

# Architecture-Level Dependability Analysis of a Medical Decision Support System<sup>1</sup>

Laura L. Pullum  
Oak Ridge National  
Laboratory

P.O. Box 2008, MS-6085  
Oak Ridge, TN 37831  
1-865-574-4602

[PullumLL@ornl.gov](mailto:PullumLL@ornl.gov)

Christopher T. Symons  
Oak Ridge National  
Laboratory

P.O. Box 2008, MS-6085  
Oak Ridge, TN 37831  
1-865-241-5952

[SymonsCT@ornl.gov](mailto:SymonsCT@ornl.gov)

Robert M. Patton  
Oak Ridge National  
Laboratory

P.O. Box 2008, MS-6085  
Oak Ridge, TN 37831  
1-865-576-3832

[PattonRM@ornl.gov](mailto:PattonRM@ornl.gov)

Barbara G. Beckerman  
Oak Ridge National  
Laboratory

P.O. Box 2008, MS-6085  
Oak Ridge, TN 37831  
1-865-576-2681

[beckermanbg@ornl.gov](mailto:beckermanbg@ornl.gov)

## ABSTRACT

Recent advances in techniques such as image analysis, text analysis and machine learning have shown great potential to assist physicians in detecting and diagnosing health issues in patients. In this paper, we describe the approach and findings of an architecture-level dependability analysis for a mammography decision support system that incorporates these techniques. The goal of the research described in this paper is to provide an initial understanding of the dependability issues, particularly the potential failure modes and severity, in order to identify areas of potential high risk. The results will guide design decisions and provide the basis of a dependability and performance evaluation program.

## Categories and Subject Descriptors

D.2.4 [Software Engineering]: Software/Program Verification – *reliability, validation*.

H.3.4 [Information Storage and Retrieval]: Systems and Software – *performance evaluation (efficiency and effectiveness)*.

## General Terms

Algorithms, Design, Reliability, Verification

## Keywords

Architecture-level analysis, failure modes, mammography images, mammography reports, medical decision support.

## 1. INTRODUCTION

Various computer-assisted technologies have been developed to assist radiologists in detecting cancer. Many of these technologies contain embedded software whose validation and verification

(V&V) process is directly tied to that of the hardware. Recent advances in machine learning software have great potential to both extend and refine the usage of the data produced by these technologies with higher validity of truth. Such software, while not embedded with a medical device, requires high confidence in its results to be accepted by potential users (e.g., per the findings in [4]) and to meet regulatory requirements, e.g., [5, 20]. Unfortunately, V&V of machine learning software that assists doctors and radiologists has been largely neglected. In order for machine learning software to be successfully used in clinical practice, this neglect must be addressed. Hence, this research was initiated towards ensuring, and providing evidence of, the dependability of machine learning software for medical decision support.

Currently, research [1-3] is being conducted to develop a multi-modal learning framework and tools for the analysis of radiology images and reports (specifically, mammography, but the framework and tools could be used in other areas of radiology). The semi-supervised machine learning framework integrates text and image modalities by transforming both text and images into feature vectors, which are produced through text and image analysis and processing. These vectors are used to find a lower dimensional space for image analysis that is smooth with respect to the cancer-specific image similarities described in the radiological reports. A classifier developed via the framework, given a set of mammography images as input, would provide an automated ability to confirm a diagnosis, e.g., abnormal or normal, and a confidence measure for that diagnosis.

This paper describes preliminary research being conducted to address the dependability requirements of the previously described decision support system. The goal of this work is to provide an initial understanding of the dependability issues, particularly the potential failure modes and effects at the architectural level, in order to identify areas of potential high risk. Section 2 describes the system architecture. Section 3 describes the architecture-level analysis conducted. Section 4 provides a summary and Section 5 discusses future work.

