Proceedings of the IASTED International Conference
SOFTWARE ENGINEERING
February 17-19, 2004, Innsbruck, Austria

AN INTEGRATED ENVIRONMENT OF S/W SPECIFICATION AND V&V FOR SAFETY-CRITICAL SYSTEMS

Seo Ryong Koo*, Poong Hyun Seong*, Junbeom Yoo**, and Sung Deok Cha**
Korea Advanced Institute of Science and Technology
* Department of Nuclear and Quantum Engineering
** Department of Electrical Engineering & Computer Science, Division of Computer Science
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea

Cheong Youn* and Hyun-chul Han**
*Chungnam National University, Department of Information and Communication
** CQCom Inc.
220 Gung-dong, Yuseong-gu, Daejeon 305-764, Korea

Abstract

As a digital system becomes more important in recent years, software specification and analysis techniques become a central problem in the safety-critical systems. Therefore, the importance of software verification and validation (V&V) based on an adequate specification is more emphasized in view of the software quality. For a thorough V&V, it should be performed throughout whole software life cycle. However, these kinds of works are very difficult to perform systematically because of manual-oriented tasks. This paper introduces various CASE tools to support the system specification for a formal based analysis according to the software life cycle. These tools are integrated through interface functions between each tool. Consequently, an integrated environment of S/W specification and V&V is proposed for safety-critical systems. Integrated environment consists of SIS-RT for concept phase, NuSRS for requirement phase, NuSDS for design phase, and NuSCM for configuration management. After further development efforts, our integrated environment is believed to turn out to be a unique and promising software development and analysis tool to support throughout whole life cycle.

Key Words

Software Tools, V&V, Specification, Safety-Critical System, Analysis, Configuration Management

1. Introduction

As a digital system has its advantages over an analog system, the use of digital systems is on increase in the safety-critical system in recent years. In the safety-critical systems such as Nuclear Power Plant (NPP) systems, it is required the very high confidence for software quality although it is infeasible to quantify software quality effectively. Recently, the concept of software verification and validation (V&V) is accepted as a way to assure the quality of new digitalized safety-critical systems [1]. And, the thorough V&V processes should be needed throughout the software development life cycle. As shown in Figure 1, life cycle consists of concept, requirements, design, implementation, and test phases. Each phase is obviously defined to separate the activities to be done in each phase. As shown in Figure 1, in IEEE Standard 1012 "Software Verification and Validation"[2], minimum V&V tasks for safety-critical systems are also defined along each phase. V&V tasks should be traceable back to software requirements and a critical software product should be understandable for independent evaluation and testing.

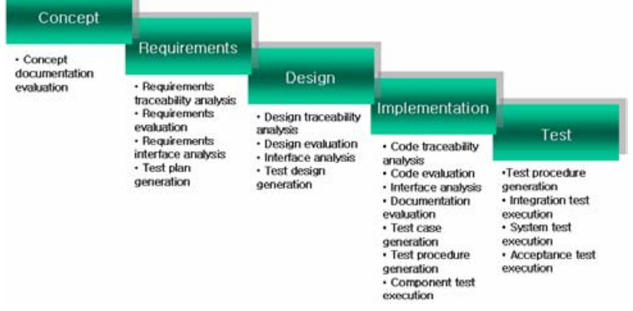

Concept
• Concept documentation evaluation
Requirements
• Requirements traceability analysis
• Requirements evaluation
• Requirements interface analysis
• Test plan generation
Design
• Design traceability analysis
• Design evaluation
• Interface analysis
• Test design generation
Implementation
• Code traceability analysis
• Code evaluation
• Interface analysis
• Documentation evaluation
• Test case generation
• Test procedure generation
• Component test execution
Test
•Test procedure generation
• Integration test execution
• System test execution
• Acceptance test execution

