U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

NUREG/CR-7233

Developing a Bayesian Belief Network Model for Quantifying the Probability of Software Failure of a Protection System

Office of Nuclear Regulatory Research

AVAILABILITY OF REFERENCE MATERIALS IN NRC PUBLICATIONS

NRC Reference Material

As of November 1999, you may electronically access NUREG-series publications and other NRC records at NRC's Library at www.nrc.gov/reading-rm.html. Publicly released records include, to name a few, NUREG-series publications; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigative reports; licensee event reports; and Commission papers and their attachments.

NRC publications in the NUREG series, NRC regulations, and Title 10, "Energy," in the *Code of Federal Regulations* may also be purchased from one of these two sources.

1. **The Superintendent of Documents**
   U.S. Government Publishing Office
   Washington, DC 20402-0001
   Internet: bookstore.gpo.gov
   Telephone: (202) 512-1800
   Fax: (202) 512-2104

2. **The National Technical Information Service**
   5301 Shawnee Rd., Alexandria, VA 22312-0002
   www.ntis.gov
   1-800-553-6847 or, locally, (703) 605-6000

A single copy of each NRC draft report for comment is available free, to the extent of supply, upon written request as follows:

Address: **U.S. Nuclear Regulatory Commission**
Office of Administration
Multimedia, Graphics, Storage, and Distribution Branch
Washington, DC 20555-0001
E-mail: distribution.resource@nrc.gov
Facsimile: (301) 415-2289

Some publications in the NUREG series that are posted at NRC's Web site address www.nrc.gov/reading-rm/doc-collections/nuregs are updated periodically and may differ from the last printed version. Although references to material found on a Web site bear the date the material was accessed, the material available on the date cited may subsequently be removed from the site.

Non-NRC Reference Material

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, transactions, *Federal Register* notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at—

The NRC Technical Library
Two White Flint North
11545 Rockville Pike
Rockville, MD 20852-2738

These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036-8002
www.ansi.org
(212) 642-4900

Legally binding regulatory requirements are stated only in laws; NRC regulations; licenses, including technical specifications; or orders, not in NUREG-series publications. The views expressed in contractorprepared publications in this series are not necessarily those of the NRC.

The NUREG series comprises (1) technical and administrative reports and books prepared by the staff (NUREG–XXXX) or agency contractors (NUREG/CR–XXXX), (2) proceedings of conferences (NUREG/CP–XXXX), (3) reports resulting from international agreements (NUREG/IA–XXXX), (4) brochures (NUREG/BR–XXXX), and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Directors' decisions under Section 2.206 of NRC's regulations (NUREG–0750).

U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

NUREG/CR-7233

Developing a Bayesian Belief Network Model for Quantifying the Probability of Software Failure of a Protection System

Manuscript Completed: July 2016
Date Published: January 2018

Prepared by:
Tsong-Lun Chu[1], Athi Varuttamaseni[1], Meng Yue[1],
Seung Jun Lee[2], Hyun Gook Kang[3], Jaehyun Cho[4], and Steve Yang[5]

[1]Brookhaven National Laboratory
[2]Ulsan National Institute of Science and Technology
[3]Korea Advanced Institute of Science and Technology
[4]Korea Atomic Energy Research Institute
[5]NUV Technology, LLC

Ming Li, NRC Project Manager

Office of Nuclear Regulatory Research

ABSTRACT

A new approach has been developed to quantify the software failure probabilities in nuclear power plant (NPP) digital instrumentation and control (I&C) systems. Specifically, this approach uses a Bayesian belief network (BBN) to model the causal relationships between the software development life cycle, the number of residual defects within software, and the software failure probability. The software development life cycle (SDLC) characteristics (e.g., development quality and verification and validation (V&V) quality), and software-self characteristics (e.g., size and complexity) are represented using a hierarchical structure. As part of the BBN model development, the SDLCs were classified into five phases: requirements, design, implementation, testing, and installation/checkout. Information for each phase (or activity) was abstracted from the relevant guidance and standards documents. A BBN sub-model was then developed for each phase to estimate the number of software defects remaining. The phase sub-models include the quality of software development and verification and validation (V&V) activities, which affect the number of defects inserted and the number of defects detected/removed in that specific phase. Three rounds of expert elicitation were used to complete the BBN model. The first two rounds used experts with knowledge and experience in the general application of software quality assurance to assist in the identification of BBN nodes, the construction of the BBN model structure (the causal relationship), and the establishment of the Node Probability Tables (NPTs) (the causal relationship quantification). The NPTs were further Bayesian updated using literature data available from the literature and the limited amount of development and V&V data. The insights gained from these elicitations were used to develop a BBN model for NPP digital safety software. The outputs from the third round of elicitations were used as inputs to the BBN model applications to two trial nuclear systems: (1) the Loop Operating Control System (LOCS) of the Advanced Test Reactor (ATR) at Idaho National Laboratory, and (2) the prototype Integrated Digital Protection System-Reactor Protection System (IDiPS-RPS) developed by the Korea Atomic Energy Research Institute (KAERI). Experts who are familiar with the software development, including V&V activities, of the two trial systems provided these inputs. The results obtained from applications of the modified BBN model to two nuclear applications as well as an assessment of the feasibility of using BBNs for quantifying software failure probabilities are discussed herein.

FOREWORD

With a shift in technology to digital systems due to analog systems approaching obsolescence and to functional advantages of digital systems, existing plants have begun to replace some current analog I&C systems, while new plant designs fully incorporate digital systems. This shift necessitates a research program to develop, and ultimately incorporate, digital I&C models into NPP Probabilistic Risk Assessments (PRAs). This work is needed to meet the objectives of the NRC's 1995 PRA Policy Statement, which encourages the use of PRA technology in all regulatory matters to the extent supported by the state of the art in PRA methods and data.

Previous digital system PRA research sponsored by the U.S. Nuclear Regulatory Commission (NRC) focused on system and hardware failure modeling. Software failures were not comprehensively addressed in those earlier studies. In contrast, the study presented in this report develops and demonstrates a method that quantifies the failure probability per demand for NPP safety software based on software lifecycle quality attributes.

Since software fails due to residual defects in the software, it is well observed that the greater the number of residual defects, the less reliable the software. The number of defects remaining should therefore have a causal relationship with the software failure probability. Since the defects are introduced during software development life cycle, the software development characteristics also have some relationships with the number of defects remaining.

Since early 1990s, Bayesian Belief Networks (BBNs) have been used to describe relationships between the development characteristics, which are normally vague and qualitative, and the (quantitative) number of defects remaining in order to calculate the software failure rate. However, little agreement exists with respect to the software development characteristics, causal relationships, and quantification of these relationships mainly due to the diversity of software and the inaccessibility of proprietary software development data to the analyst.

This study follows recent international efforts (led by South Korea and the United Kingdom) using BBN models to estimate software quality in terms of the number of defects remaining. The purpose of this study is to build on this previous work and demonstrate a practical framework for identifying software development characteristics; establishing and quantifying the causal relationships between these characteristics, estimating the number of defects remaining, and the software failure probability using expert opinion; probabilistically aggregating multiple expert inputs; and utilizing literature data and available development data to Bayesian update expert inputs to reduce uncertainties introduced from expert opinion.

This BBN modeling framework was successfully applied to two NPP applications. Due to development and operation data availability limitations, results do not necessarily represent the actual level of quality and safety of these two systems. Instead, insights on feasibility and lessons learned are summarized in this report. Recommendations for future applications of this framework are summarized including the revisit of some assumptions, customizations of BBN structures to fit specific development processes, and training the model using real development and operating data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The U.S. Nuclear Regulatory Commission (NRC) encourages the use of probabilistic risk assessment (PRA) in all regulatory matters, to the extent supported by the state-of-the-art in PRA methods and data. Though risk-informed regulatory procedures have been successfully developed for many NPP systems, a risk-informed analysis procedure for digital systems is still not fully developed. This issue is especially important in light of the shift of NPPs from analog systems to digital systems. Therefore, the NRC has established a research plan to identify and develop methods, analytical tools, and regulatory guidance for (1) including models of digital systems in nuclear power plant (NPP) PRAs, and, (2) incorporating digital systems in the NRC's risk-informed licensing and oversight activities.

Previous NRC-sponsored research has explored the possibility of addressing failures in digital instrumentation and control (I&C) systems within the framework of current NPP PRAs. Key issues that were identified from these studies and require further development are: reliability modeling for digital I&C hardware and software, dependencies among digital I&C components (including hardware/hardware, hardware/software and software/software), and operator interaction with digital systems. Addressing these issues should facilitate the integration of digital I&C failure models into current NPP PRAs.

It is generally recognized that software fails due to residual defects in the software. These defects include errors in user requirements and other defects introduced during the development and deployment processes. In this study, causal relationships between the software development lifecycle (SDLC) and the number of residual defects in software were identified and modeled using Bayesian belief networks (BBNs) in this study. The software failure probability was assumed to be proportional to the number of residual defects, for the sake of simplicity.

Three rounds of expert opinion elicitation were used to develop the BBN model. The first round focused on identifying SDLC characteristics (i.e., "nodes" for the BBN model) that are relevant to the software defect content (i.e., the number of defects) and the causal relationship between these two parameters. The first round experts are familiar with software development and have backgrounds in system engineering, software development, and quality control. The second round of elicitation quantified the aforementioned causal relationships. The experts for the second round have experience in nuclear digital I&C system development and were able to provide nuclear industry-specific quantitative inputs. Using the inputs from the first and second rounds of elicitations, a BBN model for NPP digital I&C systems was developed. The third round of elicitation provided inputs to the model for application to two trial systems: (1) the Loop Operating Control System (LOCS) of the Advanced Test Reactor (ATR) at Idaho National Laboratory and (2) the prototype Integrated Digital Protection System-Reactor Protection System (IDiPS-RPS) developed by KAERI. The number of residing defects and the failure probabilities of these two software were estimated. Where possible, expert opinion data were Bayesian updated using literature data and the limited software development data, thereby reducing the uncertainty.

A framework for applying BBN to safety-related digital I&C systems at NPPs has been established attempting to overcome the lack of available proprietary data for this study. The purpose of this work is to demonstrate the feasibility of applying BBN methodologies to digital I&C systems; to assess the accuracy, uncertainty, level of effort and limitations of using this approach; and to guide stakeholders for future applications. The selection of the nodes, and thus the construction and quantification of the causal networks, could be improved if better

insights about SDLC of nuclear applications and more data become available. The results of these two applications, however, do not necessarily represent the defect content of these two systems due to the incomplete knowledge about them and the limited data available to the research team. Therefore, these results are intended to assess the feasibility of applying this method to nuclear plant digital systems rather than supporting a specific regulatory assessment. In light of these limitations, the use of these results to support regulatory decision-making is inappropriate.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of several individuals who helped make this report a reality. We thank Drs. Bev Littlewood and Lorenzo Strigini of City University London for their assistance in formulating the Bayesian belief network approach discussed in this report. We also thank George Marts and Nancy Johnson of Idaho National Laboratory for providing documents pertaining to the Loop Operating and Control System for the Advanced Test Reactor as well as the experts who provided inputs for LOCS, and Han Seong Son of Joongbu University for providing inputs for IDiPS. We want to extend our thanks the experts recognized in Appendix C who provided valuable inputs for this study.

The Nuclear Regulatory Commission (NRC) project managers, Mr. Alan Kuritzky and Dr. Ming Li, collectively oversaw the BBN project from inception to conclusion and, together with their Branch Chief, Dr. Kevin Coyne, were instrumental in navigating through any technical difficulties encountered to ensure completion of the project. We are grateful to Dr. Joy Leggett of the NRC for her technical review and editing of this report, and to the other NRC reviewers who reviewed this report. We also express our appreciation to Avril Woodhead of Brookhaven National Laboratory (BNL) for her editorial review of the report, and to Linda Fitz, Jeanne Frejka and Maria Anzaldi, also of BNL, who put several versions of the report together and helped with the logistics of the project.

ACRONYMS AND ABBREVIATIONS

ACRS	Advisory Committee on Reactor Safeguards
AIC	Akaike information criterion
AMSAC	ATWS mitigation system actuation circuitry
ASME	American Society of Mechanical Engineers
ATIP	automatic test and interface processor
ATR	advanced test reactor
ATWS	anticipated transient without scram
BBN	Bayesian belief network
BIC	Bayesian information criterion
BN	Bayesian net
BNL	Brookhaven National Laboratory
BP	bistable processor
CCF	common cause failure
COMPSIS	computer systems important to safety
CPC	core protection calculators
CM	configuration management
CP	coincidence processor
DOE	Department of Energy
EPRI	Electric Power Research Institute
FAT	factory acceptance test
FER	fault exposure ratio
FP	function point
FSD	fault size distribution
IAEA	International Atomic Energy Agency
IDIP	International Database on Digital I&C Products
IDiPS-RPS	Integrated Digital Protection System-Reactor Protection System
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point Users Group
I&C	instrumentation and control
I/O	input/output
IEC	International Electrotechnical Commission
INL	Idaho National Laboratory
KAERI	Korea Atomic Energy Research Institute
KNICS	Korea Nuclear Instrumentation and Control System
LER	licensee event report
LOC	lines of code
LOCS	loop operating control system
MOU	memorandum of understanding
NASA	National Aeronautics and Space Administration
NEA	Nuclear Energy Agency
NPP	nuclear power plant

NPT	node probability table
NRC	Nuclear Regulatory Commission
OECD	Organization for Economic Cooperation and Development
PRA	probabilistic risk assessment
QA	quality assurance
QAP	quality assurance plan
QL	quality level
QLD	quality level determination
QSRM	quantitative software reliability method
RG	regulatory guide
RPS	reactor protection system
σ	standard deviation
SD	software design
SDLC	software development life cycle
SDP	software development plan
SDS	software design specification
SQA	software quality assurance
SRGM	software reliability growth model
SRS	software requirements specifications
SSD	safety software determination
SW	specific software
SyRS	System Requirements Specifications
V&V	verification and validation
WBS	work breakdown structure
WGRisk	Working Group on Risk Assessment

1 INTRODUCTION

1.1 Background

The U.S. Nuclear Regulatory Commission's (NRC's) current licensing process for digital systems relies on deterministic engineering criteria. In its 1995 probabilistic risk assessment (PRA) policy statement [NRC 1995a], the Commission encouraged the use of PRA technology in all regulatory matters to the extent supported by the state-of-the-art in PRA methods and data. Much has been accomplished in a broad spectrum of areas related to risk-informed regulation. However, the process of risk-informed analysis for digital systems is not yet fully developed. Since digital instrumentation and control (I&C) systems are expected to play an increasingly important safety role at nuclear power plants (NPPs), the NRC established a research plan for digital systems [NRC 2010a] defining a coherent set of projects to support regulatory needs. Some projects included in this research plan address methods of risk assessment and data for digital systems. The objective of the NRC's research on the risk of digital systems is to identify and develop methods, analytical tools, and regulatory guidance for (1) including models of digital systems in NPP's PRAs, and, (2) incorporating digital systems in the NRC's risk-informed licensing and oversight process.

Figure 1-1 depicts the inter-relationship among various activities associated with NRC's research on digital-systems PRA. The work of developing approaches to model the reliability of digital systems is being coordinated with several other areas of related research, including the identification and analysis of failure modes [Chu 2008 and 2009a], the analysis of operating experience [Korsah 2010], and the inventory and classification of digital systems [Wood 2012]. This research has benefited from interactions with the Electric Power Research Institute (EPRI), the National Aeronautics and Space Administration (NASA), and the Nuclear Energy Agency (NEA) of the Organisation for Economic Cooperation and Development (OECD). More specifically, there have been interactions with the NEA Working Group on Risk Assessment (WGRisk) and the OECD/NEA activity on Computer Systems Important to Safety (COMPSIS).

An important insight gained from initial research is the need to establish a commonly accepted basis for incorporating the behavior of the software into digital I&C system reliability models that are compatible with existing NPP PRAs[1]. For several years, Brookhaven National Laboratory (BNL) has been investigating methods and tools for the probabilistic modeling of digital systems under NRC's contracts. The results are documented in NUREG/CR-6962 [Chu 2008], and in NUREG/CR-6997 [Chu 2009a]. The NRC also sponsored the Ohio State University to investigate the modeling of digital systems using dynamic PRA methods, as detailed in NUREG/CR-6901 [Aldemir 2006], NUREG/CR-6942 [Aldemir 2007], and NUREG/CR-6985 [Aldemir 2009].

[1] Existing NPP PRAs were originally developed and updated using traditional (static) event tree and fault tree methods. To address software failures in current PRA framework, they must be included in the PRA sequences. That is, the software functions or components need to be modeled and quantified as events in the event trees or in the fault trees

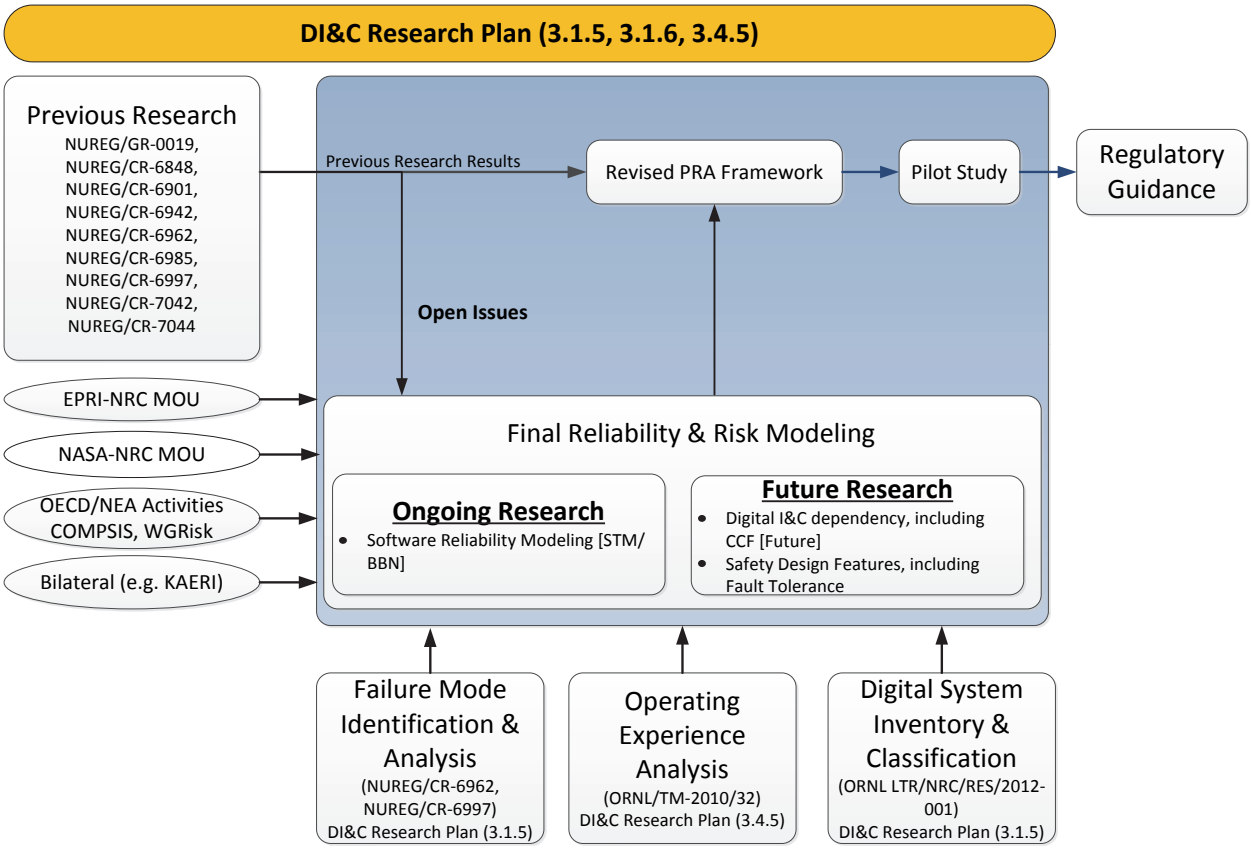

Figure 1-1 NRC Research Activities on the Reliability of Digital Systems

Software failure has not been consistently defined in the literature [IEEE 610.12, Lyu 1996], and there is no consensus on its definition. In this study, software failure is defined as follows: the triggering of a fault of the software, introduced during its developmental life cycle, that results in, or contributes to, the host (digital) system (1) failing to accomplish its intended function, or (2) initiating an undesired action. Triggering includes inputs to the software from the operating environment (i.e., the NPP's operating condition), and from the internal state of the digital system.

BNL has been exploring how software failures can be included in hardware failure reliability models and subsequently into a PRA, so that their contribution to the risk of the associated NPP can be assessed. The NRC Advisory Committee on Reactor Safeguards (ACRS) Subcommittee on Digital I&C Systems recommended NRC to conduct a study to investigate the philosophical basis of software failures. The NRC contracted BNL in 2008 to organize a panel and workshop with the goal of establishing a "philosophical basis" for incorporating software failures into models of digital-system reliability for use in PRAs [Chu 2009b]. The experts invited to the panel were recognized specialists from around the world, with knowledge of software reliability and/or PRA. The meeting resulted in the establishment of the following philosophical basis for incorporating software failures into a PRA [Chu 2009b]:

Software failure is basically a deterministic process. However, because of our incomplete knowledge, we are not able to fully account for and quantify all the variables that define the software failure process. Therefore, we use probabilistic modeling to describe and characterize it.

The Panel also agreed that:

- Software fails,
- The occurrence of software failures can be treated probabilistically,
- It is meaningful to use software failure rates and probabilities, and,
- Software failure rates and probabilities can be included in reliability models of digital systems.

Subsequently, BNL reviewed a spectrum of quantitative software reliability methods (QSRMs) to catalog potential ones that may serve to quantify the rates of software failure and the per-demand failure probabilities of digital systems at NPPs, such that the system models can be integrated into a PRA [Chu 2010]. The QSRMs were identified by reviewing research on methods of modeling digital systems that was sponsored by either the NRC or the National Aeronautics and Space Administration, performed by international organizations, and published in journals or conference proceedings. The strengths and limitations of QSRMs for PRA applications were categorized, described, and evaluated. In addition, a set of desirable characteristics for a QSRM was established. In a later study [Chu 2013], the QSRMs were evaluated against the desirable characteristics to identify candidate methods to apply in case studies.

This study continues the preceding work on software reliability by further developing the BBN method in collaboration with the Korea Atomic Energy Research Institute (KAERI), and applying it to two example systems, namely, the loop operating control system (LOCS) of the Advanced Test Reactor (ATR) at Idaho National Laboratory (INL), and the prototype Integrated Digital Protection System-Reactor Protection System (IDiPS-RPS).

1.2 Objective and Scope

The objectives of the current research are to

- Develop a BBN model for estimating the probability of software failure on demand[2] that is suitable for a PRA, and
- Apply this approach to two example systems in order to obtain insights on its feasibility, practicality, and usefulness in modeling digital systems in NPP PRAs.

Digital-protection systems modeled in a PRA may have multiple failure modes. For example, a reactor protection system (RPS) may fail to generate a reactor-trip signal when a trip condition occurs, or it may generate a spurious trip signal. The scope of this study is limited to modeling protection software failures at a NPP (represented by the probability of failure on demand). That is, the defects/faults considered in the model are those that, if triggered, would cause a system failure to generate a trip signal.

[2] By “demand”, it means a plant condition that requires the actuation of safety systems, for example, the reactor trip system.

Presently, there is no consensus on methods for modeling digital systems in NPP PRAs [NRC 2008, NEA 2009, CSNI 2015]. Different methods have been proposed, including the fault-tree method. However, it remains to be demonstrated whether such models adequately capture the dependencies and the fault-tolerant features of digital systems. There is the possibility that reliability models of digital systems may include software failures representing different software failure modes[3] at different levels of detail (e.g., the software may be modeled at the level of a system, subsystem, or module). For simplicity, this study considered only a system-level failure mode for the protection system to fail to perform its needed function. This definition is consistent with most previous QSRM applications.

[3] Software failure modes in this report are defined as the ways in which software fails from the output perspective. This definition differs from how it is defined in software failure modes and effects analysis, which is the root cause of a software failure.

2 INTRODUCTION TO BBN MODELING

2.1 Background

A Bayesian Belief Network (BBN) is a probabilistic graphical model that uses Bayesian probability, which is a degree of belief in the occurrence of any event based on prior- and observed-evidence [Heckerman 1995, Pearl 1988]. BBNs have appeared in the literature under several different names: Bayesian Nets, Belief Networks, and Causal Probabilistic Networks. BBN methodologies were developed in the 1970s and later used to predict failures in the areas of artificial intelligence, medical diagnosis, information technology, and machine failure in the 1990s [KAERI 2010]. The principles of BBN are discussed in the following sections.

2.1.1 Basic Concept of BBN

The BBN depicts a set of random variables and their conditional independencies via directed acyclic graphs. These acyclic graphs do not form loops and consist of nodes that represent random variables. The nodes are often assumed to have a discrete probability distribution, and the dependencies among them are indicated by arcs [Jensen 2002]. Usually, BBNs are employed when statistical inferences are required (i.e., when evidence for one or multiple nodes is available) and analysts wish to infer the probabilities of other nodes. From such evidence, probability calculus and Bayes' theorem are employed to infer the probabilities of unknown events. The probability of event *A* given *B,* expressed as *P(A/B)*, is determined by the Bayes probability rule as follows:

$$P(A/B)=\frac{P(B|A)P(A)}{P(B)} \quad (2\text{-}1)$$

where

P(A/B): posterior probability,
P(B/A): likelihood,
P(A): prior probability, and
P(B): marginal likelihood.

In probability theory, a joint distribution of *n* random variables can be expressed by the chain rule [Pearl 1988] as

$$P(V_1,V_2,\cdots,V_n)=P(V_1)P(V_2/V_1)P(V_3/V_2,V_1)\cdots P(V_n|V_{n-1},\cdots,V_1) \quad (2\text{-}2)$$

The chain rule is constructed by connecting the nodes with arrows in a hierarchical form, which reflects the nature of the linkage. When two nodes are connected by an arrow, the node to which the arrow is directed is termed a *child* node, while the node from which the child originates is termed a *parent* node. In this construction, a child node is influenced by the parent nodes and, in general, a node is conditionally independent of its non-descendent nodes, given its parent nodes. The concept of causality introduces a hierarchical structure to the network that is usually expressed in a manner similar to a family tree [Krieg 2001].

In probability theory, two events *A* and *B* are conditionally independent, given an event *C,* if the occurrence or non-occurrence of *A* and also of *B* are independent events in their conditional probability distribution, given *C*. Each node can be described by a local conditional probability

distribution function given its parents in the graph, that is, $P(V_i \mid parents\ (V_i))$, wherein *parents($V_i$)* indicate the parent nodes of node V_i. The parent of each node is a direct cause. A BBN represents the joint probability-distribution of all variables, which can be represented by the following chain rule that is employed for Bayesian inferences [Jensen 1996]:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P\left(V_i | parents((V_i)\right) \quad (2\text{-}3)$$

A BBN is defined completely by specifying every term on the right-hand side of Equation 2-3. Note that Equation 2-3 is based on the conditional independence assumption and is a much simpler expression compared with Equation 2-2, which assumes the dependency of every pair of variables.

To illustrate the principle of a BBN, a simple example is provided below. The example is slightly modified from the version developed by Jensen [Jensen 1996] to highlight the properties of BBNs that are important to this study.

2.1.1.1 BBN example about wet grass

Mr. Holmes and Dr. Watson are neighbors living in Los Angeles. One morning, Mr. Holmes notices that his lawn is wet as he is leaving his house. He is interested in knowing whether his wet lawn is due to rain or to having forgotten to turn off the sprinkler the previous night. While pondering these possibilities, he also notices that his neighbor, Dr. Watson's lawn is wet even though he has no sprinklers. How can we determine whether the rain or Mr. Holmes' sprinklers were the cause of his wet lawn?

Mr. Holmes' conundrum can be solved by constructing a BBN to facilitate calculation of node probabilities. Four variables having causal relationships can be used to determine the most probable causes of the two wet lawns: rain, sprinkler, Holmes' wet lawn, and Watson's wet lawn, each of which is abbreviated as *Rain, Sprinkler, H_L_Wet*, and *W_L_Wet*, respectively (Figure 2-1[1]).

[1] It is worth noting that the inherent limitation for any modeling approach is the completeness of the model. In case of this example, other factors (e.g., morning dew) could result in the observed condition (wet lawn). This example was simplified to demonstrate the BBN fundamental concepts. The BBN method, in general, is capable of capturing model details such as morning dew.

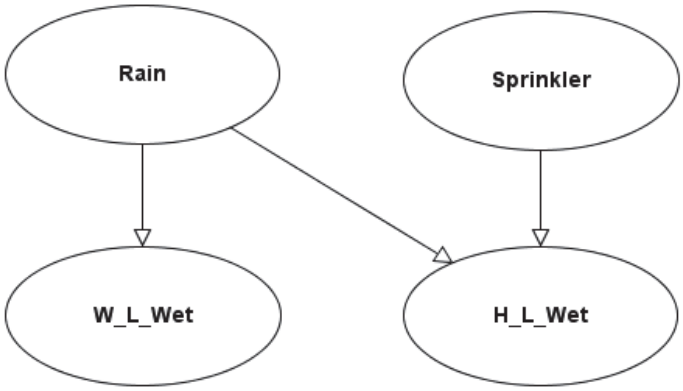

Figure 2-1 BBN Representing Holmes' Reasoning Structure About the Wet Lawn

As shown in Figure 2-1 above, each of the variables can be represented by Boolean nodes with two states (i.e., yes or no). The arrows show the causal relationship between the variables. For instance, Dr. Watson's lawn is wet due to rain, whereas Mr. Holmes' lawn could have gotten wet by the rain or by sprinklers. Thus, the *W_L_Wet* node is connected with only the *Rain* node whereas the node *H_L_Wet* is connected with two nodes, *Rain* and *Sprinkler*. The Boolean nodes in each state can be tabulated in a node probability table (NPT) as shown in Table 2-1; the values therein express the strength of the relationships between the variables.

Table 2-1 Node Probability Table for Watson's Wet Lawn

	Rain	
	Yes	No
W_L_Wet (yes)	1	0
W_L_Wet (no)	0	1

Table 2-1 shows all possible conditional probabilities, including the probability that Watson's lawn is wet given that it rained at night, that is, $P(W\_L\_Wet \mid Rain)$. Thus, if it rained, the probability of Watson's lawn being wet, $P(W\_L\_Wet)$, is unity, and if it did not rain, $P(W\_L\_Wet)$ is zero. Similarly, the NPT containing the variables *H_L_Wet*, *Rain,* and *Sprinkler* for Holmes' wet lawn is shown in Table 2-2.

Table 2-2 Node Probability Table for Holmes' Wet Lawn

Rain	**Yes**		**No**	
Sprinkler	Yes	No	Yes	No
H_L_Wet (yes)	1	1	1	0
H_L_Wet (no)	0	0	0	1

Prior to observing his wet lawn, Holmes initially has a *degree of belief* about whether it rained or whether he left the sprinklers on the previous night. This degree of belief is numerically represented as a *prior probability*. For the case of rain, Holmes could have determined the prior probability of rain based on statistical rainfall data or from expert opinion if statistical data were not available. Table 2-3 lists the prior probabilities for rain and sprinklers.

Table 2-3 Prior Probabilities Based on Initial Degree of Belief

	Yes	No
Rain	0.2	0.8
Sprinkler	0.1	0.9

Using the values in Tables 2-1 and 2-3 along with elementary probability formulas, the probability of Watson's wet lawn can be calculated as follows:

$$P(W\_L\_Wet=yes)=P(W\_L\_Wet|Rain)\times P(Rain)+P(W\_L\_Wet|no\_Rain)\times P(no\_Rain)$$
$$=(1\times 0.2)+(0\times 0.8)=0.2$$

Similarly, using Tables 2-2 and 2-3 for Holmes' lawn,

$$P(H\_L\_Wet=yes)=P(H\_L\_Wet|Rain,Sprinkler)\times P(Rain)\times P(Sprinkler)$$
$$+P(H\_L\_Wet|no\_Rain,Sprinkler)\times P(no\_Rain)\times P(Sprinkler)$$
$$+P(H\_L\_Wet|Rain,no\_Sprinkler)\times P(Rain)\times P(no\_Sprinkler)$$
$$+P(H\_L\_Wet|no\_Rain,no\_Sprinkler)\times P(no\_Rain)\times P(no\_Sprinkler)$$
$$=1\times 0.2\times 0.1+1\times 0.8\times 0.1+1\times 0.2\times 0.9+0\times 0.8\times 0.9=0.28$$

The values 0.2 and 0.28 are the prior knowledge of events *W_L_Wet* and *H_L_Wet*, respectively (Figure 2-2). The BBN updates all the node's probabilities based on the observed evidence as follows:

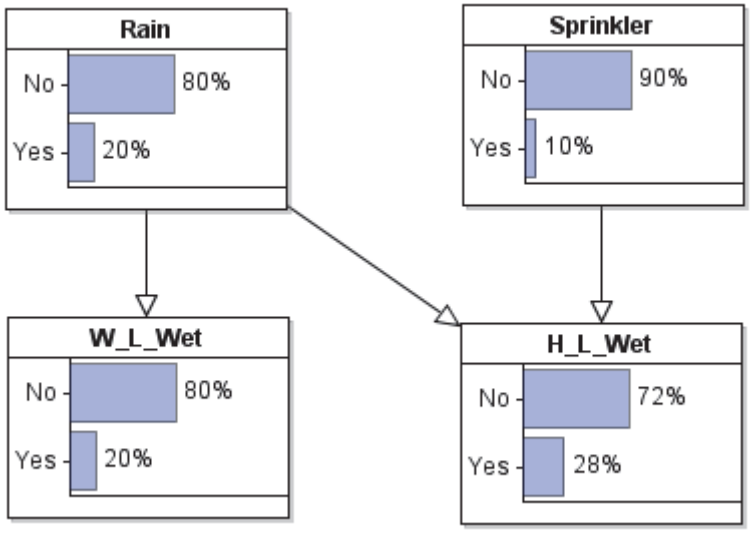

Figure 2-2 Probabilities Without Evidence

1st Evidence: When Holmes observes his lawn is wet (Figure 2-3), the evidence increases his belief about rain at night from a value of 0.2 to 0.714, and the probability of sprinkler operation increases from 0.1 to 0.357, which are estimated as follows:

$$P(Rain/H\_L\_Wet)=\frac{P(H\_L\_Wet|Rain)P(Rain)}{P(H\_L\_Wet)}$$

$$=\frac{[P(H\_L\_Wet|Rain,Sprinkler)P(Sprinkler)+P(H\_L\_Wet|Rain,no\_Sprinkler)P(no\_Sprinkler)]P(Rain)}{P(H\_L\_Wet)}$$

$$=\frac{[1\times 0.1+1\times 0.9]\times 0.2}{0.28}=0.714$$

and

$$P(Sprinkler/H\_L\_Wet)=\frac{P(H\_L\_Wet|Sprinkler)P(Sprinkler)}{P(H\_L\_Wet)}$$

$$=\frac{[P(H\_L\_Wet|Rain,Sprinkler)P(Rain)+P(H\_L\_Wet|no\_Rain,Sprinkler)P(no\_Rain)]P(Sprinkler)}{P(H\_L\_Wet)}$$

$$=\frac{[1\times 0.2+1\times 0.8]\times 0.1}{0.28}=0.3571$$

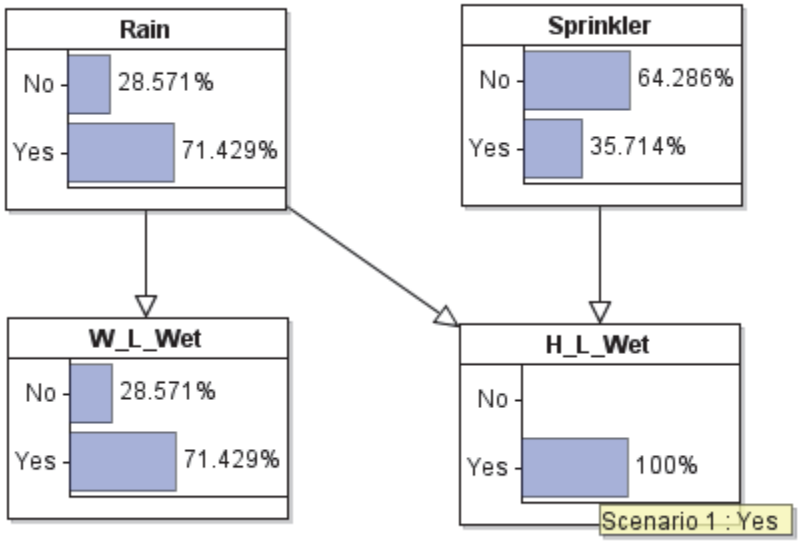

Figure 2-3 Inference From a Single Form of Evidence

2nd evidence: Later, he also recognizes that his neighbor's lawn is wet, and this evidence makes him revise his belief by increasing the probability of rain from 0.714 to 1, and at the same time, his belief about the sprinkler decreases from 0.357 to 0.1 (Figure 2-4). These changes in probability can be estimated by using Bayes' probability rules for multiple variables.

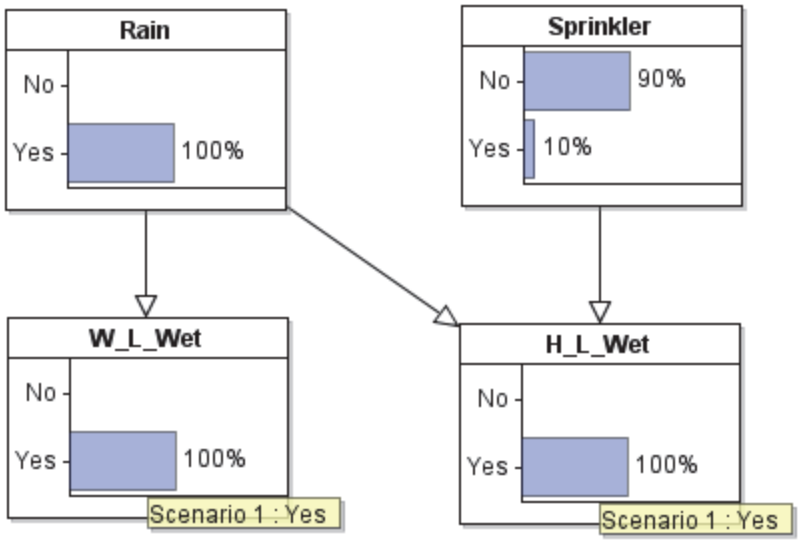

Figure 2-4 Inference From Two Forms of Evidence

2.1.2 Strengths and Limitations of BBN

The advantages of the BBN modeling approach discussed in [Fineman 2010, Fenton 2012] are summarized below:

- Causal factors are modeled explicitly in a BBN as opposed to the purely data-driven classical approaches to statistical modeling.
- A BBN expresses the reasoning from the effect to the cause, and vice versa by updating the probability distributions for each observance of the unknown variables.
- BNN reduces the burden of acquiring parameters since it requires fewer probability values and parameters compared to a full joint-probability model.
- A BBN updates earlier beliefs in light of new evidence and also enables making predictions with an incomplete data set.
- A BBN combines diverse types of evidence, including both subjective beliefs and objective statistical data.
- The approach is user-friendly because of the graphical nature by which the causal relationship among the variables is easily captured.

There are also some inherent limitations to the BBN that should be noted. The limitations outlined below are based on literature reviews [Kragt 2009, Uusitalo 2007, Pollino 2008, Nyberg 2006]:

- Complex causal relationships may be represented by different BBN structures. Therefore, the most meaningful and efficient structure might be difficult to identify.
- The independence and conditional-dependence assumptions for BBN nodes are difficult to verify.
- Quantifying the node probability tables becomes very complex as the number of input nodes increases.

- Prior probability calculations are very crucial: both overestimation and underestimation of the prior beliefs can cause erroneous results.
- Discretization of continuous variables is a common and useful way to keep the size of a BBN manageable. However, the discretization only captures the rough characteristics of the original distribution [Uusitalo 2007]. Moreover, the results are influenced by how discretization is done in terms of the number of intervals and division points [Myllymäki 2002].

2.1.3 Conditional Independence and Dependence

In a BBN, when the nodes are conditionally independent, the connection is known as a d-separation. In Figure 2-5, the two variables, *X* and *Y,* are d-separated when *Z* is known because every path between *X* and *Y* is blocked by *Z*. Variables *X* and *Y* are said to be conditionally independent if the probability distribution of *X* is independent of the value of *Y*, given the value of *Z*, that is, *P(X/Y,Z)=P(X/Z).*

If the two variables are not conditionally independent, then they are considered to be d-connected. This is illustrated in Figure 2-6, in which two nodes, *A* and *B*, are d-connected. If some knowledge about the state of *C* is available, then some knowledge of *B* may be inferred from *A*. Therefore, *A* and *B* are d-connected, or conditionally dependent when *C* is known.

In the wet-grass example in Section 2.1.1, the *Sprinkler* and *Rain* nodes are independent when there is no information about Holmes' grass. However, when information about Holmes' grass is available, *Sprinkler* and *Rain* become dependent on each other.

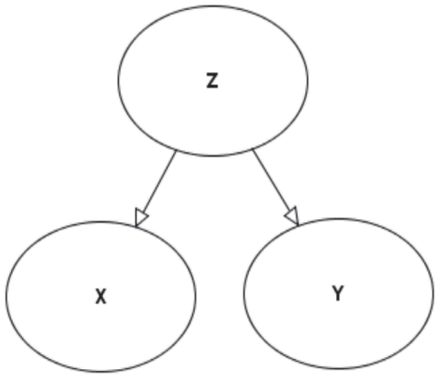

Figure 2-5 d-separated Connection

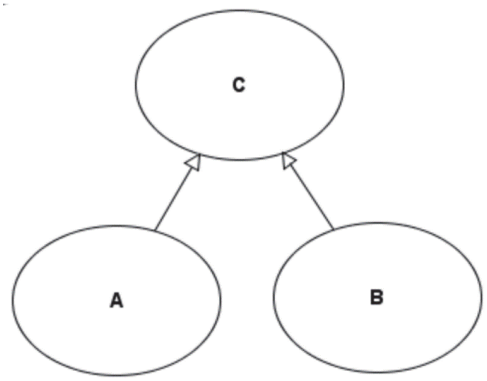

Figure 2-6 d-connected Connection

2.1.4 Structural Properties of BBN

A literature review [Fenton 2012, Krieg 2001] shows that, irrespective of complexity, a BBN is fundamentally built on three types of connections: Serial, diverging (d-separated), and converging (d-connected). The properties of these connections are discussed with examples below.

1. Serial connections

Consider the example shown in Figure 2-7 that represents a serial connection consisting of *F*, *W*, and *T*. Suppose that these three variables represent three events associated with an electrical transformer failure, in which *F* stands for a cooling fan failure, W for a transformer-winding failure, and *T* for a transformer-output failure. When there is information about a transformer's cooling fan failure (*F*), this knowledge increases our belief about a winding failure (*W*), and a transformer's output failure (*T*). In a situation where *W* is certain, then the information about *F* becomes insignificant to the incident of *T* because the belief about transformer failure will not be changed. For example, the failure of the cooling-fan is not an important piece of information in determining the transformer's output failure if a winding failure is already recognized.

It is notable that a serial connection of a BBN transmits evidence from the parent to the endmost node if the intermediate node does not have firm evidence. When the status of intermediate node, *W*, is known, the *F* and *T* nodes are conditionally independent.

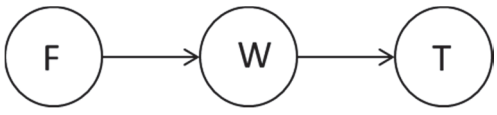

Figure 2-7 Serial Connection

2. Converging connections

In a converging connection, a child node is connected with multiple parents. This is used when there are multiple causes of a common effect. Figure 2-8 presents a converging connection with multiple parent nodes (e.g., overloading, over-frequency, or an internal fault can be causes of generator failure).

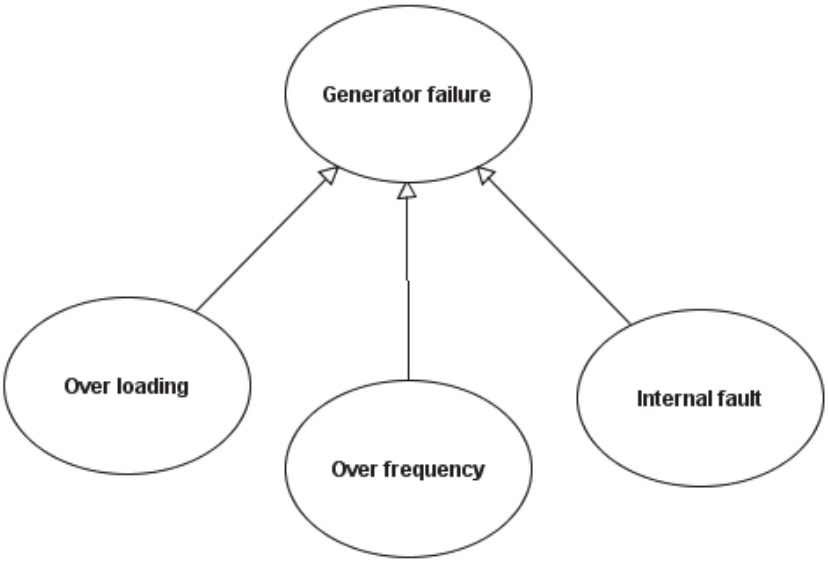

Figure 2-8 Converging Connection

In this example, when the failure of a generator is recognized, the operator's belief about all causes increases. However, when it is confirmed that the generator was overloaded at the pre-trip moment, there is a decrease in the belief about the other two causes. Hence, in a converging connection, the evidence is transmitted among the parent nodes when the status of the child is known. Thus, the parent nodes are conditionally dependent on a given status of the child.

3. Diverging connections

In a BBN, a connection is known to be diverging when a parent node possesses multiple child nodes, and influence is passed to all child nodes from the parent node [Krieg 2001]. In this type of connection, the state of the parent node can be inferred when the states of the child nodes are known, and vice versa (Figure 2-9).

The diverging connection is modeled for a common cause of many effects. The properties of this connection can be explained with the example of a loss of coolant accident (*LOCA*). In an NPP, a *LOCA* is the common cause for a drop in the pressurizer's pressure, an increase in the level of the sump, and an increase in the containment's radiation.

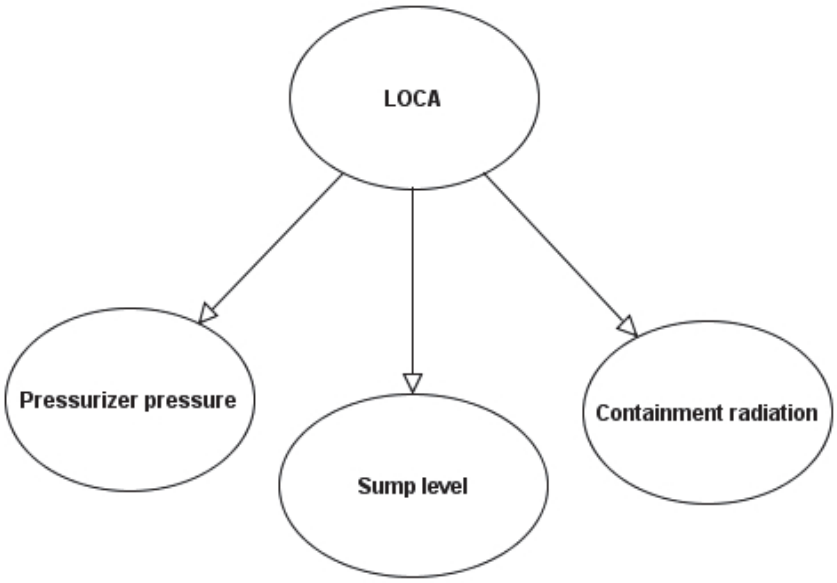

Figure 2-9 Diverging Connection

When there is no firm information about a *LOCA* occurrence, firm evidence of a drop in the pressurizer pressure *increases* the operator's belief about a *LOCA* occurrence which, in turn, raises the belief about both the containment radiation and sump level-increase. Later, when the increase in the sump level is confirmed, the operator's belief about a *LOCA,* as well as the expectation of increased containment's radiation, become stronger.

On the other hand, if the operator has concrete evidence that a *LOCA* happens in the plant, the evidence of a drop in the pressurizer pressure does not change the operator's belief about the level of the containment's radiation.

It is evident that in a diverging connection, the evidence of a child node is transmitted to the other child nodes, provided that the parent is unknown. When the parent node is given, the child nodes of a diverging connection are conditionally independent.

A diverging connection approach is well-suited to a situation when some indirect measurements are available [Fenton 2012]. When direct measurement is impossible and only indirect measures can be taken, they are called *indicators* because they provide an indication of the underlying unknown cause. Let us assume that we would like to measure the quality of a manufacturing process based on three measurable indicators: Defects found, customer satisfaction, and staff turnover. Figure 2-10 shows a BBN model with these indicator nodes.

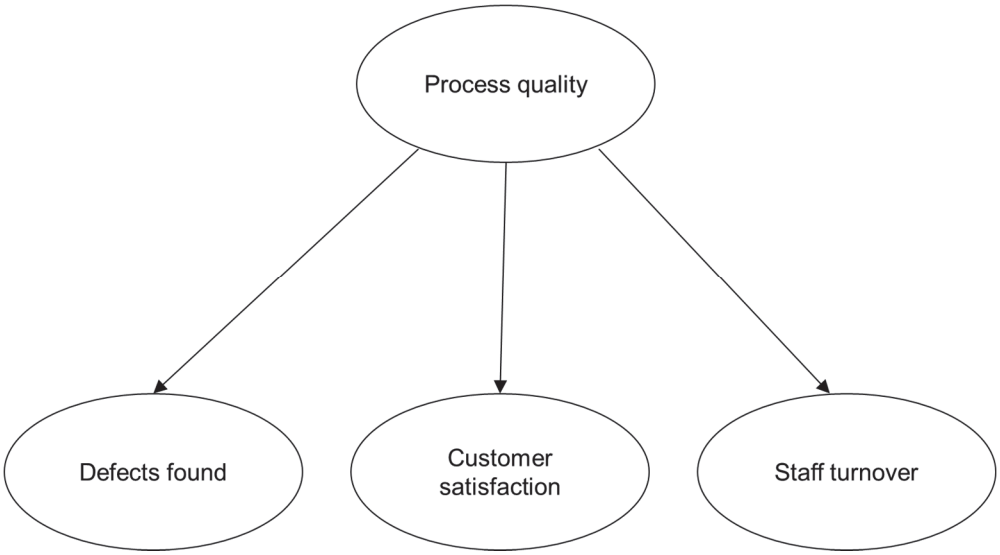

Figure 2-10 Measurement Using Indicators [Fenton 2012]

A clear relationship between process quality and the indicators is evident. The strengths of the linkages are captured in NPTs of the indicator nodes. The NPT contains probability information based on the belief relationships between the parent and child nodes in the model.

Fenton explained that the indicator nodes were correlated with each other by virtue of the structure of the BBN. In the example in Figure 2-10, the causal node (Process Quality node) is not observable, and it is measured by the indicator nodes. In this structure, the belief of an indicator node increases the beliefs of other indicator nodes. For example, high customer satisfaction and low staff turnover are expected if the number of defects found is low. The relationship is desirable when the indicator nodes reflect the true state of the underlying unknown cause. The indicator nodes are not independent when the causal node is unobservable [Fenton 2007b].

2.2 BBN Modeling Process

A BBN model may be developed in three steps: 1) structural development 2) quantification of the NPTs, and 3) evidence collection. These three steps for developing a BBN model are described below.

2.2.1 Structure Development

By using a BBN, we can estimate the certainties of unobservable events. All variables relevant to the envisioned BBN need to be identified before structuring the network. Each variable becomes a node in the BBN.

When two nodes are connected, the node that causes another specific node is denoted as a parent, and the caused node a child. A child node is influenced by the parent nodes. A node without parents is known as a *root* node, and a node without a child is known as a *leaf* node. These nodes are terminal nodes in a Bayesian network. A node that represents a variable with a single value that has been observed with a probability of one [Suermondt 1992, Krieg 2001] is termed an *evidence* node (or an observed node, or an instantiated node).

BBN nodes are discrete and can be transformed into different types (e.g., Boolean, ordered values, or integral values). Boolean nodes are used to represent propositions by using the binary values, *true* or *false*. The nodes of ordered values are used to represent node values in the order such as *low*, *medium*, *high,* and the integral value nodes express node states with integer numbers [Korb 2010].

The structuring of a BBN is the process of building a causal structure with nodes and the dependencies among them. The generic steps of designing a BBN discussed in [Heckerman 1995, Krieg 2001] are (i) recognizing the goals of the intended modeling, (ii) identifying the possible indicators or observations to achieve the goals, (iii) sorting the collected observations that are meaningful to the model, and (iv) organizing the indicators or observations into variables that have mutually-exclusive and collectively-exhaustive states.

While determining the causal structure of a BBN, it is essential to pay attention to establishing proper conditional probabilities and dependencies between information nodes and hypothesis nodes [Krieg 2001]. The links between these variables are directed from the causes to their effects. However, in the case of an obscure relationship, the arrow is directed from a higher abstraction to a lower one, or from a more general concept to a more detailed one [KAERI 2013]. The examples in Figure 2-9 and Figure 2-10 show this relationship, that is, from a higher abstraction to a lower one and from a more general concept to a more detailed one.

2.2.2 Quantification of the Node Probability Table

An NPT is used for quantifying node probabilities for various node states. The root nodes are allocated with the prior distribution, and the others are allocated with their conditional probability distributions. After depicting the structure of the network, the node probabilities are quantified by assigning a conditional probability distribution for each of the nodes and then building an NPT. For each child node, the NPT represents all possible combinations of the parent states. To prepare them, the BBN parameters should be well defined. Generally, NPTs are derived through one of the following methods: direct expert judgment, estimation from datasets, and representing the relationship between variables in terms of equations [Pollino 2008]. In this study, to account for the uncertainty associated with the estimates that different experts provided, we used NPTs that are tables of random variables rather than constants. The random variables are probabilistic distributions that were estimated using expert elicitation.

2.2.3 Evidence Collection and Posterior-Probability Calculation

The evidence in a BBN represents information about a current situation. Two types of evidence, *hard evidence* and *soft evidence*, are observed for events. *Specific evidence* or *hard evidence* refers to certainty about the evidence. For example, while examining steam generator tube rupture, if a high radiation level is observed in the secondary side, then the evidence of *Secondary Side Radiation High* is true, and therefore is known as *hard evidence*. In a BBN, when a node obtains *hard evidence*, the probability assigned for the corresponding state of the node is one.

In other situations, the evidence might be a probability distribution. For instance, suppose that the plant operator is not completely sure whether or not the secondary side radiation is high. The operator might be 80% sure. In such a case, the evidence is termed as *soft evidence or uncertain evidence*. Posterior probability is estimated by incorporating the evidence or new knowledge in the network

3 OVERALL APPROACH

Chapter 3 summarizes the overall approach to quantifying software failure probability based on the quality of software development and on verification and validation (V&V) activities performed throughout the software development life cycle. Following a brief overview of the approach in Section 3.1, subsequent sections in this chapter discuss the individual steps in more detail.

3.1 Introduction

One of the main objectives of this work is to develop a model that estimates the probability of software failure based on observations of the quality of the development and V&V activities throughout the software development lifecycle. The probability of software failure is assumed to be proportional to the number of software defects remaining after its installation for the sake of simplicity and for demonstration purposes.

Figure 3.1 shows the major steps used to develop the model which consist of three phases of expert elicitation. First phase of expert elicitation is used to identify the attribute which represents qualities of activities that are carried out to accomplish the functions of each SDLC phase and to verify causal relationships represented by the BBN structure. Last two phases of expert elicitation were used to quantify the causal relationship parameters for the model, and to provide input values of nodes for applications to the two example systems.

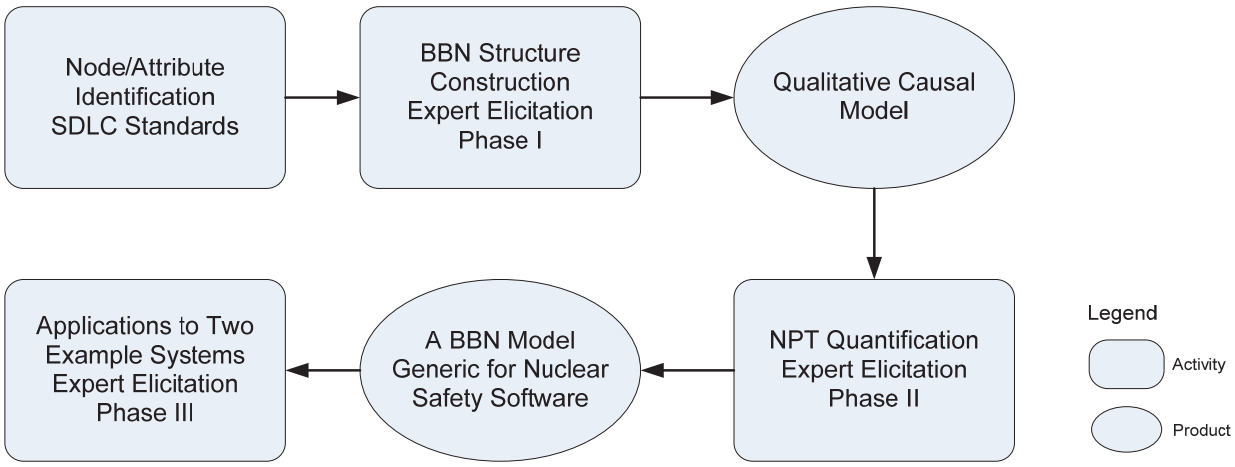

Figure 3-1 Major Steps in the Quantification of Software Failure Probability

Activities described in various software development and V&V standards were examined to develop the attributes that serve as indicators of the quality of development and of the V&V. Attributes were clearly defined and a quantification scheme (such as a three ordinal levels “low”, “medium” and “high”) was defined for each attribute (please refer to Section 4.2.1 and Chapter 4 for details).

The research team then identified causal relationships among these attributes, the intermediate nodes (such as development quality, or V&V quality), and the end node (such as the number of defects remaining or failure probability). The output of these two activities is a “Qualitative Causal Model”. This model, including the list of attributes, their definitions and quantification schemes, was reviewed by Phase I experts. It was then revised by addressing experts’ comments and

recommendations and served as input into the next step. It is worth noting that this captures the generic causal relationships among SDLC activity characteristics, development quality, and V&V quality in terms of number of defects remaining. The Phase I Experts are known professionals in software development and include systems engineers, software development managers, and software quality assurance experts.

After the qualitative causal model was consolidated and finalized, a second round of expert opinion elicitation was conducted to quantify these causal relationships in the model. It is generally recognized that this is the most challenging but critical step for the success of any BBN application. Challenges come from the large amount of data a model needs and/or lack of data. Though experts were used in this step, the experts' inputs are difficult to validate and verify due to the dearth of data, which potentially introduces large amounts of uncertainty into the model and consequently lowers its credibility.

Although most of the software development data is proprietary and not accessible to the research team, some software development measurements can be found in the literature [Jones 2008]. In addition, limited V&V data are available for the development of two trial systems. Both the literature data and trial systems' development data are observations that enable the Bayesian update of the experts' inputs to reduce uncertainty.

This study assumes that the quantitative causal relationships are "generic" for safety software development in the nuclear industry, which is unique among industries due to the homogenized SDLC process and quality assurance programs defined by regulation requirements. The causal relationship was quantified based on nuclear safety-related software experts and thus this model is appropriate to estimate the failure probability of the nuclear safety-related software. The quantitative causal relationships were further Bayesian updated using literature data and trial systems development data.

Since experts for the first two rounds dealt with "generic issues" of the BBN model, they are termed "generic experts" in this report. The model prior to its instantiation is called the model for nuclear safety software, and the two instantiated models are "system-specific models".

The last step of this study was to provide inputs of indicator nodes for each trial system. Two experts who are familiar with LOCS and IDiPS-RPS developments assigned values to nodes of these two models. The research team executed the two models using WinBugs [Spiegelhalter 2003] to obtain the number of defects remaining. A linear relationship was assumed to exist between the number of remaining defects and the failure probability for the sake of simplicity and for demonstration purposes. The coefficient of this linear relationship was estimated based on digital safety software failure and demand records, and the failure probabilities per demand for LOCS and IDiPS-RPS were estimated based on the numbers of defect remaining.

The rest of this chapter describes the BBN model in more details.

3.2 Model Overview

The goal of the model is to estimate the total software defects remaining at the end of the last phase, Installation and checkout, which then is converted to the probability of software failure. The model can be represented as shown in Figure 3-2, which starts with the number of defects remaining in the requirements phase and tracks the number of defects through all five phases of the life cycle (Requirements, Design, Implementation, Test, and Installation). Chapter 4 gives more details of the Bayesian Belief Network (BBN) model.

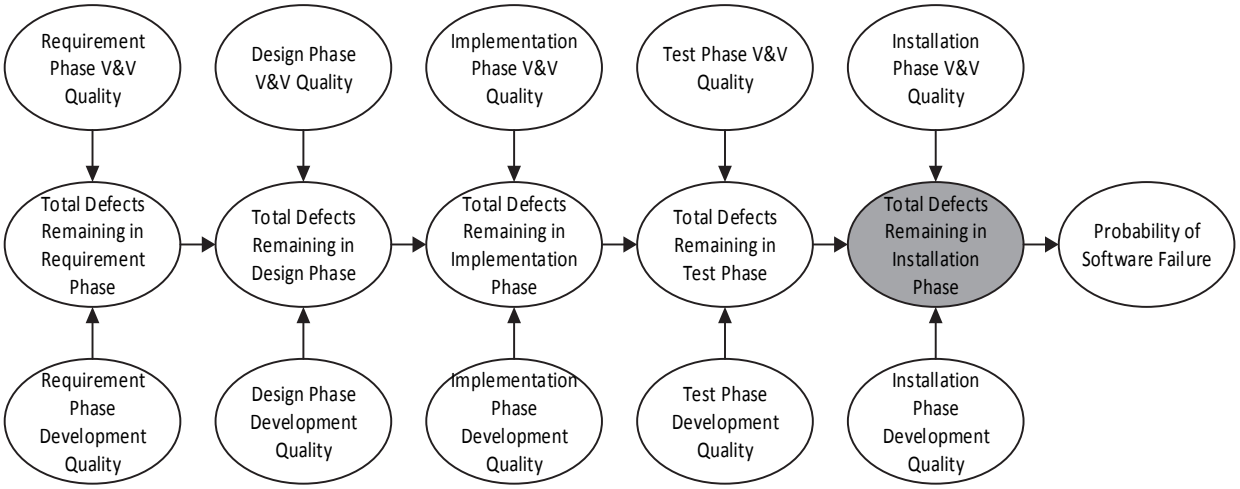

Figure 3-2 Overview of the BBN Model

The number of defects remaining in each phase is a function of the phase development quality (the developmental process adds defects) and the phase V&V quality (the V&V process removes defects) in the same phase. At the end of each phase, remaining defects are passed on to the next phase.

For example, the estimation of the number of defects remaining in the software at the end of the design phase is shown in Figure 3-3. The estimation starts from the "Quality of Development in SD Phase" and "Quality of V&V in SD phase". These quantities are represented in the Quality of Development in SD Phase and the Quality of V&V in the SD nodes in Figure 3-3. In the model, the quality nodes can be in any one of three states, High, Medium, and Low. A High-development quality state, for example, represents a very good development quality (above the norm) that will lead to fewer than normal new defects introduced in the design phase.

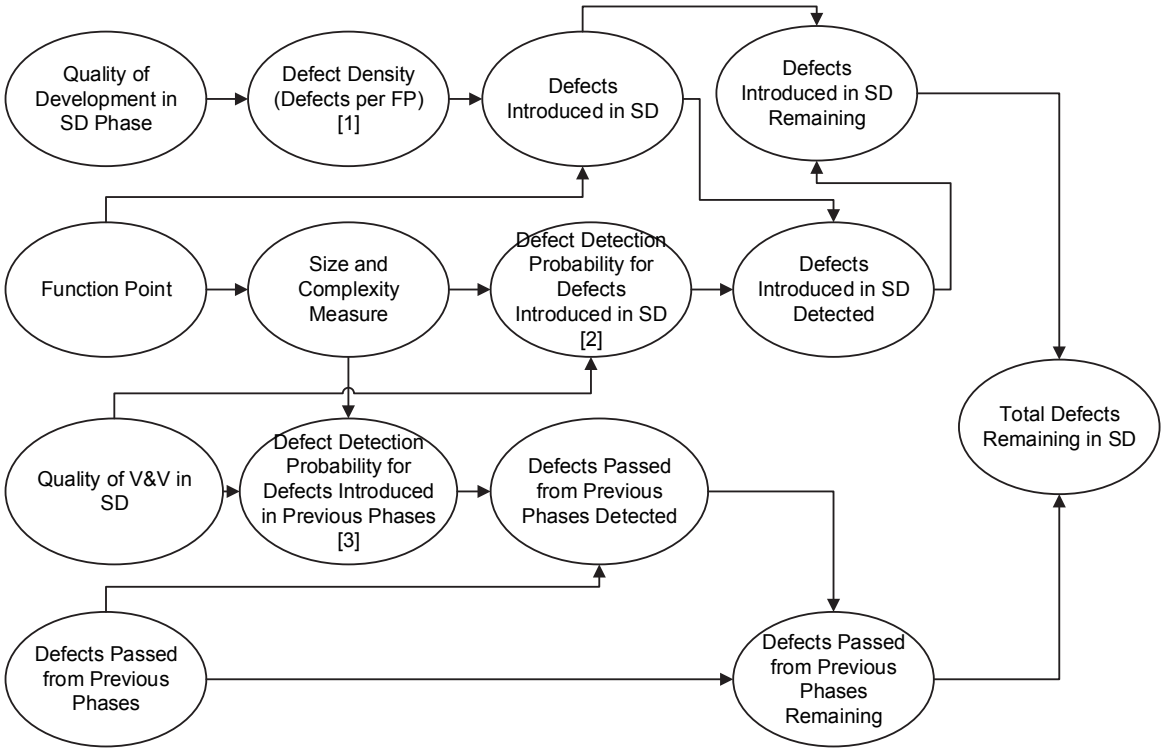

Figure 3-3 BBN Model for the SD Phase

States of "Quality of development" and "Quality of V&V" nodes determine

(1) defect density (number of defects inserted per function point[1])
(2) defect detection probability for defects introduced in the current phase
(3) detection probability for defects introduced in previous phases

These three quantities were estimated from expert elicitation. The following equations show how the total defects remaining at the end of the SD phase are calculated.

Defects remaining in SD = Remaining defects introduced in SD
+ Remaining defects passed from Req phase

(3-1)

Remaining defects introduced in SD = Defects introduced in SD

[1] Function point is a *de facto* measurement for software size. Please refer to Section 4.2.3 for more details.

- Detected defects introduced in SD

(3-2)

The number of defects introduced in a phase is a product of the Defect Density (which is the number of defects introduced in that phase per function point) and the number of function points:

*Defects Introduced in SD = (Function Point) * (Defect Density)*

(3-3)

A defect introduced in a phase has a probability $p_{previous}$ of being removed by the V&V activities in that phase. In the model, the actually number of defects removed is sampled from a binomial distribution. For example, the defects passed from previous phases that are detected in the SD phase V&V activities are sampled from a binomial distribution:

$$k \sim \binom{N_{previous}}{k} p_{previous}^{k} \left(1 - p_{previous}\right)^{N_{previous}-k},$$

(3-4)

where

k = Defects passed from previous phases detected
$N_{previous}$ = Defects passed from previous phases
$p_{previous}$ = Defect detection probability for defects introduced in previous phases

The defects introduced in the SD phase that are detected are calculated similarly.

3.3 Estimation of Development and V&V Qualities

The quality nodes for development and for V&V in the model are unobservable, which makes data collection challenging. Instead, various indicators (attributes) were developed to provide indirect indications for these quality nodes. In this work, the attributes were developed based on reviewing various guidance documents and standards on software development and V&V activities. These attributes were assigned as indicators to either the development quality node or the V&V quality node at each of the five phases of the developmental cycle. In applying the model to specific software, experts scored these attributes; the scores were then used to obtain a probability distribution (i.e., a probability indicating the likelihood of a node being in each state) for the quality and V&V nodes. An example of the attributes is shown in Table 3-1, which lists the attributes for quality of development and quality of V&V for the SD phase. Chapter 1 describes the process of identifying attributes and a scoring framework. A list of complete attributes is defined in Appendix B.

Table 3-1 Indicators (Attributes) for the Quality Nodes in the SD Phase

Quality of Development in SD Phase Indicators	Quality of V&V in SD Phase Indicators
Development of a Description of Software Architecture	Design Evaluation
Development of a Description of Software Design	Interface Analysis V&V
Traceability Analysis - Design Phase	Traceability Analysis V&V- Design Phase
Criticality Analysis- Design Phase	Criticality Analysis V&V- Design Phase
Hazard Analysis- Design Phase	Hazard Analysis V&V- Design Phase
Security Analysis- Design Phase	Security Analysis V&V- Design Phase
Risk Analysis- Design Phase	Risk Analysis V&V- Design Phase

Quality of Development in SD Phase Indicators	Quality of V&V in SD Phase Indicators
Software Component Test Plan Generation Software Integration Test Plan Generation Software Component Test Design Generation Software Integration Test Design Generation Software Qualification Test Design Generation Software Acceptance Test Design Generation Configuration Management- Design Phase Review and audit - Design phase	V&V Software Component Test Plan Generation V&V Software Integration Test Plan Generation V&V Software Component Test Design Generation V&V Software Integration Test Design Generation V&V Software Qualification Test Design Generation V&V Software Acceptance Test Design Generation Configuration Management V&V- Design Phase Review and Audit- Design Phase V&V Design Phase Activity Summary Report Generation

3.4 Use of Expert Elicitation

Three phases of expert elicitation were used to develop and verify the connections among the nodes (causal and conditional independence relations) in the BBN model, to develop the prior and conditional probability distributions, and to score the attributes. In the first phase of expert elicitation, thirteen experts were each sent a questionnaire on various aspects of the structure of the BBN model. The following topics were covered in the questionnaire:

- the appropriateness of using 5 phases to represent the development and V&V activities
- the correctness of the causal relationships represented in the BBN
- the use of 3 states to score various nodes in the model
- the correctness of the conditional independence assumptions
- the use of function points instead of Lines of Code (LOC) to measure software size
- the completeness of the attributes used as indicators for the development and V&V qualities

The responses obtained from this solicitation were used to adjust the BBN model accordingly. The answers received from the experts, together with the model adjustments that were made, are summarized in Section 7.1.