© 2010 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

SEHC '10, May 3-4, 2010, Cape Town, South Africa □  
ACM 978-1-60558-973-2/10/05 ... \$10.00

<sup>1</sup> Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

## 2. CONCEPTUAL ARCHITECTURE

The system architecture consists of three layers – the Graphical User Interface (GUI) layer, the Classifier layer, and the Data layer, organized according to the Model-View-Controller (MVC) design paradigm [21, 22]. This pattern facilitates separation of concerns, that is, the Input or Persistent Data Set layer (*Controller*), the Processing or Classifier layer (*Model*), and the Output or GUI layer (*View*). Figure 1 illustrates the conceptual architecture, highlighting the Classifier layer.

The Data layer consists of the data itself and the file system software handling the volumes of text reports and images used to train, and eventually, as input to the operational system. It is of vital importance that the data used to develop the classifier meet its requirements, e.g., that any links identifying a connection between an image and a text report be correct. There are many instances where there will be no text report relating to an image and that is not, in itself, erroneous. However, if a link is stated, it must be correct. Although data validity is of great importance to the validity of the overall system and is part of our overall research plan, it is not discussed further in this paper.

The GUI layer consists of an interface to retrieve images, and to present the classification and confidence. Research into dependability of the operational use of the system will include analysis of the GUI layer and is also part of our future research.

The Classifier layer has two modes – training and operation. The Develop Classifier, Discover Graph Kernel, Produce Text Features, and Preprocess Text Reports modules (within the bold dashed-outlined box in Figure 1) are used only during training and are included in the analysis. During training, the classifier is created via the semi-supervised machine-learning framework that integrates text and image modalities as described in the introduction. During operation, the system’s classifier, given a set of mammography images as input, provides a diagnosis and a corresponding confidence measure.

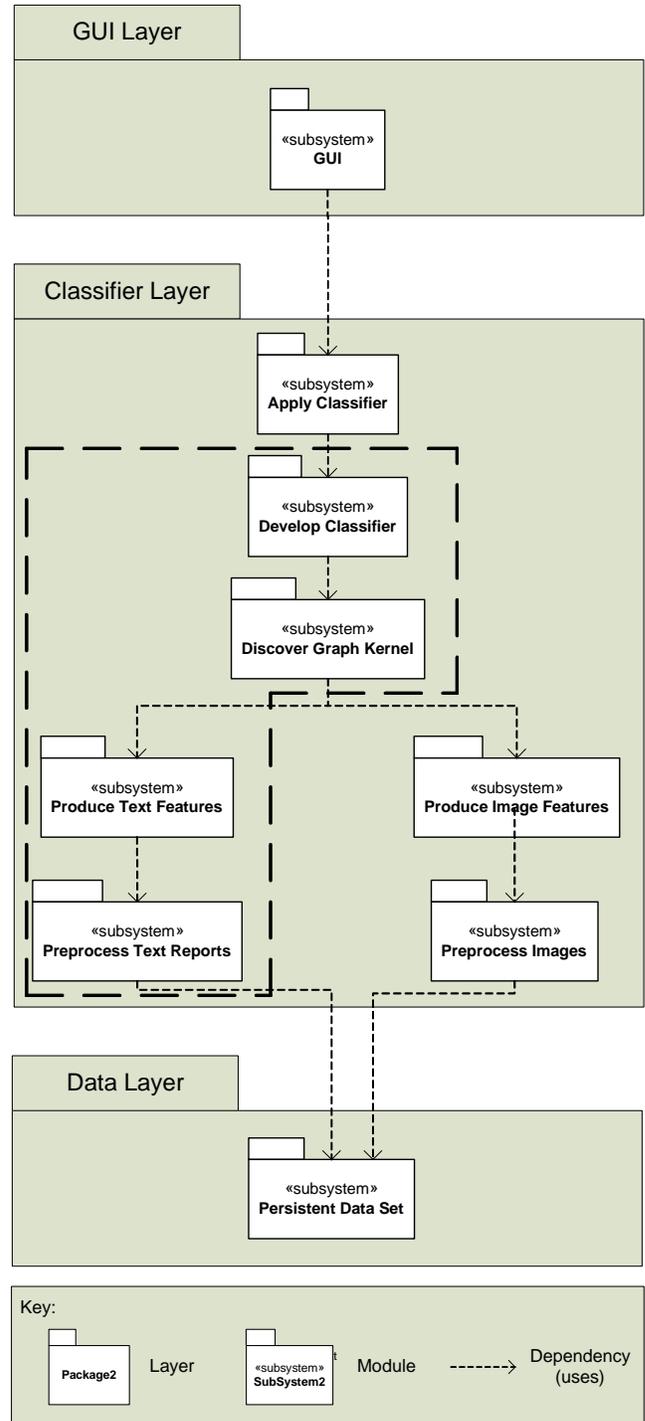
The Classifier modules are defined in Table 1 in terms of their function, input(s), and output(s). The system takes images and text reports and preprocesses them (in the Preprocess Images and Preprocess Text Reports modules, respectively), extracting or isolating the regions and text of interest. Images are preprocessed to isolate only the breast and the pectoral muscle. Text reports are preprocessed to remove headers, html or xml tags, deidentification marks, etc. The preprocessed images and text are then used to develop the image and text feature spaces and vectors in Produce Image Features and Produce Text Features modules, respectively.