Figure 1. Software V&V Tasks along Life Cycle

All of life cycle phases should be evaluated for software quality attributes such as correctness, completeness, consistency, and traceability. Therefore, it is the most important to define an effective specification method for

each software development phase. The effective specification would be absolutely helpful for the verification and validation during whole life cycle.
In this work, an integrated environment of S/W specification and V&V is proposed according to whole life cycle for safety-critical systems. Generally, software engineering environments are comprised of all the engineering tasks for software life cycle. Through integrated collections of CASE tools, software engineering environments have the greatest impact on software quality and engineer productivity. In safety-critical software fields, especially in NPP systems, a systematic V&V technique dose not exist yet due to lack of an adequate software specification throughout life cycle. Therefore, it was needed to develop an integrated environment of S/W specification and V&V. In this paper, we propose an integrated environment which consists of various CASE tools; SIS-RT for concept phase, NuSRS for requirement phase, NuSDS for design phase, NuSCM for configuration management. Our integrated environment supports not only system specifications for software development but also various kinds of V&V activities such as software inspection, traceability analysis, formal analysis and configuration management. Figure 2 shows overall scheme for an integrated environment in this work. Each CASE toolset supports each phase of S/W development life cycle and S/W V&V simultaneously. Through some special features for interface, CASE tools could be integrated in a straight forward manner. In this work, an integration and coordination of the CASE tools is one of the important features. Consequently, it could be achieved to perform a specific system specification technique throughout life cycle and an effective V&V process for safety-critical systems. This paper introduces these CASE tools within our integrated environment.

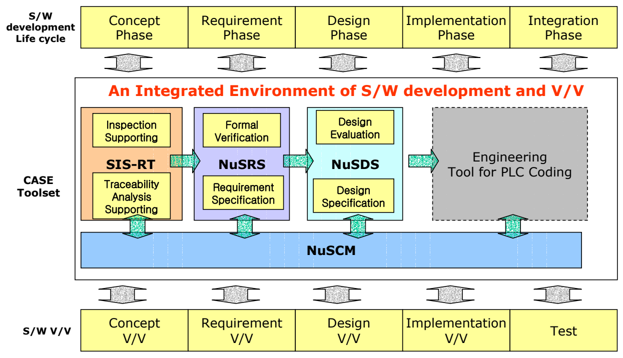


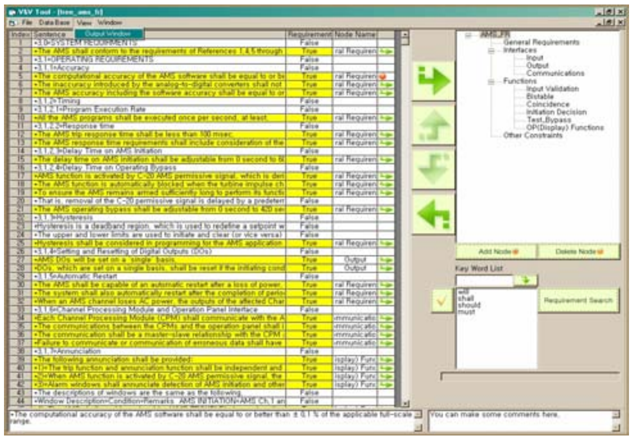
Figure 2. Overall Scheme for an Integrated Environment

2. SIS-RT

In an integrated environment, SIS-RT supports a concept phase or whole S/W life cycle phases based on documents. SIS-RT stands for Software Inspection [3] Supporting and Requirement Traceability tool. Inspection is widely believed to be an effective software V&V technique. It can provide a great increase in both productivity and product quality by reducing development time and by removing more defects than is possible without using inspection. Inspection could be applied to the whole life cycle. In SIS-RT, we integrate requirement traceability analysis capability into the software inspection support tool because requirement traceability analysis is considered as one of the items of software inspection. Additionally, formal requirement analysis and inspection meeting support capability are integrated in SIS-RT. Therefore, SIS-RT consists of a document analysis feature, a traceability analysis feature, a formal analysis feature and an inspection meeting supporting feature. SIS-RT is designed to support inspection of all software development products based on documents. In addition, SIS-RT is a PC-based application designed for use by anyone who needs to manage requirements. SIS-RT has three kinds of views; Inspection View, Traceability View, and Structure View. And there is a web page for inspection meeting in SIS-RT, but we will skip an introduction to the web page in this paper.

