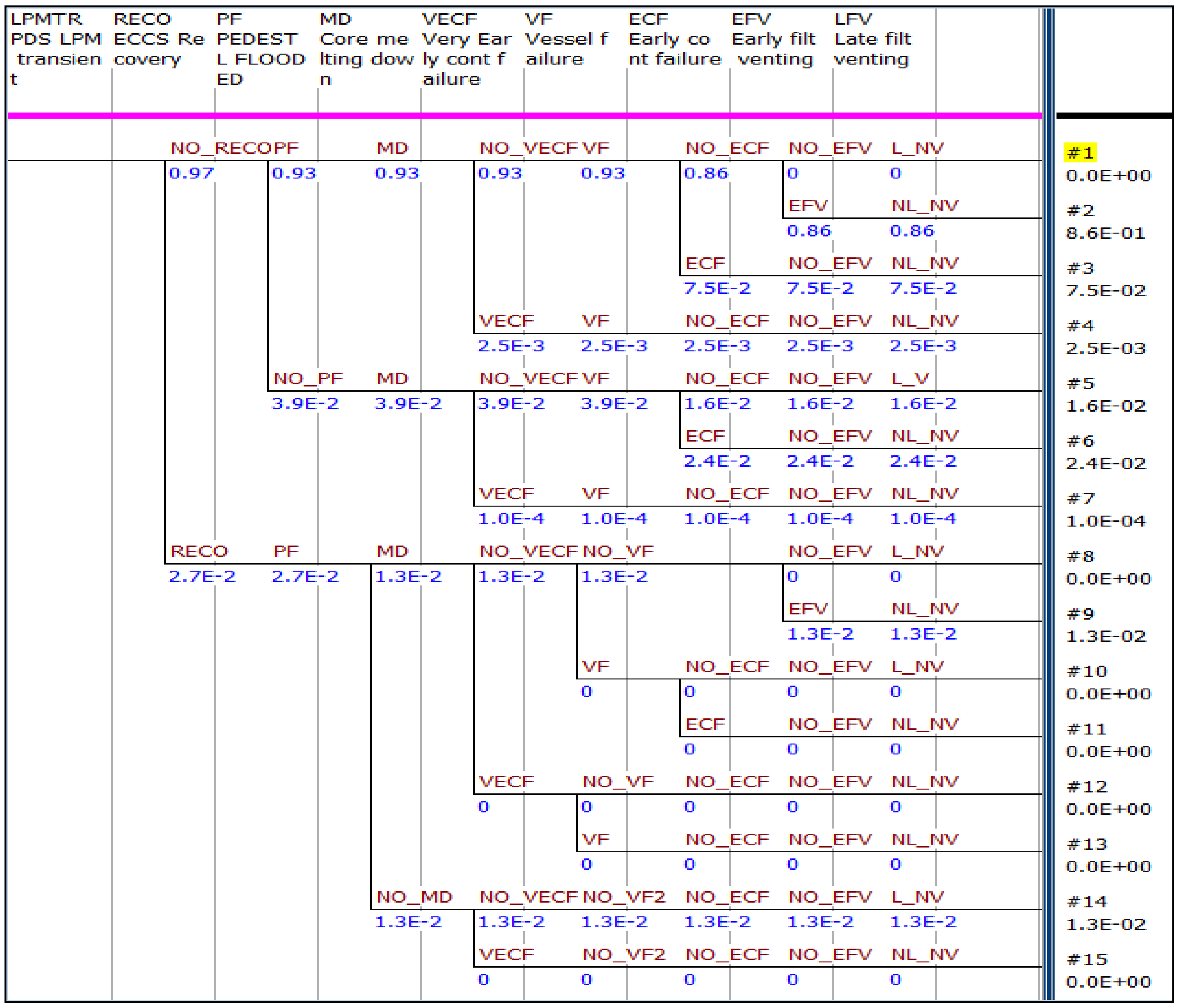

Figure 9: LPM-TR CET in our model

In the table 3 the differences and their consequences on the final results may be more visible. For example, the first two sequences in the original model led to different release categories but we had to embed the sequence 2 to sequence 1. Consequently, only the categories 2 and 3 would be fully comparable with the existing results.

Table 3: Transformation table between the two models, the reconstructed one shown with an asterix. Erroneous bins colored grey.