The Discover Graph Kernel module uses the image and text features to find a graph kernel for image classification. Using the graph kernel, the image classifier is developed (Develop Classifier module). Finally, the classifier is applied to a new image and the image is classified, e.g., as normal or abnormal, and a confidence indicator is provided for the classification.

## 3. ARCHITECTURE-LEVEL ANALYSIS

For each module, numerous approaches and (sets of) algorithms can be used to accomplish the desired functionality. The system consists of three layers, the GUI layer, the Classifier layer, and the persistent Data layer. The analysis described here is applied to

the Classifier layer. This layer consists of modules responsible for preprocessing the (text and image) data, analyzing the text reports and the images, and developing and applying the classifier as described in Section 2.



**Figure 1. Conceptual system architecture**

	Low	2-3
--	-----	-----

**Table 1. Module definitions for Classifier layer**

Module	Function	Input(s)	Output(s)
Preprocess Image (PI)	Isolate image to contain only breast and pectoral muscle	Mammography images	Isolated portions of images
Produce Image Features (PIF)	Develop the image feature space	Isolated portions of images	Image feature vectors
Preprocess Text Reports (PTR)	Clean up the text report by removing unwanted text	Radiologist text reports	Isolated text
Produce Text Features (PTF)	Develop the text feature space	Isolated text	Text feature vectors
Discover Graph Kernel (DGK)	Use the images and text features to find a graph kernel for image classification	Image feature vectors; Text feature vectors; Links	Graph kernel
Develop Classifier (DC)	Develop classifier to classify image, e.g., as normal or abnormal	Graph kernel	Classifier
Apply Classifier (AC)	Classify an image; Provide confidence indicator	Classifier	Classification; Confidence level

Each step in the functional process of the system is broad in the types of alternatives and each design decision impacts the performance and dependability of the system. The architecture-level analysis presented here provides initial insight into the system dependability.

To begin, we define the failure domain model for the system. Table 2 describes the failure model in terms of classification, confidence in that classification, and severity.

**Table 2. System failure model**

Classification	Confidence	Severity
False Negative	High	5-6
	Low	2-3
False Positive	High	4-5

A *False Positive* occurs when the system classifies the image as abnormal when it is, in fact, normal. A *False Negative* occurs when the system classifies the image as normal when it is actually abnormal. A *Low* means the calculated confidence in the classification is low, providing a sense of uncertainty that partially mitigates the inaccuracy of the classification. A *High* indicates that the calculated confidence in the classification is high, despite its inaccuracy.

The severity level is an indication of the potential severity of a given system level effect. The severity scale used here is based on a typical 6-level scale (defined in Table 3), with 6 being most severe. For this system, a false negative or false positive result with high confidence renders the support system unfit for use, particularly given that a false negative with high confidence can have life-threatening consequences. A false negative or false positive result with low confidence would likely result in retesting (the patient) or reexamination and could be frustrating, causing the user to complain or not use the “support” tool.