2.1 Inspection View

The support of document analysis with Inspection View is a main function of SIS-RT. It supports an extraction function that reads a text file and copies paragraph numbers and requirement text to a SIS-RT file. It can read any text data that is convertible to '.txt' format.
Inspection View permits users to associate database items by defining attributes; attributes attached to individual database items provide a powerful means to identify subcategories or database items and manage requirements.



Index	Sentence	Requirement	Node Name
1	•3.0•SYSTEM REQUIREMENTS	False	
2	•The AMS shall conform to the requirements of References 1,4,5 through	True	ral Requiren
3	•3.1•OPERATING REQUIREMENTS	False	
4	•3.1.1•Accuracy	False	
5	•The computational accuracy of the AMS software shall be equal to or be	True	ral Requiren
6	•The inaccuracy introduced by the analog-to-digital converters shall not	True	ral Requiren
7	•The AMS accuracy including the software accuracy shall be equal to or	True	ral Requiren
8	•3.1.2•Timing	False	
9	•3.1.2.1•Program Execution Rate	False	
10	•All the AMS programs shall be executed once per second, at least.	True	ral Requiren
11	•3.1.2.2•Response time	False	
12	•The AMS trip response time shall be less than 300 msec.	True	ral Requiren
13	•The AMS response time requirements shall include consideration of the	True	ral Requiren
14	•3.1.2.3•Delay Time on AMS Initiation	False	
15	•The delay time on AMS initiation shall be adjustable from 0 second to 60	True	ral Requiren
16	•3.1.2.4•Delay Time on Operating Bypass	False	
17	•AMS function is activated by C-20 AMS permissive signal, which is deri	True	ral Requiren
18	•The AMS function is automatically blocked when the turbine impulse ch	True	ral Requiren
19	•To ensure the AMS remains armed sufficiently long to perform its functi	True	ral Requiren
20	•That is, removal of the C-20 permissive signal is delayed by a predeterr	False	
21	•The AMS operating bypass shall be adjustable from 0 second to 420 sec	True	ral Requiren
22	•3.1.3•Hysteresis	False	
23	•Hysteresis is a deadband region, which is used to redefine a setpoint w	False	
24	•The upper and lower limits are used to initiate and clear (or vice versa)	False	
25	•Hysteresis shall be considered in programming for the AMS application.	True	ral Requiren
26	•3.1.4•Setting and Resetting of Digital Outputs (DOs)	False	
27	•AMS DOs will be set on a 'single' basis.	True	Output
28	•DOs, which are set on a single basis, shall be reset if the initiating cond	True	Output
29	•3.1.5•Automatic Restart	False	
30	•The AMS shall be capable of an automatic restart after a loss of power.	True	ral Requiren
31	•The system shall also automatically restart after the completion of perio	True	ral Requiren
32	•When an AMS channel loses AC power, the outputs of the affected Char	True	ral Requiren
33	•3.1.6•Channel Processing Module and Operation Panel Interface	False	
34	•Each Channel Processing Module (CPM) shall communicate with the A	True	mmunicatio
35	•The communications between the CPMs and the operation panel shall i	True	mmunicatio
36	•The communication shall be a master-slave relationship with the CPM	True	mmunicatio
37	•Failure to communicate or communication of erroneous data shall have	True	mmunicatio
38	•3.1.7•Annunciation	False	
39	•The following annunciation shall be provided:	True	isplay) Func
40	•1)•The trip function and annunciation function shall be independent and	True	isplay) Func
41	•2)•When AMS function is activated by C-20 AMS permissive signal, the	True	isplay) Func
42	•3)•Alarm windows shall annunciate detection of AMS initiation and other	True	isplay) Func
43	•The descriptions of windows are the same as the following.	False	
44	•Window Description•Condition•Remarks AMS INITIATION•AMS Ch.1 an	False	

•The computational accuracy of the AMS software shall be equal to or better than ± 0.1 % of the applicable full-scale range.

will
shall
should
must

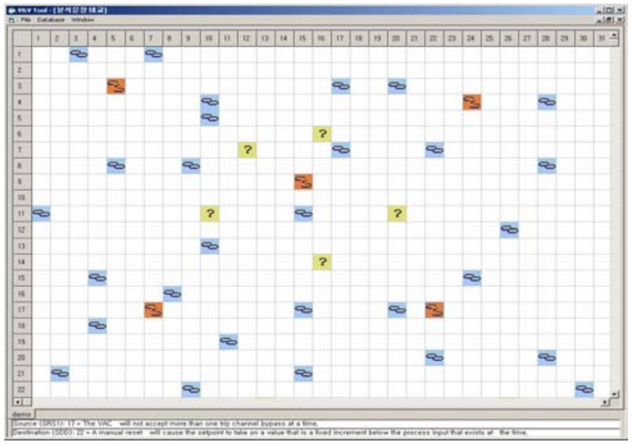
You can make some comments here.

