Implementation of V&V Tasks for Improving Nuclear I&C System Software Safety

Bao-Juan Yin[1], Jing Li[1], Ya-Qi Wang[1], Da-Hu Liu[1](✉), and You-Yuan Li[2]

[1] Nuclear and Radiation Safety Center, No. 54, HongLianNanCun, Haidian, Beijing, People's Republic of China
liudahu@chinansc.cn

[2] China Techenergy Co., Ltd., Building 5, Yongfeng Road, Haidian, Beijing, People's Republic of China

Abstract. As more computer-based instrumentation and control (I&C) systems are used in nuclear power plants (NPPs), software safety becomes more and more important for safe and reliable operation of NPPs. Based on the hidden nature of software itself, its safety needs to be guaranteed by being strictly verified and validated in the process of software development to eliminate the potential design faults. The tasks performed by verification and validation (V&V) personnel play an important role in improving safety of I&C system software used in NPPs. Three key V&V tasks including traceability analysis, hazard analysis and safety testing are highlighted; their relationship and implementation methods are discussed; the implementation methods can be applied or referenced in future software safety V&V and improving digital I&C system safety.

Keywords: V&V · I&C system · Software safety
Traceability analysis · Hazard analysis · Safety testing

1 Introduction

According to the life cycle of software, software development process can be divided into five stages [1] which are concept stage, requirements stage, design stage, implementation stage and testing stage; software [1] verification activities should cover all these stages to ensure realization of software safety. In the first four stages of software development process, software safety is verified by using selectable analysis techniques and analyzing all outputs of each stage to identify and evaluate all potential hazards; in testing stage, software safety is validated by thoroughly testing software itself, and final safety evaluation of software is performed.

As the outputs of prior stage are inputs of next stage during software development process, in order to better verification results of each stage and ensure the final software products meet requirements of the system, traceability analysis is needed in safety verification during software development process, and its results are used as the basis of application of other verification tasks. Traceability analysis is used to evaluate consistency of outputs of each development stage to system requirements [2]; hazard analysis is used to identify and evaluate all potential hazards during realization of system requirements [3]; safety testing is used to validate if the final software products are safe enough; these three verification and validation (V&V) tasks are an organic whole for improving nuclear I&C system software safety, achieving different goals and complementing each other (Fig. 1).

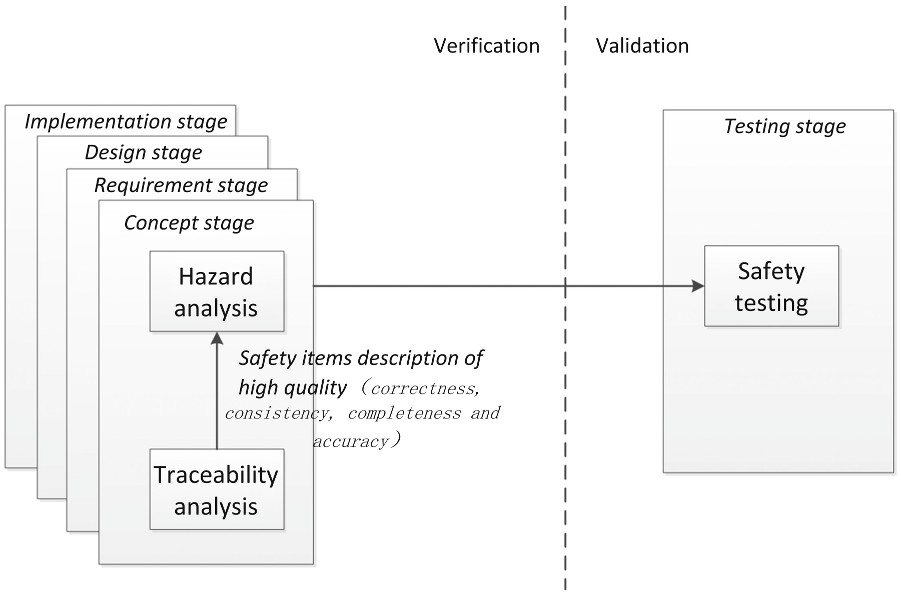

Fig. 1. Relationship of traceability analysis, hazard analysis and safety testing