**Table 3. Example failure severity scale**

Level	Description
6	Terminal injury or death
5	Major injury
4	Minor injury
3	Major system problem
2	Minor system problem
1	Slight annoyance

For the purpose of this architecture-level analysis, we generalize the system failure mode to a single “Incorrect Classification or Incorrect Confidence”. This is necessary because, at this level of analysis, one cannot determine, to any significant degree, the specific impact of a failure mode on the Classification or on the Confidence measure. The failure model defined in Table 2 will serve well when module- and lower-level analyses are conducted.

Given the system requirements, failure model, and high-level description, we initiated an architecture-level dependability analysis (as in [6], for example). The steps taken and sections in which they are discussed are:

- Failure Modes and Effects Analysis (FMEA)
  - Taxonomy (section 3.1)
  - Functional FMEA (section 3.2)
  - Detailed software FMEA (section 3.3)
- Fault Tree Analysis (section 3.4).

### 3.1 Failure Mode Taxonomy

An initial step in the analysis is to examine the system’s failure modes and effects. A failure mode taxonomy defines the breadth and depth of failure modes to be considered in the analysis. Using a combination of failure mode taxonomies (e.g., [9, 10]) as a basis, we tailored a taxonomy for use in this analysis. The tailoring includes failure mode space reduction by considering only those failure modes that are possible given requirements-

based constraints and the architecture level of the analysis. The resulting taxonomy of failure modes is provided in Figure 2.

The taxonomy defines the failure modes as either function- or input/output (I/O)-related. The only functional failure mode considered in this analysis is the incorrect realization of the module’s functionality. This can result from module implementation errors.

I/O failure modes refer to the inputs and outputs of a module. I/O.Amount refers to the number or quantity of input or output. For example, if the requirements state that the Preprocess Images module will input a set of 4 mammography images (i.e., top and side views of left and right breasts) and only one image is received, then this failure mode is referred to as I/O.Amount.Too\_Little. An I/O.Value.Incorrect failure mode covers those cases when the input to or output from a module is incorrect. An I/O.Range.Out\_of\_Range failure mode occurs when the I/O value is outside its requirements-specified bounds/limits. An I/O.Type.Mismatch failure mode includes cases when the expected I/O type and the actual I/O type do not match. The taxonomy is used in defining the failure modes as illustrated in sections 3.2 and 3.3.

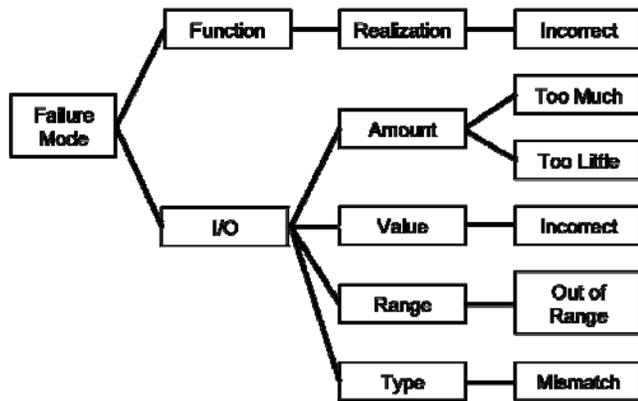


Figure 2. Tailored failure mode taxonomy

### 3.2 Functional FMEA

The process of conducting a software FMEA (failure mode and effects analysis) helps identify structural weaknesses in the design and identify missing or incorrect requirements. The primary purpose of FMEA is to identify possible failure modes of the system components and evaluate their impact on the system performance.

Software FMEA is conducted here on two levels – the system-level or functional FMEA (section 3.2) and the more detailed level (section 3.3). The functional FMEA examines each module and for each functional failure mode, determines the local effect and the effect at the system level. The results of functional FMEA on the decision support system is provided in Table 4.

Additional analysis at this level can include the detectability and reversibility of each failure mode’s effect.

