U.S. DEPARTMENT OF ENERGY | Nuclear Energy

Office Of Nuclear Energy Sensors and Instrumentation Annual Review Meeting

(A Method for Quantifying the Dependability Attributes of Software-Based Safety Critical Instrumentation and Control Systems in Nuclear Power Plants)

(Carol Smidts)
(The Ohio State University)

(NEET 2)

October 28-29, 2015

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Project Overview

Goal, and Objectives

Develop measures and methods to assess dependability attributes early and throughout the life-cycle process of software development

Participants

- University PI: Dr. Carol Smidts, The Ohio State University (Started February 1, 2014)
- Industry PI: Mr. Ted Quinn, Technology Resources (Started February 1, 2014)
- Postdoctoral researcher: Dr. Fuqun Huang, The Ohio State University (Started June 1, 2014)
- PhD Students: Xiang Li, The Ohio State University (Started May 20, 2014)
- PhD Students: Boyuan Li, The Ohio State University (Started Aug 20, 2014)

Project Overview (cont'd)

Schedule

Tasks	Date
Kick-off meeting	April 1 to May 15, 2014
Elicit the causal map describing the dependencies between dependability attributes	May15 to July 15, 2014
For each dependability attributes, elicit the causal map describing occurrence of the event of interest	May 15 to August 31, 2014
Relate measurable concepts to each concept in the event of interest level	August 31 to December 31, 2014
Assessing Coverage	December 31, 2014 to January 31, 2015
Developing Missing Measures	January 31, 2015 to June 31, 2015
Experimental Evaluation	June 31, 2015 to December 31, 2015

Accomplishments

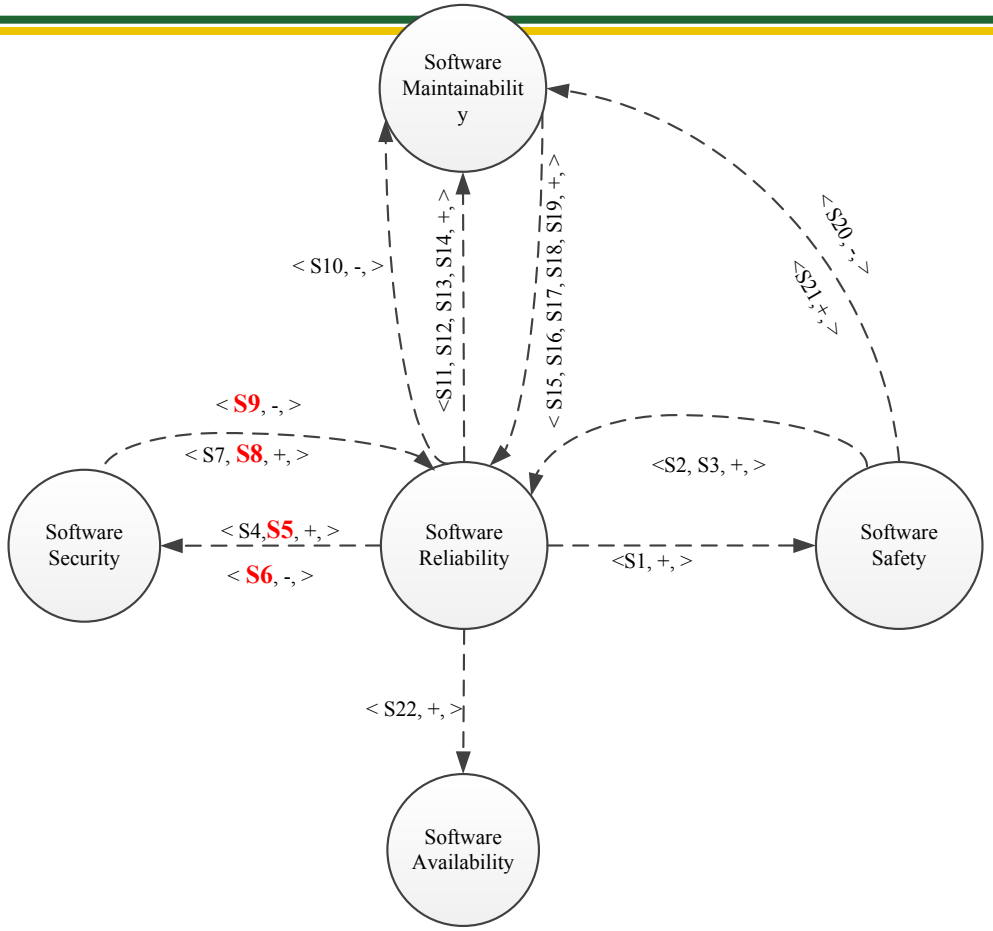
- **Designed a new notation system, Causal Mechanism Graph, to capture relationships between software dependability attributes**
  - *Data Collection based on expert opinion elicitation*

  - More than 600 experts were identified, 54 were selected based on their relevant publications demonstrating knowledge in at least two dependability attributes.
  - The expert selection procedure was inspired from the knapsack problem.
  - A series of semi-structured questionnaire was designed to elicit their knowledge.