2 Traceability Analysis

Meeting functional and nonfunctional I&C system requirements is the basis of software safety. Traceability analysis is one verification task which performs two-way traceable analysis between the outputs of one stage in software life cycle and that of its previous stage, and is process of ensuring all outputs of each latter stage could implement correctly the demands of software. Traceability analysis is a verification task that runs through the whole life cycle of software development in parallel with the software development task. Traceability analysis is implemented as follows:

(1) First of all, the tracking matrix of the product outputs in the previous stage and those in current stage of the life cycle should be established to establish the tracking relationship between them.

In order to facilitate tracking, software development documents and test documents should be prepared by using structured methods, supporting automation tools which are capable of handling all software development documents for large-scale tracking should be chosen, and tracking matrix concerning development documents should be associated with configuration management and change control, to ensure that any content changes reflect to the tracking matrix and lay foundation for other safety verification tasks.

(2) Secondly, correctness, consistency, completeness and accuracy [2] of the tracking contents should be analyzed, and complete tracing relationship between them in the contents should be established. It is verified that software completely realizes the needs of users, and no superfluous functions which are outside of user demand are achieved.

Traceability analysis is one basal task which can improve quality of analysis contents, ensure correctness, consistency, completeness and accuracy of safety items, facilitate performance of other analysis tasks, and form objective evidence of software development quality.

3 Hazard Analysis

Hazard analysis is one necessary task for SIL4 software required by IEEE 1012, but this standard only focuses on completeness of process and tasks, no operational guidance is provided. Annex D of IEEE 7-4.3.2 gives a variety of helpful information for hazard analysis. The purpose of hazard analysis is to identify and evaluate factors potentially devastating system; it focuses on system safety-critical functions, failure mechanism affecting implementation of these safety functions which may cause disastrous consequences, and prevention/control measures of these potential factors. After hazard analysis, each hazard item should be determined by the severity level of its impact, the probability of its occurrence [3], and the rating which reflects impact degree of this hazard once occurred. The assessment criteria for the severity and occurrence of the hazard items identified at different stages should be determined before performing hazard analysis.

3.1 Relationship Between Software Hazard Analysis and System Hazard Analysis

Hazard analysis is aimed at the whole I&C system and its safety. All components of the system, its surrounding environment and its external interfaces should be the object of hazard analysis [4]. As an important part of the system, software doesn't fail as time goes by; it fails at some time with specific trigging conditions due to design defects. Function failures of software may lead the system to hazard status and cause the system failure. So the purpose of software hazard analysis is to identify and evaluate software internal potential factors which may cause system fail, so as to ensure that developers can take timely measures to avoid the system hazard caused by software.

Consequently, system hazard analysis should be carried out before software hazard analysis, determine possible software factors which may cause system hazard, as the starting point of software hazard analysis. System hazard analysis needs to be completed in concept stage and by the person who is familiar with I&C application systems. Since then, software hazard analysis needs to be carried out to track the hazard items related to software identified in concept stage, and to ensure that these hazards are eliminated in the process of software design and implementation.

3.2 Software Hazard Analysis Technique

When initial hazard analysis is carried out, the systematic and structured analysis techniques should be selected for the comprehensive identification of hazard items of systems and software. The techniques, FTA [3] using top-down order and FMEA [3] using bottom-up order, are both suitable, one or both can be selected and put in practice.

The choice of whether to use software FTA technique or software FMEA technique should be according to the scale and characteristics of analyzed software and ensure that analyzed object could be completely organized from a perspective or a variety of combination of perspectives. Alternative perspectives include logical perspective, process perspective, physical perspective, development perspective and application perspective.

In addition, according to different features of software products of different stages, as well as software development and verification experience, abnormal conditions and events checklist could be established in advance as a supplement, to facilitate rapid positioning of potential hazard items. Checklist in Table 1 is suggested based on requirements from related standards [2–5] and software project experience.

