Mat-2.4177 Seminar on Case Studies in Operations Research

Tools for analyzing epistemic uncertainties in probabilistic risk analysis

Final report

Client: VTT (Technical Research Centre of Finland)

Project team:
Janne Laitonen (Project manager)
Henri Losoi
Markus Losoi
Kimi Ylilammi

Document name: EUPRA003 Final Report version 4

Version 1	Sent to Holmberg and Toppila for comments, 6.5.2013
Version 2	Proposal for the final seminar, 8.5.2013
Version 3	Comments given in the final seminar included, 17.5.2013
Version 4	Comments from Holmberg included, final version, 24.5.2013

Contents

1. Introduction

In this software development project EUPRA (tools for analyzing Epistemic Uncertainties in Probabilistic Risk Analysis), which is carried out under Aalto University's course Mat-2.4177 Seminar on Case Studies in Operations Research, a method for analyzing the epistemic uncertainties in probabilistic risk analysis (PRA) is implemented using interval probabilities. The method is based on the framework presented by Toppila and Salo [1] where they study prioritization of events under interval-valued probabilities. The implementation is done using MATLAB by MathWorks. In this report, the resulting software Basic Events Partial Organizer (BEPO) is called Software.

The client of this project is VTT (Technical Research Centre of Finland) and the project is linked to a joint project PRADA (PRA Development and Application) between VTT and Aalto University School of Science. PRADA is carried out under the Finnish Research Programme on Nuclear Power Plant Safety 2011-2014 (SAFIR2014) whose main funding organizations are the State Nuclear Waste Management Fund (VYR) and VTT [2].

The rest of this final report is structured as follows: This section ends with a description of the objectives for the project. Section 2 gives an introduction to basic terms used in probabilistic risk analysis and in this report. Also dominance is defined following the definitions in [1]. Section 3 describes briefly each of the documents written during the Software development. Section 4 describes Software implementation and section 5 Software testing including validation results. Conclusions and potential targets for future development are given in section 6. Description of the project progress and lessons learned are given in appendix. In addition, this report comes with the following documents that were created during the project: Software quality assurance plan (EUPRA004 SW QA plan) [3], Software requirement specification (EUPRA005 SW Requirement Specification) [4], Software design unit documentation (EUPRA006 SW Units Design) [5] and Software test plan (EUPRA007 SW Test Plan) [6].

1.2. Objectives of the project

VTT gave the following objectives for the project and there was no need to change them during the project:

- Get an understanding about epistemic uncertainty and its modeling with interval probabilities.
- Implement an open source MATLAB package compatible with Finnish PRA software FinPSA for analyzing epistemic uncertainties expressed through interval probabilities.
- Develop the software following good software development practices including the preparation of the following documents: requirements specification, software design specification, testing plan and quality assurance plan. Software validation report including test results must be produced in the end.

Since the implementation is based on the method presented in [1] no selections were needed among different models or methods. In addition, further development of the methodology was given lower importance while high priority was given to high quality documentation and testing. Especially, the quality and completeness of the requirements specification was considered important.

Due to some setbacks with the implementation concerning considerably long computation times, the team had to decrease the comprehensiveness of Software testing. The test cases need to give a strong confidence that Software calculates the dominance relations correctly but the tests on how the model complexity affects the calculation time are for now irrelevant due to the long computation time. Thus, the test cases do not include as large and complex models as originally planned. However, this does not conflict with the original objectives set for the project and actually complex models test rather the computational efficiency than the correctness of the Software.

2. Probabilistic risk analysis and dominance relation

Probabilistic risk analysis (PRA), also called probabilistic safety analysis (PSA) or quantitative risk analysis (QRA), is a systematic methodology for assessing the risks of technical systems and is being widely applied to many sectors, e.g., transport, energy, construction, chemical processing, aerospace, military and even project planning. In many of these areas PRA techniques have been adopted to regulatory framework by relevant authorities, such as STUK (Finnish Radiation and Nuclear Safety Authority) or US. NRC (US. Nuclear Regulatory Commission), and other organizations, e.g., IAEA (International Atomic Energy Agency) and NASA (National Aeronautics and Space Administration). PRA has several applications, e.g., identifying system weaknesses for modifications, allocating the resources for maintenance activities, prioritizing the targets for inspections, determining the risk significance of operational experience or near-misses (precursor analysis), or optimizing the test intervals. [7, 8]

In PRA, usually three basic questions are asked: What can go wrong, how likely that is, and what are the consequences? This leads to triplet definition of (engineering) risk which is quantified by *scenario, probability and consequence*. In order to assess the failure probability for a system, the failure logic is modeled, e.g., using *fault and event trees*. These trees model the failure and sequence propagation in the system starting from the system components that are the *basic events* in the model (i.e., the leaves of a fault tree). The events that may cause a system to fail are called *initiating events* (e.g., loss of electric power) which may result from various *hazards*, e.g., fires, floods, harsh weather conditions, or seismic activity. [7, 9, 10]