The second phase of elicitation covered the quantitative aspects (i.e., the prior and conditional probability distributions) of the model. In this phase, experts from 7 different organizations were selected to participate. The following parameters were elicited:

- the prior distributions for the development and V&V Quality nodes
- the prior distribution for the Function Point node
- the conditional probability distribution for the attributes, given the development and V&V quality
- the number of defects introduced, given the development quality
- the probability of removing defects, given the V&V quality

The answers from the experts were aggregated by fitting probability distributions over the data points. Section 7.2 describes the results from this round of elicitation in more detail.

Finally, the third phase of elicitation used two experts (one expert each for the LOCS and IDiPS-RPS systems) to score the attributes for the development and V&V activities of the LOCS and IDiPS-RPS systems. The scores were used to evaluate the BBN models. Details of this round of elicitation are described in Section 7.3.

3.5 Fault Size Distribution

The number of remaining defects can be converted into a probability of software failure on demand using the Fault[2] Size Distribution (FSD) method. The FSD method [Delic 1997] is similar to the Fault Exposure Ratio (FER) method that is commonly used in software reliability growth methods [Musa 1987] as a coefficient between the software's failure rate and the number of defects in a software program. In this study, the failure probability per demand is assumed to be proportional to the number of remaining defects for the sake of simplicity and for demonstration purposes only. The FSD is the coefficient of this proportional relationship. It is calculated from the average failure probability per demand derived from the operating experience and the representative number of residual defects in safety-related NPP software as described in Section 6.3 .

3.6 Evaluating the Model

To account for uncertainty in the parameters (such as the number of defects inserted at a phase given a certain development quality), conditional probability *distributions* were used in modeling the elements of the NPTs instead of constant values. These distributions were generated by fitting a probability distribution model (e.g., a normal distribution) over the data points obtained from expert elicitation. The model was evaluated using WinBUGS [Spiegelhalter 2003], which uses Markov chain Monte Carlo (MCMC) to solve the Bayesian inference problem posed in the model.

In the MCMC model, a full joint prior distribution is specified on all quantities, whether parameters or observables, and samples of the unknown parameters are generated from their posterior distribution given that some of the stochastic nodes were observed. The basic idea behind the sampling algorithm is to successively sample from the conditional distribution of each node, given all the others in the model. It can be shown that, under broad conditions, this process eventually provides samples from the joint posterior distribution of the unknown quantities [Atwood 2002]. Empirical summary statistics can be established from these samples and used to draw inferences about their true values.

For this model, the conditional-probability distributions that were estimated using expert elicitation are used as prior distributions of the model. WinBUGS generates a large number of samples (as specified by the user) corresponding to points sampled from the conditional probability distributions. These samples effectively are a set of "instantiations" of various nodes in Figure 3.3. For example, one sample will contain a point value for defect density, probability of defect removal, development quality, etc. These point values are then used in calculating the final number of defects remaining (for that instantiation) in the software as outlined in Section 3.2. Over the large number of samples taken by WinBUGS, a distribution of the final defects remaining was obtained.

[2] The term "fault" is used interchangeably with the term "defect" in this report.

4 A BBN MODEL FOR SAFETY SOFTWARE FAILURE PROBABILITY

This chapter describes the Bayesian Belief Network (BBN) model that estimates the number of residual defects in a software program and links the number of residual defects to the software's failure probability.

This BBN model (1) modifies the earlier work by Eom [Eom 2009, Eom 2013] that used Fenton's approach [Fenton 2007a, Fenton 2008] to estimate the number of remaining faults in a software program, and (2) converts the number of faults to a software failure probability using the concept of FSD [Littlewood 1980, Delic 1997].

In this study, the BBN is developed for safety-related software systems. The model structure was built using information about safety-related software and the causal relationships in the model represent those of the safety-related software. For example, the definitions of states of the nodes were in terms of safety-related software, and the questionnaires used in the expert elicitations were based on safety-related software. The specific evidence was collected from experts who are familiar with the development and V&V of the trial systems and used as inputs to the BBN model. The expert elicitation process is discussed further in Chapter 7.

This model assumes that the quality of the SDLC directly impacts the number of remaining defects. The model assumes that such impacts can be expressed in terms of the faults that may be introduced by the development activity group, and in terms of faults that can be detected and removed by the activity of the V&V group. The quality in carrying out these activities is assessed by (1) developing the required activities (called attributes) of a safety-related system for each phase of software development, and, (2) evaluating the software under study against these attributes. The quality of these attributes is aggregated via the BBN model to estimate the number of residual faults and the software's failure probability after the software is deployed in the plant.

4.1 High-Level BBN Structure

In this BBN model, the SDLC is abstracted into five phases: requirements, design, implementation, test, and installation/checkout. For each phase, a BBN model was developed to estimate the number of faults remaining in the software at the end of the phase. The use of a sequential model is a simplification of the actual SDLC that is often iterative. Since this model estimates the probability of software failure of a system during the plant's operation, the assessment using the model should be done after the installation and checkout phase is completed, and any iteration during the development process have been completed. The assessment also uses any data that's available on the number of faults detected in each phase. The following is a summary description of each phase. The BBN model of each phase models two types of activities: Development and V&V. In general, the development team carries out the development work, and the V&V team independently undertakes the V&V[1] activities.

[1] V&V is performed by an organization that is technically, managerially, and financially independent of the developmental organization.

4.1.1 Requirements Phase

The software concept phase and the software requirements phase (i.e., IEEE 1012 [IEEE 1012]) are merged as the Software Requirements phase in our BBN model. The concept activity represents the delineation of a specific implementation solution to resolve the user's problem. During the concept activity, the system's architecture is selected and the system requirements are allocated to hardware, software, and user-interface components. A key aspect of the concept document is the development or selection of the system's architectural design and the system's requirements. The purpose of formulating the Software Requirements Specifications (SRS) is to satisfy the system requirements and user needs. The SRS should be correct, consistent, complete, accurate, readable, testable, and robust.

4.1.2 Design Phase

The objective of software design is to translate the software requirements specifications into an architectural description and a design description. The design includes databases and system interfaces between hardware, operator/user, software components, and subsystems. The design activity includes the software's architectural design and detailed design.

4.1.3 Implementation Phase

The objective of implementing the software is to produce computer codes from the design and to ensure that the code is correct, accurate, and complete, and conforms to the design. In this phase, the software design is transformed into code, database structures, and related machine-executable representations. The Software Implementation activity addresses its coding and testing, including the incorporation of reused software products [IEEE Std 1012].

4.1.4 Testing Phase

Testing is an activity in which a system or component is operated under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. Test activity includes planning tests, designing them, generating test cases, and generating test procedures, executing test procedures, and evaluating/approving test results. The test-phase activities include testing the software's integration and executing qualification tests, traceability analysis, hazard analysis, security-, and risk-analyses, as appropriate.

4.1.5 Installation and Checkout Phase

During installation and checkout, the software product is installed and tested in the target environment. The Software Installation and Checkout activity supports the overall system-installation activities.

Figure 4-1 shows the high-level structure of the BBN model. The number of faults at the end of a phase becomes an input to the BBN model of the next phase. At the end of the last phase (i.e., installation and checkout), the number of faults in the software is converted into a software- failure probability on demand. Section 4.2 describes the detailed model using a representative phase. The complete model for all phases is described in Appendix A.

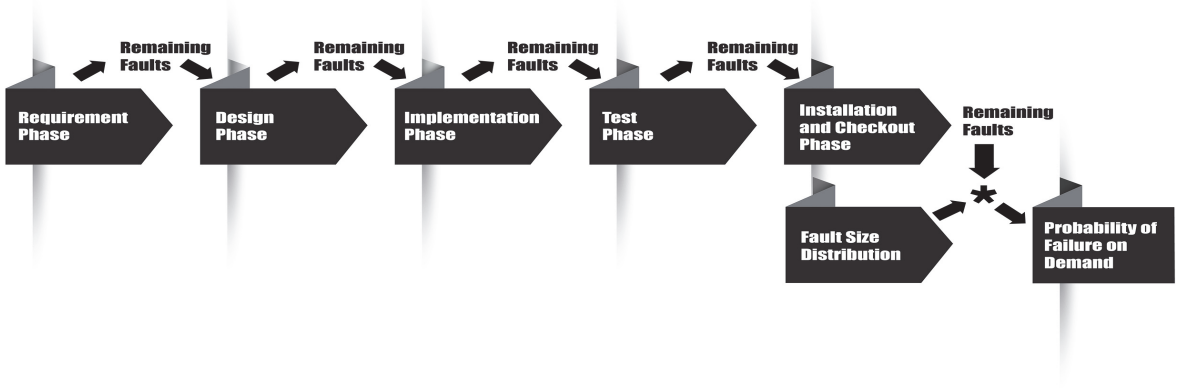

Figure 4-1 High-level Structure of the BBN Mode

4.2 A Sub-Model for SD Phase

This study models each of the five phases depicted in Figure 4-1 in a similar way. The model for each phase estimates the number of residual faults at the end of the phase. The number of residual faults in a specific phase is determined by (1) the number of defects passed from the previous phase, if the current phase is not the first phase of the SDLC; (2) the number of defects introduced during the current phase, and, (3) the number of defects detected and removed by the V&V activities undertaken in the current phase. The process of inserting and removing defects is modeled as a BBN model, as illustrated in Figure 3-3, for the SD phase. The model for other phases are the same except for the Test phase which has an additional arrow from the test activities, which affect the probability of detecting faults. Attributes for development quality and V&V quality are shown in Figure 4-2 and Figure 4-3, respectively.

An important assumption here is that the quality of the software development activities (represented by the node "Quality of development in SD" in Figure 3-3) is related directly to the defect density (per function point) in the current phase. Similarly, the quality of the V&V activities (represented by the node "Quality of V&V in SD" in Figure 3-3) is directly related to the detection (and thus removal) probability for defects introduced in current phase and defects passed from previous phase. Other factors that affect the number of faults inserted are the size and complexity of the software. A single parameter, i.e., number of function points, is used to capture the effects of both. No arrow is used to connect the number of function points (designated as “Size and Complexity Measure”) to the defect density because this number already includes an adjustment factor due to complexity. Section 4.2.3 provides a definition of function point. The number of function points also affects the probabilities of detecting defects.

The key to the modeling is to capture all major attributes that affect the qualities of the developmental and V&V activities, and in turn, the software’s reliability. In this study, we decided to use the diverging configuration for the attribute nodes. As shown in Figure 4-2 and Figure 4-3, the quality nodes in the center have attributes connected to them in a diverging configuration, wherein the attribute nodes are modeled as indicator nodes similar to the

indicator nodes of Fenton [2007b] [2]. The indicator nodes represent the quality in carrying out the associated activities of the attributes.

The attribute or “indicator nodes” each has 3 states. The quality nodes are modeled as root nodes and each have 3 states. In each state of attribute and quality nodes, a probability distribution is used to represent the uncertainty. Chapter 5 details how the attributes were developed and scored in this study.

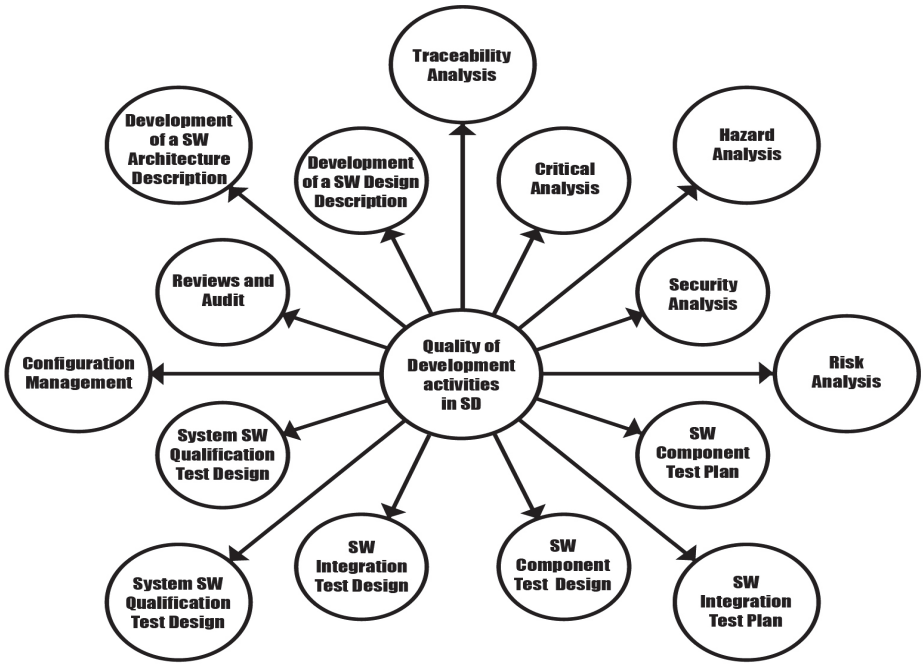

Figure 4-2 Attributes Nodes of Quality of Development in the Design Phase

[2] The indicators used in this study are related to their parent node by a numerically specified NPT, while Fenton uses ranked nodes to define this relationship.

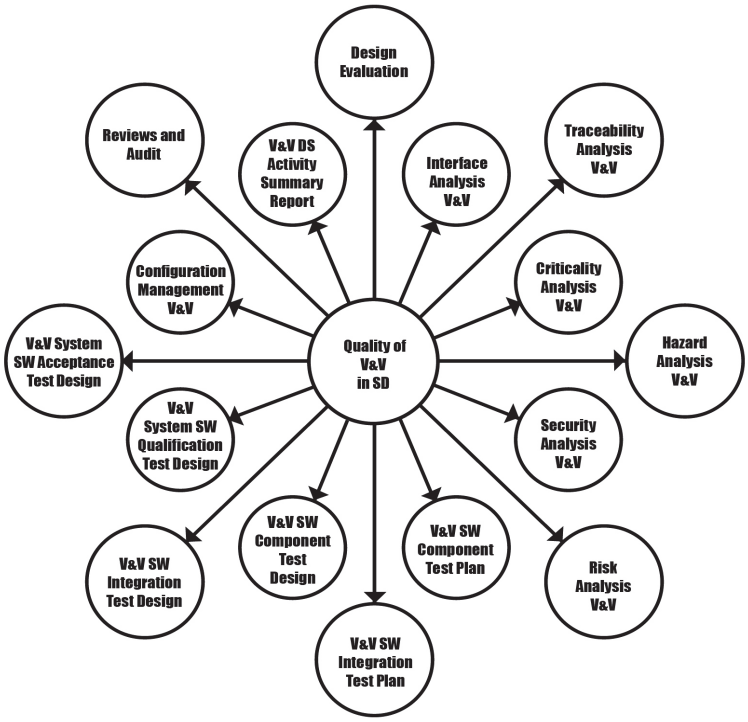

Figure 4-3 Attributes Nodes of Quality of V&V in the Design Phase

For the rest of the model (1) the number of defects per function point introduced during development is derived from the quality of development activities; (2) similarly, the two probabilities of detecting defects are determined by the quality of V&V and the complexity of the software, (3) the number of defects detected and removed is modeled by a Binomial distribution with parameters equal to the number of defects existing and the probability of detecting defects, and, (4) the number of defects remaining in the current phase is simply the sum the number of remaining defects from the current phase and the number of defects remaining from earlier phases. This number is transferred to the next phase.

We note that the BBN model is a simplified representation of the actual process of software development that involves iterations between different development phases. The required activities of different attributes in the same phase also may interact with each other. Since the attributes and the associated activities are used to assess the software's failure probability in a PRA, it is assumed that the assessment using the BBN model is done after the system is installed at the plant, has been fully tested, and is ready to be used during the plant's operation. At this time, the iterations and interactions during development have been completed.

Also, the causal relationships in the BBN model represent probabilistic relationships, i.e., following the requirements and guidance of standards does not always lead to a better development or V&V quality. However, it should increase the likelihood of better quality software. We acknowledge here the possibilities of having high-quality software, even if the required activities were not performed, and, the possibility of having low-quality software even though all of the required activities were performed. Examples for such cases can be found in the literature. The BBN model will show that the chance for these two extreme cases is small. The modeling and quantification of different nodes are summarized in the following subsections.

4.2.1 Attribute Nodes

An attribute node is a node representing the quality in carrying out one or a collection of activities associated with the node (or attribute) (Figure 4-2 and Figure 4-3) where the attribute nodes are modeled as indicator nodes which are related to their parent node by a specified NPT, similar to the indicator nodes of Fenton [2007b]. In this study, attributes and their associated activities were identified from software standards (e.g., IEEE 1012), and guidance (e.g., BTP-14 [NRC 2007]). Chapter 5 provides more details on identifying attributes. Generally, for each of the five software-development phases, the attributes represent (1) activities that are common for all phases and activities that are phase-specific (e.g., requirements specification development activities in the requirements phase), (2) analyses performed (e.g., security analysis), and (3) management functions performed (i.e., configuration management, and review and audit).

Different attributes of a quality node may have different weights in NPTs. The completion of the NPTs is one of the subjects in the round 2 expert elicitation[3]. Figure 4-2 provides the attribute nodes for the quality node representing the overall quality of the development activities in the software design phase. Similarly, Figure 4-3 shows the structure for attribute nodes of the V&V quality node of the Software Design. Appendix B contains the attributes of the complete set of the Development and V&V nodes for all 5 phases.

Each attribute node represents the quality in carrying out the associated/required activities (i.e., the attribute quality). It is modeled as an indicator node with the three states defined below:

High: In addition to satisfactorily carrying out the required activities, additional activities were undertaken that are expected to significantly improve the quality of the work, and enhance the software's reliability.

Medium: All required (or equivalent) activities were satisfactorily carried out.

Low: Some of the required activities were not carried out satisfactorily.

4.2.2 Quality of Development and Quality of V&V Nodes

The Quality of Development and Quality of V&V nodes are modeled as root nodes with three states: High, Medium, and Low. As described previously, they are connected to their respective attribute nodes in a diverging configuration (see Figure 4-2 and Figure 4-3). The states represent the overall quality of the activities. Similar to the three states defined for the attributes, the three states of the Development and V&V quality nodes are defined as follows:

High: State corresponding to the quality of the software development by a high-maturity company rigorously following established standards, and implementing additional measures to significantly improve the quality of the software.

---

[3] In this study, to account for the uncertainty associated with the estimates different experts provided, we did not use NPTs that are tables with constants. Instead, the NPTs are tables of random variables whose probabilistic distributions were estimated via expert elicitation.

Medium: State representing the quality of a software development in which all required activities for safety-related systems are completed.

Low: State representing the quality of a software development in which required activities for safety-related systems are not completed.

4.2.3 Number of Function Points

The number of function points, designated as “Size and Complexity Measure” in Figure 3-3, is a measure of the software’s size and complexity. The function point definition and its counting rules are governed by the International Function Point Users Group (IFPUG) [IFPUG]. The IFPUG function point measure represents the software’s complexity by combining the unadjusted function point count with a multiplier ranging from 0.65 to 1.35. This multiplier, calculated as a function of various predefined influential factors, roughly accounts for the software’s complexity. The number of function points is used in calculating the total number of defects inserted in a phase. It also determines the probability of detecting faults. Unlike the LOC metric, function points measurement is available in any phase of its life cycle and its value remains the same across the SDLC. The number of function points is a root node shared by the 5 phases of SDLC in our BBN model. The prior distribution of the FPs was estimated by a generic-expert elicitation (See Section 7.2). It then was estimated for the specific software program that is modeled using this BBN model.

4.2.4 Size and Complexity

This node has three states High, Medium, and Low that are defined in terms of its parent node, that is, the number of function points. It is assumed in this study that if the number of function points is less than 100, the Size and Complexity is Low; if it is higher than 1000 the Size and Complexity is High. Otherwise, the Size and Complexity is Medium. The node affects defect detection probabilities as shown in Figure 3-3.

4.2.5 Defects Density

The number of defects per function point (defect density per function point in Figure 3-3) is assumed to be determined by the quality of the developmental activities (the Quality of Development node) and the software’s complexity.

4.2.6 Defect Detecting Probability

There are two different defect detection probability nodes representing the probabilities that V&V activities detect and remove faults in the software introduced in the current phase and passed from earlier phase(s). Each is a child node of the V&V quality node and the complexity node (number of function points). Its NPT was estimated by a generic-expert elicitation.

These nodes include the defect removing activities and consider the possibility that new defects introduced during the removing process for the sake of simplicity. In addition, the faults from all precedent phases were assumed to be detected at each SDLC phase at the same defect detection probability. A more sophisticated BBN structure can capture details of this detecting/removing process.

4.2.8 Current Phase Defect Insertion per Function Point

This is the number of faults per function point multiplied by the number of function points.

4.2.9 Number of Faults from Preceding Phase(s)

The number of faults passed from previous phase(s) is the number of faults remaining in the preceding phase. It is linked to the BBN model of the preceding phase. This node is not used in the requirements phase BBN model.

4.2.10 Defects Detected Sourced from Current Phase

This is given by a Binomial distribution with the number of trials equal to the number of defects inserted in the current phase, and a probability equal to the probability of detecting these defects.

4.2.11 Defects Detected Sourced from Previous Phase(s)

This is given by a Binomial distribution with the number of trials equal to the number of defects passed from the preceding phase, and a probability equal to probability of detecting these defects.

4.2.12 Remaining Defects Sourced from Current Phase

This is the difference between the number of defects that are introduced in the current phase and the number of defects of the same kind that are detected and removed within the current phase.

4.2.13 Remaining Defects Sourced from Previous Phase(s)

This is the difference between the number of defects that are passed from the previous phase(s) and the number of defects of the same kind that are detected and removed in current phase.

4.2.14 Defects Remaining

This is the total number of defects remained at the end of each phase. This number includes all defects introduced from current phase, passed from all previous phases, detected and removed at current phase.

5 DEVELOPMENT OF ATTRIBUTES

As described in Section 4.2 , the BBN model for each SDLC phase has two nodes that represent the quality of software development and the quality of V&V. The development team carries out the development activities, while the V&V team undertakes V&V activities[1]. Each of these quality root nodes has child (attribute) nodes indicating the quality in carrying out the required activities associated with these attributes. Examples of these attributes for the software design phase are shown in Figure 4-2 and Figure 4-3.

The major source used to identify the attribute activities is the IEEE V&V standard [IEEE 1012]. The 2004 version of this standard is endorsed by Regulatory Guide 1.168 [RG 1.168-2013]. Many other guidance documents and standards were used, including IEC 60880 [IEC 60880], DO-178C [DO-178C], NUREG/CR-6101 [Lawrence 1993], and BTP-14 [NRC 2007]. In some cases, the Informative section of a standard was used as required activities. The Informative information is not a requirement, and alternative means can be used to accomplish the same objective. Additional standards, including ASME NQA-1[ASME NQA] and DOE G414 [DOE 2010], were also reviewed but provided no additional information.

Each SDLC phase has its featured activities as such different phase has different set of attributes. For example, Table 5-1 lists the 12 attributes of the development quality node of the requirements phase, and as shown in Figure 4-2 there are 14 attributes in design phase for the development quality nod. Table 5-1 also includes definitions of required activities for each attribute. The attribute is considered to be satisfied if all activities defined under it are performed satisfactorily. Appendix B contains a complete list for all attributes and associated activities for each phase.

The following further describes the attributes and offers some guidance on how they are used in assessing the quality in carrying out the activities.

The activities associated with each of the two groups of attributes, that is, the development and V&V groups, can be divided into three subgroups below.

1. Basic activities of the phase

These activities represent the activities that are carried out to accomplish the unique functions of the specific phase. For example, for the requirement-specifications phase shown in Table 5-1, Attributes 1, 2 and 3 are software development planning, concept document development, and software requirements specifications development and attributes 9 and 10 are development of qualification and acceptance test plans. They are considered the basic and unique activities of the phase. Typically, the development team carries out the development activities, while the V&V team undertakes the V&V activities. The description of the activities of the two groups reflects this relationship. Note that the development and V&V teams undertake the test activities separately while the descriptions of these activities are the same.

[1] The V&V team should be technically, financially, and managerially independent from the Development organization, as is required of a safety-related system.

2. Analysis activities

Most of the analysis tasks (e.g., Table 5-1, Attributes 4 to 8 - traceability analysis, criticality analysis, hazard analysis, security analysis, and risk analysis) appear in almost every phase and the phase-specific activities are specified. If both the development and V&V organization independently perform these tasks, then the quality scoring should be higher than when one of them performs and the other merely reviews and verifies. For this reason, the description of the analysis activities for the development and V&V group in Appendix B are kept the same, while the analyses could be done in the same way the basic activities are performed and scored. That is, the development team performs the activities, and the V&V team conducts the V&V. Note that almost all the analysis tasks appear in all five phases, with a few exceptions. For example, traceability analysis is not done in installation and checkout, and criticality analysis is not done in the test phase,

3. Management and QA functions

Table 5-1, Attributes 11 to 12 “Configuration Management”, and “Reviews and Audit” are Management Process and Quality Assurance (QA) functions. The Management Process and QA functions are common to products development (system, hardware, or software) and its associated processes (e.g., the software development phases modeled in this study), except that Reviews and Audits are not needed in the installation and checkout phase. These management-process activities and QA attributes are included as necessary attributes for developing reliable and safe software. Additionally, for these common items, detailed activities that are performed in specific phases are spelled out in the attribute tables, so that they will be considered when a specific software program is evaluated. The descriptions of the activities in Appendix B are the same for the development and V&V teams. They are used in evaluating the quality of the management and QA functions in the same way as other subgroups of attributes are used.

In Appendix B, some example additional activities are added for some attributes. As discussed in Section 4.2.1, when evaluating a software program, a “Medium” score is used if all required activities are satisfactorily carried out[2]: otherwise a “Low” score is given. The additional activities are expected to improve the quality score of the associated attributes from “Medium” to “High”. When additional activities, that are not required, are used as substitutes for required activities, it can raise the score of the associated attributes from “Low” back to “Medium”. The decision on raising the score or not should be made by experts conducting the evaluation for a specific software program based on if the additional activities significantly increase the software’s quality.

[2] Expectedly, a safety-related system of a nuclear power plant would meet the required activities of the “Medium” score of all attributes.

Table 5-1 Attributes and Associated Activities for the Development Quality Node in Requirements Phase

Attribute (1) Software Development Planning

During the software development planning process, the Software Development Plan (SDP), which describes the plan for technical project development, will be developed. The SDP outlines the management, implementation and resource characteristics. The management characteristics include purpose, organization, oversight, and risks. The implementation characteristics include measurement, procedures, and schedule. The resource characteristics include methods/tools and standards.

Required Activities

Software Development Planning tasks are as follows (reference BTP 7-14, NUREG-CR-6101, and IEC 60880)

1. Develop and generate a SDP that is consistent with the lifecycle process defined in IEEE Std 1074 (which is endorsed by Regulatory Guides 1.173).
2. Ensure that the SDP is consistent with other lifecycle plans (e.g., Software Verification and Validation Plan, Software Configuration Management Plan, Software Training Plan etc.)
3. Establish the SDP prior to starting software development activities.
4. Define activities to be performed, and input and output for each lifecycle process.
5. Specify methods, tools, and techniques including programming languages, computers, compilers, library, and links to be used.
6. Make the level of details such that a development team can execute the SDP and carry out software projects.
7. Address technical milestones consistent with the overall project schedule.
8. Provide adequate resources for processing and resolution of all safety issues raised while the development activities are being performed.
9. Resolve all safety issues through appropriate corrective modifications or mitigation dispositions.
10. Address design change including change process and regression analysis/test.

Additional tasks:

1. Plan project management oversight support.
2. Plan proposal evaluation support.
3. Review and ensure that the Work Breakdown Structure (WBS) represents all of the project scope and captures all deliverables, including internal, external, and interim deliverables. Review and ensure that the WBS decomposes the project scope into a set of deliverables that comprehensively defines the work to be performed.
4. Generate software tool plan, which describes the tools needed to support the software development effort. The plan includes a description of each tool's

performance, required inputs and associated tools, outputs generated, needed date, and cost of tool purchase or development.

Attribute (2) Development of a concept[3] documentation (i.e., System Requirements Specifications - SyRS)

The concept activity represents the delineation of a specific implementation solution to solve the user's problem. During the concept activity, the system architecture is selected and system requirements are allocated to hardware, software, and user interface components. In developing the concept document (i.e., SyRS), the development or selection of system architectural design and the development of system requirements are keys. The objective for this development activity is to generate a SyRS that conforms to the user's needs (e.g., stakeholder's requirements), and pertinent regulations/standards, and using best engineering practices. The SyRS will be updated and revised per the outcome of various analyses and V&V activities. A well-developed SyRS plays a solid starting role for the subsequent lifecycle development activities.

Required Activities

The following are the tasks performed in the development of a SyRS.

1. Develop a specific conceptual solution (e.g., system architecture as described in the concept documentation) based on user needs and acquisition needs.
2. Identify requirements from the customer, the environment, and the experience of the technical community.
3. Identify constraints of interfacing systems and constraints or limitations of proposed implementation solution.
4. Allocate functional and performance requirements (e.g., timing, response time, and throughput) to the hardware, software, and user interfaces.

[3] The detailed discussion of concept activities is only provided in IEEE 1074 [IEEE 1074] among the IEEE standards we have evaluated. In IEEE 1074, a total number of 69 activities in the SPLCP (Software Project Life Cycle Process) are grouped into 17 activity groups, and the Concept Exploration [Section A.2.1 of IEEE 1074] is one of three groups in the pre-development section of the software. The purpose of concept exploration is identification of ideas or needs, selection of potential approaches and performance of feasibility studies to refine and finalize the idea or need. The output of concept exploration will be a Statement of Need that identifies the software idea, need, or desire, the recommended implementation approach, and any data pertinent to a management decision concerning the initiation of the described development effort. The Statement of Need is the basis for the system analysis and the development of software requirements via the System Allocation activity [Section A.2.2 of IEEE 1074]. As a bridge between the concept exploration and the development of software requirements specifications, system allocation activity maps the required functions identified in the concept exploration to software, and when applicable, to hardware and people. The major activities of system allocation include (1) derivation of system functions from system requirements and identification of hardware, software, and operational requirements; (2) development of system architecture; and (3) formulation of software requirements, system interface requirements, and human and hardware requirements (if applicable), i.e., allocation of system requirements to software, interface, human and hardware (if applicable). Additional guidance for software requirements development can be found in IEEE 830 [IEEE 830].

5. For the internal and external interfaces, specify the data formats, interface protocols, frequency of data exchange at each interface, and other key performance requirements.
6. Develop application-specific requirements such as redundancy, independence (physical, electrical, and communicational independence), diversity and defense-in-depth, fail-safe, fault detection, fault isolation, and diagnostic and error recovery (e.g., Oconee SER and IEC 61508).
7. Specify maintenance requirements for the system.
8. Specify migration requirements from an existing system where applicable.
9. Build well-formed requirements.[4]
10. Organize requirements into the concept documentation (i.e., SyRS).
11. Update/revise the concept documentation (i.e., SyRS) per the various analyses activities results [i.e., traceability analysis, criticality analysis, hazard analysis, security analysis, and risk analysis, see Attributes (4), (5), (6), (7), and (8)].
12. Evaluate user documentation[5] for its completeness, correctness, and consistency with respect to requirements for user interface and for any functionality that can be invoked by the user.

Attribute (3) Development of Software Requirements Specifications (SRS)

The purpose of developing the requirements (e.g., functionality, capability, interface, qualification, safety, security, human factors, data definitions, user documentation, installation and acceptance, user operation, and user maintenance[6]) of the SRS is to satisfy system requirements and user's needs and ensure correctness, consistency, completeness, accuracy, readability, testability, and robustness in the SRS.

Required Activities

The following tasks are performed in developing SRS.

1. Develop the software requirements according to the system requirements allocated

[4] IEEE 1233 [IEEE 1233] provides guidance for developing System Requirements Specification (SyRS), which states that SyRS development is an iterative process that includes identification, construction, organization, and presentation of requirements sub-processes. Per IEEE 1233, a well-formed requirement is a statement of system functionality (a capability) that can be validated, and that must be met or possessed by a system to solve a customer problem or to achieve a customer objective, and is qualified by measurable conditions and bounded by constraints.

[5] User documentation refers to the documentation for a product or service provided to the end users. The user documentation is designed to assist end user to use the product or service. The user documentation is a part of the overall product delivered to the customer, which may include user guides, instruction manual or training materials.

[6] IEC 61508 [IEC 61508] Part 3 Section 7.2 contains detailed requirements for these activities. In general, the detailed requirements should be evaluated against the "attributes".

to software taking into consideration the assumptions, constraints, and operating environment for the system[7] (Correctness)

2. Develop the software requirements according to standards, references, regulations, policies, physical laws, and business rules[8].(Correctness)
3. Develop the sequences of states and state changes (according to logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis.) (Correctness)
4. Develop the flow of data and control to satisfy functionality and performance requirements. (Correctness)
5. Use proper data and format. (Correctness)
6. Document terms and concepts consistently. (Consistency)
7. Develop the SRS such that the function interactions and assumptions are consistent. (Consistency)
8. Develop the external and internal software interface requirements. (Correctness)
9. Develop each interface requirement with the required accuracy. (Accuracy)
10. Maintain internal consistency between the software requirements and external consistency with the system requirements. (Consistency)
11. Include the following elements in the SRS, within the assumptions and constraints of the system:

    i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging).

    ii) Hardware, software, and user-interface descriptions.

    iii) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).

    iv) Critical configuration data.

    v) System, device, and software control (e.g., initialization, transaction and state monitoring, and self-testing). (Completeness)
12. Specify logic, computational, and interface precision (e.g., truncation and rounding) satisfying the requirements in the system environment. (Accuracy)
13. Model physical phenomena to conform with system accuracy requirements and physical laws. (Accuracy)

[7] Section 6.3.1 in DO-178B: Compliance with system requirements.

[8] Section 6.3.1 in DO-178B: Conformance to standards.

14. Develop algorithms with adequate accuracy especially in the area of discontinuities (e.g., different plant operating modes) (DO-178C). (Accuracy)
15. Develop legible, understandable, and unambiguous (i.e., having one and only one interpretation) software requirements to the intended audience. (Readability)
16. Define all acronyms, mnemonics, abbreviations, terms, and symbols used in software requirements. (Readability)
17. Specify objective acceptance criteria for validating the requirements of the SRS by testing. (Testability)[9]
18. Evaluate and eliminate any potential conflicts between the requirements[10] and the hardware/software features of target computer, especially, system response times and input/output hardware. (Compatibility with the target computer [DO-178C)]
19. Specify the behavior of the software in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior. Of particular concern is the behavior of the software in the presence of unexpectedly high or low rates of message traffic. (Robustness)
20. Generate the SRS using best engineering practice and guidance as outlined in pertinent standards (i.e., IEEE Std 830).
21. Update/revise the SRS per the outcome of the various analyses results [i.e., traceability analysis, criticality analysis, hazard analysis, security analysis, risk analysis, and various additional analysis activities, see Attributes (4), (5), (6), (7), (8), and (13)].

Additional tasks:

(1) Address the software tool qualification to assure that the tool is functioning properly and it does not mask errors that it was designed to find.

[9] IEC 60880 states that the SRS shall be unequivocal, testable or verifiable, and achievable.

[10] Original wording in DO-178B is high level requirements, that is, produced directly through system requirements and system architecture.

Attribute (4) Traceability Analysis-Requirements Phase

The requirements traceability analysis is part of the development processes that ensure system and software requirements are complete, testable, and implemented correctly. Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specifications, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)[11].

The traceability analysis of software requirements is basically to trace the software requirements as stated in the SRS to the system requirements in the SyRS backward, and the user's needs and acquisition needs to the software requirements stated in the SRS forward.

Required Activities

The development tasks for the traceability analysis are as follows:

1. Establish a traceability matrix that is consistent with the development process.
2. Identify all system and software requirements.
3. Trace the system requirements (as defined in the SyRS) to acquisition needs (backward traceability) and the acquisition needs to the system requirements (forward traceability).
4. Trace the software requirements (SRS) to the corresponding system requirements (e.g., Concept Documentation SyRS) (backward traceability) and the system requirements to the corresponding software requirements (forward traceability).
5. Analyze identified relationships for correctness, consistency, completeness, and accuracy. The task criteria are as follows:

   i) The relationships between each software requirement and its system requirement should be correct. (correctness of the relationship/mapping) (Correctness)

   ii) The relationships between the software and system requirements should be specified to a consistent level of detail. (Consistency)

   iii) Every software requirement should have sufficient detail to show conformance to the system requirements. (Completeness)

   iv) All system requirements related to software should be traceable to software requirements. (Completeness)

   v) The system performance and operating characteristics should be accurately specified by the traced software requirements. (Accuracy)

[11] This definition is summarized by Gotel et al [Gotel 1994] in "An Analysis of the Requirements Traceability Problem", Proceedings of First International Conference on Requirements Engineering, 1994, which is derived from IEEE Std 830 definition "A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation".

6. Document the traceability analysis results and generate the traceability analysis report.

Attribute (5) Criticality Analysis-Requirements Phase

Criticality analysis is used to assign integrity/criticality level, which in turn is used to determine the rigor and effort of the development activities at each stage of the development lifecycle. The analysis determines how a system, system element, or component can potentially cause undesirable consequences. In this phase of development, an initial criticality analysis is performed to assign integrity level to system and software components. The criticality analysis report can be updated as the development process progresses and more detailed information becomes available.

Required Activities

For system requirements:

1. Determine whether integrity levels are established for requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions.
2. Verify that the assigned integrity levels are correct. If integrity levels are not assigned, then assign integrity levels to the system requirements.
3. Document the integrity level assigned to individual components (e.g., requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions). For software development planning purposes, the system should be assigned the same integrity level as the highest level assigned to any individual element.
4. Verify whether any component can influence the individual components assigned a higher software integrity level, and if such conditions exist, then assign that component the same higher integrity level.

For software requirements:

1. Review and update the existing criticality analysis results from the prior Criticality Task Report using the SRS.
2. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, and other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

Attribute (6) Hazard Analysis-Requirements Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012).

The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. Once the end hazardous conditions are identified, the hazard analysis typically uses proven analysis approaches/tools. In this phase of development, an initial hazard analysis is performed to identify system and software hazards. The hazard analysis is repeated in each life cycle phase and accounts for further elaboration of designs, changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012).

Required Activities

For system hazard analysis:

Analyze the potential hazards to and from the conceptual system (e.g., as documented in the SyRS). The analysis includes the following tasks:

1. Identify the potential system hazards
2. Assess the consequences of each hazard
3. Assess the probability of each hazard
4. Identify mitigation strategies for each hazard
5. Document the software hazard analysis results and generate report

For software hazard analysis:

Analyze software hazards (i.e., software conditions, including software faults and incorrect software requirements, which can lead to an accident[12]). The analysis includes the following[13]:

1. Identify potential system hazards contributed by SRS.
2. Assess the consequences of each system hazard taking into consideration the identified software hazards.
3. Assess the probability of each system hazard taking into consideration the identified software hazards.
4. Identify mitigation strategies for each hazard.
5. Document the software hazard analysis results and generate report.

All requirements for fault tolerance and failure modes should be fully specified for each operating mode. Software requirements for handling both hardware and software

[12] ISO/IEC/IEEE 24765:2010 [IEEE 24765] - Systems and software engineering vocabulary.

[13] The activity descriptions are clarified by explicitly stating that the hazard analysis is performed for system hazards contributions from software. The clarification is necessary to avoid the confusion of software hazard analysis with system hazard analysis. This change is based on the discussion in Annex J of IEEE Std. 1012.

failures should be provided, including requirements for analysis of and recovery from computer system failures. Requirements for on-line in-service testing and diagnostics should be provided.[14]

Note 1: Section 3 of NUREG/CR-6430, the requirements hazard analysis content provides guiding phrases for examining requirements. Page 25 states "A major impact of the results from the software hazards analysis is on changes to the software requirements specifications for the purpose of eliminating identified hazards that are affected by the software or that are not adequately managed by the software."

Note 2: Annex J of IEEE Std. 1012 states: "The hazard analysis may be performed by an organization within the project such as systems engineering, reliability, safety, or V&V. In any case, V&V reviews the hazard analysis for completeness and usability, and it assures that the stated hazards and contributors are clearly identified to sufficient detail to affect engineering and mitigation activities properly and to develop V&V plans and evaluation criteria."

Note 3: IEEE 7-4.3.2 [IEEE 7-4.3.3] states that the software requirements hazards analysis includes evaluation of software and interface requirements for deficiencies that can contribute to hazards. Examples of software hazards include logic errors, incorrect loop iterations, and using wrong variables.

Note 4: IEEE Std. 7-4.3.2 contains more detailed discussion of software hazard analysis. However, those discussions can, at a high level, be summarized by the four main required activities described in the checklist (hazard identification, assess consequences, assess probability, identify mitigation strategies). No text from that standard is used in the checklist.

Note 5: NUREG/CR-6430 describes one method in which hazard analysis can be performed. It is felt that the activities described in that report can also generally be categorized by the four activities in the checklist. Therefore, no text from that report is used in the checklist.

Note 6: MIL-STD-882E [MIL-STD-882E] contains discussions on hazard analysis. The overall approach and goal are similar to the description in the checklist. No specific text from that standard is used in the checklist.

Note 7: DO-178C specifies that high level system requirements that are allocated to the software should avoid introducing hazards. This requirement is covered by activity A. in the checklist.

[14] BTP 7-14 is consulted for its requirements related to safety and hazards. The discussion of fault tolerance and failure modes are found to be relevant and so are included as part of the hazard analysis activity.

Attribute (7) Security Analysis- Requirements Phase

The objective of security analysis is to ensure that system and software security vulnerabilities are identified, and required threat controls and safeguards of system and software from accidental or malicious access, use, modification, destruction, or disclosure are addressed in the system and software requirements. In this phase of development, a security analysis of the system and software requirements is performed.

Required Activities

For system requirements:

1. Review the system owner's definition of an acceptable level of security risk.
2. Analyze the system concept (e.g., as documented in the SyRS) from a security perspective and assure that potential security risks with respect to confidentiality (disclosure of sensitive information/data), integrity (modification of information/data), availability (withholding of information or services), and accountability (attributing actions to an individual/process) have been identified. Include an assessment of the sensitivity of the information/data to be processed.
3. Analyze the security risks introduced by the system itself as well as those associated with the environment with which the system interfaces.
4. Verify that the system security requirements will mitigate the identified security risks introduced by the system concept.
5. Document the system security analysis results and generate report.

For software requirements:

1. Determine that the security requirements identified in the SRS address the security risks introduced by the system concept.
2. Verify that the software security requirements will mitigate the identified security risks to an acceptable level.
3. Document the software security analysis results and generate report.

Attribute (8) Risk Analysis- Requirements Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the system or the successful development of the system. In this phase, a risk analysis of the software requirements is performed. The risk analysis is performed continuously throughout the development life cycle.

Required Activities

The risk analysis procedure is as follows:

1. In terms of requirements in the SyRS, and SRS, identify risk (technical and

managerial risks[15]) contributors, i.e., the initiating events, hazards, threats, or situations that create risks;

2. Estimate the probability of occurrence, the consequences for each risk, and the expected timing of the risk; and
3. Evaluate each risk or defined combination of risks against its applicable threshold, generation of alternatives to treat risks above their risk thresholds, and making recommendations for treatment (elimination, reduction, or mitigation of risks) based on a priority order.
4. Document the risk analysis results and generate report.

Attribute (9) Software Qualification Test Plan Development

Software qualification testing is performed on a complete, integrated system (or a system component such as software) to evaluate the system's compliance with its specified requirements. Qualification test plans can be generated once the system/software requirements are available.

Required Activities

Software Qualification Test Plan Development task is as follows::

1. Plan software qualification testing to validate software requirements.
2. Plan tracing of system requirements to software qualification test designs, cases, procedures, and results.
3. Plan documentation of software qualification test designs, cases, procedures, and results.
4. The software qualification test plan addresses the following:
    i) Conformance to all system requirements (e.g., functional, performance, security, operation, and maintenance) as complete software end items in the system environment.
    ii) Adequacy of user documentation (e.g., training materials and procedural changes).
    iii) Performance at boundaries (e.g., data and interfaces) and under stress conditions.
5. Verify that the software qualification test plan satisfies the following criteria:
    i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).

[15] The example technical and management risks may include, e.g., the possibility of negative consequences to the operation of the system and the delivery of the project, as indicated in Annex J of IEEE 1012.

ii) Test coverage of system requirements.

6. Validate that the software qualification test plan satisfies the following criteria:
   i) Appropriateness of test methods and standards used.
   ii) Conformance to expected results.
   iii) Feasibility of system qualification testing.
   iv) Feasibility and testability of operation and maintenance requirements.
7. Generate software qualification plan.

Attribute (10) Software Acceptance Test Plan Development

Software acceptance testing is conducted to determine whether or not a system (or a system component such as software) satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. Acceptance test plans can be developed once the system/software requirements are available.

Required Activities

Software Acceptance Test Plan Generation task is as follows:

1. Plan software acceptance testing to validate that the software correctly implements system and software requirements in an operational environment.
2. Plan tracing of test requirements to test software acceptance design, cases, procedures, and execution results.
3. Plan documentation of test tasks and results.
4. The software acceptance test plan addresses the following:
   i) Conformance to acceptance requirements in the operational environment.
   ii) Adequacy of user documentation.
5. Verify that the software acceptance test plan satisfies the following criteria:
   i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008 [B2]).
   ii) Test coverage of acceptance requirements.
6. Validate that the software acceptance test plan satisfies the following criteria:
   i) Conformance to expected results.
   ii) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).
7. Employ the user documentation in planning an acceptance test that is representative of the operational environment.
8. Develop software acceptance plan.

Attribute (11) Configuration Management- Requirements Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system, software and hardware product functionality, tracking program versions, generating baselines (including parameters and settings), and managing changes. The configuration management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs.

In the Requirements phase CM, the development organization focuses on defining a configuration management strategy and ensures that the phase specific configuration items (i.e., SyRS, SRS etc.) are under controlled.

Required Activities

1. Define a configuration management strategy so that configuration controls are in place to maintain and document configuration item baselines with unique identifiers. The strategy should include notification to the V&V effort for all changes made to the configuration item baselines.
2. Define and document items requiring configuration management. The items controlled should include enabling systems, tools and processes that are integral to system development and life cycle support that will be subject to V&V in order to demonstrate conformance to this Standard.
3. The status of items under configuration management is made available throughout the life cycle. Provisions should be included to assure that the V&V effort receives the current status for all configuration items required in the Concept and Requirements phase including but not limited to SyRS, SRS, and system architectural drawings.

Attribute (12) Reviews and Audit-Requirements Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028). In this phase of software development, the tasks include establishing a Review and Audit process and plan, applying it to SyRS and SRS, and tracking any anomalies.

Required Activities

The reviews and audit tasks are as follows:

1. Generate a reviews/audit plan in which review team and each team member's responsibilities, and review schedule are defined.
2. Define reviews/audit process including protocols for interfacing with other organizations (i.e., Project, V&V, and QA).

3. Document reviews/audit results (e.g., on the SyRS and SRS).
4. Establish an anomaly tracking system for tracking anomalies during the reviews/audit process.
5. Document improvement and/or lessons learned as a result of reviews/audit.

6 FROM NUMBER OF DEFECTS TO RELIABILITY

This chapter describes two methods, the fault exposure ratio (FER) method [Musa,1987] and the FSD method [Littlewood 1980 and Delic 1995 and 1997] that have been used in the literature on software reliability to convert the number of defects in a software program into reliability measures. In this study, the research team decided to use the FSD method to estimate the probability of the software failure on demand.

Section 6.1 describes the FER method. FER limitations that are common with FSD method are also discussed. Section 6.2 describes the FSD method. Section 6.3 describes how the FSD method was applied in this study.

6.1 Fault Exposure Ratio

The FER method was introduced by Musa [1987] in which he describes FER as "…the fraction of time that the 'passage' results in a failure". The FER was estimated using data from some systems available at the time (the data were collected before 1980). In Musa [1987], the failure rate of the software is calculated as the FER multiplied by the number of faults in the software. A *linear* execution frequency is defined as "…the frequency of execution assuming that there are no branches and loops" (or equivalently, the execution time of the software if each statement executes only once). The assumption that the failure rate is proportional to the number of faults is common in many of the software reliability growth-models (SRGMs) [IEEE 1633]. The FER is frequently used in calculating the software failure rate when the number of faults in the software is estimated, for example in [Neufelder 2002 and RAC 1998]. The FER is essentially the probability of failure per fault per (*linear*) execution. Considering an execution of the software as a demand on the system, the FER effectively is the probability of failure per linear execution per fault.

The FER value presented in [Musa 1987] was an average of some software projects in 1970s and most of them were developed using low level languages such as Assembly. The validity of the use of that number to modern software is questionable. In addition, this average value might bring big uncertainty when it is applied to individual software.

Since FER is used to calculate the software failure rate (or failure probability) over time, this study did not select it to calculate the per demand probability.

6.2 Fault Size Distribution

A digital protection system at an NPP is continuously running monitoring the condition of the plant in order to generate a safety action demand when needed. From a PRA perspective, many times the interests are the failure probabilities for digital protection systems under demand conditions. Although the software of the system executes in cycles, it would not be appropriate to consider that each cycle of execution as a demand as is assumed in the FER method[1]. Therefore, a failure-

[1] FER-based methods, such as some SRGMs, treat time as a continuous variable. In general, it may be possible to develop discretized versions of these methods by treating demands on the system as time steps, such that they can be directly applied to assessing the probability of software failure on demand [Chu 2010].

on-demand-based equivalence to the FER should be used. The concept of the size of faults was first introduced by Littlewood [1980]. Delic [1997] further developed the FSD.

When describing a conceptual model of software failure in his SRGM, Littlewood [1980] introduced the concept of the "size" of faults representing the variability in the likelihood (or frequency) that the faults are triggered. That is, different faults may be triggered by different triggering events with different frequencies. He indicated that the variability represents "real" randomness, that is, aleatory uncertainty. This randomness was demonstrated by Dunham [1990] using data collected from controlled experiments. A study at the Research Triangle Institute [Dunham 1988] gathered error data from the software modules of a hypothetical radar- tracking problem, and obtained error rates of faults differing by orders-of-magnitude.

Delic [1995] first introduced the term "fault size distribution" (FSD) associated with the software's failure on demand, and used it in a BBN to demonstrate how the probability of system failure on demand can be estimated. The use of a distribution to represent the variability makes the FSD somewhat different from FER which often is represented by a single number. The FSD is the distribution of the probability of failure on demand per fault. Therefore,

*Software failure probability = Number of faults in the software * FSD* (6-1)

Here, all three terms in the equation are random variables.

For the FSD, Delic suggested estimating it from the population of software products sharing the same developmental process (i.e., similar software products developed by the same vendor). In 1997, Delic [1997] used an example belief network with numerical results obtained by modeling different assumptions related to the initial number of faults, the number of faults detected, and FSD.

A basic assumption of using this distribution in estimating the probability of failure of the software on demand is that this probability is proportional to the number of faults in the software.

Another important assumption is that the software developed by the same team following same process for the same function (or use) has the same FSD [Delic 1997]. Thus, the distribution estimated based on the data from, for instance, same RPS software of different versions, deployed in different plants of the same design, can be used for a specific software of the same kind developed by the same team with the same process.

6.3 FSD Quantification

As discussed in the Section 6.2, the number of software defects can be converted to a reliability value if the FSD parameter can be estimated. Therefore, a FSD that represents the collection of all safety-related software was estimated using the operating experience of the safety-related software at nuclear power plants throughout the world. A representative (or average) number of defects remaining in this collection of safety-related software was estimated using the BBN model applied to a "base case" characterized by function point value of 50, and all indicator values of "Medium". This "base case" is assumed to represent this collection of deployed safety-related software which were reviewed and approved.

6.3.1 Operating Experience of Safety-Related Protection-System Software

The literature was surveyed to identify possible sources of operating experience that potentially can be used in estimating a FSD for safety-related *protection*-system software. A few sources were identified and used. They are summarized below, along with details on how the information they contain were used in this study. Table 6-1 summarizes the information collected in the review of operating experience of safety-related software. A total of 4957 demands with 1 failure during a test were estimated. The table shows the mean software failure probability of each data source calculated using a uniform prior distribution.

6.3.1.1 US Experience

Turkey Point Unit 3 [1994] experienced a failure of its diesel generator-load sequencer during testing in 1994. This is the only observed safety-related protection software failure in the operating experience of the US plants.

Oak Ridge National Laboratory (ORNL) performed a study [Poor 2012] for the US NRC to build a database of digital instrumentation and control (I&C) systems operating at US NPPs. It identified Core Protection Calculators (CPCs), and Eagle-21 modules which have been used in the reactor protection systems of some plants. Recently, a TXS platform was used at the Oconee plant replace its original RPS. In addition, digital diesel generator load sequencers have been used at a few plants. The ATWS mitigation system actuation Circuitry (AMSAC) at some plants also is digital. The database does not contain information on how long the digital systems have been operating at the plants, nor the number of demands that have been observed. Since the number of digital AMSACs and diesel generator load sequencers was low, they are not further considered in this study.

Bickel [2008] discusses the operating experience of CPCs that are used at many Combustion Engineering (CE) plants. One event in a Licensee Event Report (LER) involved a latent error in software design related to processing failed sensors inputs. He identified 99 reactor trips without failure.

In NUREG/CR-5500, Volume 2, Eide [1999] developed a fault-tree model for Westinghouse reactor's reactor-protection systems of which some use Eagle 21 modules. He estimated that these systems at the plants have accumulated 543.7 reactor years of operating experience, out of which 111 reactor years were from the systems that used Eagle 21 modules. During the period, a total of 1845 unplanned reactor trips were observed. Assuming the trips are evenly distributed among the Westinghouse plants, there would be 377 unplanned trips for those reactors that use Eagle-21.

To account for the operating experience since the above two studies, BNL performed additional LER searches available at the NRC website. An additional 14 and 82 automatic reactor trips with no failure were identified for CPC and Eagle 21, respectively.

Table 6-1 Operating Experience of Safety-related Protection System Software

Type of System(s)	Sources	Number of Demands (*D*)	Number of Failures (*F*)	Mean Software Failure Probability[2]	Notes
Diesel generator load Sequencer	[Turkey Point 1994] [Poore 2012]	Unknown	1	NA	At Turkey Point during testing
RPS (Core protection calculators)	[Bickel, 2008]	99+14=113	0	~0.01	
RPS (Eagle 21)	[Poore 2012] [Eide 1999]	~377+82=459	0	~0.002	Additional data from LER search
RPS (SPIN 1300)	[EPRI, 2010]	~500	0	~0.002	French 1300 MW NPP SPIN 1300 experience
RPS- South Korea	[KINS 2015]	12+13 =25	0	~0.04	Korea Hydro & Nuclear Power experience
RPS (TXS)	[Bäckström 2015]	3360	0	~0.0003	
US experience		1072	1	~0.002	*D* was conservatively estimated
Total		4957	1	~0.0004	

6.3.1.2 International Experience

France – EPRI [2010] estimated that approximately 500 reactor trips had occurred at the French 1300 Mw plants that use the SPIN 1300 reactor protection system (a Rolls-Royce platform) which is a fully digital system.

TXS platform – Jockenhovel-Barttfeld [2015] estimated that the TXS platform has been used as reactor-protection systems throughout the world with a total of 3400 demands without failure.

South Korea – During 37 reactor years of operating experience at five Korean plants of Korea Hydro and Nuclear Power (KHNP), there were 12 actual reactor trips and 13 trips during tests with no failures observed for the digital protection systems [KINS 2015].

The IAEA – IAEA compiles a database called International Database on Digital I&C Products, Platforms and Projects in Nuclear Power Plants (IDIP) [Fisseha, 2008]. It considers both safety-

[2] E(Beta (1+F, 1+D))=(1+F)/(2+f+D)

related and non-safety-related systems. It recorded a total of 268 hits with each hit having one or more records. Many of the records contain information about when the systems were put into service with internet links to the source documents. However, the specific functions (i.e., control or protection) of the systems often are not clear, and the internet links mostly are broken or unavailable. The database does not contain information on how many demands on the systems have been observed.

6.3.2 Software Failure Probability Quantification

The operating experience of the digital protection system summarized in Section 6.3.1 is assumed to be representative of digital systems developed under the SDLC described by the BBN model in this study. The earlier systems were developed under different process from modern quality assurance. Their operating experiences are included in this analysis in order to be conservative and inclusive.

The operating experience of the digital protection system summarized in Section 0 is assumed to be representative of digital systems developed under the SDLC described by the BBN model in this study. The earlier systems were developed under different process from modern quality assurance. Their operating experiences are included in this analysis in order to be conservative and inclusive.

The operating experience from was analyzed using Bayesian update assuming uniform prior distribution. shows the estimated mean software failure probability for the above safety-related protection systems using a uniform prior distribution. Due to the limited operating experience, the estimated failure probability is not small. The lowest is ~0.0003 for the TXS platform. The highest is ~0.04 for the Korean experience.

The US operating experience of one failure (Turkey Point) in 1072 demands yields a mean failure probability of ~ 0.002.

The operating experience from was used in a hierarchical Bayesian analysis [Atwood 2002] to estimate a distribution for the probability of software failure that captures the variability among the population of safety-related software. In the analysis, it was assumed that the population variability distribution was lognormally distributed, and the operating experience of Core Protection Calculators [Bickel 2008], Eagle 21 [Poore 2012 and Eide 1999], SPIN 1300 [EPRI 2010], South Korea [KINS 2015], and TELEPERM [Jockenhövel 2015] were used. Since no data were available on the number of demands on digital diesel generator sequencers, the Turkey Point failure event was not used in the analysis. The analysis resulted in a lognormal distribution with μ equal to -10.45, and sigma equal to 2.168. The characteristics of the distribution are shown in. The large uncertainty in the distribution reflects the variability among the safety-related software.

Table 6-2 Generic Distribution[3] of Software Failure Probability

μ	σ	Mean	Stand Deviation	5th Percentile	Median	95th Percentile
-10.45	2.217	5.53×10^{-4}	6.31×10^{-3}	5.71×10^{-7}	2.9×10^{-5}	1.45×10^{-3}

6.3.3 FSD Quantification

In order to quantify FSD based on the digital protection software failure probability per demand presented in, an average software size in FP is required but not available in the above operating experience study (Section 6.3.1).

A typical digital protection software contains an input module that reads sensor measurements, an internal processing logic (such as comparison logic) to trigger the actuation unit when the sensor measurements exceed their setpoints, and an output unit (actuation unit) to produce trip signal. By following the FP counting rules, one low level external input, one internal logic file, and one external output are counted. A rough estimate of 50 FPs is considered typical size of the digital protection software in this study.

In addition, attributes states for a typical digital protection software are reasonably considered as "Medium" for the BBN model as these software needs to pass the regulatory licensing review, so all required activities described in Appendix B should have been completed satisfactorily.

Given the size of 50 FPs, and "Medium" for all attributes, the number of defects remaining was calculated and given in Table 9-12. The FSD is then calculated using Equation (6-1) and the result is provided in Table 6-3.

Table 6-3 Typical FSD Distribution

Mean	σ	5th Percentile	Median	95th Percentile
1.02×10^{-4}	1.34×10^{-3}	1.24×10^{-7}	6.12×10^{-6}	3.03×10^{-4}

[3] A lognormal distribution was fitted. σ is the scale parameter and the standard deviation of the after-logged normal distribution.

7 EXPERT ELICITATIONS

Three expert elicitations were performed as a part of the construction and quantification of the BBN model. They include elicitations on (1) the structure of the BBN model, (2) the parameters of the model, and, (3) specific evidence of the two example software systems. In the first expert elicitation, thirteen experts on software development and V&V were used. Among the 13 experts, BBN experts were consulted on more theoretical questions. In the second elicitation, experts from seven different organizations with hands-on experience in development and V&V of nuclear plant systems were used. After the first two elicitations, a BBN model for safety-related software was developed. It allows generic information to be obtained for safety-related software. In the third elicitation, the BBN model was applied to two safety-related software programs to estimate the number of remaining faults and failure-probability. One expert worked on LOCS and the other one worked on IDiPS-RPS.

Figure 7-1 depicts the relationship of the three expert elicitations to the BBN model. It shows the higher-level structure of our BBN model of the design phase. The first elicitation establishes the model's structure, the second elicitation estimates the parameters for the black circle, the grey circle and the grey rectangle nodes, and the last elicitation provides specific evidence for the grey circle and grey rectangle nodes.

- Black circle nodes: The NPTs of Defect Density and defect detection probabilities were estimated in the second elicitation of the experts.
- Grey circle nodes: These nodes are root nodes of the model. The size and complexity node has a constant value for the specific software (SW). The two Quality nodes initially were estimated in the second elicitation, and later updated using the evidence on attribute nodes obtained in the third elicitation.
- White circle node: Only simple calculation is necessary for these nodes.
- White rectangle node: This node connects the model of the design phase to that of the previous requirements-specification phase.
- Grey rectangle nodes: The NPTs of these attribute nodes were estimated by experts in the second elicitation. Literature data and LOCS/IDiPS-RPS development data were used to update the model.

The elicitations were done by identifying experts, distributing the background material and questionnaires to the experts, collecting and analyzing the answers provided by the experts, asking clarifying questions and collecting revised answers from them, resolving the comments they provided, and updating the BBN model. The lists of experts, who participated in the three elicitations, are given in Appendix C. The material used in the elicitations including the packages sent to the experts and the answers the experts provided are included in [NRC 2016]. The following subsections provide a summary of each of the elicitations including the answers and comments given by the experts, how the comments were resolved, and how the answers were used in revising the BBN model and estimating the parameters. Chapter 8 documents the use of the elicitation results in quantifying both our generic- and specific-BBN models.

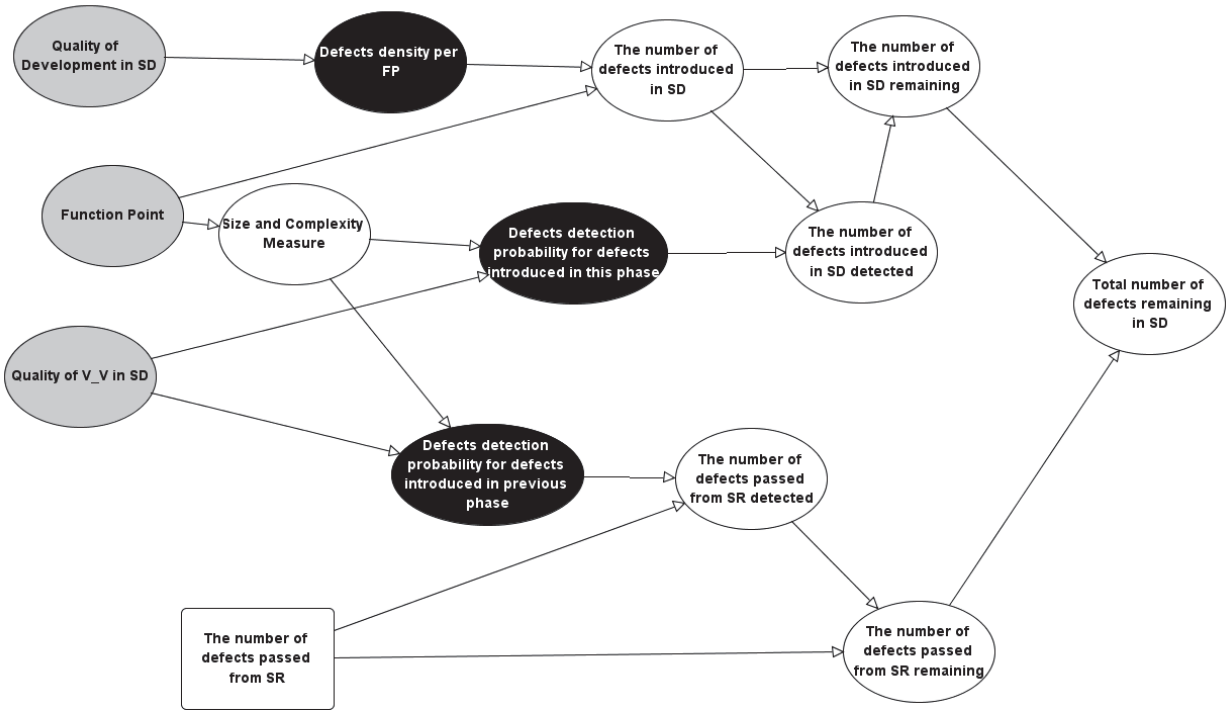

Figure 7-1 Model of the Design Phase

7.1 Elicitation Phase One - BBN Structure

The objectives of this elicitation are to verify (1) the causal relationship of the nodes in the BBN model, (2) that the BBN structure meets the associated dependence- and conditional independence-requirements, and, (3) the adequacy of the model in capturing important attributes of the software's development lifecycle. In the expert elicitation, thirteen experts on software development and V&V were used. In addition, two BBN experts were consulted on more theoretical questions of BBN modeling. Appendix C.1 presents a list of the experts who participated in this round.

The research team started the elicitation by providing the experts with a description of the preliminary BBN model that represents the causal relationships between attributes and the number of residual defects. In a questionnaire, experts were asked to review the BBN model, comments on the causal relationships therein and the use of the attributes, etc. A separate set of theoretical questions were sent to the two BBN experts. [NRC 2016a] contains the package that was sent to the experts and their responses. The following is a summary of expert's responses. The model described in Chapter 4 is the revised version that reflects expert's comments.

7.1.1 Structure Questions

The experts generally agreed that the quality of the software development process can be used to predict the probability of failure-on-demand of the software. However, some experts were concerned that the sequential nature of our model will not adequately capture the iterations present in modern software development process. This BBN model is supposed to be applied to evaluate deployed software which is after the checkout and installation phase. This means that all the iterations present in the development have been completed, and the model evaluates the final abstracted 5-phase process, in which an individual phase might be a combination of multiple iterations. The research team

assumed the intermediate products after each iteration are not required to be evaluated and modeled for this BBN. This assumption does not block the future practices to model the iterated process chronically. This inevitably will increase the size and complexity of the model and amount of parameters. A trade-off might need between the feasibility and precision.

Another concern related to the model is that not all attributes present in the model will be applicable to all development-process models. Likewise, there may be attributes important to particular development models that are not captured in the model. The research team agreed with this comment that the model developed in this study is only for demonstration purpose only. The research team was not able to capture all development details due to all kinds of limitations to access proprietary design information. However, by following the framework this study developed, future practitioners should be able to develop vendor specific BBN models that are customized to specific design conditions. The expert opinion or literature data might not be needed for such models as "real data" might become available.

There were suggestions that the defect density and detection probability depend on the size and complexity of the software. The research team therefore added a link between the Size and Complexity node and the two detection probability nodes. The effect on the defect density node is considered captured by the use of number of function points in calculating the number of defects inserted.

Some experts pointed out that some attributes in one phase may depend on one another. For example, the quality of criticality analysis may affect the quality of other attributes. The research team agreed that weak dependencies might exist, but could be ignored in this model for considerations of simplicity and for the purpose of demonstration.

Some experts commented that the development quality could affect V&V quality in that it is hard to do a good V&V if the product from the development has poor quality. Similarly, the V&V quality can affect the developmental quality. This study assumes the V&V activities are conducted independently from the development activities. The effect of V&V activities in correcting development defects is reflected by the detection and removal of faults explicitly modeled in the BBN.

Experts also pointed out that since the V&V activities for the current phase are designed to catch defects inserted in the current phase, they are less likely to catch defects from other phases. Hence, a separate detection-probability node to the model to account for the different detection probabilities for defects from the current phase, and those from previous phases was added.

There also were suggestions that personnel quality should be included as indicators of development and V&V qualities. The research team believed the personnel quality is the root cause was not at the same level of detail as indicators, and indicators adequately reflect the quality of the personnel.

On the question of using function point or LOC to represent size and complexity, some experts agreed that the function-point metric is better. However, many argued that data on software function points are much more limited than for LOC. The research team found adequate data on function point and converting coefficient from LOC to function point in the literature [Jones 2008] so decided to use the function-point in this study.

Some experts commented that the three states, particularly for the development and V&V quality nodes, could not provide sufficient resolution to capture variability in quality activity implementation. There was one suggestion that in addition to use the three states for the attributes, the quality nodes should use an expanded numerical scale. In addition, some experts thought the three states were not

well defined in the sense that it is not clear what activities constitute the "extra activity" that would qualify an attribute score to a "High" state. Considerations under the selection and definition of the scoring scales include the amount of data required for NPTs, and ease for expert inputs. Since this study demonstrates a framework that practitioners can follow in the future to apply BBN techniques to NPP applications, this model is not viewed as a final solution to the actual NPP applications. The selection of nodes, their scoring scale, the causal relationships and NPTs are expected to be adjusted to meet specific NPP application development situations.

7.1.2 Theoretical Questions

Two BBN experts were asked about their opinion on the genericness of a BBN model, the sources of uncertainties that should be accounted for in such a model, and the concept of FSD, etc.

The experts agreed that this BBN model could represent NPP safety software and could be used to quantify the per demand failure probability. With that, this model can be characterized as a model "specific" for NPP safety software. This model can also be applied to a specific safety software to assess its number of faults and failure probability. In addition, the experts agreed that the probabilistic distributions of the nodes, and the NPTs should represent the variability among the class of safety-related software.