Seq	Bin	Seq*	Bin*
1	6	1	6
2	4	1	6
3	5	2	5
4	4	2	5
5	3	3	3
6	2	4	2
7	4	5	4
8	4	5	4
9	3	6	3

10	2	7	2
11	6	8	6
12	5	9	5
13	6	10	6
14	4	10	6
15	5	10	6
16	4	10	6
17	3	11	3
18	2	12	2
19	2	13	2
20	6	14	6
21	5	14	6
22	2	15	2
23	1	-	-

4.4 Results

4.4.1 Result comparison

In this chapter, we present some results of the new model and compare them to the results of the original model. Note that the model does not work properly. However, these results may indicate probable errors in the model.

The results of the *Level 2* are the frequency (probability) of certain event sequence and its consequences, namely fraction of different fission products released. In the results presented here, probabilities are conditional, assuming plant damage states. Releases from the containment in the CET LPM-TR is presented are the table below. The release fraction of Xenon is highest in all branches, which is physically justified as Xenon leaks out whenever the containment is not solid.

Table 4: Frequencies and releases (fractions) from different LPM-TR branches

Sequence	Prob	XE	I	CS	TE
Seq 1	5,30E-02	7,70E-01	2,10E-04	1,60E-04	9,10E-05
Seq 2	8,00E-01	7,70E-01	2,40E-04	1,70E-04	8,90E-05
Seq 3	8,20E-02	1,00E+00	5,20E-01	5,50E-01	3,30E-01
Seq 4	2,00E-03	7,70E-01	2,90E-01	2,20E-01	1,20E-01
Seq 5	1,30E-02	1,00E+00	1,10E-03	1,10E-03	7,60E-04
Seq 6	2,10E-02	1,00E+00	5,20E-01	5,50E-01	3,30E-01
Seq 7	6,20E-05	1,00E+00	7,20E-01	7,10E-01	4,20E-01
Seq 8	9,00E-05	6,70E-01	5,50E-05	4,80E-05	2,90E-05
Seq 9	1,10E-02	6,70E-01	5,60E-05	5,00E-05	3,00E-05
Seq 10	3,20E-03	6,80E-01	6,40E-05	5,90E-05	3,70E-05
Seq 11	1,50E-04	7,20E-01	8,50E-02	9,00E-02	5,50E-02
Seq 12	2,50E-06	6,70E-01	2,20E-01	1,70E-01	9,30E-02
Se1 13	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
Seq 14	1,60E-02	6,70E-01	5,50E-05	4,80E-05	2,90E-05
Seq 15	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00

4.4.2 Verification of the results

As the aim of the project was to import the model into the new program environment, the verification of *FinPSA* and *SPSA* equality in calculation belongs into the scope of the project. As an appendix to the model description, Okkonen (1995) provides some results that could be utilized in the verification procedure.

The problem in the verification is that there is limited number of Monte Carlo-simulations in use in *FinPSA* demo version. Also, the random number generators were reported to differ a bit in Okkonen's report and *FinPSA* but they were later parametrized so that the distributions would be as identical as possible – still lacking the identical random number sequences[6]. We have to rely on the "average" results that are attained from 100 (*FinPSA*) and 500 (*SPSA*) MC-simulations. If we could ensure that the random numbers are the same, the variation of the results should be smaller.

The results per category are presented in the following table. The most notable error in average results is the decade error in the probability of the category 2. That error is much diminished when looking at the median results. In these skew distributions, the median is a more robust measure of centrality and approaches it faster.

Table 5: Frequencies and release fractions of the comparable categories. The results of the reconstructed model are shown with an asterix.

Bin	Prob [%]	XE [%]	I [%]	CS [%]	TE [%]
	Average results				
CAT2	1,8 %	79,0 %	22,0 %	17,0 %	9,6 %
CAT2*	0,2 %	77,0 %	31,0 %	24,0 %	13,0 %
CAT3	9,9 %	100,0 %	45,0 %	46,0 %	29,0 %
CAT3*	10,0 %	100,0 %	52,0 %	55,0 %	33,0 %
	Median results				
CAT2	0,4 %	94,0 %	14,0 %	9,1 %	3,8 %
CAT2*	0,1 %	93,0 %	25,0 %	16,0 %	7,0 %
CAT3	8,5 %	100,0 %	42,0 %	40,0 %	22,0 %
CAT3*	8,1 %	100,0 %	57,0 %	58,0 %	28,0 %

Another way of verification would be to compare so-called point values. They are results of simulations using the expected values of the variables. They should reflect the same results for the identical models. However, we did not have the point values of the original model available and therefore that was not possible.

Because the models differ, the verification is conducted in a limited way and it was not given much weight in the end.