The solution of the model can be presented by *minimal cut sets* which are the minimal combinations of events that lead to system failure (so-called *TOP-event*). The resulting TOP-event can be expressed by basic event intersections (AND-logic) and unions (OR-logic), e.g., TOP = A + B * C where '+' and '*' correspond to OR-logic and AND-logic, respectively, and A, B, and C are the basic events (this example contains two minimal cut sets A and B*C). After solving the Boolean logic for the TOP-event, the probabilities and frequencies of the basic and initiating events are used for quantifying the probability (or frequency) of the system failure, e.g., the frequency estimate for a nuclear accident. In addition, the importance of the basic events can then be analyzed using various *risk importance measures* such as Fussell-Vesely and Birnbaum. [7, 9, 10]

In order to construct the fault and event trees and to calculate the Boolean logic for complex systems, many computational software have been developed worldwide, e.g., FinPSA by STUK and VTT [11], RiskSpectrum by Scandpower [12], XFTA [13] as a result of the Open-PSA initiative [14], SAPHIRE by the Idaho National Laboratory [15], RISKMAN by ABS Consulting [16], or CAFTA by OSyS [17]. Many of these softwares (e.g., FinPSA or RiskSpectrum) use probability distributions and Monte Carlo simulation for analyzing uncertainties or for sensitivity studies. In this project another method is implemented: the effect of epistemic uncertainties on basic event dominance relations can be analyzed using interval probabilities.

In general, or at least in Bayesian sense, uncertainty can be presented by *aleatory and epistemic uncertainty*, i.e., uncertainty due to randomness and due to lack of knowledge, respectively. Aleatory uncertainty can be quantified, e.g., by a probability estimate and epistemic uncertainty represents the uncertainty on the value of this estimate. Traditionally in PRA basic event failure probability or frequency represents the aleatory uncertainty while the probability distribution for the failure probability contains the epistemic uncertainty.

The method for analyzing epistemic uncertainties in this project is based on the framework presented by Toppila and Salo [1], where the uncertainty about the basic event probabilities is expressed by intervals. These intervals indicate the knowledge - or the lack of it - on the true values of the model parameters and contain the belief about the plausible range in which the probabilities may locate. Since there is epistemic uncertainty in the event probabilities, the risk importance measures are impacted by this uncertainty. In the method, *dominance relations* for the importance measures are established. One event is said to dominate another if its risk importance measure is at least as high for all event probabilities that are within their respective intervals and strictly higher for some probabilities. In a more mathematical way, event A dominates event B based on the risk importance measure I if and only if

a) $I(A, p) \geq I(B, p)$ for all probabilities p within the intervals and

b) $I(A, p) > I(B, p)$ for some probabilities p within the intervals,

where $I(X,p)$ is the value of risk importance measure I of event X using probabilities p (vector). [1, 18]

For instance, consider a system with TOP = A + B with $p(A) = 0.05$ and $p(B) = 0.01$. Assuming the rare event approximation (only S1-sum included), Fussell-Vesely for A is $FV(A) = p(A)/[p(A)+p(B)] = 0.83$ and for B $FV(B) = p(B)/[p(A)+p(B)] = 0.16$. Thus, $FV(A) > FV(B)$ and A dominates B in terms of Fussell-Vesely. Now, let $p(A) \in [0.01; 0.1]$ and $p(B) \in [0.005; 0.05]$. In order to calculate the dominances, the equation above is rewritten: $FV(A, p) - FV(B, p) \geq 0$ for all p. In this case, this yields to $p(A) - p(B) \geq 0$ for all p. Since this is not true given the intervals above (the intervals overlap), A does not dominate B in terms of Fussell-Vesely. If the epistemic uncertainty can be reduced, the intervals become narrower, say $p(A) \in [0.03; 0.07]$ and $p(B) \in [0.005; 0.025]$. These intervals yield to conclusion that A dominates B in terms of Fussell-Vesely since $p(A) - p(B) > 0$ for all p is true.

Similar reasoning can be performed for larger and more complex systems in order to find out the effect of epistemic uncertainties on basic event dominance relations using interval probabilities. In this project this method is implemented using MATLAB. The detailed description of the method and the algorithm can be found in [1].

3. Software documents

In order to develop Software following good software development practices the following four documents were requested: the quality assurance plan [3], the requirement specification [4], the software design unit documentation [5] and the test plan [6]. In addition, software validation report including test results was required which is included into this final report. This section describes briefly the contents and the purpose for each document mentioned above.

