NuSEE: AN INTEGRATED ENVIRONMENT OF SOFTWARE SPECIFICATION AND V&V FOR PLC BASED SAFETY-CRITICAL SYSTEMS

SEO RYONG KOO[1], POONG HYUN SEONG[1*], JUNBEOM YOO[2], SUNG DEOK CHA[2], CHEONG YOUN[3] and HYUN-CHUL HAN[4]
Korea Advanced Institute of Science and Technology
[1]Department of Nuclear and Quantum Engineering
[2]Department of Electrical Engineering & Computer Science, Division of Computer Science
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
[3]Chungnam National University, Department of Information and Communication
[4]CQCom Inc.
220 Gung-dong, Yuseong-gu, Daejeon, 305-764, Korea
[*]Corresponding author. E-mail : phseong@kaist.ac.kr

Received August 10, 2004
Accepted for Publication February 1, 2006

As the use of digital systems becomes more prevalent, adequate techniques for software specification and analysis have become increasingly important in nuclear power plant (NPP) safety-critical systems. Additionally, the importance of software verification and validation (V&V) based on adequate specification has received greater emphasis in view of improving software quality. For thorough V&V of safety-critical systems, V&V should be performed throughout the software lifecycle. However, systematic V&V is difficult as it involves many manual-oriented tasks. Tool support is needed in order to more conveniently perform software V&V. In response, we developed four kinds of computer aided software engineering (CASE) tools to support system specification for a formal-based analysis according to the software lifecycle. In this work, we achieved optimized integration of each tool. The toolset, NuSEE, is an integrated environment for software specification and V&V for PLC based safety-critical systems. In accordance with the software lifecycle, NuSEE consists of NuSISRT for the concept phase, NuSRS for the requirements phase, NuSDS for the design phase and NuSCM for configuration management. It is believed that after further development our integrated environment will be a unique and promising software specification and analysis toolset that will support the entire software lifecycle for the development of PLC based NPP safety-critical systems.

KEYWORDS : V&V, Software Specification, Safety-Critical System, Tool

1. INTRODUCTION

Digital systems offer numerous advantages over analog systems, and as such their use in safety-critical systems has greatly expanded in recent years. In safety-critical systems such as a Nuclear Power Plant (NPP), extremely high-confidence for software quality is demanded. However, effective quantification of software quality is infeasible. For more than a decade, the concept of software Verification and Validation (V&V) throughout the software development lifecycle has been accepted as a means to assure the quality of new digitalized safety-critical systems [1].

Generally, the software lifecycle consists of concept, requirements, design, implementation and test phases. Each phase is clearly defined to separate the activities to be conducted within it. As shown in Figure 1, in IEEE Standard 1012 for "Software Verification and Validation," minimum V&V tasks for safety-critical systems are also defined for each phase [2]. The V&V tasks should be traceable back to the software requirements, and a critical software product should be understandable for independent evaluation and testing. The products of all lifecycle phases should also be evaluated for software quality attributes such as correctness, completeness, consistency and traceability. Therefore, it is critical to define an effective specification method for each software development phase and V&V tasks are based on effective specifications during the whole software lifecycle.

In this work, an Integrated Environment (IE) approach for software specification and V&V was proposed in accordance with the above software V&V tasks during the entire

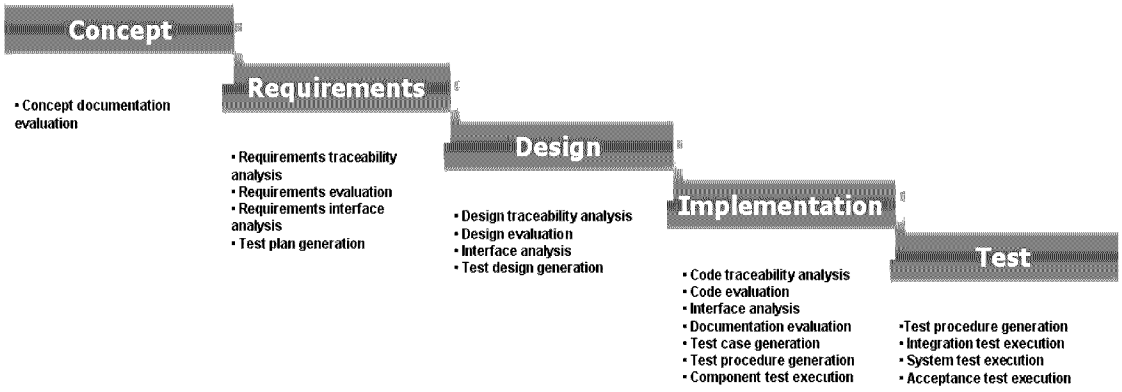


Fig. 1. Software V&V Tasks During the Lifecycle

software lifecycle for safety-critical systems. To implement this IE approach, we developed the NuSEE (Nuclear Software Engineering Environments) toolset to support and integrate the entire software lifecycle systematically. In safety-critical software fields, especially in NPP Instrumentation and Control (I&C) systems, a systematic and mature V&V technique does not exist due to the lack of an adequate software specification technique that operates throughout the lifecycle. In NPP software fields, there have been many attempts to use various specification and V&V techniques. Among the techniques, formal specification and analysis, software inspection, traceability analysis and configuration management are the most widely used in the development of safety-critical software of NPPs. However, most system specification and V&V tasks are extremely labor-intensive, and thus most users desire some kinds of tool support.

Accordingly, an integrated environment of software specification and V&V during the entire software lifecycle is needed. Thus far, nuclear engineers have depended on manual-oriented tasks for system documentation and V&V efforts, and thus they also strongly feel the necessity of integrated tool-support. In this work, the authors designed and developed the NuSEE toolset to support the entire software lifecycle for nuclear safety-critical systems. The NuSEE toolset achieves optimized integration of work products throughout the software lifecycle of safety-critical systems. The main features of NuSEE are effective documentation evaluation and management, formal specification and analysis and system configuration management. NuSEE consists of four major tools in accordance with the software lifecycle; NuSISRT (Nuclear Software Inspection Support and Requirements Traceability) for the concept phase, NuSRS (Nuclear Software Requirements Specification and analysis) for the requirement phase, NuSDS (Nuclear Software Design Specification and analysis) for the design phase and NuSCM (Nuclear Software Configuration Management) for configuration management. NuSEE is based on an IE approach and supports system specifications for software development as well as various kinds of V&V activities such as software inspection, traceability analysis, formal analysis and configuration management.

Figure 2 shows the overall features of NuSEE. Each CASE tool simultaneously supports each phase of the software development lifecycle and software V&V process. Through special features for the interface, CASE tools can be integrated in a straightforward manner. One of the important features of this work is integration and coordination of CASE tools. Also, using the NuSEE toolset, potential errors can be found at an early point throughout the software lifecycle, and thus nuclear engineers can fix them with the lowest cost and impact on system design. Nuclear engineers can thereby reduce the time and cost required for the development of software while user convenience is also enhanced. Consequently, NuSEE can achieve a specific system specification technique that is utilized throughout the lifecycle and an effective V&V process for safety-critical systems. NuSEE is a tool for building bridges between specialists in system engineering, software engineering and safety engineering. This paper introduces these CASE tools within our integrated environment.