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Accomplishments (cont'd)

The dependencies between software dependability attributes

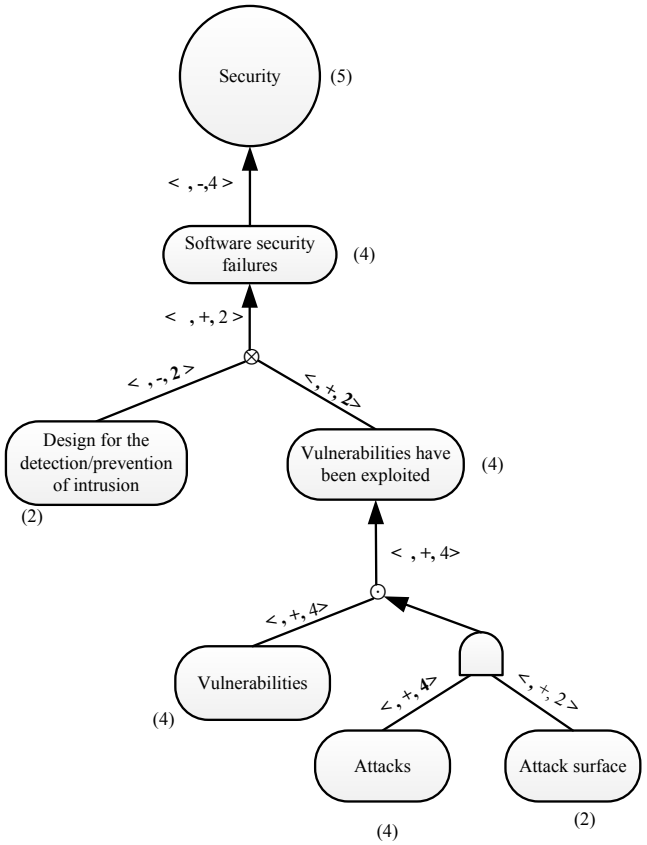


Accomplishments (cont'd)

- **For each dependability attribute, elicit the causal mechanism graph describing occurrence of the event of interest**
  - Experts' responses to the questionnaires also contain detailed information on the causal factors that result in failures of the dependability attributes. For instance, software security failures are caused by the factors shown in the figure in the next slide.
  - The method used to extract the causal failure mechanisms includes:
    1) Merging of the individual causal maps related to a particular dependability attribute;
    2) Slicing of the map which retains only consensus concepts and relations.

Accomplishments (cont'd)

- **Consensus causal mechanism graph (Example: for software security)**



Accomplishments (cont'd)

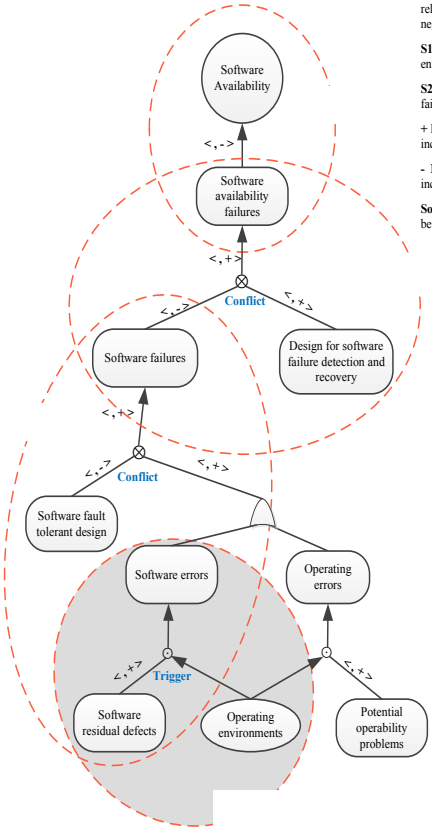
- **Relate measurable concepts to each concept in the event of interest level**
  - Identify measureable characteristics and corresponding measures for the outcome of interest associated with each software dependability attribute
    - Based on the causal map for each dependability attribute, questionnaires are designed to elicit experts' opinions on the measurable concepts and corresponding measures for each event of interest. For instance, a measureable concept for software security is "vulnerability", and the experts are asked to provide the measures for "vulnerability".
    - The next four slides provide the frameworks used to elicit measures and example results for software availability, software safety, software security and software maintainability.