3.1. Quality assurance plan

Software quality assurance plan [3] defines the general quality requirements for the project and Software. The plan sets the top-level quality requirements for the other documents, i.e., requirement specification [4], design specification [5] and test plan [6], and defines for instance the responsibilities and documentation practices as well as version control or backup and security issues. IEEE Standard for Software Quality Assurance Plans [19] was utilized and the requirements defined by VTT were taken into account to construct the quality assurance plan.

3.2. Requirement specification

To ensure understanding between the client and the project team, a requirement specification document [4] was produced. The specification was made in collaboration with the client in an iterative manner. The requirement specification defines requirements and possible future features of Software. Software was then built according to the requirements given in the requirements specification document.

The requirement specification document defines the purpose of Software, its possible users, main quality attributes, overall requirements (e.g., the functions of the software at a general level), user interface and the requirements for documentation. The main quality attributes of Software were defined to be openness, efficiency, maintainability and reliability. These attributes are described in the requirement specification [4].

3.3. Software design unit document

The software design unit document [5] presents an overview of the source code units and introduces Software briefly. The document contains step-by-step instructions how to install Software and how to run a simple example. Therefore, it is recommended that a new user of Software would read this document before using Software.

3.4. Test plan

Test plan [6] outlines the testing techniques and specifications. The algorithm for which a Matlab package was done is presented in [1]. The testing techniques such as unit-testing and system-testing are ways to guarantee, e.g., software integrity, correctness and consistency. Besides simple runs and comparisons, graphical visualization is utilized to understand the abundance of data: for example dominance matrices are visualized over the whole confidence interval. Most importantly, the test plan lists the test cases used for Software validation.

4. Software implementation

Software is used as an extension for FinPSA, which provides input data for Software. This input data contains basic events, their probability distributions and minimal cut sets. In addition to these, Software will also require user to input a confidence level to be used. Software takes minimal cut sets as given and does not consider whether they are logical or reasonable.

Software supports computation of dominance relations for basic events. These dominance relations are based on risk importance measures whose differences are expressed as multilinear functions. The computational framework of the software is based on the research paper [1].

Software consists of five modules: FinPSA parser, Dominance relations calculator, MLF (Multilinear function) kernel interface, Listlist implementation and SCT (Symbolic Computation Toolbox) implementation. The system pipeline can be seen in figure 1 below.

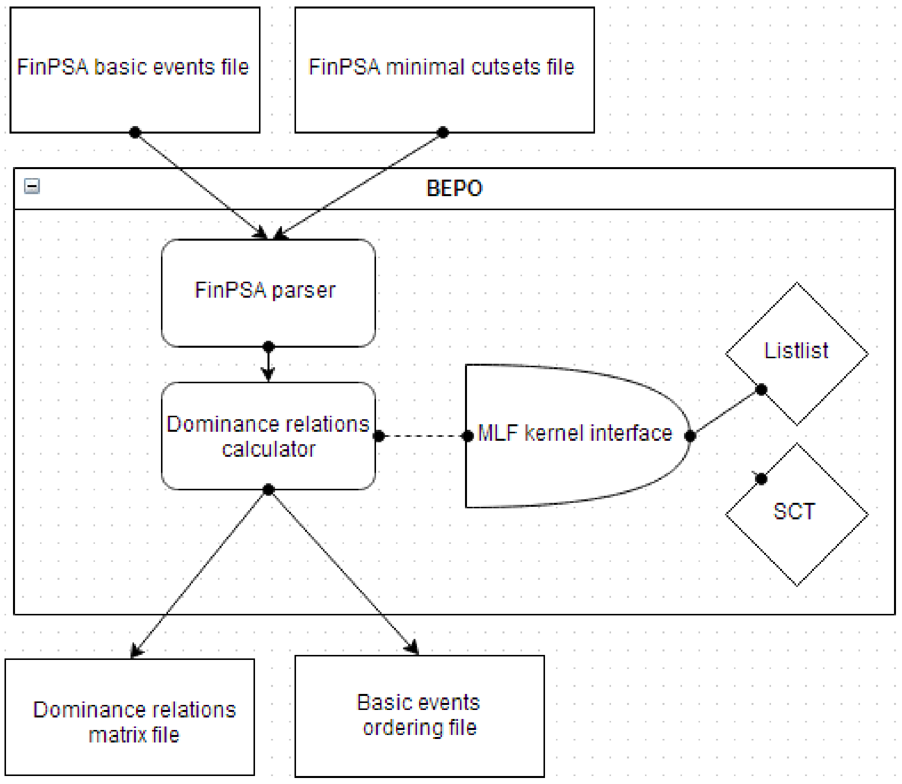
FinPSA basic events file
FinPSA minimal cutsets file
BEPO
FinPSA parser
Dominance relations calculator
MLF kernel interface
Listlist
SCT
Dominance relations matrix file
Basic events ordering file

Figure 1: BEPO software units and the input-output pipeline. Data flows downwards in the figure. Dominance relations calculator uses MLF kernel interface to perform computations with multilinear functions. Listlist and SCT are implementations of this interface, and the current version of the software utilizes only Listlist.