The opinions on the use of different BBN configurations such as "Diverging", "Converging", "Noisy-Or", and "Ranked Node" were solicited from two experts with prestigious academic background. As discussed in Chapter 4, this study used the diverging configuration to better describe the causal relationship and eight discretized distributions to represent NPT values to reduce uncertainties introduced by using scalar mean values. Future studies could explore different configurations and different NPT representation schemes. Extended theoretical discussions on this topic are out of the scope of this study.

Regarding the use of FSD, one expert questioned that its use as a proportionality constant (a random variable) between the number of faults and software failure probability suggests the faults have the same likelihood of being triggered. The software defect trigger mechanism given a known defect with known location is in theory deterministic. In other words, if the input condition and the software internal status repeat, the defect will be triggered repeatedly. The software internal status is also determined by input conditions (both current inputs and past inputs). For a given known defect, its triggering probability is "modulated" by the input distributions. In reality, the remaining defects are normally unknown in terms of their types and locations. This unknown makes an "analytical" analysis for trigger probability impossible. A practical alternative is to collect failure events and estimate the number of remaining defects and back calculate an averaged per defect failure probability, and extrapolate this value to similar software developed using similar process by similar teams. In this study an FSD distribution (please refer to Section 6.3.2) was estimated by examining literature NPP digital I&C demand failure data. This approach demonstrates one way to bridge the gap between the number of defects and per demand failure probability. Large uncertainty is expected for this method. Better estimates on software triggering mechanism remains an active research area for both academia and industry. Detailed discussion on this topic is out of the scope of this study.

7.2 Elicitation Phase Two - Generic Parameters of the BBN Model

The objective of this study's expert elicitation is to estimate the node probability tables of the child nodes of the BBN model. The NPT values are expected to be NPP safety software specific. Therefore, experts in the nuclear industry with experience in managing and developing safety-

related software were identified. Seven experts from different organizations who participated in the elicitation are named in Appendix C.2.

The questionnaire of this round of elicitation includes a brief description of our BBN model, definition of the terms, explanation of the questions, and tables for the experts to fill out. Because each expert might have limited knowledge on quantitative relationships capture in this BBN model, experts were advised that they did not have to answer questions that they did not have the requisite knowledge or experience to answer. The questionnaire and the answers the experts provided are included in [NRC 2016b].

Of the seven experts who participated in this phase, two experts completed the entire questionnaire. Some experts only provided point estimates to some questions, instead of a discretized probability distribution, or only answered the question conditional on the "Medium" state but not the "High" and "Low" states. Two experts believed the Installation and Checkout phase did not contribute to fault insertion or removal.

Some of the values given by the experts required additional processing. For example, for fault detecting probability, the three percentiles requested need be further converted to a probability distribution that was modeled using the WinBUGS tool [Spiegelhalter 2003]. In addition, the effect of software size/complexity on fault detection probability was represented as coefficient of the base-case detection probability. That is, experts were asked to provide the detection probability for the Medium state of V&V quality, and then, to provide coefficients for the High and Low state. Therefore, a manual calculation is needed. The details of the distribution fitting are given below.

7.2.1 Distribution Fitting Based on Expert Opinions

A potential limitation of using expert opinions to estimate the quantities of NPTs for software V&V, specified in the BBN model, comes from the diversity of the experts' opinions. Chapter 1 describes some sensitivity calculations done by removing some of the experts' inputs. Multiple experts may provide widely diverse opinions which should be treated in an integrated manner to estimate NPTs for a specific software development process. This variety in the experts' opinions could be better caught in a probabilistic manner, thus, the uncertainty associated with specified NPTs should be represented by using the distribution of the experts' opinions. There are seven categories of NPTs specified in the developed BBN model as follows:

- Prior Distribution of Development Nodes
- Prior Distribution of V&V Nodes
- Number of Function Points
- Attribute Nodes
- Defect Density
- Defect Detection Probability for Current Phase
- Defect Detection Probability for Previous Phase

In this study, the distributions that give the best fit for the empirical distribution sampled from the experts' opinions were used to represent the uncertainty associated with the NPTs for various nodes specified in the BBN model. In this process, continuous univariate distributions were used to fit the empirical distribution sampled from the data by using the distribution-fitting techniques provided by MATLAB's ALLFITDIST method [Sheppard 2012]. In this study, the continuous univariate distributions used for the distribution fitting of the experts' opinion include Gamma, Logistic, Lognormal, Normal, Weibull, Beta, and Pareto distributions. The parameters of the fitted distribution were reported and

used as input data for the WinBUGS program to specify the distribution of each node probability. Table 7-1 summarizes the distribution fitting results for each node.

Table 7-1 Summary on the Distribution Fitting of the NPTs Based on Experts' Opinion

Subject	Parameters to be estimated	Format of experts' opinion	Distribution Fitting method	Notes
Development quality	Prior distribution over *High*, *Medium*, and *Low* quality	A discretized distribution (per phase)	A Beta distribution was fitted to the 7 or fewer estimates, one per expert.	- Beta distribution showed the lowest AIC and BIC values in 9 out of 15 data sets.
V&V quality	Prior distribution over *High*, *Medium*, and *Low* quality	A discretized distribution (per phase)	A Beta distribution was fitted to the 7 or fewer estimates, one per expert.	- Beta distribution showed the lowest AIC and BIC values in 9 out of 15 data sets.
Number of function points	Prior distribution over High, Medium, and Low complexity state	A discretized distribution	A Beta distribution was fitted to the 7 or fewer estimates, one per expert	- Beta distribution showed the lowest AIC and BIC values in 2 out of 3 data sets.
NPT of attribute nodes	Conditional distribution of attribute nodes given *High*, *Medium*, and *Low* Development or V&V quality	A discretized distribution (per condition, per phase)	A Normal distribution was fitted to the 7 or fewer estimates, one per expert.	- Normal distribution showed the lowest AIC and BIC values in 230 out of 1125 data sets. The normal distributions were truncated beyond 0-1in the WinBUGS calculations.
NPT of defect density	Number of defect per function point given *High*, *Medium*, and *Low* Development quality and Complexity	5%, 50%, and 95%tiles (per phase, per condition)	A Gamma distribution was fitted to the 7 or fewer estimates, one per expert.	- A normal distribution was fitted by setting the median estimate as the mean and the standard deviation estimated using the 5th and 95th percentiles. - Monte Carlo simulation was used to generate an empirical distribution by sampling from the seven or fewer estimated normal distributions. - The values of generated samples were bounded to $[0,\infty]$. - Gamma distribution showed the lowest AIC and BIC values in 9 out of 15 data sets.
NPT of detection Probability for current phase	Defect detection probability given *High*, *Medium*, and *Low* V&V quality and Complexity	5%, 50%, and 95%tiles (per phase, per condition)	A Beta distribution was fitted to the 7 or fewer estimates, one per expert.	- A normal distribution was fitted by setting the Median estimate as the mean and the standard deviation estimated using the 5th and 95th percentiles. - Monte Carlo simulation was used to generate an empirical distribution by sampling from the seven or fewer estimated normal distributions. - The values of generated samples were bounded to [0,1]. - Beta distribution showed the lowest AIC and BIC values in 9 out of 15 data sets.
NPT of detection Probability for previous phase	Defect detection probability given V *High*, *Medium*, and *Low* V&V quality and Complexity	5%, 50%, and 95%tiles (per phase, per condition)	A Beta distribution was fitted to the 7 or fewer estimates, one per expert.	- A normal distribution was fitted by setting the Median estimate as the mean and the standard deviation estimated using the 5th and 95th percentiles. - Monte Carlo simulation was used to generate an empirical distribution by sampling from the seven or fewer estimated normal distributions. - The values of generated samples were bounded to [0,1]. - Beta distribution showed the lowest AIC and BIC values in 9 out of 15 data sets.

The following provides a description on how the uncertainty associated with the specified NPTs in the developed BBN model is represented by deriving the best distribution fit based on the expert elicitation for each NPT. More details can be found in Appendix D.

- NPT for the Prior Distribution of Quality Nodes

Each phase of software development has a development quality node and a V&V quality node. A prior distribution needs to be estimated for each such node over its three states *High*, *Medium*, and *Low*. The ten quality nodes of the five phases are assumed to be independent, and each node's prior distribution was estimated by seven experts.

Since the expert elicitation was given as a point estimate in the case of quality nodes, the best distribution fit was analyzed based on the point estimates given as experts' opinions for each quality node. Table D-1 and Table D-2 in Appendix D respectively show the possible fitted distributions for the prior distribution of the development and the V&V process.

To find the best fit for the data, AIC[1] and BIC[2] measures were used [Schwarz 1978]. Low values for estimated AIC and BIC imply low expected information loss, thus, the model with the lower value of AIC and BIC is preferred when fitting the distribution to the data [Kass 1995]. The Beta distribution showed the lowest AIC and BIC values in 9 out of 15 data sets in both cases for the prior distribution of development quality and V&V quality, respectively, as shown in Table D-1 and Table D-2.

Based on the distribution fitting results, a Beta distribution was used to represent the uncertainty associated with the NPT for the prior distribution. The first shape parameter (α) and second shape parameter (β) of the fitted Beta distribution were reported for NPTs of the prior distribution, as shown in Table D-3 and Table D-4.

- NPT for the Number of Function Points

In case of the number of function points, which is used in calculating the defect density and the defect detection probability, point estimates were given as an experts' opinion on the probabilities over its three complexity states (*Low* for FP≤100, *Medium* for 100≤FP≤1000, and *High* for 1000≤FP≤1500).

Therefore, the best distribution fit for the number of function points over each complexity state was analyzed, based on the point estimates given as an experts' opinion. Table D-5 shows the possible distribution for the fitted distributions in case of the NPTs of the number of function points. We found that the Beta distribution showed the lowest AIC and BIC values for 2 out of 3 data sets, as shown in Table D-5.

---

[1] The Akaike information criterion (AIC) is a measure of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Hence, AIC provides a means for model selection. AIC offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. AIC is defined as $-2\ln(\mathrm{L}) + 2\mathrm{k}$, where k is the number of estimated parameters in the model and L is the maximized value of the likelihood function of the model.

[2] Bayesian information criterion (BIC) or Schwarz criterion (also SBC, SBIC) is a criterion for model selection among a finite set of models. BIC is derived to serve as an asymptotic approximation to a transformation of the Bayesian posterior probability of a candidate model. BIC is defined as $-2\ln(\mathrm{L}) + \mathrm{k}\ln(\mathrm{n})$, where k is the number of estimated parameters in the model, L is the maximized value of the likelihood function of the model, and n is the number of data points or observations.

Based on the results of distribution fitting, the Beta distribution was used to represent the uncertainty associated with the number of function points. Table D-6 shows the first shape parameter (α) and second shape parameter (β) of the fitted Beta distribution for the number of function points over its three complexity states.

- NPT for Attribute Node

For the case of the attribute nodes, there are nine elements of the NPT for each attribute. These are derived from the three states of development and V&V qualities, respectively (e.g., High/High, High/Medium, and High/Low).

Since the expert elicitation was given as a point estimate for attribute nodes, the best distribution fit for each attribute node was analyzed based on the point estimates given as experts' opinions. Table D-7 shows the possible distribution for the fitted distributions for the NPTs of the attribute nodes. The Normal distribution showed the lowest AIC and BIC values for 230 out of 1125 data sets for attribute nodes, as shown in Table D-7.
Based on the results of distribution fitting, the Normal distribution was used to represent the uncertainty associated with the NPT for attribute nodes. Table D-8 shows the mean (μ) and standard deviation (σ) of the fitted Normal distribution for the attribute nodes. Since the distributions represent conditional probabilities, they were truncated in the WinBUGS calculations.

- NPT for Defect Density and Defect Detection Probability

For the NPT of defect density (the number of defects per function point) and defect detection probability, a normal distribution was fitted based on the estimates of three percentiles (5^{th}, 50^{th}, and 95^{th} percentile) given by experts. Considering normal distribution of $X \sim N(\mu,\sigma^2)$, both the mean and median of X are derived as μ ,and the interval of X with $1-\alpha$ probability is derived as $[\mu-\sigma z_{\alpha/2}, \mu+\sigma z_{\alpha/2}]$ [Hayter 2012], where z is a unit normal variable. Considering the interval of X with 90% coverage, the 5^{th} percentile value of X is derived as $\mu-\sigma z_{0.05}$,and the 95^{th} percentile value of X is derived as $\mu+\sigma z_{0.05}$ where $z_{0.05}$ is 1.645. Based on the estimates from the three percentiles, the 50^{th} percentile estimate of the experts' opinions is considered as the mean, μ, of the fitted normal distribution and the difference between the 5^{th} and 95^{th} percentile values is considered as the length of the 90% coverage interval, i.e., $2\sigma z_{0.05}$, thus deriving the standard deviation, σ, of the fitted normal distribution.

A Monte Carlo simulation was used to generate an empirical distribution by sampling from the seven estimated normal distributions for each specified development quality in the case of defect density, and for each specified V&V quality and complexity in case of defect detection-probability. For defect density, the samples from the normal distributions whose values are lower than 0 were truncated considering that the defect density is defined in [0, ∞]. Similarly, for defect detection probability, the samples whose values fell outside interval of [0,1] were truncated.

As shown in Table D-9, Table D-10, and Table D-11, the Gamma distribution and Beta distribution well expressed the empirical distribution compared to other distributions in the cases of defect density and defect detection probability, respectively. In terms of defect density, the Gamma distribution showed the lowest AIC and BIC values for 8 out of 15 data sets. In case of defect detection probability, the Beta distribution showed the lowest AIC and BIC values for all of data sets for both current phase and previous phase.

Based on the results of distribution fitting, the Gamma distribution and Beta distribution were used to represent the uncertainty associated with defect density and the probability of defect detection, respectively. The shape parameters (α)[3] and rate parameters (β)[4] of the fitted Gamma distribution were reported for defect density node, as shown in Table D-12. Table D-13 and Table D-14 shows the first shape parameter (α) and second shape parameter (β) of the fitted Beta distribution for the probability of defect detection probability for the current and previous phase, respectively.

7.2.2 Bayesian Update of the NPTs Using Evidence Data

One of the key features of the developed BBN model is that when the evidence for the NPTs is observed from reference or the report data, the NPTs can be updated based on Bayes' theorem. Although Bayes' theorem is mathematically simple, practical difficulties of its implementation lie in deriving the normalizing constant, the denominator in Equation (7-1) [O'Hagan 1994]. Here, the product of the prior $P(\vartheta)$ and likelihood function $P(x|\vartheta)$ must be integrated over the domain of the parameters ϑ being estimated.

$$P(\vartheta/x)=\frac{P(\vartheta)P(x|\vartheta)}{\int P(\vartheta)P(x/\vartheta)d\vartheta} \quad (7\text{-}1)$$

In this study, as one of the solutions to this integration problem, a conjugate prior family of distributions was considered to infer the parameter of a posterior distribution $P(\vartheta/x)$ given observation or evidence *x*. Based on the Bayesian update concept considering the conjugate prior, the NPTs, derived from expert elicitation, can be updated from the available evidence data.

In Bayes' theorem, if the posterior distributions $P(\vartheta/x)$ are in the same family as the prior probability distribution $P(\vartheta)$, the prior and posterior are then called conjugate distributions, and the prior is called the conjugate prior for the likelihood function.

For example, consider a random variable which consists of the number of successes in *n* Bernoulli trials with unknown probability of success *q* in [0, 1]. This random variable will follow the

---

[3] Shape parameters is a numerical parameter of a family of probability distributions that allow a distribution or density function to take on a variety of shapes. In case of Gamma probability density function (*f*) with shape *k* and scale *b*, the following properties are satisfied: 1) If $0 < k \leq 1$, *f* is concave upward, 2) If $k = 1$ *f* is decreasing with *f*(0)=1, and 3) If $1 < k$, *f* increases and then decreases, with mode at (*k*-1)*b.*

[4] The reciprocal of the scale parameter is defined as the rate parameter, particularly in the context of the Poisson process, in case of Gamma distribution. If a family of probability distributions is such that there is a parameter *s* (and other parameters θ) for which the cumulative distribution function satisfies $F(x; s, \theta) = F(x/s; 1, \theta)$, then s is defined as a scale parameter.

Binomial distribution, with a probability mass function that can be expressed as a function of q, having the form for some constants a and b as:

$$p(x)=\binom{n}{x}q^x(1-q)^{n-x} \propto q^a(1-q)^b \tag{7-2}$$

Considering the Beta distribution, which is a conjugate prior for the Bernoulli likelihood, the prior distribution is as follows:

$$p(q)=\frac{q^{\alpha-1}(1-q)^{\beta-1}}{B(\alpha,\beta)} \tag{7-3}$$

where α and β are chosen to reflect any existing belief or information and q is the parameters of the underlying model. In this context, α and β are called prior hyperparameters (parameters of the prior). Considering that we sample a random variable q, and get s successes and f failures, the posterior distribution and its hyperparameters can be derived as follows:

$$P(s,f/q=x)=\binom{n}{x}x^s(1-x)^f \tag{7-4}$$

$$P(x)=\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{7-5}$$

$$P(q=x/s,f)=\frac{P(x)P(s,f|x)}{\int P(x)P(s,f|x)d\vartheta}=\frac{x^{s+\alpha-1}(1-x)^{f+\beta-1}}{B(s+\alpha,f+\beta)} \tag{7-6}$$

where the posterior is formulated as a Beta distribution with parameters ($s+\alpha$, $f+\beta$). Then this posterior distribution can be used as the prior for more samples, with the hyperparameters adding further information as it is introduced.

In a similar manner, the posterior distribution and its hyperparameters for the Gamma conjugate prior and Poisson likelihood can be derived as follows:

$$P(x/\lambda) \propto \lambda^{\sum_i x_i} e^{-\lambda n} \tag{7-7}$$

$$P(\lambda)=\frac{\beta^{\alpha}}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda} \tag{7-8}$$

$$P(\lambda/x)\propto P(x/\lambda)P(\lambda)\propto \lambda^{\sum x_i+\alpha-1}e^{-(n+\beta)\lambda} \tag{7-9}$$

where the n number of evidence $x=x_1,x_2,\cdots,x_n$ is drawn from the Poisson distribution at a rate of λ. Here, the posterior distribution can be formulated as a Gamma distribution with ($\alpha+\sum_i x_i$, $\beta+n$). In summary, the conjugate prior can be updated with the likelihood to obtain the posterior distribution, showing the same distribution type with the conjugate prior, and its hyperparameters can be derived, as shown in Table 7-2.

Table 7-2 Example of Conjugate Prior and Associated Likelihood Function

Likelihood	Likelihood model parameters	Conjugate prior Distribution	Prior hyper-parameters	Posterior hyperparameters
Bernoulli	*p* (probability)	Beta	$\alpha,\ \beta$	$\alpha + \sum_i x_i,\ \beta + n - \sum_i x_i$
Poisson	λ (rate)	Gamma	$\alpha,\ \beta$	$\alpha + \sum_i x_i,\ \beta + n$

Based on the Bayesian update concept considering the conjugate prior family of distribution, the parameters of the distribution representing the NPTs in the BBN model were updated. As an application of the method to update the NPTs, the parameters of the distribution representing the NPTs of defect density and defect detection probability at current phase were updated using reference data and the IDiPS-RPS and LOCS anomaly reports. Note that the NPTs in the developed BBN model can be updated in a similar manner for any other observed evidence.

7.2.2.1 Bayesian Update of the NPTs using Evidence from Reference Data

In this study, the NPTs for defect density and defect detection probability at current phase in the BBN model were updated considering the conjugate prior family of those using the reference data [Jones 2008] on the software defect potentials[5] and defect removal efficiency[6] of software as the evidence for the Bayesian update.

The number of defects in each phase of the SDLC was estimated to be a Gamma distribution based on expert elicitation, and was updated using Poisson likelihood for the Bayesian update based on the reference data for the software defect potentials. In the analysis, we assumed that the mean defect density was estimated in the expert elicitation process, therefore, the software defect potentials in the reference data are treated as observations, or mean (λ) of Poisson likelihood. For the simplicity, the number of inserted defects was estimated by using the mean values of the Poisson process instead of sampling from the Poisson distribution.

Table 7-3 shows approximate quality levels for various software applications, including important quality metrics such as defect potentials and defect removal efficiency. In this study, among the various software types, we assume that CMM Level 5 joined with Six-Sigma represents *High* development or V&V quality, CMM Level 4 represents *Medium* development or V&V quality, and Spiral represents *Low* development or V&V quality. Since the research team does not have enough information regarding how many cases were analyzed for deriving the defect potentials and their removal efficiency in the reference, for simplicity, n (# of observations) was assumed to be 1.

[5] The software defect potential is defined in [Jones 2008] as defect introduction rate per function point. It is a global number across the SDLC.

[6] The defect removal efficiency is defined in [Jones 2008] as defect detected and removal ratio to the total number of defects introduced. It is a global number across the SDLC.

Table 7-3 Selected Software Development Quality Levels and Defect Characteristics

Type	Defect Potentials (per function point)	Defect Removal Efficiency
CMM5 + Six-Sigma	4.80	98.00%
CMM Level 4	6.00	93.00%
Spiral	6.50	85.00%

In Table 7-3, the defect potential refers to the sum of possible errors in the software from five separate sources: errors in requirements, errors in design, errors in source code, errors in user documentation, and errors associated with bad fixes or secondary errors introduced while fixing a primary error. Table 7-4 illustrates the overall distribution of software errors among the various categories of origin points from the industry segment.

For the Bayesian update of the defect density NPT in the BBN model, the evidence for the defect density in each phase of the SDLC was derived as the multiplication of the defect potential and defect origin percentage. Since the defect origin in the installation phase was not reported, the software defect origin percent for the Installation and Checkout phase was assumed to be 0%. The code bugs and bad fix bugs were assumed to be defects from the Implementation and Test phases, respectively. The document bugs were not considered defects in this study.

Table 7-4 Software Defect Phase Allocations [Jones 2008]

Type	Defect Origin Percentage
Requirements Bugs	10%
Design Bugs	25%
Code Bugs	40%
Document Bugs	15%
Bad Fix Bugs	10%
Total	100%

Based on the evidence from the reference data, a Poisson distribution is used to represent the likelihood for the conjugate prior (defect density) which was derived from expert elicitation represented as a Gamma distribution. Table 7-5 shows the updated results for the defect density which includes the posterior hyperparameters of the updated Gamma distribution for each development quality at each SDLC phase.

Table 7-5 Bayesian Updated Defect Density

Phase	Development Quality	Alpha	Beta	Mean	Variance
Requirements	High	0.912	1.551	0.588	0.379
	Medium	1.062	1.381	0.769	0.557
	Low	1.064	1.171	0.909	0.777
Design	High	1.656	1.368	1.210	0.885
	Medium	1.976	1.253	1.578	1.259
	Low	2.125	1.152	1.844	1.601
Implementation	High	2.421	1.332	1.818	1.365
	Medium	2.924	1.196	2.444	2.043
	Low	3.129	1.197	2.615	2.184
Test	High	1.000	2.306	0.434	0.188
	Medium	1.223	1.924	0.636	0.330
	Low	1.064	1.171	0.909	0.777
Installation and Checkout	High	0.571	2.464	0.232	0.094
	Medium	0.621	1.680	0.370	0.220
	Low	0.580	1.380	0.420	0.304

Regarding the NPT for defect detection probability at current phase, the defect removal efficiency reported in the reference data was used as evidence, as shown in Table 7-3. Here, the defect removal efficiency refers to the percentage of defects removed before delivery of the software to its users. In this study, the defect removal efficiency reported in the reference data was assumed to be the defect detection probability at each SDLC phase.

For the Bayesian update of the defect detection probability NPT, a Bernoulli distribution was used to represent the likelihood for the conjugate prior, here meaning the defect detection probability at current phase represented as a Beta distribution in the expert elicitation. In the analysis, the software defect removal efficiency in the reference data was treated as an observation, specifically the probability (p) of Bernoulli likelihood, since the expert elicitation on the defect detection probability corresponds to the distribution of p.

After updating the NPT for defect detection probability with the evidence from the reference data, the posterior hyperparameters of the Beta distribution for each V&V quality at each SDLC phase were estimated. The software in the reference data was assumed to have Medium complexity. Therefore, based on the updated mean and variance of the Beta distribution for Medium complexity, the posterior hyperparameters of the Beta distribution for the High and Low complexity cases were estimated by γ as follows:

$$(updated\ mean)_{i,j}=(updated\ mean)_{M,j}*\ \gamma_{i,j} \tag{7-10}$$

$$(updated\ variance)_{i,j}=(variance)_{i,j}*\frac{(updated\ mean)_{i,j}}{(mean)_{i,j}} \tag{7-11}$$

$$\gamma_{i,j}=\frac{(mean)_{i,j}}{(mean)_{M,j}} \tag{7-12}$$

where *i* represents the degree of complexity (*H*: High, *M*: Medium, *L*: Low) and *j* represents the degree of V&V quality (*H*: High, *M*: Medium, *L*: Low). Here, *mean* and *variance* denotes the mean and variance of the prior Beta distribution derived from the expert elicitation, and the *updated mean* and *updated variance* denote the mean and variance of the posterior Beta distribution updated from the reference data.

Table 7-6 shows the updated result of defect detection probability at current phase for the Requirements phase using reference data. See Table D-15 for the updated results of the NPT for every SDLC phase.

Table 7-6 Bayesian Updated "Current Phase Defect Detection Probability" Results in Requirements Phase

Phase	Complexity	V&V Quality	Alpha	Beta	Mean	Variance
Requirements	High	High	3.012	1.680	0.642	0.040
	High	Medium	2.299	1.728	0.571	0.049
	High	Low	1.272	1.780	0.417	0.060
	Medium	High	6.446	1.427	0.819	0.017
	Medium	Medium	3.911	1.498	0.723	0.031
	Medium	Low	1.637	1.510	0.520	0.060
	Low	High	3.739	0.588	0.864	0.022
	Low	Medium	4.087	1.056	0.795	0.027
	Low	Low	1.810	1.403	0.563	0.058

7.2.2.2 Bayesian Update of the NPTs using Evidence from Anomaly Report Data

Based on the defect estimates data reported in IDiPS-RPS and LOCS anomaly reports, the defect density NPT in the BBN model was Bayesian updated considering the conjugate prior family of distributions. In this study, the defect estimates in the anomaly reports for both applications were assumed to be the number of defects detected in each SDLC phase, and used as evidence to update the defect density NPT. Tables 7-7 and 7-8 show the defect estimates reported in the IDiPS-RPS and LOCS anomaly reports, respectively.

In the analysis, regarding IDiPS-RPS, the number of defects detected at the Test phase was assumed to be the sum of defect estimates reported in the Integration and Validation phases. In addition, the number of defects in the Installation and Checkout phase was not considered since the system had not yet been installed, thus, the defect density for Installation and Checkout phase was not updated with the IDiPS-RPS data.

In the case of defect estimates for LOCS, the number of anomaly reports was assumed to be the number of defects detected at each SDLC phase, as shown in Table 7-8. In this study, the defect estimates reported for both IDiPS-RPS and LOCS applications were used as evidence to update the defect density NPT.

Table 7-7 Defect Estimates from the IDiPS-RPS Anomaly Report [KAERI 2010]

Type	Phase	Estimated Defects
BP	Requirements	6
	Design	16
	Implementation	3
BP/CP/ATIP/COM	Integration	4
BP/CP/ATIP/COM	Validation (System Testing)	4

Table 7-8 LOCS Defect Estimates Based on Anomaly Report

Phase	Estimated Defects
Requirements	1
Design	2
Implementation	2
Test	2
Installation and Checkout	2
Total	9

Based on the IDiPS-RPS and LOCS report data on the defect estimates, the number of defects x_j introduced in j SDLC phase can be derived as follows:

$$x_j*(P_{j,H}*V_{j,H}+P_{j,M}*V_{j,M}+P_{j,L}*V_{j,L})=y_j \quad (7\text{-}13)$$

where y_j is the defect estimates from both anomaly reports, $P_{j,i}$ is the defect detection probability at i V&V quality in j SDLC phase, and $V_{j,i}$ is the posterior distribution for i V&V quality in j SDLC phase.

From the derived number of defects at each SDLC phase (x_j), the number of defects for each development quality was derived using the posterior distribution of development quality for IDiPS-RPS and LOCS, as follows:

$$N_{j,H}*D_{j,H}+N_{j,M}*D_{j,M}+N_{j,L}*D_{j,L}=x_j \quad (7\text{-}14)$$

$$N_{j,H}=\beta_{j,H,M}*N_{j,M}, N_{j,L}=\beta_{j,L,M}*N_{j,M} \quad (7\text{-}15)$$

$$\beta_{j,H,M}=\frac{(updated\ mean)_{j,H}}{(updated\ mean)_{j,M}},\ \beta_{j,L,M}=\frac{(updated\ mean)_{j,L}}{(updated\ mean)_{j,M}} \quad (7\text{-}16)$$

where x_j is the number of defects in each SDLC phase derived from Equation (7-13), $N_{j,i}$ is the mean number of defects at i development quality in j SDLC phase, and $D_{j,i}$ is the posterior distribution for i development quality in j SDLC phase. Here, $\beta_{j,H,M}$ is the ratio of the updated mean of High development quality to that of Medium development quality, and $\beta_{j,L,M}$ is the ratio of the updated mean of Low development quality to that of Medium development quality.

The number of defects per FP, or defect density, for each SDLC phase was derived by dividing the number of defects by the number of FPs (56 and 41 for IDiPS-RPS and LOCS, respectively).

The evidence from IDiPS-RPS and LOCS anomaly reports was used to update the defect density results from the reference data by considering the Gamma conjugate prior with Poisson likelihood, as shown in Table 7-9.

Table 7-9 Bayesian Updated Results for Defect Density From the IDiPS-RPS and LOCS Anomaly Report Data

Phase	Development Quality	Alpha	Beta	Mean	Variance
Requirements	High	1.1043	3.5507	0.3110	0.0876
	Medium	1.3130	3.3811	0.3883	0.1149
	Low	1.3596	3.1705	0.4288	0.1353
Design	High	2.2558	3.3680	0.6698	0.1989
	Medium	2.7565	3.2526	0.8475	0.2606
	Low	3.0353	3.1522	0.9629	0.3055
Implementation	High	2.5989	3.3317	0.7800	0.2341
	Medium	3.1588	3.1963	0.9883	0.3092
	Low	3.3807	3.1969	1.0575	0.3308
Test	High	1.2439	4.3055	0.2889	0.0671
	Medium	1.5775	3.9236	0.4021	0.1025
	Low	1.5630	3.1705	0.4930	0.1555
Installation and Checkout	High	0.6106	3.4640	0.1763	0.0509
	Medium	0.6838	2.6797	0.2552	0.0952
	Low	0.6514	2.3803	0.2737	0.1150

7.3 Elicitation Phase Three - Specific Parameters of the BBN Model

The objective of this expert elicitation was to evaluate attributes for specific software. The experts used were familiar with the development of the LOCS and the IDiPS-RPS. Appendix C.3 lists the experts.

An elicitation questionnaire that contains definitions of node states, descriptions of attributes, and a guidance for answering the questions was sent to the experts. The guidance explains how to score the quality of carrying out the activities associated with each attribute, and tabulates a complete list of attributes and their associated activities. Experts were asked to score each attribute on the basis of High, Medium, and Low. The questionnaire with the guidance and the answers the expert provided are given in [NRC 2016c]. In addition, [NRC 2016c] lists the Round 3 answers for LOCS and IDiPS-RPS.

8 APPLICATIONS TO NUCLEAR DIGITAL I&C SYSTEMS

A BBN model has been developed in previous chapters to estimate the number of defects remaining and thus the on demand failure probability of NPP digital I&C safety systems. This chapter describes two applications of this model to ATR LOCS and IDiPS RPS.

8.1 Application to LOCS

8.1.1 LOCS Development Activities

The Loop 2A LOCS software was determined by quality-level (QL) analysts as QL-2. This designation determines activities in the software life-cycle management and quality controls [INL 2009a, INL 2010b] and the level of rigor for software-development life cycle activities. The QL-2 designator indicates that failure of the software creates a medium risk. The required activities for the QL-2 items are described in EXH-13620-2 [INL 2010b].

In the ATR software management plan [INL 2010a], the lifecycle of the LOCS software is divided into three main phases: Development, operation, and retirement. The focus of this study is on the development phase and its associated activities. As specified in [INL 2010a], the development phase comprises planning and requirements, design, implementation and closeout. The implementation activities are further broken into procurement, fabrication, assembly, construction, testing, coding, and turn-over. The entire developmental process is governed by the requirements specified in the NQA-1 standard [ASME NQA].

Design inputs and requirements, deliverables, verification methods, and organizational interfaces are identified in the planning and requirements phase. As part of this activity, the safety software determination (SSD) is prepared, and the quality level determination (QLD) is completed. The result of the QLD determines the appropriate software quality assurance (SQA) requirements to be followed for the rest of the lifecycle activities. Other products of the planning and requirements include the quality assurance plan (QAP), and the V&V plan.

Design activities for the LOCS software are consistent with those specified in DOE-G 414.1-4 [DOE 2010]. These activities include performing and documenting the design analyses, documenting risks and hazards and their mitigation, and preparing qualification and testing documents. Design implementation, described in [DOE 2010] includes coding the final software product, completing the qualification, and testing documents. The Software Project Manager ensures the proper adherence to the requirements for testing and validating. The criteria and rigor of the tests are based on the quality level of the software.

The traceability requirements of the LOCS software are specified in TFR-499 (“2A Loop Instrumentation and Operating Control System”) [INL 2010c]. The requirements for the Metso Automation portion of the system vendor (Metso Automation) are set out in SPC-988 (“Distributed Control System for Loop 2A at the Advanced Test Reactor”) [INL 2009b].

The software development standard assigns different responsibilities to different individuals associated with software development. The owner of the software is responsible for determining its appropriate safety level. The quality-level analyst identifies and documents the software’s quality level. The quality level, in turn, determines the applicable software-management activities that are performed throughout the software’s life cycle. Management assigns the software’s owner, the software’s project manager, the technical lead, and the quality-level analyst.

The testing as part of the V&V activities for the LOCS software include the factory acceptance test (FAT), the grooming test, and the system-operability (SO) test [INL 2014a]. The FAT is developed by Metso Automation (the vendor), and is used to verify that the system meets the requirements specified by the requirement specification. The grooming test, performed at the ATR facility, is specified by a written test-procedure that contains steps to exercise and verify the software and hardware functions. Anomalies found during the test are corrected, and the grooming test repeated until an acceptable baseline is obtained. The V&V report includes the anomaly reports that document each anomaly detected thereby. Additionally, the anomaly report documents the description and location (in the code) of the anomaly, its impact and cause, the criticality level, and recommendations and resolutions associated with it.

The ATR software testing includes the loop 100-day functional test, the loop-equipment checks, the loop system test, and the ATR loop distributed Control System (DCS) integrated system operational test. The tests were developed and controlled according to [INL 2010a].

The FAT was performed by Metso Automation witnessed by the software technical lead and the SQA specialist. A total of 71 software anomalies were found from the LOCS factory acceptance testing [INL 2014b]. Seven of these anomalies were associated with the safety system. The grooming test was performed at the ATR after the system was installed. This test found 11 anomalies, all of which were resolved. Two of the anomalies were associated with software and one also was associated with the safety system. The final set of tests (system operability tests) was performed after the grooming tests using the previously established, detailed operating procedures (DOPs). The V&V report indicates that the system operability tests found 18 errors; all were resolved. Eleven of these anomalies were associated with software and none were related to the safety system.

8.1.2 LOCS Results

Table 8 1 summarizes the scores provided by this expert. Table 8 2 and Table 8 3 list the number of defects introduced in each phase and the number of defects remaining at the end of each phase, respectively. Table 8 4 and Table 8 5 list the probability of detecting defects for defects introduced in the current phase and in previous phases, respectively.

Table 8-1 Attribute Evaluation Results of LOCS

Phase	High Attributes	Medium Attributes	Low Attributes
Requirements Development	1	7	4
Requirements V&V	0	10	5
Design Development	0	9	6
Design V&V	0	7	9
Implementation Development	0	8	8
Implementation V&V	0	9	9
Test Development	0	7	3
Test V&V	0	8	3
Installation/Checkout Development	0	5	0
Installation/Checkout V&V	0	6	1

Table 8-2 Number of Defects Introduced in Each Phase for LOCS

Phase	Number of Defects Introduced				
	Mean	**σ**	**5th**	**50th**	**95th**
Defects introduced in Requirements Phase	15.93	29.18	0	0	82
Defects introduced in Design Phase	35.54	43.71	0	41	123
Defects introduced in Implementation Phase	42.35	47.85	0	41	123
Defects introduced in Test Phase	16.52	29.15	0	0	82
Defects introduced in Installation Phase	10.48	24.24	0	0	41

Table 8-3 Number of Defects Remaining at the End of Each Phase for LOCS

Phase	**Number of Defects Remaining**				
	Mean	**σ**	**5th**	**50th**	**95th**
Defects remaining at the end of Requirements Phase	3.279	8.244	0	0	19
Defects remaining at the end of Design Phase	17.9	23.74	0	10	66
Defects remaining at the end of Implementation Phase	29.06	30.12	0	21	88
Defects remaining at the end of Test Phase	13.19	14.83	0	9	42
Defects remaining at the end of Installation Phase	6.018	8.602	0	3	23

Table 8-4 Defect-detection Probability for Defects Introduced in the Current Phase of LOCS Development

Phase	**Defect Detection Probability**				
	Mean	**σ**	**5th**	**50th**	**95th**
Requirements Phase	0.79	0.16	0.47	0.83	0.99
Design Phase	0.57	0.22	0.19	0.58	0.91
Implementation Phase	0.61	0.25	0.17	0.64	0.96
Test Phase	0.73	0.14	0.47	0.74	0.93
Installation Phase	0.80	0.14	0.54	0.83	0.97

Table 8-5 Defect-detection Probability for Defects Introduced in Previous Phases of LOCS Development

Phase	Defect Detection Probability				
	Mean	σ	5th	50th	95th
Design Phase	0.22	0.14	0.04	0.20	0.48
Implementation Phase	0.30	0.17	0.07	0.28	0.60
Test Phase	0.70	0.16	0.41	0.72	0.93
Installation Phase	0.70	0.19	0.33	0.73	0.96

Table 8-6 and Table 8-7 summarizes the probability of the development and V&V qualities, respectively. These distributions were obtained by updating the prior distributions (from the expert elicitation) with the attribute scores elicited from the specific expert (third-round of expert elicitation). It is worth noting that posterior distributions show that both "Development Quality" nodes and "V&V Quality" nodes for all SDLC phases are not likely to be "High". This observation is consistent with definitions for node score levels, as "Medium" represents activities adequate to licensing review, extra activities were not pursued by industry.

Table 8-6 Posterior Distribution for Development Quality at Each Phase

	High	Medium	Low
Requirements Phase	0.00	0.97	0.03
Design Phase	0.00	0.84	0.16
Implementation Phase	0.00	0.34	0.66
Test Phase	0.00	0.98	0.02
Installation Phase	0.01	0.99	0.00

Table 8-7 Posterior Distribution for V&V Quality at Each Phase

	High	Medium	Low
Requirements Phase	0.00	0.99	0.01
Design Phase	0.00	0.01	0.99
Implementation Phase	0.00	0.03	0.97
Test Phase	0.00	0.98	0.02
Installation Phase	0.00	1.00	0.00

The number of faults remaining at the end of the installation and checkout phase given in Table 8-3 was used with the FSD in Table 6-3 to obtain a distribution for the probability of LOCS software failure-on-demand in Table 8-8, below.

Table 8-8 LOCS Failure on Demand Probability

Mean	σ	5th Percentile	Median	95th Percentile
6.04E-04	1.79E-05	0.00E+00	8.96E-06	1.42E-03

8.2 Application to IDiPS-RPS

8.2.1 IDiPS-RPS Development Activities

The IDiPS-RPS is a digitalized reactor-protection system developed in the Korea Nuclear Instrumentation and Control System (KNICS) project for newly constructed NPPs), as well as for upgrading existing analog-based reactor protection system (RPSs). It has the same function as an analog-based RPS to automatically generate a reactor-trip signal, and engineered safety-features actuation signals whenever process variables reach their corresponding predefined trip set-points. IDiPS-RPS consists of four redundant channels located in rooms that are electrically- and physically-isolated. As shown in Figure 8-1, each channel is composed of four main processors: The bistable processor (BP), the coincidence processor (CP), the automatic test and interface processor (ATIP), and the cabinet operator module (COM) [Park 2012, Eom 2013].

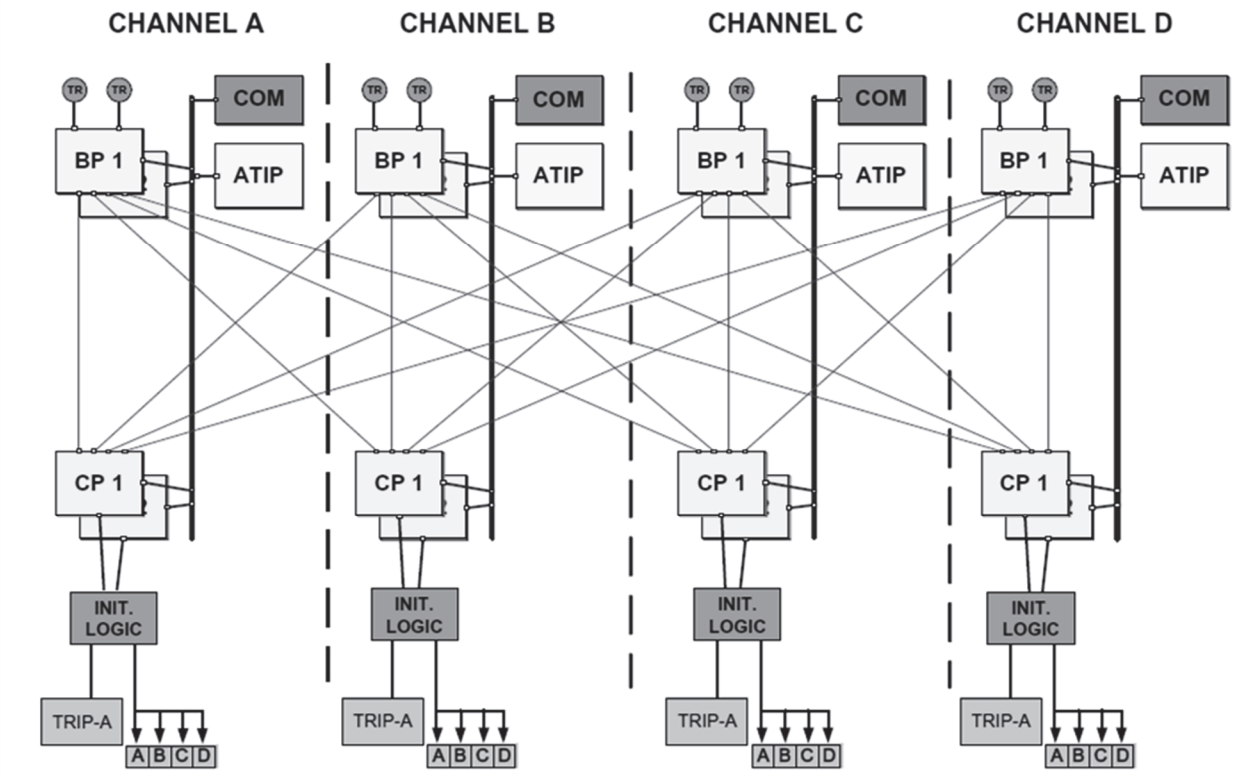

Figure 8-1 Overall Configuration of the IDiPS-RPS [Park 2012]

The BP compares measured process variables with predefined trip set-points for determining the trip state. The CP generates a hardware-actuating trip signal through a two-out-of-four voting logic, with the trip signals received from its corresponding BPs of four channels. When a channel is bypassed, the trip logic is changed to two-out-of-three voting logic. The ATIP performs various functions for validating the integrity of the BPs and CPs, and an equipment diagnosis for all the

processors in the same channel. The ATIP also tests the RPS status indications, alarms, and functions to verify the operational status of the BP and CP, and transmits this information to the COM for display to the operators. Each ATIP is connected with the other channels' ATIPs, and to an engineered safety-features-component control system. The COM is composed of a computer-based system and a hardware-based system. The computer system in the COM displays information on the operational status of all the processors in one channel of the IDiPS-RPS. The hardware system is responsible for protection-related controls, such as a channel bypass and an initiation-circuit reset. Each channel of the IDiPS-RPS has two redundant BPs and CPs. BP1 and BP2 in channel A are connected to process variables in a reverse order from each other. Also, the trip logic in a BP is executed in the reverse order to the other redundant BP to provide diversity [Park 2012].

Software implements the trip functions of the BP, the voting logic of the CP, and the test functions of the ATIP. Thus, any malfunction of the software in the BP or CP may result in irreversible consequences. Moreover, the software in the BPs and CPs in the four channels is identical, except for some minor differences. This correspondence holds the potential for inducing a common-cause failure (CCF) in the IDiPS-RPS, which might affect the safety of an NPP.

Most of the software in the RPS is classified into a safety–critical or safety-related class. Thus, the software used in the IDiPS-RPS was developed under a rigorous procedure, with independent V&V.

Figure 8-2 shows the SDLC, and the V&V activities performed during the development of the IDiPS-RPS software. The main V&V activities in the requirements and design phases are the evaluation of licensing suitability, the detailed inspection, and the traceability analysis. The purpose of evaluating the suitability for licensing is to confirm whether the software requirements and design descriptions possess the required software properties. In the V&V activities, the acceptance criteria are established according to BTP-7-14 [NRC 2007]. The independent V&V team in the KNICS project carefully established well-defined checklists for each of the characteristic properties, and a systematic evaluation of the licensing suitability was performed based on these checklists [Eom 2013].

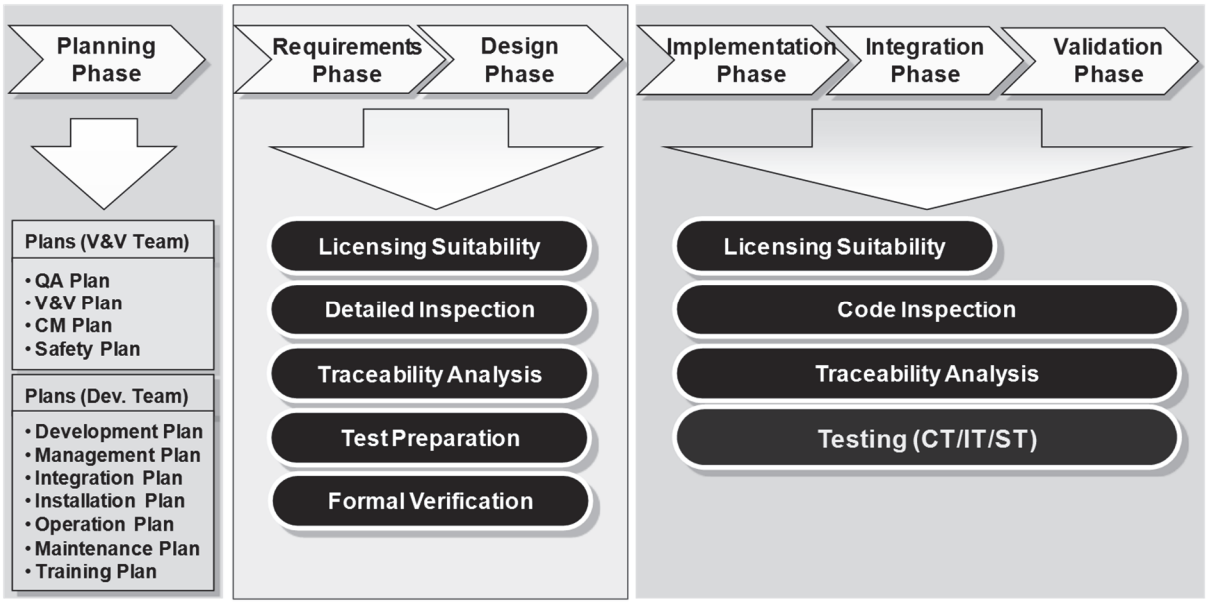

Figure 8-2 Software V&V Activities for BP Software [Park 2012]

The detailed inspection of the software requirements specification (SRS) and the software design specification (SDS) was conducted by the so-called Fagan inspection method [Fagan 1976]. This detailed inspection focuses on the functional behavior of the software system. It evaluates development documents for their correctness, completeness, and consistency. Each viewpoint is divided into four sub-viewpoints: functional definition, input- and output-definitions, behavior specification, and interfaces. The analyses for evaluating licensing suitability together with the sub-viewpoints in the detailed inspection were undertaken based on checklists that previously were derived. They were refined carefully with the aid of many software V&V experts based on various standards, guidelines, and some related reports, as well as on their V&V experiences. A bidirectional traceability analysis was carried out between the requirements of the IDiPS-RPS system and SRS, or the SRS and SDS. The results of the traceability analysis were tabulated at the unit-functional module level [Eom 2013].

8.2.2 IDiPS-RPS Results

From the third round of eliciting expert opinion for the IDiPS-RPS, the developmental- and V&V-activities carried out during the IDiPS-RPS development process were evaluated. The expert who evaluated them was involved in the IDiPS-RPS development, and also participated in the first- and second-round expert-opinion elicitations in this work. The development experience of the expert is related to the prototype of the IDiPS-RPS, rather than to its final version that is supposed to be adopted in digitalized NPPs after through revision. The evaluation results for the prototype are shown in Table 8-9.

Table 8-9 Attribute Evaluation Results of IDiPS-RPS

Phase	High Attributes	Medium Attributes	Low Attributes
Requirements Development	0	9	3
Requirements V&V	0	7	8
Design Development	0	12	3
Design V&V	0	6	10
Implementation Development	0	11	5
Implementation V&V	0	4	14
Test Development	0	7	3
Test V&V	0	0	11
Installation/Checkout Development	NA	NA	NA
Installation/Checkout V&V	NA	NA	NA

As mentioned above, since the prototype of the IDiPS-RPS was used, there are no data for the installation-and-checkout-phase. Therefore, in the application, the attributes in this phase are assumed as ‘Medium’.

For the specific software application, attribute evaluations and the number of function points of the target software must be provided as the inputs for the BBN model. The number of FPs was estimated approximately based on the LOC), with the conversion factors as shown in Table 8-10 because only the LOC data was available for the IDiPS-RPS [Park 2012]. In that table, the “nominal level” of a programming language is the number of code statements in the basic

assembly language that would provide about the same set of functions as one state in a higher-level language. The data in the table indicates the ranges and mean values in the number of source-code statements required to encode one function point for selected languages. The counting rules for the source code are based on logical statements, rather than on physical lines of code. The deviations are due to variations in individual programming styles, and to variations in how the code actually is counted. A more accurate conversion is obtained when logical statements are counted rather than physical lines [Jones 2008].

Table 8-10 Ratios of Source-code Statements to Function Points for Selected Programming Languages [Jones 2008]

Language	Nominal Level	Source Statements per Function Point		
		Low	Mean	High
Basic Assembly	1.0	200	320	450
C	2.5	60	128	170
FORTRAN	3.0	75	107	160
COBOL	3.0	65	107	150
C++	6.0	30	53	125
Ada 9X	6.5	28	49	110
SQL	27.0	7	12	15

The BP software of IDiPS-RPS was developed using a function block diagram, and was converted to C source code by the compiler. The total LOC of the C source program for the BP software is 18,652, and the LOC related to trip functions is 13,047. For a more accurate conversion, the logical statements of the BP C code were counted and the LOC of logical statements for trip functions was identified as 7,186. Since the source statements per function point of C language is 60, 128, and 170 for low, mean, and high, respectively, the estimated function points of the BP code is in the range of 42 to 120 (Low: 120 FPs, Mean: 56 FPs, and High: 42 FPs).

Table 8-11 to Table 8-14 show the evaluation results of the IDiPS-RPS BP software. In this evaluation, it was assumed that the BP software has the mean source statements per function point. Table 8-11 shows the number of defects introduced in each phase of developing the IDiPS-RPS.

Table 8-11 Number of Defects Introduced in Each Phase for IDiPS-RPS

	Mean	σ	5th	50th	95th
Defects introduced in Requirements Phase	21.78	39.85	0	0	112
Defects introduced in Design Phase	47.52	58.94	0	56	168
Defects introduced in Implementation Phase	55.28	63.76	0	56	168
Defects introduced in Test Phase	22.55	39.76	0	0	112
Defects introduced in Installation Phase	14.37	33.23	0	0	56

Table 8-12 shows the estimated number of defects remaining at the end of each phase.

Table 8-12 Number of Defects Remaining at the End of Each Phase for IDiPS-RPS

	Mean	σ	5th	50th	95th
Defects remaining at the end of Requirements Phase	8.67	19.50	0.00	0.00	47.00
Defects remaining at the end of Design Phase	27.40	34.45	0.00	17.00	96.00
Defects remaining at the end of Implementation Phase	41.31	41.59	0.00	31.00	123.00
Defects remaining at the end of Test Phase	23.58	25.09	0.00	16.00	73.00
Defects remaining at the end of Installation Phase	9.89	13.19	0.00	5.00	36.00

Table 8-13 and Table 8-14 respectively show the detection probabilities for defects introduced in the current and previous phases.

Table 8-13 Defect-detection Probability for Defects Introduced in the Current Phase of IDiPS-RPS

	Mean	σ	5th	50th	95th
Requirements Phase	0.60	0.25	0.17	0.63	0.95
Design Phase	0.57	0.22	0.18	0.58	0.91
Implementation Phase	0.60	0.25	0.17	0.63	0.95
Test Phase	0.64	0.16	0.35	0.65	0.88
Installation Phase	0.80	0.14	0.54	0.83	0.97

Table 8-14 Defect Detection Probability for Defects Introduced in Previous Phases of IDiPS-RPS

	Mean	σ	5th	50th	95th
Design Phase	0.22	0.13	0.04	0.20	0.48
Implementation Phase	0.29	0.16	0.07	0.27	0.59
Test Phase	0.63	0.18	0.31	0.64	0.90
Installation Phase	0.70	0.19	0.33	0.73	0.96

Table 8-15 and Table 8-16, respectively show the posterior distribution for Development Quality and V&V Quality.

Table 8-15 Posterior Distribution for Development Quality at Each Phase

	High	Medium	Low
Requirements Phase	0.00	0.99	0.01
Design Phase	0.00	1.00	0.00
Implementation Phase	0.00	1.00	0.00
Test Phase	0.00	0.99	0.01
Installation Phase	0.01	0.99	0.00

Table 8-16 Posterior Distribution for V&V Quality at Each Phase

	High	Medium	Low
Requirements Phase	0.00	0.17	0.83
Design Phase	0.00	0.00	1.00
Implementation Phase	0.00	0.00	1.00
Test Phase	0.00	0.00	1.00
Installation Phase	0.00	1.00	0.00

Table 8-17 and show the estimated number of defects introduced and remaining in each phase for three different levels of source statements per function point. Since the ratio of source statements per function point determines the number of FPs, the final remaining defects of low source statements per function point is about four times greater than that of high source statements per function point. If the exact number of FPs is identified by source course analysis, this uncertainty might be eliminated.

Table 8-17 Number of Defects Introduced in each Phase

Phase	Source statements per function point					
	Low (FP=120)		Medium (FP=56)		High (FP=42)	
	Mean	σ	Mean	σ	Mean	σ
Requirements	46.9	85.67	21.54	39.78	16.29	29.98
Design	102.2	126.7	47.53	59.07	35.83	44.3
Implementation	119.1	137.1	55.45	64.03	41.65	47.88
Test	48.05	84.97	22.78	39.78	16.89	29.56
Installation/Checkout	30.64	70.96	14.39	33.34	10.56	24.9

Table 8-18 Number of Defects Remaining at the End of Each Phase

Phase	Source statements per function point					
	Low (FP=120)		Medium (FP=56)		High (FP=42)	
	Mean	σ	Mean	σ	Mean	σ
Requirements	20.89	44.91	8.584	19.59	6.574	14.82
Design	67.2	82.36	27.27	34.37	20.76	26.15
Implementation	103	101.4	41.3	41.45	31.25	31.41
Test	59.3	58.96	23.63	25.04	17.77	18.88
Installation/Checkout	30.33	37.33	9.842	12.96	7.359	9.892

Even though the IDiPS-RPS is a safety-critical system, many attributes were evaluated as *Low* which means some of the required activities were not carried out satisfactorily. It is noted that the IDiPS prototype was evaluated for this study and its SDLC was less vigor than the final IDiPS product.

The number of faults remaining at the end of the installation and check out phase given in

Table 8-12 was used with the FSD in Table 6-3 to obtain a distribution for the probability of software Table failure on demand of IDiPS-RPS in Table 8-19 below.

Table 8-19 Probability of IDiPS-RPS Software Failure on Demand

Mean	σ	5th Percentile	Median	95th Percentile
9.51E-04	1.10E-02	0.00E+00	1.89E-05	2.39E-03

9 UNCERTAINTY, SENSITIVITY, AND IMPORTANCE ANALYSES

9.1 Sources of Uncertainty

A BBN model consists of nodes and their score levels, NPTs connecting the parent nodes and child nodes, and evidence (scores for nodes) collected as inputs to a BBN application. The following sections discuss sources of uncertainties in this study and the actions that were taken to reduce and mitigate them.

9.1.1 BBN Structure Uncertainties

The BBN model developed in this study assumes that causal relationships exist between SDLC characteristics, software production characteristics (such as size and complexity) and the number of remaining defects. However, because the SDLC characteristics can be classified in several ways, a variety of different causal relationships can subsequently be developed, as evidenced by comparison of literature studies. A set (or many sets) of complete and independent SDLC characteristics might exist. This study abstract the SDLC into development characteristics and V&V characteristics following activities defined in IEEE 1012 for demonstration purpose.

The completeness and independency of the set of nodes were evaluated by generic experts in this study. Although most experts endorsed the model, uncertainties are believed to be introduced from this perspective.

9.1.2 Parameter Uncertainties

The NPTs quantify causal relationships in a BBN. Depending upon the size of a BBN model, large NPTs can be generated that require large amounts of data for calculations. Such data is sometimes not publically available mainly due to proprietary limitations, as is the case for this study. This leads to uncertainties in parameter estimation.

Expert opinion is used in this study to acquire the data needed to construct NPTs. Since the BBN developed in the study targets NPP safety applications, a number of experts in nuclear digital I&C were used to generate NPT tables. The experts' inputs were aggregated and the end results were presented in the form of distributions rather than scalar values to better represent the variability of NPT values.

In addition, the research team identified limited software development data across many industries from the literature as well as project data for the two trial software developments. The NPTs based on expert inputs were Bayesian updated with literature data. This results in less uncertainties associated with NPTs and end results.

9.1.3 Input Uncertainties

The BBN model constructed in this study was applied to two example systems: LOCS and IDiPS-RPS. An INL staff member who was LOCS's user was used to score the nodes for LOCS application. The LOCS function point was estimated by an NRC staff. One IDIPS-

RPSdevelopment team member was used to score IDIPS-RPSnodes, and back firing[1] was used to back calculate IDiPS-RPS function points from IDiPS-RPS LOC.

Development team members and professional function point counting experts are recommended to score BBN nodes. The research team thus concluded inputs to the BBN for two applications are uncertain from their true values due to availability limits to LOCS team member and function point counting experts. Access to the complete development document for both projects were limited. These limits are expected to be lifted for future practitioner's real applications where full access to SDLC documents would be available.

9.2 Software Failure Probability vs Software Size

Chapter 5 defines the "medium" value for the "Development Quality" and "V&V Quality" nodes as "All required (or equivalent) activities were satisfactorily carried out." By this definition a software with all nodes of "medium" values should be able to pass the NRC licensing review and be a representative of deployed safety software in US NPPs in terms of the number of remaining defects (density) and failure probability per demand. This section demonstrates such fault content characteristics with respect to its size (and complexity) in function points.

A simple sensitivity study was conducted to calculate the BBN model with "Medium" scores for all nodes by varying the software size from 20 FPs to 5,000 FPs. Table 9-1 summarizes results of this experiment.

Table 9-1 Number of Remaining Defects vs Software Size in FPs

FP	20	50	100	200	500	1000	2000	5000
Requirements	1.99	4.38	11.13	21.87	54	167	332	837
Design	5.17	11.76	32.63	64.46	161	505	1001	2513
Implementation	6.49	14.74	49.62	98.44	245	772	1547	3859
Test	4.78	10.52	28.28	55.80	138	442	884	2217
Installation/checkout	3.14	6.37	18.29	35.30	87	313	630	1567
Remaining Defect Density per FP	0.16	0.13	0.18	0.18	0.17	0.31	0.31	0.31

The "Installation/checkout" row represents the number of remaining defects. The defect density per function point was represented by the row of "Defect Density per FP". An approximate flat defect density is observed for low and medium size (less than 500 FPs) safety software. The defect density almost doubles as the software size goes up. Such results are consistent with industry experiences as the bigger size the software project, the more difficult to management the development, the more defects remained in the software, and the less reliable. This observation coincides with one of the reliability principles that keeping the system simple to assure the reliability.

[1] "Back firing" refers to the lines of code and the number of function points ratio for a specific coding language.

9.3 Attribute Contribution Analysis

In this section, the research team analyzed the contribution of attributes to the quality nodes (i.e., Development and V&V quality nodes). Development Quality in each phase determines the density of defects, and V&V Quality determines the probability of detecting those defects. Development Quality and V&V Quality in each phase were estimated through the corresponding attributes. In the second expert-opinion elicitation, defect densities and detection probabilities for five phases and NPTs were estimated.

The developed BBN model has 125 attributes for estimating the Development Quality and V&V Quality of the five phases: 12 development attributes and 15 V&V attributes in the requirements phase, 15 development attributes and 16 V&V attributes in the design phase, 16 development attributes and 18 V&V attributes in the implementation phase, 10 development attributes and 11 V&V attributes in the testing phase, and 5 development attributes and 7 V&V attributes in the installation and checkout phase. From the second elicitation of expert opinion, experts estimated the NPTs of attributes in the form of conditional probabilities between the attributes, and their Development Quality or V&V Quality. For instance, an expert provided an estimation of the conditional probabilities of the Software Development Planning attribute for a given *High* Development Quality in the requirements phase as *High* = 0.6, *Medium* = 0.3, and *Low* = 0.1.

Based on the expert elicitation on the conditional probability of attribute for given Development Quality or V&V Quality, it is important to identify which attribute is a strong indicator of Development Quality or V&V Quality. For example, in the requirements phase, the conditional probabilities of the System/Software Qualification Test Plan Generation attribute for a given *High* Development Quality are 0.72, 0.26, and 0.03, respectively, for *High*, *Medium*, and *Low*. Those of the Configuration Management attribute are 0.59, 0.30, and 0.11 for *High*, *Medium*, and *Low,* respectively. This indicates that the System/Software Qualification Test Plan Generation attribute is stronger indicator of the *High* Development Quality compared to the Configuration Management attribute.

In this study, the indication measure to identify which attribute is a strong or weak indicator for Development Quality or V&V Quality is proposed by adopting the generally used method in statistical theory for checking the linearity of the data points in two-dimensional space [Hayter 2012]. A linearly proportional indicator can be considered an ideal one. The commonest method to decide how much the data points follow linearity, or close to the line of equality, is to consider the integral of vertical deviations along the interval of interest, as shown in Figure 9-1. The method involves calculating the deviation, or closeness, between the data points and the line of equality as below

$$\delta = \int_{x_1}^{x_2} \epsilon_i \, dx = \int_{x_1}^{x_2} |f(x_i) - x_i| \, dx \qquad (9\text{-}1)$$

where

δ: deviation between the data points and the line of equality,

x_1, x_2: interval of interest,

$f(x_i)$: y_i in a data point which consists of paired observation (x_i, y_i).

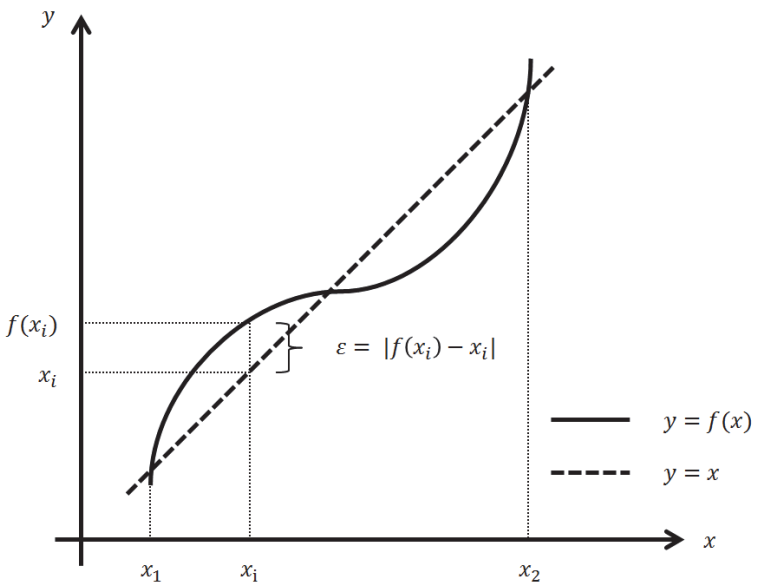

Figure 9-1 Example of Linearity Analysis for the Data Consisting of Paired Observations in Two-dimensional Space

Based on the same principle for checking the linearity of the data points, the indication measure of an attribute for Development Quality or V&V Quality, I, is proposed to identify how much the attribute well indicates the Development Quality or V&V Quality, where I is defined as below

$$I = 1 - \Delta \quad (9\text{-}2)$$

$$\Delta = (P(A_M/Q_H)\,P(A_L/Q_H))P(Q_H) + (P(A_H|Q_M) + P(A_L/Q_M))P(Q_M) + (P(A_H/Q_L) + P(A_M/Q_L))P(Q_L)$$

$$= 1 - P(A_H/Q_H)P(Q_H) - P(A_M/Q_M)P(Q_M) - P(A_L/Q_L)P(Q_L) \quad (9\text{-}3)$$

where

I = indication measure of attribute for Development Quality or V&V Quality,

$P(A_H/Q_H)$ = conditional probability of *High* attribute for *High* Development Quality or V&V Quality,

$P(A_M/Q_H)$ = conditional probability of *Medium* attribute for *High* Development Quality or V&V Quality,

$P(A_L/Q_H)$ = conditional probability of *Low* attribute for *High* Development Quality or V&V Quality,

$P(A_H/Q_M)$ = conditional probability of *High* attribute for *Medium* Development Quality or V&V Quality,

$P(A_M/Q_M)$ = conditional probability of *Medium* attribute for *Medium* Development Quality or V&V Quality,

$P(A_L/Q_M)$ = conditional probability of *Low* attribute for *Medium* Development Quality or V&V Quality,

$P(A_H/Q_L)$ = conditional probability of *High* attribute for *Low* Development Quality or V&V Quality,

$P(A_M/Q_L)$= conditional probability of *Medium* attribute for *Low* Development Quality or V&V Quality,

$P(A_L/Q_L)$= conditional probability of *Low* attribute for *Low* Development Quality or V&V Quality,

$P(Q_H)$ = probability of a *High* Development Quality or V&V Quality,

$P(Q_M)$= probability of a *Medium* Development Quality or V&V Quality, and

$P(Q_L)$ = probability of a *Low* Development Quality or V&V Quality,

Based on the equations 9-2 and 9-3, the attribute that can be considered as a strong indicator for Development Quality or V&V Quality is identified among various attributes defined in five phases. Since indicating performance is a matter of interest, the probability of a Development Quality or V&V Quality to be *High*, *Medium*, *Low* is considered to be one third. Tables 9-1 to Table 9-10 show the conditional probabilities of attributes for a given Development Quality or V&V Quality and the corresponding indication measure, I, which was calculated from the Equations 9-2 and 9-3.

9.4 Development and V&V Quality Contribution Analysis

In this section, the contribution analysis for Development Quality and V&V Quality was performed by directly inserting evidence to Development Quality or V&V Quality. In the analysis, only *High* Development Quality or *High* V&V Quality in each phase is considered because we are focusing on safety-related software. It is not usual to expect that safety-related systems have *Low* Development Quality or *Low* V&V Quality. The result of the model with one *High* Development Quality or *High* V&V Quality is compared to with the case of all *Medium* quality in all phases. Table 9-11 shows the evaluation results when Development Quality and V&V Quality in all phases are *Medium* and the number of FPs is 50.