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Accomplishments (cont'd)

Relate measurable concepts to each concept in the event of interest level (cont'd)

Entity class	*Software residual defects*			
Types				
Attributes	*Origin*	*Impact* (Expert_Rich, Matias)	*Amount* (Expert_Yennun, M)	*Density* (Expert_Miroslaw, Matias)
Measures	Base measure: 1) stages of software development 2) structural or functional components	(Expert_Rich) **Cost impact**: sum of defects number at different development stages (Expert_M) with different weights: earlier defects weight more. **System impact**: impact levels to the system, derived from the density and origin in terms of components (Expert_Matias).	(Expert_ M) The number of defects remaining in the software after release	(Expert_Miroslaw, Matias) The number of defects left in the program per lines of code (f/LOC).
Measurement Approaches	….	….	….	
Measurement instruments	….	….	….	
Causal factors	….			
Correlative factors	….			



Notes

< arg1, arg2 > Describes the scenario for which either a positive or a negative relation is present. Arg1 represents the scenario, while arg2 represents the positive or negative relation.

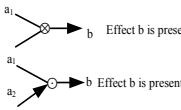
S1: The scenarios/contexts/circumstances/pre-conditions under which operating environments trigger software residual defects.

S2: The scenarios/contexts/circumstances/pre-conditions under which software failures are not detected or recovered.

+ Positive influence: a "positive" influence is said to exist from A to B when an increase in A leads to an increase in B, and a decrease in A leads to a decrease in B.

- Negative influence: a "negative" influence is said to exist from A to B when an increase in A leads to a decrease in B, and a decrease in A leads to an increase in B.

Software Availability Failure: the inability of a software system or component being operational and accessible when required for use.



Operability: the capability of the software component to enable the user (system developer) to operate and control it.

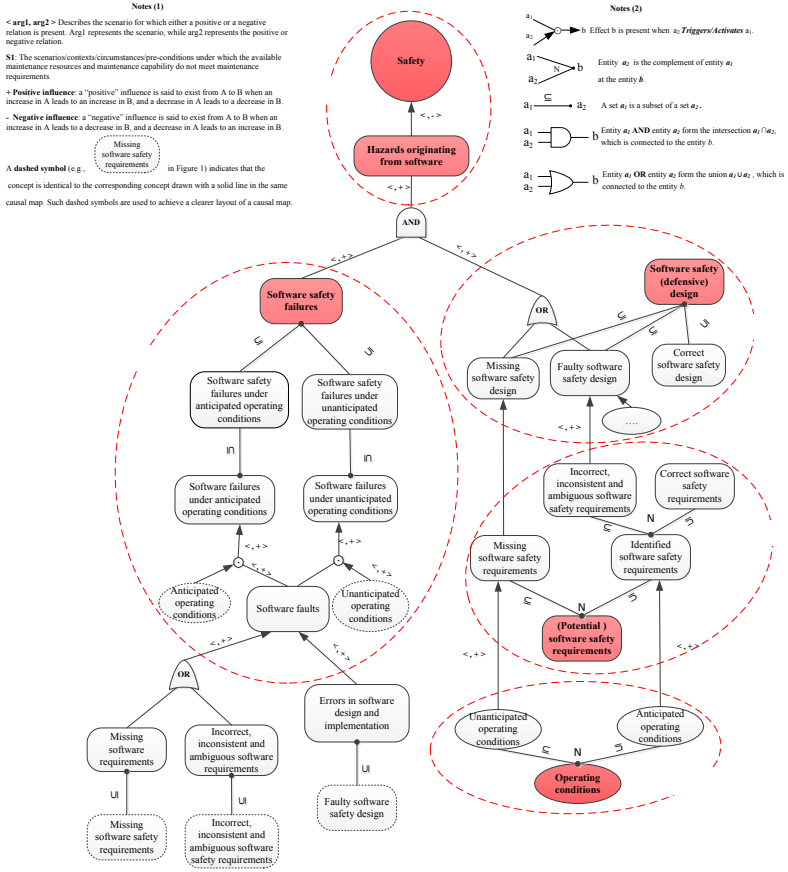
- Eliciting measures for software availability

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Accomplishments (cont'd)

Relate measurable concepts to each concept in the event of interest level (cont'd)