2. RELATED WORKS

2.1 Integrated Environment (IE) Approach for NuSEE Toolset

For the development of the NuSEE toolset, we propose an integrated environment (IE) approach to support software development and V&V processes. An IE approach is utili-

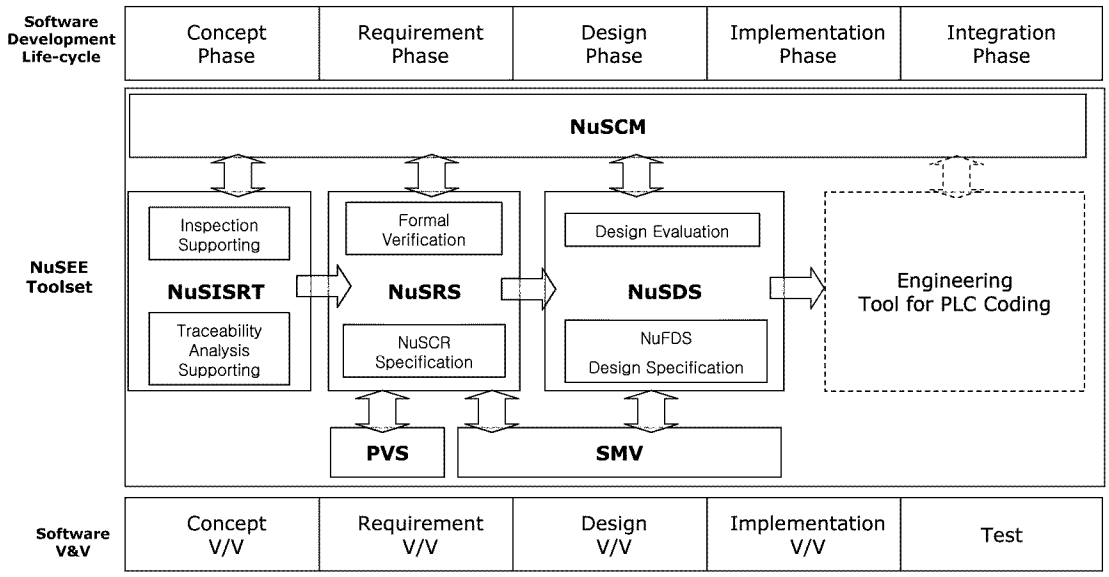


Fig. 2. Overall Features of NuSEE

zed for NPP safety-critical systems based on Programmable Logic Controller (PLC) software language. Currently, in NPP I&C systems, PLCs developed by various companies are being considered for safety-class hardware. Therefore, the IE approach focuses on the target systems implemented in the PLC hardware, and supports specific V&V processes for PLC software development. The major role of the IE approach is to achieve better integration between PLC software development and the V&V process for NPP safety-critical systems. In the IE approach, system specification based on formalism can be supported for checking V&V activities such as completeness, consistency and correctness. Furthermore, system specification of the IE approach focuses on ease of use and understanding. Thus far, most formal methods are so complex and hard to use that nuclear engineers have avoided using them despite their practical benefits.

The IE approach supports optimal V&V tasks for safetycritical systems based on PLC software language throughout the software lifecycle. For more efficient V&V, software requirements inspection, requirements traceability, and formal specification and analysis are integrated in this approach. Also, the IE approach allows project-centered configuration management. All documents and results in this approach can be systematically managed by using the function of configuration management.

Figure 3 shows the major features of the IE approach. It consists of document evaluation, requirements traceability, formal requirements specification and V&V, and effective design specification and V&V. In the document evaluation, document analysis based on a Fagan inspection [3] is supported throughout the software lifecycle. In the requirements traceability, sentence comparison based on the inspection results is supported by using the Requirements Traceability (RT) matrix. We also developed sentence similarity calculation algorithms for effective requirements traceability of documents written in both English and Korean [4]. Document evaluation and traceability analysis are major tasks of the concept and requirements phases. In the concept phase, which is the initial phase of a software development project, the needs of the user are described and evaluated through documentation. The requirements phase is the period in the software lifecycle when requirements such as the functional and performance capabilities of a software product are defined and documented. In the formal requirements specification and V&V, we developed the NuSCR (Nuclear Software Cost Reduction) approach [5], which is a formal method that is suitable for NPP systems. Using the NuSCR approach, formal requirements specification and analysis can be possible within the IE approach.

Finally, effective design specification and V&V are supported in the IE approach. We also developed the NuFDS (Nuclear FBD-style Design Specification) approach, which is a suitable design method for NPP systems [6,7]. In view of design analysis, NuFDS supports a design consistency check using ESDTs (Extended Structured Decision Tables), architecture analysis and model checking.

Among the software lifecycle phases, the software design phase is a process of translating software requirements into software structures that are to be built in the implementation phase. Hence, a well-formed design specification is very