Figure 3. Inspection View of SIS-RT

Figure 3 shows a screen shot of the Inspection View of SIS-RT. Inspection View reads source document, identifies requirement, and extracts them for importing into the database. Inspection View automatically finds and extracts requirements based on a set of keywords defined by the user. As requirements are found, they are highlighted as shown in Figure 3. The user may also manually select and identify requirements. Inspection View enables us to produce a user-defined report that shows various types of inspection results. Through the right-hand side window shown in Figure 3, user can compose checklists for systematic inspection and then SIS-RT can directly support the software inspection with this functional window. Requirements to be found by the tool are located in suitable checklist site using various arrow buttons in the checklist window. In this way, each inspector examines requirements and generates the inspection result documents.

2.2 Traceability View

Traceability View of SIS-RT supports the requirement traceability analysis as shown in Figure 4. Traceability View provides mechanisms to easily establish and analyze traceability through the real-time visual notification of change. This capability allows users to pinpoint its impact across the project and assess coverage for verification and validation. Through the Traceability View, it is possible to analyze traceability between source documents and destination documents. Traceability View of SIS-RT supports normal parent/child links to manage requirements. Furthermore, it supports peer links between items in the database and general documents to provide an audit trail showing compliance to quality standards or contractual conditions.



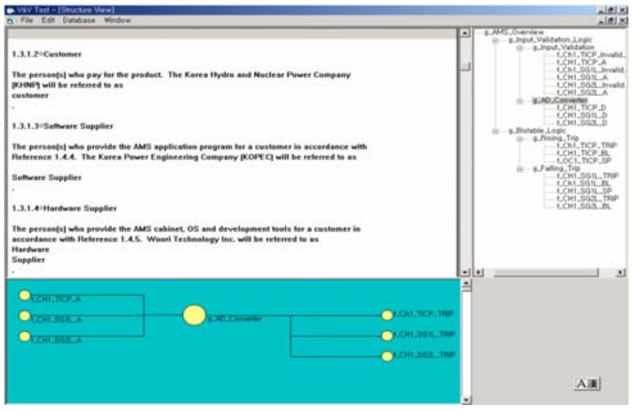
Source (SRS1): 17 = The VAC will not accept more than one trip channel bypass at a time.
Destination (SDD): 22 = A manual reset will cause the setpoint to take on a value that is a fixed increment below the process input that exists at the time.

Figure 4. Traceability View of SIS-RT

As shown in Figure 4, the column number represents a requirement of source document and the row number represents destination document. The relationships between source and destination are expressed through a matrix window with linked and unlinked chain. The linked chains mean that source requirements are reflected into destination requirements. The unlinked chains represent that source and destination requirements are changed, thus it is necessary to verify the change between source and destination documents. The question marks mean that it is difficult to define traceability between requirements. At this time, it is necessary to verify requirements by other analyzer. In order to more easily support traceability analysis, Traceability View has an additional function to calculate the similarity between requirements using a similarity calculation algorithm [4]. Through this function, Traceability View can automatically represent the similarity by percentage and then this similarity result is very helpful to user and analyzer. Now, we proposed algorithms to calculate the similarity for both English and Korean documents. In this way, traceability analysis between documents could be performed through the Traceability View.

2.3 Structure View

As an interface function in our integrated environment, Structure View of SIS-RT enables the effective transition into NuSRS. Figure 5 shows a screen shot of Structure View of SIS-RT. Through the Structure View, we can analyze system development documents in view of system's structure and then these analysis results help us generate a formal specification from a natural language document in requirement phase. In the structural analysis of systems through the Structure View, it is the most important to define inputs/outputs and functions. Therefore, we proposed Input-Process-Output structure type in this work. In the Structure View, several tabular forms help users easily build up Input-Process-Output structure and Input-Process-Output structure is represented in right-hand side window as a tree type. After structure analysis, Structure View generates a result file written in XML language and then it is transferred to NuSRS. With this file, FOD could be drawn automatically in the NuSRS.