Entity class	Software safety requirements[2]		
Entity sub-class/ Entity/ Types	Requirements are the statement of the problem to be solved. Requirements differ from the system specification in that the specification is the solution to the problem stated by the requirements. For software safety requirements, the problem derives from the application domain. Determining the requirements is thus the primary responsibility of the domain experts, although the statement of the problem (the requirements) has to admit the possibility of a solution (the specification). Thus, computer engineers need to be consulted to ensure that this circumstance is possible. Any subclasses that might exist in the area of requirements are only going to be visible to the domain experts. One can speculate about topics such as incompleteness (errors of omission) but the determination will have to rest with the domain experts.		
Attributes	***Attribute #1:*** Completeness	***Attribute #2:*** Consistency	***Attribute #N:*** Accuracy
Measures	Degree of belief that stated requirements are complete Formal models of requirements	Degree of belief that stated requirements are consistent Formal models of requirements	Degree of belief that stated requirements are accurate Formal models of requirements
Measurement Approaches	Expert judgment Proof of the absence of faults to the extent that the requirements can be modeled in a formal model	Expert judgment Proof of the absence of faults to the extent that the requirements can be modeled in a formal model	Expert judgment Proof of the absence of faults to the extent that the requirements can be modeled in a formal model
Causal factors or mechanisms	Human error, because the determination of requirements is informal and largely lacking in any form of mechanical analysis or assessment. This limitation is fundamental, because a formal statement of requirements relies upon an interpretation of the associated logic, and this interpretation further relies upon the meanings that humans give to terms and phrases in natural languages.		
Correlative factors			



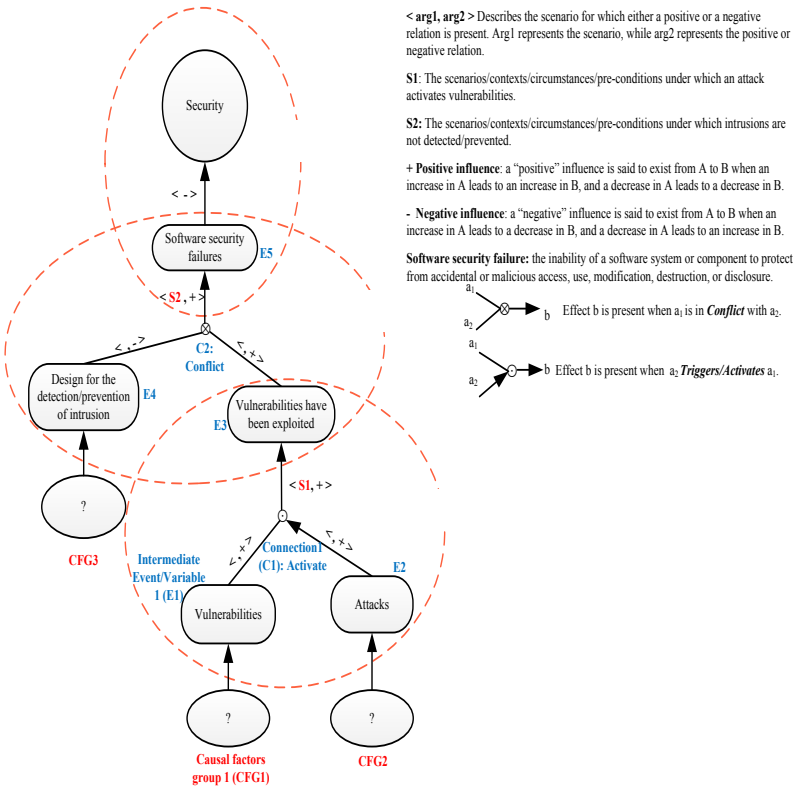
- Eliciting measures for software safety

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Accomplishments (cont'd)

Relate measurable concepts to each concept in the event of interest level (cont'd)

Entity class	*Attacks*		
Entity sub-class or Entity (Types)	*Attack*		
Attributes	Likelihood	Difficulty	Impact
Measures	1. Probability of an attack; 2. Probability of it succeeding (in Exploiting a vulnerability); 3. Probability of it failing; 4. The number of observed attempts to exploit a known software vulnerability	1. Difficulty level; 2. Time required to carry the attack; 3. Resource required to carry the attack; 4. Accessibility level (insider, remote) for the attacker	1. Number of people and systems affected; 2. Severity (derived measure), S(A) = normalization [i(C) + i(I) + i(V)]; 3. Cost of the attack for the attacker
Measurement Approaches	Potential attacks can be measured in terms of available exploits using public sources of exploits, e.g., the Metasploit DB; Attacks that already happened can be measured by consulting published statistics about security incidents, e.g., from CERT		
Measurement instruments	Public sources of exploits, e.g., the Metasploit DB; Published statistics about security incidents, e.g., from CERT		
Causal factors	1. Malicious motivation 2. …….		
Correlative factors	Number of times it has actually taken place in the past (and succeeded or failed).		
Analyzer's summary	These measures are very clear and useful for attack assessment.		