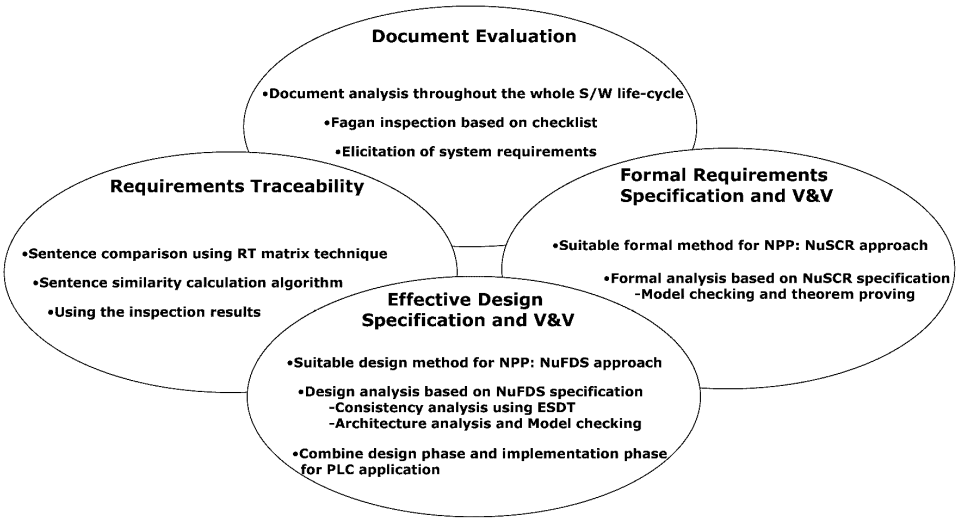


Fig. 3. Major Features of IE Approach

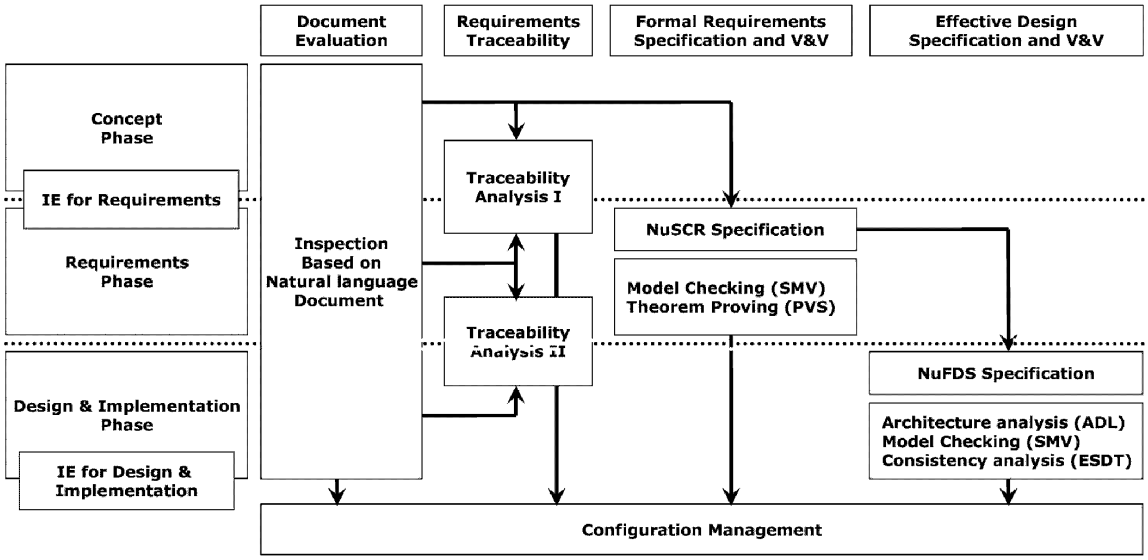


Fig. 4. Overall Scheme of IE Approach

useful for the coding step during the implementation phase. Therefore, an implementation product, such as code, should be easily translated from the design specifications. Among PLC software languages, the Function Block Diagram (FBD) is considered an efficient and intuitive language for the implementation phase. As a characteristic of PLC software, in particular, the boundary between design phase and implementation phase is not clear. This means that the level of design almost corresponds with implementation in the PLC software. Since the FBD language of a PLC is usually similar to a design feature in the design phase, it is necessary to combine the design phase with the implementation phase in developing a PLC-based system. Consequently, we can reduce the coding time and cost by

combining the design phase and the implementation phase, especially for a PLC application. The major contribution of the NuFDS approach is achieving better integration between the design and implementation phases for PLC applications.

In this work, the IE approach is the basis for development of the NuSEE toolset. This approach constitutes an adequate and effective technique for software development and V&V for the development of safety-critical systems based on PLC software language. In order to integrate the whole process of the software lifecycle, the function of the interface and flow-through of the process are the most important considerations in this work. Therefore, we focused on the sequence of processes between each feature in the IE approach. Figure 4 shows the overall scheme of the IE approach. The IE approach can be divided into two categories: IE for requirements [8], which is oriented in the requirements phase, and IE for design & implementation [6,7], which is oriented in the combined design and implementation phase. For efficient support of the IE approach, the NuSEE toolset was developed in this work. NuSEE consists of four CASE tools: NuSISRT, NuSRS, NuSDS and NuSCM.

2.2 Existing Tools

This section briefly describes the engineering tools for software development and analysis. Each tool has characteristics that are unique to the objectives and target systems. Because the NuSEE toolset is based on the integrated environment (IE) approach, the toolset is adequate for the development and analysis of safety-critical systems implemented in the PLC.

In industrial projects, the following tools were developed and widely used: Analyst Pro (developed by Goda Software Inc., Japan), MOBY/PLC (developed by the University of Oldenburg, Germany), SPACE (developed by Siemens, Germany), SCADE Suite (developed by Esterel Technology), Phapsody (developed by I-Logix), RequisitePro (developed by IBM Rational), SpecTRM (developed by Safeware Engineering Co.) and designsafe (Design Safety Engineering, Inc.). Table 1 compares specific features of the listed tools, namely the development methodology, the coverage of the software life cycle phases, the V&V activities (such as document evaluation, requirements traceability, model checking, safety analysis and configuration management), the possibility of simulation or code generation and the applicability to PLC-based systems.

The following describes each tool in greater detail.

2.2.1 Analyst Pro

"Analyst Pro is a complete management system designed to manage every aspect of your project," says its developer, Goda Software Inc. of Japan. "Analyst Pro helps you speedily make project decisions and meet changing client needs without loss of time, efficiency, or focus." The system supports requirements change management, requirements analysis and requirements traceability. As shown in Table 1, Analyst Pro focuses on the requirements phase. Therefore, in application to PLC-based NPP systems, AnalystPro can only be partially used to analyze requirements.

Table 1. Comparative Results of Existing Tools

	Development Methodology	S/W Development Phases				V&V				Simulation or Code generation	NPP Application (PLC)
		R	D	I	T	Document Evaluation	RT	Formal analysis	CM		
NuSEE	IE approach	O	O	O		O	O	various	O	FBD spec.	Yes
Analyst Pro	X	O				O	O	X	O	X	Partially
MOBY/PLC	X		O	O		X	X	model checking	X	PLC code (ST)	Yes
SPACE	X		O	O	O	X	X	X	X	C code	Yes
SCADE Suite	X		O	O		X	X	design verifier	O	both	partially
Rhapsody	UML	O	O	O	O	X	X	X	X	C code	Partially
RequisitePro	X	O				X	O	X	X	X	Partially
SpecTRM	Intent specification	O	O	O	O	Guideline to generate specification				Model simulation	No
designsafe	X	O	O	O	O	Hazard identification				X	No

2.2.2 MOBY/PLC

The University of Oldenburg in Germany developed a system called Modeling of Distributed Systems/PLC (MOBY/PLC). The system provides simulation and verification methods that can validate a design. Furthermore, it can translate designs into an executable source code for a real machine (PLC). MOBY/PLC supports the software design phase and the implementation phase, and it can be connected to the model checkers Kronos and UPPAAL. The most attractive characteristic of MOBY/PLC is its applicability to PLC-based systems, because it can directly generate a PLC code. Nonetheless, MOBY/PLC has weak V&V capability even though it supports connection to model checkers.

2.2.3 SPACE

The tool Specification and Coding Environment (SPACE) was developed by SIEMENS for the engineering and maintenance of TELEPERM XS digital safety instrumentation and control systems. SPACE provides tool-supported design verification and simulation-based software validation for TELEPERM XS (TXS) systems. SIEMENS is a well known PLC manufacturer and NPP vendor, and the target system of SPACE corresponds precisely with our NuSEE toolset, and consequently the two approaches share many common features. As shown in Table 1, although SPACE supports the software life cycle from the design phase to the testing phase, its V&V capability is poor. As a result, SPACE focuses solely on software development and code generation. Moreover, because SPACE was developed for practical use in industrial projects, its verification ability is inadequate with respect to the development of PLC-based NPP systems.

2.2.4 SCADE Suite

SCADE Suite was developed by Esterel Technologies, Inc.. In the avionics industry, SCADE Suite for Avionics is the de facto standard for the development of safety-critical embedded software. SCADE Suite supports system modeling, system simulation, and safety checks. In addition, it supports code generation and can be integrated with other tools such as Simulink, DOORS, and Altia. As shown in Table 1, SCADE Suite focuses on the design and implementation phase, and its most powerful features are code generation and simulation.

2.2.5 Rhapsody

Rhapsody was developed by I-Logix, a company that is well known for the Statemate toolset. Now the industry's leading model-driven development environment for systems, software and testing, Rhapsody is based on UML 2.0 and SysML. It has a unique ability to extend its modeling environment to enable functional and object-oriented design methodologies to run together in the same environment. However, as shown in Table 1, Rhapsody cannot efficiently support V&V activities but focuses on UML modeling support and C code generation.

2.2.6 RequisitePro

Developed by IBM Rational, RequisitePro is a requirements and use case management tool for project teams who want to improve the communication of project goals, enhance collaborative development, reduce project risk and increase the quality of applications before deployment. As shown in Table 1, RequisitePro's only specialty is requirements traceability analysis, a feature upon which we based the traceability view of the NuSEE toolset.