Table 9-2 Attribute Conditional Probabilities Given Development Quality and Attribute Indication Measure (*I*) in Requirements Phase

Attribute	Development Quality Level			*I*
	High	Medium	Low	
System/Software Qualification Test Plan Generation	0.72	0.65	0.71	0.6910
Development of Software Requirements Specifications	0.72	0.66	0.68	0.6857
Development of a Concept Documentation	0.68	0.64	0.66	0.6587
System/Software Acceptance Test Plan Generation	0.67	0.66	0.65	0.6580
Traceability Analysis - Requirements Phase	0.57	0.66	0.71	0.6440
Security Analysis - Requirements Phase	0.64	0.64	0.63	0.6343
Software Development Planning	0.63	0.64	0.63	0.6337
Criticality Analysis - Requirements Phase	0.64	0.64	0.62	0.6317
Hazard Analysis - Requirements Phase	0.61	0.62	0.63	0.6187
Review and Audit - Requirements Phase	0.59	0.61	0.57	0.5887
Risk Analysis - Requirements Phase	0.54	0.61	0.61	0.5847
Configuration Management - Requirements Phase	0.59	0.58	0.56	0.5760

Table 9-3 Attribute Conditional Probabilities Given Development Quality and Attribute Indication Measure (*I*) in the Design Phase

Attribute	Development Quality Level			*I*
	High	Medium	Low	
Software Component Test Design Generation	0.75	0.70	0.75	0.7333
Software Integration Test Design Generation	0.75	0.69	0.73	0.7203
Development of Software Design Description	0.79	0.63	0.73	0.7153
Software Integration Test Plan Generation	0.71	0.63	0.68	0.6733
Software Qualification Test Design Generation	0.71	0.62	0.68	0.6707
Development of a Software Architecture Description	0.73	0.59	0.70	0.6703
Hazard Analysis - Design Phase	0.65	0.64	0.71	0.6670
Software Acceptance Test Design Generation	0.68	0.65	0.66	0.6637
Criticality Analysis - Design Phase	0.66	0.64	0.68	0.6613
Software Component Test Plan Generation	0.71	0.61	0.65	0.6567

Traceability Analysis - Design Phase	0.62	0.63	0.71	0.6513
Security Analysis - Design Phase	0.63	0.65	0.65	0.6417
Reviews and Audit - Design Phase	0.63	0.60	0.65	0.6253
Risk Analysis - Design Phase	0.62	0.61	0.61	0.6130
Configuration Management - Design Phase	0.61	0.61	0.59	0.6027

Table 9-4 Attribute Conditional Probabilities Given Development Quality and Attribute Indication Measure (*I*) in Implementation Phase

Attribute	Development Quality Level			I
	High	Medium	Low	
Software Component Test Execution	0.82	0.70	0.73	0.7473
Software Integration Test Case Generation	0.75	0.64	0.71	0.7003
Software Qualification Test Case Generation	0.75	0.63	0.71	0.6957
Component Test Case Generation	0.75	0.64	0.68	0.6920
Software Integration Test Procedure Generation	0.75	0.66	0.64	0.6830
Software Component Test Procedure Generation	0.71	0.66	0.68	0.6827
Security Analysis	0.64	0.66	0.73	0.6773
Source Code and Source Code Documentation Generation	0.67	0.65	0.71	0.6750
Hazard Analysis	0.66	0.64	0.72	0.6703
Software Qualification Test Procedure Generation	0.71	0.63	0.66	0.6650
Software Acceptance Test Case Generation	0.69	0.59	0.67	0.6523
Criticality Analysis	0.66	0.62	0.67	0.6487
Traceability Analysis	0.60	0.60	0.68	0.6277
Risk Analysis	0.62	0.59	0.61	0.6037
Reviews & Audits	0.59	0.62	0.59	0.5990
Configuration Management	0.56	0.55	0.56	0.5587

Table 9-5 Attribute Conditional Probabilities Given Development Quality and Attribute Indication Measure (*I*) in Testing Phase

Attribute	Development Quality Level			*I*
	High	Medium	Low	
Software Integration Test Execution	0.79	0.65	0.71	0.7167
Software Acceptance Test Execution	0.71	0.69	0.73	0.7063
Software Qualification Test Execution	0.75	0.64	0.73	0.7060
Software Acceptance Procedure Generation	0.67	0.64	0.71	0.6727
Traceability Analysis - Test Phase	0.63	0.65	0.73	0.6693
Hazard Analysis - Test Phase	0.64	0.64	0.69	0.6567
Security Analysis - Test Phase	0.62	0.62	0.69	0.6433
Risk Analysis - Test Phase	0.67	0.59	0.60	0.6200
Reviews & Audits	0.54	0.61	0.65	0.5970
Configuration Management	0.52	0.55	0.56	0.5447

Table 9-6 Attribute Conditional Probabilities Given Development Quality and Attribute Indication Measure (*I*) in Installation and Checkout Phase

Attribute	Development Quality Level			*I*
	High	Medium	Low	
Installation Procedure Generation	0.85	0.63	0.75	0.7433
Installation and Checkout	0.80	0.65	0.77	0.7400
Hazard Analysis - Installation and Checkout Phase	0.67	0.64	0.77	0.6933
Security Analysis - Installation and Checkout Phase	0.67	0.62	0.75	0.6800
Risk Analysis - Installation and Checkout Phase	0.69	0.54	0.62	0.6167

Table 9-7 Attribute Conditional Probabilities Given V&V Quality and Attribute Indication Measure (I) in Requirements Phase

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
Software Requirements Evaluation	0.73	0.67	0.75	0.7180
V&V System/Software Qualification Test Plan Generation	0.79	0.64	0.71	0.7120
Traceability Analysis V&V - Requirements Phase	0.69	0.67	0.75	0.7043
Hardware/Software/User Requirements Allocation Analysis	0.73	0.66	0.71	0.6993
Concept Documentation Evaluation	0.69	0.67	0.73	0.6960
Interface Analysis V&V - Requirements Phase	0.69	0.69	0.71	0.6953
Software V&V Planning	0.75	0.64	0.68	0.6920
V&V Requirements Phase Activity Summary Report Generation	0.67	0.67	0.65	0.6627
Hazard Analysis V&V - Requirements Phase	0.67	0.64	0.67	0.6590
Security Analysis V&V - Requirements Phase	0.64	0.64	0.67	0.6507
Criticality Analysis V&V - Requirements Phase	0.64	0.63	0.66	0.6430
V&V Software Acceptance Test Plan Generation	0.64	0.64	0.61	0.6310
Reviews and Audit V&V- Requirements Phase	0.59	0.63	0.61	0.6120
Risk Analysis V&V - Requirements Phase	0.59	0.60	0.61	0.6003
Configuration Management Assessment- Requirements Phase	0.55	0.56	0.55	0.5520

Table 9-8 Attribute Conditional Probabilities Given V&V Quality and Attribute Indication Measure (*I*) in Design Phase

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
Design Evaluation	0.82	0.66	0.73	0.7357
Traceability Analysis V&V-Design Phase	0.73	0.70	0.75	0.7277
V&V Software Integration Test Design Generation	0.77	0.66	0.71	0.7130
V&V Software Component Test Design Generation	0.73	0.69	0.67	0.6927
Interface Analysis V&V	0.69	0.69	0.67	0.6817
V&V Software Acceptance Test Design Generation	0.71	0.64	0.67	0.6703
V&V Software Qualification Test Design Generation	0.73	0.61	0.67	0.6687
V&V Design Phase Activity Summary Report Generation	0.64	0.66	0.70	0.6663
V&V Software Integration Test Plan Generation	0.68	0.66	0.64	0.6607
V&V Software Component Test Plan Generation	0.68	0.64	0.64	0.6560
Security Analysis V&V-Design Phase	0.62	0.64	0.68	0.6477
Hazard Analysis V&V-Design Phase	0.62	0.63	0.67	0.6377
Review and Audit- Design Phase	0.66	0.61	0.61	0.6273
Criticality Analysis V&V-Design Phase	0.62	0.61	0.64	0.6243
Risk Analysis V&V- Design Phase	0.59	0.61	0.59	0.5970
Configuration Management V&V- Design Phase	0.55	0.59	0.55	0.5613

Table 9-9 Attribute Conditional Probabilities Given V&V Quality and Attribute Indication Measure (I) in Implementation Phase

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
V&V Software Component Test Execution	0.82	0.70	0.71	0.7417
V&V Software Component Test Case Generation	0.75	0.64	0.67	0.6867
V&V Software Integration Test Case Generation	0.75	0.64	0.67	0.6867
Traceability Analysis V&V	0.69	0.63	0.73	0.6820

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
Source Code and Source Code Documentation Evaluation	0.71	0.61	0.71	0.6743
V&V Software Component Test Procedure Generation	0.71	0.64	0.62	0.6537
V&V Software Integration Test Procedure Generation	0.71	0.64	0.62	0.6537
V&V Software Qualification Test Procedure Generation	0.71	0.64	0.62	0.6537
V&V Software Qualification Test Case Generation	0.71	0.64	0.61	0.6530
Hazard Analysis V&V	0.64	0.64	0.67	0.6507
Security Analysis V&V	0.64	0.64	0.63	0.6367
V&V Implementation Phase Activity Summary Report Generation	0.67	0.63	0.61	0.6347
Review and Audit V&V	0.66	0.61	0.61	0.6273
Criticality Analysis V&V	0.62	0.63	0.60	0.6153
Interface Analysis V&V	0.62	0.59	0.63	0.6117
V&V Software Acceptance Test Case Generation	0.64	0.64	0.56	0.6113
Risk Analysis V&V	0.63	0.59	0.53	0.5863
Configuration Management V&V	0.55	0.57	0.55	0.5543

Table 9-10 Attribute Conditional Probabilities Given V&V Quality and Attribute Indication Measure (I) in Testing Phase

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
V&V Software Integration Test Execution	0.81	0.68	0.71	0.7317
V&V Software Acceptance Test Execution	0.81	0.68	0.71	0.7317
V&V Software Qualification Test Execution	0.77	0.68	0.67	0.7043
Traceability Analysis V&V- Test Phase	0.65	0.67	0.68	0.6680
V&V Software Acceptance Procedure Generation	0.68	0.64	0.63	0.6480
Hazard Analysis V&V- Test Phase	0.64	0.63	0.67	0.6460
Security Analysis V&V- Test Phase	0.64	0.62	0.67	0.6433
V&V Test Phase Activity Summary Report Generation	0.67	0.61	0.58	0.6213

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
Review and Audit V&V - Test Phase	0.63	0.60	0.61	0.6117
Risk Analysis V&V- Test Phase	0.67	0.58	0.56	0.6013
Configuration Management V&V - Test Phase	0.61	0.58	0.52	0.5707

Table 9-11 Attribute Conditional Probabilities Given V&V Quality and Attribute Indication Measure (I) in Installation and Checkout Phase

Attribute	V&V Quality Level			*I*
	High	Medium	Low	
Installation Checkout V&V	0.80	0.65	0.75	0.7333
Hazard Analysis V&V - Installation and Checkout Phase	0.67	0.62	0.72	0.6700
Security Analysis V&V - Installation and Checkout Phase	0.67	0.62	0.72	0.6700
Installation Configuration Audit V&V	0.70	0.61	0.70	0.6700
V&V Installation and Checkout Phase Activity Summary Report Generation	0.70	0.61	0.65	0.6533
V&V Final Report Generation	0.70	0.61	0.65	0.6533
Risk Analysis V&V - Installation and Checkout Phase	0.72	0.52	0.59	0.6100

Table 9-12 BBN Model Parameters for all *Medium* Development Quality and V&V Quality

Phase	Defects introduced in the current phase		Detection probability for defects passed from the previous phase		Detection probability for defects introduced in the current phase		Detected defects passed from the previous phase		Detected defects introduced in the current phase		Defect density		Defects remaining	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Requirements	19.71	35.9	-	-	0.79	0.16	-	-	15.64	29.28	0.39	0.72	4.07	10.15
Design	42.61	52.56	0.46	0.26	0.79	0.17	1.86	5.40	33.82	43.23	0.85	1.05	11.00	17.05
Implementation	49.45	56.96	0.48	0.25	0.84	0.15	5.25	9.66	41.49	49.03	0.99	1.14	13.71	17.99
Test	19.88	35.25	0.70	0.16	0.73	0.14	9.61	13.16	14.54	26.42	0.40	0.70	9.45	13.08
Installation/ Checkout	12.63	29.35	0.70	0.19	0.80	0.14	6.64	9.77	10.12	23.88	0.25	0.59	5.32	9.12

Table 9-13 shows the defect density and the number of defects introduced in each phase when Development Quality is changed from Medium to *High*. For instance, as shown in, when Development Quality in the requirements phase is *Medium*, the estimated numbers of defects introduced in the requirements phase is 19.71. If it is changed to *High*, the estimated number of defects decreases to 15.49 as shown in Table 9-13. It is observable that High Development Quality considerably decreases the defects introduced by more than 20% in all phases.

Since Development Quality in the test phase is connected to not only the defect density node but also the defect detection probability nodes, the change of Development Quality affects the detect detection probabilities as shown in Table 9-14. High Development Quality of the test phase increases the defect detection probability for the defects introduced in the current phase by 13.53% and that for the defects passed from the previous phase by 13.53%.

Table 9-15 shows the detection probabilities for the defects introduced in the current phase and passed from the previous phase in each phase when V&V Quality is changed from Medium to High. For instance, in, when V&V Quality in the requirements phase is Medium, the estimated detection probability for the defects introduced in the current phase is 0.79. If it changes to High, the detection probability increases to 0.86. High V&V Quality increases the defect detection probability for defects introduced in each phase by from 7.61% to 18.36%. In the case of the defect detection probability for defects passed from the previous phase, it increases by from 23.64 to 31.40%.

High Development Quality and *High* V&V Quality decrease the number of defects remaining in the current phase by decreasing the defect density and increasing the defect detection probabilities as shown in Table 9 13 to Table 9 15. However, it was observed that High Development Quality and High V&V Quality in each phase have a different contribution on the final number of remaining defects. As shown in, for instance, while High Development Quality in the requirements phase decreases the final number of defects by 0.04%, that in the installation/checkout phase lowers the final number of defects by 14.94%. Since the defects introduced in early phases could be detected throughout the later phases, Development Quality and V&V Quality in early phases have relatively small contributions to the final number of defects. On the other hand, Development Quality and V&V Quality in the last phase, the installation/checkout phase, have much greater effect on the final result.

The recursive process in SDLC is not considered explicitly in the developed model for estimating the total defects remaining in the software. In practice, if a significant defect made in the design phase is found in the installation/checkout phase, the development process should be restarted again from the design phase.

High Development Quality and *High* V&V Quality in the first phase decrease the number of initial defects, and reduce the defects in following phases. Table 9 17 shows the changes in detected defects passed from the previous phases by High Development Quality and High V&V Quality. ΔMean implies the change from the case of all Medium quality. High Development Quality and High V&V Quality in only the installation/checkout phase increase the detected defects passed from the previous phase.

Table 9-13 Defect Density and Number of Defects Introduced in Each Phase with *High* Development Quality

Phase	Defect density per function point				Defects introduced in the current phase			
	Mean	SD	△Mean	△SD	Mean	SD	△Mean	△SD
Requirement	0.31	0.63	-20.54%	-12.08%	15.49	31.65	-21.41%	-11.84%
Design	0.66	0.93	-21.96%	-11.48%	33.17	46.47	-22.15%	-11.59%
Implementation	0.79	1.02	-20.65%	-10.88%	39.28	50.78	-20.57%	-10.85%
Test	0.29	0.60	-27.58%	-14.36%	14.49	29.98	-27.11%	-14.95%
Installation/Checkout	0.18	0.48	-28.60%	-18.53%	8.924	24.04	-29.34%	-18.09%

Table 9-14 Probabilities of Defect Detection in the Test Phase with *High* Development Quality

Defect detection probability for defects introduced in the current phase				Defect detection probability for defects passed from the previous phase			
Mean	SD	△Mean	△SD	Mean	SD	△Mean	△SD
0.83	0.10	13.53%	-31.89%	0.79	0.11	13.53%	-32.69%

Table 9-15 Defect Detection Probabilities in Each Phase with *High* V&V Quality

Phase	Defect detection probability for defects introduced in the current phase				Defect detection probability for defects passed from the previous phase			
	Mean	SD	△Mean	△SD	Mean	SD	△Mean	△SD
Requirement	0.86	0.15	9.34%	-7.06%	-	-	-	-
Design	0.86	0.16	9.44%	-8.41%	0.57	0.29	24.67%	11.50%
Implementation	0.90	0.13	7.61%	-14.47%	0.63	0.25	31.40%	1.64%
Test	0.86	0.07	18.36%	-51.44%	0.87	0.07	23.64%	-58.19%
Installation/Checkout	0.92	0.07	15.33%	-49.45%	0.91	0.07	29.37%	-62.09%

Table 9-16 Number of Defects Remaining in the Current Phase and the Final Number of Defects Remaining

Phase	Condition	Number of defects remaining in the current phase		Final number of defects remaining	
		Mean	△Mean	Mean	△Mean
Requirement	Development Quality = High	3.21	-21.23%	5.32	-0.04%
	V&V Quality = High	2.60	-36.19%	5.33	0.11%
Design	Development Quality = High	8.92	-18.88%	5.19	-2.52%
	V&V Quality = High	7.46	-32.23%	5.18	-2.59%
Implementation	Development Quality = High	11.99	-12.55%	5.15	-3.20%
	V&V Quality = High	8.82	-35.66%	4.86	-8.74%
Test	Development Quality = High	5.27	-44.19%	4.10	-22.91%
	V&V Quality = High	4.64	-50.92%	3.81	-28.36%
Installation/ Checkout	Development Quality = High	4.53	-14.94%	4.53	-14.94%
	V&V Quality = High	1.88	-64.62%	1.88	-64.62%

Table 9-17 Number of Detected Defects Passed from Previous Phase

Phase	Condition	Number of detected defects passed from previous phase							
		In design phase		In implementation phase		In test phase		In installation/checkout phase	
		Mean	△Mean	Mean	△Mean	Mean	△Mean	Mean	△Mean
Requirement	Development Quality = High	1.46	-21.56%	5.05	-3.87%	9.50	-1.19%	6.57	-1.11%
	V&V Quality = High	1.18	-36.56%	4.83	-7.94%	9.37	-2.49%	6.57	-1.01%
Design	Development Quality = High	-	-	4.30	-18.17%	8.87	-7.71%	6.43	-3.21%
	V&V Quality = High	2.28	22.58%	3.55	-32.40%	8.24	-14.29%	6.30	-5.14%
Implementation	Development Quality = High	-	-	-	-	8.47	-11.85%	6.22	-6.39%
	V&V Quality = High	-	-	6.79	29.31%	6.23	-35.16%	5.61	-15.53%
Test	Development Quality = High	-	-	-	-	10.81	12.49%	3.69	-44.43%
	V&V Quality = High	-	-	-	-	11.90	23.83%	3.25	-51.08%
Installation/ Checkout	Development Quality = High	-	-	-	-	-	-	-	-
	V&V Quality = High	-	-	-	-	-	-	8.64	30.12%

The cost of fixing a detected defect depends on the phase in which the defect is found. Usually a defect detected in later phases costs more to fix because of the recursive fixing process. Assuming the cost to fix one detected defect passed from the previous phase is *C, 2C, 3C,* and *4C*, in the design phase, implementation phase, test phase, and installation/checkout phase, respectively, Table 9-18 shows the total cost required for the recursive fixing process when one Development Quality or V&V Quality change from *Medium* to *High*. The total cost is

The total cost of fixing detected defects = Δ Mean in the design phase × *C*
+ Δ Mean in the implementation phase × 2*C*
+ Δ Mean in the test phase × 3*C*
+ Δ Mean in the installation/checkout phase × 4*C*

The enhancement in quality reduces the recursive fixing cost by reducing defects. *High* Quality in the implementation phase entails the most reduction, and the installation/checkout phase incurs additional cost for the recursive fixing process.

Table 9-18 Cost of Fixing Detected Defects

Phase	Condition	Cost
Requirement	Development Quality = High	-0.37
	V&V Quality = High	-0.63
Design	Development Quality = High	-0.72
	V&V Quality = High	-1.03
Implementation	Development Quality = High	-0.6
	V&V Quality = High	-1.11
Test	Development Quality = High	-1.4
	V&V Quality = High	-1.32
Installation/ Checkout	Development Quality = High	0
	V&V Quality = High	1.2

9.5 Sensitivity Analysis for Diversity in the Expert's Opinion

This section discusses the sensitivity analyses undertaken to observe the effects of diverse opinions in the second round of expert elicitation. As mentioned above, diverse estimations were given by the experts, especially in estimating defect density and detection probability for those defects transferred from the previous phase. Due to the experts' different backgrounds and experience of the experts, they could have different opinions on the same question. This diversity caused huge deviations in the application results in Chapter 9 because all diverse estimations were considered in developing the BBN model. In the sensitivity analysis, demonstrations were undertaken to observe the effects of elimination of those expert estimations that differ significantly from the other estimations.

The sensitivity studies were performed for the conditional probabilities of the following attributes; defect density, and defect detection probabilities for the defects introduced in the current phase and the defects passed from the previous phase(s). The sensitivity analysis results were compared with the results of the base case in which *Development Quality* and *V&V Quality* are all "Medium" and the number of FPs is 50.

9.5.1 Attributes

In the second expert-opinion elicitation, the experts' opinions on the conditional probabilities of an attribute for a given Development Quality or V&V Quality were estimated by point values. Two significantly different (furthest from the average) estimations were removed.

As shown in Table 9-19, slight changes in the result were caused by the exclusion of the two most different estimations. The number of defects introduced in the current phase changed by less than 1.83%. The probability of detecting the defects introduced in the current phase and those that passed from the previous phase changed, respectively, by less than 0.70% and 0.64% respectively. This is because the experts' answers for the attribute conditional probabilities were relatively consistent, and the diverse opinions in attributes do not have much effect on the result.

9.5.2 Defect Density

The experts' opinions for defect density were given as the estimated 5th, 50th, and 95th percentiles. Significant diversity was observed in the estimations of defect density. For instance, in defect density for the requirements phase, the 50th percentile estimations from experts were 0.012, 0.1, 0.7, 1.0, and 4.0 for a given *Medium* Development Quality.

Distributions of the smaller variance were selected as the experts' opinions, whose 50th percentile is the furthest from the average 50th percentile from all experts. Monte Carlo simulation was used to generate an empirical distribution by sampling from the estimated normal distributions while excluding the two most different estimations for each specified Development Quality. As shown in Table D-15, Gamma distributions were able to well express the empirical distribution compared to other distributions, showing the lowest AIC and BIC values for 9 out of 15 data sets. Based on the scale and shape parameters of the fitted Gamma distribution for defect density, literature data on US software developments and limited V&V and testing results for the two trial software, as described in Section 7.2.2., were used to Bayesian update the defect density node and as input data for the sensitivity study, as shown in Table 9-20.

Defect density, having more diverse estimations, showed a larger effect from excluding the significantly different estimations. Except in the implementation phase, considerable differences were observed in overall phases. Especially, as shown in Table 9-21, in the test and installation/checkout phases, the mean decreased by 21.33% and 59.18%, and the standard deviation decreased by 12.26% and 35.78%, respectively.

Table 9-19 Number of Defects Introduced in Current Phase and Defect Detecting Probabilities Excluding the Two Most Significantly Different Opinions for Attributes

Phase	The number of defects introduced in the current phase				Detection probability for defects introduced in the current phase				Detection probability for defects passed from the previous phase			
	Mean	SD	ΔMean	ΔSD	Mean	SD	ΔMean	ΔSD	Mean	SD	ΔMean	ΔSD
Requirement	19.35	34.93	-1.83%	-2.70%	0.80	0.16	0.70%	1.56%				
Design	42.36	52.30	-0.59%	-0.49%	0.79	0.17	0.38%	-2.18%	0.46	0.26	-0.20%	0.50%
Implementation	50.09	57.44	0.28%	0.84%	0.84	0.15	0.11%	-2.13%	0.48	0.25	-0.31%	0.68%
Test	20.10	35.50	1.11%	0.71%	0.73	0.14	-0.03%	0.43%	0.70	0.16	0.64%	-1.13%
Installation/ checkout	12.81	29.97	1.43%	2.11%	0.80	0.14	0.46%	-2.00%	0.70	0.19	-0.21%	2.21%

Table 9-20 Gamma Distribution Fit for Defect Density NPT Excluding Two Most Significantly Different Opinions

Phase	Development Quality	Scale	Shape
Requirement	High	1.8677	0.1393
	Medium	1.8668	0.2215
	Low	1.8888	0.2883
Design	High	2.4569	0.2629
	Medium	3.0176	0.2855
	Low	8.5872	0.2383
Implementation	High	2.4701	0.2607
	Medium	3.3981	0.3042
	Low	3.6047	0.3152
Test	High	1.1090	0.1740
	Medium	1.5202	0.2027
	Low	2.0218	0.2287
Installation and Checkout	High	0.3933	0.3113
	Medium	0.2544	0.3931
	Low	0.3121	0.4312

Table 9-21 Number of Defects Introduced in the Current Phase When Excluding the Two Most Significantly Different Opinions for Defect Density

Phase	Defects introduced in the current phase			
	Mean	SD	△Mean%	△SD%
Requirement	20.54	35.27	4.21%	-1.75%
Design	43.45	53.14	1.97%	1.10%
Implementation	51.43	58.09	2.96%	1.98%
Test	15.64	30.93	-21.33%	-12.26%
Installation/Checkout	5.155	18.85	-59.18%	-35.78%

9.5.3 Detection Probability for the Defects Introduced in the Current Phase

The experts' opinions for probability of defect detection were given as the estimated 5th, 50th, and 95th percentiles. The same approach was used to selecting the most significantly different estimations of defects density for obtaining the probability of detecting the defects in the current phase. Monte Carlo simulation was employed to generate an empirical distribution by sampling from the estimated normal distributions while excluding the two most significantly different estimations for each specified V&V Quality and complexity. As shown in Table D-16, Beta distribution were able to well express the empirical distribution compared to other distributions, showing the lowest AIC and BIC values for 19 out of 36 data sets. Based on the first shape parameter (α) and second shape parameter (β) of the fitted Beta distribution for defect detection probability at current phase, literature data on US software developments, described in Section 7.2.2.1, were used to Bayesian update the defect detection probability at current phase node, and were used as input data to specify the distribution of each node's probability for the sensitivity study, as shown in Table 9-22. The estimations for the detection probability for the defects introduced in the current phase showed a relatively high consistency. As shown in Table 9-23, the

detection probabilities were changed by -1.13%, 1.85%, -1.42%, 1.33% and 1.29% respectively, in the requirement, design, implementation, test, and installation/checkout phases.

Table 9-22 Beta Distribution for Defect Detection Probability NPT in Current Phase by Excluding the Two Most Significantly Different Opinions

Phase	Develop-ment quality	V&V quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Requirement	-	High	5.356	2.670	5.553	1.581	2.930	0.371
		Medium	4.737	2.546	4.371	1.620	3.761	1.057
		Low	4.155	3.522	3.277	3.118	3.010	2.444
Design	-	High	4.262	1.967	3.920	1.241	3.484	0.313
		Medium	5.163	3.329	4.432	1.950	5.358	1.301
		Low	5.002	4.703	4.378	4.686	3.975	3.576
Implementation	-	High	9.192	2.834	9.328	1.484	6.990	0.606
		Medium	7.472	3.845	7.033	2.286	8.852	1.829
		Low	3.991	4.453	3.705	3.154	3.876	2.897
Test	High	High	163.866	37.308	74.432	7.806	55.567	4.642
		Medium	47.402	22.399	34.856	11.334	27.932	7.295
		Low	20.294	14.107	24.102	11.979	15.609	7.730
	Medium	High	31.540	9.453	22.618	3.977	21.408	2.724
		Medium	47.473	27.421	37.596	15.793	32.325	11.380
		Low	40.336	40.477	35.888	28.818	33.617	24.068
	Low	High	34.773	13.243	23.753	5.810	21.080	4.005
		Medium	46.181	32.287	38.474	20.359	200.122	91.145
		Low	36.283	43.694	33.106	32.559	31.363	27.883
Installation and Checkout	-	High	28.558	10.411	53.757	6.962	21.491	1.298
		Medium	18.129	9.602	22.300	7.196	18.508	4.353
		Low	6.598	8.653	5.668	6.254	5.788	5.168

Table 9-23 Defect-detection Probability for the Defects Introduced in the Current Phase When Excluding the Two Most Significantly Different Opinions

Phase	Detection probability for defects introduced in the current phase			
	Mean	SD	△Mean%	△SD%
Requirement	0.78	0.17	-1.13%	7.25%
Design	0.80	0.14	1.85%	-15.65%
Implementation	0.83	0.11	-1.42%	-26.53%
Test	0.74	0.07	1.33%	-53.17%
Installation/Checkout	0.81	0.08	1.29%	-42.53%

9.5.4 Detection Probability for the Defects Passed from the Previous Phase

Considering the probability of defect detection for the previous phase, the differences between the experts' opinions were higher compared to those for the defects introduced in the current phase. In the estimations of the detection probability for the defects passed from the previous phase, bipolar responses were observed in some phases. For instance, for *High* V&V Quality in the requirements phase, three experts provided high detection probabilities (0.85, 0.8, and 0.75), and two experts provided low detection probabilities (0.3 and 0.12). Therefore, two sensitivity studies were conducted for obtaining this detection probability.

In the first approach, the two most significantly different estimations were excluded as in other sensitivity studies, in which the furthest estimations from the average were eliminated. Monte Carlo simulation was used to generate an empirical distribution by sampling from the estimated normal distributions, while excluding the two most significantly different estimations for each specified V&V Quality and complexity. As shown in Table D-17, Beta distribution was able to well express the empirical distribution compared to other distributions, showing the lowest AIC and BIC values for 13 out of 27 data-sets. The first shape parameter (α) and second shape parameter (β) of the fitted Beta distribution for each node, as shown in Table 9-24, were used as input data to specify the distribution of each node's probabilities for the sensitivity study.

Table 9-24 Beta Distribution for Detection Probability NPT of Defects in the Previous Phase, After Excluding the Two Most Significantly Different Opinions

Phase	Develop-ment quality	V&V quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Design	-	High	3.321	3.981	2.523	1.605	1.684	0.803
		Medium	1.938	4.013	2.280	3.163	2.013	2.118
		Low	1.553	9.759	2.339	11.618	3.375	10.711
Implementation	-	High	4.741	5.106	6.366	3.676	2.781	1.024
		Medium	2.332	4.636	3.883	5.443	3.614	3.257
		Low	1.657	6.836	2.640	8.661	4.698	11.945
Test	High	High	167.388	39.257	78.087	8.672	17.105	1.801
		Medium	47.777	23.008	22.539	3.427	32.734	10.044
		Low	20.304	14.403	16.578	8.913	15.707	7.968
	Medium	High	186.031	57.145	114.714	20.238	72.603	8.697
		Medium	46.359	32.891	38.616	20.786	34.719	16.098
		Low	40.376	41.205	35.909	29.377	33.771	24.653
	Low	High	197.445	76.785	138.496	34.610	104.930	19.978
		Medium	43.619	37.150	37.890	25.266	34.897	20.494
		Low	36.286	44.365	33.028	33.029	31.407	28.420
Installation and Checkout	-	High	68.317	24.009	41.222	8.748	18.778	1.861
		Medium	57.820	33.963	39.257	16.818	27.936	9.047
		Low	4.542	6.686	4.095	5.015	2.082	1.328

In the second approach, bipolar estimations were categorized into high-group and low-group, with K-Means clustering, which is a partitioning method that treats observations in the data as objects having locations and distances from each other. It partitions the objects into the *K* number of mutually exclusive clusters, such that the objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible [MacQueen 1967]. In this study, K-Means clustering was used to distinguish two groups of high- and low-defect detection probabilities using an iterative algorithm that assigns objects to clusters such that the sum of the distances from each object to its cluster centroid, over all clusters, is a minimum value. The first shape parameter (α), and the second shape parameter (β) of the fitted Beta distribution for each node based on partitioning results are reported in and Table 9-26 were used as input data to specify the distribution of each node's probability for the sensitivity study.

Table 9-25 Beta Distribution Fit for the NPT of High Probability of Defect Detection for the Expert Group in Previous Phase

Phase	Develop-ment quality	V&V quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Design	-	High	28.402	11.064	21.948	5.503	9.105	1.463
		Medium	28.001	18.295	22.001	10.596	25.704	9.705
		Low	17.068	37.160	15.360	28.588	16.428	27.440
Implemen-tation	-	High	28.363	11.051	16.100	4.902	9.102	1.462
		Medium	30.746	21.809	3.434	2.065	14.234	6.076
		Low	8.119	16.442	7.408	12.750	7.569	11.967
Test	High	High	167.01	39.174	77.970	8.661	44.065	2.707
		Medium	20.928	5.791	14.148	3.332	14.156	1.749
		Low	17.618	5.404	12.700	2.393	12.195	1.817
	Medium	High	35.319	10.299	24.779	4.181	24.781	3.012
		Medium	17.665	5.418	12.708	2.394	12.196	1.816
		Low	19.436	7.543	13.367	3.385	11.842	2.403
	Low	High	26.832	9.279	19.475	4.248	19.624	3.338
		Medium	19.499	7.566	13.364	3.383	11.844	2.401
		Low	20.561	9.883	14.637	4.873	12.386	3.368
Installation and Checkout	-	High	31.068	8.932	14.918	2.288	44.505	2.217
		Medium	14.880	6.663	10.904	3.289	10.988	2.561
		Low	10.588	5.703	7.804	2.956	8.050	2.678

Table 9-26 Beta Distribution Fit for the NPT of Low-defect Detection Probability Expert Group in Previous Phase

Phase	Development quality	V&V quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Design	-	High	1.999	6.116	3.183	12.045	6.925	20.326
		Medium	1.890	11.732	1.417	4.998	1.102	2.399
		Low	3.519	54.938	5.590	58.907	5.562	42.075
Implemen-tation	-	High	2.856	7.130	2.895	6.365	6.390	10.527
		Medium	1.319	6.508	1.514	4.991	1.193	2.178
		Low	3.609	74.137	2.866	31.790	2.244	12.158
Test	High	High	29.034	7.615	17.822	2.569	27.902	2.705
		Medium	40.768	21.710	20.557	13.699	30.967	10.313
		Low	24.672	19.173	20.357	12.190	20.375	11.121
	Medium	High	197.747	76.913	138.697	34.669	123.340	27.810
		Medium	33.584	28.584	28.470	18.950	21.877	13.263
		Low	28.443	34.739	25.519	25.485	20.808	19.294
	Low	High	194.429	137.947	163.167	87.854	152.868	77.704
		Medium	31.326	31.934	27.419	22.403	21.844	16.446
		Low	25.158	36.934	23.037	28.121	19.286	22.022
Installation and Checkout	-	High	69.321	161.749	39.184	26.118	15.807	2.611
		Medium	35.697	238.901	38.071	113.273	35.252	57.519
		Low	0.911	5.099	1.142	5.333	1.470	5.592

While a slight change was observed in the first approach, significant changes were seen in the group approach. As depicted in, the probability of defect detection for defects passed from the previous phase is changed by 6.17%, 9.35%, -2.40%, and 7.97% in the design, implementation, test, and installation/checkout phase, respectively. In contrast, by dividing the estimations of the high- and low-detection probability group, the probability of detecting defects passed from the previous phase considerably increased or decreased in the design, implementation, and installation/checkout phase because of the bipolar estimations from experts, as shown in Table 9-28 and Table 9-29. Only in the test phase were the results almost the same.

Table 9-27 Defect-detection Probability for the Defects Passed From the Previous Phase When the Two Most Significantly Different Opinions are Excluded

Phase	Detection probability for defects passed from the previous phase			
	Mean	σ	△Mean	△σ
Design	0.49	0.22	6.17%	-15.08%
Implementation	0.52	0.18	9.35%	-29.00%
Test	0.68	0.06	-2.40%	-59.49%
Installation/Checkout	0.76	0.07	7.97%	-62.94%

Table 9-28 Defect-detection Probability for the Defects Passed From the Previous Phase of High-group

Phase	Detection probability for defects passed from the previous phase			
	Mean	σ	△Mean	△σ
Design	0.73	0.07	57.61%	-71.60%
Implementation	0.70	0.10	46.06%	-60.36%
Test	0.87	0.09	24.37%	-45.98%
Installation/Checkout	0.81	0.10	15.86%	-45.95%

Table 9-29 Defect-detection Probability for the Defects Passed From the Previous Phase of Low-group

Phase	Detection probability for defects passed from the previous phase			
	Mean	σ	△Mean	△σ
Design	0.31	0.22	-31.54%	-15.81%
Implementation	0.35	0.23	-26.33%	-8.80%
Test	0.62	0.08	-11.00%	-49.58%
Installation/Checkout	0.38	0.05	-45.61%	-71.55%

9.5.5 Significantly Different Estimations Exclusion in the Applications

The sensitivity analysis for the LOCS and IDiPS-RPS applications were performed, excluding the most significantly different estimations, as mentioned the previous sections. Two most different estimations in attributes, defect density, detection probability for the current phase, and detection probability for the previous phase were excluded simultaneously.

The analysis result of the LOCS application is shown in from Table 9-30 to Table 9-33. The defects introduced in each phase by 6.87%, 14.52%, 6.49%, -22.32% and -59.91% in the requirement, design, implementation, testing, and installation/checkout phase respectively, as shown from Table 9-30 to Table 9-34. The number of introduced defects decreased in the test and installation/checkout phases.

Table 9-30 Defects Introduced in the Current Phase of LOCS Application When the Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	17.11	29.24	6.87%	-0.44%
Design	40.53	48.79	14.52%	11.62%
Implementation	45.15	48.96	6.49%	2.36%
Test	12.74	24.99	-22.32%	-14.36%
Installation/Checkout	4.117	15.29	-59.91%	-35.86%

The defect detection probability for defects introduced in each phase decreased in all phases except the test and installation/checkout phases, as shown in Table 9-31.

Table 9-31 Detection Probability for Defects Introduced in the Current Phase of LOCS Application When Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	0.78	0.17	-1.81%	5.25%
Design	0.53	0.17	-6.67%	-22.04%
Implementation	0.58	0.18	-5.30%	-26.05%
Test	0.73	0.07	1.05%	-50.09%
Installation/Checkout	0.81	0.08	0.76%	-40.61%

Table 9-32 shows the defect detection probability for defects passed from the previous phase. The detection probability decreased in all phases except the design and installation/checkout phases. The standard deviations in both probabilities were considerably reduced compared to the means.

Table 9-32 Detection Probability for Defects Passed from the Previous Phase of LOCS Application When the Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Design	0.24	0.12	8.89%	-16.34%
Implementation	0.29	0.12	-3.48%	-29.11%
Test	0.68	0.07	-3.33%	-57.15%
Installation/Checkout	0.75	0.07	7.89%	-63.17%

Table 9-33 shows the defects remaining in each phase. The estimated final number of defects of LOCS decreased from 6.02 to 4.32, and its standard deviation also decreased from 8.60 to 5.01.

Table 9-33 Defects Remaining in Each Phase of LOCS Application When Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	3.77	8.86	15.71%	8.46%
Design	21.87	26.37	22.52%	10.20%
Implementation	34.50	30.89	19.13%	2.45%
Test	14.47	12.92	10.12%	-13.06%
Installation/Checkout	4.32	5.01	-27.87%	-41.47%

The analysis result of IDiPS-RPS is shown in Table 9-34 to Table 9-37. The defects introduced in each phase by 5.28%, 0.53%, 3.58%, -22.24%, and -60.17% in the requirement, design, implementation, test, and installation/checkout phase respectively, as shown in Table 9-34.

Table 9-34 Defects Introduced in the Current Phase of IDiPS-RPS Application When Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	22.93	39.53	5.28%	-1.22%
Design	47.56	58.36	0.53%	-0.38%
Implementation	57.30	65.27	3.58%	2.48%
Test	17.45	34.54	-22.24%	-12.84%
Installation/Checkout	5.65	21.10	-60.71%	-36.60%

The defect detection probability for defects introduced in each phase increased in all phases except the design, implementation, and test phases as shown in Table 9-35.

Table 9-35 Detection Probability for Defects Introduced in the Current Phase of IDiPS-RPS Application When Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	0.61	0.21	1.43%	-12.87%
Design	0.53	0.17	-12.48%	-30.03%
Implementation	0.57	0.18	-5.33%	-27.70%
Test	0.58	0.06	-8.54%	-60.19%
Installation/Checkout	0.81	0.08	0.66%	-40.73%

Table 9-36 shows the defect-detection probability for defects passed from the previous phase. The detection probability decreased in two phases and increased in two phases. While the change in detection probability was less than 10%, the standard deviation was considerably reduced by eliminating estimations that were too diverse.

Table 9-36 Detection Probability for Defects Passed from the Previous Phase of IDiPS-RPS Application When the Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Design	0.24	0.11	9.20%	-17.53%
Implementation	0.28	0.11	-3.96%	-32.49%
Test	0.58	0.06	-8.04%	-64.04%
Installation/Checkout	0.76	0.07	7.99%	-64.25%

Table 9-37 shows the defects remaining in each phase. The estimated final number of defects of this system decreased from 9.89 to 7.53, and its standard deviation also dropped from 13.19 to 7.92.

Table 9-37 Defects Remaining in Each Phase of IDiPS-RPS Application When the Most Significantly Different Opinions are Excluded

Phase	Mean	σ	△Mean	△σ
Requirement	8.96	18.39	3.31%	-6.70%
Design	29.22	33.99	6.56%	-1.16%
Implementation	45.49	40.54	10.01%	-2.67%
Test	26.53	23.36	12.65%	-7.01%
Installation/Checkout	7.53	7.92	-24.29%	-40.54%

In this study, sensitivity analysis was performed to observe the effect of diverse opinions based on the second round of elicitation provided by seven experts. In summary, the standard deviation for the number of defects remaining in each phase was reduced by excluding the most significantly different estimations. By increasing the number of expert elicitations, the uncertainty associated with the diverse opinion is expected to decrease. In addition, the standard deviation for the number of defects remaining in each phase is expected to decrease considerably.

10 SUMMARY AND CONCLUSIONS

A BBN model was developed to estimate the number of defects in nuclear safety-related software and the resulting probability of software failure on demand. The model captures NPP safety-related SDLC activity quality indicators and product information, establishes the quantitative causal relationships between these indicators and the number of remaining defects, and further estimates software failure probability. The quantitative parameters in this model are estimated using expert opinions from NPP safety software experts. Literature data on US software developments and limited V&V and testing results for the two trial software were used to Bayesian update the expert opinions. This model was then applied to two trial software packages: INL ATR LOCS and KAERI IDiPS-RPS.

Insights into several aspects of BBN construction have been gained, including attribute identification, causal relationship establishment and quantification, parameter Bayesian updating, and expert opinion elicitation and aggregation. Due to the proprietary nature of the two trial software packages and their development processes, the number of defects remaining and failure probabilities reported in this report are for demonstration purpose only. Use of these results to evaluate the quality and associated safety of these two software packages is not recommended.

10.1 Accomplishments

10.1.1 BBN Attributes

The objective of this research is to estimate the number of residual defects and the software failure probability from SDLC characteristics and software product characteristics using a BBN model. Therefore, this model could be used before the software is deployed. These characteristics are also termed as “attributes” or “indicator nodes” in this report. A complete and independent set of attributes is required to ensure that the BBN model captures all software development perspectives and potential causal relationship in terms of the number residual defects.

The identification of this “ideal” set is challenging as:

- A standard SDLC followed by all NPP safety-related software vendors does not exist.
    - The failure mechanism of the SDLC (i.e., how development activities fail and how defects are introduced, detected and removed) is not fully understood.

A large number of SDLC standards were examined in this study to identify the ideal set of attributes. The activities and associated characteristics listed in IEEE Standard 1012 were selected. Given an SDLC phase, the intermediate nodes of “Development Quality” and “V&V Quality”, which are complete and independent by definition, were identified and assumed to be directly related to the number of defects introduced, detected and removed. A number of attributes representing activities identified in IEEE 1012 (Refer to Chapter 5 for details) was selected to logically contribute to the “Development Quality” and “V&V Quality” for each SDLC phase. Although no missing attributes were identified by experts, the completeness and independency of this set of attributes were not fully justified in this study due to the limits of its scope (demonstration purpose) and schedule.

Further study of a complete and independent set of attributes that represent SDLC characteristics with respect to estimating the number of defects is recommended.

10.1.2 NPT Quantification

One of the biggest challenges of this study is obtaining the data required to construct/quantify NPTs. Although the BBN technique is capable of integrating different types of information (e.g., quantitative vs qualitative, continuous vs discrete), the large amount of data required for the BBN frequently prohibited BBNs from being used in practical applications.

When confronted with a dearth of data for conducting analyses, the traditional solution in the nuclear industry is to use expert elicitation. Although this approach was very successful in the past for some NRC studies (e.g., seismic studies), it meets with skepticism when used in safety-related software reliability studies. This is can be due to 1) the lack of perceived credibility of the chosen experts and their inputs, and to 2) the ineffective presentation of the experts' opinions.

10.1.2.1 Multi-Rounds Expert Elicitation

Data from experts who are familiar with nuclear safety-related software development and are knowledgeable about its quality assurance and operating experience are rarely available in the literature, which is due mainly to proprietary limits of nuclear industry.

Though the attributes and the qualitative causal relationships between attributes and the number of defects (the BBN structure or topology) might be generic across industries, the quantitative cause relationships (NPTs) had to be nuclear industry-specific. Based on this requirement, a multi-round elicitation approach was used.

The first round used "generic" experts to finalize the selection of attributes and the causal network. Experts in software development and management and in software reliability engineering were selected from across industries and academia. The second round used nuclear industry experts having experience in safety-related software development, V&V, and management to quantify NPTs. It was assumed for this study that using a multi-round expert elicitation helps produce a less biased BBN structure and credible, nuclear-specific NPTs.

10.1.2.2 Expert Opinion Aggregation

Experts were requested to estimate NPTs in the form of distributions. Their answers were expressed as distributions as opposed to point estimates in order to account for uncertainties in the experts' opinions. Some[1] of the experts' answers were then aggregated into a distribution again using Monte Carlo simulation to capture the uncertainty of NPT quantification.

10.1.2.3 NPT Quantification and Bayesian Update

In this study, a framework was developed to effectively integrate expert opinions and other sources of evidence for NPT quantification. The NPTs were quantified with the expert opinions expressed as distributions for effective accommodation of the uncertainty of expert opinions. Then, by using literature and operating experience, some key parameters in the NPTs were Bayesian updated. This demonstrates a framework that can effectively and systematically

[1] In Table 7-1, only the last 3 rows of experts inputs were Monte Carlo simulated.

integrate different kinds of available source information to quantify BBN NPTs. As more evidence and observations become available in the future, key parameters in the NPTs can be updated to further reduce NPT uncertainties.

10.1.2.4 Sensitivity Analysis

Contribution analyses were performed for the attributes and Development/V&V Quality. Each attribute which links Development Quality or V&V Quality has different contributions. In this study, an indication measure was proposed to identify the relative contribution of an attribute to the Development Quality or V&V Quality in each phase.

Since SDLC phases have different contributions to the final number of residual defects, the contribution of Development Quality or V&V Quality in each phase was analyzed. Results showed that relatively fewer residual defects were expected by improving the quality in the test phase and installation/checkout phase.

Such contribution analyses and results can be used to align more efforts towards the activities (nodes) with higher contributions to optimize the SDLC with respect to targeting fewer residual defects.

In consideration of the variety of experts' specialties, the outlier treatment was investigated. The outlier-eliminated analysis provided smaller standard deviations in the distributions, which result in less uncertainty of calculated software reliability. Relatively large discrepancies were observed for defect density node and detection probabilities for the defects passed from the previous phase node.

10.1.3 Representation of Typical NPP Safety-Related Software

As defined in Chapter 5, the “Medium” state for an indicator attribute represents required activities associated with this attribute that are completed satisfactorily with respect to the US regulatory licensing review. Therefore, it is reasonable and conservative to assume that any licensing approved safety-related software in the US NPPs has “Medium” scores for all attributes.

As discussed in Section 6.3 , a sole-function safety software has a representative size of 50 FPs. Subsequently, a rough estimate of the number of defects remaining using the BBN model developed in this study can be estimated. This estimate was used to estimate the average FSD.

10.1.4 Possible Applications

This model may be applied to estimate either the failure probability for deployed safety-related NPP software or the number of residual defects. Such results can be used to evaluate the quality of the digital I&C systems in addition to estimating potential reactor risk. The intermediate number of defects can also be used to gain quality insights on the software development process.

10.2 Assumptions, Limitations and Future Work

Causal relationships between the qualities of the SDLC activities and the software product in terms of the number of residual defects (and ultimately, the software failure probability) are observed to be correlated across software industries. Modeling these relationships in a BBN, however, leads to disagreement among investigators regarding its feasibility and accuracy. This study demonstrates a practical and credible BBN solution. Assumptions behind this model should

be revisited to summarize the limitations of this work and identify potential future work to improve current model.

The SDLC was abstracted into five waterfall phases in this study. The modern software developments, such as spiral or agile, are normally much more complicated with many more iterations than this abstraction. Though the overall defects introduction, detection, and removal for multiple iterations can be captured, some chronical details may be missed in this model. In addition, the unavailability of the SDLC details for a real NPP software package to the research team limits modeling granularity. Future research is recommended to model a real NPP software development process.

The identification of a set of complete and independent attributes comprise the foundation of a credible BBN model. The set of attributes identified herein are based on IEEE STD 1012 and are technically sound. Future work should either verify that this set of attributes is complete and independent or develop a better set of attributes. The identification based on a real NPP software development process is also recommended.

Due to lack of sufficient data, expert opinions were used for NPT quantification. In the second round elicitation, large diverse opinions were observed for some nodes. The variety of opinions may be due to different levels of knowledge and experience among experts. To reduce the uncertainty caused by diverse opinions, other sources of evidence were used for Bayesian updates and a sensitivity study was performed. Due to the scarcity of raw data, the number of observations was assumed to be one in this study. Future work is recommended to reduce the uncertainty in NPT quantification using more data points.

Studies have demonstrated that software failure probability is a function of residual defects (locations and types) and the ways in which software is used (operational profile). This study simplifies this function as an average linear relationship FSD. This has been identified as an area suitable for further study should additional work be performed in the future. Global experience data was collected and a hypothetical NPP safety software size was counted in FP to back-calculate the average FSD used in the two trial system applications. More data points for a real safety software is recommended to be collected and used to calibrate this FSD.

The number of residual defects estimated from this BBN method in theory includes any type of defects such as incomplete requirements defects, which might be introduced during user requirements solicitation process. For example, functions that cover unanticipated plant conditions could be omitted if the user was unaware of these conditions. The BBN approach described in this report includes defects might lead to a “real” failure (no actuation under demand conditions, also referred to type I failure in the literature), and those that cause a spurious failure (actuation under no condition, also referred to type II failure in the literature). However, the BBN model used in this study did not differentiate between these two types of defects.

Coverage of the unanticipated conditions and impacts of spurious actuations become significant considerations in digital I&C safety evaluations for NPPs. These areas have been identified as areas for future studies.

11 REFERENCES

[Aldemir 2006] Aldemir, T., et al., “Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments,” NUREG/CR-6901, February 2006.

[Aldemir 2007] Aldemir, T., et al., “Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments,” NUREG/CR-6942, October 2007.

[Aldemir 2009] Aldemir, T., et al., “A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems,” NUREG/CR-6985, February 2009.

[ASME NQA] American Society of Mechanical Engineers, "Quality Assurance Requirements for Nuclear Facility Applications," 2000.

[Atwood 2002] C.L., Atwood, “Handbook of Parameter Estimation for Probabilistic Risk Assessment”, NUREG/CR-6823, September 2002.

[Backstrom 2014] Backstrom, O., et al., “Software reliability analysis for PSA”, NKS (Nordic Nuclear Safety Research), NKS-304, March 2014.

[Bickel 2008] Bickel, J.H., “Risk implications of digital reactor protection system operating experience,” *Reliability Engineering and System Safety,* Vol. 93, pp.107–124, 2008.

[Chu 2008] Chu, T.L., et al., “Traditional Probabilistic Risk Assessment Methods for Digital Systems,” NUREG/CR-6962, October 2008.

[Chu 2009a] Chu, T.L., et al., “Modeling a Digital Feedwater Control System Using Traditional Probabilistic Risk Assessment Methods,” NUREG/CR-6997, September 2009.

[Chu 2009b] Chu, T.L., et al., “Workshop on Philosophical Basis for Incorporating Software Failures into a Probabilistic Risk Assessment,” Brookhaven National Laboratory, Technical Report, BNL-90571-2009-IR, November 2009.

[Chu 2010] Chu, T.L., et al., “Review Of Quantitative Software Reliability Methods,” Brookhaven National Laboratory, BNL-94047-2010, September 2010.

[Chu 2013] Chu, T. L., Yue, M., Martinez-Guridi, G., and Lehner, J., “Development of Quantitative Software Reliability Models for Digital Protection Systems of Nuclear Power Plants,” NUREG/CR-7044, October 2013.

[Chu 2015] T.L. Chu, et al., “Development of a Statistical Testing Approach for Quantifying Software Reliability and Its Application to an Example System,” draft NUREG/CR, submitted to the NRC on August 1, 2015.

[CSNI 2015] Nuclear Energy Agency, Committee on the Safety of Nuclear Installations, “Failure Modes Taxonomy for Reliability Assessment of Digital I&C Systems for PRA,” NEA/CSNI/R(2014)16.

[Delic 1995] Delic, K.A., Mazzanti, F., and Strigini, L., “Formalising a Software Safety Case via Belief Networks,” Centre for Software Reliability, City University, London, U.K, SHIP/T046 v1.9, September 27, 1995.

[Delic 1997] Delic, K.A., Mazzanti, F., and Strigini, L., “Formalizing Engineering Judgment on Software Dependability via Belief Networks,” Sixth IFIP International Working Conference on Dependable Computing for Critical Applications, “Can We Rely on Computers?” Garmisch-Partenkirchen, Germany, IEEE Computer Society Press, pp. 291-305, 1997.

[DO-178C] Radio Technical Commission for Aeronautics, “Software Considerations in Airborne Systems and Equipment Certification,” RTCA/DO-178C, Prepared by Special Committee 205 (SC-205), 2012.

[DOE 2010] Department of Energy, “Safety Software Guide for Use with 10 CFR 830 Subpart A, Quality Assurance Requirements, and DOE O 414.1C, Quality Assurance,” DOE G 414.1-4, Approved June 17, 2005, Certified November3, 2010.

[Dunham 1988] Dunham, J. R., and Lauterbach, L. A., “An Experiment in Software Reliability Additional Analyses Using Data from Automated Replications,” Research Triangle Institute, NASA Contractor Report 178395, January 1988.

[Dunham 1990] Dunham, J., Research Triangle Institute and Finelli, G.B., NASA Langley Research Center, "Real-time Software Failure Characterization," Proceedings of the Fifth Annual Conference on Computer Assurance, 1990, COMPASS '90, 'Systems Integrity, Software Safety and Process Security'., 25-28 June 1990.

[Eide 1999] Eide, S. A., Beck, S. T., Calley, M. B., Galyean, W. J., Gentillon, C. D., Khericha, S. T., Novack, S. D., and Wierman, T. E., “Reliability Study: Westinghouse Reactor Protection System, 1984-1995,” NUREG/CR-5500, Vol. 2, April 1999.

[Eom 2009] Eom, H-S., et.al., “Reliability Assessment of a Safety-Critical Software by Using Generalized Bayesian Nets,” 6th ANS Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interfaces Technologies (NPIC&HMIT 2009), Knoxville, Tennessee, April 5-9, 2009.

[Eom 2013] Eom, H.S., Park, G.Y.,Jang, S.C., Son, H.S.,Kang, H.G.,"V&V-based remaining fault estimation model for safety–critical software of a nuclear power plant," Annals of Nuclear Energy, Vol. 51, pp.38–49, 2013.

[EPRI 2010] Electric Power Research Institute, "Estimating Failure Rates in Highly Reliable Digital Systems," EPRI 1021077, 2010.

[Fagan 1976] Fagan, M.E., "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal, Vol. 15, 3, pp.182, 1976.

[Fenton 2007a] Fenton, N., et al., "Predicting Software Defects in Varying Development Lifecycles Using Bayesian Nets," *Information and Software Technology*, Vol. 49, pp. 32-43, 2007.

[Fenton 2007b] Fenton, N. E., Neil, M. and Caballero, J.G., "Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 10, October 2007.

[Fenton 2008] Fenton, N., Neil, M., and Marquez, D.," Using Bayesian Networks to Predict Software Defects and Reliability," Proceedings for the Institution of Mechanical Engineers, Vol. 222, Part O, Journal of Risk and Reliability, 2008.

[Fenton 2012] Fenton N. E., Neil, M., Risk Assessment and Decision Analysis with Bayesian Network, 2012.

[Fineman 2010] Fineman, M., 2010, "Improved Risk Analysis for Large Project: Bayesian Networks Approach," PhD thesis, Queen Mary University, London.

[Fisseha, 2008] Fisseha, B., 'International Database on Digital I&C Products, Platforms and Projects in Nuclear Power Plants: IDIP," IAEA, Power Point Slides, Regional Workshop on Modernization Projects of NPP Instrumentation and Control Systems Related to Power Uprate and License Renewal, April 14-18, 2008.

[Heckerman 1995] Heckerman, D., "A tutorial on learning with Bayesian networks," Technical Report MSR-TR-95-06, Microsoft Research, Microsoft Corporation, 1995.

[Gotel 1994] Gotel, O.C., and Finkelstein, C.W., "An Analysis of the Requirements Traceability Problem," Proceedings of the First International Conference on Requirements Engineering, 1994.

[Hayter 2012] Hayter, Anthony. Probability and statistics for engineers and scientists. Cengage Learning, 2012.

[IEC 60880] International Electrotechnical Commission, “Nuclear power plants Instrumentation and control systems important to safety Software aspects for computer-based systems performing category A functions,” IEC 60880, 2006.

[IEC 61508] International Electrotechnical Commission, “Function Safety of Electrical/Electronic/Programmable Safety-Related Systems,” Parts 1-7, IEC 61508, various dates.

[IEEE 7-4.3.2 2010] IEEE, “IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations,“ IEEE Std 7-4.3.2™-2010, June 17, 2010.

[IEEE 610.12] IEEE, “Systems and software engineering Vocabulary,“ IEEE Std 61012, ISO/IEC/IEEE 24765, First edition, December 15, 2010.

[IEEE 829] IEEE, “IEEE Standard for Software and System Test Documentation,” IEEE Computer Society, IEEE Std 829-2008, July 18, 2008.

[IEEE 830] IEEE, “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Computer Society, IEEE Std 830-1998, October 20, 1998.

[IEEE 1012] IEEE, “IEEE Standard for System and Software Verification and Validation”, IEEE Computer Society, IEEE Std 1012™-2012, May 25, 2012.

[IEEE 1233] IEEE, “IEEE Guide for Developing System Requirements Specifications,”, IEEE Computer Society, IEEE Std 1233, 1998.

[IEEE 1633] Institute of Electrical and Electronics Engineers (IEEE), ”IEEE Recommended Practice on Software Reliability,” IEEE Standard 1633-2008, March 27, 2008.

[IEEE 1074] IEEE, “IEEE Standard for Developing a Software Project Life Cycle Process,”, IEEE Computer Society, IEEE Std 1074, Piscataway, NJ, 2006.

[IEEE 24765] IEEE, “Systems and software engineering Vocabulary”, IEEE Std 24765-2010, December 15, 2010.

[IFPUG] International Function Point Users Group, http://www.ifpug.org.

[INL 2009a] Software Quality Assurance Plan for Loop 2A Instrumentation and Operating Control System, PLN-2739, Idaho National Laboratory, 2009.

[INL 2009b] Specification for Distributed Control System for Loop 2A at the Advanced Test Reactor, SPC-988, Idaho National Laboratory, 2009.

[INL 2010a] Design Control and Quality Assurance for Structures, Systems or Components that include Software at the ATR Complex, PLN-2293, Idaho National Laboratory, 2010.

[INL 2010b] Laboratory-wide Procedure: Software Quality Assurance, LWP-13620, Idaho National Laboratory, 2010.

[INL 2010c] Technical and Functional Requirements for 2A Loop Instrumentation and Operating Control System, TFR-499, Idaho National Laboratory, 2010.

[INL 2014a] Verification and Validation (V&V) Plan for 2A Loop Instrumentation and Operating Control System, PLN-2740, Idaho National Laboratory, 2014.

[INL 2014b] Verification and Validation (V&V) Report for 2A Loop Instrumentation and Operating Control System, PLN-4681, Idaho National Laboratory, 2014.

[Jensen 1996] Jensen, F.V., An Introduction to Bayesian Networks, Springer-Verlag New York Inc., 1996.

[Jensen 2002] Jensen, F.V., *Bayesian Networks and Decision Graphs*, Springer, 2002.

[Jockenhövel-Barttfeld 2015] Mariana Jockenhövel-Barttfeld, et al., “Modelling Software Failures of Digital I&C in Probabilistic Safety Analyses based on the TELEPERM® XS Operating Experience”, *atw International Journal for Nuclear Power*, Vol. 60 (2015) | Issue 3 ı March 2015.

[Jones 2008] Capers Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, Third edition, 2008.

[KAERI 2010] KAERI/TR-4092/2010, Reliability Assessment Method Of Reactor Protection System Software by Using V&V based Bayesian Nets, Korea Atomic Energy Research Institute.

[KAERI 2013] KAERI, “Guideline for Reliability Analysis of Safety-Critical Software” Technical Report KAERI/TR-5222/2013, Korea Atomic Energy Research Institute, 2013.

[Kass 1995] Kass, Robert E., and Adrian E. Raftery. "Bayes factors." Journal of the American Statistical Association 90.430, 773-795, 1995.

[KINS 2015] http://opis.kins.re.kr/opis?act=KEOBA1100R.

[Korsah 2010] Korsah, K., et al., “An Investigation of Digital Instrumentation and Control System Failure Modes,” Oak Ridge National Laboratory, ORNL/TM-2010/32, March 2010.

[Korb 2010] Korb, K.B. and Nicholson, A.E., Bayesian Artificial Intelligence, Second Edition, CRC Press, New York, 2010.

[Kragt 2009] Kragt, M. E., “A beginners guide to Bayesian network modeling for integrated catchment management,” Landscape Logic Technical Report No. 9, Department of the Environment, Water, Heritage and the Arts, Australia, July 2009.

[Krieg 2001] Krieg, M. L., “A Tutorial on Bayesian Belief Networks,” DSTO Electronics and Surveillance Research Laboratory, Department of Defense, Australia, 2001.

[Lawrence 1993] Lawrence, J.D., “Software Reliability and Safety in Nuclear Reactor Protection Systems,” NUREG/CR-6101, June 11, 1993.

[Littlewood 1980] Littlewood, B., "Theories of Software Reliability: How good are they and how can they be improved?", *IEEE Transactions on Software Engineering*, SE-6, pp. 489-500, 1980.

[Lyu 1996] Lyu, M.R., Editor in Chief, Handbook of Software Reliability Engineering, McGraw-Hill, 1996.

[MacQueen 1967] MacQueen, James. "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967.

[MIL-STD-882E] Department of Defense, “Department of Defense Standard Practice System Safety,” MIL-STD-882E, May 11, 2012.

[Musa 1987] Musa, J.D., Iannino, A., and Okumoto, K., *Software Reliability: Measurement, Prediction, Application*, New York: McGraw-Hill, 1987.

[Myllymaki 2002] Myllymäki, P., et al., “B-Course: a web-based tool for Bayesian and causal data analysis,” Int. J. Artif. Intell. Tools, 11 (3), pp. 369–387, 2002.

[NEA 2009] Nuclear Energy Agency, “Recommendations On Assessing Digital System Reliability In Probabilistic Risk Assessments Of Nuclear Power Plants,” NEA/CSNI/R(2009)18, December 17, 2009.

[Neufelder 2002] Neufelder, A.M., “The Facts about Predicting Software Defects and Reliability,” *The Journal of the Reliability Analysis Center (RAC)*, Second Quarter – 2002.

[NRC 1995a] U.S. NUCLEAR REGULATORY COMMISSION, “Use of Probabilistic Risk Assessment Methods in Nuclear Regulatory Activities,” Final Policy Statement, August 16, 1995.

[NRC 2007] U.S. NUCLEAR REGULATORY COMMISSION, “Guidance on Software Reviews for Digital Computer-Based Instrumentation and

Control Systems," NUREG-800, Standard Review Plan, Branch Technical Position 7-14, Revision 5, March 2007.

[NRC 2008] U.S. NUCLEAR REGULATORY COMMISSION, "Review of New Reactor Digital Instrumentation and Control Probabilistic Risk Assessment," Interim Staff Guidance, DI&C-ISG-03, August 11, 2008.

[NRC 2010a] U.S. NUCLEAR REGULATORY COMMISSION, "NRC Digital System Research Plan FY2010-FY2014," February 2010.

[NRC 2016] U.S. NUCLEAR REGULATORY COMMISSION, "BBN Expert Opinion Elicitation Materials and Responses", ML16201A129, July 2016.

[NRC 2016a] U.S. NUCLEAR REGULATORY COMMISSION, "A BBN Model for the Probability of Software Failure on Demand, Round 1 Expert Opinion Elicitation", ML16201A140, July 2016.

[NRC 2016b] U.S. NUCLEAR REGULATORY COMMISSION, "A BBN Model for the Probability of Software Failure on Demand, Round 2 Expert Opinion Elicitation", ML16201A141, July 2016.

[NRC 2016c] U.S. NUCLEAR REGULATORY COMMISSION, "A BBN Model for the Probability of Software Failure on Demand, Round 3 Expert Opinion Elicitation", ML16201A144, July 2016.

[Nyberg 2006] Nybert, J. B., Marcot, B. G., and Sulyma, R., "Using Bayesian Belief Networks in Adaptive Management," Canadian Journal of Forest Research, vol. 36, pp 3104-16, 2006.

[Park 2012] G.Y. Park, H.S. Eom, S.C. Jang, H.G. Kang, "Software failure probability assessment by Bayesian inference," Nuclear Technology, Vol. 193, pp. 107-118, 2012.

[Pearl 1988] J. Pearl, Probabilistic Reasoning in Intelligent Systems, San Francisco CA: Morgan Kaufmann, 1988.

[Pollino 2008] Pollino, C. A. and Hart, B.T., Developing Bayesian Networks Within A Risk Assessment Framework, 4th Biennial Meeting of iEMSs, 2008.

[Poore 2012] Poore III, W. P., et al., "Initial Inventory of Digital I&C Systems Used in Domestic Nuclear Power Plants," Proprietary, Oak Ridge National Laboratory, LTR/NRC/RES/2012-002, January 2012.

[RAC 1998] Reliability Analysis Center, and Performance Technology, "New System Reliability Assessment Method," Prepared for Rome Laboratory, IITRI Project No. A06830, June 1, 1998.

[RG 1.168-2013] US NRC, (Draft was issued as DG-1267, dated August 2012), Verification, Validation, Reviews, and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants

[RG 1.173-2013] U.S. NUCLEAR REGULATORY COMMISSION, “Developing Software Life-Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants,” Regulatory Guide 1.173, Revision 1, July 2013.

[Schwarz 1978] Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics 6.2, 461-464, 1978.

[Sheppard 2012] Sheppard, M., “Fit all valid parametric probability distributions to data,” tech.rep., MATLAB File Exchange, 2012.

[Smidts 2004] Smidts, C, Li, M., et al, “Preliminary Validation of a Methodology for Assessing Software Quality”, NUREG/CR-6848, July 2004

[Smidts 2011] Smidts, C, Shi, Y., et al, “A Large Scale Validation of a Methodology for Assessing Software Reliability”, NUREG/CR-7042, July 2011

[Spiegelhalter 2003] Spiegelhalter, David, et al. "WinBUGS user manual." (2003).

[Suermondt 1992] Suermondt, H. J. “Explanation in Bayesian belief networks,” PhD Thesis, Stanford, University, Stanford, CA, 1992.

[Turkey Point 1994] Turkey Point Units 3 and 4, “Design Defect in Safeguards Bus Sequencer Test Logic Places Both Units Outside the Design Basis,” Licensee Event Report, 94-005-01, February 9, 1995.

[Uusitalo 2007] Uusitalo, L., “Advantages and challenges of Bayesian networks in environmental modeling,” Ecological Modelling, Vol. 203, pp. 312-318, 2007.

[Wood 2012] Wood, R. T., et al., “Classification Approach for Digital I&C Systems at U.S. Nuclear Power Plants,” Oak Ridge National Laboratory, Letter Report LTR/NRC/RES/2012-001, February 2012.

APPENDIX A
DETAILED BBN MODEL OF ALL PHASES

In this work, the software development life cycle (SDLC) is categorized into five phases: Requirement, Design, Implementation, Test, and Installation/checkout. In each phase, the number of remaining defects is estimated based on the quality of development and the quality of verification and validation (V&V). The basic process to estimate the probability of software failure is shown in Figure A-1. In each phase, the development and the V&V quality nodes are evaluated based on the attributes to obtain the number of defects remaining at the end of that phase. This number of defects becomes an input to the model for the next phase. The total defects remaining in the last phase (Installation phase) is multiplied with the fault size distribution to obtain the probability of software failure.

The estimation of the total defects remaining in the Requirement, Design, Implementation, Test, and Installation phases are shown in Figure A-2, Figure A-5, Figure A-8, Figure A-11, and Figure A-14, respectively. The estimation process for the Requirement phase is different from the other phases because it does not have a previous phase. The Test phase (Figure A-11) has additional arrows from the development quality node to the detection probabilities nodes because test activities aim to eliminate defects.

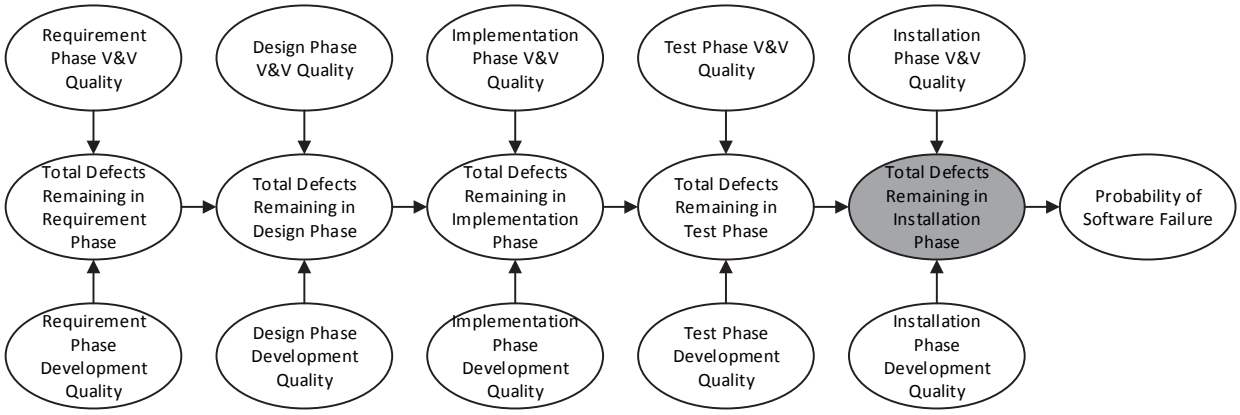

Figure A-1 Overview of the BBN Model

Within each phase, the "development quality" node and the "V&V quality" node are connected to their attribute nodes with a diverging configuration. These nodes have three states (*Low, Medium, and High*). The attribute nodes also have the same three states. Figure 11-3 shows the "development quality" node and its attributes in the Requirement phase. The relationship between the quality nodes and attribute nodes are developed using expert estimations in the second expert opinion elicitation.