### 3.3 Detailed Software FMEA

The detailed software FMEA examines each module for each I/O or data failure mode and describes the local effect and the effect at the system level. Table 5 presents partial results for the detailed software FMEA, for the Preprocess Images module. Some failure modes may not be applicable to all modules. For example, in the system under study, an out of range input does not apply to the Preprocesses Images module. Other failure modes cause the system to crash, e.g., an I/O.Amount.Too\_Little failure mode. This failure mode is easy to detect and is an obvious candidate for mitigation.

Most of the failure modes lead to an “Incorrect Classification or Incorrect Confidence” effect at the system level. To be most useful, the modules’ alternative algorithms and their performance should be analyzed at the next level of detail. This can assist algorithm selection and the detailed design process.

Table 4. Functional FMEA for Classifier layer modules

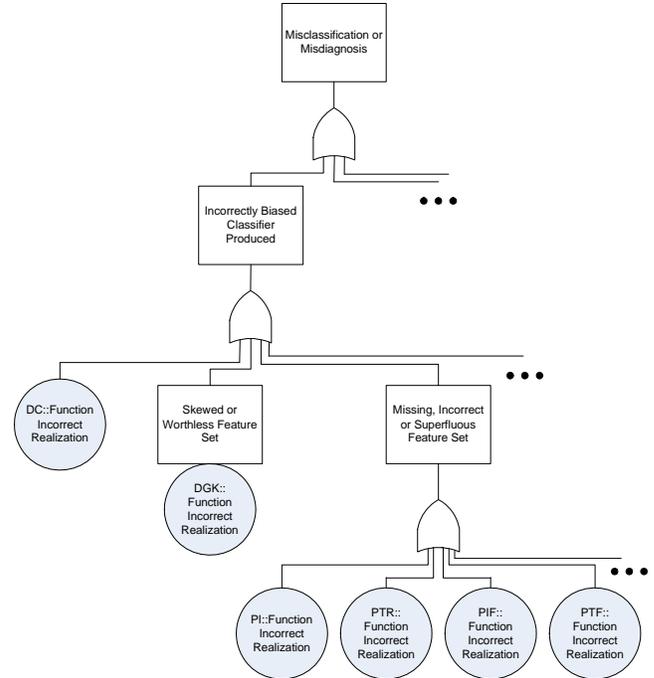
Module	Failure Mode	Local Effect	System Effect
Preprocess Images (PI)	Function. Incorrect_ Realization	Images are preprocessed incorrectly; Image features missing, incorrect, or superfluous	Incorrect Classification or Incorrect Confidence
Preprocess Text Reports (PTR)	Function. Incorrect_ Realization	Text reports preprocessed incorrectly; Text features missing, incorrect, or superfluous	Incorrect Classification or Incorrect Confidence
Produce Image Features (PIF)	Function. Incorrect_ Realization	Incorrect image features extracted; image features missing, incorrect or superfluous	Incorrect Classification or Incorrect Confidence
Produce Text Features (PTF)	Function. Incorrect_ Realization	Text features missing, incorrect or superfluous	Incorrect Classification or Incorrect Confidence
Discover Graph Kernel (DGK)	Function. Incorrect_ Realization	Worthless or skewed feature space	Incorrect Classification or Incorrect Confidence
		Module failure - crash	System failure - crash
Develop Classifier (DC)	Function. Incorrect_ Realization	Error in optimization routine that sets the weights; Effect - bad classifier	Incorrect Classification or Incorrect Confidence
Apply Classifier	Function. Incorrect	Incorrect decision;	Incorrect Classification

(AC)	Realization	Unwarranted or incorrect confidence in result	or Incorrect Confidence
------	-------------	---	-------------------------

could be to use Monte Carlo analysis [14, 15] to propagate failure uncertainty through the fault tree to determine the expected value and variance of the top event’s probability. However, analytical approaches using a) natural language qualifiers of probability ranges [16, 17] and uncertainty propagation through the fault tree [18] or b) fuzzy logic and approximate reasoning [19] are more appropriate at this point.