As can be seen in the figure 1, Software takes two input files: Basic events file and minimal cut sets file. These files are then handled in FinPSA parser, and the output of the parser is passed to Dominance relations calculator. Finally, dominance relations, order of the basic events and their intervals are printed on the command prompt and/or into two user defined files.

Software is divided into separate modules to ensure easy maintenance. The modules are made in such a way that they have as little as possible dependencies on the other modules to be able to switch or replace modules with the minimal effort. For example, adding a support for a new input file type can be made by creating a new parser for it and then simply switching from FinPSA parser to the new parser.

4.1. FinPSA parser

FinPSA parser transforms input files into formats which can be used in Dominance relations calculator. The parser supports two FinPSA input files: basic events files and minimal cut sets files. The parser also takes a confidence interval as an input parameter to be able to calculate the upper and the lower bounds for the probability intervals of basic events from the distributions contained in the basic events file.

The parser was programmed in such a way that new features could be added easily. Currently the parser extracts only necessary information from the input files, such as the names of basic events, their probability distributions and the parameters of these distributions. However, new variables and new files can be added with ease.

4.2. Multilinear function handling procedures

Dominance relations calculator depends on the algorithm presented in [1]. This algorithm is based on symbolic computations with multilinear functions, and in order to perform these computations with Matlab, a module was created. This module implements MLF kernel interface (explained below) and was named Listlist because it stores a multilinear function as two nested lists. This implementation is slow because the terms of a multilinear function are handled in loops, which are known to be slow in Matlab.

To address efficiency problems of Listlist implementation, SCT implementation was developed. SCT implementation uses Symbolic Math Toolbox™ [20] to perform symbolic computations with multilinear functions. However, SCT implementation turned out to be slower than Listlist implementation. Because of this, Software uses Listlist implementation as default.

Multilinear function handling procedures are made so that the handling procedure can be switched easily. MLF kernel interface is an abstraction layer, which enables Software to have changeable multilinear function handling procedures. More detailed description of MLF kernel interface and its implementations can be found in the software unit design documentation [5].

4.3. Dominance relations calculator

Dominance relations calculator computes a dominance matrix by performing pair-wise dominance comparisons between basic events in the order specified by the output of the FinPSA parser. The result of a dominance comparison can be based on four different reasons. Dominance relations are irreflexive, asymmetric and transitive, and these three properties can be utilized in dominance comparisons. However, at this point of development, Software does not utilize transitivity property. If all of these three properties are inapplicable in a dominance comparison, the algorithm and proposition 1 in [1] are utilized.

5. Software testing and validation

Software testing consists of unit-testing and system-testing following the software test plan and quality assurance plan. Several unit and system tests were performed during the implementation as well as with the final version of Software in order to validate the code. These tests are listed in the test plan and below. Besides the unit and system test cases, infographics such as dominance matrix visualization were created and are illustrated in the test plan. In addition, this section contains a brief Software performance analysis to pinpoint some possible areas for future performance improvement.

5.1. Test cases

The unit test-cases are listed below

ID	**Files**	**Description**	**Manually tested & validated**
[U1]	test_mlfadd_case_1.m	First addition test.	Yes
[U1]	test_mlfadd_case_2.m	Second addition test.	Yes
[U2]	test_mlfeval_case_1.m	First test about the evaluation of multilinear function at a point.	Yes
[U2]	test_mlfeval_case_2.m	Second test about the evaluation of multilinear function at a point.	Yes
[U3]	test_mlfponterms_case_1.m	First test about factorization and deducing negative terms and positive terms.	Yes
[U3]	test_mlfponterms_case_2.m	Second test about factorization and deducing negative terms and positive terms.	Yes
[U4]	test_mlfsmult_case_1.m	Test about multiplying a multilinear function with a scalar.	Yes
[U5]	test_mlfsubst_case_1.m	Test about reducing the multilinear function with some substitutions.	Yes
[U6]	test_jlpsk_or_case_1.m	First dominance test.	Yes
[U6]	test_jlpsk_or_case_2.m	Second dominance test.	Yes
[U6]	test_atj_7_case_1.m	Third dominance test that tests the seven case example from [1].	Yes
[U6]	test_atj_7_case_2.m	Fourth dominance test that tests the seven case example from [1].	Yes

where every test belong to the unit tester and the identifier such as [U1] and [U6] refers to the SW Test plan [6]. Next the system tests are listed below