1.3.1.2 Customer

The person(s) who pay for the product. The Korea Hydro and Nuclear Power Company (KHNP) will be referred to as customer

1.3.1.3 Software Supplier

The person(s) who provide the AMS application program for a customer in accordance with Reference 1.4.4. The Korea Power Engineering Company (KOPEC) will be referred to as

Software Supplier

1.3.1.4 Hardware Supplier

The person(s) who provide the AMS cabinet, OS and development tools for a customer in accordance with Reference 1.4.5. Woori Technology Inc. will be referred to as Hardware Supplier

Figure 5. Structure View of SIS-RT

3. NuSRS

Though formal methods, such as Statechart [5], CPN [6], RSML [7], and SCR [8], are also considered as effective V&V harness, they are not easy to be used properly in safety-critical systems because of their mathematical nature. However, formal specification can lessen requirements errors by reducing ambiguity and imprecision and by clarifying instances of inconsistency and incompleteness.

The Atomic Energy of Canada Limited (AECL) approach specifies a methodology and format for the specification of software requirements for safety critical software used in real-time control and monitoring systems in nuclear systems. It is a SCR-style SRS verification method based on Parnas' four variable method. A system reads environment states through monitored variables that are transformed into input variables. The output values of the output variables are calculated and are changed into control variables. The AECL provides two different views of the requirements. A larger view is the FOD and each of the function in it is described by the smaller view of the SDT. The AECL approach specifies all requirements of the nuclear control system in the FOD and SDT notations. This is somewhat complex in cases where timing requirements and history related requirements are considered. This difficulty of specification is modified in the NuSCR approach.

The NuSCR approach is an extended formal verification method of the existing SCR-style AECL approach [9]. The NuSCR specification language was originally designed to simplify the complex specification techniques of certain requirements in the AECL approach. It is an improved method in describing behavior of the history related requirements and timing requirements of the nuclear control system by specifying them in automata and timed-automata respectively. In the existing AECL method, all specifications including history related requirements and timing requirements are specified with only one type of function node in the FOD and with SDT tables. However, the NuSCR uses three different types of nodes in the FOD to specify the properties derived from the requirements. The types consist of nodes that specify history related requirements that are described in automata [10], timing requirements that are described in timed-automata [11], and nodes that specify all other requirements exclusive of the previous two types of functional requirements.

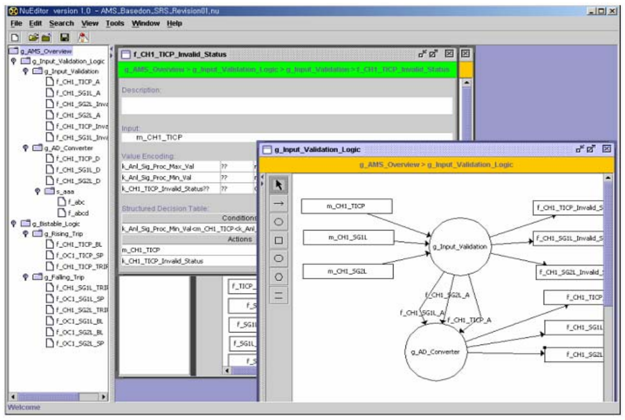


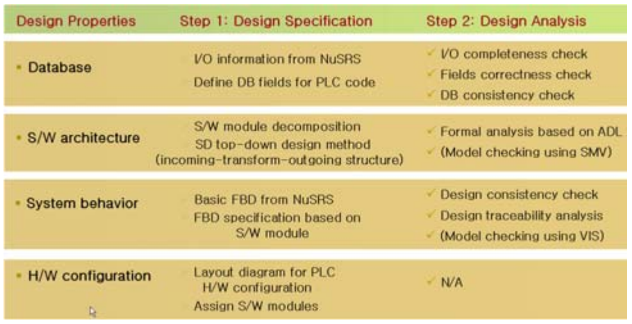
Figure 6. NuSRS

NuSRS is an editor for requirement specification based on NuSCR approach. Figure 6 shows a screen shot of NuSRS. The NuSRS is a platform independent tool made with JAVA for formally specifying the SRS of the nuclear control system. It provides environment to draw FOD and SDT and allows automata diagrams to be built from the nodes of the FOD. The Editor also gives a hierarchical view of the SRS described as can be seen on the left side of Figure 6. Additionally, NuSRS also generates a result file written in XML language that includes all of information in NuSRS and then it is transferred to NuSDS.