**Table 5. Partial software FMEA data table for Classifier layer, Preprocess Images module**

Module	Failure Mode	Local Effect	System Effect
Preprocess Images (PI)	I/O.Amount. Too_Much	Missing information	Incorrect Classification or Incorrect Confidence
	I/O.Amount. Too_Little	Missing information	Incorrect Classification or Incorrect Confidence
	I/O.Amount. Too_Little	Module failure	System Failure (crash)
	I/O.Value. Incorrect_Value	Missing input/information	Incorrect Classification or Incorrect Confidence
	I/O.Range. Out_of_Range	n.a.	n.a.
	I/O.Type.Data. Type_Mismatch	Missing information	Incorrect Classification or Incorrect Confidence



**Figure 3. Partial fault tree for the Classifier layer**

### 3.4 Fault Tree

Fault tree models [9, 10] have long been used for qualitative and quantitative analysis of the failure modes of critical systems. A fault tree provides a mathematical and graphical representation of the combinations of events that can lead to system failure. The construction of a fault tree model can provide insight into the system by illuminating potential weaknesses with respect to dependability. A fault tree can help with the diagnosis of failure symptoms by illustrating which combinations of events could lead to failure symptoms that are observed in the field. The quantitative analysis of a fault tree is used to determine the probability of system failure, given the probability of occurrence for events.

Figure 3 presents a partial fault tree for the Classifier layer. The top-level event is *Misclassification or Misdiagnosis*. The fault tree represents the combinations of events that can lead to a misdiagnosis in the training phase, in the Classifier layer. In Figure 3, we see that even in this partial fault tree, there are several events and combinations of events that can lead to a misdiagnosis. At this time and level of analysis, accurate values for probabilities of event failures are not available. Given this uncertainty in the fault tree inputs, a next step in the analysis

## 4. SUMMARY

We have presented the approach and results of preliminary research to address dependability requirements of an application of multi-modal learning to medical decision support. The initial analysis is conducted at the software architecture level and includes domain analysis to determine the system failure model, functional and detailed software FMEA, and fault tree development. The analysis has identified some obvious areas for failure mode detection and mitigation. Going through the analysis has also provided valuable insight into the system and its potential areas of risk, as well as raising awareness of the need for and value of software dependability.

## 5. FUTURE WORK

This preliminary research has provided the basis for future research in machine learning dependability in general and analysis of the medical decision support system specifically. Near

term analysis includes conducting uncertainty propagation and analysis with the architecture-level fault tree.

Each module in the architecture has a broad range of possible approaches and algorithm alternatives, and each design decision impacts the performance and dependability of the system. Future research will examine the design alternatives at the module level and study their impact on system dependability. Additional research will include performance analysis of the various alternatives, along with the dependability analysis to enable design choices that allow a global optimum of dependability and performance to be reached. An overarching goal of providing guidance for V&V and dependability analysis (as in [11-13] for neural networks) for machine learning is envisioned.

## 6. ACKNOWLEDGMENTS

Our thanks to Robert M. Nishikawa, Ph.D., Department of Radiology, University of Chicago for providing the large dataset of mammography data (both reports and images), from which test subsets were chosen.

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05-00OR22725.