ID	Files	System (MCSs)	Bounds	Manually tested & validated
[S1]	D_TestCase1_BE_Evendist_v2.txt_TestCase1_MCS.txt_B D_TestCase1_BE_Evendist_v2.txt_TestCase1_MCS.txt_FV	A + B	[0.010,0.100], [0.005,0.050]	Yes
[S1]	D_TestCase2_BE.txt_TestCase2_MCS.txt_B D_TestCase2_BE.txt_TestCase2_MCS.txt_FV	A + B + C + D	[0.030,0.060], [0.020,0.035], [0.015,0.025], [0.009,0.011]	Yes
[S1]	D_TestCase3_BE.txt_TestCase3_MCS.txt_B D_TestCase3_BE.txt_TestCase3_MCS.txt_FV	A + B * C	[0.010,0.030], [0.010,0.030], [0.015,0.025]	Yes
[S2]	D_TestCase4_BE_Evendist_[001-003].txt_TestCase4_MCS.txt_FV_ D_TestCase4_BE_Evendist_[001-003].txt_TestCase4_MCS.txt_B_	Minimal cut sets can be found in [1].	[0.010,0.030] for each of the seven basic events	Yes
[S2]	D_TestCase4_BE_Evendist_[001-005].txt_TestCase4_MCS.txt_FV_ D_TestCase4_BE_Evendist_[001-005].txt_TestCase4_MCS.txt_B_	Minimal cut sets can be found in [1].	[0.010,0.050] for each of the seven basic events	Yes

Finally, Software was tested with more resource-intensive cases and other system tests given below

ID	Files	Bounds	In the unit tester	In the system tester	Validated
[S3]	toppila_RHRBASIC.TXT toppila_RHRMCS.TXT	See [1], table III.	No	No	Yes
[S4]	TestCase INT-TF BE lognormal etal.txt TestCase INT-TF BE point value.txt TestCase INT-TF MCS.txt	Uncertainty distributions from FinPSA using different confidence intervals and point values.	No	No	Yes
[S5]	TestCase Floating BE.txt TestCase Floating MCS.txt	[0.01,0.10], [0.02,2], [0.0299...,1.4999...], [0.05,1.6666...]	No	Yes	Yes

The simple test cases [S1] and [S5] were created and manually calculated by the team in order to compare the results and assure that the resulting dominance matrices are equivalent. Test cases [S2] and [S3] were compared to the results presented in [1]. Test case [S4] was compared to the results calculated by FinPSA (more information is given below in the following subsection). It was concluded that Software calculated the correct results in every test case.

5.2. Test case INT-TF [S4]

In order to test that the software is able to read various inputs taken from FinPSA, a PRA model of a nuclear power plant was utilized. The test case represents one accident sequence initiated by a loss of feed water resulting to loss of reactivity control during power operation. The case contains 16 minimal cut sets and 11 basic events including common cause failures. The uncertainty distributions for basic event failure probabilities consist of LogNormal, Beta and Gamma distributions. First, the case was utilized as such in order to make sure that Software is able to read the input, calculate the probability intervals given a confidence level and calculate the dominance matrix. The resulting matrix could not be validated due to missing reference matrix though.

Second, the dominances were calculated using point values for failure probabilities (expectation values taken from FinPSA input). Since with point values the dominance relations must correspond the ranks defined by risk importance measures, the dominance matrix could be validated by comparing it to the risk importance measures calculated with FinPSA. The resulting dominance matrix, dominance ranking, lower and upper bounds for failure probabilities as well as Fussell-Vesely importance measures, expectation values for failure probabilities and Fussell-Vesely ranking can be seen below. As seen, the probability intervals correspond the expectation values and the dominance ranking corresponds the Fussell-Vesely ranking.

Software (BEPO) output															**FinPSA output**		
Basic event	222JARH...Z5/121	222JARJ...Z5/121	314V002V1AZ-8/14	314V0XXV1JZ	354JARJ...Z-5/14	516.S54R.BZ-ABC	516.S54R.BZ-ABCD	516.S54R.BZ-ABD	516.S54R.BZ-ACD	516.S54R.BZ-BCD	INT-TF	**Dominance ranking**	**Lower bound**	**Upper bound**	**Fussell-Vesely**	**Probability**	**FV ranking**
222JARH...Z5/121	0	1	0	1	0	1	0	1	1	1	0	**5**	5.98E-06	5.98E-06	7.28E-02	5.98E-06	**5**
222JARJ...Z5/121	0	0	0	1	0	0	0	0	0	0	0	**7**	6.40E-07	6.40E-07	7.79E-03	6.40E-07	**7**
314V002V1AZ-8/14	1	1	0	1	1	1	1	1	1	1	0	**2**	1.09E-05	1.09E-05	9.94E-01	1.09E-05	**2**
314V0XXV1JZ	0	0	0	0	0	0	0	0	0	0	0	**8**	6.50E-08	6.50E-08	5.93E-03	6.50E-08	**8**
354JARJ...Z-5/14	1	1	0	1	0	1	1	1	1	1	0	**3**	2.88E-05	2.88E-05	3.51E-01	2.88E-05	**3**
516.S54R.BZ-ABC	0	1	0	1	0	0	0	0	0	0	0	**6**	5.80E-06	5.80E-06	7.06E-02	5.80E-06	**6**
516.S54R.BZ-ABCD	1	1	0	1	0	1	0	1	1	1	0	**4**	2.35E-05	2.35E-05	2.86E-01	2.35E-05	**4**
516.S54R.BZ-ABD	0	1	0	1	0	0	0	0	0	0	0	**6**	5.80E-06	5.80E-06	7.06E-02	5.80E-06	**6**
516.S54R.BZ-ACD	0	1	0	1	0	0	0	0	0	0	0	**6**	5.80E-06	5.80E-06	7.06E-02	5.80E-06	**6**
516.S54R.BZ-BCD	0	1	0	1	0	0	0	0	0	0	0	**6**	5.80E-06	5.80E-06	7.06E-02	5.80E-06	**6**
INT-TF	1	1	1	1	1	1	1	1	1	1	0	**1**	2.70E-01	2.70E-01	1.00E+00	0.27	**1**