[6] When a fixed seed number is used, the sampled pseudorandom numbers are always the same because they are generated using the same algorithm and initial condition. Two models can then be verified as equal if the results are identical with all the same initial random seed numbers.

5 Suggestions to *FinPSA* development

In this section, ideas for *FinPSA* development are proposed according to our experiences. The necessity of them depends on the future desires of the program.

5.1 Problems with the interpreter in *FinPSA*

First, we imported and modified the *level 2 event trees* for the nuclear power plant model. The tree consists of events that correspond to some function. The functions represent some physical model of the nuclear power plant functions like *core melt down*. Those functions are defined with a programming language integrated to *FinPSA*.

When importing the programming code of the physical model, we recognized that the basic operations between two user-defined functions did not work out correctly. Any such operation – *addition*, *subtraction*, *division*, *multiplication* or *power* – gives zero as an output. For example defining ("=") a as the division of two user-defined functions $f(\cdot)$and $g(\cdot)$ is always zero.

$$a = \frac{f(x)}{g(y)},$$

That leads eventually to other errors later on, such as division by zero, infinite loops or erroneous results. For instance an increment inside a while loop calculated as a sum of three functions never increased and the loop end condition was never met. Because of such errors, the bug was found even though the error messages were confusing.

The developers of the program are working in order to fix the bug but we will not get the correct version before this project ends. As a work-around we separated every function calls to different rows as local variables, like

$$y_1 = f(x)$$

$$y_2 = g(y)$$

$$a = \frac{y_1}{y_2},$$

gives the correct result for variable a. We replaced such basic operations so that they now use only local variables but such work-around is prone to programming errors. For example a single line presenting a few basic operations between variables and user-defined functions defining variable a so that

$$a = \frac{f_3(x_2) + e^{g(y_1)+f1(x_2+y_2)}}{f_2(x_1) + \frac{g_2(y_2)}{f_3(y_3) + x_3}}$$

had to be transformed into (one possibility)

$$a = g(y_1)$$
$$a = a + f_1(x_2 + y_2)$$
$$a = \exp(a)$$
$$a = f_3(x) + a$$

$$b = g_2(y_2)$$
$$c = f_3(y_3) + x_3$$
$$b = \frac{b}{c}$$
$$b = f_2(x_1) + b$$
$$a = \frac{a}{b},$$

that should result to an correct answer for variable a.

In all, the bug made importing work harder, error-prone and increased the project workload and affected the motivation of the project group.

5.2 Debugging tools for the program

Another suggestion is that *FinPSA* could incorporate better debugging tools. At the moment one can use the built-in function *halt()* to debug the model. The function halts the program, shows the values of the variable at that time and in cases shows the error point in the code. This facilitated our work significantly because the function could be used to determine the variable/ function causing the errors. However, in addition serious disadvantages were found. For example our work-around function for division operation included a *halt()*-call, but there were hundreds of these division()-calls and the tool did not indicate *the location* of the error. Frequently we had to find the location of the error by manually setting *halt()*-calls to different locations. We implemented only a restricted demo model of a nuclear power plant that is relative easy to debug even semi-manually as we did. If one develops a more complicated model of some system then more powerful debugging tools would be a major help: Where the program does halt?

In the future *FinPSA* could involve a more powerful debugging tool that could introduce, for instance, a more accurate error locator. For example, it could tell the line where the *halt()* occurs. In addition, the *halt()*-function could list the variables in alphabet order and save the list to a text file to facilitate debugging.

5.3 Tool for editing the code

FinPSA involves an integrated text editor made for reading and editing the code. In our work, we experienced that the integrated text editor is a bit ineffective for example compared to *Notepad++*. Some cons that could be improved are listed next

- Search is a bit ineffective and it seems to show tabs differently than *Notepad++*
- *Text* editor does not show which brackets are pairs (helpful for studying and writing formulas)
- No efficient replace function
- clumsy to open from *GUI*

The major disadvantage using un-integrated separate text editor for editing *FinPSA* programing files is that when two different programs uses the same file (*I/O*) without any kind of loading-saving control there exists a danger to loss data. One does not know certainly if *FinPSA* gives the same code for the interpreter that is edited in *Notepad++*. Because of the function call problem described earlier, it would have been sticky to use only *FinPSA* text editor. The editor seems to be sufficient for minor edits and repairs.