The structural hazard analysis techniques can ensure hazard items can be fully identified as early as possible, and checklist analysis technique will take full advantage of existing experience, combination of these two types of analysis technique can identify hazards rapidly and comprehensively.

So it is suggested to use FTA and/or FMEA as the main analysis technique and checklist as complementarity.

3.2.1 Hazard Analysis in Software Development Process

Safety critical functions in NPP I&C systems include performing signal acquisition, data transmission, logic processing, control and display outputs, etc.

Hazard analysis in concept stage should cover the potential hazards brought by external factors and from the system itself for implementation above safety critical functions. This task includes identifying potential hazards in the system, assessing the severity and possibility of occurrence of each hazard, and identifying mitigation strategies for each hazard.

The requirements stage should determine software contribution to the system hazards, identify software key requirements related to system hazard items, and confirm whether processing, control or mitigation measures are exist in software for these hazards.

The hazard analysis in design stage and implementation stage should verify the logic and the relevant data elements to design and implement the key needs correctly and not introduce new hazards.

The hazard analysis in test stage should focus on if all identified hazards have been eliminated by proper measures and correct implementation, no new hazards been introduced.

Table 1. Suggested checklist

Stage	Checked items
Concept	(a) Functions related potential initial events of nuclear power plant; (b) Safety critical functions of the I&C system itself; (c) Running environment restrictions; (d) Potential failures caused by hardware; (e) Potential software failures caused by hardware failure; (f) Potential failures caused by operators.
Requirement	(a) Missing or inconsistent software functional execution sequence requirements; (b) Wrong requirements of mismatch of data structure and function; (c) Unreasonable demand about software execution time, I/O transmission rate, memory/storage space allocation and other aspects.
Design	(a) Logic error; (b) Data structure error; (c) Data usage and processing errors, including: data initialization, data access, data range, units and dimensions, etc. (d) Interface errors, including inconsistency between this software and other software, between this software and hardware; (e) Timing errors; (f) Extreme cases that is ignored; (g) Design contents not consistent with software requirements.
Implementation	(a) Logic error; (b) Data definition, usage and processing errors, including: data initialization, data access, data range, units and dimensions, etc. (c) Interface errors, including inconsistent parameter passing, data size, unit, byte order, bit sequence, etc. (d) Code coverage: It should be analyzed that 100% coverage cannot be achieved despite of execution of all test cases; (e) Whether actual software execution time, I/O transmission rate, memory/storage space allocation are satisfied with software requirements.
Testing	(a) Whether software safety critical functions are properly implemented; (b) Whether software performance meets system requirements; (c) Whether interface with other software operates as expected.

3.2.2 Hazard Analysis Procedure

With FTA technique and application perspective, the procedure in Fig. 2 is performed during each stage of software development process. Explanations are given for some nodes of this procedure.

(1) Determining system function module and data flow

If FTA does not exist, V&V staff should analyze the function module and data flow based on the requirements and design documents.

- Function module analysis: the function module should be identified and partitioned based on the requirements and design documents, module name and its function should be recorded.