5.3. Performance analysis

The performance of Software was analyzed by solving Fussell-Vesely dominances for four different systems. These computations were performed under Windows 7 operating system with a laptop that contains an Intel Core i5 2430M (dual-core) processor and 4096 Mbytes of DDR3 random-access memory.

The computation times of performance analysis tests are listed on the table 1. From this table it can be seen that Listlist implementation outperforms SCT implementation significantly. This can result from at least two different reasons. First, Symbolic Math Toolbox was a new tool for the project group, and, thus, the toolbox may not have been utilized optimally. Second, the toolbox supports many symbolic computation operations (e.g., integration) that were not needed in this project. I.e., the support of unnecessary operations may render the toolbox too complex and, thus, too slow for this project.

Table 1: The computation times of Software when Fussell-Vesely dominances were solved for four different systems. These computation times were recorded for both MLF kernel implementations.

			MLF kernel implementation	**MLF kernel implementation**
System	**Basic events**	**Minimal cut sets**	**Time (Listlist) [s]**	**Time (SCT) [s]**
A+B*C	3	2	0.096124	0.975192
A+B+C+D	4	4	0.204339	5.075722
The seven-component example in [1] with probability intervals [0.01,0.03]	7	8	17.969800	339.049813 (5.65 min)
The seven-component example in [1] with probability intervals [0.01,0.05]	7	8	25.538970	1670.360805 (27.83 min)
RHR system in [1]	31	147	15368.96 (4.27h)	N/A (would last too long)

The dominance relations computations of the residual heat removal system (RHRS) were taken into closer inspection because it proved to be the most demanding case. This case also demonstrated that Software is too slow for practical uses and is severely beaten by the Mathematica implementation utilized in [1]. With Listlist implementation Software computed the correct Fussel-Vesely dominances in 4.27 hours while the Mathematica implementation in [1] is reported to perform these computations in less than 15 seconds.

The figure 2 shows the time consumption of the six most time-consuming functions of Software when it was tested with Listlist kernel by solving the Fussel-Vesely dominance relations of the RHR system in [1]. The figure shows that domrels() is the most time-consuming function. This is an expected result as all other functions listed in the figure 2 are being called by it. However, this function is called only once as can be seen on the table 2.

The table 2 shows that mlfnneg() is being called over 5 million times. The huge number of function calls may be explained by the facts that the function is recursive and its time complexity is exponential in terms of basic events. The total time consumption of the function is also the second highest of all the functions in the figure 2, over 3 hours. Because this function is called so many times and it depends on the functions in the MFL kernel interface, even small optimizations of MLF kernel implementations would increase the overall performance enormously. Also, the function mlfsubst() is called over 3 million times and the function mlfeval() 18 million times.