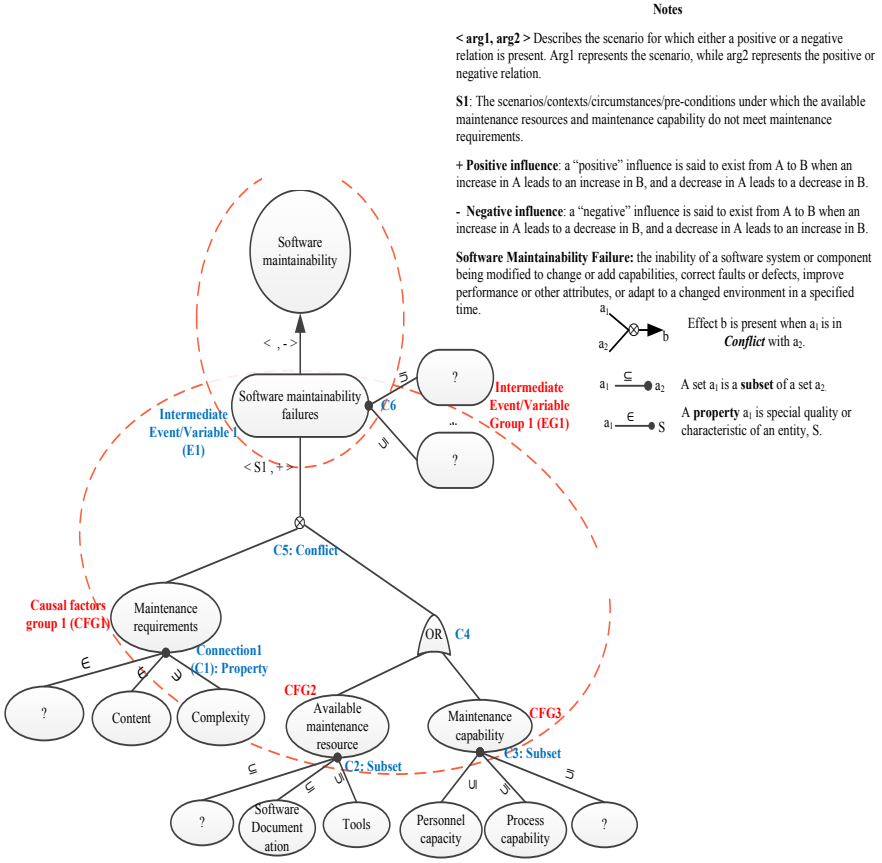
- Eliciting measures for software security

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Accomplishments (cont'd)

Relate measurable concepts to each concept in the event of interest level (cont'd)

Entity class	*maintenance capability*		
Entity sub-class or Entity (Types)	Tool capability	Staff capability	
Attributes	Availability/successful usage of tools	Experience	Compatibility of maintenance process established
Measures	. code complexity • change impact • re-engineering • Regression • testing • Defect management	• Years of experience with the technology of the maintained system • Years of experience with the maintained system • Years of experience with the specific role in the maintenance project • Years of experience with the software development tools used	The degree to which the maintenance organization's established processes are compatible with the specified or actual maintainability
Measurement Approaches	1. code complexity analysis: complexity measurement 2. ….		
Measurement instruments			
Causal factors	…..		
Correlative factors			
Analyzer's summary			



- Eliciting measures for software maintainability

Accomplishments (cont'd)

Assessing Coverage

- The coverage of measures for each software dependability attribute is assessed at three levels:
  - ***Attribute level***