The function points of a target software is the input of the "size and complexity measure" node and "the number of defects introduced in a phase" node. The "size and complexity measure" node has three states (*Low*, *Medium*, and *High*). If the number of function points is below 100, the size and complexity measure node is *Low*. The number of function points between 100 and 1000 corresponds to *Medium* complexity, and above 1000, the complexity is *High*.

"The number of defects introduced in this phase" node is determined by the multiplication of the "function points" and "the number of defects inserted per function point" nodes. The NPTs of the

“defects detection probability for defects introduced in this phase” and the “defects detection probability for defects introduced in previous phases” nodes are developed based on the expert estimates. “The number of defects passed from the previous phases detected” and “the number of defects introduced in this phase detected” nodes are modeled by a binomial distribution. In “the number of defects passed from the previous phases detected” node, the number of trials is “the number of defects passed from the previous phases” and the probability of success is the “defects detection probability for defects introduced in the previous phases”. In “the number of defects passed from the previous phases detected” node, the number of trials is “the number of defects introduced in this phase” and the probability of success is the “defects detection probability for defects introduced in this phase”.

“The number of defects introduced in this phase remaining” node is the difference between “the number of defects introduced in this phase” and “the number of defects introduced in this phase detected”. “The number of defects introduced in the previous phases remaining” node is the difference between “the number of defects introduced in the previous phases” and “the number of defects introduced in the previous phases detected”. The “total number of defects remaining in this phase” node is the sum of “the number of defects passed from the previous phases remaining” and “the number of defects introduced in the current phase remaining”.

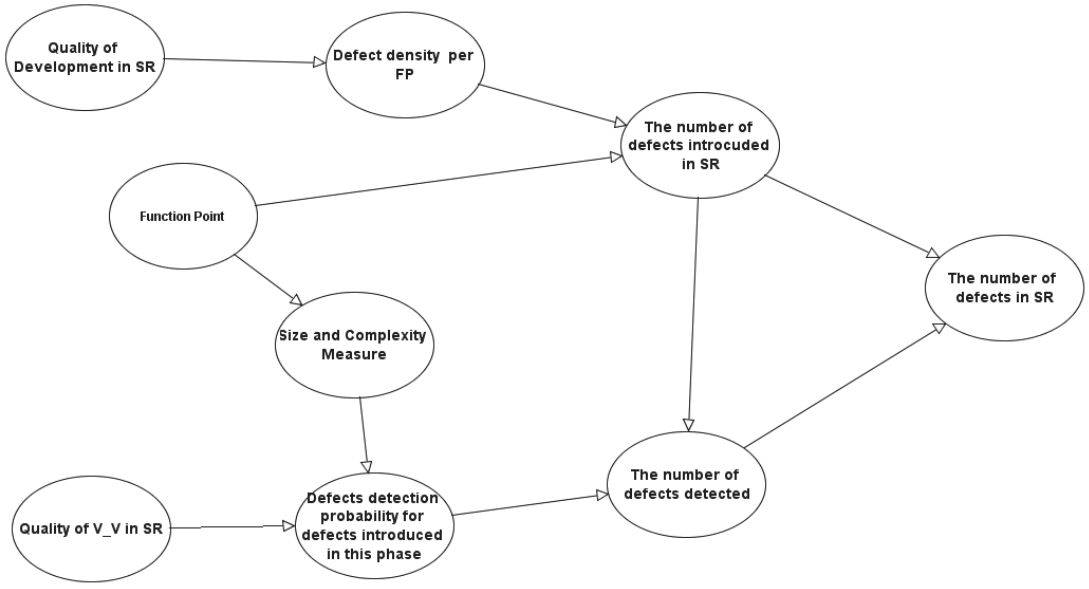

Figure A-2 The Number of Defects Estimation in the Requirement (SR) Phase

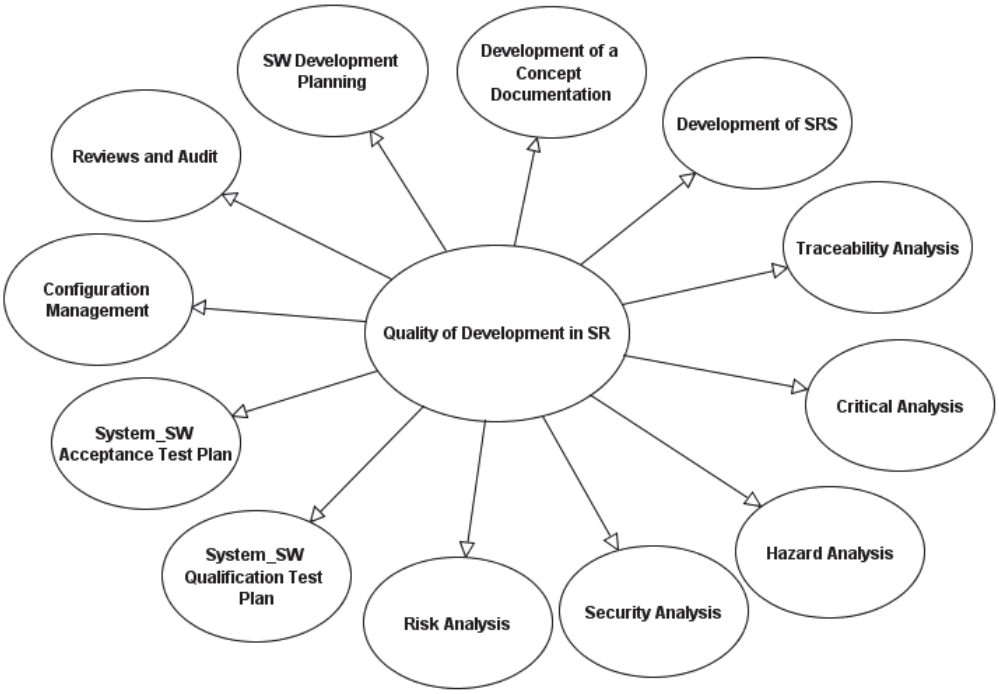

Figure A-3 Quality of Development in the Requirement (SR) Phase

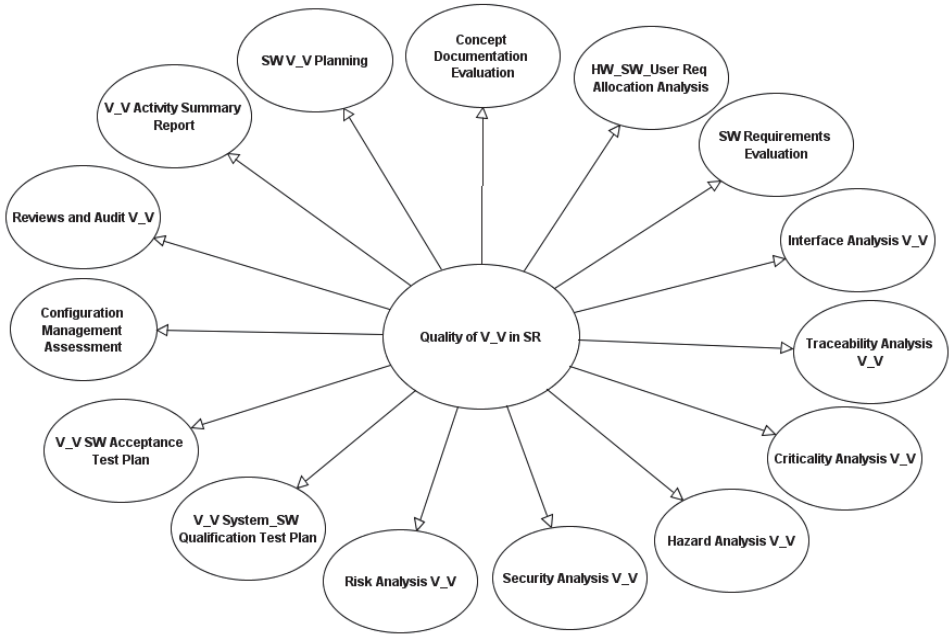

Figure A-4 Quality of V&V in the Requirement (SR) Phase

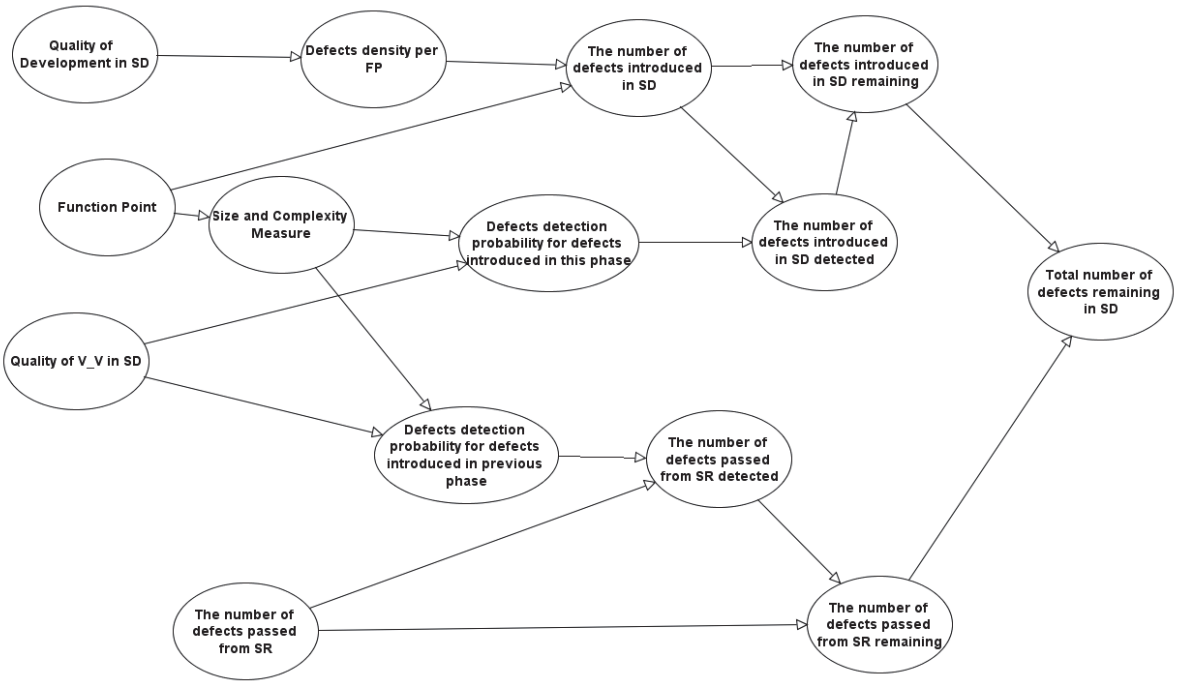

Figure A-5 The Number of Defects Estimation in the Design (SD) Phase

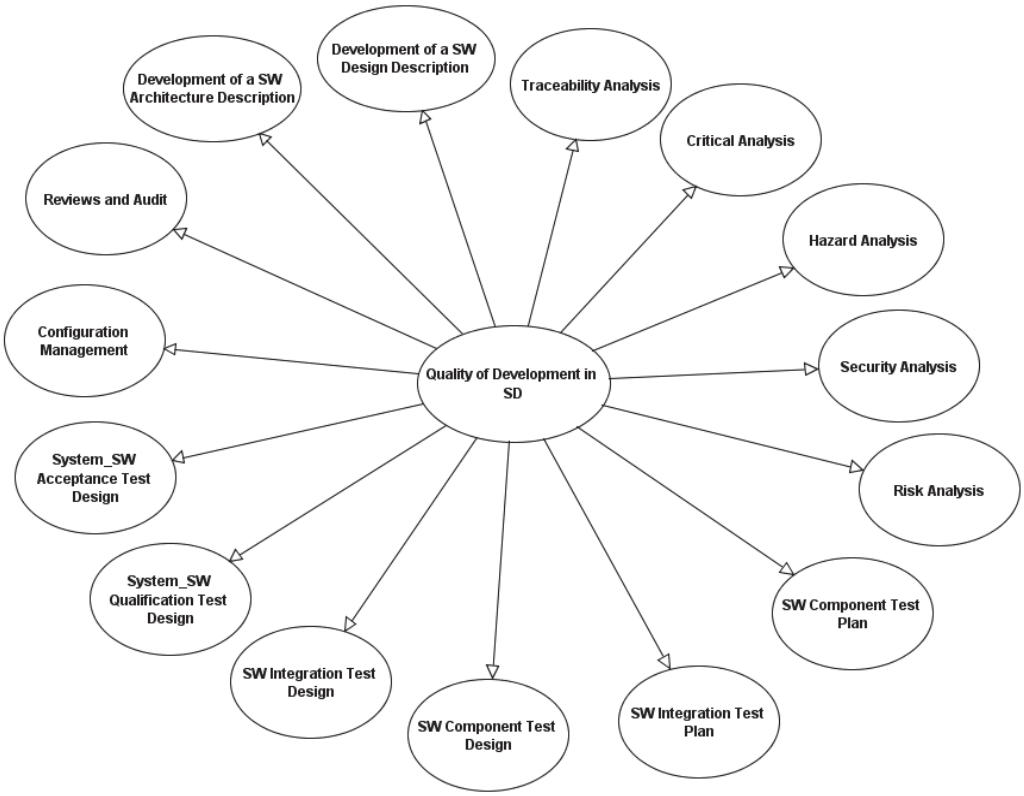

Figure A-6 Quality of Development in the Design (SD) Phase

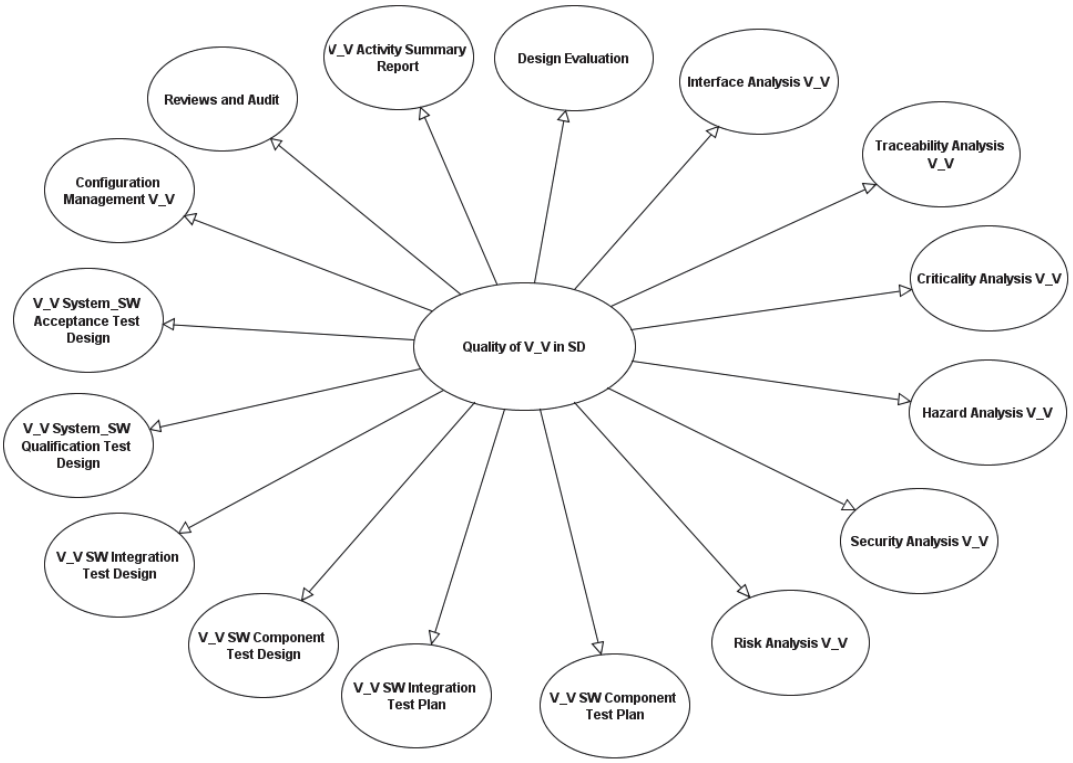

Figure A-7 Quality of V&V in the Design (SD) Phase

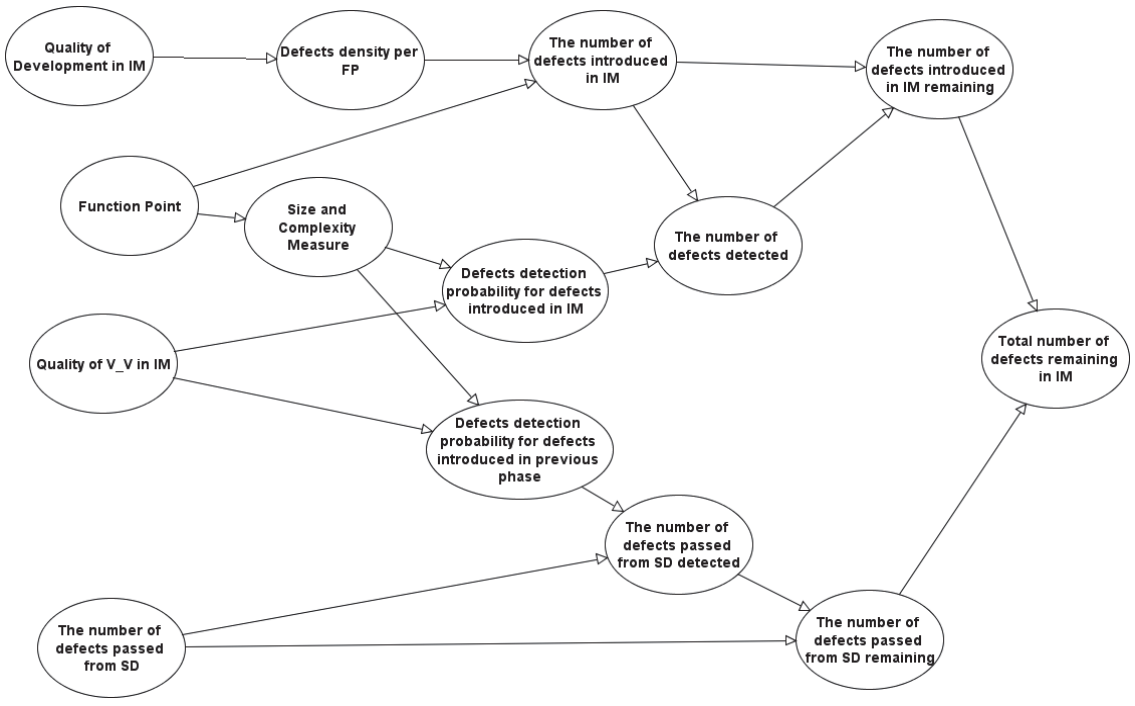

Figure A-8 The Number of Defects Estimation in the Implementation (IM) Phase

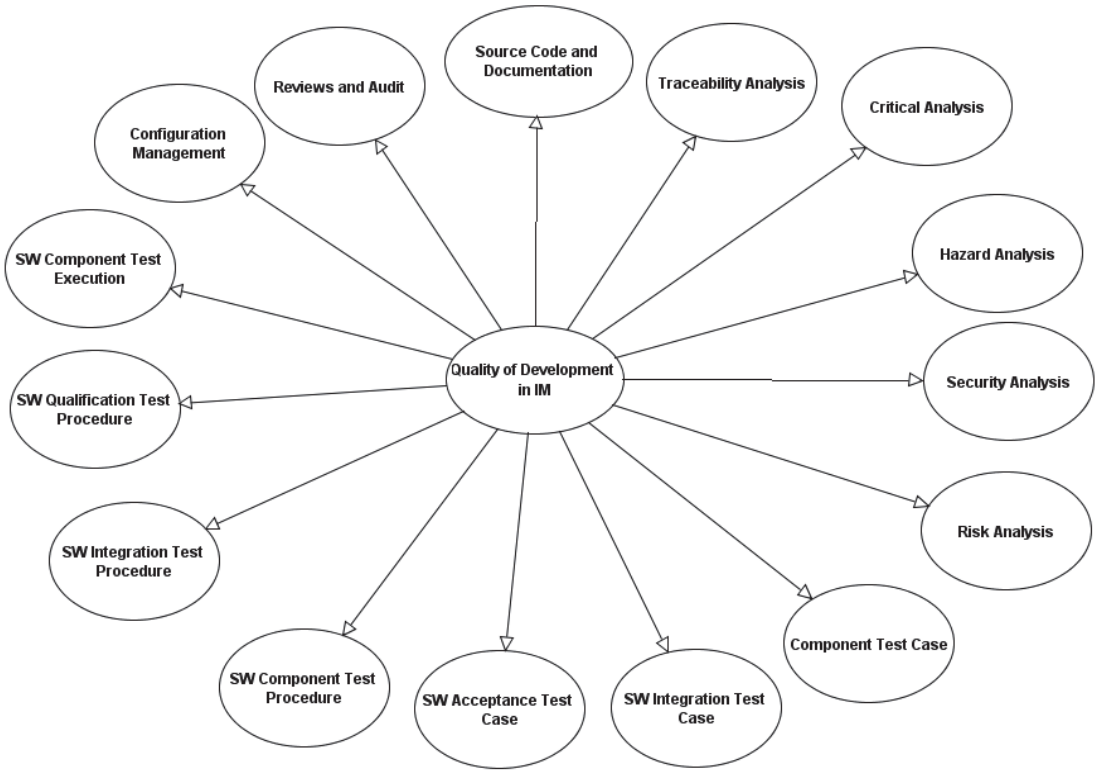

Figure A-9 Quality of Development in the Implementation (IM) Phase

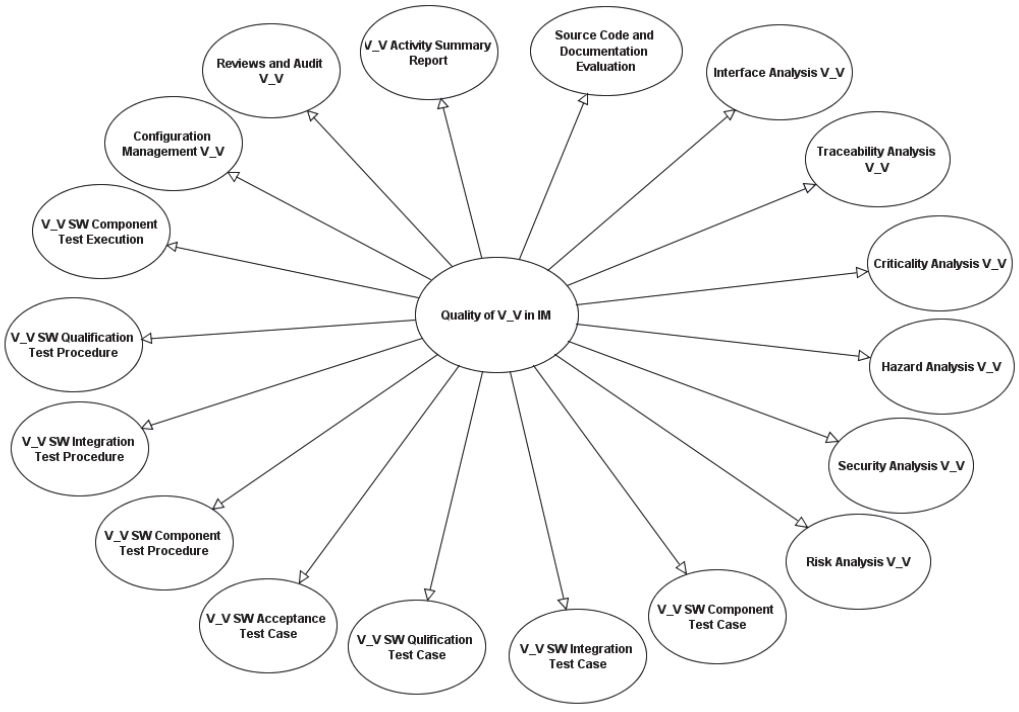

Figure A-10 Quality of V&V in Implementation (IM) Phase

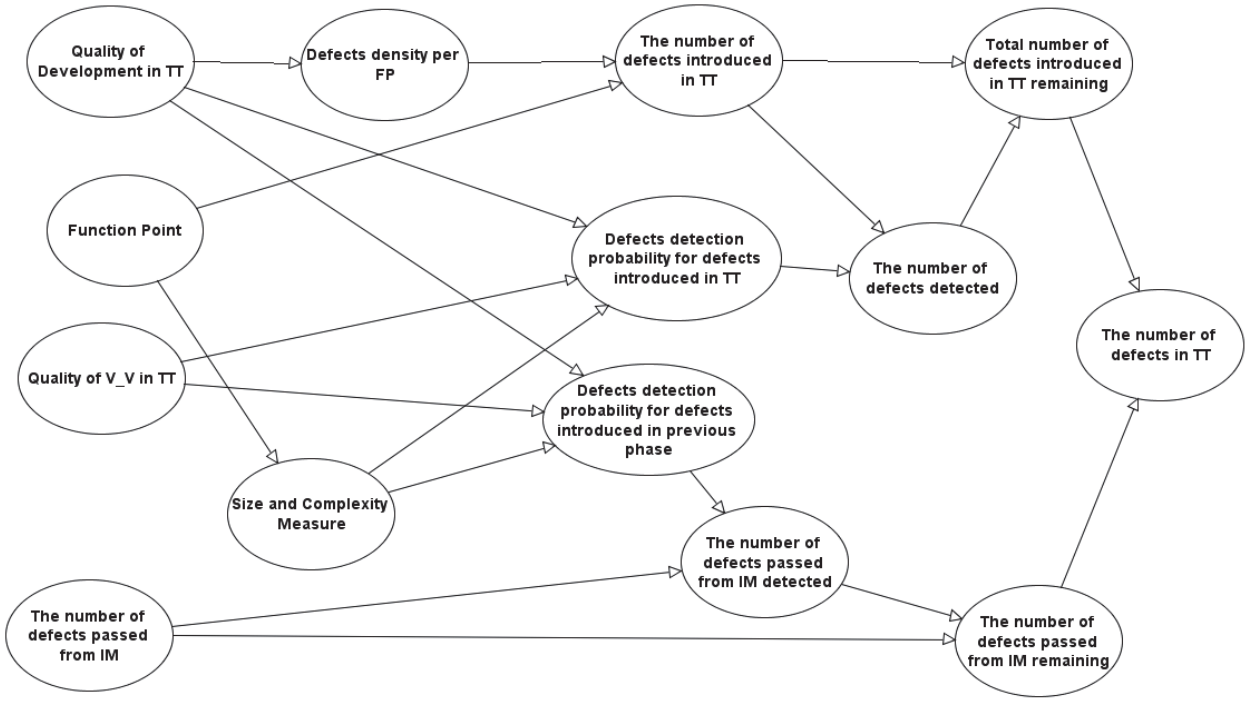

Figure A-11 The Number of Defects Estimation in the Test (TT) Phase

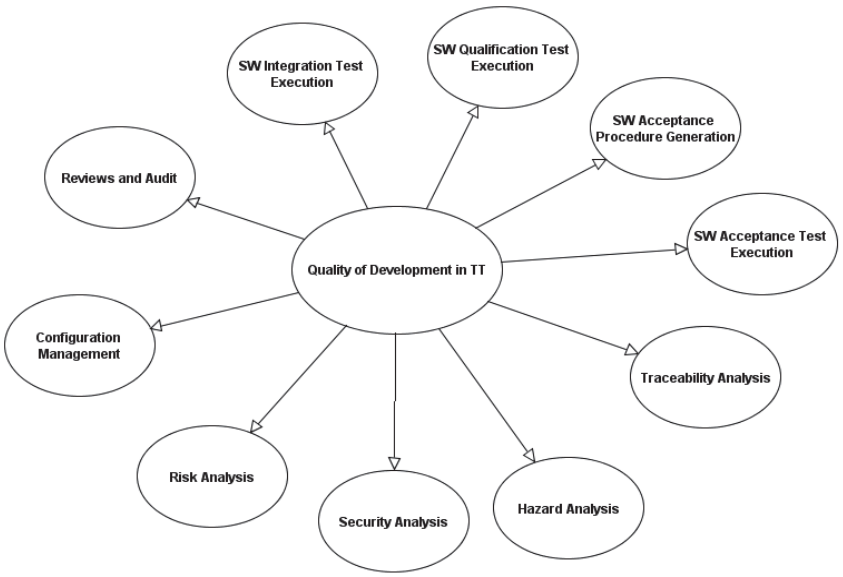

Figure A-12 Quality of Development in the Test (TT) Phase

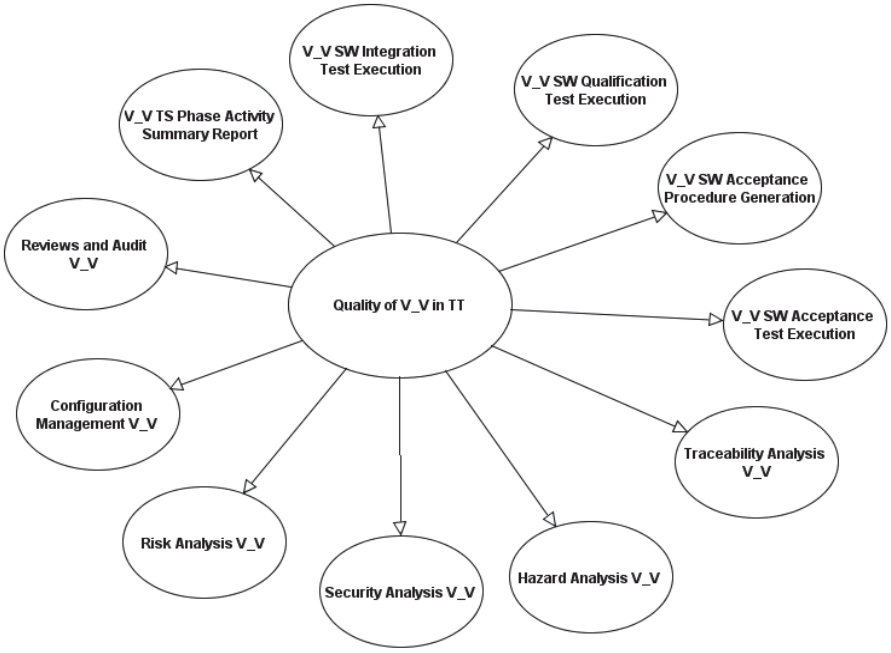

Figure A-13 Quality of V&V in the Test (TT) Phase

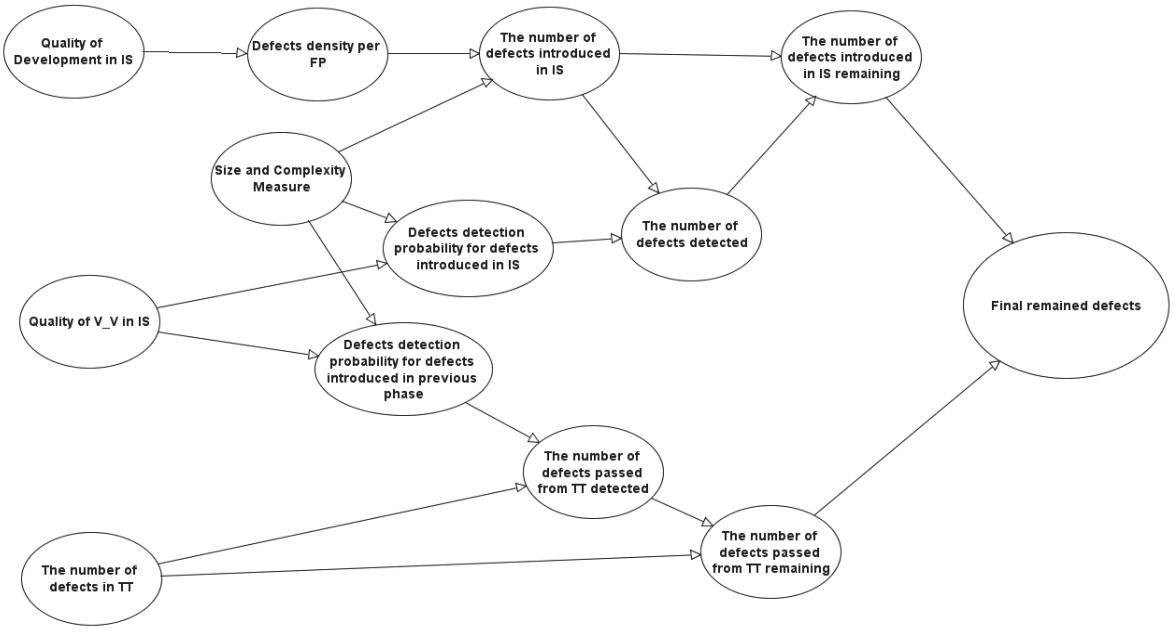

Figure A-14 The Number of Defects Estimation in the Installation and Checkout (IS) Phase

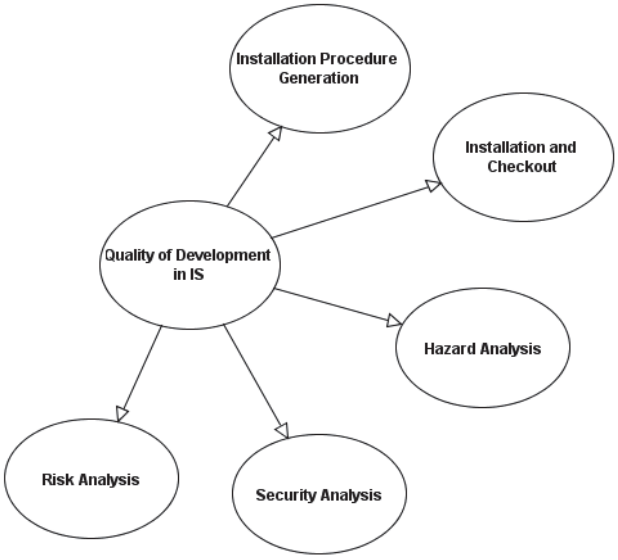

Figure A-15 Quality of Development in the Installation and Checkout (IS) Phase

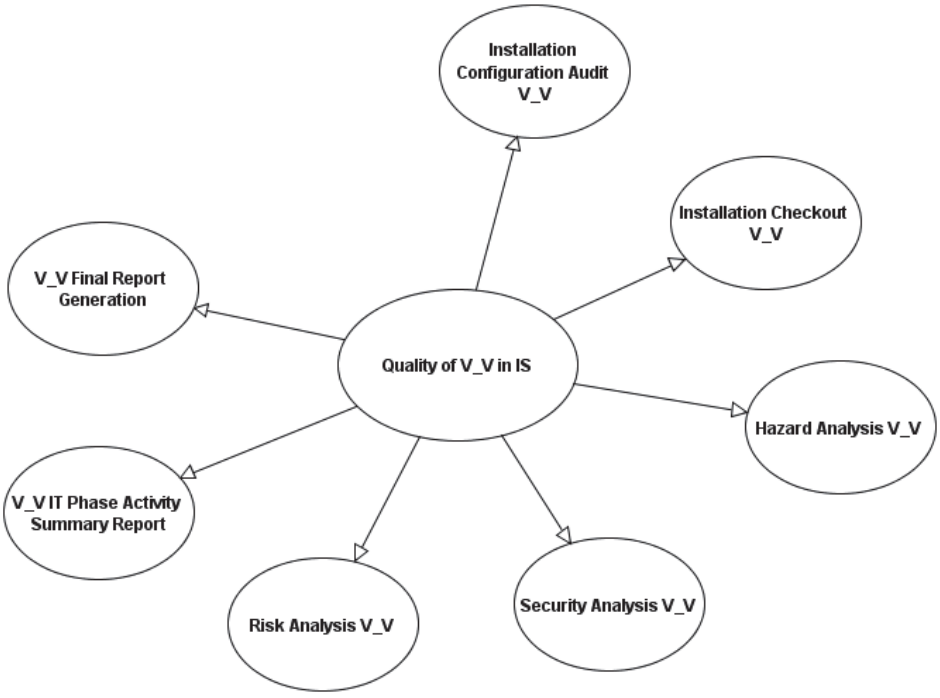

Figure A-16 Quality of V&V in the Installation and Checkout (IS) Phase

APPENDIX B
DETAILED ATTRIBUTES OF ALL PHASES

B.1 Attributes of Software Requirements Specifications Development Activities

The Software Concept phase and Software Requirements phase (i.e., IEEE 1012) are merged together as the Requirements Specifications phase in our BBN model. The concept activity represents delineating a specific implementation solution to solve the user's problem. During developing the concept, the system's architecture is selected and system's requirements are allocated to hardware, software, and user-interface components. In developing the concept document (i.e., the SyRS), the development or selection of system's architectural design and the development of the system's requirements are key. The purpose of developing the SRS is to satisfy system requirements and user's needs, and ensure the correctness, consistency, completeness, accuracy, readability, testability, and robustness in the SRS. More detailed development activities are specified in the table below.Each of the numbered items therein is called an attribute and is only considered to be satisfied if all of the activities defined under it are performed *satisfactorily*. All of the items are phase-specific except items "Software Development Planning[1]", "Configuration Management", and "Reviews and Audit", which are generic because they belong to the Management Process and QA functions other than the Development Process that we are dealing with. The Management Process and QA functions are common to products (system, hardware, or software) development and the associated process (as in each phase of the development lifecycle process). However, these management process activities and QA attributes are included in our consideration because they are necessary attributes for developing reliable and safe software. Additionally, even with these common items, phase-specific activities will be spelled out so that they will be dealt with, and/or merely updated with available specific phase-input data in each phase because they need to be evaluated phase by phase. After the requirement specifications phase, a design phase will be initiated.

Finally, examples of additional activities are added for some attributes. These additional activities are expected to increase the quality score in the BBN model.

The requirement specifications activities include the following attributes with more detailed activities specified in the table below.

(1) Software Development Planning
(2) Development of a concept documentation
(3) Development of Software Requirements Specifications
(4) Traceability Analysis- Requirements Specifications Phase
(5) Criticality Analysis- Requirements Specifications Phase
(6) Hazard Analysis- Requirements Specifications Phase
(7) Security Analysis- Requirements Specifications Phase
(8) Risk Analysis- Requirements Specifications Phase
(9) System/Software Qualification Test Plan Generation

---

[1] Software development planning continues throughout the software development phases and is continuously reviewed and updated.

(10) System/Software Acceptance Test Plan Generation
(11) Configuration Management- Requirements Specifications Phase
(12) Reviews and Audit- Requirements Specifications Phase

(1) Software Development Planning

During the software development planning process, the Software Development Plan (SDP), which describes the plan for technical project development, will be developed. The SDP outlines the management, implementation and resource characteristics. The management characteristics include purpose, organization, oversight, and risks. The implementation characteristics include measurement, procedures, and schedule.

The resource characteristics include methods/tools and standards.

Software Development Planning task is as follows (in reference to BTP 7-14, NUREG-CR-6101, and IEC 60880)

a. Develop and generate a SDP that is consistent with the lifecycle process defined in IEEE Std 1074 (which is endorsed by Regulatory Guides 1.173).
b. Ensure that the SDP is consistent with other lifecycle plans (e.g., Software Verification and Validation Plan, Software Configuration Management Plan, Software Training Plan etc.)
c. Establish the SDP prior to starting software development activities.
d. Define activities to be performed, and input and output for each lifecycle process.
e. Specify methods, tools, and techniques including programming languages, computers, compilers, library, and links to be used.
f. Make the level of details such that a Development team can execute the Development plan and carry out software projects.
g. Address technical milestones consistent with the overall project schedule.
h. Provide adequate resources for processing and resolution of all safety issues raised while the Development activities are being performed.
i. Resolve all safety issues through appropriate corrective modifications or mitigation dispositions.
j. Address design change including change process and regression analysis/test.

Additional tasks:

a. Plan project management oversight support.
b. Plan proposal evaluation support.
c. Review and ensure that the Work Breakdown Structure (WBS) represents all of the project scope and captures all deliverables, including internal, external, and interim deliverables. Review and ensure that the WBS decomposes the project scope into a set of deliverables that comprehensively defines the work to be performed.
d. Generate software tool plan, which describes the tools needed to support the software development effort. The plan includes a description of each tool's performance, required inputs and associated tools, outputs generated, needed date, and cost of tool purchase or development.

(2) Development of a concept[2] documentation (i.e., System Requirements Specifications - SyRS)

The concept activity represents the delineation of a specific implementation solution to solve the user's problem. During the concept activity, the system architecture is selected and system requirements are allocated to hardware, software, and user interface components. In developing the concept document (i.e., SyRS), the development or selection of system architectural design and the development of system requirements are key. The objective for this development activity is to generate a SyRS that conforms to the user's needs (e.g., stakeholder's requirements), and pertinent regulations/standards, and using best engineering practices. The SyRS will be updated and revised per the outcome of various analyses and V&V activities. A well-developed SyRS plays a solid starting role for the subsequent lifecycle development activities.

The following are the tasks performed in the development of a SyRS.

a. Develop a specific conceptual solution (e.g., system architecture as described in the concept documentation) based on user needs and acquisition needs.
b. Identify requirements from the customer, the environment, and the experience of the technical community.
c. Identify constraints of interfacing systems and constraints or limitations of proposed implementation solution.
d. Allocate functional and performance requirements (e.g., timing, response time, and throughput) to the hardware, software, and user interfaces.
e. For the internal and external interfaces, specify the data formats, interface protocols, frequency of data exchange at each interface, and other key performance requirements.
f. Develop application-specific requirements such as redundancy, independence (physical, electrical, and communicational independence), diversity and defense-in-

[2] The concept activities are not mentioned and discussed except in IEEE standards such as 1012 and 1074. The detailed discussion of concept activities is only provided in IEEE 1074 among the IEEE standards we have evaluated. In IEEE 1074, a total number of 69 activities in the SPLCP (Software Project Life Cycle Process) are grouped into 17 activity groups, and the Concept Exploration [Section A.2.1 of IEEE 1074] is one of three groups in the pre-development section of the software. The purpose of concept exploration is identification of ideas or needs, selection of potential approaches and performance of feasibility studies to refine and finalize the idea or need. The output of concept exploration will be a Statement of Need that identifies the software idea, need, or desire, the recommended implementation approach, and any data pertinent to a management decision concerning the initiation of the described development effort. The Statement of Need is the basis for the system analysis and the development of software requirements via the System Allocation activity [Section A.2.2 of IEEE 1074]. As a bridge between the concept exploration and the development of software requirements specifications, system allocation activity maps the required functions identified in the concept exploration to software, and when applicable, to hardware and people. The major activities of system allocation include (1) derivation of system functions from system requirements and identification of hardware, software, and operational requirements; (2) development of system architecture; and (3) formulation of software requirements, system interface requirements, and human and hardware requirements (if applicable), i.e., allocation of system requirements to software, interface, human and hardware (if applicable). Additional guidance for software requirements development can be found in IEEE 830.

depth, fail-safe, fault detection, fault isolation, and diagnostic and error recovery (e.g., Oconee SER and IEC 61508).

g. Specify maintenance requirements for the system.
h. Specify migration requirements from an existing system where applicable.
i. Build well-formed requirements.[3]
j. Organize requirements into the concept documentation (i.e., SyRS).
k. Update/revise the concept documentation (i.e., SyRS) per the various analyses activities results [i.e., traceability analysis, criticality analysis, hazard analysis, security analysis, and risk analysis, see Attributes (4), (5), (6), (7), and (8)].
l. Evaluate user documentation[4] for its completeness, correctness, and consistency with respect to requirements for user interface and for any functionality that can be invoked by the user.

(3) Development of Software Requirements Specifications (SRS)

The purpose of developing the requirements (e.g., functionality, capability, interface, qualification, safety, security, human factors, data definitions, user documentation, installation and acceptance, user operation, and user maintenance[5]) of the SRS is to satisfy system requirements and user's needs and ensure correctness, consistency, completeness, accuracy, readability, testability, and robustness in the SRS.

The following tasks are performed in developing SRS.

a. Develop the software requirements according to the system requirements allocated to software taking into consideration the assumptions, constraints, and operating environment for the system[6] (Correctness)
b. Develop the software requirements according to standards, references, regulations, policies, physical laws, and business rules.[7] (Correctness)
c. Develop the sequences of states and state changes (according to logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis.) (Correctness)

[3] IEEE 1233 provides guidance for developing System Requirements Specification (SyRS), which states that SyRS development is an iterative process that includes identification, construction, organization, and presentation of requirements sub-processes. Per IEEE 1233, a well-formed requirement is a statement of system functionality (a capability) that can be validated, and that must be met or possessed by a system to solve a customer problem or to achieve a customer objective, and is qualified by measurable conditions and bounded by constraints.

[4] User documentation refers to the documentation for a product or service provided to the end users. The user documentation is designed to assist end user to use the product or service. The user documentation is a part of the overall product delivered to the customer, which may include user guides, instruction manual or training materials.

[5] IEC 61508 Part 3 Section 7.2 contains detailed requirements for these activities. In general, the detailed requirements should be evaluated against the "attributes".

[6] Section 6.3.1 in DO-178B: Compliance with system requirements.

[7] Section 6.3.1 in DO-178B: Conformance to standards.

d. Develop the flow of data and control to satisfy functionality and performance requirements. (Correctness)
e. Use proper data and format. (Correctness)
f. Document terms and concepts consistently. (Consistency)
g. Develop the SRS such that the function interactions and assumptions are consistent. (Consistency)
h. Develop the external and internal software interface requirements. (Correctness)
i. Develop each interface requirement with the required accuracy. (Accuracy)
j. Maintain internal consistency between the software requirements and external consistency with the system requirements. (Consistency)
k. Include the following elements in the SRS, within the assumptions and constraints of the system:

   i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging).
   ii) Hardware, software, and user-interface descriptions.
   iii) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).
   iv) Critical configuration data.
   v) System, device, and software control (e.g., initialization, transaction and state monitoring, and self-testing). (Completeness)

l. Specify logic, computational, and interface precision (e.g., truncation and rounding) satisfying therequirements in the system environment. (Accuracy)
m. Model physical phenomena to conform with system accuracy requirements and physical laws. (Accuracy)
n. Develop algorithms with adequate accuracy especially in the area of discontinuities (e.g., different plant operating modes) (DO-178C). (Accuracy)
o. Develop legible, understandable, and unambiguous (i.e., having one and only one interpretation) software requirements to the intended audience. (Readability)
p. Define all acronyms, mnemonics, abbreviations, terms, and symbols used in software requirements. (Readability)
q. Specify objective acceptance criteria for validating the requirements of the SRS by testing. (Testability)[8]
r. Evaluate and eliminate any potential conflicts between the requirements[9] and the hardware/software features of target computer, especially, system response times and input/output hardware. (Compatibility with the target computer [DO-178C)]

[8] IEC 60880 states that the SRS shall be unequivocal, testable or verifiable, and achievable. Oconee only states in the SRS that acceptance criteria can be established. The V&V report states “SRS contains objective acceptance criteria required for testability.” The SER states that these are reviewed and it is concluded that the system is testable.

[9] Original wording in DO-178B is high level requirements, that is, produced directly through system requirements and system architecture.

s. Specify the behavior of the software in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior. Of particular concern is the behavior of the software in the presence of unexpectedly high or low rates of message traffic. (Robustness)
t. Generate the SRS using best engineering practice and guidance as outlined in pertinent standards (i.e., IEEE Std 830).
u. Update/revise the SRS per the outcome of the various analyses results[i.e., traceability analysis, criticality analysis, hazard analysis, security analysis, risk analysis, and various additional analysis activities, see Attributes (4), (5), (6), (7), (8), and (13)].

Additional tasks:

a. Address the software tool qualification to assure that the tool is functioning properly and it does not mask errors that it was designed to find.

(4) Traceability Analysis- Requirements Specifications Phase

The requirements traceability analysis is part of the development processes that ensure system and software requirements are complete, testable, and implemented correctly. Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specifications, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)[10].

The traceability analysis of system and software requirements is basically to trace the system requirements stated in the SyRS to the user's needs and acquisition needs and software requirements as stated in the SRS to the system requirements in the SyRS backward, and the user's needs and acquisition needs to the system requirements in the SyRS and the system requirements in the SyRS to the software requirements stated in the SRS forward.

The development tasks for the traceability analysis are as follows:

a. Establish a traceability matrix that is consistent with the development process.
b. Identify all system and software requirements.
c. Trace the system requirements (as defined in the SyRS) to acquisition needs (backward traceability) and the acquisition needs to the system requirements (forward traceability).

[10] This definition is summarized by Gotel at al "An Analysis of the Requirements Traceability Problem", Proceedings of First International Conference on Requirements Engineering, 1994, which is derived from IEEE Std 830 definition "A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation".

d. Trace the software requirements (SRS) to the corresponding system requirements (e.g., Concept Documentation SyRS) (backward traceability) and the system requirements to the corresponding software requirements (forward traceability).
e. Analyze identified relationships for correctness, consistency, completeness, and accuracy. The task criteria are as follows:

   i) The relationships between each software requirement and its system requirement should be correct. (correctness of the relationship/mapping) (Correctness)
   ii) The relationships between the software and system requirements should be specified to a consistent level of detail. (Consistency)
   iii) Every software requirement should have sufficient detail to show conformance to the system requirement. (Completeness)
   iv) All system requirements related to software should be traceable to software requirements. (Completeness)
   v) The system performance and operating characteristics should be accurately specified by the traced software requirements. (Accuracy)

f. Document the traceability analysis results and generate the traceability analysis report.

(5) Criticality Analysis- Requirements Specifications Phase

Criticality analysis is used to assign integrity/criticality level, which in turn is used to determine the rigor and effort of the development activities at each stage of the development lifecycle. The analysis determines how a system, system element, or component can potentially cause undesirable consequences. In this phase of development, an initial criticality analysis is performed to assign integrity level to system and software components. The criticality analysis report can be updated as the development process progresses and more detailed information becomes available.

For system requirements:

a. Determine whether integrity levels are established for requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions.
b. Verify that the assigned integrity levels are correct. If integrity levels are not assigned, then assign integrity levels to the system requirements.
c. Document the integrity level assigned to individual components (e.g., requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions). For software development planning purposes, the system should be assigned the same integrity level as the highest level assigned to any individual element.
d. Verify whether any component can influence the individual components assigned a higher software integrity level, and if such conditions exist, then assign that component the same higher integrity level.

For software requirements:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report using the SRS.

b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, and other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(6) Hazard Analysis- Requirements Specifications

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012).

The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. Once the end hazardous conditions are identified, the hazard analysis typically uses proven analysis approaches/tools. In this phase of development, an initial hazard analysis is performed to identify system and software hazards. The hazard analysis is repeated in each life cycle phase and accounts for further elaboration of designs, changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012).

For system hazard analysis:

Analyze the potential hazards to and from the conceptual system (e.g., as documented in the SyRS). The analysis includes the following tasks:

a. Identify the potential system hazards
b. Assess the consequences of each hazard
c. Assess the probability of each hazard
d. Identify mitigation strategies for each hazard
e. Document the software hazard analysis results and generate report

For software hazard analysis, analyze software hazards (i.e., software conditions, including software faults and incorrect software requirements, which can lead to an accident[11]). The analysis includes the following[12]:

a. Identify potential system hazards contributed by software requirements as documented in the SRS.

[11] ISO/IEC/IEEE 24765:2010 - Systems and software engineering vocabulary.

[12] The activity descriptions are clarified by explicitly stating that the hazard analysis is performed for system hazards contributions from software. The clarification is necessary to avoid the confusion of software hazard analysis with system hazard analysis. This change is based on the discussion in Annex J of IEEE Std. 1012.

b. Assess the consequences of each system hazard taking into consideration the identified software hazards.
c. Assess the probability of each system hazard taking into consideration the identified software hazards.
d. Identify mitigation strategies for each hazard.
e. Document the software hazard analysis results and generate report.

All requirements for fault tolerance and failure modes should be fully specified for each operating mode. Software requirements for handling both hardware and software failures should be provided, including requirements for analysis of and recovery from computer system failures. Requirements for on-line in-service testing and diagnostics should be provided.[13]

Note 1: Section 3 of NUREG/CR-6430 which is on requirement hazard analysis provides guide phrases for examining requirements. Page 25 states "A major impact of the results from the software hazards analysis is on changes to the software requirements specifications for the purpose of eliminating identified hazards that are affected by the software or that are not adequately managed by the software."

Note 2: Annex J of IEEE Std. 1012 states: "The hazard analysis may be performed by an organization within the project such as systems engineering, reliability, safety, or V&V. In any case, V&V reviews the hazard analysis for completeness and usability, and it assures that the stated hazards and contributors are clearly identified to sufficient detail to affect engineering and mitigation activities properly and to develop V&V plans and evaluation criteria."

Note 3: IEEE 7-4.3.2 states that the software requirements hazards analysis includes evaluation of software and interface requirements for deficiencies that can contribute to hazards. Examples of software hazards include logic errors, incorrect loop iterations, and using wrong variables.

Note 4: IEEE Std. 7-4.3.2 contains more detailed discussion of software hazard analysis. However, those discussions can, at a high level, be summarized by the four main required activities described in the checklist (hazard identification, assess consequences, assess probability, identify mitigation strategies). No text from that standard is used in the checklist.

Note 5: NUREG/CR-6430 describes one method in which hazard analysis can be performed. It is felt that the activities described in that report can also generally be categorized by the four activities in the checklist. Therefore, no text from that report is used in the checklist.

Note 6: MIL-STD-882E contains discussions on hazard analysis. The overall approach and goal are similar to the description in the checklist. No specific text from that standard is used in the checklist.

[13] BTP 7-14 is consulted for its requirements related to safety and hazards. The discussion of fault tolerance and failure modes are found to be relevant and so are included as part of the hazard analysis activity.

Note 7: DO-178C specifies that high level system requirements that are allocated to the software should avoid introducing hazards. This requirement is covered by activity A. in the checklist.

(7) Security Analysis- Requirements Specifications Phase

The objective of security analysis is to ensure that system and software security vulnerabilities are identified, and required threat controls and safeguards of system and software from accidental or malicious access, use, modification, destruction, or disclosure are addressed in the system and software requirements. In this phase of development, a security analysis of the system and software requirements is performed.

For system requirements:

a. Review the system owner's definition of an acceptable level of security risk.
b. Analyze the system concept (e.g., as documented in the SyRS) from a security perspective and assure that potential security risks with respect to confidentiality (disclosure of sensitive information/data), integrity (modification of information/data), availability (withholding of information or services), and accountability (attributing actions to an individual/process) have been identified. Include an assessment of the sensitivity of the information/data to be processed.
c. Analyze the security risks introduced by the system itself as well as those associated with the environment with which the system interfaces.
d. Verify that the system security requirements will mitigate the identified security risks introduced by the system concept.
e. Document the system security analysis results and generate report.

For software requirements:

a. Determine that the security requirements identified in the SRS address the security risks introduced by the system concept.
b. Verify that the software security requirements will mitigate the identified security risks to an acceptable level.
c. Document the software security analysis results and generate report.

(8) Risk Analysis- Requirements Specifications Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the system or the successful development of the system. In this phase of development, a risk analysis of the system and software requirements is performed. The risk analysis is performed continuously throughout the development life cycle.

Procedure of risk analysis is as follows:

a. In terms of requirements in the SyRS, and SRS, identify risk (technical and managerial risks[14]) contributors, i.e., the initiating events, hazards, threats, or situations that create risks;
b. Estimate the probability of occurrence, the consequences for each risk, and the expected timing of the risk; and
c. Evaluate each risk or defined combination of risks against its applicable threshold, generation of alternatives to treat risks above their risk thresholds, and making recommendations for treatment (elimination, reduction, or mitigation of risks) based on a priority order.
d. Document the risk analysis results and generate report.

(9) Software Qualification Test Plan Generation

Software qualification testing is performed on a complete, integrated system (or a system component such as software) to evaluate the system's compliance with its specified requirements. Qualification test plans can be generated once the system/software requirements are available.

Software Qualification Test Plan Generation task is as follows:

a. Plan software qualification testing to validate software requirements.
b. Plan tracing of system requirements to software qualification test designs, cases, procedures, and results.
c. Plan documentation of software qualification test designs, cases, procedures, and results.
d. The software qualification test plan addresses the following:

   i) Conformance to all system requirements (e.g., functional, performance, security, operation, and maintenance) as complete software end items in the system environment.
   ii) Adequacy of user documentation (e.g., training materials and procedural changes).
   iii) Performance at boundaries (e.g., data and interfaces) and under stress conditions.

e. Verify that the software qualification test plan satisfies the following criteria:

   i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
   ii) Test coverage of system requirements.

f. Validate that the software qualification test plan satisfies the following criteria:

   i) Appropriateness of test methods and standards used.
   ii) Conformance to expected results.
   iii) Feasibility of system qualification testing.

[14] The example technical and management risks may include, e.g., the possibility of negative consequences to the operation of the system and the delivery of the project, as indicated in Annex J of IEEE 1012.

iv) Feasibility and testability of operation and maintenance requirements.

g. Generate software qualification plan.

(10) Software Acceptance Test Plan Generation

Software acceptance testing is conducted to determine whether or not a system (or a system component such as software) satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. Acceptance test plans can be generated once the system/software requirements are available.

Software Acceptance Test Plan Generation task is as follows:

a. Plan software acceptance testing to validate that the software correctly implements system and software requirements in an operational environment.
b. Plan tracing of test requirements to test software acceptance design, cases, procedures, and execution results.
c. Plan documentation of test tasks and results.
d. The software acceptance test plan addresses the following:

   i) Conformance to acceptance requirements in the operational environment.
   ii) Adequacy of user documentation.

e. Verify that the software acceptance test plan satisfies the following criteria:

   i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008 [B2]).
   ii) Test coverage of acceptance requirements.

f. Validate that the software acceptance test plan satisfies the following criteria:

   i) Conformance to expected results.
   ii) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

g. Employ the user documentation in planning an acceptance test that is representative of the operational environment.
h. Generate software acceptance plan.

(11) Configuration Management- Requirements Specifications Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system, software and hardware product functionality, tracking program versions, generating baselines (including parameters and settings), and managing changes. The configuration

management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs.

In the Concept and Requirements phase CM, the development organization focuses on defining a configuration management strategy and ensures that the phase specific configuration items (i.e., SyRS, SRS etc.) are under controlled.

a. Define a configuration management strategy so that configuration controls are in place to maintain and document configuration item baselines with unique identifiers. The strategy should include notification to the V&V effort for all changes made to the configuration item baselines.
b. Define and document items requiring configuration management. The items controlled should include enabling systems, tools and processes that are integral to system development and life cycle support that will be subject to V&V in order to demonstrate conformance to this Standard.
c. The status of items under configuration management is made available throughout the life cycle. Provisions should be included to assure that the V&V effort receives the current status for all configuration items required in the Concept and Requirements phase including but not limited to SyRS, SRS, and system architectural drawings.

(12) Reviews and Audit - Requirements Specifications Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028). In this phase of software development, the tasks include establishing a Review and Audit process and plan, applying it to SyRS and SRS, and tracking any anomalies.

The reviews and audit tasks are as follows:

a. Generate a reviews/audit plan in which review team and each team member's responsibilities, and review schedule are defined.
b. Define reviews/audit process including protocols for interfacing with other organizations (i.e., Project, V&V, and QA).
c. Document reviews/audit results (e.g., on the SyRS and SRS).
d. Establish an anomaly tracking system for tracking anomalies during the reviews/audit process.
e. Document improvement and/or lessons learned as a result of reviews/audit.

B.2 Attributes of Software Requirements Specifications V&V Activities

Software V&V is the process of determining whether the requirements for a system or component are complete and correct; that the products of each lifecycle phase fulfill the requirements or conditions imposed by the previous phase; and that final systems or components comply with specific requirements.

Per IEEE Std 1012, the software V&V is a technical discipline of systems engineering. The purpose of software V&V is to help the development organization build quality into the software during its development lifecycle. V&V processes offer an objective assessment of software

products and processes through the software (SW) lifecycle. The assessment demonstrates whether the SW requirements are correct, complete, accurate, consistent, and testable.

The requirement specifications V&V activities include the following attributes with more detailed activities specified in the table below.

(1) Software V&V Planning
(2) Concept Documentation Evaluation
(3) Hardware/Software/User Requirements Allocation Analysis
(4) Software Requirements Evaluation
(5) Interface Analysis V&V- Requirement Specifications Phase
(6) Traceability Analysis V&V- Requirement Specifications Phase
(7) Criticality Analysis V&V- Requirement Specifications Phase
(8) Hazard Analysis V&V- Requirement Specifications Phase
(9) Security Analysis V&V- Requirement Specifications Phase
(10) Risk Analysis V&V- Requirement Specifications Phase
(11) V&V System/Software Qualification Test Plan Generation
(12) V&V Software Acceptance Test Plan Generation
(13) Configuration Management Assessment- Requirement Specifications Phase
(14) Reviews and Audit V&V- Requirement Specifications Phase
(15) V&V Requirements Specifications Phase Activity Summary Report Generation

The table below is used to evaluate the quality of the Verification & Validation (V&V) for safety-related system/software at the Concept and Requirements phase. A few exemplary *additional* activities are provided for pertinent attributes. These additional activities are expected to increase the quality score of the associated attributes in the BBN model.

Note:
For most of the analysis tasks, for example, traceability analysis, criticality analysis, hazard analysis, security analysis, and risk analysis, if both the Development and V&V organization independently performs them, the quality weighting should be higher than either one of them performs and the other review and verify.

(1) Software V&V Planning (IEEE Std 1012, and IEC 60880):

During the software V&V process, a software V&V plan will be developed. The plan outlines the management, implementation and resource characteristics. The management characteristics include purpose, organization, oversight, and risks. The implementation characteristics include measurement, procedures, and schedule. The resource characteristics include methods/tools and standards.

Software V&V Planning task is as follows (mainly adopted from IEC 60880):

a. Develop and generate a software V&V plan that is consistent with the V&V process defined in IEEE Std 1012 and the lifecycle process defined in IEEE Std 1074.[15]
b. Establish the software V&V plan[16] prior to starting software V&V activities.
c. Ensure the plan documents all the criteria, the techniques and tools to be utilized in the V&V process.
d. Describe the activities to be performed to evaluate each item of software, each tool involved in the software development process, and each phase to show whether the software requirements specification is met.
e. Make the level of detail such that a V&V team can execute the V&V plan and reach an objective judgment on whether or not the software meets its requirements.
f. The V&V plan addresses:
    - i) Selection of V&V strategies, either systematic, random or both, with test case selection according to either required functions, special features of program structure, or both;
    - ii) Selection and utilization of the software V&V tools;
    - iii) Execution of V&V;
    - iv) Documentation of V&V activities;
    - v) Evaluation of V&V results gained from V&V equipment directly and from tests, evaluation of whether the safety requirements are met.
g. The tests performed should extensively exercise the software. Among the criteria required in the plan, test coverage[17] criteria should be considered of prime importance.

[15] Per BTP 7-14, the NRC staff's acceptance of software for safety system functions is based upon (1) confirmation that acceptable plans were prepared to control software development activities, (2) evidence that the plans were followed in an acceptable software lifecycle, and (3) evidence that the process produced acceptable design output. To reach software high reliability and confidence, V&V activities must start with acceptable plans that is consistent with IEEE Std 1012 and IEEE Std 1074.

[16] IEC 60880 uses "verification plan" to mean "V&V plan" in the IEEE Std 1012 terminology. Generally, the "verification" word used in the IEC 60880 is replaced with "verification & validation" to be consistent with the US standards.

[17] IEEE Std 0610.12 defines test coverage as "the degree to which a given test or set of tests addresses all specified requirements for a given system or component." IEEE Std 12207.1 refers "test coverage as breadth and depth for assuring sufficiency of testing." When discussing "Design Attributes to Eliminate Consideration of CCF", BTP 7-19 defines testability as "A system is sufficiently simple such that every possible combination of inputs and every possible sequence of device states are tested and all outputs are verified for every case (100% tested)."

h. The V&V plan identifies any objective evidence required to confirm the extent of testing. For that purpose the choice on the test coverage criteria according to the design is justified and documented.
i. Provide adequate resources for the processing and resolution of all safety issues raised during the V&V activities performed either during software development by the supplier or by a third-party assessment.
j. Resolve all safety issues through appropriate corrective modifications or mitigating dispositions.

Additional tasks:

a. Plan project management oversight support.
b. Plan proposal evaluation support.
c. Review and verify that the Work Breakdown Structure (WBS) represents all of the project scope and captures all deliverables, including internal, external, and interim deliverables. Review and ensure that the WBS decomposes the project scope into a set of deliverables that comprehensively defines the work to be performed.
d. Generate software tool plan which describes the tools needed to support the software V&V effort. The plan includes a description of each tool's performance, required inputs and associated tools, outputs generated, needed date, and cost of tool purchase or development.
e. Ensure that selected tools addressed in the V&V plan be qualified in accordance with guidance provided in IEEE Std 7-4.3.2.

(2) Concept Documentation Evaluation

The concept activity represents the delineation of a specific implementation solution to solve the user's problem. The objective of the V&V Concept Documentation Evaluation is to ensure that concept documents (e.g., SyRS) produced by the Development team conform to the user's needs (e.g., stakeholder's requirements), and pertinent regulations/standards, and using best engineering practices. The concept documents will be updated and revised per the outcome of various analyses and V&V activities.

The V&V task is as follows:

a. Validate that the concept documentation (SyRS) satisfies user needs and is consistent with acquisition needs.
b. Validate constraints of interfacing systems and constraints or limitations of proposed approach.
c. Analyze system requirements and validate that the following satisfy user needs:
    i) System functions.
    ii) End-to-end system performance.
    iii) Feasibility and testability of the functional requirements.
    iv) System architecture design.
    v) Operation and maintenance requirements and environments.
    vi) Migration requirements from an existing system where applicable.

(3) Hardware/Software/User Requirements Allocation Analysis[18]

During the concept activity, the system architecture is selected and system requirements are allocated to hardware, software, and user interface components. The objective of hardware/software/user requirements allocation analysis is to verify the correctness, accuracy, and completeness of the concept requirement allocation to the hardware, software, and user interfaces against user needs.

The tasks are as follows:

a. Verify that performance requirements (e.g., timing, response time, and throughput) allocated to the hardware, software, and user interfaces satisfy user needs. (Correctness)
b. Verify that the internal and external interfaces specify the data formats, interface protocols, frequency of data exchange at each interface, and other key performance requirements to demonstrate satisfaction of user requirements. (Accuracy)
c. Verify that application-specific requirements such as functional diversity, fault detection, fault isolation, and diagnostic and error recovery satisfy user needs. (Completeness)
d. Verify that the user's maintenance requirements for the system are completely specified. (Completeness)
e. Verify that the migration from existing system and replacement of the system satisfies user needs. (Completeness)
f. Review and verify user documentation[19] for its completeness, correctness, and consistency with respect to requirements for user interface and for any functionality that can be invoked by the user.[20]

(4) Software Requirements Evaluation

The purpose of the V&V requirements evaluation is to review and verify the SRS to ensure that the SRS satisfies system requirements. Through the evaluation of the software requirements (e.g., functional, capability, interface, qualification, safety, security, human factors, data definitions, user documentation, installation and

[18] This task is listed in accordance with the IEEE Std 1012. However, the task is not listed as an explicit task in the Development node since activities in Hardware/Software/User Requirements Allocation Analysis are embedded in the SyRS and SRS development activities

[19] User documentation refers to the documentation for a product or service provided to the end users. The user documentation is designed to assist end user to use the product or service. The user documentation is a part of the overall product delivered to the customer, which may include user guide, instruction manual or training materials.

[20] IEEE Std 1012 lists "User Documentation Evaluation" as an optional task. However, Regulatory Guide 1.168 states that "User documentation is important to the safe operation and proper maintenance of safety system software. The requirements of Criterion III, 'Design Control,' for correctly translating the design basis of safety system software into specifications, procedures, drawings, and instructions, apply to software documentation, including user documentation."

acceptance, user operation, and user maintenance), V&V ensures the correctness, consistency, completeness, accuracy, readability, and testability of the SRS.

The specific task is as follows:

a. Verify and validate that the software requirements satisfy the system requirements allocated to software within the assumptions, constraints, and operating environment for the system[21]. (Correctness)
b. Verify that the software requirements comply with standards, references, regulations, policies, physical laws, and business rules[22]. (Correctness)
c. Validate the sequences of states and state changes using logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis. (Correctness)
d. Validate that the flow of data and control satisfy functionality and performance requirements. (Correctness)
e. Validate data usage and format. (Correctness)
f. Verify that all terms and concepts are documented consistently. (Consistency)
g. Verify that the function interactions and assumptions are consistent and satisfy system requirements and acquisition needs. (Consistency)
h. Verify that there is internal consistency between the software requirements and external consistency with the system requirements. (Consistency)
i. Verify that the following elements in the SRS are within the assumptions and constraints of the system: (Completeness)

   i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging).
   ii) Hardware, software, and user-interface descriptions.
   iii) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).
   iv) Critical configuration data.
   v) System, device, and software control (e.g., initialization, transaction and state monitoring, and self-testing).

j. Validate that the logic, computational, and interface precision (e.g., truncation and rounding) satisfy the requirements in the system environment. (Accuracy)
k. Validate that the modeled physical phenomena conform to system accuracy requirements and physical laws. (Accuracy)
l. Algorithms are accurate: ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities (DO-178C). (Accuracy)
m. Verify that the documentation is legible, understandable, and unambiguous to the intended audience.) (Readability *(or style [BTP 7-14]))*
n. Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms, and symbols. (Readability *(or style [BTP 7-14]))*

[21] Section 6.3.1 in DO-178C: Compliance with system requirements.