2.2.7 SpecTRM

Developed by Safeware Engineering Co., SpecTRM features intent specifications, which refers to a new way of structuring the system and requirements specifications that support system, safety and software engineering tasks. SpecTRM includes SpecTRM-RL, an executable requirements specification language; the language can be used to construct executable, analyzable models that are readable enough to act as the software specification. The SpecTRM toolset and the SpecTRM-RL modeling language support the construction of complete requirements specifications, including some automated analysis for common omissions and mistakes. SpecTRM supports systematic specifications for the entire lifecycle but offers only guidelines for generating specifications. As a result, its V&V capability is relatively weak for a safety-critical system.

2.2.8 designsafe

Designsafe, which was developed especially for safety analysis by Design Safety Engineering, Inc., gives designers a quick and easy tool for evaluating design risks. It helps companies identify potential hazards and provides elimination methods for the entire software lifecycle. In particular, as shown in Table 1, designsafe focuses solely on identifying hazards.

As noted, these tools have some deficiencies because they focus not on the entire software lifecycle but on specific phases of the cycle. For example, Analyst Pro and MOBY/PLC mostly focus on the requirements phase, whereas SPACE is effectively used in the design and testing phase. In addition, the tools are slightly inadequate in terms of verification ability because they were developed for practical use in industrial projects. Consequently, to cover the entire software lifecycle and various V&V demands in the development of safety-critical systems, we need adequate tools for real projects, thus motivating the development of the NuSEE toolset in the present study. Because it is based on our IE approach, the NuSEE toolset systematically supports most phases of the software development lifecycle, along with various V&V activities with respect to safety analysis.

3. NuSEE TOOLSET

3.1. NuSISRT

NuSISRT stands for nuclear software inspection support and requirements traceability tool. In the IE approach, NuSISRT supports a concept phase or all software lifecycle phases based on documents written in natural language. Inspection is widely believed to be an effective software V&V technique and is extensively used in work related to nuclear systems. Inspection can provide a great increase in both productivity and product quality by reducing development time and by removing more defects than is possible without its use. The outstanding feature of inspections is that they can be applied to the whole lifecycle.

In NuSISRT, we integrated requirements traceability analysis capability into the software inspection support tool, as this is also considered a major item of software V&V harnesses. Additionally, structure decomposition and analysis and inspection meeting support capabilities are integrated in NuSISRT. That is, NuSISRT comprises tools for document evaluation, traceability analysis, structural analysis and inspection meeting support. Designed to support the inspection of all software development documents, NuSISRT is a PC-based application designed for anyone who needs to manage requirements. To support our approach systematically, NuSISRT has three kinds of views: an inspection view, a traceability view and a structure view. It also has a Web page for inspection meetings; however, because of the similarity to general web pages on the Internet, this aspect is not discussed here.

3.1.1 Inspection View

The support of document evaluation with the inspection view is a main function of NuSISRT. It supports an extracting function that reads a text file and copies paragraph numbers and requirement text to a NuSISRT file. It can read any text data that is convertible to '.txt' format. It also supports the manual addition of individual requirements and can import from various formats. The inspection view permits users to associate database items by defining attributes; the attributes attached to individual database items provide a powerful means of identifying subcategories or database items and of managing requirements. The inspection view of NuSISRT supports normal parent-child links for managing requirements. Furthermore, it supports peer links between items in a database and general documents; the peer links provide an audit trail that shows compliance to quality standards or contractual conditions.

Figure 5 shows an example of the inspection view of NuSISRT. The inspection view reads source documents, identifies requirements, and extracts the requirements for

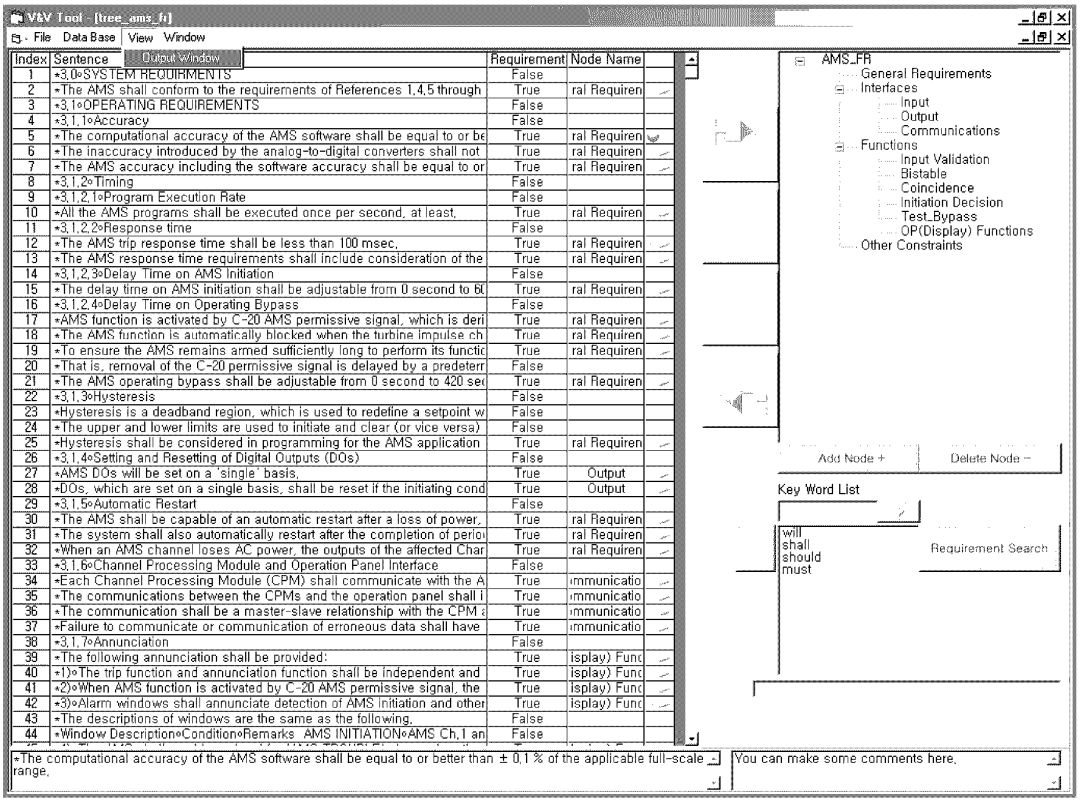


Fig. 5. Inspection View of NuSISRT

Table 2. Inspection Results of AMS through Inspection View

FR (Concept)	Completeness		Consistency		Correctness	
	V&V item	Comment	V&V item	Comment	V&V item	Comment
S/W function define	1	0	1	0	2	1
I/O variables define	25	5	3	1	5	2
S/W behavior define	15	3	1	0	5	3
Interface define	1	0	1	0	1	0
SRS (Requirements)	**Completeness**		**Consistency**		**Correctness**	
	V&V item	**Comment**	**V&V item**	**Comment**	**V&V item**	**Comment**
S/W function define	1	0	1	0	2	0
I/O variables define	25	10	3	2	5	1
S/W behavior define	15	4	1	0	5	1
Interface define	1	0	1	0	1	0

import into the database. The inspection view automatically finds and extracts requirements based on a set of keywords defined by the user. As the requirements are found, they are highlighted, as shown in Figure 5. The user can also manually select and identify requirements. The inspection view enables the production of a user-defined report that shows various types of inspection results. The user builds up the architecture of the desired reports in the right-hand window of Figure 5. If the user writes down checklists in this window, NuSISRT can directly support the software inspection with this functional window. The requirements to be found by the tool are located in a suitable checklist site using various arrow buttons in the window. In this way, each inspector can examine the requirements and generate the inspection result documents with the aid of NuSISRT.