4. NuSDS

Among software life cycle, software design is a process of translating problem requirements into software structures that are to be built in the implementation phase [12]. In general industry, a Software Design Specification (SDS) should be produced at this software design phase. It describes overall system architecture and contains a definition of the control structure model. It should be evaluated for software quality attributes such as correctness, completeness, consistency, and traceability. Therefore, it is the most important to define an effective specification method for the software design phase. The effective specification would be absolutely helpful for the design verification and validation. Also, a well-formed design specification is very useful for the coding in implementation phase because a design phase is just previous stage of the implementation phase in life cycle. Therefore, an implementation product such as code should be easily translated from the design specification. In NPP software fields, Programmable Logic Controller (PLC) [13] is widely used for the safety-critical systems. However, a systematic design specification and analysis technique for the implementation based on PLC does not exist yet.

In this paper, a software design specification and analysis technique for the safety critical software based on PLC is proposed. Now, for the tool supporting, we are developing NuSDS based on the proposed techniques which is the tool, especially for the software design specification in nuclear fields. SDS is a description to show how to create a design which accurately and completely satisfies the behavior and constraints in the SRS. During the coding phase of the software life cycle it is then a relatively simple matter to transform the design into sequences of executable statements written in a particular computer language. Therefore, in this work, an adequate specification technique is needed for the systematic verification and easily translating into implementation phase. Because the SDS can be considered as a blueprint when we build a house, we should have well-formed design features for the right specification. NuSDS supports the design specification features for generating the SDS of nuclear systems. It consists of four major specifications; Database, Software Architecture, System Behavior, and PLC Hardware Configuration. SDS could be generated using these four major specifications in NuSDS.



Design Properties	Step 1: Design Specification	Step 2: Design Analysis
· Database	I/O information from NuSRS Define DB fields for PLC code	✓ I/O completeness check ✓ Fields correctness check ✓ DB consistency check
· S/W architecture	S/W module decomposition SD top-down design method (incoming-transform-outgoing structure)	✓ Formal analysis based on ADL ✓ (Model checking using SMV)
· System behavior	Basic FBD from NuSRS FBD specification based on S/W module	✓ Design consistency check ✓ Design traceability analysis ✓ (Model checking using VIS)
· H/W configuration	Layout diagram for PLC H/W configuration Assign S/W modules	✓ N/A

Figure 7. Special Features of NuSDS

Figure 7 shows special features of NuSDS. NuSDS can be divided into two steps. In step 1, NuSDS fully supports design specification along the software design specification technique proposed in this work. And then, based on these design specification, NuSDS partially supports design analysis. It means that NuSDS can support translating into input language for model checking and help to connect to other V&V tools in step 2. Now, the development of NuSDS step 1 is finished and NuSDS step 2 will be added when it will be required for design analysis. Figure 8 shows a simple scratch of NuSDS. NuSDS consists of tree-like information window about input/output and function decomposition, software architecture window, FBD-style specification window, layout diagram for PLC hardware configuration, and database window.