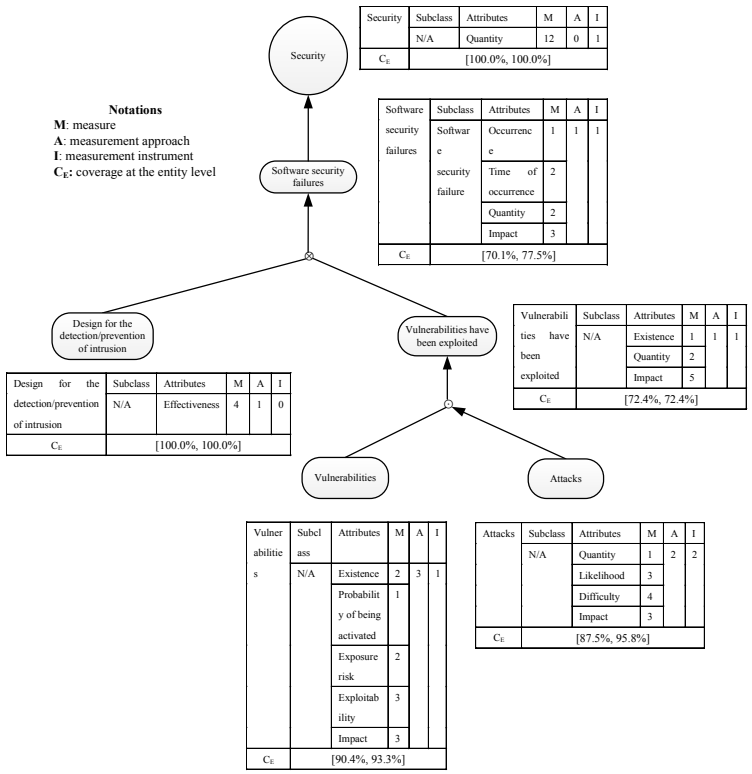
$$C_A(\textit{Software Dependability Attribute}) = \frac{\sum_{e=1}^{E}\sum_{a=1}^{A} C_A(a,e)}{\sum_{e=1}^{E}\sum_{a=1}^{A} 1}$$

    - where *E* is the total number of ***Entities***, and A is the total number of the ***Properties*** for the ***Entity*** *e.*
  - ***Entity level***
    - Use capture-recapture models to estimate the extent to which an ***Entity's Properties*** are covered
  - ***Relation level***

$$C_R(\textit{Dependability Attribute}) = \frac{\text{Number of Total Edges} - \text{Uncovered Edges}}{\text{Number of Total Edges}}$$

Accomplishments (cont'd)

- These three levels provide a structural perspective on the coverage. They together provide insights into whether measurable propagation paths exist from the concepts at the bottom of the causal mechanism graph to those at the top of the causal mechanism graph



Security	Subclass	Attributes	M	A	I
	N/A	Quantity	12	0	1
C_E	[100.0%, 100.0%]				

Software security failures	Subclass	Attributes	M	A	I
	Software security failure	Occurrence	1	1	1
		Time of occurrence	2		
		Quantity	2		
		Impact	3		
C_E	[70.1%, 77.5%]				

Design for the detection/prevention of intrusion	Subclass	Attributes	M	A	I
	N/A	Effectiveness	4	1	0
C_E	[100.0%, 100.0%]				

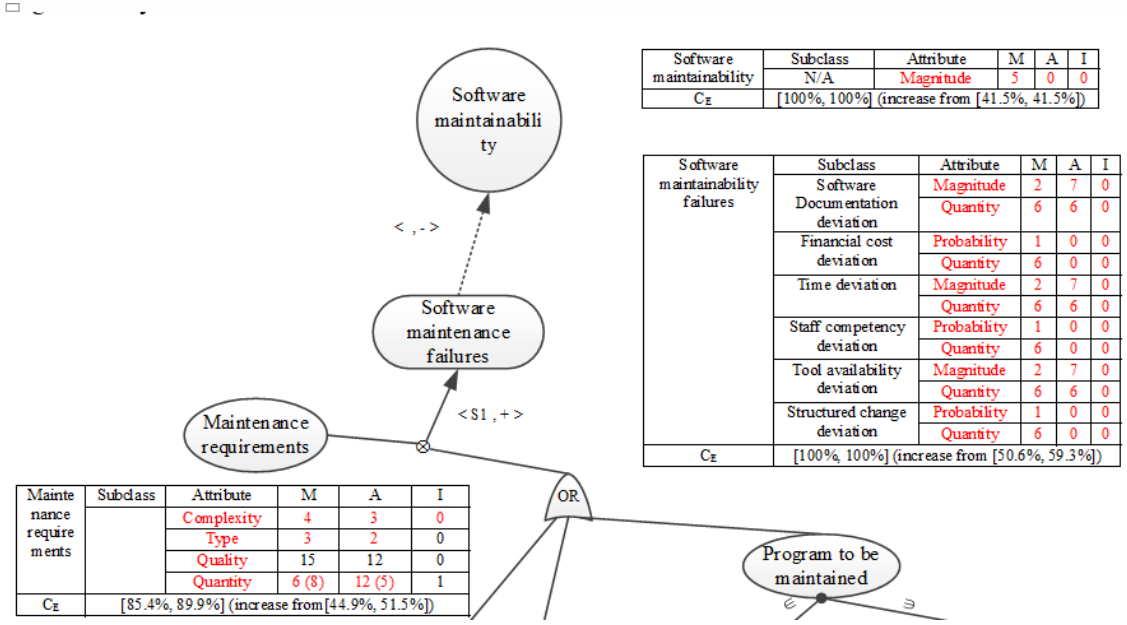
Vulnerabilities have been exploited	Subclass	Attributes	M	A	I
	N/A	Existence	1	1	1
		Quantity	2		
		Impact	5		
C_E	[72.4%, 72.4%]				