[22] Section 6.3.1 in DO-178C: Conformance to standards.

o. Verify that that there are objective acceptance criteria for validating the requirements of the SRS. (Testability)
p. Ensure that no conflicts exist between the high-level requirements and the hardware/software features of target computer, especially, system response times and input/output hardware. [Compatibility with the target computer (DO-178C)]
q. Ensure that each high-level requirement and low-level requirement can be verified. (Verifiability) (DO-178C)
r. Ensure that the software low-level requirements satisfy the high-level requirements and that derived requirements and the design basis for their existence are correctly defined. (Compliance of low-level requirements to high-level requires)
s. Each requirement, and all requirements taken together, have one and only one interpretation. (Unambiguity (BTP 7-14))
t. The behavior of the software in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior are fully specified. (Robustness (BTP 7-14))
u. Of particular concern is the behavior of the software in the presence of unexpectedly high or low rates of message traffic.

(5) Interface Analysis[23] V&V-Requirement Specifications

The objective of the V&V interface analysis is to verify and validate that the requirements for software interfaces with hardware, user, operator, and other systems are correct, consistent, complete, accurate, and testable.

The task is as follows:

a. Validate the external and internal system and software interface requirements. (Correctness)
b. Verify that the interface descriptions are consistent within the SRS. (Consistency)
c. Verify that each interface is described and includes data format and performance criteria (e.g., timing, bandwidth, accuracy, safety, and security). (Completeness)
d. Verify that each interface provides information with the required accuracy. (Accuracy)
e. Verify that there are objective acceptance criteria for validating the interface requirements. (Testability)

(6) Traceability Analysis V&V- Requirement Specifications Phase

The requirements traceability analysis is part of the V&V processes that ensure system and software requirements are complete, traceable, testable, and implemented correctly. Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins,

[23] This task is listed in accordance with the IEEE Std 1012. However, the task is not listed as an explicit task in the Development node since activities in the Interface Analysis are embedded in the SRS development activities.

through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases[24]). The traceability analysis of system and software requirements is basically to trace the system requirements stated in the SyRS to the user's needs and acquisition needs and software requirements as stated in the SRS to the system requirements in the SyRS backward, and the user's needs and acquisition needs to the system requirements in the SyRS and the system requirements in the SyRS to the software requirements stated in the SRS forward.

The V&V tasks for the traceability analysis are as follows:

a. Establish a traceability matrix that is consistent with the V&V process (as defined in the V&V plan).
b. Identify all system and software requirements.
c. Trace the system requirements (as defined in the SyRS) to acquisition needs (backward traceability) and the acquisition needs to the system requirements (forward traceability).
d. Trace the software requirements (as defined in the SRS) to the corresponding system requirements (e.g., Concept Documentation SyRS) (backward traceability) and the system requirements to the corresponding software requirements (forward traceability).
e. Analyze identified relationships for correctness, consistency, completeness, and accuracy. The task criteria are as follows:

   i) The relationships between each software requirement and its system requirement should be correct. (correctness of the relationship/mapping) (Correctness)
   ii) The relationships between the software and system requirements should be specified to a consistent level of detail. (Consistency)
   iii) Every software requirement should have sufficient detail to show conformance to the system requirement. (Completeness)
   iv) All system requirements related to software should be traceable to software requirements. (Completeness)
   v) The system performance and operating characteristics should be accurately specified by the traced software requirements. (Accuracy)

f. Document the traceability analysis results and generate the traceability analysis report.

[24] This definition is summarized by Gotel at al "An Analysis of the Requirements Traceability Problem", Proceedings of First International Conference on Requirements Engineering, 1994, which is derived from IEEE Std 830 definition "A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation".

(7) Criticality Analysis V&V- Requirement Specifications Phase

Criticality analysis is used to assign integrity/criticality level, which in turn is used to determine the rigor and effort of the V&V activities at each stage of the V&V lifecycle process. The analysis determines how a system, system element, or component can potentially cause undesirable consequences. Criticality analysis report can be updated as the V&V process progresses and more detailed information becomes available. Since the system being evaluated is assumed to be a safety-related system of a nuclear power plant, the system criticality level should be 4, while not every lower level component has to be of the same level. In this phase of development, an initial criticality analysis is performed to assign integrity level to system and software components. The criticality analysis report can be updated as the development process progresses and more detailed information becomes available.

For system requirements:

a. Determine whether integrity levels are established for requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions.
b. Verify that the assigned integrity levels are correct. If integrity levels are not assigned, then assign integrity levels to the system requirements.
c. Document the integrity level assigned to individual components (e.g., requirements, detailed functions, software modules, hardware elements, subsystems, or other partitions). For V&V planning purposes, the system should be assigned the same integrity level as the highest level assigned to any individual element.
d. Verify whether any component can influence the individual components assigned a higher software integrity level, and if such conditions exist, then assign that component the same higher integrity level.

For software requirements:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report using the SRS.
b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, and other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(8) Hazard Analysis V&V- Requirement Specifications Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012).

The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. Once the end hazardous conditions are identified, the hazard analysis typically uses proven analysis approaches/tools to identify the contributors or combination of contributors (within the bounds of required system fault tolerance) to reaching the hazardous condition. The contributing causes may be the result of hardware or software faults, human actions (e.g., procedures), or hostile environmental conditions. In this phase

of development, an initial hazard analysis is performed to identify system and software hazards. The hazard analysis is repeated in each lifecycle phase and accounts for further elaboration of designs, changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012).

For system hazard analysis:

Analyze the potential hazards to and from the conceptual system (e.g., as documented in the SyRS). The analysis includes:

a. Identify the potential system hazards
b. Assess the consequences of each hazard
c. Assess the probability of each hazard
d. Identify mitigation strategies for each hazard
e. Document the system hazard analysis results and generate report.

For software hazard analysis, analyze software hazards (i.e., software conditions, including software faults and incorrect software requirements, which can lead to an accident). The analysis performs the following[25]:

a. Identify potential system hazards contributed by software requirements as documented in the SRS.
b. Assess the consequences of each system hazard taking into consideration the identified software hazards.
c. Assess the probability of each system hazard taking into consideration the identified software hazards.
d. Identify mitigation strategies for each hazard.
e. Document the software hazard analysis results and generate report.

(9) Security Analysis V&V- Requirements Specifications Phase

The objective of security analysis is to ensure that system and software security vulnerabilities are identified, and required threat controls and safeguards of system and software from accidental or malicious access, use, modification, destruction, or disclosure are addressed in the system and software requirements. In this phase of development, a security analysis of the system and software requirements is performed.

For system requirements:

a. Review the system owner's definition of an acceptable level of security risk.
b. Analyze the system concept (e.g., as documented in the SyRS) from a security perspective and assure that potential security risks with respect to confidentiality (disclosure of sensitive information/data), integrity (modification of information/data),

[25] The activity descriptions are clarified by explicitly stating that the hazard analysis is performed for system hazards contributions from software. The clarification is necessary to avoid the confusion of software hazard analysis with system hazard analysis. This change is based on the discussion in Annex J of IEEE Std. 1012.

availability (withholding of information or services), and accountability (attributing actions to an individual/process) have been identified. Include an assessment of the sensitivity of the information/data to be processed.

c. Analyze the security risks introduced by the system itself as well as those associated with the environment with which the system interfaces.
d. Verify that the system security requirements will mitigate the identified security risks to an acceptable level.
e. Document the system security analysis results and generate report.

For software requirements:

a. Determine that the security requirements identified in the SRS address the security risks introduced by the system concept.
b. Verify that the software security requirements will mitigate the identified security risks to an acceptable level.
c. Document the software security analysis results and generate report.

(10) Risk Analysis V&V- Requirements Specifications Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the system or the successful development of the system. In this phase of development, a risk analysis of the system and software requirements is performed. The risk analysis should be performed continuously throughout the development lifecycle.

Procedure of risk analysis is as follows:

a. In terms of requirements in the SyRS, and SRS, identify risk (technical and managerial risks[26]) contributors, i.e., the initiating events, hazards, threats, or situations that create risks;
b. Estimate the probability of occurrence, the consequences for each risk, and the expected timing of the risk; and
c. Evaluate each risk or defined combination of risks against its applicable threshold, generation of alternatives to treat risks above their risk thresholds, and making recommendations for treatment (elimination, reduction, or mitigation of risks) based on a priority order.
d. Document the risk analysis results and generate report.

(11) V&V Software Qualification Test Plan Generation

- **V&V software qualification testing is performed on a complete, integrated system (or a system component such as software) to evaluate the system's**

[26] The example technical and management risks may include, e.g., the possibility of negative consequences to the operation of the system and the delivery of the project, as indicated in Annex J of IEEE 1012.

compliance with its specified requirements. Qualification test plans can be generated once the system/software requirements are available.

V&V Software Qualification Test Plan Generation task is as follows:

a. Plan software qualification testing to validate software requirements.
b. Plan tracing of system requirements to software qualification test designs, cases, procedures, and results.
c. Plan documentation of software qualification test designs, cases, procedures, and results.
d. The software qualification test plan addresses the following:

   i) Conformance to all system requirements (e.g., functional, performance, security, operation, and maintenance) as complete software end items in the system environment.
   ii) Adequacy of user documentation (e.g., training materials and procedural changes).
   iii) Performance at boundaries (e.g., data and interfaces) and under stress conditions.

e. Verify that the software qualification test plan satisfies the following criteria:

   i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
   ii) Test coverage of software requirements.

f. Validate that the software qualification test plan satisfies the following criteria:

   i) Appropriateness of test methods and standards used.
   ii) Conformance to expected results.
   iii) Feasibility of software qualification testing.

   iv) Feasibility and testability of operation and maintenance requirements.

g. Generate and/or update software qualification plan.

(12) V&V Software Acceptance Test Plan Generation

V&V software acceptance testing is conducted to determine whether or not a system (or a system component such as software) satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. Acceptance test plans can be generated once the system/software requirements are available.

V&V Software Acceptance Test Plan Generation task is as follows:

a. Plan software acceptance testing to validate that the software correctly implements system and software requirements in an operational environment.
b. Plan tracing of test requirements to test software acceptance design, cases, procedures, and execution results.
c. Plan documentation of test tasks and results.
d. The software acceptance test plan addresses the following:

   i) Conformance to acceptance requirements in the operational environment.
   ii) Adequacy of user documentation.

e. Verify that the software acceptance test plan satisfies the following criteria:

   i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008 [B2]).
   ii) Test coverage of acceptance requirements.

f. Validate that the software acceptance test plan satisfies the following criteria:

   i) Conformance to expected results.
   ii) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

g. Generate software acceptance plan.

Additional tasks:

h. Employ the user documentation in planning an acceptance test that is representative of the operational environment.

(13) Configuration Management Assessment- Requirement Specifications Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The V&V processes use and provide inputs to the Configuration Management Process throughout theproject lifecycle. The V&V processes assess the Configuration Management Process to assure that thefollowing occur:

a. A configuration management strategy is defined for the program that will assure configuration controls are in place to maintain and document configuration item baselines with unique identifiers. The strategy should include notification to the V&V effort for all changes made to the configuration item baselines.
b. Items requiring configuration management are defined and documented. The items controlled should include enabling systems, tools, and processes that are integral to system development and lifecycle support that will be subject to V&V to demonstrate conformance to this standard.
c. The status of items (e.g., SyRS, SRS, and V&V products in Concept and Requirements phase) under configuration management is made available throughout the lifecycle. Provisions should be included to assure that the V&V effort receives the current status for all configuration items.

If Configuration Management issues are discovered during the V&V effort that indicate configuration baselines are not established or controlled or if the configuration of released items is not controlled, the issues should be documented and provided to the system developer and to the V&V customer for resolution.

(14) Reviews and Audit V&V- Requirement Specifications Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028).

The V&V reviews and audit tasks are as follows:

a. Generate a reviews/audit plan in which review team and each team member's responsibilities, and review schedule are defined.
b. Define a reviews/audit process including protocols for interfacing with other organizations (i.e., Project, Development, QA etc.).
c. Document reviews/audit results (e.g., on the SyRS, SRS, and V&V products).
d. Establish an anomaly tracking system for tracking anomalies during the reviews/audit process.
e. Document improvement and/or lessons learned as a result of reviews/audit.

(15) V&V Requirements Specifications Phase Activity Summary Report Generation

V&V Requirements Specifications Phase activity summary report summarizes the results of V&V tasks performed in the phase.

Specific task for the V&V Requirements Specifications Phase Activity Summary Report Generation is as follows:

a. Document software V&V activities conducted in the Requirements Specifications phase
b. Provide background information of standards, and source documents with revisions.
c. Provide confirmation that the V&V plan was followed.
d. Document the individual who performed the task, the portion of the task performed by the individual (if multiple performers), and the date that the task was performed.
e. Summarize anomaly issues identified, the resolution process of them, and recommendations whether to proceed to the Design phase of the software development lifecycle.

B.3 Attributes of Software Design Activities

The objective of Software Design is to translate the software requirements specifications (SRS) into a software architecture description (SAD) and a software design description (SDD). The design includes the databases and the system's interfaces (e.g., hardware, operator/user, software components, and subsystems). The Software Design activity addresses software architectural design and its detailed design. After the design activities are done, software implementation starts.

Various analysis activities including traceability, hazard, criticality, security, and risk will ensure that the requirements are traceable to the design in both the forward and backward directions, no new hazard are introduced into the design, critical and security requirements are addressed in the design, and risk can be minimized as the lifecycle process proceeds.

With the available SRS and SDD, the software component and integration plan can be prepared. Meanwhile, all the test design activities (i.e., software component, its integration, qualification and acceptance testing) can be commenced.

The primary risks of not creating and documenting a disciplined specification of software design are that it may be impossible to be sure that all requirements are implemented in the design, and that no design elements exist that are not required. Either of these cases can create a hazard. (CR-6101)

The software design activities include the following attributes with more detailed activities specified in the table below.

(1) Development of a Software Architecture Description
(2) Development of Software Design Description
(3) Traceability Analysis - Design Phase
(4) Criticality Analysis- Design Phase
(5) Hazard Analysis- Design Phase
(6) Security Analysis- Design Phase
(7) Risk Analysis- Design Phase
(8) Software Component Test Plan Generation
(9) Software Integration Test Plan Generation
(10) Software Component Test Design Generation
(11) Software Integration Test Design Generation
(12) Software Qualification Test Design Generation
(13) Software Acceptance Test Design Generation
(14) Configuration Management- Design Phase
(15) Reviews and Audit- Design Phase

(1) Development of a Software Architecture Description (SAD)

The software architecture provides the high level structures of a software system. It can be defined as the set of structures needed to reason about the software system, which comprise the software elements, the relations between them, and the properties of both elements and relations.[27]

The software architecture design documents software architecture for facilitating communication between stakeholders, capturing early decisions about the high-level design, and allowing reuse of design components between projects.[28]

[27] Clements, Paul; Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford. Documenting Software Architectures: Views and Beyond, Second Edition. Boston: Addison-Wesley, 2010.

[28] Bass, Len; Paul Clements, Rick Kazman. Software Architecture In Practice, Third Edition. Boston: Addison-Wesley. pp. 21–24, 2012.

The development of a SAD task comprises the following:

a. Develop a SAD that satisfies all software requirements specified in the SRS that shows the various software processes, databases, files, messages, and screen designs. (completeness)
b. Develop the SAD to address all operating modes specified in the SRS, including initialization, operational, shut-down, maintenance, and test modes. (completeness)
c. Identify actions to be taken in the event of error detection and verify that the propagation of errors is controlled via a well-structured modular design. (reliability)
d. Separate the safety functions from normal operating and overhead functions, with well-defined and strictly controlled interfaces between them. Any online maintenance features should be included. (safety)
e. Describe all timing limitations, the strategy for handling each, the required margins, and the method of measuring those margins. Provide a timing specification for each architectural element, in terms of minimum and maximum times for execution. Describe scheduling mechanisms and inter-process communication methods. Ensure that operations are performed in the correct sequence. (timing)
f. Each software architectural element is compatible with the SRS, the hardware architecture, documented descriptions and known properties of the operational and hardware environment, and other software elements. Timing specifications of each software element should be consistent with the specifications of the other elements with which it interacts and with the expected performance of the system as a whole. Uniform and consistent terminology, notation, and definitions should be used. (consistency)
g. Prepare the architecture description in conformance to the developer's style guide. Provide the rationale for architectural decisions. (Style)
h. Provide adequate information so that it is possible to construct specific analyses, reviews, and tests to verify that the architecture satisfies the software requirements. (Verifiability)

(2) Development of Software Design Description (SDD)

Software Design Description (SDD) is a representation of software created to facilitate analysis, planning, implementation, and decision-making. The SDD is used as a medium for communicating software design information and may be thought of as a blueprint or model of the system. (IEEE 2012) An SDD shows *exactly* how the software requirements (defined in the SRS) are implemented in the software modules and programs. (CR-6101)

Development of SDD activities are as follows:

a. Develop the software design that uses a hierarchical decomposition into layers of design elements. A design element may be a software system, subsystem, or module; database, file, data structure, or other data store; message; program or process. Each element should include its attributes such as its function, a list of elements it interacts with, the way it interacts with elements, the resources it needs (e.g., disk space), the method by which the element carries out its function, and a list

of requirements implemented by this element. The software elements should be refined into lower levels containing software units that can be coded, compiled, and tested. (IEEE 12207, 7.1.4.3.1.5)

b. Design the software to be consistent with the architectural design, and that the design elements are mutually consistent. Design elements should be consistent with documented descriptions and known properties of the operational environment within which the software will execute. Input and output specifications specified in the software design should be consistent with interface requirements imposed by the hardware or pre-developed software products. Timing specifications of each detailed design element should be consistent with the timing specifications of the architectural element of which it is a part. Models, algorithms, and numerical techniques specified in the software design should agree with standard references where such are applicable. A uniform and consistent terminology, notation, and definitions should be used. Models, algorithms, and numerical techniques specified in the software design should be mathematically mutually compatible. (Consistency)
c. Design the software to operate correctly in the presence of unexpected, incorrect, anomalous and improper (1) input, (2) hardware behavior, or (3) software behavior. In particular, the software should not fail, and should not provide incorrect outputs, in the presence of these conditions. Pay attention to those values of input variables that are physically possible to the device, even if logically impossible in the application (to account for sensor errors, communication line noise, and similar concerns). (Robustness)
d. Evaluate all equations, algorithms, and control logic for potential errors. All equations and algorithms should be defined to a sufficient level of detail to permit coding. Data structure design should ensure that the code elements will correctly initialize data, correctly access stored data, and correctly scale dimension data. The detailed design should ensure that no data item can be used before it is initialized, can have its value changed in an unanticipated manner, or can have its value changed by an unanticipated design element. The detailed design should ensure that no data item can be changed in an unanticipated manner. (Correctness)
e. Specify the actions of each software unit for the entire domain of each input variable (for example, the complete span of instrument inputs or clock/calendar time). The design should be sufficiently complete to permit implementation to take place. Actions should be specified for all situations anticipated in the SRS. Equipment, human, hardware, and software interfaces should be *correctly* and fully specified. Equations, algorithms, and control logic should be *correctly* and fully specified. (Completeness)
f. Define and document test requirements and the schedule for testing software units. The test requirements should include stressing the software unit at the limits of its requirements. (IEEE 12207, 7.1.4.3.1.5)
g. Provide adequate justification for the use of the following — Floating point arithmetic.— Recursion.— Interrupts, except for periodic timer interrupts. — Multi-processing on a single processor. — Dynamic memory management. — Event-driven communications between processes.
h. Consider potential inconsistencies when multiple design methods are used.

i. Use simple static and dynamic structures[29] with minimal connections between design elements.
j. Check the validity of the input to each module.
k. Design the software such that single failures of individual elements will not cause safety system failure. (Reliability)
l. The detailed design introduces no new safety hazards into the safety system. (Safety)
m. Unauthorized changes are prevented, detected, or mitigated as appropriate. (Security) (BTP-14, B.3.3.3.1)
n. Ensure the time delay between stimulus and response is deterministic. (Timing)
o. Show the digital computer timing is consistent with the limiting response times and characteristics of the computer hardware, software, and data communications systems.
p. Develop the SDD in conformance with the developer's style guide. Provide the rationale for design decisions. Identify the programming language standards. Identify those language features which will not be used without justification. (Style)
q. Provide adequate information to construct specific analyses, reviews, and tests to verify that the design satisfies the software architecture. (Verifiability)

(3) Traceability Analysis - Design Phase
The traceability analysis of Software Design Development is to trace design elements (SDD) to requirements (SRS), and requirements to design elements; analyze trace relationships for correctness, consistency, and completeness; verify all the traces; and update the RTM.

The Development tasks for the traceability analysis are as follows:

a. Trace the software design (SDD) to software requirements (SRS) (backward traceability) and the software requirements (SRS) to the software design (SDD) (forward traceability).
b. Analyze identified relationships for correctness, consistency, completeness, and accuracy. The task criteria are as follows:
    - i) Validate the relationship between each design element and the software requirement(s). (Correctness)
    - ii) Verify that the relationships between the design elements and the software requirements are specified to a consistent level of detail. (Consistency)
    - iii) Verify that all design elements are traceable from the software requirements; and all software requirements are traceable to the design elements. (Completeness)
c. Update the traceability matrix and generate the traceability analysis report.

[29] dynamic architectures: the architecture may evolve during the execution of the system, e.g. components are created, deleted, or reconfigured at runtime.

(4) Criticality Analysis- Design Phase

Criticality analysis is used to assign integrity/criticality level (e.g., to design elements) which in turn is used to determine the rigor and effort of the Development activities at each stage of the Development lifecycle process. The software integrity level assignment will be continually reviewed and updated by conductingthe Development criticality analysis task throughout the software development process.

The Software Design Development criticality analysis task is as follows:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report by determining integrity levels of design elements using the SDD.
b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(5) Hazard Analysis- Design Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012).

The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. The hazard analysis is repeated in each lifecycle phase and accounts for further elaboration of designs (e.g., logic design and associated data elements in the software design phase), changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012).

The Software Design Development hazard analysis is as follows:

a. Ensure the logic design and associated data elements that implement critical requirements introduce no new hazards.
b. Assess the identified mitigation strategies to ensure each hazard is prevented, mitigated, or controlled (any unmitigated hazards are documented and addressed as part of the system and software operations).
c. Update the hazard analysis report.

(6) Security Analysis- Design Phase

The objective of security analysis performed by the Development effort is to verify that the system-required threat controls and safeguards are correctly implemented and to validate that they provide the desired levels of protection of system vulnerabilities. The Software Design Development security analysis is to evaluate software architectures and designs to determine whether security functions meet required capabilities, whether additional threat controls are needed, and whether design changes are needed to remove vulnerabilities.

The specific task is as follows:

a. Ensure that the architecture and detailed design outputs adequately address the identified security requirements. This assurance includes both the system itself and security risks introduced as a result of interfacing with external components.
b. Ensure the identified security threats and vulnerabilities are prevented, controlled, or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of the system and software operations).

(7) Risk Analysis - Design Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the software or the successful development of the software.

Risk analysis will be performed continuously throughout the software development lifecycle. For example, in this phase, the SAD and SDD are reviewed and used in updating prior risk analysis report.

Procedure of risk analysis in the Design Phase is as follows:

a. Review and update risk analysis using prior task reports with Design Phase outputs (e.g., SAD and SDD).
b. Provide recommendations to eliminate, reduce, or mitigate the risks.

(8) Software Component Test Plan Generation

Software Component Test is conducted to verify the correct implementation of the design and compliance with program requirements for one software element (e.g., unit, module) or a collection of software elements. Software Component Test Plan describes the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. (see ISO/IEC/IEEE 24765-2010) Software Component Test Plan can be prepared once the SRS and SDD are available.

The Software Component Test Plan Development task is as follows:

a. Plan Development software component testing to ensure that the software components (e.g., units and source code modules) correctly implement component requirements.

b. Plan tracing of design requirements to test design, cases, procedures, and results.
c. Plan documentation of test tasks and results.
d. The software component test plan addresses the following:
    i) Conformance to design requirements.
    ii) Assessment of timing, sizing, and accuracy.
    iii) Performance at boundaries and interfaces and under stress and error conditions.
    iv) Measures of requirements test coverage and software reliability and maintainability.
e. Ensure that the software component test plan conforms to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
f. Ensure that the software component test plan satisfies the following criteria:
    i) Traceable to the software requirements and design.
    ii) External consistency with the software requirements and design.
    iii) Internal consistency between unit requirements.
    iv) Test coverage of requirements in each unit.
    v) Feasibility of software integration and testing.
    vi) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

(9) Software Integration Test Plan Generation

Integration Testing is an orderly progression of testing of incremental pieces of the software program in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated to show compliance with the program design, and capabilities and requirements of the system. Software Integration Test Plan describes the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. (see ISO/IEC/IEEE 24765-2010) Software Integration Test Plan can be prepared once the SRS and SDD are available.

The Software Integration Test Plan Development task is as follows:
a. Plan software integration testing to ensure that the software correctly implements the software requirements and design as each software component (e.g., units or modules) is incrementally integrated with each other.
b. Plan tracing of requirements to test design, cases, procedures, and results.
c. Plan documentation of test tasks and results.
d. The software integration test plan addresses the following:
    i) Conformance to increasingly larger set of functional requirements at each stage of integration.
    ii) Assessment of timing, sizing, and accuracy.
    iii) Performance at boundaries and under stress conditions.
    iv) Measures of requirements test coverage and software reliability.
e. Ensure that the software integration test plan satisfies the following criteria:
    i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
f. Ensure that the software integration test plan satisfies the following criteria:
    i) Traceable to the system requirements.

ii) External consistency with the system requirements.
iii) Internal consistency.
iv) Test coverage of the software requirements.
v) Appropriateness of test standards and methods used.
vi) Conformance to expected results.
vii) Feasibility of software qualification testing.
viii) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

(10) Software Component Test Design Generation

Software Component Test design specifies the details of the test approach for a software component feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). It can be prepared once the software requirements specification (SRS) and software design description (SDD) are available, and the Software Component Test Plan has completed.

The Development task is as follows:

a. Design tests for Development software component testing.
b. Continue tracing required by the Development software component test plan.
c. Ensure that the software component test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Ensure that the Development software component test designs satisfy the criteria in Development Attribute (8) of this Software Design Development node.

(11) Software Integration Test Design Generation

Software Integration Test Design specifies the details of the test approach the software integration test feature or combination of features and identifies the associated tests. (ISO/IEC/IEEE 24765-2010) It can be prepared once the SRS and SDD are available, and the Software Integration Test Plan has completed.

The Software Integration Test Design Development task is as follows:

a. Design tests for Development software integration testing.
b. Continue tracing required by the Development software integration test plan. Verify that the Development software integration test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
c. Ensure that the software integration test designs satisfy the criteria in Development Attribute (9) of this Software Design Development node.

(12) Software Qualification Test Design Generation

Software qualification testing is performed on a complete, integrated system (or a system component such as software) to evaluate the system's compliance with its specified requirements. Software qualification test design specifies the details of the test approach for a software feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). It can be prepared once the SyRS, SRS, SDD are available and the Software Integration Test Plan has completed.

The Development task is as follows:

a. Design tests for software qualification testing.
b. Continue tracing required by the software qualification test plan. Ensure that the Development software qualification test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
c. Ensure that the Development software qualification test designs satisfy the criteria in Development Attribute (9) of *Attributes of Software Requirement Development Activities*.

(13) Software Acceptance Test Design Generation

Software acceptance testing is conducted to determine whether or not a system (or a system component such as software) satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. Acceptance test design specifies the details of the test approach for a software/system feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). The software acceptance test design can be prepared once the SyRS, SRS, SDD are available and the Software Acceptance Test Plan has completed.

The Development task is as follows:

a. Design tests for Development software acceptance testing.
b. Continue tracing required by the Development software acceptance test plan. Ensure that the Development software acceptance test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
c. Ensure that the software acceptance test designs satisfy the criteria in Development Attribute (10) of *Attributes of Software Requirement Development Activities*.

(14) Configuration Management-Design Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system (i.e., SAD and SDD of the design phase), software and hardware product functionality, tracking program versions, generating baselines (including parameters and settings), and managing changes. The configuration management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs.

In the Design phase CM, the development organization ensures:

a. The defined configuration management strategy has been executed in the Design Phase. Notification to the V&V effort for all changes made to the configuration item baselines has been timely prepared to facilitate projects progress.
b. Each design element should be placed in the configuration management system as a configuration item.
c. The status of items under configuration management is made available throughout the life cycle. Provisions should be included to assure that the V&V effort receives the current status for all configuration items required in the Design phase including but not limited to SDD, SAD, and software architectural drawings if any.

(15) Reviews and Audit- Design Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews (e.g., review of SAD and SDD in this phase), inspections, walk-throughs, and audits (IEEE Std 1028).

The reviews and audit tasks for the Design Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, V&V, and QA) has been followed faithfully.
b. Document reviews/audit results (e.g., on the SDD and SAD).
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.
d. Document improvement and/or lessons learned as a result of reviews/audit.

B.4 Attributes of Software Design V&V Activities

In software design, the software requirements are transformed into architecture and a detailed design for each software component. The design includes databases and system interfaces (e.g., hardware, operator/user, software components, and subsystems). The Software Design V&V activity addresses the software's architectural design and its detailed design. V&V test planning continues during the Software Design V&V activity.

The objective of Software Design V&V is to demonstrate that the design is a correct, accurate, and a complete transformation of the software requirements, and that no unintended features are introduced.

The V&V effort includes the following Software Design V&V attributes (IEEE Std 1012)[30] with more detailed activities specified in the table below:

(1) Design Evaluation
(2) Interface Analysis V&V
(3) Traceability Analysis V&V- Design Phase
(4) Criticality Analysis V&V- Design Phase
(5) Hazard Analysis V&V- Design Phase
(6) Security Analysis V&V- Design Phase
(7) Risk Analysis V&V- Design Phase
(8) V&V Software Component Test Plan Generation
(9) V&V Software Integration Test Plan Generation
(10) V&V Software Component Test Design Generation
(11) V&V Software Integration Test Design Generation
(12) V&V Software Qualification Test Design Generation
(13) V&V Software Acceptance Test Design Generation
(14) Configuration Management V&V- Design Phase
(15) Review and Audit- Design Phase
(16) V&V Design Phase Activity Summary Report Generation

(1) Design Evaluation

Evaluate the software architectural design (SAD) and software design elements (SDD) for correctness, consistency, completeness, accuracy, readability, and testability.

The task criteria are as follows:

a. Verify and validate that the software design satisfies the software requirements. (Correctness)
b. Verify that the software design complies with standards, references, regulations, policies, physical laws, and business rules. (Correctness)
c. Validate the design sequences of states and state changes using logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis. (Correctness)
d. Validate that the detailed design and its element interactions do not result in unnecessary, unintended, or deleterious consequences. (Correctness)
e. Validate that the flow of data and control satisfy functionality and performance requirements. (Correctness)
f. Validate data usage and format. (Correctness)
g. Assess the appropriateness of design methods and standards used. (Correctness)
h. Verify that all terms and design concepts are documented consistently. (Consistency)
i. Verify that there is internal consistency between the design elements and external consistency with architectural design. (Consistency)

[30] In our model in the Design Phase, the task order is rearranged to be consistent with the Concept/Requirements Phase

j. Verify that the following elements are in the SDD, within the assumptions and constraints of the system (Completeness):
    i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting, and logging).
    ii) Process definition and scheduling.
    iii) Hardware, software, and user interface descriptions.
    iv) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).
    v) Critical configuration data.
    vi) System, device, and software control (e.g., initialization, transaction and state monitoring, and self-testing).
    vii) Verify that the SDD satisfy specified configuration management procedures.
k. Validate that the logic, computational, and interface precision (e.g., truncation and rounding) satisfy the requirements in the system environment. (Accuracy)
l. Validate that the modeled physical phenomena conform to system accuracy requirements and physical laws. (Accuracy)
m. Verify that the documentation is legible, understandable, and unambiguous to the intended audience. (Readability)
n. Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms, symbols, and design language, if any. (Readability)
o. Verify that there are objective acceptance criteria for validating each software design element and the system design. (Testability)
p. Verify that each software design element is testable to objective acceptance criteria. (Testability)

(2) Interface Analysis V&V

The objective of the Software Design V&V interface analysis is to verify and validate that the software design interfaces with hardware, user, operator, software, and other systems for correctness, consistency, completeness, accuracy, and testability.

The task criteria are as follows:

a. Validate the external and internal software interface design in the context of system requirements. (Correctness)
b. Verify that the interface design is consistent between the SDD. (Consistency)
c. Verify that each interface is described and includes data format and performance criteria (e.g., timing, bandwidth, accuracy, safety, and security). (Completeness)
d. Verify that each interface provides information with the required accuracy. (Accuracy)
e. Verify that there are objective acceptance criteria for validating the interface design. (Testability)

(3) Traceability Analysis V&V- Design Phase

The traceability analysis of Software Design V&V is to trace design elements (SDD) to requirements (SRS), and requirements to design elements. Analyze relationships for correctness, consistency, and completeness.

TheV&V tasks for the traceability analysis are as follows:

a. Trace the software design (SDD) to software requirements (SRS) (backward traceability) and the software requirements (SRS) to the software design (SDD) (forward traceability).
b. Analyze identified relationships for correctness, consistency, and completeness. The task criteria are as follows:

    i) Validate the relationship between each design element and the software requirement(s). (Correctness)
    ii) Verify that the relationships between the design elements and the software requirements are specified to a consistent level of detail. (Consistency)
    iii) Verify that all design elements are traceable from the software requirements; and all software requirements are traceable to the design elements. (Completeness)

c. Update the traceability matrix and generate the traceability analysis report.

(4) Criticality Analysis V&V- Design Phase

Criticality analysis is used to assign integrity/criticality level (e.g., to design elements) which in turn is used to determine the rigor and effort of the V&V activities at each stage of the V&V lifecycle process. The software integrity level assignment should be continually reviewed and updated by conductingthe V&V criticality analysis task throughout the software development process.

The Software Design V&V criticality analysis task is as follows:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report by determining the integrity level of design elements using the SDD.
b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(5) Hazard Analysis V&V- Design Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012). The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. The hazard analysis is repeated in each lifecycle phase and accounts for further elaboration of designs(e.g., logic design and associated data elements in the software design phase), changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012).

The Software Design V&V hazard analysis is as follows:

a. Verify the logic design and associated data elements that implement critical requirements introduce no new hazards.

b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated, or controlled (any unmitigated hazards are documented and addressed as part of the system and software operations).
c. Update the hazard analysis report

(6) Security Analysis V&V- Design Phase

The objective of security analysis performed by the V&V effort is to verify that the system-required threat controls and safeguards are correctly implemented and to validate that they provide the desired levels of protection of system vulnerabilities. The Software Design V&V security analysis is to evaluate software architectures and designs to determine whether security functions meet required capabilities, whether additional threat controls are needed, and whether design changes are needed to remove vulnerabilities.

The specific task is as follows:

a. Verify that the architecture and detailed design outputs adequately address the identified security requirements. This verification includes both the system itself and security risks introduced as a result of interfacing with external components.
b. Verify the identified security threats and vulnerabilities are prevented, controlled, or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of the system and software operations).

(7) Risk Analysis V&V- Design Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the software or the successful development of the software.

Risk analysis should be performed continuously throughout the software development lifecycle. For example, in this phase, the SAD and SDD are reviewed and used in updating prior risk analysis report.

Procedure of risk analysis is as follows:

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.

(8) V&V Software Component Test Plan Generation

Software Component Test is conducted to verify the correct implementation of the design and compliance with program requirements for one software element (e.g., unit, module) or a collection of software elements. Software Component Test Plan describes the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. (see ISO/IEC/IEEE 24765-2010) Software Component Test Plan can be prepared once the SRS and SDD are available.

The Software Component Test Plan V&V task is as follows:

a. Plan V&V software component testing to validate that the software components (e.g., units and source code modules) correctly implement component requirements.
b. Plan tracing of design requirements to test design, cases, procedures, and results.
c. Plan documentation of test tasks and results.
d. The V&V software component test plan should address the following:

i) Conformance to design requirements.
ii) Assessment of timing, sizing, and accuracy.
iii) Performance at boundaries and interfaces and under stress and error conditions.
iv) Measures of requirements test coverage and software reliability and maintainability.

e. Verify that the V&V software component test plan conforms to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).

f. Validate that the V&V software component test plan satisfies the following criteria:

i) Traceable to the software requirements and design.
ii) External consistency with the software requirements and design.
iii) Internal consistency between unit requirements.
iv) Test coverage of requirements in each unit.
v) Feasibility of software integration and testing.
vi) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

(9) V&V Software Integration Test Plan Generation

Integration Testing is an orderly progression of testing of incremental pieces of the software program in which software elements are combined and tested until the entire software has been integrated to show compliance with the program design, and capabilities and requirements of the software/system. Software Integration Test Plan describes the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. (see ISO/IEC/IEEE 24765-2010) Software Integration Test Plan can be prepared once the SRS and SDD are available.

The Software Integration Test Plan V&V task is as follows:

a. Plan V&V software integration testing to validate that the software correctly implements the software requirements and design as each software component (e.g., units or modules) is incrementally integrated with each other.
b. Plan tracing of requirements to test design, cases, procedures, and results.
c. Plan documentation of test tasks and results.
d. The V&V software integration test plan addresses the following:

i) Conformance to increasingly larger set of functional requirements at each stage of integration.
ii) Assessment of timing, sizing, and accuracy.
iii) Performance at boundaries and under stress conditions.
iv) Measures of requirements test coverage and software reliability.

e. Verify that the V&V software integration test plan satisfies the following criteria:

i) Conformance to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).

f. Validate that the V&V software integration test plan satisfies the following criteria:

i) Traceable to the system requirements.
ii) External consistency with the system requirements.
iii) Internal consistency.
iv) Test coverage of the software requirements.
v) Appropriateness of test standards and methods used.
vi) Conformance to expected results.
vii) Feasibility of software qualification testing.
viii) Feasibility of operation and maintenance (e.g., capability to be operated and maintained in accordance with user needs).

(10) V&V Software Component Test Design Generation

Software Component Test design specifies the details of the test approach for a software component feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). Software Component Test Design can be prepared once the software requirements specification (SRS) and software design description (SDD) are available, and the Software Component Test Plan has completed.

The V&V task is as follows:

a. Design tests for V&V software component testing.
b. Continue tracing required by the V&V software component test plan.
c. Verify that the V&V software component test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software component test designs satisfy the criteria in V&V Attribute (8) of this Software Design V&V node.

(11) V&V Software Integration Test Design Generation

Software Integration Test Design specifies the details of the test approach the software integration test feature or combination of features and identifies the associated tests. (ISO/IEC/IEEE 24765-2010) Software Integration Test Design can be prepared once the SRS and SDD are available, and the Software Integration Test Plan has completed.

The Software Integration Test Design V&V task is as follows:

a. Design tests for V&V software integration testing.
b. Continue tracing required by the V&V software integration test plan.
c. Verify that the V&V software integration test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software integration test designs satisfy the criteria in V&V Attribute (9) of this Software Design V&V node.

(12) V&V Software Qualification Test Design Generation

Software qualification testing is performed on a complete, integrated system (or a system component such as software) to evaluate the system's compliance with its specified requirements. Software qualification test design specifies the details of the test approach for a software feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). The software Integration test design can be prepared once the SyRS, SRS, SDD are available and the Software Integration Test Plan has completed.

The V&V task is as follows:

a. Design tests for V&V software qualification testing.
b. Continue tracing required by the V&V software qualification test plan.
c. Verify that the V&V software qualification test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software qualification test designs satisfy the criteria in V&V Attribute(11) of Software Requirement V&V Activities.

(13) V&V Software Acceptance Test Design Generation

Software acceptance testing is conducted to determine whether or not a system (or a system component such as software) satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. Acceptance test plans can be generated once the system/software requirements are available. Acceptance test design specifies the details of the test approach for a software/system feature or combination of features and identifies the associated tests (see ISO/IEC/IEEE 24765-2010). The software acceptance test design can be prepared once the SyRS, SRS, SDD are available and the Software Acceptance Test Plan has completed.

The V&V task is as follows:

a. Design tests for V&V software acceptance testing.
b. Continue tracing required by the V&V software acceptance test plan.
c. Verify that the V&V software acceptance test designs conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software acceptance test designs satisfy the criteria in V&V Attribute (12) of Software Requirement V&V Activities.

(14) Configuration Management V&V- Design Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system, software and hardware product functionality, tracking program versions, generating baselines

(including parameters and settings), and managing changes. The configuration management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs.

In the Design phase CM, the V&V organization should ensure:

a. The defined configuration management strategy has been executed in the Design Phase.
b. Interfaces with the Development organization have been adequate to facilitate projects progress.
c. All changes made to the configuration item baselines have been documented and verified & validated.
d. The status of items under configuration management is made available throughout the life cycle.

(15) Review and Audit- Design Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028).

The reviews and audit tasks for the Design Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, Development, and QA) has been followed faithfully.
b. Reviews/audit results (e.g., on the SDD, various analysis reports, and V&V Design Phase products) should be documented.
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.
d. Improvement and/or lessons learned should be documented as a result of reviews/audit.

(16) V&V Design Phase Activity Summary Report Generation

V&V Design Phase activity summary report summarizes the results of V&V tasks performed in the phase.

Specific task for the V&V Design Phase Activity Summary Report Generation is as follows:

a. Document software V&V activities conducted in the Design phase
b. Provide background information of standards, and source documents with revisions.
c. Provide confirmation that the V&V plan was followed.
d. Document the individual who performed the task, the portion of the task performed by the individual (if multiple performers), and the date that the task was performed.
e. Summarize anomaly issues identified, the resolution process of them, and recommendations whether to proceed to the Implementation phase of the software development lifecycle.

B.5 Attributes of Software Implementation Activities

In software implementation, the software design is transformed into code, database structures, and related machine-executable representations. The objective of Software Implementation Development is to produce codes from design and ensure that the transformations are correct, accurate, and complete.

Additionally, the Software Implementation activity addresses testing the software testing including test planning, test case generation, test procedures, and test analysis [IEEE Std 1012]. After the implementation phase, the testing phase can be started.

The Development effort performs the following Software Implementation Development tasks[31]:

1) Source Code and Source Code Documentation Generation
2) Traceability Analysis-Implementation Phase
3) Criticality Analysis-Implementation Phase
4) Hazard Analysis-Implementation Phase
5) Security Analysis-Implementation Phase
6) Risk Analysis-Implementation Phase
7) Component Test Case Generation
8) Software Integration Test Case Generation
9) Software Qualification Test Case Generation
10) Software Acceptance Test Case Generation
11) Software Component Test Procedure Generation
12) Software Integration Test Procedure Generation
13) Software Qualification Test Procedure Generation
14) Software Component Test Execution

Additionally, Configuration Management and Reviews & Audits are listed as required attributes because configuration management and QA functions are included in the Development and V&V nodes.

(1) Source Code and Source Code Documentation Generation

Generate the source code components (source code and source code documentation) from design (e.g., SAD and SDD).

The specific development tasks are as follows:

a. Translate software design as described in SAD and SDD into source code and source code documentation. (Correctness)
b. The source code components should comply with standards, references, regulations, policies, physical laws, and business rules. (Correctness)

[31] In our model in the Implementation Phase, the task order is rearranged to be consistent with the Concept/Requirements Specification Phases

c. The source code component sequences of states and state changes should be correctly implemented using logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis. (Correctness)
d. The software code and its interactions with other elements should not result in unnecessary, unintended, or deleterious consequences. (Correctness)
e. The flow of data and control satisfy functionality and performance requirements. (Correctness)
f. Data usage and format are correctly implemented. (Correctness)
g. Ensure the appropriateness of coding methods and standards. (Correctness)
h. All terms and code concepts are documented consistently. (Consistency)
i. There should be internal consistency between the source code components. (Consistency)
j. There should be external consistency with the software design and requirements. (Consistency)
k. The following elements should be in the source code, within the assumptions and constraints of the system: (Consistency)

    i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting, and logging).
    ii) Process definition and scheduling.
    iii) Hardware, software, and user interface descriptions.
    iv) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).
    v) Critical configuration data.
    vi) System, device, and software control (e.g., initialization, transaction and state monitoring, defensive programming practices, and self-testing).

l. The source code documentation satisfies specified coding standards. (Consistency)
m. The logic, computational, and interface precision (e.g., truncation and rounding) in the system environment meet the system and software requirements. (Accuracy)
n. The modeled physical phenomena conform to system accuracy requirements and physical laws. (Accuracy)
o. The source code documentation is legible, understandable, and unambiguous to the intended audience. (Readability)
p. The source code documentation defines all acronyms, mnemonics, abbreviations, terms, and symbols. (Readability)
q. There should be objective acceptance criteria for validating each source code component. (Testability)
r. Each source code component should be testable against objective acceptance criteria. (Testability)

(2) Traceability Analysis - Implementation Phase

The traceability analysis of Software Implementation Phase is to trace source code components to corresponding design elements as specified in SAD and SDD), and design elements to source code components. Analyze relationships for correctness, consistency, and completeness. The tasks for the traceability analysis are as follows:

a. Trace the source code components to corresponding design specification(s), and design specification(s) to source code components.

b. Analyze identified relationships for correctness, consistency, and completeness. The task criteria are as follows:

    i) Validate the relationship between the source code components and design element(s). (Correctness)
    ii) Verify that the relationships between the source code components and design elements are specified to a consistent level of detail. (Consistency)
    iii) Verify that all source code components are traceable from the design elements; and that all design elements are traceable to the source code components. (Completeness)

c. Update the traceability matrix and generate the traceability analysis report.

(3) Criticality Analysis - Implementation Phase

Criticality analysis is used to assign integrity/criticality level (e.g., to source code modules), which in turn is used to determine the rigor and effort of the development activities at each phase of the lifecycle process. The software integrity level assignment should be continually reviewed and updated by conductingthe criticality analysis task throughout the software development process.

The Software Implementation Phase criticality analysis task is as follows:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report using the source code.
b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, or other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(4) Hazard Analysis - Implementation Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012). The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. The hazard analysis is repeated in each lifecycle phase and accounts for further elaboration of designs, changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012). The hazard analysis in Implementation Phase addresses hazards in the source codes and their associated data elements. The Software Implementation Phase hazard analysis is as follows:

a. Ensure that the implementation and associated data elements correctly implement the critical requirements and introduce no new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated, or controlled (any unmitigated hazards are documented and addressed as part of the system and software operations).
c. Update the hazard analysis.

(5) Security Analysis - Implementation Phase

The objective of security analysis performed by the Development effort is to ensure that the system-required threat controls and safeguards are correctly implemented and to validate that they provide the desired levels of protection of system vulnerabilities. The Software Implementation Development security analysis is to evaluate software source code to determine whether security functions are implemented and meet required capabilities, the implementation does not introduce new security risk, and whether additional threat controls are needed to remove any vulnerabilities. The specific task is as follows:

a. Ensure that the implementation is completed in accordance with the system design in that it addresses the identified security risks and that the implementation does not introduce new security risks through coding flaws, or compiler error.
b. Ensure the identified security threats and vulnerabilities are prevented, controlled, or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of the system and software operations).
c. Update the security analysis.

(6) Risk Analysis - Implementation Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the software or the successful development of the software.

Risk analysis should be performed continuously throughout the software development lifecycle. Risk analysis in the Implementation Phase addresses risks that may be introduced in the source codes and their associated date elements. Procedure of risk analysis in Implementation Phase is as follows:

a. Review and update risk analysis using prior task reports using the source code.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.

(7) Software Component Test Case Generation

Software Component Test Case specifies inputs, predicted results, and a set of execution conditions for Software Component Test. The development task produces software Component Test Case specification; and validates that the software component test cases satisfy the criteria specified in the Software Component Test Plan.

The tasks are as follows:

a. Develop test cases for software component testing (e.g., statistical sampling, boundary conditions, and code coverage).
b. Continue tracing required by the software component test plan.
c. Verify that the software component test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software component test cases satisfy the criteria in the Design Phase Attribute (8).

(8) Software Integration Test Case Generation

Software Integration Test Case specifies inputs, predicted results, and a set of execution conditions for Software Integration Test. The development task produces software Integration Test Case specification; and validates that the software Integration test cases satisfy the criteria specified in the Software Integration Test Plan.

The tasks are as follows:

a. Develop test cases for software integration testing.
b. Continue tracing required by the software integration test plan.
c. Verify that the software integration test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software integration test cases satisfy the criteria in the Design Phase Attribute (9).

(9) Software Qualification Test Case Generation

Software Qualification Test Case specifies inputs, predicted results, and a set of execution conditions for Software Qualification Test. The development task includes software Qualification Test Case specification; and validates that the software Integration test cases satisfy the criteria specified in the Software Qualification Test Plan.

The tasks are as follows:

a. Develop test cases for software qualification testing.
b. Continue tracing required by the software qualification test plan.
c. Verify that the software qualification test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software qualification test cases satisfy the criteria in the Requirements Specification Phase Attribute (9).

(10) Software Acceptance Test Case Generation

Software Acceptance Test Case specifies inputs, predicted results, and a set of execution conditions for Software Acceptance Test. The development task includes software Acceptance Test Case specification; and validates that the software Acceptance test cases satisfy the criteria specified in the Software Acceptance Test Plan.

The tasks are as follows:

a. Develop test cases for software acceptance testing.
b. Continue tracing required by the software acceptance test plan.
c. Verify that the software acceptance test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software acceptance test cases satisfy the criteria in the Requirements Specification Phase Attribute (10).

(11) Software Component Test Procedure Generation

Software Component Test Procedure specifies a sequence of actions for the execution of the software component test. The development task generates Software Component Test Procedure; and validates that the software Component Test Procedures satisfy the criteria specified in the Software Component Test Plan.

The tasks are as follows:

a. Develop test procedures for software component testing.
b. Continue tracing required by the software component test plan.
c. Verify that the software component test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software component test procedures satisfy the criteria in the Design Phase Attribute (8).

(12) Software Integration Test Procedure Generation

Software Integration Test Procedure specifies a sequence of actions for the execution of the software integration test. The development task generates Software Integration Test Procedure; and validates that the software Integration Test Procedures satisfy the criteria specified in the Software Integration Test Plan.

The tasks are as follows:

a. Develop test procedures for software integration testing.
b. Continue tracing required by the software integration test plan.
c. Verify that the software integration test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software integration test procedures satisfy the criteria in the Design Phase Attribute (9).

(13) Software Qualification Test Procedure Generation

Software Qualification Test Procedure specifies a sequence of actions for the execution of the software qualification test. The development task generates Software Qualification Test Procedure; and validates that the software Qualification Test Procedures satisfy the criteria specified in the Software Qualification Test Plan.

The tasks are as follows:

a. Develop test procedures for software qualification testing.
b. Continue tracing required by the software qualification test plan.
c. Verify that the software qualification test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software qualification test procedures satisfy the criteria in the Requirements Specification Phase Attribute (9).

(14) Software Component Test Execution

Software Component Test Execution carries out an instruction, process, or computer program for software components. The development task performs component test; documents and analysis the results; and validates the test results satisfy the test acceptance criteria.

The task are as follows:

a. Perform software component testing.
b. Analyze test results to validate that software correctly implements the design.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.
d. Document the results as required by the software component test plan.
e. Use the software component test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between the actual and expected test results.

(15) Configuration Management - Implementation Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system, software and hardware product functionality, tracking program versions, generating baselines (including parameters and settings), and managing changes. The Development CM in the Implementation Phase focuses on the source codes and associated documentation configuration control. The configuration management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs.

In the Implementation phase CM, the Development organization ensures:

a. The defined configuration management strategy has been executed in the Implementation Phase.
b. Interfaces with the V&V organization have been adequate to facilitate projects progress.
c. All changes made to the configuration item (e.g., source code release) baselines have been documented and reviewed and approved.
d. The status of items under configuration management is made available throughout the lifecycle.

(16) Review and Audit - Implementation Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028).

The Development review and audit in the Implementation focuses on the source codes and associated documentation. The specific reviews and audit tasks for the Implementation Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, V&V, and QA) has been followed faithfully.
b. Reviews/audit results (e.g., on the source code, and source code documents) should be documented.
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.
d. Improvement and/or lessons learned should be documented as a result of reviews/audit.

B.6 Attributes of Software Implementation V&V Activities

In software implementation, the software design is transformed into code, database structures, and related machine executable representations. The objective of Software Implementation V&V is to verify and validate that these transformations are correct, accurate, and complete. Additionally, the Software Implementation V&V activity addresses software testing including test planning, test case generation, test procedures, and test analysis [IEEE Std 1012].

The V&V effort performs the following Software Implementation V&V tasks [IEEE Std 1012][32]:

1) Source Code and Source Code Documentation Evaluation
2) Interface Analysis V&V - Implementation Phase
3) Traceability Analysis V&V - Implementation Phase
4) Criticality Analysis V&V - Implementation Phase
5) Hazard Analysis V&V - Implementation Phase
6) Security Analysis V&V - Implementation Phase
7) Risk Analysis V&V - Implementation Phase
8) V&V Software Component Test Case Generation
9) V&V Software Integration Test Case Generation
10) V&V Software Qualification Test Case Generation
11) V&V Software Acceptance Test Case Generation
12) V&V Software Component Test Procedure Generation
13) V&V Software Integration Test Procedure Generation
14) V&V Software Qualification Test Procedure Generation
15) V&V Software Component Test Execution
16) Configuration Management V&V - Implementation Phase
17) Review and Audit V&V - Implementation Phase
18) V&V Implementation Phase Activity Summary Report Generation

[32] In our model in the Implementation Phase, the task order is rearranged to be consistent with the Concept/Requirements Specification Phases

Additionally, Configuration Management and Reviews & Audits are listed as required attributes because configuration management and QA functions are included in the Development and V&V nodes.

(1) Source Code and Source Code Documentation Evaluation

Evaluate the source code components (source code and source code documentation) for correctness, consistency, completeness, accuracy, readability, and testability. The V&V task criteria are as follows:

a. Verify and validate that the source code component satisfies the software design. (Correctness)
b. Verify that the source code components comply with standards, references, regulations, policies, physical laws, and business rules. (Correctness)
c. Validate the source code component sequences of states and state changes using logic and data flows coupled with domain expertise, prototyping results, engineering principles, or other basis. (Correctness)
d. Validate the software code and its interactions with other elements do not result in unnecessary, unintended, or deleterious consequences. (Correctness)
e. Validate that the flow of data and control satisfy functionality and performance requirements. (Correctness)
f. Validate data usage and format. (Correctness)
g. Assess the appropriateness of coding methods and standards. (Correctness)
h. Verify that all terms and code concepts are documented consistently. (Consistency)
i. Verify that there is internal consistency between the source code components. (Consistency)
j. Validate external consistency with the software design and requirements. (Consistency)
k. Verify that the following elements are in the source code, within the assumptions and constraints of the system: (Consistency)

    i) Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting, and logging).
    ii) Process definition and scheduling.
    iii) Hardware, software, and user interface descriptions.
    iv) Performance criteria (e.g., timing, sizing, speed, capacity, accuracy, precision, safety, and security).
    v) Critical configuration data.
    vi) System, device, and software control (e.g., initialization, transaction and state monitoring, defensive programming practices, and self-testing).

l. Verify that the source code documentation satisfies specified coding standards. (Consistency)
m. Validate the logic, computational, and interface precision (e.g., truncation and rounding) in the system environment. (Accuracy)
n. Validate that the modeled physical phenomena conform to system accuracy requirements and physical laws. (Accuracy)
o. Verify that the documentation is legible, understandable, and unambiguous to the intended audience. (Readability)
p. Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms, and symbols. (Readability)

q. Verify that there are objective acceptance criteria for validating each source code component. (Testability)
r. Verify that each source code component is testable against objective acceptance criteria. (Testability)

(2) Interface Analysis V&V - Implementation Phase

Verify and validate that the software source code interfaces with hardware, user, operator, software, and other systems for correctness, consistency, completeness, accuracy, and testability. The V&V task criteria are as follows:

a. Validate the external and internal software interface code in the context of system requirements. (Correctness)
b. Verify that the interface code is consistent between source code components and to external interfaces (i.e., hardware, user, operator, and other software). (Consistency)
c. Verify that each interface is described and includes data format and performance criteria (e.g., timing, bandwidth, accuracy, safety, and security). (Completeness)
d. Verify that each interface provides information with the required accuracy. (Accuracy)
e. Verify that there are objective acceptance criteria for validating the interface code. (Testability)

(3) Traceability Analysis V&V - Implementation Phase

The traceability analysis of Software Implementation Phase V&V is to trace source code components to corresponding design elements as specified in SDD), and design elements to source code components. Analyze relationships for correctness, consistency, and completeness. The V&V tasks for the traceability analysis are as follows:

a. Trace the source code components to corresponding design specification(s), and design specification(s) to source code components.
b. Analyze identified relationships for correctness, consistency, and completeness. The task criteria are as follows:
    i) Validate the relationship between the source code components and design element(s). (Correctness)
    ii) Verify that the relationships between the source code components and design elements are specified to a consistent level of detail. (Consistency)
    iii) Verify that all source code components are traceable from the design elements; and that all design elements are traceable to the source code components. (Completeness)
c. Update the traceability matrix and generate the traceability analysis report.

(4) Criticality Analysis V&V - Implementation Phase

Criticality analysis is used to assign integrity/criticality level (e.g., source code modules), which in turn is used to determine the rigor and effort of the V&V activities at each phase of the V&V lifecycle process. The software integrity level assignment

should be continually reviewed and updated by conductingthe V&V criticality analysis task throughout the software development process.

The Software Implementation Phase criticality analysis V&V task is as follows:

a. Review and update the existing criticality analysis results from the prior Criticality Task Report using the source code.
b. Implementation methods and interfacing technologies may cause previously assigned integrity levels to be raised or lowered for a given software element (i.e., requirement, module, function, subsystem, or other software partition). Verify that no inconsistent or undesired integrity consequences are introduced by reviewing the revised integrity levels.

(5) Hazard Analysis V&V - Implementation Phase

Hazard is an intrinsic property or condition that has the potential to cause harm or damage. Hazard is a source of potential harm or a situation with a potential for harm in terms of human injury, damage to health, property, or the environment, or some combination of these (IEEE Std 1012). The hazard analysis is a system engineering activity that will account for the system design, operational conditions, system physical constraints, and regulations to identify hazardous conditions that could lead to adverse consequences. The hazard analysis is repeated in each lifecycle phase and accounts for further elaboration of designs, changes to intended system use and operations, and the emergence of new hazardous conditions (IEEE Std 1012). Implementation Phase V&V addresses source codes and associated data elements potential hazards. The Software Implementation hazard analysis V&V is as follows:

a. Verify that the implementation and associated data elements correctly implement the critical requirements and introduce no new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated, or controlled (any unmitigated hazards are documented and addressed as part of the system and software operations).
c. Update the hazard analysis.

(6) Security Analysis V&V - Implementation Phase

The objective of security analysis performed by the V&V effort is to verify that the system-required threat controls and safeguards are correctly implemented and to validate that they provide the desired levels of protection of system vulnerabilities. The Software Implementation V&V security analysis is to evaluate software source code to determine whether security functions are implemented and meet required capabilities, the implementation does not introduce new security risk, and whether additional threat controls are needed to remove any vulnerabilities. The specific task is as follows:

a. Verify that the implementation is completed in accordance with the system design in that it addresses the identified security risks and that the implementation does not introduce new security risks through coding flaws, or compiler error.

b. Verify the identified security threats and vulnerabilities are prevented, controlled, or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of the system and software operations).
c. Update the security analysis.

(7) Risk Analysis V&V - Implementation Phase

The objective of risk analyses is to identify technical and management risks that have a measureable possibility of negative consequences to either the operation of the software or the successful development of the software.

Risk analysis should be performed continuously throughout the software development lifecycle. The Implementation Phase V&V risk analysis addresses potential risk in the source codes and their associated data elements. Procedure of risk analysis V&V in Implementation Phase is as follows:

a. Review and update risk analysis using prior task reports using the source code.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.

(8) V&V Software Component Test Case Generation

V&V Software Component Test Case specifies inputs, predicted results, and a set of execution conditions for Software Component Test. The V&V task produces software Component Test Case specification; and validates that the software component test cases satisfy the criteria specified in the V&V Software Component Test Plan.

The tasks are as follows:

a. Develop test cases for software component testing for V&V (e.g., statistical sampling, boundary conditions, and code coverage).
b. Continue tracing required by the V&V software component test plan.
c. Verify that the V&V software component test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the software component test cases satisfy the criteria in the Design Phase Attribute (8) for V&V.

(9) V&V Software Integration Test Case Generation

V&V Software Integration Test Case specifies inputs, predicted results, and a set of execution conditions for Software V&V Integration Test. The V&V task generates Integration Test Case specification; and validates that the software Integration test cases satisfy the criteria specified in the Software Integration Test Plan.

The tasks are as follows:

a. Develop test cases for V&V software integration testing.
b. Continue tracing required by the V&V software integration test plan.
c. Verify that the V&V software integration test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).

d. Validate that the V&V software integration test cases satisfy the criteria in the Design Phase Attribute (9) for V&V.

(10) V&V Software Qualification Test Case Generation

V&V Software Qualification Test Case specifies inputs, predicted results, and a set of execution conditions for Software Qualification Test V&V. The V&V task includes software Qualification Test Case specification; and validates that the software Integration test cases satisfy the criteria specified in the V&V Software Qualification Test Plan.

The tasks are as follows:

a. Develop test cases for V&V software qualification testing.
b. Continue tracing required by the V&V software qualification test plan.
c. Verify that the V&V software qualification test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software qualification test cases satisfy the criteria in the V&V Requirements Specification Phase Attribute (11).

(11) V&V Software Acceptance Test Case Generation

V&V Software Acceptance Test Case specifies inputs, predicted results, and a set of execution conditions for Software Acceptance Test for V&V. The V&V task includes software Acceptance Test Case specification; and validates that the software Acceptance test cases satisfy the criteria specified in the V&V Software Acceptance Test Plan .

The tasks are as follows:

a. Develop test cases for V&V software acceptance testing.
b. Continue tracing required by the V&V software acceptance test plan.
c. Verify that the V&V software acceptance test cases conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software acceptance test cases satisfy the criteria in the V&V Requirements Specification Phase Attribute (12).

(12) V&V Software Component Test Procedure Generation

V&V Software Component Test Procedure specifies a sequence of actions for the execution of the software component test. The V&V task generates Software Component Test Procedure; and validates that the software Component Test Procedures satisfy the criteria specified in the V&V Software Component Test Plan.

The tasks are as follows:

a. Develop test procedures for V&V software component testing.
b. Continue tracing required by the V&V software component test plan.

c. Verify that the V&V software component test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software component test procedures satisfy the criteria in the V&V Design Phase Attribute (8).

(13) V&V Software Integration Test Procedure

V&V Software Integration Test Procedure specifies a sequence of actions for the execution of the software integration tests. The V&V task generates V&V Software Integration Test Procedure; and validates that the V&V software Integration Test Procedures satisfy the criteria specified in the V&V Software Integration Test Plan.

The tasks are as follows

a. Develop test procedures for V&V software integration testing .
b. Continue tracing required by the V&V software integration test plan.
c. Verify that the V&V software integration test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software integration test procedures satisfy the criteria in the V&V Design Phase Attribute (9).

(14) V&V Software Qualification Test Procedure

V&V Software Qualification Test Procedure specifies a sequence of actions for the execution of the software qualification test. The V&V task generates Software Qualification Test Procedure; and validates that the software Qualification Test Procedures satisfy the criteria specified in the Software Qualification Test Plan for V&V.

The tasks are as follows:

a. Develop test procedures for V&V software qualification testing.
b. Continue tracing required by the V&V software qualification test plan.
c. Verify that the V&V software qualification test procedures conform to project-defined test document purpose, format, and content (e.g., IEEE Std 829-2008).
d. Validate that the V&V software qualification test procedures satisfy the criteria in the V&V Requirements Specification Phase Attribute (11).

(15) Execution of V&V Software Component Tests

V&V Software Component Test Execution carries out an instruction, process, or computer program for the software component. The V&V task performs component tests; documents and analysis the results; and validates the test results satisfy the test acceptance criteria.

The tasks are as follows:

a. Perform V&V software component testing.
b. Analyze test results to validate that software correctly implements the design.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.

d. Document the results as required by the V&V software component test plan.
e. Use the V&V software component test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between the actual and expected test results.

(16) Configuration Management V&V – Implementation Phase

Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements (IEEE Std 0610.12).

The purpose of the CM is to establish a process for describing the system, software and hardware product functionality, tracking program versions, generating baselines (including parameters and settings), and managing changes. The configuration management process should be adequate for the development complexity, system size, integrity level, project plans, and user needs. In the Implementation V&V CM, V&V reviews and verifies that the source code documents comply with the CM plan.

In the implementation phase CM, the V&V organization should ensure:

a. The defined configuration management strategy has been executed in the Implementation Phase.
b. Interfaces with the Development organization have been adequate to facilitate projects progress.
c. All changes made to the configuration item (e.g., source code release) baselines have been documented and verified & validated.
d. The status of items under configuration management is made available throughout the life cycle.

(17) Review and Audit V&V – Implementation Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028). In the Implementation Phase, V&V performs reviews and audit on the source code documents as well as the process.

The specific reviews and audit specific tasks for the Implementation Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, Development, and QA) has been followed faithfully.
b. Reviews/audit results (e.g., on the source code, various analysis reports, and V&V Implementation Phase products) should be documented.
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.
d. Improvement and/or lessons learned should be documented as a result of reviews/audit.

(18) V&V Implementation Phase Activity Summary Report Generation

V&V Implementation Phase activity summary report summarizes the results of V&V tasks performed in the phase.

Specific task for the V&V Implementation Phase Activity Summary Report Generation is as follows:

a. Document software V&V activities conducted in the Implementation phase.
b. Provide background information of standards, and source documents with revisions.
c. Provide confirmation that the V&V plan was followed.
d. Document the individual who performed the task, the portion of the task performed by the individual (if multiple performers), and the date that the task was performed.
e. Summarize anomaly issues identified, the resolution process of them, and recommendations whether to proceed to the Test phase of the software development lifecycle.

B.7 Attributes of Software Test Activities

With the completion of the Implementation Phase, the development enters into the Test Phase.

Test is an activity in which a system or component is executed under specified conditions, the results are observedor recorded, andan evaluation is made of some aspect of the system or component. Test activity includes test planning, test design,test case generation, and the generation of test procedures,that are carried out in various phases. However, test phase activities consist of software integration tests and executing qualification tests, the procedures for generating and executing acceptance test procedures and various analyses (e.g., traceability analysis, hazard analysis, security, and risk analyses) specific for the Test Phase. After the test phase, the software can be installed and tested in the target environment.

Test Phase activities are as follows [IEEE Std 1012]:

1) Software Integration Test Execution
2) Software Qualification Test Execution
3) Software Acceptance Procedure Generation
4) Software Acceptance Test Execution
5) Traceability Analysis - Test Phase
6) Hazard Analysis - Test Phase
7) Security Analysis - Test Phase
8) Risk Analysis - Test Phase

Additionally, Configuration Management and Reviews & Audits are listed as required tasks.

(1) Software Integration Test Execution

The objective of software integration test execution is to assure that the software requirements and system requirements allocated to software are validated as each software component (e.g., unit or module) is incrementally integrated.

The Software Integration Test Execution task is as follows:

a. Perform software integration testing.
b. Analyze test results to verify that the software components are integrated correctly.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.
d. Document the results as required by software integration test plan.
e. Use the software integration test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(2) Software Qualification Test Execution

The objective of software qualification test execution is to assure that the integrated software product satisfies its requirements. Software qualification (e.g., demonstration, analysis, inspection, or test) is performed on the complete software element.

The Software Qualification Test Execution task is as follows:

a. Perform software qualification testing.
b. Analyze test results to validate that the software satisfies the system requirements.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.
d. Document the results as required by the software qualification test plan.
e. Use the software qualification test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(3) Software Acceptance Test Procedure Generation

In Development Software Acceptance Test Procedure Generation, Development specifies a sequence of actions for the execution of the software acceptance test. The development task generates Software Acceptance Test Procedure; and validates that the software Acceptance Test Procedure satisfy the criteria specified in the Software Acceptance Test Plan.

The Software Acceptance Test Generation task is as follows:

a. Develop test procedures for software acceptance testing.
b. Continue the tracing required by the software acceptance test plan.
c. Verify that the software acceptance test procedures conform to Project-defined test document purpose, format, and content (e.g., see IEEE Std 829-2008 [B3]).
d. Validate that the software acceptance test procedures satisfy the criteria in the Requirements Specification Phase Attribute (10).

(4) Software Acceptance Test Execution

The objective of software acceptance test execution is to assure that the software satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the integrated software product.

The Software Acceptance Test Execution task is as follows:

a. Perform software acceptance testing.
b. Analyze test results to validate that the software satisfies the system requirements.
c. Validate that the test results trace to test criteria established by the test traceability in the software acceptance test planning documents.
d. Document the results as required by the software acceptance test plan.
e. Use the software acceptance test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(5) Traceability Analysis- Test Phase

The objective of test phase traceability analysis is to ensure completeness and correctness of test activities. The task analyzes relationships in the Integration, Qualification, and Acceptance Test Plans, Designs, Cases, and Procedures for correctness and completeness.

The Test Phase traceability analysis task is as follows:

a. Verify that there is a valid relationship between the Test Plans, Designs, Cases, and Procedures. (Correctness)
b. Verify that all Test Procedures are traceable to the Test Plans. (Completeness)

(6) Hazard Analysis - Test Phase

The objective of hazard analyses is to ensure no new hazard being introduced into the system during test.

The Software Test Phase hazard analysis task is as follows:

a. Verify that the test instrumentation does not introduce new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated or controlled (any unmitigated hazards are documented and addressed as part of system and software operations).
c. Update the hazard analysis.

(7) Security Analysis - Test Phase

The objective of test phase security analysis is to ensure no increase security risk in the test.

The Test Phase security analysis task is as follows:

a. Verify that the implemented system does not increase security risk.

b. Verify the identified security threats and vulnerabilities are prevented, controlled or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of system and software operations).
c. Update the security analysis.

(8) Risk Analysis - Test Phase

The objective of risk analyses is to ensure risk associated with test activities being identified and mitigated.

Procedure of test phase risk analysis is as follows:

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.
c. Update the risk analysis.