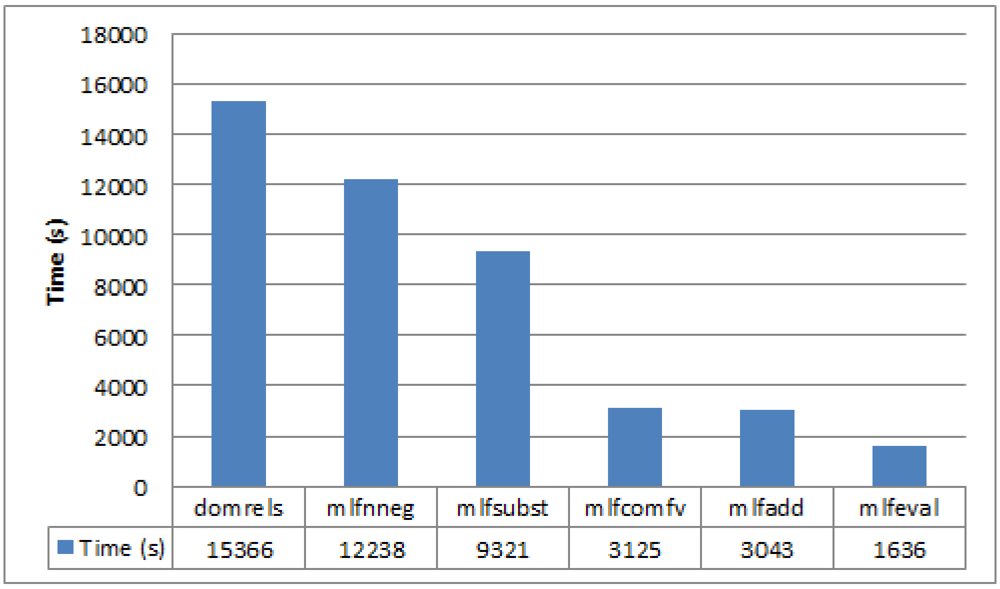

Figure 2: The six most time-consuming functions of Software when it was tested with Listlist kernel by solving the Fussel-Vesely dominance relations of the RHR system in [1]. The functions are described in [5].

Table 2: The number of function calls of the functions in the figure 2.

	domrels	**mlfnneg**	**mlfsubst**	**mlfcomfv**	**mlfadd**	**mlfeval**
Number of function calls	1	5160767	3466290	783	783	18024282

6. Conclusions

In this software development project a method for analyzing the epistemic uncertainties in probabilistic risk analysis was implemented by four students from Aalto University School of Science. The method is based on the framework presented by Toppila and Salo [1] where they study prioritization of events under interval-valued probabilities. The implementation was done using Matlab. The client of the project was VTT.

The team was able to produce all the required deliverables and develop the requested Matlab tool. The documentation consists of project plan, interim report, final report including software validation report, software quality assurance plan, requirement specification, design specification, and test plan. Each one of these documents was briefly described in this final report. The implemented software is able to calculate the dominance relations as described in [1], output the dominance matrices and is compatible with Finnish PRA software FinPSA as was required by the client. The implemented software was tested and validated using various test cases described and documented in the software test plan and in this report.

The biggest setback was that the implementation is computationally too slow for practical purposes. The team recognized some potential improvements that may be considered in the possible future development. These improvements to enhance the computational performance of the software are discussed below.

Dominance relations calculator can be made faster in multiple ways. The biggest bottlenecks were identified as being the multilinear functions handling procedures. These procedures could be improved by representing multilinear functions in a more efficient manner. For example, multilinear functions could be expressed as matrices. This could be beneficial because Matlab is intended to be fast on matrix computations. Also, current implementations, SCT and Listlist, could be improved by programming them in a more efficient manner. For example, the expertise of Symbolic Math Toolbox was low in the project team and the time available was too short for gaining enough expertise. Thus, SCT implementation could be improved by performing a comprehensive study on all the methods that Toolbox provides to perform symbolic computations with multilinear functions.

Reasoning behind dominance relations in the dominance relations calculator are currently based on irreflexivity, nonnegativity [1] and asymmetricity. The implementation could also benefit of using transitivity based reasoning. This would reduce the amount of calls made to multilinear functions calculator and, thus, making the implementation more efficient. Transitivity property could be utilized in conjunction with risk measure point estimates that FinPSA computes. Currently, Software performs dominance comparisons in the same order that basic events are listed in FinPSA input. Instead of this order, dominance comparisons could be performed in descending importance order specified by risk measure point estimates. In this way, Software may find dominances (ones in a dominance matrix) more rapidly, and, therefore, utilize asymmetricity and transitivity chains (e.g., if A dominates B and B dominates C, then A dominates C) more frequently.

It is possible that Mathematica by Wolfram offers better computational performance for symbolic computation compared to MATLAB by MathWorks. Therefore, it is advised to consider which software to use in the future. In addition, the method might be developed to take into account the correlations between different parameters instead of treating them completely independent: For instance, if the failure probability of one component is considered to locate near the upper bound of the confidence level, identical components should behave in a similar manner.

References

[1] Toppila A, Salo A. A Computational Framework for Prioritization of Events in Fault Tree Analysis under Interval-valued Probabilities, IEEE Transactions on Reliability, 2013 (to appear).

[2] SAFIR2014, The Finnish Research Programme on Nuclear Power Plant Safety 2011-2014, Interim Report.

[3] Software quality assurance plan, EUPRA004 SW QA Plan

[4] Software requirement specification, EUPRA005 SW Requirement Specification

[5] Software design unit documentation, EUPRA006 SW Units Design

[6] Software Test plan, EUPRA007 SW Test Plan

[7] Bedford T, Cooke R. Probabilistic Risk Assessment: Foundations and Methods. Cambridge University Press, 2003.