In a single document, while there are many sentences not all of them are requirements. Therefore, we have to elicit adequate requirement sentences for more effective inspection. Then, using the inspection view of NuSISRT, a software requirements inspection based on checklist can be performed by each inspector. As a simple example, we performed an inspection for an ATWS Mitigation System (AMS) based on our approach with the inspection view of NuSISRT. We examined a Functional Requirements (FR) document [18] in the concept phase and a Software Requirements Specification (SRS) document [19] in the requirement phase. In view of three V&V criteria, completeness, consistency and correctness, the checklist was composed by the authors for checking the S/W function definition, I/O variable definition, S/W behavior definition, and interface definition. As shown in Table 2, we found some comments for AMS according to the V&V items in the checklist.

3.1.2 Traceability View

The traceability view of NuSISRT supports a requirements traceability analysis between two kinds of system documents. The traceability view provides mechanisms to easily establish and analyze traceability between requirements sentences through real-time visual notification of change in the Requirements Traceability (RT) matrix form. This capability allows users to pinpoint its impact across the project and assess coverage for verification and validation. For the traceability analysis, an identification number of requirements should be assigned to each requirement sentence elicited from the inspection view, as noted in section 3.1.1. Then, in the RT matrix, the relation between the source requirements and the destination requirements should be described for the requirements traceability analysis. Using this result pertaining to the relation, we can analyze the traceability between documents. Figure 6 shows schematically illustrates an example of requirements traceability using the traceability view of NuSISRT.

Through the traceability view, it is possible to analyze traceability between source documents and destination documents. The traceability view of NuSISRT 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.

As shown in Figure 7, the column number of the matrix represents a requirement of the source document and the row number of the matrix represents the destination document. The relationships between source and destination are expressed through a matrix window

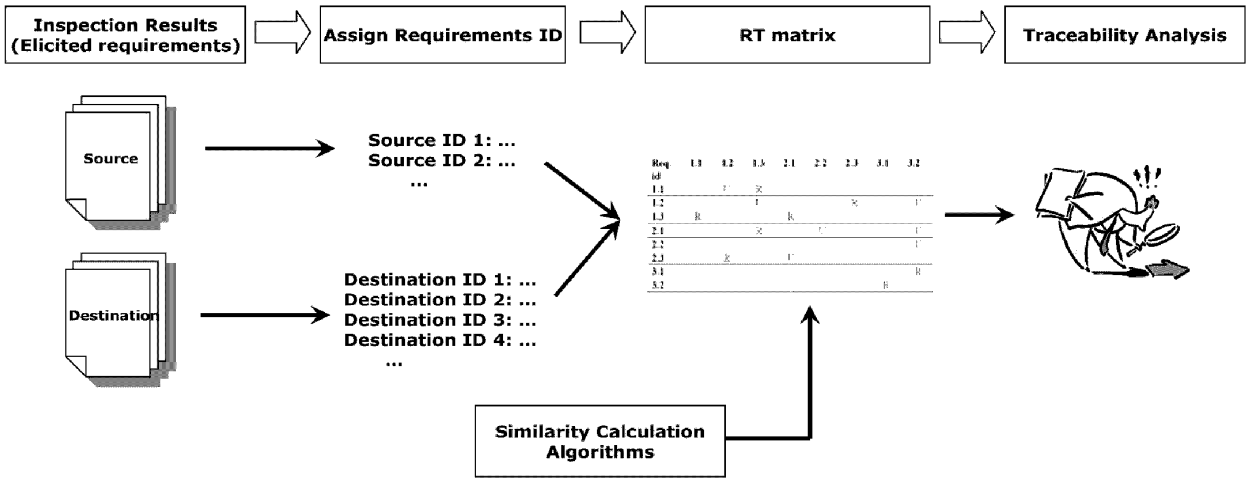


Fig. 6. Schematic Diagram of Requirements Traceability

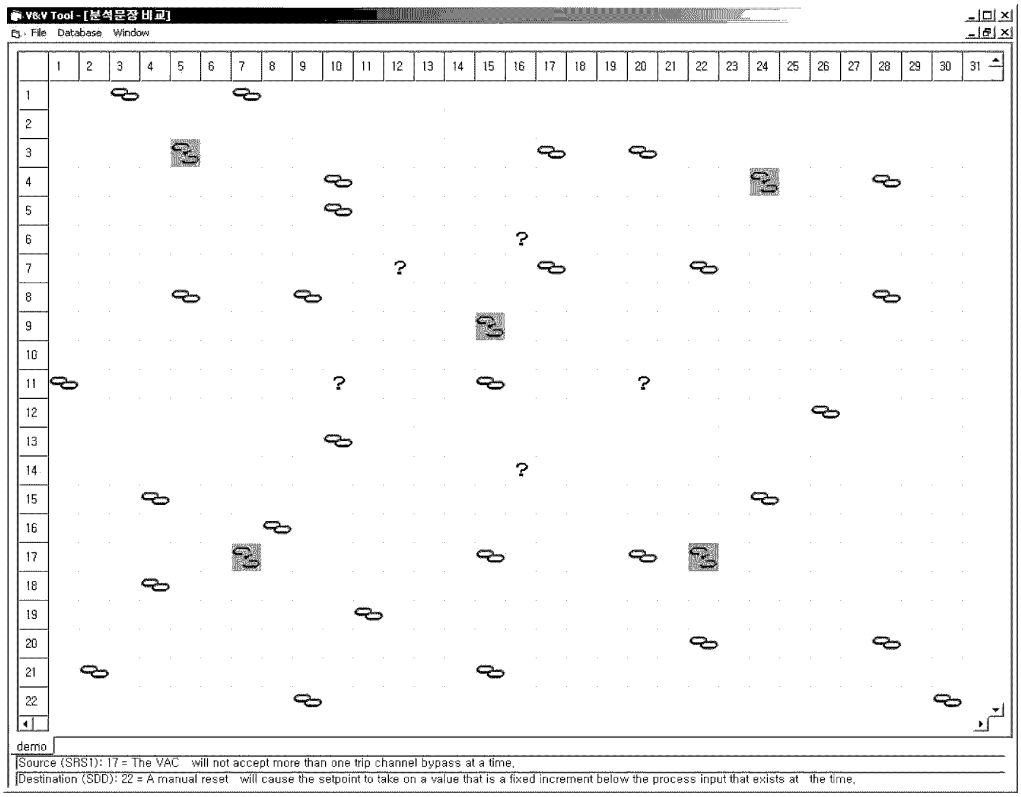


Fig. 7. Traceability View of NuSISRT

with linked and unlinked chains. The linked chains indicate that source requirements are reflected into destination requirements. The unlinked chains represent that source and destination requirements are changed, and thus it is necessary to verify the change between the source and destination documents. The question marks denote that it is difficult to define traceability between requirements. In this instance, it is necessary to verify requirements by another analyzer.

In order to more easily support traceability analysis, the traceability view has an additional function to calculate the similarity between requirements using similarity calculation algorithms [4]. Through this function, the traceability view can automatically represent the similarity by percentage, as shown in Figure 8. This similarity result is then helpful to the user and analyzer. We now propose algorithms to calculate the similarity for both English and Korean documents. In this way, a traceability analysis between documents can be performed through the traceability view.

As an example, we also applied the traceability view to AMS [18,19]. Table 3 shows the results of a traceability analysis for a very early version of AMS. Based on the results of inspection using the inspection view, there were 96 requirement sentences in the FR document and there were 142 requirement sentences in the SRS document. Among them, 60 requirement sentences of FR were reflected in the SRS document and we could not find a relation for 21 requirement sentences of FR. We could not determine the relation for 15 requirement sentences of FR.