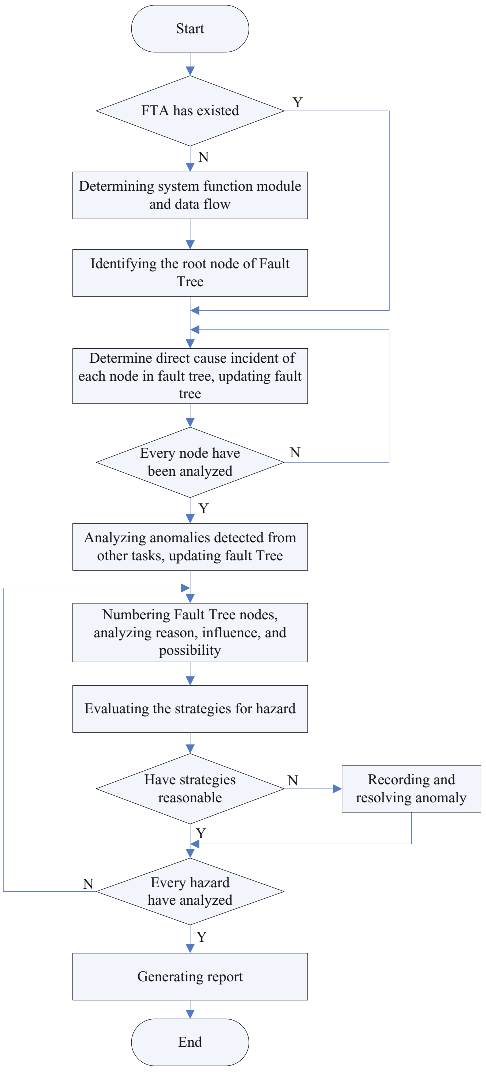
Start
FTA has existed
Y
N
Determining system function module and data flow
Identifying the root node of Fault Tree
Determine direct cause incident of each node in fault tree, updating fault tree
Every node have been analyzed
N
Y
Analyzing anomalies detected from other tasks, updating fault Tree
Numbering Fault Tree nodes, analyzing reason, influence, and possibility
Evaluating the strategies for hazard
Have strategies reasonable
N
Recording and resolving anomaly
Y
N
Every hazard have analyzed
Y
Generating report
End

Fig. 2. Hazard Analysis procedure using FTA technique

- Data flow analysis: with external entities as the inputs and outputs of data flow, input interfaces and output interfaces of data flow, data flow paths, data transferring between function modules, and its function should be recorded.

(2) Identifying the root node of Fault Tree

The root node of Fault Tree should be determined firstly. One unacceptable failure of system safety function should be selected as the root node of Fault Tree, and perhaps more than one Fault Tree are formed.

(3) Determine direct cause event of each node in fault tree, updating fault tree

Direct cause to all the nodes in a fault tree should be analyzed in turn. For each node, analyze its direct causes according to development documents, the depth of analysis should be consistent with detailed level of current development documents. If one analysis result is not identified in fault tree, add it as a new node.

(4) Numbering Fault Tree nodes, analyzing reason, influence, and possibility

Each fault Tree node shall have a unique number, and be recorded on the Fault Tree. Number shall accord to the following:

- Root node is the first level.
- The number of various level is separated with the ".".
- For node in the same level of Fault Tree, the serial number begins from 001.
- The number of each node shall include its father node number.

The reason, influence, and probability of fault for the bottom Fault Tree node should be analysed, the probability of its occurrence and severity level of its impact should be classified according to Table 2.

Table 2. Definition of probability and severity level of faults

Probability	Severity level			
	Fatal	Serious	General	Prompt
Frequent	1	1	2	3
Probable	1	1	2	3
Occasional	1	2	3	4
Infrequent	2	3	3	4
Impossible	3	3	3	4

4 Safety Testing

Through software hazard analysis, we can find the potential software hazard factors as early as possible, and provide the basis for the developer taking measures as soon as possible. Yet analysis is only a static verification means, which cannot replace software dynamic testing. Unless testing which is realized by software actual operation is performed, it cannot be confirmed whether software safety is finally realized.

Based on software safety analysis of the operational profile, the purpose of safety testing is to select and simulate operating environments which impact software safety to find out the potential faults, collect sufficient data for software safety assessment and confirm no defects affect actual software safe operation exist. Safety testing is different from routine function and performance testing whose purpose is to ensure that all software requirements are implemented, which should be a prerequisite for safety testing, and software safety testing is of no significance without achievement of software requirements.

Based on combination of software operational profile and test input coverage, completion of safety testing selects proper test cases. The selected test cases are more close to actual software safety requirement, more conducive to identification and elimination of major faults. In addition, software safety testing should be fully covered with a variety of input conditions and their combination, including the following: legal domain, illegal domain, boundary value and variable input value combination.