[8] Himanen R, Julin A, Jänkälä K, Holmberg J-E, Virolainen R. Risk-Informed Regulation and Safety Management of Nuclear Power Plants—On the Prevention of Severe Accidents, Risk Analysis, Vol. 32, Issue 11, pp 1978–1993, November 2012.

[9] Modarres M. Risk Analysis in Engineering - Techniques, Tools, and Trends. Taylor & Francis Group, 2006.

[10] Hoyland A, Rausland M. System Reliability Theory – Models and Statistical Methods. John Wiley & Sons, 1994.

[11] http://www.stuk.fi/ydinturvallisuus/ydinvoimalaitokset/en_GB/finpsa/, viewed 2.5.2013

[12] www.riskspectrum.com/, viewed 2.5.2013

[13] http://www.lix.polytechnique.fr/~rauzy/xfta/xfta.htm, viewed 2.5.2013

[14] http://www.open-psa.org, viewed 2.5.2013

[15] https://saphire.inl.gov/, viewed 2.5.2013

[16] http://www.absconsulting.com/riskman.cfm, viewed 2.5.2013

[17] http://www.o-sys.com/products-services/fault-tree-analysis/, viewed 2.5.2013

[18] Holmberg J-E et al. PRA development and application (PRADA). SAFIR2014 Interim Report. 2013.

[19] IEEE Standard for Software Quality Assurance Plans, IEEE Std 730-1998.

[20] http://www.mathworks.se/products/symbolic/, viewed 20.4.2013.

Appendix: Progress of the project and self assessment

In general, the project advanced fairly well and, for the most part, the team was able to fulfill the objectives set for the project. The team was able to produce all the required deliverables and develop the requested Matlab tool. Producing and maintaining a good quality in the documentation consisting of project plan, interim report, final report including software validation report, software quality assurance plan, requirement specification, design specification, and test plan required quite a lot of resources in addition to the software implementation and testing. Some delays did occur but mostly they had only small influences on the overall progress of the project. These delays were mainly due to underestimated resources needed for the software implementation and heavy workload for the team in terms of work and exams.

The biggest setbacks were that the implementation is computationally too slow for practical purposes and that the test results were documented at the last moment causing quality reductions. Possible reasons for these difficulties, on the project manager's opinion, were intensive schedule that did not allow more time to be used for developing a better implementation and some challenges in working as a group.

The team had meetings approximately once per week having almost 15 meetings in total. These meetings usually consisted of going through the status of the project, dividing the tasks and solving many of the problems together. In addition, to ensure that the project evolved into the right direction the team met the client five times. These meetings had a pre-planned agenda, e.g., commenting the document drafts or demonstration of Software and usually set up some deadline for the team. The team also met professor Salo once at the beginning of the project. The project manager wrote a memorandum (in Finnish) on each meeting with VTT and professor Salo in order to make sure that everyone in the team has the same information available. In addition to meetings, the team communicated using e-mails and, e.g., the project manager received almost 250 e-mails during the project.

Working as a group turned out to be more challenging than expected. A significant amount of time in meetings of the project team was spent in discussing about irrelevant issues. This could have been alleviated by having a very strict schedule for the meetings. Also work effort between the members of the group varied. This was mainly due to the fact that the team members were specialized into different parts of the project and different tasks turned out to be more time consuming than was predicted.

Not all members of the group had taken the risk analysis course which was recommended for this project. Therefore, the suitable background knowledge of the team members was not at the same level. In addition, team members had different expectations and quality requirements for the project. This increased work effort on some of the team members.

It was intended that Software documents would ensure good quality of the software, but producing them was a significant effort that consumed a considerable amount of time from the actual software development. However, the project educated how to calculate and store symbolic computations. The project was also an example how software projects are done in VTT.

The project manager acknowledges the comments above written by the team member(s). The project was somewhat challenging for several reasons: First, the initial knowledge on risk analysis varied a lot among the team members so especially in the beginning it was a challenge to make sure that everyone understand the basic terms in a similar way. Second, the schedule and strict deadline did not enable to change the chosen course in the implementation radically wherefore, perhaps, for instance the resulting Software is rather slow. The team members did not find writing the several documents very appealing, especially when the implementation required more resources, neither which caused some difficulties in motivating the group to complete them. Also the limited use of FinPSA (only the project manager had the possibility to use it) caused some limitations. Lastly, there were sometimes major difficulties in working as a group and making the team meetings efficient instead of debating on irrelevant opinions or trying to distribute the tasks and the workload equally among the team members. Nevertheless, it must be stated that the whole group was delightfully ready to have the meetings in the

evenings after working hours or during weekends. Also the active steering by Jan-Erik Holmberg and Antti Toppila as well as professor Salo was much appreciated.