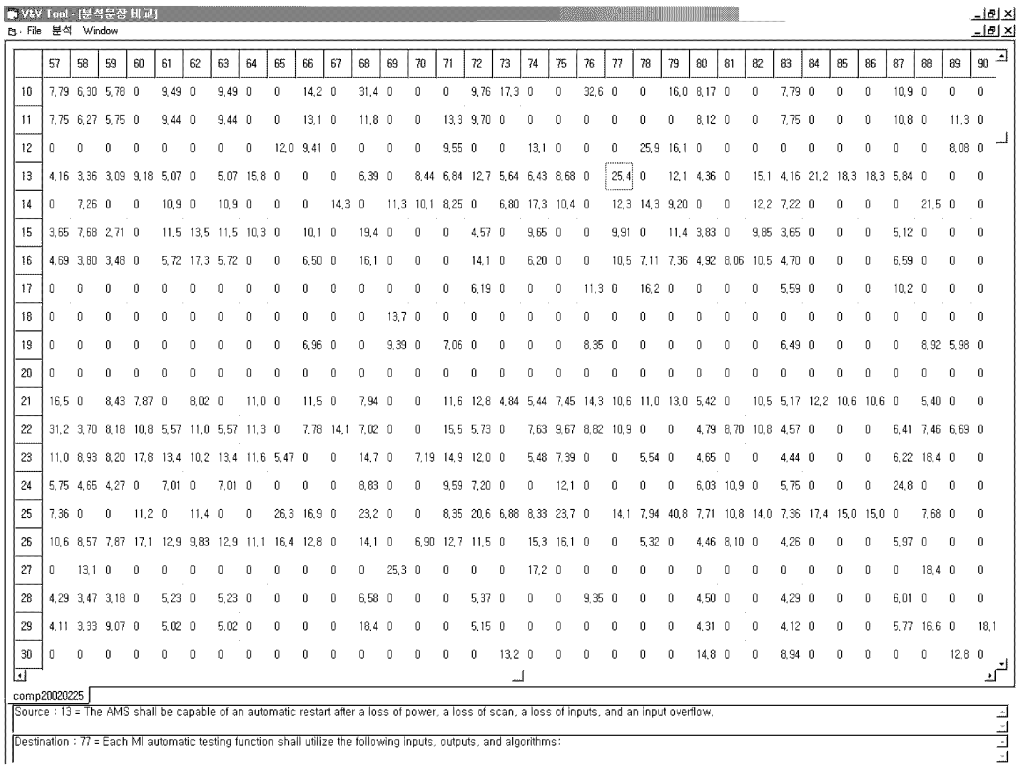


Fig. 8. An Example of Similarity Calculation

Table 3. Requirements Traceability Results of AMS through Traceability View

AMS	Number of requirement	Relation	No relation	Questionable
FR	96	60	21	15
SRS	142	N/A	N/A	N/A

3.1.3 Structure View

As one of the interfacing functions in the IE approach, the structure view of NuSISRT enables effective translation into NuSRS. Figure 9 shows a screen shot of the structure view of NuSISRT.

Through the structure view, we can analyze system development documents in view of the system's structure. These analysis results then help us generate a formal specification from a natural language document in the requirements phase. In the structural analysis of systems through the structure view, it is critical to define inputs/outputs and functions. Therefore, we propose an Input-Process-Output structure type in this work. In the structure view, several tabular forms help users easily build up the Input-Process-Output structure. This structure is represented in the right-hand side window as a tree type. After the structure analysis, the structure view generates a result file written in XML language, which is then transferred to NuSRS. With this file, FOD could be drawn automatically in NuSRS. Table 4 shows an example of structure decomposition for AMS.

3.2. NuSRS

Though formal methods such as Statechart [9], CPN [10], RSML [11], and SCR [12] are also considered as effective V&V harnesses, it is not easy to use them 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 Software Requirements Specification (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, the Function Overview Diagram (FOD) and each of the functions in the FOD is described by the smaller view of

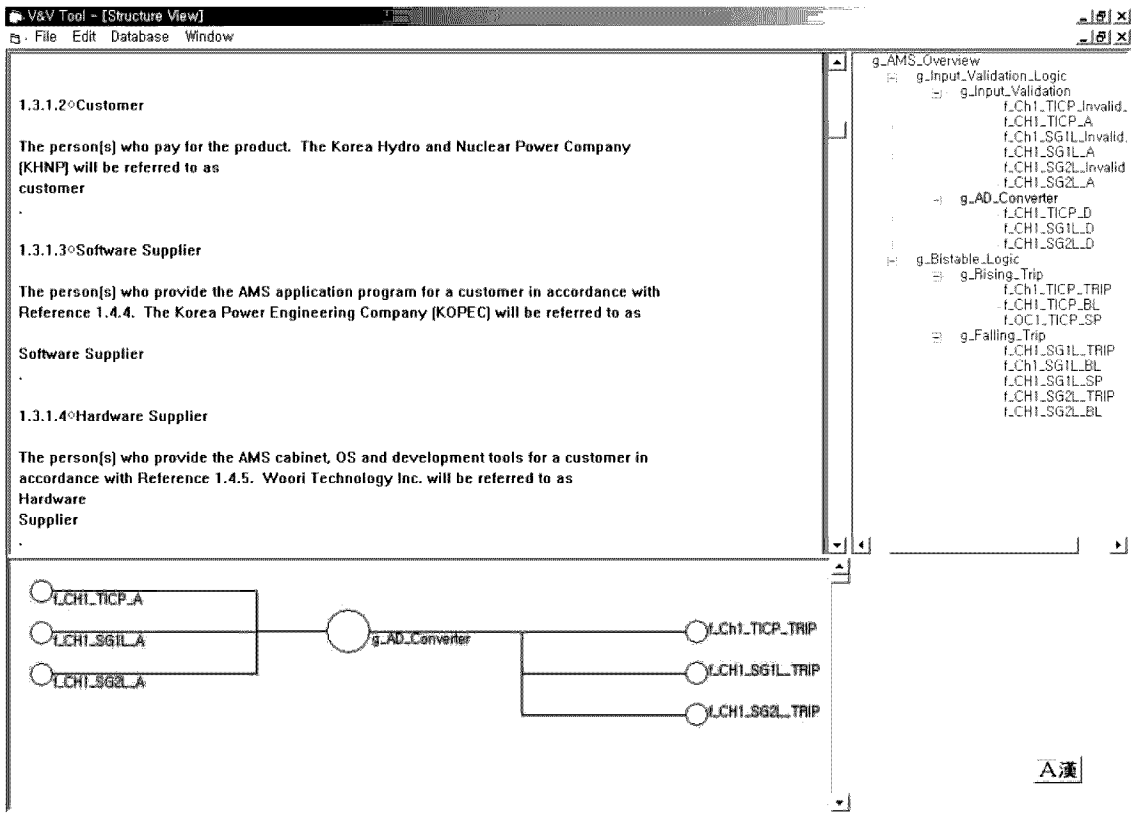


Fig. 9. Structure View of NuSISRT

Table 4. An Example of Structure Decomposition for AMS

Input	Process	Output
m_CH1_TICP m_CH1_SG1L m_CH1_SG2L	g_Input_Validation_Logic	f_CH1_TICP_Invalid_Status f_CH1_SG1L_Invalid_Status f_CH1_SG2L_Invalid_Status f_CH1_TICP_D f_CH1_SG1L_D f_CH1_SG2L_D
f_CH1_TICP_D f_TICP_Set_Point f_TICP_Trip_Time_Delay f_TICP_Trip_Hys_Set_Point f_CH1_SG1L_D f_CH1_SG2L_D f_SG1L_Set_Point f_SG1L_Trip_Time_Delay f_SG1L_Trip_Hys_Set_Point f_SG2L_Set_Point f_SG2L_Trip_Time_Delay f_SG2L_Trip_Hys_Set_Point	g_Bistable_Logic	f_Ch1_TICP_TRIP f_CH1_TICP_BL f_OC1_TICP_SP f_Ch1_SG1L_TRIP f_OC1_SG1L_BL f_OC1_SG1L_SP f_CH1_SG2L_TRIP f_OC1_SG2L_BL f_OC1_SG2L_SP
...	...	...

the Structured Decision Table (SDT). The AECL approach specifies all requirements of the nuclear control system in 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 a formal verification method that is an extension of the existing SCR-style AECL approach [13]. 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 terms of 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, 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 [14], timing requirements that are described in timed-automata [15], and nodes that specify all other requirements exclusive of the previous two types of functional requirements.