Vulnerabilities	Subclass	Attributes	M	A	I
	N/A	Existence	2	3	1
		Probability of being activated	1		
		Exposure risk	2		
		Exploitability	3		
		Impact	3		
C_E	[90.4%, 93.3%]				

Attacks	Subclass	Attributes	M	A	I
	N/A	Quantity	1	2	2
		Likelihood	3		
		Difficulty	4		
		Impact	3		
C_E	[87.5%, 95.8%]				

Accomplishments (cont'd)

- Developing Missing Measures
  - For attributes that are not covered completely, we develop new questionnaires to collect missing measures.
  - By collecting data for missing measures, the coverage of each attribute is recalculated. As a result, most of the coverage increase.



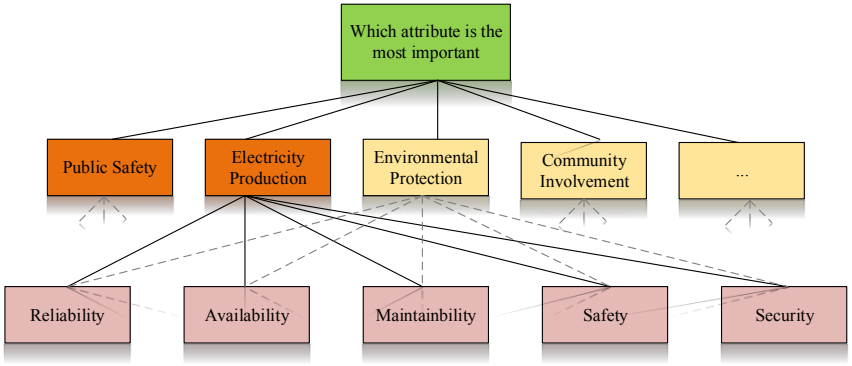
Software maintainability	Subclass	Attribute	M	A	I
	N/A	Magnitude	5	0	0
C_E	[100%, 100%] (increase from [41.5%, 41.5%])				

Software maintainability failures	Subclass	Attribute	M	A	I
	Software Documentation deviation	Magnitude	2	7	0
		Quantity	6	6	0
	Financial cost deviation	Probability	1	0	0
		Quantity	6	0	0
	Time deviation	Magnitude	2	7	0
		Quantity	6	6	0
	Staff competency deviation	Probability	1	0	0
		Quantity	6	0	0
	Tool availability deviation	Magnitude	2	7	0
		Quantity	6	6	0
	Structured change deviation	Probability	1	0	0
		Quantity	6	0	0
C_E	[100%, 100%] (increase from [50.6%, 59.3%])				

Maintenance requirements	Subclass	Attribute	M	A	I
		Complexity	4	3	0
		Type	3	2	0
		Quality	15	12	0
		Quantity	6 (8)	12 (5)	1
C_E	[85.4%, 89.9%] (increase from [44.9%, 51.5%])				

Accomplishments (cont'd)

- **Evaluating the relative importance of dependability attributes**
  - A questionnaire was designed to evaluate the relative importance of the various dependability attributes in the context of a nuclear reactor protection system.
  - The Analytical Hierarchy Process (AHP) method was used to analyze the data obtained