5.4 The graphical user interface (*GUI*)

We found the graphical user interface (*GUI*) in the program clumsy in places. For example creating CET-trees, adding branches and functions is difficult for one who is not familiar with the program. For example, in order to add a function to a branch, there is a certain small area where the user needs to hoover mouse and press a certain key on the keyboard simultaneously. However, this is not a major disadvantage because the future users are *PSA*-experts and, to our understanding, often relatively good programmers, and their number is quite small. In addition, creating and editing CETs is only a minor part of the whole modeling and as one gets familiar with the program, the editing is eventually easy, fast and simple.

When adding branch functions to *FinPSA event trees,* the functions could be automatically appended to the code sections linked to the branch. This could reduce the possibility of naming and linking functions incorrectly. However, this automation could cause other errors to the program when implemented incorrectly.

5.5 Other discussion

In our group there were only two people importing the programming code to the new program. We did the importing work in turns and shared the current back up file of the model via Google Drive and that was the *version control system* we used. In the end of the project we got to know that *FinPSA* supports *common projects* shared via network disk. That kind of functionality would be obligatory for larger groups working on wider project even though our methodology worked well in this small group.

In the debugging stage, we had to debug values of several physical variables. Name of the variables are not strictly consistent. Sometimes this results in confusion: user does not know if the variable is for example temperature or mass. Knowing the physical meaning of the variables would simplify the debugging stage: if temperature is negative, one knows right away that the problem is there. In addition, every variable and function could have congruent name structure so that uninitiated person can understand the model efficiently. However, this is not a feature in *FinPSA* but related to the construction of the CET-code.

6 Conclusion

The objectives of the project were “to develop a realistic but simplified example *PRA* model using the *FinPSA*-tool” and “to test the usability and performance of the newly developed *level 2* module for the *FinPSA*-tool”. The first objective is partly met, even though the objective altered during the project due to the restrictions set by the demo version of *FinPSA*. Therefore, the model of the *level 2* is not identical with the original model, described by Okkonen (1995). However, the *level 2* model is integrated to the *level 1* and gives results in some cases. The results of the *level 2* could not be verified completely but the results attained reveals some equalities if compared to the original model.

The challenge in the project was that we could not reconstruct the model that would be equal with the original one. The reason was the restrictions set by the demo version of *FinPSA*. However, during the project course the restrictions were only a minor challenge as the majority of the time was dedicated to the debugging of the model. In the end of the project, we could deduce that the model we constructed is at least partly correct with high probability. The results of the comparable release categories shows that the results of release category 3 are probably correct but the difference in the probability of the release category 2 suggests that some errors might exist.

The errors found were partly caused by our project team. We first constructed the model with “dummy” code and later replaced the code with correct code. This might have been the reason for it that we did not recognize that there are relevant parts of code outside of the model constructed. For example the events not included in the model (due to the restrictions set by the program) had an effect on some of the global variables that was first recognized by the project advisors in the late phase of the project. For example the code in the omitted section RCS depressurization affected global variables and actually differed between the CETs.

The model is still a bit unstable and uncertain. One could improve, verify and validate the model. For now, we hope that the problems in the model will be fixed and the model could be used for educational use and demonstration of *Level 2 PSA*.

The major achievement of this project was the assistance for VTT to debug, test and give suggestions to develop *FinPSA*. We found a severe compiler-level error of *FinPSA* CET-language that was also surprise for VTT. On the other hand, that was the major advantage of this project for the developers – the earlier such bug is found the smaller the consequences become. From this aspect, the project was successful: the developers got user-experiences and development ideas from non-experienced users.

References

Barlow, Richard E., and Frank Proschan. "Importance of system components and fault tree events." *Stochastic Processes and their Applications* 3, no. 2 (1975): 153-173.

Van Campenhout, Jan M., and Thomas M. Cover. "Maximum entropy and conditional probability." *Information Theory, IEEE Transactions on* 27, no. 4 (1981): 483-489.

Drouin, M. *Guidance on the Treatment of Uncertainties Associated with PRAs in Risk-informed Decision Making: Main Report*. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, Office of Nuclear Reactor Regulation, 2009.

O'Hagan, Anthony, Caitlin E. Buck, Alireza Daneshkhah, J. Richard Eiser, Paul H. Garthwaite, David J. Jenkinson, Jeremy E. Oakley, and Tim Rakow. *Uncertain judgements: eliciting experts' probabilities*. John Wiley & Sons, 2006.

Okkonen, Timo. *Development of a parametric containment event tree model for a severe BWR accident.* Finnish Centre for Radiation and Nuclear Safety. Helsinki 1995.

Appendix

A PSA2-Documentation