(9) Configuration Management- Test Phase

The objective of test phase configuration management (CM) is to track versions oftest documentation including test reportsand various analysis reports, and manage changes during update of test documentation and analysis reports, and test execution process.

The test phase CM task is as follows:

a. Ensure that the defined configuration management strategy has been executed in the Test Phase.
b. Ensure that interfaces with the V&V organization have been adequate to facilitate projects progress.
c. Ensure all changes made to the configuration item (e.g., test procedures and test reports release) baselines have been documented and verified & validated.
d. Ensure the status of items under configuration management is made available throughout the life cycle.

(10) Review and Audit - Test Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028). In the Test Phase, the test plans and procedures, executions process, and test results will be reviewed and audited to ensure that the test plans were followed; and the system and software requirements are fully addressed.

The Development reviews and audit tasks for the Test Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, V&V, and QA) has been followed faithfully.
b. Reviews/audit results (e.g., Test Phase products) should be documented.
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.

d. Improvement and/or lessons learned should be documented as a result of reviews/audit.

B.8 Attributes of Software V&V Test Activities

With the completion of the Implementation Phase, the V&V enters into the Test Phase.

The test phase is an activity in which a system or component is executed under specified conditions, the results are observedor recorded, andan evaluation is made of some aspect of the system or component. Test activity includes test planning, test design, generating test cases, and test procedures generations, which are carried out in various phases. However, test phase V&V activities consist of software integration test and qualification test execution, acceptance test procedure generation and execution, and various analyses (e.g., traceability analysis, hazard analysis, security, and risk analyses) specific to the Test Phase. After the test phase is completed, the software can be installed and tested in the target environment.

Test Phase V&V activities are as follows [IEEE Std 1012]:

1) V&V Software Integration Test Execution
2) V&V Software Qualification Test Execution
3) V&V Software Acceptance Procedure Generation
4) V&V Software Acceptance Test Execution
5) Traceability Analysis V&V – Test Phase
6) Hazard Analysis V&V – Test Phase
7) Security Analysis V&V – Test Phase
8) Risk Analysis V&V – Test Phase
9) Configuration Management V&V – Test Phase
10) Review and Audit V&V – Test Phase
11) V&V Test Phase Activity Summary Report Generation

Additionally, Configuration Management and Reviews & Audits are listed as required V&V tasks.

(1) V&V Software Integration Test Execution

The objective of V&V software integration test execution is to assure that the software requirements and system requirements allocated to software are validated as each software component (e.g., unit or module) is incrementally integrated.

The V&V Software Integration Test Execution task is as follows:

a. Perform V&V software integration testing.
b. Analyze test results to verify that the software components are integrated correctly.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.
d. Document the results as required by the V&V software integration test plan.
e. Use the V&V software integration test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(2) V&V Software Qualification Test Execution

The objective of V&V software qualification test execution is to assure that the integrated software product satisfies its requirements. Software qualification (e.g., demonstration, analysis, inspection, or test) is performed on the complete software element.

The V&V Software Qualification Test Execution task is as follows:

a. Perform V&V software qualification testing.
b. Analyze test results to validate that the software satisfies the system requirements.
c. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.
d. Document the results as required by the V&V software qualification test plan.
e. Use the V&V software qualification test results to validate that the software satisfies the test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(3) V&V Software Acceptance Test Procedure Generation

V&V Software Acceptance Test Procedure specifies a sequence of actions for the execution of the software acceptance test. The V&V task generates Software Acceptance Test Procedure; and validates that the software Acceptance Test Procedure satisfy the criteria specified in the V&V Software Acceptance Test Plan.

The V&V Software Acceptance Test Generation task is as follows:

a. Develop test procedures for V&V software acceptance testing.
b. Continue the tracing required by the V&V software acceptance test plan.
c. Verify that the V&V software acceptance test procedures conform to Project-defined test document purpose, format, and content (e.g., see IEEE Std 829-2008 [B3]).
d. Validate that the V&V software acceptance test procedures satisfy the criteria in the V&V Requirements Specification Phase Attribute (12).

(4) V&V Software Acceptance Test Execution

The objective of V&V software acceptance test execution is to assure that the software satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the integrated software product.

The V&V Software Acceptance Test Execution task is as follows:

a. Perform V&V software acceptance testing.
b. Analyze test results to validate that the software satisfies the system requirements.
c. Validate that the test results trace to test criteria established by the test traceability in the V&V software acceptance test planning documents.
d. Document the results as required by the V&V software acceptance test plan.
e. Use the V&V software acceptance test results to validate that the software satisfies the V&V test acceptance criteria.
f. Document discrepancies between actual and expected test results.

(5) Traceability Analysis V&V- Test Phase

The objective of test phase V&V traceability analyses is to ensure completeness and correctness of V&V test activities. V&V analyzes relationships in the V&V Integration, Qualification, and Acceptance Test Plans, Designs, Cases, and Procedures for correctness and completeness.

The Test Phase V&V traceability analysis task is as follows:

a. Verify that there is a valid relationship between the V&V Test Plans, Designs, Cases, and Procedures. (Correctness)
b. Verify that all V&V Test Procedures are traceable to the V&V Test Plans. (Completeness)

(6) Hazard Analysis V&V- Test Phase

The objective of hazard analyses is to ensure no new hazard being introduced into the system during test.

The Software Test Phase V&V hazard analysis task is as follows:

a. Verify that the test instrumentation does not introduce new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated or controlled (any unmitigated hazards are documented and addressed as part of system and software operations).
c. Update the hazard analysis

(7) Security Analysis V&V – Test Phase

The objective of test phase V&V security analysis is to ensure no increase security risk in the test.

The Test Phase V&V security analysis task is as follows:

a. Verify that the implemented system does not increase security risk.
b. Verify the identified security threats and vulnerabilities are prevented, controlled or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of system and software operations).
c. Update the security analysis.

(8) Risk Analysis V&V – Test Phase

The objective of risk analyses is to ensure risk associated with test activities being identified and mitigated.

Procedure of test phase V&V risk analysis is as follows:

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.
c. Update the risk analysis.

(9) Configuration Management – V&V Test Phase

The objective of test phase V&V configuration management (CM) is to track versions oftest documentation including test reportsand various analysis reports, and manage changes during update of test documentation and analysis reports, and test execution process.

The test phase V&V CM task is as follows:

a. Ensure that the defined configuration management strategy has been executed in the Test Phase.
b. Ensure that interfaces with the Development organization have been adequate to facilitate projects progress.
c. Ensure all changes made to the configuration item (e.g., test procedures and test reports release) baselines have been documented and verified & validated.
d. Ensure the status of items under configuration management is made available throughout the life cycle.

(10) Review and Audit V&V – Test Phase

A review is a process or meeting during which a system/software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval. Review types include management reviews, technical reviews, inspections, walk-throughs, and audits (IEEE Std 1028). In the Test Phase, the test plans and procedures, executions process, and test results will be reviewed and audited to ensure that the test plans were followed; and the system and software requirements are fully addressed.

The V&V reviews and audit tasks for the Test Phase are as follows:

a. Reviews/audit plan including protocols for interfacing with other organizations (i.e., Project, Development, and QA) has been followed faithfully.
b. Reviews/audit results (e.g., Development and V&V Test Phase products) should be documented.
c. Anomaly reports have been generated during the reviews/audit process and dispositioned in accordance with the audit plan.
d. Improvement and/or lessons learned should be documented as a result of reviews/audit.

(11) V&V Test Phase Activity Summary Report Generation

V&V Test Phase activity summary report summarizes the results of V&V tasks performed in the phase.

Specific task for the V&V Test Phase Activity Summary Report Generation is as follows:

a. Document software V&V activities conducted in the Test phase.
b. Provide background information of standards, and source documents with revisions.
c. Provide confirmation that the V&V plan was followed.
d. Document the individual who performed the task, the portion of the task performed by the individual (if multiple performers), and the date that the task was performed.

e. Summarize anomaly issues identified, the resolution process of them, and recommendations whether to proceed to the Installation and Checkout phase of the software development lifecycle.

B.9 Attributes of Software Installation and Checkout Activities

In installation and checkout, the software product is installed and tested in the target environment. The Software Installation and Checkout activity supports the system installation activities.

The objective of Software Installation and Checkout is to verify and validate the correctness of the software installation in the target environment.

The Software Installation and Checkout task is as follows [IEEE Std 1012]:

1. Installation Procedure Generation
2. Installation and Checkout
3. Hazard Analysis - Installation and Checkout Phase
4. Security Analysis - Installation and Checkout Phase
5. Risk Analysis - Installation and Checkout Phase

(1) Installation Procedure Generation

In Installation Procedure Generation, a sequence of actions for the installation of the software is specified. The task generates Software Installation Procedures; and validates that the SoftwareInstallation Procedures satisfy the system and software safety and security requirements as specified in the SyRS and SRS; as well as plant's installation requirements.

The Installation Procedure Generation task is as follows:

a. Develop procedural steps for the software installation procedures.
b. The installation procedures should satisfy system and software safety and security requirements.
c. The installation procedures should meet the installation requirements in plant's technical specification.
d. Update/revise the Software Installation Procedures per various analysis results (e.g., hazard analysis, security analysis, and risk analysis) and V&V recommendations.

(2) Installation and Checkout

During Installation and Checkout, the software is installed in the target environment.

The objective of Installation and Checkout is to assurethe correctness of the installed software version, no adversary impact of the new software version to the system, and requirements for continuous operation and service during transition are met.

The Installationand Checkout task is as follows:

a. Install the software in the target environment per the procedural steps as specified in the Software Installation Procedures.

Note: If installation is performed by the customer or a third party, then the delineation of responsibility between the development organization and the customer or the third party should be defined in such a way that misunderstandings in communications between the two organizations are kept to a minimum. (BTP 7-14)

b. Verify that the software code and databases initialize, execute, and terminate as specified.
c. In the transition from one version of software to the next, validate that the software can be replaced with the new version without adversely affecting or degrading the functionality of the remaining system components.
d. Verify the requirements for continuous operation and service during transition, including requirements for user notification are met.
e. Anomalies discovered during installation should be documented and resolved prior to placing the software into operation. (BTP 7-14)

(3) Hazard Analysis – Installation and Checkout Phase

The objective of hazard analyses is to ensure no new hazard being introduced into the system during installation and checkout.

The Software Installation and Checkout hazard analysis is as follows:

a. Verify that the installation procedures and installation environment does not introduce new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated or controlled (any unmitigated hazards are documented and addressed as part of system and software operations).
c. Update the hazard analysis.

(4) Security Analysis – Installation and Checkout Phase

The objective of Installation and Checkout phase security analysis is to ensure no increase security risk in the Installation and Checkout

The specific task is as follows:

a. Verify that the installed software does not introduce new or increased vulnerabilities or security risks to the overall system.
b. Verify the identified security threats and vulnerabilities are prevented, controlled or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of system and software operations).
c. Update the security analysis.

(5) Risk Analysis – Installation and Checkout Phase

The objective of Installation and Checkout analyses is to ensure risk associated with Installation and Checkout activities being identified and mitigated.

Procedure of risk analysis in Installation and Checkout Phase is as follows:

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.
c. Update the risk analysis.

B.10 Attributes of Software Installation and Checkout V&V Activities

In installation and checkout, the software product is installed and tested in the target environment. The Software Installation and Checkout V&V activity supports the software system's installation activities.

The objective of Software Installation and Checkout V&V is to verify and validate the correctness of the software installation in the target environment.

The Software Installation and Checkout V&V task is as follows [IEEE Std 1012]:

Installation Configuration Audit V&V
Installation Checkout V&V
Hazard Analysis V&V – Installation and Checkout Phase
Security Analysis V&V – Installation and Checkout Phase
Risk Analysis V&V – Installation and Checkout Phase
V&V Installation and Checkout Phase Activity Summary Report Generation
V&V Final Report Generation

(1) Installation Configuration Audit V&V

The objective of the V&V Installation Configuration Audit is to confirm the integrity of a systems product prior to delivery. This audit is held prior to software delivery to verify that all requirements specified in the Software Requirements Specification have been met and that the software and its documentation are internally consistent.

The V&V Installation Configuration Audit task is as follows:

a. Verify that all software products required to correctly install and operate the software are present in the installation package.
b. Verify that the software configuration items' actual functionality and performance is consistent with the relevant requirement specification.
c. Evaluate a software product's configuration items to confirm that all components in the as-built version map to their specifications.
d. Validate that all site-dependent parameters or conditions are correctly specified in the relevant requirements specifications.

(2) Installation CheckoutV&V

The objective of V&V Installation Checkout Verify is to assure the correctness of the installed software version, no adversary impact of the new software version to the system, and requirements for continuous operation and service during transition are met.

The V&V Installation Checkout task is as follows:

a. Conduct analyses or tests to verify that the installed software corresponds to the software subjected to V&V.
b. Verify that the software code and databases initialize, execute, and terminate as specified.
c. In the transition from one version of software to the next, validate that the software can be replaced with the new version without adversely affecting or degrading the functionality of the remaining system components.
d. Verify the requirements for continuous operation and service during transition, including requirements for user notification are met.

(3) Hazard Analysis V&V – Installation and Checkout Phase

The objective of hazard analyses V&V is to ensure no new hazard being introduced into the system during installation and checkout.

The V&V task is as follows:

a. Verify that the installation procedures and installation environment does not introduce new hazards.
b. Assess the identified mitigation strategies to verify each hazard is prevented, mitigated or controlled (any unmitigated hazards are documented and addressed as part of system and software operations).
c. Update the hazard analysis.

(4) Security Analysis V&V – Installation and Checkout Phase

The objective of Installation and Checkout phase security analysis V&V is to ensure no increase security risk in the Installation and Checkout

The specific task is as follows:

a. Verify that the installed software does not introduce new or increased vulnerabilities or security risks to the overall system.
b. Verify the identified security threats and vulnerabilities are prevented, controlled or mitigated (any unmitigated threats and vulnerabilities are documented and addressed as part of system and software operations).
c. Update the security analysis.

(5) Risk Analysis V&V – Installation and Checkout Phase

The objective of Installation and Checkout risk analyses V&V is to ensure risk associated with Installation and Checkout activities being identified and mitigated.

Procedure of risk analysis in Installation and Checkout Phase is as follows:

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.
c. Update the risk analysis.

(6) V&V Installation and Checkout Phase Activity Summary Report Generation

V&V Installation and Checkout Phase activity summary report summarizes the results of V&V tasks performed in the phase.

Specific task for the V&V Installation and Checkout Phase Activity Summary Report Generation is as follows:

a. Document software V&V activities conducted in the Requirements Specifications phase.
b. Provide background information of standards, and source documents with revisions.
c. Provide confirmation that the V&V plan was followed.
d. Document the individual who performed the task, the portion of the task performed by the individual (if multiple performers), and the date that the task was performed.
e. Summarize anomaly issues identified, the resolution process of them, and recommendations.

(7) V&V Final Report Generation

The V&V Final Report provides summary regarding implementation of the SVVP at conclusion of V&V effort, which should include:

a. Identification of all V&V documentation.
b. Summary of all Lifecycle V&V Activities.
c. Summary of all V&V task results.
d. Summary of all V&V activity anomalies and their resolutions.
e. Assessment of overall software quality based upon a review of the V&V activity.
f. Lessons learned/process improvements and recommendations, if any, regarding the overall development process for future nuclear safety-related application software projects.

APPENDIX C
LISTS OF EXPERTS

C.1 BBN Structure and Theory Experts

Organization	Experts
Doosan Heavy Industry, South Korea	Koo, Seo Ryong
Idaho National Laboratory	Marts, George
Joongbu University, South Korea	Son, Han Seong
KEPCO Engineering and Construction Company, South Korea	Baek, Seung Min
Korea Advanced Institute of Science and Technology (KAIST), South Korea	Baik, Jongmoon
NASA	Costello, Kenneth
NASA	Vorndran, Kenneth
NRC	Stattel, Richard
NUV Technology LLC	Yang, Steve
Queen Mary University of London	Fenton, Norman
Raytheon Company	Gullo, Lou
Raytheon Company	Peterson, Jon
SoftRel LLC	Neufelder, A.M.
University of California, Los Angeles	Mosleh, Ali

C.2 Generic Experts on Model Parameters

1.1	Organization	1.2	Experts
1.3	AREVA NP	1.4	Bates, Lionel
1.5	Doosan Heavy Industry, South Korea	1.6	Koo, Seo Ryong
1.7	Joongbu University, South Korea	1.8	Son, Han Seong
1.9	KEPCO Engineering and Construction Company, South Korea	1.10	Baek, Seung Min
1.11	Korea Advanced Institute of Science and Technology (KAIST), South Korea	1.12	Baik, Jongmoon
1.13	Queen Mary University of London	1.14	Fenton, Norman
1.15	Southern Engineering Services, Inc.	1.16	Geddes, Bruce and Sean Kelley

C.3 Specific Experts

System	Expert	Organization
Loop Operating and Control System (LOCS)	Marts, George	Idaho National Laboratory
Korea Nuclear Instrumentation and Control System (KNICS)	Son, Han Seong	Joongbu University, South Korea

APPENDIX D
DETAILED EXPERTS' OPINION DISTRIBUTION FITTING

Table D-1 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the Prior Distribution of the Development Quality by Phase

Phase	Development Quality	Rank	1	2	3	4
Requirement	High	Dist. Name	lognormal	gamma	beta	weibull
		NLogL	-7.7556	-7.6517	-7.612	-7.4083
		BIC	-11.9276	-11.7198	-11.6405	-11.233
		AIC	-11.5111	-11.3034	-11.224	-10.8165
	Medium	Dist. Name	beta	weibull	normal	logistic
		NLogL	-2.3077	-2.0777	-1.8167	-1.7579
		BIC	-1.0319	-0.57194	-0.04984	0.067664
		AIC	-0.61537	-0.15546	0.36664	0.48414
	Low	Dist. Name	exponential	lognormal	gamma	weibull
		NLogL	-3.4117	-4.2837	-3.8558	-3.6822
		BIC	-5.0316	-4.9838	-4.1281	-3.7808
		AIC	-4.8234	-4.5674	-3.7116	-3.3643
Design	High	Dist. Name	beta	gamma	weibull	exponential
		NLogL	-4.3587	-3.5409	-2.9095	-1.9302
		BIC	-5.134	-3.4982	-2.2355	-2.0686
		AIC	-4.7175	-3.0817	-1.819	-1.8603
	Medium	Dist. Name	beta	weibull	normal	logistic
		NLogL	-0.96094	-0.37359	-0.24473	-0.00206
		BIC	1.6616	2.8363	3.0941	3.5794
		AIC	2.0781	3.2528	3.5105	3.9959
	Low	Dist. Name	weibull	normal	logistic	beta
		NLogL	-8.3188	-8.2535	-8.1129	-7.757
		BIC	-13.054	-12.9234	-12.6423	-11.9306
		AIC	-12.6375	-12.507	-12.2258	-11.5141
Implementation	High	Dist. Name	beta	gamma	weibull	exponential
		NLogL	-3.9496	-2.7722	-2.136	-0.74327
		BIC	-4.3156	-1.9609	-0.68843	0.30521
		AIC	-3.8991	-1.5444	-0.27195	0.51345
	Medium	Dist. Name	beta	weibull	gamma	normal
		NLogL	-0.41816	0.07155	0.24137	0.34764
		BIC	2.7472	3.7266	4.0663	4.2788
		AIC	3.1637	4.1431	4.4827	4.6953
	Low	Dist. Name	beta	weibull	gamma	lognormal
		NLogL	-6.8029	-6.7719	-6.7551	-6.5938
		BIC	-10.0222	-9.9603	-9.9266	-9.604
		AIC	-9.6058	-9.5438	-9.5102	-9.1875
Test	High	Dist. Name	beta	gamma	weibull	exponential
		NLogL	-4.9335	-4.2335	-3.726	-2.1206
		BIC	-6.2835	-4.8834	-3.8685	-2.4495
		AIC	-5.867	-4.4669	-3.452	-2.2413
	Medium	Dist. Name	beta	normal	weibull	logistic
		NLogL	-0.52479	0.34571	0.37135	0.45841
		BIC	2.5339	4.2749	4.3262	4.5003
		AIC	2.9504	4.6914	4.7427	4.9168
	Low	Dist. Name	exponential	beta	gamma	weibull
		NLogL	-5.9413	-6.201	-5.9745	-5.9443
		BIC	-10.0908	-8.8185	-8.3654	-8.3051
		AIC	-9.8825	-8.402	-7.9489	-7.8886
Installation and Checkout	High	Dist. Name	exponential	beta	weibull	gamma
		NLogL	0.20784	-0.44703	0.057045	0.11427
		BIC	1.802	1.8785	2.8867	3.0011

		AIC	2.4157	3.1059	4.1141	4.2285
	Medium	Dist. Name	beta	weibull	gamma	lognormal
		NLogL	-0.08532	0.50018	0.51179	0.60413
		BIC	2.6019	3.7729	3.7962	3.9809
		AIC	3.8294	5.0004	5.0236	5.2083
	Low	Dist. Name	logistic	normal	weibull	beta
		NLogL	-7.7664	-7.6997	-7.5889	-7.1577
		BIC	-12.7602	-12.6269	-12.4052	-11.5429
		AIC	-11.5328	-11.3995	-11.1778	-10.3155

Table D-2 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the Prior Distribution of the V&V Quality by Phase

Phase	Development Quality	Rank	1	2	3	4
Requirement	High	Dist. Name	beta'	'gamma'	'weibull'	'normal'
		NLogL	-4.8455'	'-4.4316'	'-4.2309'	'-3.4925'
		BIC	-5.7991'	'-4.9713'	'-4.5701'	'-3.0931'
		AIC	-5.6909'	'-4.8631'	'-4.4619'	'-2.9849'
	Medium	Dist. Name	beta'	'weibull'	'gamma'	'normal'
		NLogL	-1.5441'	'-1.1417'	'-0.96866'	'-0.94867'
		BIC	0.80363'	'1.6083'	'1.9545'	'1.9945'
		AIC	0.91181'	'1.7165'	'2.0627'	'2.1027'
	Low	Dist. Name	gamma'	'beta'	'weibull'	'lognormal'
		NLogL	-4.2313'	'-4.2097'	'-4.19'	'-3.9344'
		BIC	-4.5708'	'-4.5275'	'-4.4882'	'-3.9769'
		AIC	-4.4626'	'-4.4193'	'-4.38'	'-3.8687'
Design	High	Dist. Name	beta'	'gamma'	'weibull'	'logistic'
		NLogL	-3.5973'	'-3.0924'	'-2.8552'	'-1.5236'
		BIC	-3.3028'	'-2.293'	'-1.8185'	'0.84463'
		AIC	-3.1946'	'-2.1848'	'-1.7103'	'0.95281'
	Medium	Dist. Name	beta'	'weibull'	'gamma'	'lognormal'
		NLogL	-1.1983'	'-0.8373'	'-0.77224'	'-0.65659'
		BIC	1.4951'	'2.2172'	'2.3473'	'2.5786'
		AIC	1.6033'	'2.3254'	'2.4555'	'2.6868'
	Low	Dist. Name	gamma'	'lognormal'	'beta'	'weibull'
		NLogL	-4.3503'	'-4.3223'	'-4.2604'	'-4.2311'
		BIC	-4.8088'	'-4.7528'	'-4.6291'	'-4.5704'
		AIC	-4.7006'	'-4.6446'	'-4.5209'	'-4.4622'
Implementation	High	Dist. Name	beta'	'gamma'	'weibull'	'lognormal'
		NLogL	-2.7556'	'-1.8441'	'-1.5273'	'-0.15936'
		BIC	-1.6194'	'0.20364'	'0.83727'	'3.5731'
		AIC	-1.5112'	'0.31181'	'0.94545'	'3.6813'
	Medium	Dist. Name	beta'	'weibull'	'normal'	'gamma'
		NLogL	-0.75823'	'-0.48945'	'-0.29615'	'-0.28117'
		BIC	2.3754'	'2.9129'	'3.2995'	'3.3295'
		AIC	2.4835'	'3.0211'	'3.4077'	'3.4377'
	Low	Dist. Name	lognormal'	'gamma'	'beta'	'weibull'
		NLogL	-6.421'	'-6.0457'	'-5.8757'	'-5.7803'
		BIC	-8.9503'	'-8.1996'	'-7.8597'	'-7.6687'
		AIC	-8.8421'	'-8.0914'	'-7.7515'	'-7.5605'
Test	High	Dist. Name	beta'	'gamma'	'weibull'	'lognormal'
		NLogL	-3.4601'	'-2.9098'	'-2.709'	'-1.6618'
		BIC	-3.0285'	'-1.9278'	'-1.5262'	'0.56823'
		AIC	-2.9203'	'-1.8196'	'-1.4181'	'0.67641'
	Medium	Dist. Name	beta'	'normal'	'logistic'	'weibull'
		NLogL	-0.29368'	'0.68809'	'0.76042'	'0.9225'
		BIC	3.3045'	'5.268'	'5.4127'	'5.7368'
		AIC	3.4126'	'5.3762'	'5.5208'	'5.845'
	Low	Dist. Name	beta'	'weibull'	'gamma'	'normal'
		NLogL	-5.9479'	'-5.8538'	'-5.7512'	'-5.2884'
		BIC	-8.0039'	'-7.8158'	'-7.6106'	'-6.685'
		AIC	-7.8958'	'-7.7076'	'-7.5024'	'-6.5769'
Installation and Checkout	High	Dist. Name	beta'	'weibull'	'gamma'	'lognormal'
		NLogL	-0.95432'	'-0.81955'	'-0.81255'	'-0.60489'
		BIC	1.9832'	2.2527'	'2.2667'	'2.682'
		AIC	2.0914'	2.3609'	'2.3749'	'2.7902'
	Medium	Dist. Name	beta'	weibull'	'gamma'	'normal'
		NLogL	-0.17752'	0.38864'	'0.47152'	'0.64694'
		BIC	3.5368'	4.6691'	'4.8349'	'5.1857'
		AIC	3.645'	4.7773'	'4.943'	'5.2939'

Phase	Development Quality	Rank	1	2	3	4
	Low	Dist. Name	beta'	gamma'	'weibull'	'lognormal'
		NLogL	-4.3285'	-4.2999'	'-4.2991'	'-4.1087'
		BIC	-4.7652'	-4.7079'	'-4.7064'	'-4.3257'
		AIC	-4.657'	-4.5997'	'-4.5983'	'-4.2175'

Table D-3 Beta Distribution Fit for the NPT of the Prior Distribution of Development Quality

Phase	Development Quality	α	β
Requirement	High	4.42	22.04
	Medium	4.47	2.73
	Low	1.25	4.49
Design	High	0.56	1.74
	Medium	2.40	1.81
	Low	4.12	22.06
Implementation	High	0.47	1.14
	Medium	1.70	1.56
	Low	2.82	15.00
Test	High	0.45	1.29
	Medium	1.75	1.19
	Low	0.83	5.42
Installation and Checkout	High	0.99	1.66
	Medium	1.28	1.05
	Low	2.83	32.73

Table D-4 Beta Distribution Fit for the NPT of the Prior Distribution of V&V Quality

Phase	V&V Quality	α	β
Requirement	High	0.67	3.23
	Medium	3.21	2.21
	Low	2.10	7.19
Design	High	0.60	2.15
	Medium	2.70	2.11
	Low	1.78	6.66
Implementation	High	0.49	1.27
	Medium	2.12	1.82
	Low	1.90	9.90
Test	High	0.42	1.14
	Medium	1.45	1.06
	Low	1.17	7.14
Installation and Checkout	High	1.05	2.26
	Medium	1.45	1.10
	Low	1.28	8.56

Table D-5 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the Number of Function Point by Complexity State

Complexity State	Rank	1	2	3	4
High (1000≤FP≤1500)	Dist. Name	Beta	gamma	lognormal	weibull
	NLogL	-5.9762	-5.9234	-5.9151	-5.8768
	BIC	-8.0607	-7.955	-7.9384	-7.8619
	AIC	-7.9525	-7.8468	-7.8303	-7.7537
Medium (100≤FP≤1000)	Dist. Name	Gamma	lognormal	beta	weibull
	NLogL	-2.7659	-2.7318	-2.7284	-2.6215
	BIC	-1.64	-1.5717	-1.565	-1.3513
	AIC	-1.5319	-1.4635	-1.4569	-1.2431
Low (FP≤100)	Dist. Name	Beta	lognormal	exponential	gamma
	NLogL	-2.8571	-2.7442	-1.7693	-2.7336
	BIC	-1.8225	-1.5965	-1.5928	-1.5753
	AIC	-1.7143	-1.4883	-1.5387	-1.4671

Table D-6 Beta Distribution Fit for the NPT of the Number of Function Point

Complexity State	α	β
High (1000≤FP≤1500)	2.7871	10.2211
Medium (100≤FP≤1000)	4.0749	4.0253
Low (FP≤100)	1.5870	3.9460

Table D-7 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the Attribute Nodes

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Software Development Planning	Dist Name	weibull	logistic	lognormal	gamma	logistic	normal	weibull	beta	lognormal	gamma	normal	logistic	normal	logistic	beta	gamma	gamma	beta
	BIC	-7.32	-6.49	-4.42	-4.36	-21.81	-21.53	-14.20	-13.90	-8.28	-8.21	-9.07	-8.63	-12.28	-12.18	-1.46	-1.38	-1.04	-1.03
Development of a Concept Documentation	Dist Name	gamma	lognormal	logistic	weibull	logistic	normal	weibull	normal	lognormal	gamma	lognormal	gamma	logistic	normal	beta	weibull	gamma	lognormal
	BIC	-5.03	-5.02	-11.45	-11.31	-11.99	-11.39	-11.23	-10.86	-8.28	-8.21	-19.16	-18.32	-21.81	-21.53	-3.55	-3.55	-3.03	-3.02
Development of Software Requirements Specifications	Dist Name	lognormal	gamma	weibull	normal	normal	logistic	beta	gamma	lognormal	gamma	lognormal	gamma	normal	logistic	beta	weibull	gamma	lognormal
	BIC	-8.77	-8.59	-9.17	-8.75	-19.44	-19.16	-11.91	-11.85	-10.73	-10.52	-15.29	-14.77	-19.44	-19.16	-4.29	-4.22	-4.59	-4.52
Traceability Analysis - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	beta	weibull	normal	logistic	beta	weibull	lognormal	gamma
	BIC	-2.79	-2.37	-7.56	-7.44	-6.81	-6.31	-7.72	-7.36	-10.73	-10.52	-10.21	-10.20	-9.16	-8.61	-6.71	-6.61	-3.66	-3.51
Criticality Analysis - Requirements Specifications Phase	Dist Name	beta	lognormal	weibull	beta	normal	logistic	normal	logistic	beta	weibull	normal	logistic	normal	logistic	logistic	weibull	gamma	lognormal
	BIC	0.55	0.93	-3.82	-3.80	-7.58	-7.09	-4.67	-4.15	-5.47	-5.15	-8.05	-7.65	-7.58	-7.09	-7.56	-7.44	-1.86	-1.80
Hazard Analysis - Requirements Specifications Phase	Dist Name	beta	weibull	logistic	weibull	logistic	normal	normal	logistic	beta	weibull	beta	weibull	logistic	normal	logistic	weibull	logistic	beta
	BIC	-0.28	-0.27	-6.58	-6.35	-6.35	-5.53	-7.14	-6.63	-3.71	-3.60	-11.60	-11.59	-5.93	-4.92	-8.20	-7.97	-0.86	-0.86
Security Analysis - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	gamma	beta	beta	gamma	logistic	normal	logistic	weibull	gamma	lognormal
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-5.33	-4.97	-6.57	-6.51	-15.14	-15.13	-10.37	-9.63	-4.71	-4.67	-1.68	-1.55
Risk Analysis - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	beta	weibull	lognormal	beta	normal	logistic	weibull	normal	beta	weibull
	BIC	0.24	0.39	-6.50	-6.26	-2.90	-2.23	-3.69	-2.96	-0.77	-0.48	-9.71	-9.67	-4.07	-3.72	-5.54	-5.29	0.57	1.03
System/Software Qualification Test Plan Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	lognormal	gamma	beta	weibull	normal	logistic	logistic	normal	weibull	normal	beta	gamma
	BIC	-2.86	-2.44	-4.51	-4.48	-23.25	-22.41	-10.68	-10.31	-2.54	-1.97	-6.37	-5.68	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
System/Software Acceptance Test Plan Generation	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	weibull	normal	gamma	beta	normal	logistic	logistic	normal	logistic	weibull	logistic	normal
	BIC	-5.27	-5.21	-6.40	-6.22	-23.96	-23.46	-12.35	-11.95	-7.93	-7.92	-11.72	-11.27	-10.37	-9.63	-8.85	-8.67	-2.49	-2.21
Configuration Management - Requirements Specifications Phase	Dist Name	weibull	beta	weibull	logistic	normal	logistic	weibull	normal	beta	weibull	normal	logistic	normal	logistic	logistic	lognormal	weibull	beta
	BIC	-3.79	-3.54	-24.14	-23.94	-4.83	-4.74	-11.96	-11.61	-3.43	-3.20	-7.24	-7.02	-4.86	-4.41	-32.95	-32.07	-4.53	-4.33
Review and Audit - Requirements Specifications Phase	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	normal	logistic	beta	gamma	beta	gamma	normal	logistic	weibull	normal	gamma	lognormal
	BIC	-0.24	-0.07	-10.73	-10.55	-4.86	-4.41	-4.90	-4.52	-0.45	-0.17	-10.93	-10.85	-3.92	-3.38	-5.45	-5.30	0.09	0.16
Development of a Software Architecture Description	Dist Name	beta	weibull	weibull	beta	logistic	normal	weibull	logistic	lognormal	gamma	weibull	normal	normal	logistic	weibull	beta	beta	gamma
	BIC	-3.07	-2.48	-8.04	-7.99	-11.16	-11.12	-16.44	-16.02	-9.74	-9.46	-15.57	-15.03	-8.47	-7.94	-7.20	-6.96	-2.16	-2.14
Development of Software Design Description	Dist Name	beta	weibull	beta	weibull	normal	logistic	weibull	normal	lognormal	gamma	weibull	beta	normal	logistic	beta	lognormal	beta	weibull

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
	BIC	-6.36	-5.59	-8.35	-8.25	-23.96	-23.46	-13.64	-13.25	-10.69	-10.64	-11.45	-11.30	-6.41	-5.94	-6.83	-6.82	-0.82	1.30
Traceability Analysis - Design Phase	Dist Name	gamma	lognormal	weibull	normal	normal	logistic	normal	logistic	beta	weibull	lognormal	gamma	normal	logistic	beta	weibull	beta	weibull
	BIC	-0.78	-0.77	-5.84	-5.71	-7.58	-7.09	-3.55	-3.03	-4.67	-4.35	-15.83	-15.68	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
Criticality Analysis - Design Phase	Dist Name	lognormal	gamma	weibull	beta	normal	logistic	normal	logistic	beta	weibull	weibull	normal	normal	logistic	weibull	beta	beta	weibull
	BIC	-0.45	-0.33	-4.51	-4.48	-8.57	-8.04	-5.66	-5.55	-5.13	-4.78	-10.10	-9.82	-6.93	-6.42	-6.63	-6.37	-0.44	0.62
Hazard Analysis - Design Phase	Dist Name	lognormal	gamma	weibull	normal	normal	logistic	normal	logistic	lognormal	gamma	weibull	normal	logistic	normal	weibull	beta	beta	weibull
	BIC	-3.56	-3.52	-4.72	-4.65	-23.96	-23.46	-8.92	-8.41	-8.28	-8.21	-12.35	-11.95	-9.84	-9.09	-6.63	-6.37	-1.53	-0.63
Security Analysis - Design Phase	Dist Name	gamma	lognormal	logistic	weibull	logistic	normal	logistic	normal	beta	weibull	weibull	beta	logistic	normal	beta	weibull	beta	weibull
	BIC	-1.68	-1.55	-6.12	-6.01	-11.28	-10.37	-6.71	-6.11	-6.67	-6.45	-14.12	-14.03	-9.84	-9.09	-1.49	-1.32	0.62	1.61
Risk Analysis - Design Phase	Dist Name	gamma	lognormal	weibull	normal	normal	logistic	logistic	normal	beta	weibull	weibull	normal	logistic	normal	weibull	beta	beta	weibull
	BIC	-0.78	-0.77	-5.84	-5.71	-7.58	-7.09	-3.00	-2.64	-0.64	-0.32	-12.35	-11.95	0.55	0.72	-4.73	-4.56	2.48	4.25
Software Component Test Plan Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	beta	weibull	weibull	normal	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-7.57	-7.07	-2.24	-1.72	-11.96	-11.61	-8.57	-8.04	-11.45	-11.31	-3.56	-3.52
Software Integration Test Plan Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	lognormal	gamma	weibull	normal	logistic	normal	weibull	logistic	gamma	lognormal
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-9.65	-9.13	-5.38	-5.36	-13.25	-12.84	-11.99	-11.39	-11.33	-11.20	-4.59	-4.52
Software Component Test Design Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	normal	logistic	beta	weibull	lognormal	gamma	logistic	normal	beta	weibull	beta	weibull
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-11.72	-11.27	-5.07	-4.73	-13.86	-13.44	-11.39	-10.78	-7.48	-7.40	-2.89	-1.62
Software Integration Test Design Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	normal	logistic	gamma	lognormal	beta	gamma	logistic	normal	weibull	normal	beta	weibull
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-11.72	-11.27	-4.48	-4.45	-13.22	-13.17	-11.39	-10.78	-8.18	-7.86	-3.07	-2.48
Software Qualification Test Design Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-6.26	-6.19	-3.34	-3.11	-16.22	-15.86	-8.57	-8.04	-7.60	-7.32	-1.23	-1.03
Software Acceptance Test Design Generation	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	logistic	normal	gamma	lognormal	lognormal	gamma	normal	logistic	weibull	logistic	gamma	lognormal
	BIC	-4.59	-4.52	-6.46	-6.41	-23.96	-23.46	-6.51	-6.48	-3.24	-3.21	-17.21	-17.18	-8.57	-8.04	-11.33	-11.20	-3.03	-3.02
Configuration Management - Design Phase	Dist Name	weibull	logistic	weibull	logistic	logistic	normal	lognormal	gamma	beta	weibull	weibull	beta	normal	logistic	logistic	weibull	gamma	lognormal
	BIC	-4.95	-4.24	-24.14	-23.94	-6.49	-5.38	-12.20	-12.14	-3.13	-2.88	-11.28	-11.07	-4.86	-4.41	-10.73	-10.55	-0.24	-0.07
Reviews and Audit - Design Phase	Dist Name	beta	weibull	weibull	normal	normal	logistic	normal	logistic	gamma	beta	lognormal	gamma	normal	logistic	weibull	beta	beta	weibull
	BIC	1.14	2.04	-6.20	-5.97	-3.46	-2.90	-4.90	-4.52	-0.40	-0.39	-12.96	-12.62	-3.30	-2.71	-6.68	-6.61	1.18	2.65
Source Code and Source Code Documentation Generation	Dist Name	gamma	lognormal	weibull	beta	logistic	normal	normal	logistic	beta	gamma	weibull	normal	logistic	normal	weibull	beta	beta	weibull
	BIC	-2.42	-2.39	-6.04	-5.81	-11.99	-11.39	-11.49	-11.15	-9.09	-9.06	-10.11	-10.09	-9.84	-9.09	-6.63	-6.37	-1.53	-0.63

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Traceability Analysis - Implementation Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	gamma	beta	beta	gamma	normal	logistic	weibull	beta	beta	weibull
	BIC	-1.76	-1.67	-6.12	-6.01	-8.43	-7.97	-3.55	-3.03	-6.48	-6.40	-13.82	-13.78	-6.93	-6.42	-6.63	-6.37	-0.44	0.62
Criticality Analysis - Implementation Phase	Dist Name	lognormal	gamma	weibull	beta	normal	logistic	normal	logistic	gamma	lognormal	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-0.45	-0.33	-4.51	-4.48	-8.57	-8.04	-5.66	-5.55	-5.66	-5.64	-13.89	-13.54	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Hazard Analysis - Implementation Phase	Dist Name	gamma	lognormal	logistic	weibull	logistic	normal	logistic	normal	gamma	beta	weibull	beta	logistic	normal	weibull	beta	beta	weibull
	BIC	-3.03	-3.02	-6.46	-6.41	-11.99	-11.39	-6.32	-6.11	-6.57	-6.51	-14.12	-14.03	-9.84	-9.09	-6.71	-6.63	-1.60	-0.51
Security Analysis - Implementation Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	beta	logistic	normal	beta	weibull	beta	weibull
	BIC	-3.95	-3.83	-6.40	-6.22	-11.99	-11.39	-9.82	-9.53	-10.73	-10.52	-12.73	-12.56	-9.84	-9.09	-6.17	-6.01	-1.47	0.23
Risk Analysis - Implementation Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	beta	gamma	lognormal	gamma	logistic	normal	weibull	normal	beta	normal
	BIC	-3.31	-3.04	-6.74	-6.49	-8.75	-8.14	0.77	0.83	0.25	0.52	-10.22	-9.66	2.35	2.76	-6.24	-5.98	3.23	5.88
Component Test Case Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	logistic	normal	lognormal	gamma	weibull	logistic	normal	logistic	weibull	normal	beta	gamma
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-11.94	-11.72	-8.28	-8.21	-21.78	-21.68	-8.57	-8.04	-7.60	-7.32	-1.23	-1.03
Software Integration Test Case Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	logistic	normal	lognormal	gamma	weibull	logistic	logistic	normal	weibull	normal	beta	gamma
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-11.94	-11.72	-8.28	-8.21	-21.78	-21.68	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Software Qualification Test Case Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	logistic	normal	lognormal	gamma	weibull	logistic	logistic	normal	weibull	normal	beta	gamma
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-10.95	-10.86	-6.94	-6.85	-20.16	-19.71	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Software Acceptance Test Case Generation	Dist Name	beta	gamma	weibull	beta	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-1.44	-0.93	-4.51	-4.48	-11.39	-10.78	-6.11	-6.00	-5.04	-4.81	-17.85	-17.17	-3.74	-3.57	-7.48	-7.18	0.33	1.66
Software Component Test Procedure Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	lognormal	gamma	lognormal	gamma	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-14.36	-14.33	-10.73	-10.52	-14.36	-14.33	-8.57	-8.04	-7.60	-7.32	-1.23	-1.03
Software Integration Test Procedure Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-14.96	-14.88	-10.73	-10.52	-18.79	-18.28	-9.51	-9.00	-11.45	-11.31	-3.95	-3.83
Software Qualification Test Procedure Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	weibull	normal	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	lognormal	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-13.64	-13.25	-6.94	-6.85	-15.01	-14.88	-7.47	-6.68	-7.60	-7.32	-0.45	-0.33
Configuration Management	Dist Name	beta	weibull	weibull	logistic	normal	logistic	weibull	normal	gamma	logistic	weibull	logistic	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-3.37	-3.33	-24.14	-23.94	-4.86	-4.41	-11.96	-11.61	-4.79	-4.66	-13.94	-13.64	-5.49	-5.15	-10.73	-10.55	-0.80	-0.72
Reviews & Audits	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	normal	logistic	beta	weibull	beta	weibull	normal	logistic	weibull	logistic	beta	gamma
	BIC	-0.24	-0.07	-10.73	-10.55	-4.86	-4.41	-6.77	-6.18	-1.13	-0.89	-10.36	-10.25	-4.68	-4.19	-8.38	-8.07	0.41	0.53
Software Component Test Execution	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	lognormal	gamma	beta	weibull	lognormal	gamma	normal	logistic	beta	lognormal	beta	weibull

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
	BIC	-7.01	-5.93	-10.12	-9.48	-23.96	-23.46	-14.36	-14.33	-9.90	-9.65	-21.88	-21.26	-6.41	-5.94	-6.83	-6.82	-0.82	1.30
Software Integration Test Execution	Dist Name	beta	weibull	lognormal	gamma	logistic	normal	lognormal	gamma	beta	weibull	lognormal	gamma	normal	logistic	beta	weibull	beta	weibull
	BIC	-4.17	-2.28	-10.12	-9.48	-11.20	-10.37	-10.61	-10.25	-4.55	-4.10	-17.21	-17.18	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
Software Qualification Test Execution	Dist Name	beta	weibull	beta	weibull	logistic	normal	weibull	normal	beta	gamma	weibull	beta	normal	logistic	beta	weibull	beta	weibull
	BIC	-2.89	-1.62	-7.48	-7.40	-11.39	-10.78	-16.53	-16.02	-6.42	-6.35	-15.01	-14.88	-7.86	-7.33	-7.48	-7.40	-1.41	0.14
Software Acceptance Procedure Generation	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	logistic	normal	lognormal	gamma	weibull	normal	logistic	normal	weibull	normal	beta	gamma
	BIC	-5.27	-5.21	-11.45	-11.31	-13.09	-12.28	-10.95	-10.86	-4.75	-4.53	-16.53	-16.02	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Software Acceptance Test Execution	Dist Name	beta	gamma	weibull	normal	logistic	normal	normal	logistic	beta	weibull	lognormal	gamma	normal	logistic	beta	weibull	beta	weibull
	BIC	-2.55	-2.40	-7.60	-7.32	-11.99	-11.39	-10.46	-10.02	-3.81	-3.34	-15.42	-15.13	-7.86	-7.33	-7.48	-7.40	-1.41	0.14
Traceability Analysis - Test Phase	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	logistic	normal	lognormal	gamma	beta	gamma	logistic	normal	beta	weibull	beta	weibull
	BIC	-1.68	-1.55	-9.52	-9.29	-8.43	-7.97	-6.32	-6.11	-4.41	-4.36	-12.62	-12.55	-9.84	-9.09	-6.17	-6.01	-1.47	0.23
Hazard Analysis - Test Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	logistic	normal	gamma	beta	weibull	beta	logistic	normal	weibull	normal	beta	weibull
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-6.32	-6.11	-6.57	-6.51	-14.12	-14.03	-10.37	-9.63	-6.24	-5.98	-1.15	-0.42
Security Analysis - Test Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	gamma	lognormal	weibull	normal	logistic	normal	weibull	normal	beta	weibull
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-5.66	-5.55	-5.66	-5.64	-13.89	-13.54	-9.84	-9.09	-5.57	-5.33	-1.15	-0.42
Risk Analysis - Test Phase	Dist Name	gamma	lognormal	weibull	beta	logistic	normal	logistic	normal	beta	weibull	normal	logistic	logistic	normal	weibull	beta	normal	logistic
	BIC	-2.42	-2.39	-6.04	-5.81	-11.99	-11.39	4.25	4.74	1.10	1.93	-8.05	-7.55	5.39	6.12	-7.27	-7.25	7.25	7.34
Configuration Management	Dist Name	weibull	beta	logistic	lognormal	weibull	beta	weibull	normal	gamma	logistic	weibull	logistic	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-6.09	-5.95	-32.95	-32.07	-7.53	-7.49	-11.96	-11.61	-4.79	-4.66	-13.94	-13.64	-5.49	-5.15	-10.73	-10.55	-0.80	-0.72
Reviews & Audits	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	beta	gamma	weibull	beta	normal	logistic	weibull	beta	beta	weibull
	BIC	-1.80	-1.46	-10.73	-10.55	-7.30	-6.92	-6.48	-5.84	-0.70	-0.67	-9.84	-9.81	-3.30	-2.71	-6.68	-6.61	1.18	2.65
Installation Procedure Generation	Dist Name	beta	weibull	lognormal	gamma	logistic	normal	lognormal	gamma	beta	gamma	gamma	beta	logistic	normal	lognormal	gamma	beta	weibull
	BIC	-7.59	-7.35	-10.85	-9.83	-21.53	-20.92	-5.04	-4.99	-1.52	-1.23	-10.85	-10.84	-6.37	-5.98	-5.24	-4.91	-0.63	1.41
Installation and Checkout	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	lognormal	gamma	beta	weibull	weibull	beta	logistic	normal	lognormal	gamma	beta	weibull
	BIC	-4.41	-3.50	-6.63	-6.38	-18.90	-18.33	-5.85	-5.59	-2.42	-2.16	-13.20	-13.13	-6.27	-5.66	-6.20	-5.78	-1.27	0.81
Hazard Analysis - Installation and Checkout Phase	Dist Name	gamma	lognormal	weibull	logistic	logistic	normal	lognormal	gamma	lognormal	gamma	weibull	normal	normal	logistic	weibull	beta	beta	gamma
	BIC	-2.78	-2.78	-3.88	-3.81	-21.53	-20.92	-12.86	-12.74	-7.96	-7.78	-8.04	-7.73	-18.90	-18.33	-6.11	-5.93	-4.19	-3.88
Security Analysis - Installation and Checkout Phase	Dist Name	gamma	lognormal	weibull	logistic	logistic	normal	lognormal	gamma	lognormal	gamma	weibull	normal	normal	logistic	weibull	beta	gamma	lognormal
	BIC	-2.78	-2.78	-3.88	-3.81	-21.53	-20.92	-10.76	-10.53	-5.62	-5.56	-8.11	-7.73	-18.90	-18.33	-5.41	-5.21	-3.18	-3.17

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Risk Analysis - Installation and Checkout Phase	Dist Name	beta	gamma	beta	weibull	logistic	normal	lognormal	gamma	beta	logistic	normal	logistic	logistic	normal	lognormal	gamma	logistic	normal
	BIC	0.03	0.71	-2.72	-2.61	-8.16	-7.91	-0.64	1.02	2.95	3.37	-3.29	-2.90	7.13	7.72	-6.63	-6.38	6.92	7.10
Software V&V Planning	Dist Name	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	lognormal	gamma	normal	logistic	logistic	normal	weibull	normal	gamma	lognormal
	BIC	-5.69	-5.66	-8.04	-7.99	-19.44	-19.16	-17.98	-17.42	-8.28	-8.21	-9.56	-9.51	-12.41	-12.18	-9.17	-8.75	-4.59	-4.52
Concept Documentation Evaluation	Dist Name	lognormal	logistic	logistic	weibull	logistic	normal	logistic	normal	lognormal	gamma	weibull	beta	logistic	normal	weibull	beta	beta	weibull
	BIC	-8.82	-8.77	-11.45	-11.31	-28.08	-26.78	-10.25	-10.18	-7.61	-7.28	-18.79	-18.28	-11.16	-11.12	-8.04	-7.99	-3.07	-2.48
Hardware/Software/User Requirements Allocation Analysis	Dist Name	lognormal	gamma	weibull	normal	normal	logistic	logistic	normal	lognormal	gamma	weibull	normal	logistic	normal	weibull	normal	beta	gamma
	BIC	-5.35	-5.23	-7.60	-7.32	-23.96	-23.46	-11.94	-11.72	-5.81	-5.60	-17.98	-17.42	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Software Requirements Evaluation	Dist Name	lognormal	gamma	weibull	normal	normal	logistic	logistic	normal	lognormal	gamma	weibull	beta	normal	logistic	weibull	beta	lognormal	gamma
	BIC	-5.35	-5.23	-7.60	-7.32	-23.96	-23.46	-10.25	-10.18	-7.61	-7.28	-18.79	-18.28	-19.44	-19.16	-8.04	-7.99	-5.69	-5.66
Interface Analysis V&V - Requirement Specifications Phase	Dist Name	lognormal	logistic	logistic	weibull	logistic	normal	normal	logistic	gamma	lognormal	weibull	beta	logistic	normal	weibull	normal	beta	gamma
	BIC	-8.82	-8.77	-11.45	-11.31	-28.08	-26.78	-11.49	-11.15	-4.48	-4.45	-16.47	-16.35	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Traceability Analysis V&V - Requirements Specifications Phase	Dist Name	beta	weibull	weibull	normal	logistic	normal	normal	logistic	lognormal	gamma	beta	gamma	logistic	normal	beta	weibull	beta	weibull
	BIC	-1.15	-0.42	-6.24	-5.98	-10.37	-9.63	-9.82	-9.53	-7.61	-7.28	-11.91	-11.85	-11.39	-10.78	-7.48	-7.40	-2.89	-1.62
Criticality Analysis V&V - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	weibull	normal	lognormal	gamma
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-9.65	-9.13	-4.12	-3.84	-13.25	-12.84	-7.47	-6.68	-7.60	-7.32	-0.45	-0.33
Hazard Analysis V&V - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-5.27	-5.21	-6.74	-6.49	-28.08	-26.78	-10.46	-10.02	-4.75	-4.53	-13.64	-13.25	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Security Analysis V&V - Requirements Specifications Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-10.46	-10.02	-4.75	-4.53	-13.64	-13.25	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Risk Analysis V&V - Requirements Specifications Phase	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	normal	logistic	gamma	beta	beta	weibull	normal	logistic	weibull	normal	beta	gamma
	BIC	-0.24	-0.07	-6.58	-6.35	-4.83	-4.74	-4.61	-4.07	-0.40	-0.39	-11.03	-10.99	-3.78	-3.19	-5.82	-5.58	1.67	2.59
V&V System/Software Qualification Test Plan Generation	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	lognormal	gamma	beta	weibull	lognormal	gamma	normal	logistic	beta	weibull	beta	weibull
	BIC	-4.17	-2.28	-7.06	-6.10	-23.96	-23.46	-10.61	-10.59	-5.47	-5.15	-17.21	-17.18	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
V&VSoftwareAcceptanceTestPlanGeneration	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	weibull	normal	lognormal	gamma	lognormal	gamma	normal	logistic	logistic	weibull	gamma	beta
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-14.89	-14.67	-13.85	-13.61	-19.22	-19.07	-7.56	-7.05	-8.37	-8.23	-0.41	-0.34
Configuration Management Assessment- Requirement Specifications Phase	Dist Name	weibull	beta	logistic	lognormal	lognormal	beta	weibull	normal	logistic	beta	weibull	normal	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-5.56	-5.12	-32.95	-32.07	-8.32	-8.21	-11.09	-10.72	-4.68	-4.64	-11.09	-10.72	-5.25	-4.86	-9.17	-8.96	-0.48	-0.28
Reviews and Audit V&V- Requirement Specifications Phase	Dist Name	gamma	lognormal	logistic	weibull	normal	logistic	normal	logistic	beta	weibull	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
	BIC	-0.24	-0.07	-10.73	-10.55	-4.86	-4.41	-6.83	-6.47	-1.66	-1.45	-14.23	-13.53	-3.78	-3.19	-5.82	-5.58	1.67	2.59
V&V Requirements Specifications Phase Activity Summary Report Generation	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	weibull	normal	beta	weibull
	BIC	-5.27	-5.21	-11.45	-11.31	-13.09	-12.28	-8.54	-8.34	-7.61	-7.28	-19.22	-19.07	-6.77	-6.24	-5.25	-5.00	0.62	1.61
Design Evaluation	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	logistic	normal	lognormal	gamma	weibull	normal	logistic	normal	beta	weibull	beta	weibull
	BIC	-7.01	-5.93	-10.12	-9.48	-23.96	-23.46	-10.25	-10.18	-10.73	-10.52	-20.64	-19.52	-9.84	-9.09	-6.17	-6.01	-1.47	0.23
Interface Analysis V&V	Dist Name	lognormal	logistic	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	weibull
	BIC	-8.82	-8.77	-11.45	-11.31	-28.08	-26.78	-9.82	-9.53	-8.88	-8.86	-18.79	-18.28	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Traceability Analysis V&V- Design Phase	Dist Name	beta	weibull	beta	weibull	logistic	normal	normal	logistic	lognormal	gamma	lognormal	gamma	logistic	normal	beta	weibull	beta	weibull
	BIC	-1.47	0.23	-6.17	-6.01	-9.84	-9.09	-9.82	-9.53	-6.51	-6.41	-15.29	-14.77	-11.39	-10.78	-7.48	-7.40	-2.89	-1.62
Criticality Analysis V&V- Design Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	weibull	logistic	normal	logistic	weibull	normal	beta	lognormal
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-9.65	-9.13	-6.17	-6.01	-16.44	-16.02	-6.79	-6.01	-6.24	-5.98	0.55	0.93
Hazard Analysis V&V- Design Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	weibull	logistic	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-10.46	-10.02	-6.94	-6.85	-16.42	-16.11	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Security Analysis V&V- Design Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	gamma
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-9.07	-8.63	-8.28	-8.21	-14.20	-13.90	-8.57	-8.04	-7.60	-7.32	-1.23	-1.03
Risk Analysis V&V- Design Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	logistic	normal	beta	weibull	lognormal	gamma	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.70	-3.24	-6.74	-6.49	-8.57	-7.81	-2.88	-2.74	0.56	1.46	-9.72	-9.55	0.15	0.21	-6.24	-5.98	2.70	4.29
V&V Software Component Test Plan Generation	Dist Name	beta	gamma	weibull	beta	logistic	normal	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-1.23	-1.03	-4.51	-4.48	-11.99	-11.39	-14.20	-13.90	-8.28	-8.21	-16.47	-16.35	-9.51	-9.00	-11.45	-11.31	-3.95	-3.83
V&V Software Integration Test Plan Generation	Dist Name	beta	gamma	weibull	beta	logistic	normal	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-1.23	-1.03	-4.51	-4.48	-11.99	-11.39	-14.96	-14.88	-10.73	-10.52	-18.79	-18.28	-9.51	-9.00	-11.45	-11.31	-3.95	-3.83
V&V Software Component Test Design Generation	Dist Name	beta	weibull	lognormal	beta	logistic	normal	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	weibull	normal	beta	weibull
	BIC	-1.41	0.14	-4.73	-4.68	-11.39	-10.78	-7.98	-7.46	-8.88	-8.86	-21.88	-21.26	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
V&VSoftwareIntegrationTestDesignGeneration	Dist Name	beta	weibull	lognormal	gamma	logistic	normal	normal	logistic	beta	weibull	weibull	beta	normal	logistic	beta	weibull	beta	weibull
	BIC	-2.39	0.12	-7.06	-6.10	-11.20	-10.37	-5.33	-4.97	-5.70	-5.39	-19.58	-19.50	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
V&VSoftwareQualificationTestDesignGeneration	Dist Name	beta	weibull	lognormal	beta	logistic	normal	normal	logistic	gamma	beta	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-1.41	0.14	-4.73	-4.68	-11.39	-10.78	-4.14	-3.73	-4.73	-4.72	-23.66	-22.54	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
V&V Software Acceptance Test Design Generation	Dist Name	beta	gamma	weibull	normal	logistic	normal	normal	logistic	gamma	beta	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-2.55	-2.40	-7.60	-7.32	-11.99	-11.39	-5.90	-5.71	-6.57	-6.51	-20.63	-19.88	-7.58	-7.09	-6.24	-5.98	-0.18	0.61

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Configuration Management V&V- Design Phase	Dist Name	weibull	beta	logistic	lognormal	lognormal	beta	normal	logistic	logistic	gamma	beta	weibull	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-5.56	-5.12	-32.95	-32.07	-8.32	-8.21	-6.70	-6.41	-2.59	-2.44	-10.36	-10.25	-5.25	-4.86	-9.17	-8.96	-0.48	-0.28
Review and Audit- Design Phase	Dist Name	beta	weibull	weibull	normal	normal	logistic	normal	logistic	beta	weibull	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	0.54	1.57	-7.01	-6.75	-4.23	-4.09	-4.91	-4.48	-0.77	-0.48	-14.23	-13.74	-3.78	-3.19	-5.82	-5.58	1.67	2.59
V&V Design Phase Activity Summary Report Generation	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	-3.95	-3.83	-11.45	-11.31	-9.51	-9.00	-8.19	-7.97	-5.81	-5.60	-17.12	-16.84	-9.20	-8.69	-8.31	-7.96	-2.16	-2.14
Source Code and Source Code Documentation Evaluation	Dist Name	beta	gamma	weibull	beta	normal	logistic	logistic	normal	gamma	lognormal	weibull	normal	logistic	normal	weibull	normal	beta	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-6.51	-6.48	-7.57	-7.52	-15.80	-15.61	-11.99	-11.39	-7.60	-7.32	-2.55	-2.40
Interface Analysis V&V - Implementation Phase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	logistic	weibull	gamma	lognormal
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-6.11	-6.00	-5.04	-4.81	-17.85	-17.17	-8.43	-7.97	-9.52	-9.29	-1.68	-1.55
TraceabilityAnalysisV&V-ImplementationPhase	Dist Name	beta	weibull	weibull	normal	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	beta	weibull	beta	weibull
	BIC	-1.15	-0.42	-6.24	-5.98	-10.37	-9.63	-8.47	-7.93	-6.94	-6.85	-13.64	-13.25	-7.86	-7.33	-7.48	-7.40	-1.41	0.14
CriticalityAnalysisV&V-ImplementationPhase	Dist Name	lognormal	gamma	logistic	weibull	normal	logistic	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-3.54	-3.27	-6.74	-6.49	-9.51	-9.00	-8.47	-7.93	-6.94	-6.85	-13.64	-13.25	-7.82	-7.08	-9.52	-9.29	-1.76	-1.67
HazardAnalysisV&V-ImplementationPhase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-9.07	-8.63	-8.28	-8.21	-14.20	-13.90	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Security Analysis V&V - Implementation Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	beta	normal	logistic	logistic	weibull	gamma	lognormal
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-9.07	-8.63	-8.28	-8.21	-14.20	-13.90	-8.43	-7.97	-9.52	-9.29	-1.68	-1.55
RiskAnalysisV&V-ImplementationPhase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	beta	weibull	lognormal	gamma	logistic	normal	logistic	weibull	beta	normal
	BIC	-4.10	-3.91	-6.74	-6.49	-12.41	-12.18	0.61	0.79	0.01	0.34	-13.92	-13.72	1.54	1.83	-9.52	-9.29	3.20	4.27
V&V Software Component Test Case Generation	Dist Name	beta	weibull	lognormal	beta	normal	logistic	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	weibull
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-14.20	-13.90	-8.28	-8.21	-16.47	-16.35	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
V&VSoftwareIntegrationTestCaseGeneration	Dist Name	beta	weibull	lognormal	beta	normal	logistic	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	weibull	normal	beta	weibull
	BIC	-2.89	-1.62	-4.73	-4.68	-23.25	-22.41	-14.20	-13.90	-8.28	-8.21	-16.47	-16.35	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
V&VSoftwareQualificationTestCaseGeneration	Dist Name	beta	gamma	weibull	beta	normal	logistic	weibull	beta	lognormal	gamma	weibull	beta	normal	logistic	logistic	weibull	gamma	beta
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-14.20	-13.90	-8.28	-8.21	-16.47	-16.35	-7.56	-7.05	-8.37	-8.23	-0.41	-0.34
V&VSoftwareAcceptanceTestCaseGeneration	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-5.76	-5.51	-3.72	-3.60	-17.21	-17.18	-4.26	-3.63	-8.27	-8.08	0.63	0.71
V&V Software Component Test Procedure Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	logistic	weibull	lognormal	gamma

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-5.76	-5.51	-3.72	-3.60	-17.21	-17.18	-8.57	-7.81	-11.45	-11.31	-3.54	-3.27
V&V Software Integration Test Procedure Generation	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-5.76	-5.51	-3.72	-3.60	-17.21	-17.18	-8.57	-7.81	-11.45	-11.31	-3.54	-3.27
V&VSoftwareQualificationTestProcedureGeneration	Dist Name	beta	gamma	weibull	beta	normal	logistic	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-2.55	-2.40	-4.51	-4.48	-23.96	-23.46	-5.76	-5.51	-3.72	-3.60	-17.21	-17.18	-8.57	-7.81	-11.45	-11.31	-3.54	-3.27
V&V Software Component Test Execution	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	normal	logistic	lognormal	gamma	normal	logistic	normal	logistic	beta	weibull	beta	weibull
	BIC	-7.01	-5.93	-10.12	-9.48	-23.96	-23.46	-9.07	-8.63	-2.62	-2.49	-11.72	-11.27	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
Configuration Management V&V – Implementation Phase	Dist Name	weibull	beta	logistic	lognormal	lognormal	beta	normal	logistic	gamma	lognormal	weibull	beta	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-5.56	-5.12	-32.95	-32.07	-8.32	-8.21	-7.28	-6.86	-2.25	-2.24	-8.05	-7.92	-5.25	-4.86	-9.17	-8.96	-0.48	-0.28
Review and Audit V&V – Implementation Phase	Dist Name	beta	weibull	weibull	normal	normal	logistic	normal	logistic	beta	weibull	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	0.54	1.57	-7.01	-6.75	-4.23	-4.09	-4.91	-4.48	-0.77	-0.48	-14.23	-13.74	-3.78	-3.19	-5.82	-5.58	1.67	2.59
V&V Implementation Phase Activity Summary Report Generation	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	normal	normal	logistic	logistic	weibull	gamma	beta
	BIC	-5.27	-5.21	-11.45	-11.31	-13.09	-12.28	-4.14	-3.73	-2.93	-2.86	-21.14	-20.66	-7.56	-7.05	-8.37	-8.23	-0.41	-0.34
V&V Software Integration Test Execution	Dist Name	beta	weibull	lognormal	gamma	logistic	normal	normal	logistic	lognormal	gamma	normal	logistic	normal	logistic	beta	weibull	beta	weibull
	BIC	-6.01	-4.68	-12.33	-11.64	-12.35	-11.62	-5.95	-5.81	-0.72	-0.69	-11.95	-11.66	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
V&V Software Qualification Test Execution	Dist Name	beta	weibull	beta	lognormal	logistic	normal	normal	logistic	lognormal	gamma	normal	logistic	normal	logistic	weibull	normal	beta	weibull
	BIC	-4.35	-3.36	-8.80	-8.77	-12.53	-12.03	-5.95	-5.81	-0.72	-0.69	-11.95	-11.66	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
V&V Software Acceptance Procedure Generation	Dist Name	lognormal	logistic	weibull	logistic	logistic	normal	logistic	normal	lognormal	gamma	weibull	normal	normal	logistic	logistic	weibull	gamma	lognormal
	BIC	-8.10	-7.85	-10.87	-10.49	-14.44	-13.73	-6.51	-6.48	-3.72	-3.60	-17.81	-17.42	-8.43	-7.97	-9.52	-9.29	-1.68	-1.55
V&VSoftware Acceptance Test Execution	Dist Name	beta	weibull	lognormal	gamma	logistic	normal	normal	logistic	lognormal	gamma	normal	logistic	normal	logistic	beta	weibull	beta	weibull
	BIC	-6.01	-4.68	-12.33	-11.64	-12.35	-11.62	-5.95	-5.81	-0.72	-0.69	-11.95	-11.66	-6.93	-6.42	-6.17	-6.01	-0.37	1.53
Traceability Analysis V&V- Test Phase	Dist Name	logistic	normal	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	-2.49	-2.21	-9.52	-9.29	-11.28	-10.37	-9.82	-9.53	-7.61	-7.28	-14.36	-14.33	-8.57	-8.04	-7.60	-7.32	-1.23	-1.03
Hazard Analysis V&V- Test Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	weibull	logistic	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-10.46	-10.02	-6.94	-6.85	-16.42	-16.11	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
Security Analysis V&V- Test Phase	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	gamma	lognormal	weibull	normal	normal	logistic	weibull	normal	beta	weibull
	BIC	-3.95	-3.83	-6.74	-6.49	-13.09	-12.28	-5.95	-5.81	-5.66	-5.64	-15.80	-15.61	-7.58	-7.09	-6.24	-5.98	-0.18	0.61
RiskAnalysisV&V-TestPhase	Dist Name	gamma	lognormal	weibull	beta	logistic	normal	logistic	normal	beta	weibull	normal	logistic	logistic	normal	weibull	beta	normal	logistic
	BIC	-2.42	-2.39	-6.04	-5.81	-11.99	-11.39	3.97	4.36	1.31	2.04	-9.23	-8.82	5.05	5.63	-6.71	-6.63	7.02	7.24

Attribute		High Development Quality						Medium Development Quality						Low Development Quality					
		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	Rank	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Configuration Management V&V - Test Phase	Dist Name	beta	weibull	logistic	weibull	normal	logistic	normal	logistic	gamma	lognormal	weibull	normal	logistic	normal	logistic	weibull	lognormal	gamma
	BIC	-0.36	-0.27	-10.73	-10.55	-4.83	-4.74	-7.24	-7.02	-2.51	-2.43	-11.09	-10.72	-7.15	-6.77	-9.17	-8.96	-1.62	-1.16
Review and Audit V&V - Test Phase	Dist Name	beta	weibull	weibull	normal	normal	logistic	normal	logistic	gamma	beta	lognormal	gamma	normal	logistic	weibull	normal	beta	gamma
	BIC	1.14	2.04	-7.01	-6.75	-4.10	-3.59	-4.90	-4.52	-0.40	-0.39	-12.96	-12.62	-4.20	-3.68	-5.82	-5.58	1.77	2.40
V&V Test Phase Activity Summary Report Generation	Dist Name	lognormal	gamma	logistic	weibull	logistic	normal	normal	logistic	lognormal	gamma	beta	gamma	normal	logistic	logistic	weibull	lognormal	gamma
	BIC	-5.27	-5.21	-11.45	-11.31	-13.09	-12.28	-4.11	-3.67	-2.73	-2.53	-18.07	-18.02	-7.19	-6.43	-8.37	-8.23	-0.78	-0.69
Installation Configuration Audit V&V	Dist Name	lognormal	gamma	weibull	logistic	logistic	normal	lognormal	gamma	gamma	lognormal	beta	gamma	logistic	normal	weibull	beta	beta	weibull
	BIC	-5.85	-5.63	-7.71	-7.67	-21.53	-20.92	-5.85	-5.59	-4.05	-3.95	-9.38	-9.35	-6.78	-6.45	-4.11	-4.06	0.04	1.05
Installation Checkout V&V	Dist Name	beta	weibull	lognormal	gamma	normal	logistic	lognormal	gamma	beta	weibull	weibull	beta	logistic	normal	lognormal	gamma	beta	weibull
	BIC	-4.41	-3.50	-6.63	-6.38	-18.90	-18.33	-5.85	-5.59	-2.42	-2.16	-13.20	-13.13	-6.37	-5.98	-5.24	-4.91	-0.63	1.41
Hazard Analysis V&V - Installation and Checkout Phase	Dist Name	gamma	lognormal	weibull	logistic	logistic	normal	weibull	beta	lognormal	gamma	weibull	normal	logistic	normal	weibull	beta	beta	weibull
	BIC	-2.78	-2.78	-3.88	-3.81	-21.53	-20.92	-10.32	-10.14	-5.62	-5.56	-9.24	-8.93	-8.16	-7.91	-5.41	-5.21	-1.21	-0.52
Security Analysis V&V - Installation and Checkout Phase	Dist Name	gamma	lognormal	weibull	logistic	logistic	normal	weibull	beta	lognormal	gamma	weibull	normal	logistic	normal	weibull	beta	beta	weibull
	BIC	-2.78	-2.78	-3.88	-3.81	-21.53	-20.92	-10.32	-10.14	-5.62	-5.56	-9.24	-8.93	-8.16	-7.91	-5.41	-5.21	-1.21	-0.52
Risk Analysis V&V-Installation and Checkout Phase	Dist Name	beta	weibull	beta	weibull	normal	logistic	lognormal	gamma	beta	logistic	normal	logistic	logistic	normal	lognormal	gamma	normal	logistic
	BIC	-1.21	-0.52	-2.72	-2.61	-18.90	-18.33	0.63	1.74	2.91	3.47	-4.09	-3.75	7.09	7.49	-6.63	-6.38	7.18	7.25
V&V Installation and Checkout Phase Activity Summary Report Generation	Dist Name	lognormal	gamma	weibull	logistic	logistic	normal	gamma	lognormal	gamma	lognormal	weibull	normal	logistic	normal	logistic	weibull	beta	weibull
	BIC	-5.85	-5.63	-7.71	-7.67	-21.53	-20.92	-6.14	-6.14	-4.05	-3.95	-14.81	-13.99	-7.53	-7.14	-6.07	-6.06	-0.46	-0.37
V&V Final Report Generation	Dist Name	lognormal	gamma	weibull	logistic	logistic	normal	gamma	lognormal	gamma	lognormal	weibull	normal	logistic	normal	logistic	weibull	beta	weibull
	BIC	-5.85	-5.63	-7.71	-7.67	-21.53	-20.92	-6.14	-6.14	-4.05	-3.95	-14.81	-13.99	-7.53	-7.14	-6.07	-6.06	-0.46	-0.37