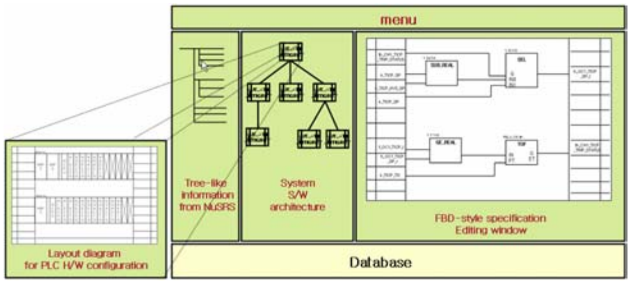
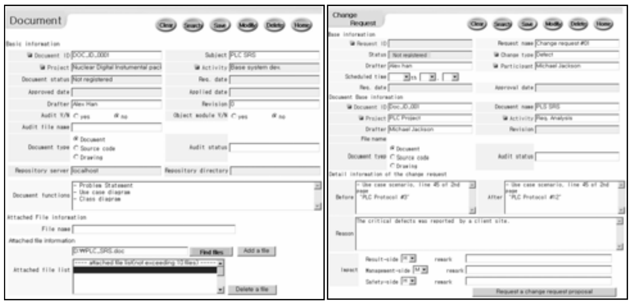

menu
Tree-like information from NuSRS
System S/W architecture
FBD-style specification Editing window
Layout diagram for PLC H/W configuration
Database

Figure 8. NuSDS

5. NuSCM

The Software Configuration Management (SCM) is an activity, which configures the form of a system (documents, programs, and hardware) and systematically manages and controls modifications used to compile plans, development and maintenance. Actually, many kinds of documents for system development and V&V process are produced during software life cycle in safety-critical systems. In guaranteeing high quality in the software development phase and producing reliable products, it is important to control and govern documents. Software quality management should be seriously valued in both the development phase as well as in the modification and maintenance phase. Even while operating the software, requests in modification continue to be received, so in order to confront these requests specific corresponding plans should be established. If modification requests are not properly processed in the software maintenance phase, this will result in deterioration in quality and declination in the life of the software. Especially in systems where safety is seriously valued, the chance of accidents due to the software may increase. Because of this, when recent research institutes and companies carry out projects, they are making attempts to automate systematic management of various documents containing information which will satisfy high quality and reliability.

NuSCM is a project-centered software configuration management system especially for nuclear safety system. In our integrated environment, NuSCM supports to manage all of system development documents, V&V documents, and codes systematically throughout whole life cycle. Additionally, for the interface between NuSCM and other tools, NuSCM manages all of result files produced from SIS-RT, NuSRS, and NuSDS. Recently, since most software systems are compatible regardless of location and users are able to easily approach it, they are being developed based on the advantageous Web (World Wide Web). NuSCM was also designed and embodied using the web. Figure 9 shows a document management view and a change request view in NuSCM.



Basic information
Document ID: DOC_ID_0001
Subject: PLC SRS
Project: Nuclear Digital Instrumental pack
Activity: Base system dev.
Document status: Not registered
Res. date:
Approved date:
Applied date:
Drafter: Alex Han
Revision: 0
Audit Y/N: yes [✓ no]
Object module Y/N: yes [✓ no]
Audit file name:
Document type: [✓ Document] Source code Drawing
Audit status:
Repository server: localhost
Repository directory:
Document functions: - Problem Statement - Use case diagram - Class diagram
Attached File information
File name:
Attached file information
D:\PLC_SRS.doc
Find files
Add a file
Attached file list: ----- attached file list(not exceeding 10 files) -----
Delete a file

Base information
Request ID:
Request name: Change request #01
Status: Not registered
Change type: Defect
Drafter: Alex han
Participant: Michael Jackson
Scheduled time:
Req. date:
Approval date:
Document Base information
Document ID: Doc_ID_001
Document name: PLS SRS
Project: PLC Project
Activity: Req. Analysis
Drafter: Michael Jackson
Revision:
File name
Document type: [✓ Document] Source code Drawing
Audit status:
Detail information of the change request
Before: - Use case scenario, line 45 of 2nd page "PLC Protocol #3"
After: - Use case scenario, line 45 of 2nd page "PLC Protocol #12"
Reason: The critical defects was reported by a client site.
Impact: Result-side H remark
Management-side M remark
Safety-side H remark
Request a change request proposal