NuSRS is an editor for requirement specifications based on the NuSCR approach. Figure 10 shows a screen shot of NuSRS. NuSRS is a platform independent tool made with JAVA for formally specifying the SRS of a nuclear control system. It provides an 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 described SRS, as can be seen on the left side of Figure 10. Additionally, NuSRS also generates a result file written in XML language that includes all of the information in NuSRS, which is then transferred to NuSDS.

As an example, we selected the KNICS Reactor Protection System (RPS) [20] in this section. The KNICS RPS includes Bistable Processors (BPs) and Coincidence Processors (CPs) as its major components. The RPS BP periodically accepts inputs from 18 different safety sensors installed in the system, and it performs necessary comparisons against predefined trip logics and setpoint values. Figure 11 shows an example of NuSCR specification of KNICS RPS. Figure 11(a) is a FOD for *g_Fixed_Setpoint_Rising_Trip_with_OB*, fixed set-point rising trip logic in BP (bistable processor) of RPS, where *g_* denotes the group prefix. Boxed nodes represent inputs and outputs. SDT, shown in Figure 11(b), defines the function variable *f_X_Valid* appearing in the FOD. If the value of *f_X* is between *k_X_MIN* and *k_X_MAX*, the output value *f_X Valid* is 0, indicating a normal case. Otherwise, the output value is 1. NuSCR allows multiple and related terms be written together on the same row. That is, in the AECL-notation, one would have no option but to divide into two rows: *(f_X >= k_X_MIN)* and *(f_X <= k_X_MAX)*. This example is too trivial for a developer to appreciate the difference in expressiveness. However, in the Wolsung SDS2 [13],

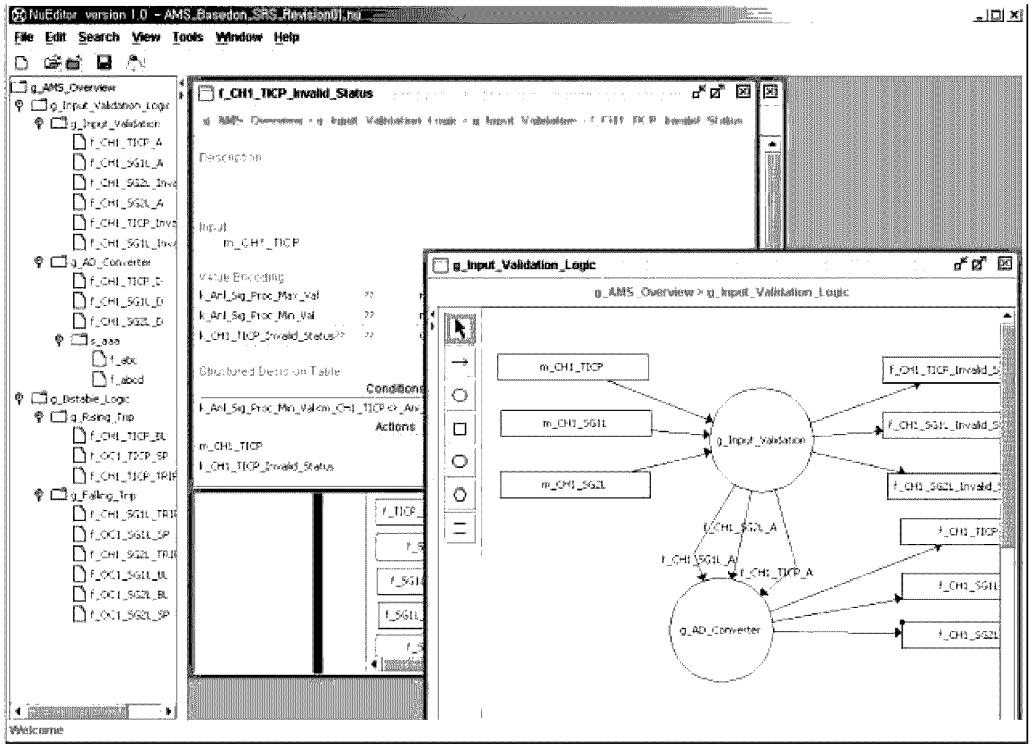
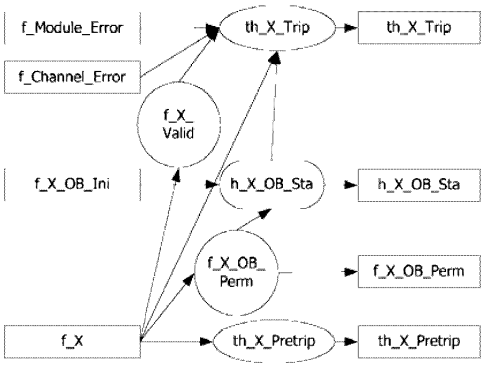


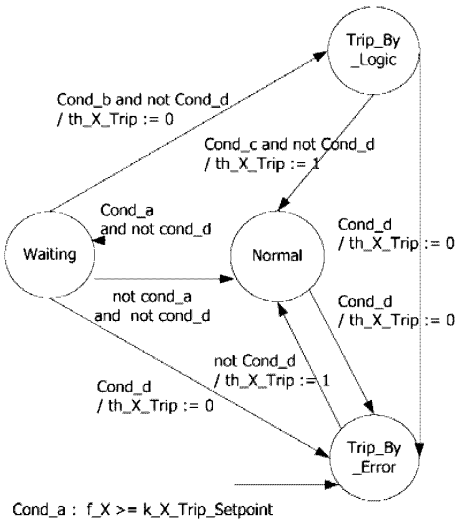
Fig. 10. Editing Windows of NuSRS



(a) FOD for *g_Fixed_Setpoint_Rising_Trip_with_OB*

Conditions		
k_X_MIN <= f_X <= k_X_MAX	T	F
Actions		
f_X_Valid := 0	X	
f_X_Valid := 1		X

(b) SDT for function variable node *f_X_Valid*



(c) TTS for timed history variable node *th_X_Trip*

Fig. 11. NuSCR Specification Example Using NuSRS

which is considerably simpler in complexity than KNICS RPS, the most complex SDT consists of 16 rows and 12 columns, because complex equations have to be decomposed into "primitive" fragments. Domain experts have repeatedly emphasized that mathematical equations used in trip logics, no matter how complex they are, are well-understood and proven-correct as a whole to domain experts and that they need not be artificially fragmented in the specification. Figure 11(c), TTS for *th_X_Trip*, illustrates how behavior of a timed-history variable node is captured. It is interpreted as follows: "If condition *f_X* ≥ *k_X_Trip_Setpoint* is satisfied in state *Normal*, it enters *Waiting* state. If the condition remains true for *k_Trip_Delay* period while in Waiting state, the system generates the trip signal 0. If *f_X_Valid*, *f_Module_Error*, or *f_Channel_Error* occur, then a trip signal is immediately produced. In the state *Trip_By_Error* or *Trip_By_Logic*, if the trip conditions are canceled, the system returns to a *Normal* state and the output 1 is generated." The TTS expression in *Cond_b [k_Trip_Delay, k_Trip_Delay]* means that the condition must remain true for *k_Trip_Delay* unit times. In AECL-style notation, behavior related to a time-dependent state transition was written in tabular notation, and domain experts preferred automata notation to tabular notation.