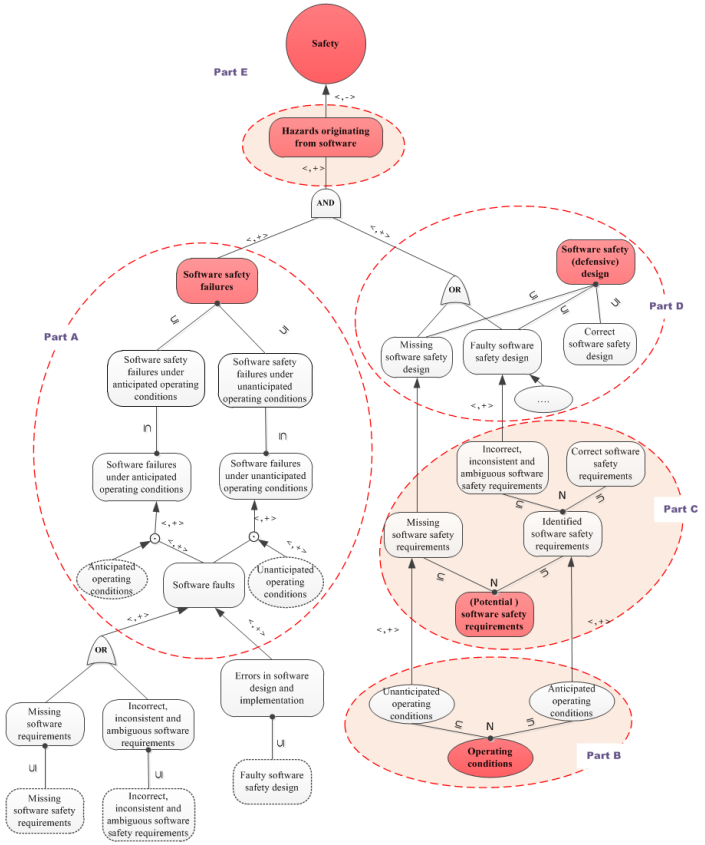
AHP structure for the different goals

Attribute	Relative importance
Safety	0.26667372
Reliability	0.20710062
Security	0.20451856
Availability	0.19482231
Maintainability	0.12688479

Accomplishments (cont'd)

Experimental Evaluation and Attributes Quantification

- The most important dependability attribute for a Reactor Protection System was determined by the nuclear stakeholders to be "Safety" .
- The case study therefore focuses on the evaluation of software safety.
- We focus on a limited scope, i.e. the first phase of development - the requirements phase
- The causal map was tailored to the requirements phase and is being translated into a Bayesian Belief Network for quantification.



Accomplishments (cont'd)

Number of experts who contributed to this project

Covered dependability and/or attributes	Expert Panel #1	Expert Panel #2	Expert Panel #3	Expert Panel #4		
	Focus on dependencies and causal mechanism	**Focus on** causal mechanism verification and measurement	**Focus on** missing measures	**Focus on** importance ranking		
				Academia	Government	Industry
Software dependability	8	-	-	2	4	6
Software reliability	7	-	-			
Software safety	5	5	4			
Software security	5	7	4			
Software availability	5	6	1			
Software maintainability	4	6	3			
Total number of experts for each panel	**11**	**24**	**12**	**12**		
Total number of experts for the project	**59**					

Technology Impact

- ***Impact on software dependability research***

  Designed a new powerful notation system, called causal mechanism graph (CMG), to elicit and represent experts' cause-effect knowledge in the software dependability domain.

- These notations enable practitioners to model causal mechanisms more accurately , and effectively capture the recurrent patterns of comprehensive causal mechanisms existing in the software dependability domain, i.e., *activate* and *conflict*.
- CMG allows researchers to model causal mechanisms in a "robust" manner: when an expert's knowledge on a causal mechanism is very accurate, notations are available to model the mechanisms accurately; when an expert's knowledge is vague (e.g., only causal factors and their influence types are identified), the corresponding causal mechanism graph can be reduced to a conventional causal map and/or a Bayesian Network.

Technology Impact (cont'd)

- ***Impact on software dependability research (cont'd)***

  Designed a systematic measurement framework for software dependability.

- This framework consists of two components: the Causal Mechanism Graph (CMG) and the Ontology of Measurement (OM). The CMG provides systematic solutions to "what concepts should be measured", "why these concepts should be measured" and "when these concepts can be measured", while the OM provides answers to "how these concepts should be measured".
- The framework is an "integrated" framework that can be applied to different attributes as it is from a cause-effect perspective. The quantification can be both prediction and/or estimation, since the framework allows practitioners to incorporate evidence at various phases of the software lifecycle, e.g. failures occurring at the time of software system operation, and process maturity at the time of development.

U.S. DEPARTMENT OF ENERGY
Nuclear Energy

Technology Impact (cont'd)

- ***Impacts on the Nuclear Industry***
- Identified the set of important variables that practitioners should control to reduce software dependability risks.
- Determined the importance ranking of software dependability attributes according to the concerns of the stakeholders. This importance ranking will provide guidance for management and certification of software dependability in the nuclear industry.
- Obtained a large set of measures for quantifying software reliability, safety, security, availability and maintainability. These measures were elicited from a total of 59 domain experts.
- These measures can be used to guide development, which will enhance dependability of the final software product, and to help build a safety/dependability case.

Conclusion

- ***The project identified and modeled the causal mechanisms that influence software dependability, and provided an integrated framework to assess software dependability.***
- ***The models and methods obtained in the project can be further used to improve software dependability design, guide software dependability risk management, and ultimately reduce dependability risks of* Software-Based Safety Critical Instrumentation and Control Systems in Nuclear Power Plants.**