Table D-8 Normal Distribution Fit for the NPT of the Attribute Nodes

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Software Development Planning	0.625	0.117	0.325	0.147	0.050	0.032	0.193	0.073	0.643	0.113	0.164	0.103	0.063	0.068	0.303	0.190	0.633	0.178
Development of a Concept Documentation	0.675	0.129	0.267	0.082	0.058	0.074	0.221	0.091	0.643	0.113	0.136	0.063	0.050	0.032	0.292	0.146	0.658	0.153
Development of Software Requirements Specifications	0.717	0.098	0.242	0.092	0.042	0.038	0.186	0.090	0.657	0.098	0.157	0.079	0.042	0.038	0.275	0.141	0.683	0.133
Traceability Analysis - Requirements Specifications Phase	0.567	0.175	0.295	0.105	0.138	0.108	0.143	0.113	0.657	0.098	0.200	0.100	0.068	0.089	0.223	0.116	0.708	0.150
Criticality Analysis - Requirements Specifications Phase	0.642	0.215	0.270	0.143	0.088	0.101	0.179	0.141	0.636	0.138	0.186	0.111	0.088	0.101	0.295	0.105	0.617	0.169
Hazard Analysis - Requirements Specifications Phase	0.606	0.188	0.297	0.113	0.097	0.120	0.176	0.118	0.619	0.157	0.205	0.089	0.081	0.126	0.289	0.100	0.631	0.181
Security Analysis - Requirements Specifications Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.186	0.135	0.636	0.125	0.179	0.070	0.063	0.085	0.312	0.130	0.625	0.170
Risk Analysis - Requirements Specifications Phase	0.536	0.215	0.303	0.114	0.161	0.149	0.190	0.151	0.612	0.193	0.198	0.108	0.119	0.135	0.276	0.122	0.606	0.213
System/Software Qualification Test Plan Generation	0.717	0.157	0.258	0.136	0.025	0.027	0.183	0.107	0.648	0.180	0.169	0.125	0.058	0.074	0.233	0.103	0.708	0.159
System/Software Acceptance Test Plan Generation	0.667	0.129	0.300	0.114	0.033	0.026	0.207	0.084	0.657	0.113	0.136	0.085	0.063	0.085	0.287	0.096	0.650	0.158
Configuration Management - Requirements Specifications Phase	0.589	0.149	0.297	0.027	0.114	0.127	0.212	0.086	0.583	0.159	0.205	0.117	0.139	0.127	0.306	0.014	0.556	0.137
Review and Audit - Requirements Specifications Phase	0.589	0.193	0.272	0.085	0.139	0.127	0.226	0.139	0.605	0.198	0.169	0.099	0.132	0.137	0.296	0.122	0.572	0.201
Development of a Software Architecture Description	0.725	0.157	0.208	0.102	0.067	0.075	0.221	0.064	0.586	0.107	0.193	0.067	0.072	0.094	0.228	0.108	0.700	0.164
Development of Software Design Description	0.792	0.124	0.192	0.102	0.017	0.026	0.200	0.076	0.629	0.095	0.171	0.091	0.072	0.111	0.203	0.119	0.725	0.223
Traceability Analysis - Design Phase	0.617	0.186	0.295	0.118	0.088	0.101	0.200	0.153	0.629	0.147	0.171	0.070	0.080	0.107	0.212	0.125	0.708	0.225
Criticality Analysis - Design Phase	0.658	0.196	0.258	0.136	0.083	0.093	0.164	0.131	0.643	0.143	0.193	0.098	0.080	0.107	0.237	0.113	0.683	0.207
Hazard Analysis - Design Phase	0.650	0.148	0.317	0.129	0.033	0.026	0.150	0.104	0.643	0.113	0.207	0.084	0.055	0.089	0.237	0.113	0.708	0.188
Security Analysis - Design Phase	0.625	0.170	0.303	0.116	0.072	0.080	0.143	0.127	0.650	0.126	0.207	0.073	0.055	0.089	0.295	0.187	0.650	0.224

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Risk Analysis - Design Phase	0.617	0.186	0.295	0.118	0.088	0.101	0.179	0.163	0.614	0.195	0.207	0.084	0.130	0.202	0.262	0.131	0.608	0.276
Software Component Test Plan Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.176	0.115	0.612	0.179	0.212	0.086	0.083	0.093	0.267	0.082	0.650	0.148
Software Integration Test Plan Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.164	0.099	0.629	0.138	0.207	0.079	0.058	0.074	0.258	0.080	0.683	0.133
Software Component Test Design Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.136	0.085	0.700	0.141	0.164	0.085	0.050	0.077	0.200	0.110	0.750	0.173
Software Integration Test Design Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.136	0.085	0.686	0.146	0.179	0.081	0.050	0.077	0.225	0.099	0.725	0.157
Software Qualification Test Design Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.200	0.126	0.621	0.168	0.179	0.064	0.083	0.093	0.233	0.103	0.683	0.181
Software Acceptance Test Design Generation	0.683	0.133	0.283	0.113	0.033	0.026	0.193	0.124	0.650	0.161	0.157	0.061	0.083	0.093	0.258	0.080	0.658	0.153
Configuration Management - Design Phase	0.614	0.143	0.297	0.027	0.089	0.121	0.183	0.090	0.605	0.164	0.212	0.091	0.139	0.127	0.272	0.085	0.589	0.193
Reviews and Audit - Design Phase	0.631	0.232	0.247	0.116	0.122	0.142	0.226	0.139	0.598	0.197	0.176	0.090	0.124	0.144	0.229	0.114	0.647	0.245
Source Code and Source Code Documentation Generation	0.667	0.160	0.275	0.117	0.058	0.074	0.150	0.087	0.650	0.104	0.200	0.096	0.055	0.089	0.237	0.113	0.708	0.188
Traceability Analysis - Implementation Phase	0.600	0.176	0.303	0.116	0.097	0.094	0.200	0.153	0.600	0.126	0.200	0.076	0.080	0.107	0.237	0.113	0.683	0.207
Criticality Analysis - Implementation Phase	0.658	0.196	0.258	0.136	0.083	0.093	0.164	0.131	0.621	0.135	0.214	0.075	0.088	0.101	0.245	0.116	0.667	0.204
Hazard Analysis - Implementation Phase	0.658	0.153	0.283	0.113	0.058	0.074	0.157	0.127	0.636	0.125	0.207	0.073	0.055	0.089	0.228	0.113	0.717	0.191
Security Analysis - Implementation Phase	0.642	0.146	0.300	0.114	0.058	0.074	0.143	0.098	0.657	0.098	0.200	0.082	0.055	0.089	0.212	0.125	0.733	0.204
Risk Analysis - Implementation Phase	0.617	0.160	0.292	0.111	0.092	0.092	0.221	0.208	0.586	0.210	0.193	0.117	0.147	0.239	0.245	0.116	0.608	0.311
Component Test Case Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.164	0.085	0.643	0.113	0.193	0.045	0.083	0.093	0.233	0.103	0.683	0.181
Software Integration Test Case Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.164	0.085	0.643	0.113	0.193	0.045	0.058	0.074	0.233	0.103	0.708	0.159
Software Qualification Test Case Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.171	0.091	0.629	0.125	0.200	0.050	0.058	0.074	0.233	0.103	0.708	0.159
Software Acceptance Test Case Generation	0.692	0.180	0.258	0.136	0.050	0.077	0.207	0.127	0.593	0.148	0.200	0.058	0.097	0.139	0.231	0.105	0.672	0.226
Software Component Test Procedure Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.171	0.076	0.657	0.098	0.171	0.076	0.083	0.093	0.233	0.103	0.683	0.181
Software Integration Test Procedure Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.186	0.069	0.657	0.098	0.157	0.053	0.092	0.086	0.267	0.082	0.642	0.146
Software Qualification Test Procedure Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.200	0.076	0.629	0.125	0.171	0.070	0.108	0.102	0.233	0.103	0.658	0.196
Configuration Management	0.564	0.149	0.297	0.027	0.139	0.127	0.212	0.086	0.548	0.143	0.240	0.075	0.164	0.120	0.272	0.085	0.564	0.193

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Reviews & Audits	0.589	0.193	0.272	0.085	0.139	0.127	0.183	0.121	0.619	0.186	0.198	0.100	0.141	0.129	0.271	0.097	0.589	0.203
Software Component Test Execution	0.817	0.129	0.167	0.103	0.017	0.026	0.171	0.076	0.700	0.100	0.129	0.049	0.072	0.111	0.203	0.119	0.725	0.223
Software Integration Test Execution	0.792	0.174	0.167	0.103	0.042	0.080	0.193	0.110	0.650	0.150	0.157	0.061	0.080	0.107	0.212	0.125	0.708	0.225
Software Qualification Test Execution	0.750	0.173	0.200	0.110	0.050	0.077	0.186	0.063	0.643	0.127	0.171	0.070	0.075	0.099	0.200	0.110	0.725	0.199
Software Acceptance Procedure Generation	0.667	0.129	0.267	0.082	0.067	0.068	0.171	0.091	0.643	0.151	0.186	0.063	0.058	0.074	0.233	0.103	0.708	0.159
Software Acceptance Test Execution	0.708	0.159	0.233	0.103	0.058	0.074	0.157	0.093	0.686	0.157	0.157	0.073	0.075	0.099	0.200	0.110	0.725	0.199
Traceability Analysis - Test Phase	0.625	0.170	0.278	0.092	0.097	0.094	0.157	0.127	0.650	0.150	0.193	0.084	0.055	0.089	0.212	0.125	0.733	0.204
Hazard Analysis - Test Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.157	0.127	0.636	0.125	0.207	0.073	0.063	0.085	0.245	0.116	0.692	0.188
Security Analysis - Test Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.164	0.131	0.621	0.135	0.214	0.075	0.055	0.089	0.253	0.122	0.692	0.188
Risk Analysis - Test Phase	0.667	0.160	0.275	0.117	0.058	0.074	0.243	0.276	0.593	0.228	0.164	0.111	0.180	0.317	0.220	0.109	0.600	0.348
Configuration Management	0.522	0.118	0.306	0.014	0.172	0.108	0.212	0.086	0.548	0.143	0.240	0.075	0.164	0.120	0.272	0.085	0.564	0.193
Reviews & Audits	0.539	0.189	0.272	0.085	0.189	0.107	0.190	0.124	0.605	0.192	0.205	0.102	0.124	0.144	0.229	0.114	0.647	0.245
Installation Procedure Generation	0.850	0.112	0.140	0.089	0.010	0.022	0.210	0.124	0.630	0.164	0.160	0.065	0.056	0.100	0.194	0.131	0.750	0.224
Installation and Checkout	0.800	0.137	0.180	0.110	0.020	0.027	0.200	0.122	0.650	0.150	0.150	0.050	0.046	0.103	0.184	0.123	0.770	0.217
Hazard Analysis - Installation and Checkout Phase	0.670	0.144	0.290	0.124	0.040	0.022	0.140	0.055	0.640	0.089	0.220	0.084	0.020	0.027	0.210	0.102	0.770	0.125
Security Analysis - Installation and Checkout Phase	0.670	0.144	0.290	0.124	0.040	0.022	0.150	0.071	0.620	0.110	0.230	0.084	0.030	0.027	0.220	0.110	0.750	0.137
Risk Analysis - Installation and Checkout Phase	0.690	0.201	0.250	0.150	0.060	0.082	0.280	0.349	0.540	0.261	0.180	0.130	0.200	0.392	0.180	0.110	0.620	0.368
Software V&V Planning	0.750	0.122	0.208	0.102	0.042	0.038	0.179	0.057	0.643	0.113	0.179	0.099	0.075	0.069	0.242	0.092	0.683	0.133
Concept Documentation Evaluation	0.692	0.102	0.267	0.082	0.042	0.020	0.171	0.095	0.671	0.125	0.157	0.053	0.067	0.075	0.208	0.102	0.725	0.157
Hardware/Software/User Requirements Allocation Analysis	0.733	0.129	0.233	0.103	0.033	0.026	0.164	0.085	0.657	0.140	0.179	0.057	0.058	0.074	0.233	0.103	0.708	0.159
Software Requirements Evaluation	0.733	0.129	0.233	0.103	0.033	0.026	0.171	0.095	0.671	0.125	0.157	0.053	0.042	0.038	0.208	0.102	0.750	0.122
Interface Analysis V&V - Requirement Specifications Phase	0.692	0.102	0.267	0.082	0.042	0.020	0.150	0.087	0.686	0.146	0.164	0.063	0.058	0.074	0.233	0.103	0.708	0.159
Traceability Analysis V&V - Requirements Specifications Phase	0.692	0.188	0.245	0.116	0.063	0.085	0.143	0.098	0.671	0.125	0.186	0.090	0.050	0.077	0.200	0.110	0.750	0.173

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Criticality Analysis V&V - Requirements Specifications Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.164	0.099	0.629	0.160	0.207	0.079	0.108	0.102	0.233	0.103	0.658	0.196
Hazard Analysis V&V - Requirements Specifications Phase	0.667	0.129	0.292	0.111	0.042	0.020	0.157	0.093	0.643	0.151	0.200	0.076	0.088	0.101	0.245	0.116	0.667	0.204
Security Analysis V&V - Requirements Specifications Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.157	0.093	0.643	0.151	0.200	0.076	0.088	0.101	0.245	0.116	0.667	0.204
Risk Analysis V&V - Requirements Specifications Phase	0.589	0.193	0.297	0.113	0.114	0.127	0.198	0.142	0.598	0.197	0.205	0.094	0.136	0.139	0.251	0.119	0.614	0.246
V&V System/Software Qualification Test Plan Generation	0.792	0.174	0.192	0.150	0.017	0.026	0.207	0.102	0.636	0.138	0.157	0.061	0.080	0.107	0.212	0.125	0.708	0.225
V&V Software Acceptance Test Plan Generation	0.642	0.146	0.292	0.111	0.067	0.068	0.214	0.069	0.643	0.079	0.143	0.053	0.102	0.101	0.290	0.099	0.608	0.188
Configuration Management Assessment- Requirement Specifications Phase	0.547	0.128	0.306	0.014	0.147	0.117	0.219	0.092	0.562	0.142	0.219	0.092	0.169	0.123	0.284	0.094	0.547	0.203
Reviews and Audit V&V- Requirement Specifications Phase	0.589	0.193	0.272	0.085	0.139	0.127	0.205	0.121	0.633	0.180	0.162	0.089	0.136	0.139	0.251	0.119	0.614	0.246
V&V Requirements Specifications Phase Activity Summary Report Generation	0.667	0.129	0.267	0.082	0.067	0.068	0.186	0.107	0.671	0.125	0.143	0.053	0.093	0.108	0.257	0.125	0.650	0.224
Design Evaluation	0.817	0.129	0.167	0.103	0.017	0.026	0.171	0.095	0.657	0.098	0.171	0.049	0.055	0.089	0.212	0.125	0.733	0.204
Interface Analysis V&V	0.692	0.102	0.267	0.082	0.042	0.020	0.157	0.098	0.686	0.107	0.157	0.053	0.088	0.101	0.245	0.116	0.667	0.204
Traceability Analysis V&V- Design Phase	0.733	0.204	0.212	0.125	0.055	0.089	0.143	0.098	0.700	0.129	0.157	0.079	0.050	0.077	0.200	0.110	0.750	0.173
Criticality Analysis V&V- Design Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.164	0.099	0.614	0.135	0.221	0.064	0.113	0.108	0.245	0.116	0.642	0.215
Hazard Analysis V&V- Design Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.157	0.093	0.629	0.125	0.214	0.063	0.088	0.101	0.245	0.116	0.667	0.204
Security Analysis V&V- Design Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.164	0.103	0.643	0.113	0.193	0.073	0.083	0.093	0.233	0.103	0.683	0.181
Risk Analysis V&V- Design Phase	0.592	0.163	0.292	0.111	0.117	0.093	0.193	0.162	0.607	0.224	0.200	0.112	0.163	0.193	0.245	0.116	0.592	0.280
V&V Software Component Test Plan Generation	0.683	0.181	0.258	0.136	0.058	0.074	0.193	0.073	0.643	0.113	0.164	0.063	0.092	0.086	0.267	0.082	0.642	0.146
V&V Software Integration Test Plan Generation	0.683	0.181	0.258	0.136	0.058	0.074	0.186	0.069	0.657	0.098	0.157	0.053	0.092	0.086	0.267	0.082	0.642	0.146
V&V Software Component Test Design Generation	0.725	0.199	0.225	0.147	0.050	0.077	0.171	0.111	0.686	0.107	0.129	0.049	0.088	0.101	0.245	0.116	0.667	0.204
V&V Software Integration Test Design Generation	0.767	0.207	0.192	0.150	0.042	0.080	0.186	0.135	0.664	0.138	0.150	0.050	0.080	0.107	0.212	0.125	0.708	0.225

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
V&V Software Qualification Test Design Generation	0.725	0.199	0.225	0.147	0.050	0.077	0.214	0.146	0.614	0.144	0.171	0.039	0.088	0.101	0.245	0.116	0.667	0.204
V&V Software Acceptance Test Design Generation	0.708	0.159	0.233	0.103	0.058	0.074	0.200	0.129	0.636	0.125	0.164	0.048	0.088	0.101	0.245	0.116	0.667	0.204
Configuration Management V&V- Design Phase	0.547	0.128	0.306	0.014	0.147	0.117	0.212	0.122	0.590	0.169	0.198	0.100	0.169	0.123	0.284	0.094	0.547	0.203
Review and Audit- Design Phase	0.656	0.223	0.239	0.108	0.106	0.134	0.219	0.139	0.612	0.193	0.169	0.085	0.136	0.139	0.251	0.119	0.614	0.246
V&V Design Phase Activity Summary Report Generation	0.642	0.146	0.267	0.082	0.092	0.086	0.193	0.110	0.657	0.140	0.150	0.065	0.080	0.088	0.220	0.098	0.700	0.164
Source Code and Source Code Documentation Evaluation	0.708	0.159	0.258	0.136	0.033	0.026	0.193	0.124	0.607	0.117	0.200	0.065	0.058	0.074	0.233	0.103	0.708	0.159
Interface Analysis V&V - Implementation Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.207	0.127	0.593	0.148	0.200	0.058	0.097	0.094	0.278	0.092	0.625	0.170
Traceability Analysis V&V - Implementation Phase	0.692	0.188	0.245	0.116	0.063	0.085	0.171	0.107	0.629	0.125	0.200	0.076	0.075	0.099	0.200	0.110	0.725	0.199
Criticality Analysis V&V - Implementation Phase	0.617	0.157	0.292	0.111	0.092	0.086	0.171	0.107	0.629	0.125	0.200	0.076	0.122	0.099	0.278	0.092	0.600	0.176
Hazard Analysis V&V - Implementation Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.164	0.103	0.643	0.113	0.193	0.073	0.088	0.101	0.245	0.116	0.667	0.204
Security Analysis V&V - Implementation Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.164	0.103	0.643	0.113	0.193	0.073	0.097	0.094	0.278	0.092	0.625	0.170
Risk Analysis V&V - Implementation Phase	0.633	0.147	0.292	0.111	0.075	0.069	0.236	0.206	0.593	0.205	0.171	0.081	0.188	0.221	0.278	0.092	0.533	0.271
V&V Software Component Test Case Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.193	0.073	0.643	0.113	0.164	0.063	0.088	0.101	0.245	0.116	0.667	0.204
V&V Software Integration Test Case Generation	0.750	0.173	0.225	0.147	0.025	0.027	0.193	0.073	0.643	0.113	0.164	0.063	0.088	0.101	0.245	0.116	0.667	0.204
V&V Software Qualification Test Case Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.193	0.073	0.643	0.113	0.164	0.063	0.102	0.101	0.290	0.099	0.608	0.188
V&V Software Acceptance Test Case Generation	0.642	0.146	0.292	0.111	0.067	0.068	0.207	0.130	0.636	0.160	0.157	0.061	0.149	0.133	0.296	0.101	0.556	0.216
V&V Software Component Test Procedure Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.207	0.130	0.636	0.160	0.157	0.061	0.117	0.093	0.267	0.082	0.617	0.157
V&V Software Integration Test Procedure Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.207	0.130	0.636	0.160	0.157	0.061	0.117	0.093	0.267	0.082	0.617	0.157
V&V Software Qualification Test Procedure Generation	0.708	0.159	0.258	0.136	0.033	0.026	0.207	0.130	0.636	0.160	0.157	0.061	0.117	0.093	0.267	0.082	0.617	0.157
V&V Software Component Test Execution	0.817	0.129	0.167	0.103	0.017	0.026	0.164	0.103	0.700	0.173	0.136	0.085	0.080	0.107	0.212	0.125	0.708	0.225
Configuration Management V&V – Implementation Phase	0.547	0.128	0.306	0.014	0.147	0.117	0.190	0.117	0.569	0.176	0.240	0.115	0.169	0.123	0.284	0.094	0.547	0.203
Review and Audit V&V – Implementation Phase	0.656	0.223	0.239	0.108	0.106	0.134	0.219	0.139	0.612	0.193	0.169	0.085	0.136	0.139	0.251	0.119	0.614	0.246

Attribute	High Development Quality						Medium Development Quality						Low Development Quality					
	High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality		High Attribute Quality		Medium Attribute Quality		Low Attribute Quality	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
V&V Implementation Phase Activity Summary Report Generation	0.667	0.129	0.267	0.082	0.067	0.068	0.214	0.146	0.629	0.168	0.157	0.045	0.102	0.101	0.290	0.099	0.608	0.188
V&V Software Integration Test Execution	0.808	0.143	0.153	0.086	0.038	0.072	0.179	0.129	0.679	0.195	0.143	0.084	0.080	0.107	0.212	0.125	0.708	0.225
V&V Software Qualification Test Execution	0.767	0.147	0.187	0.099	0.047	0.070	0.179	0.129	0.679	0.195	0.143	0.084	0.088	0.101	0.245	0.116	0.667	0.204
V&V Software Acceptance Procedure Generation	0.683	0.108	0.253	0.082	0.063	0.061	0.193	0.124	0.636	0.160	0.171	0.057	0.097	0.094	0.278	0.092	0.625	0.170
V&V Software Acceptance Test Execution	0.808	0.143	0.153	0.086	0.038	0.072	0.179	0.129	0.679	0.195	0.143	0.084	0.080	0.107	0.212	0.125	0.708	0.225
Traceability Analysis V&V-Test Phase	0.650	0.158	0.278	0.092	0.072	0.080	0.157	0.098	0.671	0.125	0.171	0.076	0.083	0.093	0.233	0.103	0.683	0.181
Hazard Analysis V&V- Test Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.157	0.093	0.629	0.125	0.214	0.063	0.088	0.101	0.245	0.116	0.667	0.204
Security Analysis V&V- Test Phase	0.642	0.146	0.292	0.111	0.067	0.068	0.179	0.129	0.621	0.135	0.200	0.065	0.088	0.101	0.245	0.116	0.667	0.204
Risk Analysis V&V- Test Phase	0.667	0.160	0.275	0.117	0.058	0.074	0.264	0.269	0.579	0.231	0.157	0.102	0.213	0.304	0.228	0.113	0.558	0.341
Configuration Management V&V - Test Phase	0.614	0.188	0.272	0.085	0.114	0.127	0.205	0.117	0.576	0.172	0.219	0.092	0.194	0.108	0.284	0.094	0.522	0.197
Review and Audit V&V - Test Phase	0.631	0.232	0.239	0.108	0.131	0.135	0.226	0.139	0.598	0.197	0.176	0.090	0.144	0.134	0.251	0.119	0.606	0.245
V&V Test Phase Activity Summary Report Generation	0.667	0.129	0.267	0.082	0.067	0.068	0.221	0.147	0.614	0.175	0.164	0.056	0.127	0.104	0.290	0.099	0.583	0.191
Installation Configuration Audit V&V	0.700	0.112	0.260	0.089	0.040	0.022	0.200	0.122	0.610	0.124	0.190	0.074	0.066	0.095	0.234	0.126	0.700	0.209
Installation Checkout V&V	0.800	0.137	0.180	0.110	0.020	0.027	0.200	0.122	0.650	0.150	0.150	0.050	0.056	0.100	0.194	0.131	0.750	0.224
Hazard Analysis V&V - Installation and Checkout Phase	0.670	0.144	0.290	0.124	0.040	0.022	0.170	0.067	0.620	0.110	0.210	0.074	0.060	0.082	0.220	0.110	0.720	0.175
Security Analysis V&V - Installation and Checkout Phase	0.670	0.144	0.290	0.124	0.040	0.022	0.170	0.067	0.620	0.110	0.210	0.074	0.060	0.082	0.220	0.110	0.720	0.175
Risk Analysis V&V - Installation and Checkout Phase	0.720	0.175	0.250	0.150	0.030	0.027	0.310	0.336	0.520	0.259	0.170	0.120	0.230	0.383	0.180	0.110	0.590	0.371
V&V Installation and Checkout Phase Activity Summary Report Generation	0.700	0.112	0.260	0.089	0.040	0.022	0.220	0.110	0.610	0.124	0.170	0.045	0.076	0.089	0.274	0.102	0.650	0.177
V&V Final Report Generation	0.700	0.112	0.260	0.089	0.040	0.022	0.220	0.110	0.610	0.124	0.170	0.045	0.076	0.089	0.274	0.102	0.650	0.177

Table D-9 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for Defect Density

Phase	Development Quality	Rank	1	2	3	4
Requirement	High	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	1922149.15	1939641.88	2062486.87	3772124.49
		BIC	3844329.16	3879314.61	4125004.59	7544264.41
		AIC	3844302.31	3879287.76	4124977.75	7544250.98
	Medium	Dist. Name	weibull	gamma	lognormal	exponential
		NLogL	4559592.09	4560662.70	4708513.30	5957749.31
		BIC	9119215.03	9121356.26	9417057.46	11915514.04
		AIC	9119188.18	9121329.41	9417030.61	11915500.62
	Low	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	7039038.00	7269648.93	7505569.34	9427642.45
		BIC	14078106.84	14539328.71	15011169.52	18855300.33
		AIC	14078079.99	14539301.86	15011142.68	18855286.91
Design	High	Dist. Name	gamma	weibull	lognormal	exponential
		NLogL	4587357.14	4610847.01	4731051.49	6039204.51
		BIC	9174745.12	9221724.86	9462133.82	12078424.45
		AIC	9174718.28	9221698.02	9462106.98	12078411.03
	Medium	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	6517484.40	6752792.46	6890442.34	8161334.41
		BIC	13034999.65	13505615.78	13780915.53	16322684.24
		AIC	13034972.81	13505588.93	13780888.68	16322670.82
	Low	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	9281301.63	9685507.76	9854906.71	10936492.60
		BIC	18562634.11	19371046.36	19709844.26	21873000.61
		AIC	18562607.27	19371019.52	19709817.41	21872987.19
Implementation	High	Dist. Name	gamma	weibull	lognormal	exponential
		NLogL	5960560.69	6129922.15	6525242.05	7032724.39
		BIC	11921152.23	12259875.14	13050514.94	14065464.20
		AIC	11921125.38	12259848.29	13050488.10	14065450.78
	Medium	Dist. Name	gamma	weibull	lognormal	exponential
		NLogL	8984001.40	9152037.08	9538932.12	9903234.00
		BIC	17968033.65	18304105.01	19077895.08	19806483.43
		AIC	17968006.80	18304078.16	19077868.23	19806470.01
	Low	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	8834173.15	9010730.34	9051692.56	9935665.65
		BIC	17668377.14	18021491.53	18103415.97	19871346.73
		AIC	17668350.29	18021464.68	18103389.12	19871333.31
Test	High	Dist. Name	gamma	weibull	exponential	lognormal
		NLogL	-538829.75	-347484.55	392627.84	430146.64
		BIC	-1077628.67	-694938.26	785271.09	860324.11
		AIC	-1077655.50	-694965.09	785257.67	860297.28
	Medium	Dist. Name	gamma	weibull	exponential	lognormal
		NLogL	2572412.25	2621045.38	3025620.19	3036420.66
		BIC	5144855.34	5242121.60	6051255.80	6072872.16
		AIC	5144828.49	5242094.75	6051242.37	6072845.31
	Low	Dist. Name	lognormal	weibull	gamma	exponential
		NLogL	7040251.67	7270858.26	7506573.92	9427835.12
		BIC	14080534.18	14541747.37	15013178.69	18855685.66
		AIC	14080507.33	14541720.52	15013151.84	18855672.24
Installation and Checkout	High	Dist. Name	gamma	weibull	lognormal	logistic
		NLogL	-7481520.6495	-7085559.8935	-6846270.3338	2827774.388
		BIC	-14963010.9207	-14171089.4086	-13692510.2892	5655579.1544'
		AIC	-14963037.299	-14171115.787	-13692536.6676	5655552.776
	Medium	Dist. Name	gamma	weibull	lognormal	logistic
		NLogL	-3242154.5727	-2573106.1407	-1889899.1251	7730450.276
		BIC	-6484278.2961	-5146181.432	-3779767.4008	15460931.4015
		AIC	-6484305.1455	-5146208.2814	-3779794.2502	15460904.5521
	Low	Dist. Name	gamma	weibull	lognormal	logistic
		NLogL	-898556.0721	-254047.663	506242.4797	10488443.0364
		BIC	-1797081.2947	-508064.4766	1012515.8089	20976916.9222
		AIC	-1797108.1442	-508091.3261	1012488.9594	20976890.0728

Table D-10 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the NPT of Defect Detection Probability for Current Phase

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
Requirement	-	High	High	Dist. Name	beta	normal	weibull	logistic
				NLogL	-1299711	-1050397	-1043919	-881292
				BIC	-2599392	-2100763	-2087808	-1762552
				AIC	-2599419	-2100790	-2087835	-1762579
		Medium		Dist. Name	beta	weibull	logistic	normal
				NLogL	-4066104	-3704182	-3442257	-3114793
				BIC	-8132178	-7408332	-6884483	-6229555
				AIC	-8132205	-7408359	-6884510	-6229582
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-3497031	-2426363	-2278487	-2012243
				BIC	-6994032	-4852695	-4556944	-4024456
				AIC	-6994058	-4852722	-4556970	-4024483
		High	Medium	Dist. Name	beta	normal	weibull	logistic
				NLogL	-846227	-626821	-607778	-451593
				BIC	-1692423	-1253610	-1215524	-903155
				AIC	-1692450	-1253637	-1215551	-903182
		Medium		Dist. Name	beta	weibull	normal	logistic
				NLogL	-2308111	-2001185	-1827962	-1731624
				BIC	-4616192	-4002338	-3655892	-3463216
				AIC	-4616219	-4002365	-3655919	-3463244
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-2892944	-2436278	-2169112	-2128732
				BIC	-5785856	-4872524	-4338194	-4257433
				AIC	-5785883	-4872551	-4338221	-4257460
		High	Low	Dist. Name	beta	weibull	gamma	normal
				NLogL	-1055211	-917138	-725390	-659178
				BIC	-2110390	-1834244	-1450749	-1318325
				AIC	-2110417	-1834271	-1450776	-1318352
		Medium		Dist. Name	beta	weibull	normal	gamma
				NLogL	-791827	-473816	-395469	-169301
				BIC	-1583624	-947600	-790907	-338570
				AIC	-1583651	-947627	-790934	-338597
		Low		Dist. Name	beta	weibull	normal	gamma
				NLogL	-807083	-561967	-468902	-268673
				BIC	-1614136	-1123903	-937773	-537316
				AIC	-1614163	-1123930	-937801	-537343
Design	-	High	High	Dist. Name	beta	weibull	normal	logistic
				NLogL	-1952857	-1519335	-1324315	-1162910
				BIC	-3905682	-3038639	-2648599	-2325789
				AIC	-3905709	-3038666	-2648627	-2325816
		Medium		Dist. Name	beta	weibull	logistic	normal
				NLogL	-3691098	-3249949	-2992555	-2797236
				BIC	-7382165	-6499867	-5985080	-5594442
				AIC	-7382192	-6499894	-5985107	-5594469
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-3615327	-2268889	-2092567	-1790741
				BIC	-7230623	-4537747	-4185103	-3581452
				AIC	-7230649	-4537774	-4185130	-3581479
		High	Medium	Dist. Name	beta	weibull	normal	logistic
				NLogL	-1192079	-1059380	-1029220	-820763
				BIC	-2384128	-2118729	-2058409	-1641495
				AIC	-2384155	-2118756	-2058436	-1641522
		Medium		Dist. Name	beta	weibull	normal	logistic
				NLogL	-2430667	-2197654	-2086323	-1911665
				BIC	-4861304	-4395276	-4172614	-3823300
				AIC	-4861331	-4395303	-4172641	-3823327
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-2775211	-2379456	-2207021	-2115887
				BIC	-5550391	-4758881	-4414011	-4231743
				AIC	-5550418	-4758908	-4414038	-4231770
		High	Low	Dist. Name	beta	weibull	normal	logistic
				NLogL	-1254330	-1135716	-1065643	-984055
				BIC	-2508629	-2271401	-2131255	-1968079
				AIC	-2508656	-2271428	-2131282	-1968106
		Medium		Dist. Name	beta	normal	weibull	logistic
				NLogL	-969825	-744582	-645515	-514695
				BIC	-1939620	-1489134	-1290999	-1029358
				AIC	-1939647	-1489161	-1291026	-1029385
		Low		Dist. Name	beta	normal	weibull	logistic
				NLogL	-1229169	-1088749	-1035607	-871426
				BIC	-2458308	-2177467	-2071183	-1742822
				AIC	-2458335	-2177495	-2071210	-1742849

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
— E P —	-	High	High	Dist. Name	beta	weibull	normal	logistic
				NLogL	-873452	-151435	-146732	50950.73

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
				BIC	-1746873	-302840	-293434	101932.3
				AIC	-1746900	-302867	-293461	101905.5
		Medium		Dist. Name	beta	weibull	logistic	normal
				NLogL	-3474368	-2661040	-2231585	-2198068
				BIC	-6948704	-5322049	-4463138	-4396104
				AIC	-6948731	-5322076	-4463165	-4396131
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-4437572	-2772929	-2583696	-1989730
				BIC	-8875113	-5545827	-5167362	-3979430
				AIC	-8875139	-5545853	-5167388	-3979457
		High	Medium	Dist. Name	beta	normal	weibull	logistic
				NLogL	-363696	65733.04	146535.6	237031.5
				BIC	-727361	131497	293102.1	474093.8
				AIC	-727388	131470.1	293075.2	474066.9
		Medium		Dist. Name	beta	weibull	normal	logistic
				NLogL	-1391154	-733790	-620979	-503439
				BIC	-2782277	-1467549	-1241926	-1006847
				AIC	-2782304	-1467576	-1241954	-1006874
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-2953507	-2723196	-2590304	-2332665
				BIC	-5906983	-5446361	-5180577	-4665298
				AIC	-5907009	-5446387	-5180604	-4665325
		High	Low	Dist. Name	beta	weibull	gamma	normal
				NLogL	-609460	-204785	-38190.2	-15335.8
				BIC	-1218889	-409539	-76349.5	-30640.6
				AIC	-1218916	-409566	-76376.5	-30667.6
		Medium		Dist. Name	beta	normal	logistic	weibull
				NLogL	-346568	-111438	73070.8	145887.5
				BIC	-693106	-222844	146172.7	291806.1
				AIC	-693133	-222872	146145.6	291779
		Low		Dist. Name	beta	normal	weibull	logistic
				NLogL	-715709	-548299	-520332	-345181
				BIC	-1431387	-1096568	-1040633	-690331
				AIC	-1431415	-1096595	-1040660	-690358
Test	High	High	High	Dist. Name	logistic	normal	beta	weibull
				NLogL	-4921544	-4762136	-4747864	-4722148
				BIC	-9843059	-9524242	-9495699	-9444265
				AIC	-9843085	-9524268	-9495725	-9444291
		Medium		Dist. Name	weibull	beta	logistic	normal
				NLogL	-4785597	-4693259	-4688970	-4545179
				BIC	-9571163	-9386488	-9377910	-9090328
				AIC	-9571189	-9386514	-9377936	-9090354
		Low		Dist. Name	weibull	beta	logistic	normal
				NLogL	-4720107	-4690073	-4545570	-4423467
				BIC	-9440184	-9380115	-9091110	-8846905
				AIC	-9440210	-9380141	-9091136	-8846930
		High	Medium	Dist. Name	lognormal	gamma	logistic	normal
				NLogL	-3479608	-3464246	-3420525	-3415820
				BIC	-6959186	-6928462	-6841020	-6831610
				AIC	-6959212	-6928488	-6841045	-6831635
		Medium		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-3281383	-3276114	-3250967	-3222512
				BIC	-6562736	-6552198	-6501903	-6444995
				AIC	-6562762	-6552224	-6501929	-6445021
		Low		Dist. Name	gamma	lognormal	normal	logistic
				NLogL	-3160928	-3160656	-3148486	-3102373
				BIC	-6321827	-6321282	-6296941	-6204716
				AIC	-6321852	-6321307	-6296967	-6204742
		High	Low	Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2524937	-2516708	-2462677	-2408219
				BIC	-5049844	-5033386	-4925325	-4816409
				AIC	-5049870	-5033412	-4925351	-4816435
		Medium		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2243053	-2238183	-2192855	-2129016
				BIC	-4486076	-4476336	-4385681	-4258002
				AIC	-4486102	-4476362	-4385707	-4258028
		Low		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2127938	-2124443	-2082549	-2016643
				BIC	-4255846	-4248856	-4165069	-4033256
				AIC	-4255871	-4248882	-4165095	-4033282

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
Test	Medium	High	High	Dist. Name	logistic	lognormal	gamma	normal
				NLogL	-4493735	-4412489	-4410644	-4393557
				BIC	-8987440	-8824949	-8821258	-8787084
				AIC	-8987466	-8824975	-8821284	-8787110
		Medium		Dist. Name	logistic	normal	gamma	lognormal
				NLogL	-4315737	-4266985	-4254772	-4243357
				BIC	-8631444	-8533939	-8509514	-8486685
				AIC	-8631470	-8533965	-8509540	-8486711
		Low		Dist. Name	normal	gamma	lognormal	logistic
				NLogL	-3988697	-3969738	-3956173	-3953283
				BIC	-7977363	-7939447	-7912316	-7906536
				AIC	-7977389	-7939473	-7912341	-7906562
		High	Medium	Dist. Name	lognormal	gamma	normal	beta
				NLogL	-2438746	-2428222	-2375234	-2328905
				BIC	-4877461	-4856414	-4750438	-4657780
				AIC	-4877487	-4856440	-4750463	-4657806
		Medium		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-2152786	-2145172	-2099899	-2012433
				BIC	-4305543	-4290313	-4199767	-4024837
				AIC	-4305569	-4290339	-4199793	-4024862
		Low		Dist. Name	gamma	lognormal	normal	weibull
				NLogL	-1897774	-1895690	-1871561	-1824662
				BIC	-3795518	-3791350	-3743092	-3649294
				AIC	-3795543	-3791375	-3743118	-3649319
		High	Low	Dist. Name	lognormal	gamma	normal	beta
				NLogL	-2133833	-2085683	-1938218	-1937235
				BIC	-4267636	-4171337	-3876407	-3874439
				AIC	-4267662	-4171363	-3876433	-3874465
		Medium		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1826078	-1778791	-1633842	-1555412
				BIC	-3652126	-3557552	-3267655	-3110794
				AIC	-3652152	-3557577	-3267681	-3110820
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1600334	-1559841	-1425865	-1366060
				BIC	-3200637	-3119653	-2851700	-2732090
				AIC	-3200663	-3119679	-2851726	-2732116
	Low	High	High	Dist. Name	gamma	lognormal	normal	beta
				NLogL	-3247992	-3245286	-3236607	-3185884
				BIC	-6495954	-6490543	-6473184	-6371738
				AIC	-6495980	-6490569	-6473209	-6371764
		Medium		Dist. Name	gamma	normal	lognormal	weibull
				NLogL	-2968907	-2966672	-2962464	-2889670
				BIC	-5937783	-5933314	-5924899	-5779310
				AIC	-5937809	-5933340	-5924925	-5779336
		Low		Dist. Name	normal	weibull	gamma	lognormal
				NLogL	-2647916	-2647763	-2633626	-2619086
				BIC	-5295802	-5295496	-5267222	-5238142
				AIC	-5295828	-5295522	-5267248	-5238168
		High	Medium	Dist. Name	lognormal	gamma	normal	beta
				NLogL	-2440156	-2430220	-2373060	-2359562
				BIC	-4880283	-4860411	-4746091	-4719095
				AIC	-4880308	-4860437	-4746117	-4719121
		Medium		Dist. Name	lognormal	gamma	normal	beta
				NLogL	-2133793	-2124984	-2070868	-1993157
				BIC	-4267556	-4249937	-4141706	-3986285
				AIC	-4267582	-4249963	-4141731	-3986311
		Low		Dist. Name	beta	weibull	normal	gamma
				NLogL	-1911286	-1859024	-1777983	-1709395
				BIC	-3822543	-3718019	-3555936	-3418761
				AIC	-3822569	-3718044	-3555962	-3418786
		High	Low	Dist. Name	lognormal	gamma	beta	normal
				NLogL	-2147169	-2097178	-1975812	-1935945
				BIC	-4294308	-4194326	-3951593	-3871861
				AIC	-4294334	-4194352	-3951619	-3871886
		Medium		Dist. Name	lognormal	gamma	normal	beta
				NLogL	-1835359	-1785849	-1625879	-1610712
				BIC	-3670689	-3571667	-3251728	-3221393
				AIC	-3670715	-3571693	-3251754	-3221419
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1605139	-1560991	-1407785	-1371979
				BIC	-3210249	-3121952	-2815539	-2743929
				AIC	-3210275	-3121978	-2815565	-2743954

Phase	Development Quality	Complexity	V&V Quality	Rank	1	2	3	4
Installation and Checkout	-	High	High	Dist. Name	beta	weibull	normal	logistic
				NLogL	-2456391	-2181566	-1894327	-1819800
				BIC	-4912751	-4363101	-3788622	-3639570
				AIC	-4912778	-4363128	-3788649	-3639597
		Medium		Dist. Name	beta	weibull	normal	gamma
				NLogL	-5903018	-5756223	-5655983	-5594404
				BIC	-1.2E+07	-1.2E+07	-1.1E+07	-1.1E+07
				AIC	-1.2E+07	-1.2E+07	-1.1E+07	-1.1E+07
		Low		Dist. Name	beta	weibull	logistic	normal

				NLogL	-5732120	-5479116	-5206592	-5176379
				BIC	-1.1E+07	-1.1E+07	-1E+07	-1E+07
				AIC	-1.1E+07	-1.1E+07	-1E+07	-1E+07
		High	Medium	Dist. Name	beta	logistic	normal	weibull
				NLogL	-1066772	-725133	-721852	-662129
				BIC	-2133514	-1450235	-1443672	-1324226
				AIC	-2133540	-1450262	-1443699	-1324253
		Medium		Dist. Name	beta	weibull	logistic	normal
				NLogL	-2244992	-1934343	-1767554	-1635325
				BIC	-4489954	-3868654	-3535078	-3270620
				AIC	-4489981	-3868681	-3535105	-3270647
		Low		Dist. Name	beta	weibull	logistic	normal
				NLogL	-3363131	-3179937	-2952260	-2701150
				BIC	-6726232	-6359844	-5904489	-5402270
				AIC	-6726259	-6359871	-5904516	-5402297
		High	Low	Dist. Name	beta	normal	weibull	logistic
				NLogL	-1537749	-1430383	-1395190	-1207449
				BIC	-3075466	-2860735	-2790350	-2414868
				AIC	-3075493	-2860762	-2790376	-2414894
		Medium		Dist. Name	beta	weibull	normal	gamma
				NLogL	-1636487	-1523404	-1461767	-1240821
				BIC	-3272944	-3046777	-2923503	-2481611
				AIC	-3272971	-3046804	-2923530	-2481638
		Low		Dist. Name	beta	weibull	normal	logistic
				NLogL	-2038658	-1988014	-1917805	-1702968
				BIC	-4077284	-3975997	-3835580	-3405905
				AIC	-4077311	-3976023	-3835607	-3405932

Table D-11 Best Fitted Distribution Sorted by Bayesian Information Criterion (BIC) for the NPT of Defect Detection Probability for the Previous Phase

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
Design	-	High	High	Dist. Name	beta	weibull	normal	gamma
				NLogL	-372985	123947.7	297221.7	308118.7
				BIC	-745940	247926.2	594474.2	616268.3
				AIC	-745967	247899.4	594447.3	616241.5
		Medium		Dist. Name	beta	normal	weibull	logistic
				NLogL	-100610	1091782	1120373	1379376
				BIC	-201190	2183594	2240777	2758784
				AIC	-201216	2183567	2240750	2758757
		Low		Dist. Name	beta	weibull	gamma	lognormal
				NLogL	-151243	803251.3	871065.3	1032294
				BIC	-302455	1606533	1742161	2064618
				AIC	-302481	1606507	1742135	2064592
		High	Medium	Dist. Name	beta	weibull	gamma	lognormal
				NLogL	-1116899	-769721	-722310	-331452
				BIC	-2233768	-1539411	-1444589	-662874
				AIC	-2233795	-1539438	-1444616	-662901
		Medium		Dist. Name	beta	weibull	gamma	normal
				NLogL	-493986	26071.58	150911.7	339389.5
				BIC	-987942	52174.01	301854.2	678809.9
				AIC	-987969	52147.16	301827.3	678783.1
		Low		Dist. Name	beta	weibull	gamma	normal
				NLogL	-158922	545732.9	700911	754648.7
				BIC	-317813	1091497	1401853	1509328
				AIC	-317840	1091470	1401826	1509301
		High	Low	Dist. Name	beta	gamma	weibull	lognormal
				NLogL	-4264252	-4199696	-4190713	-4181295
				BIC	-8528473	-8399361	-8381395	-8362559
				AIC	-8528500	-8399387	-8381422	-8362586
		Medium		Dist. Name	beta	lognormal	gamma	weibull
				NLogL	-3744313	-3717746	-3706677	-3671829
				BIC	-7488595	-7435461	-7413323	-7343628
				AIC	-7488622	-7435488	-7413350	-7343655
		Low		Dist. Name	beta	gamma	weibull	lognormal
				NLogL	-3462287	-3414023	-3386702	-3359835
				BIC	-6924543	-6828015	-6773374	-6719640
				AIC	-6924570	-6828042	-6773401	-6719667
Implementation	-	High	High	Dist. Name	beta	weibull	normal	gamma
				NLogL	-558189	-194433	-109565	47431.57
				BIC	-1116347	-388835	-219099	94894
				AIC	-1116373	-388862	-219126	94867.15
		Medium		Dist. Name	beta	normal	weibull	logistic
				NLogL	-504965	136704.8	225868.4	312011.2
				BIC	-1009899	273440.4	451767.7	624053.3
				AIC	-1009926	273413.6	451740.9	624026.4
		Low		Dist. Name	beta	weibull	normal	gamma
				NLogL	-454826	251513.2	330405.1	419313.3
				BIC	-909622	503056.9	660840.5	838657
				AIC	-909648	503030.5	660814.1	838630.6
		High	Medium	Dist. Name	beta	weibull	gamma	normal
				NLogL	-1092942	-674572	-576152	-298907
				BIC	-2185853	-1349113	-1152274	-597783
				AIC	-2185880	-1349140	-1152301	-597810
		Medium		Dist. Name	beta	weibull	normal	gamma
				NLogL	-620470	-163554	-111576	50964.8
				BIC	-1240909	-327077	-223122	101960.4
				AIC	-1240936	-327104	-223149	101933.6
		Low		Dist. Name	beta	normal	weibull	logistic
				NLogL	-236730	296857.3	332080.9	512987.1
				BIC	-473429	593745.4	664192.6	1026005
				AIC	-473456	593718.5	664165.8	1025978
		High	Low	Dist. Name	beta	weibull	gamma	lognormal
				NLogL	-2907097	-2729152	-2671572	-2215352
				BIC	-5814163	-5458273	-5343114	-4430674
				AIC	-5814189	-5458300	-5343140	-4430701
		Medium		Dist. Name	beta	weibull	gamma	normal
				NLogL	-2429260	-2309408	-2202562	-1903169
				BIC	-4858489	-4618785	-4405092	-3806308
				AIC	-4858516	-4618812	-4405119	-3806335
		Low		Dist. Name	beta	weibull	normal	gamma
				NLogL	-2398175	-2364047	-2221285	-2166334
				BIC	-4796319	-4728063	-4442538	-4332638
				AIC	-4796346	-4728090	-4442565	-4332665

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
Test	High	High	High	Dist. Name	logistic	normal	beta	gamma
				NLogL	-5273468	-5159970	-5132453	-5125128
				BIC	-1.1E+07	-1E+07	-1E+07	-1E+07
				AIC	-1.1E+07	-1E+07	-1E+07	-1E+07
		Medium		Dist. Name	weibull	logistic	beta	normal
				NLogL	-5048954	-5028334	-4964800	-4936872
				BIC	-1E+07	-1E+07	-9929570	-9873714
				AIC	-1E+07	-1E+07	-9929596	-9873740
		Low		Dist. Name	weibull	beta	logistic	normal
				NLogL	-4985886	-4927824	-4866656	-4807026
				BIC	-9971742	-9855619	-9733282	-9614022
				AIC	-9971767	-9855645	-9733308	-9614047
		High	Medium	Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-3059450	-3047398	-3001943	-2968217
				BIC	-6118870	-6094765	-6003856	-5936404
				AIC	-6118896	-6094791	-6003882	-5936430
		Medium		Dist. Name	beta	weibull	normal	gamma
				NLogL	-1954614	-1936906	-1883242	-1810623
				BIC	-3909198	-3873783	-3766453	-3621217
				AIC	-3909224	-3873809	-3766479	-3621242
		Low		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2789670	-2786270	-2763715	-2704663
				BIC	-5579310	-5572510	-5527400	-5409296
				AIC	-5579336	-5572535	-5527425	-5409321
		High	Low	Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2146647	-2130612	-2056713	-1984725
				BIC	-4293265	-4261194	-4113396	-3969419
				AIC	-4293290	-4261219	-4113422	-3969445
		Medium		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-1913664	-1903591	-1845718	-1762183
				BIC	-3827298	-3807151	-3691407	-3524335
				AIC	-3827324	-3807177	-3691432	-3524361
		Low		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-1839334	-1830126	-1775969	-1696327
				BIC	-3678639	-3660222	-3551909	-3392623
				AIC	-3678665	-3660248	-3551935	-3392649
	Medium	High	High	Dist. Name	logistic	lognormal	gamma	normal
				NLogL	-4539529	-4519694	-4504904	-4465285
				BIC	-9079029	-9039359	-9009779	-8930540
				AIC	-9079055	-9039384	-9009805	-8930566
		Medium		Dist. Name	logistic	lognormal	gamma	normal
				NLogL	-4380855	-4380249	-4377965	-4365833
				BIC	-8761679	-8760468	-8755901	-8731637
				AIC	-8761705	-8760494	-8755927	-8731662
		Low		Dist. Name	normal	gamma	lognormal	logistic
				NLogL	-4055819	-4052060	-4047267	-3992661
				BIC	-8111609	-8104091	-8094503	-7985292
				AIC	-8111635	-8104117	-8094529	-7985318
		High	Medium	Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2115058	-2070746	-1945146	-1866320
				BIC	-4230087	-4141461	-3890262	-3732611
				AIC	-4230113	-4141487	-3890287	-3732636
		Medium		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-1888348	-1849563	-1738998	-1648955
				BIC	-3776665	-3699096	-3477967	-3297880
				AIC	-3776691	-3699122	-3477993	-3297906
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1729131	-1698886	-1606207	-1509681
				BIC	-3458232	-3397742	-3212384	-3019333
				AIC	-3458258	-3397768	-3212410	-3019359
		High	Low	Dist. Name	lognormal	gamma	normal	beta
				NLogL	-1765082	-1708302	-1536743	-1532329
				BIC	-3530133	-3416573	-3073457	-3064628
				AIC	-3530159	-3416599	-3073482	-3064653
		Medium		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1478400	-1423754	-1258566	-1214384
				BIC	-2956770	-2847479	-2517102	-2428739
				AIC	-2956796	-2847504	-2517128	-2428764
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1302227	-1254478	-1101819	-1068930
				BIC	-2604423	-2508926	-2203608	-2137831
				AIC	-2604449	-2508952	-2203634	-2137857

Phase	Development Quality	Complexity	V&V quality	Rank	1	2	3	4
Test	Low	High	High	Dist. Name	logistic	lognormal	gamma	normal
				NLogL	-4540058	-4521243	-4506443	-4466849
				BIC	-9080086	-9042456	-9012856	-8933668
				AIC	-9080112	-9042482	-9012882	-8933693
		Medium		Dist. Name	logistic	lognormal	gamma	normal
				NLogL	-4377690	-4376190	-4374050	-4362145
				BIC	-8755351	-8752350	-8748070	-8724261
				AIC	-8755377	-8752376	-8748096	-8724287
		Low		Dist. Name	normal	gamma	lognormal	logistic
				NLogL	-4055179	-4051183	-4046259	-3992034
				BIC	-8110329	-8102336	-8092489	-7984039
				AIC	-8110354	-8102362	-8092515	-7984065
		High	Medium	Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-2115973	-2071828	-1946572	-1867522
				BIC	-4231916	-4143627	-3893114	-3735014
				AIC	-4231942	-4143652	-3893140	-3735040
		Medium		Dist. Name	lognormal	gamma	normal	logistic
				NLogL	-1887803	-1849111	-1738742	-1648422
				BIC	-3775575	-3698191	-3477455	-3296814
				AIC	-3775601	-3698217	-3477480	-3296840
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1727385	-1697114	-1604386	-1508020
				BIC	-3454740	-3394197	-3208743	-3016010
				AIC	-3454766	-3394223	-3208768	-3016036
		High	Low	Dist. Name	lognormal	gamma	normal	beta
				NLogL	-1765522	-1708698	-1537065	-1533310
				BIC	-3531015	-3417366	-3074100	-3066590
				AIC	-3531041	-3417391	-3074126	-3066615
		Medium		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1478181	-1423531	-1258316	-1214130
				BIC	-2956332	-2847032	-2516603	-2428229
				AIC	-2956358	-2847058	-2516629	-2428255
		Low		Dist. Name	lognormal	gamma	normal	weibull
				NLogL	-1303894	-1256083	-1103348	-1070255
				BIC	-2607758	-2512136	-2206667	-2140481
				AIC	-2607784	-2512162	-2206693	-2140507
Installation and Checkout	-	High	High	Dist. Name	beta	weibull	normal	logistic
				NLogL	-1114640	-613773	-482365	-419786
				BIC	-2229250	-1227517	-964700	-839543
				AIC	-2229277	-1227543	-964726	-839569
		Medium		Dist. Name	beta	weibull	normal	logistic
				NLogL	-2880646	-2547672	-2318657	-2220877
				BIC	-5761262	-5095313	-4637285	-4441724
				AIC	-5761288	-5095340	-4637311	-4441750
		Low		Dist. Name	beta	weibull	normal	logistic
				NLogL	-5346065	-5025287	-4645275	-4617096
				BIC	-1.1E+07	-1E+07	-9290519	-9234163
				AIC	-1.1E+07	-1E+07	-9290546	-9234189
		High	Medium	Dist. Name	beta	normal	logistic	weibull
				NLogL	-262008	232715.5	318584.6	397204.6
				BIC	-523985	465461.3	637199.7	794439.6
				AIC	-524011	465434.9	637173.3	794413.2
		Medium		Dist. Name	beta	weibull	normal	logistic
				NLogL	-659794	-23539.9	6741.033	112152.8
				BIC	-1319558	-47049.3	13512.47	224336
				AIC	-1319584	-47075.7	13486.07	224309.6
		Low		Dist. Name	beta	weibull	normal	logistic
				NLogL	-1325720	-794973	-653150	-545464
				BIC	-2651409	-1589916	-1306270	-1090898
				AIC	-2651436	-1589943	-1306297	-1090924
		High	Low	Dist. Name	beta	weibull	gamma	normal
				NLogL	-360509	336389	389400.6	506594.2
				BIC	-720987	672808.4	778831.5	1013219
				AIC	-721013	672782	778805.2	1013192
		Medium		Dist. Name	beta	weibull	gamma	normal
				NLogL	-102509	701454.4	786787	893210.5
				BIC	-204988	1402939	1573604	1786451
				AIC	-205014	1402913	1573578	1786425
		Low		Dist. Name	beta	weibull	normal	gamma
				NLogL	-19601.6	740160.3	848405.5	876625
				BIC	-39172.8	1480351	1696841	1753280
				AIC	-39199.1	1480325	1696815	1753254

Table D-12 Gamma Distribution Fit for the NPT of Defect Density

Phase	Development Quality	Shape	Scale
Requirement	High	0.4316	1.8157
	Medium	0.4619	2.6240
	Low	0.4143	5.8638
Design	High	0.4556	2.7173
	Medium	0.4760	3.9583
	Low	0.4999	6.5687
Implementation	High	0.5007	3.0148
	Medium	0.5238	5.0947
	Low	0.5293	5.0782
Test	High	0.5201	0.7660
	Medium	0.6225	1.0827
	Low	0.4143	5.8635
Installation and Checkout	High	0.5710	0.6827
	Medium	0.6207	1.4711
	Low	0.5797	2.6308

Table D-13 Beta Distribution Fit for the NPT of Defect Detection Probability in the Current Phase

Phase	V&V quality	Development quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Requirement	High	-	3.088	1.833	7.022	1.751	4.297	0.790
	Medium	-	2.361	1.975	4.264	1.919	4.731	1.511
	Low	-	1.229	2.296	1.690	2.192	1.934	2.168
Design	High	-	3.759	2.023	5.998	1.698	3.985	0.762
	Medium	-	2.716	2.554	4.567	2.284	4.569	1.480
	Low	-	1.322	2.587	1.785	2.421	2.400	2.668
Implementation	High	-	2.342	1.383	5.307	1.377	4.996	0.702
	Medium	-	1.709	1.563	2.902	1.533	5.853	1.659
	Low	-	0.955	1.619	1.411	1.668	2.004	2.016
Test	High	High	49.439	12.288	37.230	12.392	20.170	9.693
	Medium	High	20.190	8.413	10.971	6.832	10.855	8.154
	Low	High	11.500	7.159	7.579	6.792	7.141	7.674
	High	Medium	16.798	4.489	26.782	5.463	14.690	4.904
	Medium	Medium	31.704	4.032	8.321	3.834	8.619	4.951
	Low	Medium	14.771	4.147	6.100	4.284	6.041	5.211
	High	Low	25.493	2.291	21.684	3.541	11.445	3.264
	Medium	Low	13.316	3.197	6.900	2.813	8.049	4.146

Phase	V&V quality	Development quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
	Low	Low	7.546	3.087	5.146	3.254	5.186	4.065
Installation and Checkout	High	-	5.450	2.457	13.085	1.985	13.505	1.250
	Medium	-	2.818	2.160	4.832	2.154	6.862	2.034
	Low	-	2.148	3.624	2.931	3.818	4.113	4.334

Table D-14 Beta Distribution Fit for the NPT of the Defect Detection Drobability in the Previous Phase

Phase	V&V quality	Development quality	High Complexity		Medium Complexity		Low Complexity	
			α	β	α	β	α	β
Design	High	-	1.3095	1.7345	1.2688	1.0514	1.0873	0.8117
	Medium	-	1.0128	2.1429	1.1353	1.7571	1.2079	1.4215
	Low	-	1.1601	6.0078	1.5224	6.3937	1.9101	6.7609
Implementation	High	-	1.7034	2.0265	1.9422	1.4440	1.6298	0.9597
	Medium	-	1.0646	2.2064	1.2764	1.9959	1.4098	1.5461
	Low	-	1.1049	4.0713	1.4181	4.2264	2.1135	5.1104
Test	High	High	64.1113	15.6275	37.8885	12.2042	15.0247	6.7230
	Medium	High	15.5834	6.7702	7.6550	4.7268	7.7542	5.7831
	Low	High	8.3954	4.8811	5.7545	4.8246	5.5333	5.5696
	High	Medium	37.1415	4.5249	26.3886	5.1372	11.2259	3.4673
	Medium	Medium	7.3323	2.5763	5.8803	2.7207	5.9910	3.4135
	Low	Medium	6.4427	2.7710	4.5403	2.9566	4.5498	3.6371
	High	Low	29.0140	2.4546	21.5491	3.3479	9.4293	2.5205
	Medium	Low	10.3286	2.6785	5.2210	2.1903	5.1108	2.6217
	Low	Low	5.8949	2.2929	3.9427	2.3282	3.9133	2.8338
Installation and Checkout	High	-	3.2360	1.7659	6.2600	1.5622	13.8766	1.4457
	Medium	-	1.6810	1.5003	2.2365	1.3126	3.1949	1.3703
	Low	-	0.8258	1.3336	0.8505	1.0864	0.9799	1.0904

Table D-15 Bayesian Updated Result for the NPT of Defect Detection Probability at Current Phase from Reference Textbook Data

Phase	Complexity	V&V quality	Development quality	Alpha	Beta	Mean	Variance
Requirement	High	High	-	3.012	1.680	0.642	0.040
	High	Medium	-	2.299	1.728	0.571	0.049
	High	Low	-	1.272	1.780	0.417	0.060
	Medium	High	-	6.446	1.427	0.819	0.017
	Medium	Medium	-	3.911	1.498	0.723	0.031
	Medium	Low	-	1.637	1.510	0.520	0.060
	Low	High	-	3.739	0.588	0.864	0.022
	Low	Medium	-	4.087	1.056	0.795	0.027
	Low	Low	-	1.810	1.403	0.563	0.058
Design	High	High	-	3.621	1.789	0.669	0.035
	High	Medium	-	2.671	2.264	0.541	0.042
	High	Low	-	1.382	2.044	0.403	0.054
	Medium	High	-	5.446	1.341	0.802	0.020
	Medium	Medium	-	4.244	1.817	0.700	0.030
	Medium	Low	-	1.755	1.712	0.506	0.056
	Low	High	-	3.336	0.524	0.864	0.024
	Low	Medium	-	3.930	1.024	0.793	0.028
	Low	Low	-	2.269	1.748	0.565	0.049
Implementation	High	High	-	2.255	1.225	0.648	0.051
	High	Medium	-	1.637	1.272	0.563	0.063
	High	Low	-	0.957	1.176	0.449	0.079
	Medium	High	-	4.730	1.051	0.818	0.022
	Medium	Medium	-	2.564	1.073	0.705	0.045
	Medium	Low	-	1.306	1.050	0.554	0.074
	Low	High	-	3.836	0.410	0.903	0.017
	Low	Medium	-	4.352	0.832	0.840	0.022
	Low	Low	-	1.793	1.181	0.603	0.060
Test	High	High	High	48.615	11.761	0.805	0.003
	High	Medium	High	19.336	7.330	0.725	0.007
	High	Low	High	11.156	6.261	0.641	0.013
	Medium	High	High	36.812	11.989	0.754	0.004
	Medium	Medium	High	10.749	6.233	0.633	0.013
	Medium	Low	High	7.508	6.189	0.548	0.017
	Low	High	High	20.044	9.475	0.679	0.007

Phase	Complexity	V&V quality	Development quality	Alpha	Beta	Mean	Variance
	Low	Medium	High	10.726	7.557	0.587	0.013
	Low	Low	High	7.131	7.104	0.501	0.016
	High	High	Medium	10.558	1.279	0.892	0.008
	High	Medium	Medium	44.689	11.028	0.802	0.003
	High	Low	Medium	15.643	6.342	0.712	0.009
	Medium	High	Medium	26.192	5.173	0.835	0.004
	Medium	Medium	Medium	8.001	3.376	0.703	0.017
	Medium	Low	Medium	5.951	3.796	0.610	0.022
	Low	High	Medium	14.519	4.744	0.754	0.009
	Low	Medium	Medium	8.403	4.477	0.652	0.016
	Low	Low	Medium	5.967	4.726	0.558	0.021
	High	High	Low	24.051	2.022	0.922	0.003
	High	Medium	Low	12.033	2.493	0.828	0.009
	High	Low	Low	7.018	2.497	0.738	0.018
	Medium	High	Low	21.058	3.308	0.864	0.005
	Medium	Medium	Low	6.565	2.431	0.730	0.020
	Medium	Low	Low	4.976	2.840	0.637	0.026
	Low	High	Low	11.276	3.138	0.782	0.011
	Low	Medium	Low	7.795	3.703	0.678	0.017
	Low	Low	Low	5.091	3.647	0.583	0.025
Installation/Checkout	High	High	-	5.384	2.365	0.695	0.024
	High	Medium	-	2.740	1.900	0.591	0.043
	High	Low	-	2.205	3.069	0.418	0.039
	Medium	High	-	12.448	1.775	0.875	0.007
	Medium	Medium	-	4.483	1.730	0.722	0.028
	Medium	Low	-	2.934	3.080	0.488	0.036
	Low	High	-	12.356	1.037	0.923	0.005
	Low	Medium	-	6.000	1.458	0.805	0.019
	Low	Low	-	4.015	3.324	0.547	0.030

NRC FORM 335
(12-2010)
NRCMD 3.7

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

(See instructions on the reverse)

1. REPORT NUMBER
(Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)
NUREG/CR-7233
BNL-NUREG-112737-2016
KAERI/TR-6006/2015

2. TITLE AND SUBTITLE
Developing a Bayesian Belief Network Model for Quantifying the Probability of Software Failure of a Protection System

3. DATE REPORT PUBLISHED

MONTH	YEAR
January	2018

4. FIN OR GRANT NUMBER
NRC-HQ-60-15-D-0006

5. AUTHOR(S)
Tsong-Lun Chu, Athi Varuttamaseni, Meng Yue,
Seung Jun Lee, Hyun Gook Kang,
Jaehyun Cho,
Steve Yang

6. TYPE OF REPORT
Technical

7. PERIOD COVERED (Inclusive Dates)

8. PERFORMING ORGANIZATION - NAME AND ADDRESS (If NRC, provide Division, Office or Region, U. S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)
Brookhaven National Laboratory
P.O. Box 5000
Upton, NY 11973

9. SPONSORING ORGANIZATION - NAME AND ADDRESS (If NRC, type "Same as above", if contractor, provide NRC Division, Office or Region, U. S. Nuclear Regulatory Commission, and mailing address.)
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

10. SUPPLEMENTARY NOTES

11. ABSTRACT (200 words or less)
A new approach has been developed to quantify the software failure probabilities in nuclear power plant digital instrumentation and control systems. Specifically, this approach uses a Bayesian belief network (BBN) to model the causal relationships between the software development life cycle, the number of residual defects within software, and the software failure probability. The software development life cycle (SDLC) characteristics and software-self characteristics (e.g., size and complexity) are represented using a hierarchical structure. Information for each SDLC phase (or activity) was abstracted from the relevant guidance and standards documents. A BBN sub-model was then developed for each phase to estimate the number of software defects remaining.

Three rounds of expert elicitation were used to complete the BBN model. The first two rounds assisted in the identification of nodes, the construction of the model, and the establishment of the Node Probability Tables (NPTs). The NPTs were further updated using literature data available from the literature and the limited amount of development and V&V data. The model was applied to two trial nuclear systems based on the third round experts' inputs. The results as well as the feasibility of using BBNs for quantifying software failure probabilities are discussed herein.

12. KEY WORDS/DESCRIPTORS (List words or phrases that will assist researchers in locating the report.)
Digital Instrumentation and Controls, PRA, Software Failure Probability

13. AVAILABILITY STATEMENT
unlimited

14. SECURITY CLASSIFICATION
(This Page)
unclassified
(This Report)
unclassified

15. NUMBER OF PAGES

16. PRICE

Federal Recycling Program

UNITED STATES
NUCLEAR REGULATORY COMMISSION
WASHINGTON, DC 20555-0001

OFFICIAL BUSINESS

NUREG/CR-7233

Developing a Bayesian Belief Network Model for Quantifying the Probability of Software Failure of a Protection System

January 2018