The procedure of safety testing is as follows:

(1) First of all, safety testing needs to start with determination of software safety critical functions, and to determine software safe operation profile. This step is the main difference between the safety testing and routine testing, and the content can refer to the result of hazard analysis process.
(2) Secondly, safety test cases which are able to fully cover various input conditions are designed.
(3) Then, according to test cases, test environments are set up, tests are executed, and testing process, all phenomenon and results are recorded in detail.
(4) Finally, based on analysis of the test results, software safety is evaluated.

5 Application

This above implementation method has been applied in one digital I&C System platform which is developed based on microprocessor, although this method is not restricted to any specific technology.

In this project, each design and test item in all documents including system specification, software specification, design description, test designs, test cases and test procedures, was assigned with one unique number which was used to build up the tracking relationship between each other in sequence of development activities. One commercial software tool, namely Rational RequisitePro, was used to output the final tracking matrix which can exhibit tracking connections directly and clearly. Based on these matrixes, V&V staff evaluated accuracy, consistency, completeness and accuracy of each tracking pair to conclude whether the documentation achieve traceability. If not, anomaly would be provided to producer of the documentation. Therefore, strictly speaking, traceability analysis is not software safety verification, yet it is one integral part of software safety verification, just because software safety verification is impossible without excellent traceability.

For design items which achieved traceability, hazard analysis was the next step of safety verification. For a digital I&C system platform, its hazards may come from any dissatisfaction of functional and nonfunctional requirements, so an assumed system which comprises all typical components of this platform was verified. FTA technique was applied in hazard analysis, and the top node of the fault tree was malfunction of this assumed system including unexpected trip and no-trip as required for reactor protection system from application perspective, going through the control and information flow all tree nodes were analyzed and all potential hazards were identified, the bottom-nodes of the fault tree were all possible failures of each typical component relevant to the top-node. The failure-safe criteria was applicable to hazard analysis, that is, all failures of each component were controlled and result in system safety finally. Abnormal conditions and events checklist of each stage were also applied, each component was checked by each check item. The hazards identified in previous stage were verified whether any control or mitigation measure was taken in current stage and whether any change in risk rating if occurred.

With the help of hazard analysis of each stage, all identified potential hazard factors with or without mitigation measures were considered in construction of safety testing operational profile and design of test cases. Possible input combinations were used in test cases to ensure no situation of breaking failure-safe criteria exists. The effect of measures and actual operational result were verified, satisfaction of safety requirements was validated.

6 Summaries

Software safety V&V tasks which cover traceability analysis, hazard analysis and safety testing are needed to verify the output results of different stages in the process of software development; they are mutual support between each other, so as to ensure that software safety is verified and validated comprehensively. Software safety V&V tasks have been applied and proved effective in one digital I&C system platform development project. They have played an important role in improving software safety.

With further application and more experience gained, more appropriate safety V&V techniques, such as probability safety analysis techniques, will be tried, and these three tasks will become more applicable for software V&V of digital I&C system used in nuclear power plants.

References

1. The Institute of Electrical and Electronics Engineers, Inc.: IEEE std 1012 IEEE Standard for Software Verification and Validation. The Institute of Electrical and Electronics Engineers, Inc., New York (2004)
2. International Electro Technical Commission: CEI/IEC 61508-7 Functional safety of electrical/electronic/programmable electronic safety-related systems. International Electro Technical Commission, Switzerland (2010)
3. The Institute of Electrical and Electronics Engineers, Inc.: IEEE std 7-4.3.2 IEEE Standard for Digital Computers in Safety System of Nuclear Power Generating Stations. The Institute of Electrical and Electronics Engineers, Inc., New York (2003)
4. The Institute of Electrical and Electronics Engineers, Inc.: IEEE std 1228 IEEE Standard for Software Safety Plan. The Institute of Electrical and Electronics Engineers, Inc., New York (1994)
5. International Electro Technical Commission: IEC 60880 Nuclear power plants-Instrumentation and control systems important to safety-Software aspects for computer-based systems performing category A functions. International Electro Technical Commission, Switzerland (2006)