Figure 9. NuSCM

6. Conclusion

In this paper, an integrated environment of S/W specification and V&V for safety-critical systems was proposed. Integrated environment systematically supports formal based specification technique according to life cycle and also supports various kinds of effective V&V techniques based on proposed specification specifically for nuclear fields. For the tool supporting, SIS-RT, NuSRS, NuSDS, and NuSCM which are CASE tools for each life cycle phase were developed. SIS-RT is a special tool for software inspection and traceability analysis and thus it can be used in concept phase or all of documents based phases. For the specific nuclear software fields, NuSRS and NuSDS support to generate documents such as SRS in requirement phase and SDS in design phase respectively. Also they support a formal based analysis. NuSCM is a project-centered software configuration management tool. NuSCM manages and controls the modification of all of system development and V&V documents. And all result files from SIS-RT, NuSRS, and NuSDS could be managed through NuSCM. In order to combine various tools gracefully, this paper considered the interface functions between each tool as one of the important features in our integrated environment. In Table 1, each tool in the environment is summarized according to three point of views; S/W development life cycle supporting, main functions, and special advantages. Consequently, after further development efforts, our integrated environment is believed to be a unique and promising software development and analysis tool to support throughout whole life cycle.

	S/W development life cycle	Main functions	Advantages
SIS-RT	▪System concept phase ▪Whole phases based on documents	▪Documents inspection supporting ▪Documents traceability analysis supporting ▪System structure analysis supporting	▪Systematic checklist management ▪Reducing time of inspection work ▪Minimize human error ▪Effective traceability analysis ▪Interface with NuSRS
NuSRS	▪Software requirement phase	▪Formal method (NuSCR) editing supporting ▪Theorem proving (PVS) supporting ▪Model checking (NuSMV) supporting	▪Formal method for nuclear fields ▪Effective system formal specification ▪Formal requirement analysis ▪Interface with NuSDS
NuSDS	▪Software design phase	▪System Database, S/W Architecture, S/W Behavior, H/W Configuration specification supporting ▪Model checking supporting ▪Traceability analysis supporting	▪Optimal design technique for nuclear fields ▪Effective system design specification ▪Easiness of PLC programming ▪Formal design analysis
NuSCM	▪Whole phases	▪Project centered configuration management supporting ▪Change request form in nuclear fields supporting ▪Source code management supporting	▪CM technique for nuclear fields ▪Various documents style supporting ▪Interface with V/V tools

Table 1. Summary of Each Tool

7. Acknowledgement

This research was supported by KNICS (Korea Nuclear I&C System) R&D center and NRL project.

References

[1] EPRI, "Handbook for verification and validation of digital systems Vol.1: Summary", *EPRI TR-103291*, Vol.1, 1994

[2] IEEE, "IEEE Standard for Software Verification and Validation", *An American National Standard*, 1998

[3] M.E. Fagan, “Design and Code Inspections to Reduce Errors in Program Development,” *IBM system Journal*, Vol. 15, No. 3, pp. 182-211, 1976.

[4] Yeong-Jae, Yoo, “Development of a Traceability Analysis Method based on Case Grammar for NPP Requirement Documents written in Korean Language”, *M.S. Thesis*, Department of Nuclear and Quantum Engineering, KAIST, 2003

[5] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, pp.231-274, 1987.

[6] Kurt Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use Volume 1* (Springer-Verlag Berlin Heidelberg, 1997).

[7] N.G. Leveson, M.P.E. Heimdahl, H. Hildreth, and J.D. Reese, "Requirements Specification for Process-Control Systems," *IEEE Transaction on Software Engineering*, vol.20, no.9, Sept. 1994.

[8] C. Heitmeyer and B. Labaw, “Consistency Checking of SCR-style Requirements Specification”, *International Symposium on Requirements Engineering*, March, 1995.

[9] WolsongnNPP 2/3/4, “Software Work Practice Procedure for the Specification of SRS for Safety Critical Systems,” *Design Document no. 00-68000-SWP-002, Rev. 0*, Sept. 1991.

[10] J. Hopcroft and J. Ullman, *Introduction to Automata Theory* (Language and Computation, Addison-Wesley, 1979).

[11] R. Alur and David L. Dill, “A theory of Timed Automata,” *Theoretical Computer Science Vol. 126*, No. 2, pp. 183-236, April 1994.

[12] Roger S. Pressman, *Software Engineering: A practitioner's approach* (McGRAW-HILL Book Co, 2001).

[13] IEC, "IEC Standard 61131-3: Programmable controllers-Part 3", *IEC 61131*, 1993