## 7. REFERENCES

- [1] Symons, C.T., Kerekes, R., Paquit, V.C., Patton, R., Gleason, S.S., and Beckerman, B. 2009. A multimodal, semi-supervised learning system for building better decision support systems for the analysis of mammograms. In Radiological Society of North America 95th Scientific Assembly and Annual Meeting Program (Oak Brook, Ill. USA). RSNA 2009. SSA11-08.
- [2] Patton, R. M., Beckerman, B. G., and Potok, T. E. 2008. Analysis of mammography reports using maximum variation sampling. In Proceedings of the 4th GECCO Workshop on Medical Applications of Genetic and Evolutionary Computation (Atlanta, USA). MedGEC'08. ACM Press, New York, NY.
- [3] Patton, R. M., Potok, T. E., Beckerman, B. G., and Treadwell, J. N. 2009. A genetic algorithm for learning significant phrase patterns in radiology reports. In Proceedings of the 5th GECCO Workshop on Medical Applications of Genetic and Evolutionary Computation (Montreal, Canada, July 2009). MedGEC'09. ACM Press, New York, NY.
- [4] Varonen, H., Kortteisto, T., and Kaila, M. 2008. What may help or hinder the implementation of computerized decision support systems (CDSs): a focus group study with physicians. *Family Practice* 2008; **25**: 162-167.
- [5] U.S. Food and Drug Administration. October 21, 2009. Draft Guidance for Industry and FDA Staff: Computer-Assisted Detection Devices Applied to Radiology Images and Radiology Device Data – Premarket Notification [510(k)] Submissions. Center for Devices and Radiological Health.
- [6] Tekinerdogan, B., Sozer, H., and Aksit, M. 2008. Software architecture reliability analysis using failure scenarios. *Journal of Systems and Software* 81 (2008) 558-575.
- [7] Li, B., Li, M., Ghose, S., and Smidts, C. 2003. Integrating Software into PRA. In Proceedings of the 14<sup>th</sup> International Symposium on Software Reliability Engineering. (ISSRE'03).
- [8] Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*. 1 (Jan.-Mar. 2004) 11-33.
- [9] Ye, F., and Kelly, T. 2004. Contract-based justification for COTS component within safety-critical applications. In 9<sup>th</sup> Australian Workshop on Safety Related Programmable Systems (Brisbane, Australia, 2004). SCS'04. Australian Computer Society, Inc. 13-22.
- [10] Pullum, L. L., and Dugan, J. B. 1996. Fault tree models for the analysis of complex computer-based systems. In Proceedings of the Annual Reliability and Maintainability Symposium. RAMS'96. 200-207.
- [11] Taylor, B. J., Darrah, M. A., Pullum, L. L., et al. 2005. Methods and Procedures for the Verification and Validation of Neural Networks. Brian Taylor, ed., Springer-Verlag, 2005.
- [12] Pullum, L. L., Taylor, B. J., and Darrah, M. A. 2007. Guidance for the Verification and Validation of Neural Networks. IEEE Computer Society Press (Wiley), 2007.
- [13] Pullum, L. L., Darrah, M. A., and Taylor, B. J. 2004. Independent verification and validation of neural networks – developing practitioner assistance. *Software Tech News*. July, 2004.
- [14] Hickman, J.W., et al. 1983. PRA procedures guide: A guide to the performance of probabilistic risk assessments for nuclear power plants. USA: Nuclear Regulatory Commission. NUREG/CR-2300, 1983.
- [15] Ripley, B.D. 1987. Stochastic simulation. Wiley, 1987.
- [16] Beyth-Marom, R. 1982. How probable is probable? A numerical translation of verbal probability expressions. *Journal Forecast* 1982. 1:257-269.
- [17] Wallsten, T.S., Budescu, D.V., Rappaport, A., Zwick, R., and Forsyth, B. 1986. Measuring the vague meanings of probability terms. *Journal Experimental Psychology: General* 1986. 115(4):348-365.
- [18] Hauptmanns, U. 2002. Analytical propagation of uncertainties through fault trees. *Reliability Engineering and System Safety* 76 (2002) 327-329.
- [19] Pillay, A., and Wang, J. 2003. Modified failure mode and effects analysis using approximate reasoning. *Reliability Engineering and System Safety* 79 (2003) 69-85.
- [20] U.S. Food and Drug Administration. May 11, 2005. Guidance for Industry and FDA Staff: Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices. Center for Devices and Radiological Health and Center for Biologics Evaluation and Research.
- [21] Reenskaug, T. 1979. THING-MODEL-VIEW-EDITOR - an example from a planning system. Technical note, Xerox PARC, May 1979.

[22] Reenskaug, T. 1979. MODELS - VIEWS -  
CONTROLLERS. Technical note, Xerox PARC, December

1979.