Similarly, *h_X_OB_Sta*, shown in Figure 11(a), is a history variable node defined as FSM. FSM is the same as TTS except that time constraints are missing. All constructs in NuSCR, s.t. FOD, SDT, FSM and TTS are familiar notations to domain engineers and software developers. NuSCR has been evaluated as being easy to specify and understand by domain engineers.

3.3 NuSDS

Within a software lifecycle, software design is a process of translating problem requirements into software structures that are to be built in the implementation phase [16]. In general industry, a Software Design Specification (SDS) should be produced at the software design phase. SDS describes the 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, the most important task is to define an effective specification method for the software design phase. Effective specification is also important for design V&V. Further, a well-formed design specification is very useful for coding in the implementation phase, because the design phase is simply the previous stage of the implementation phase in the lifecycle. Therefore, an implementation product such as a code should be easily translated from the design specification. In NPP software fields, a Programmable Logic Controller (PLC) is widely used for safety-critical systems [17]. However, a systematic design specification and analysis technique for implementation based on PLC has not yet been developed.

In this work, NuFDS, a software design specification and analysis technique for safety critical software based on PLC, is proposed. NuFDS stands for nuclear FBD-style design specification. Function Block Diagram (FBD) is one of the PLC software languages. For tool support, we developed NuSDS based on the NuFDS approach. This tool is designed particularly for the software design specifications in nuclear fields. SDS is a description to show how to create a design that accurately and completely satisfies the behavior and constraints in the SRS. During the coding phase of the software lifecycle it then becomes 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 systematic verification and easily translation into the implementation phase. NuSDS supports 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.

Figure 12 shows special features of NuSDS. NuSDS can be divided into two steps. In step 1, NuSDS fully supports design specifications according to the NuFDS approach proposed in this work. Then, based on these design specifications, NuSDS partially supports design analysis. This means that NuSDS can support translation into an input language for model checking and aid in connection to other V&V tools in step 2. The development of NuSDS step 1 is now finished. NuSDS step 2 is added when required for the design analysis. For the design analysis, NuSDS has been integrated with architecture description language and a model checker.

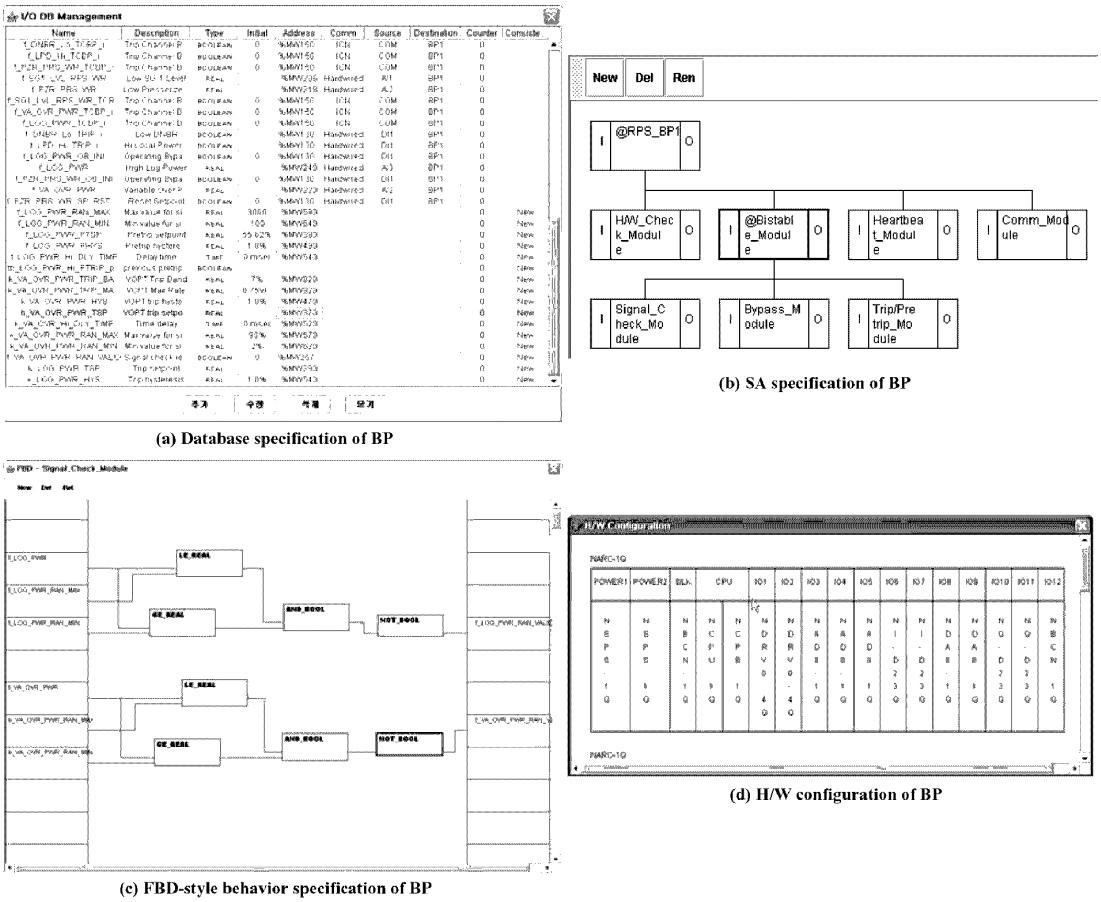
Figure 13 shows the major features of the NuSDS tool, according to NuFDS specifications. The NuSDS consists of a tree-like information window showing input/output and function decomposition, a software architecture window, an FBD-style specification window, a layout diagram for PLC hardware configuration and a database specification window.

As an example, we simply applied NuSDS to KNICS RPS, as outlined in section 3.2. Figure 13 shows a section of the BP design specifications based on the NuFDS. The NuSDS tool is used to specify the design of the BP. Figure 13(a) represents the I/O database specification of the BP and Figure 13(b) shows the Software Architecture (SA) specification of the BP using an architecture design block. In the BP, the SA is composed of *H/W_Check_Module, Bistable_Module, Hearbeat_Module*, and *Comm_Module* as its major architecture design blocks. Each major-architecture contains sub-architectural modules. Figure 13(c) represents the FBD-style specification of the *Signal_Check_Module* of the BP. This FBD-style specification includes the interactions between the function blocks and the I/O variables. Finally, Figure 13(d) shows the hardware layout diagram for the PLC hardware configuration.

Through the software design specification using the

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 SMV
▪ **H/W configuration**	Layout diagram for PLC H/W configuration Assign S/W modules	✓ Informal analysis

Fig. 12. Special Features of NuSDS



(a) Database specification of BP

(b) SA specification of BP

(c) FBD-style behavior specification of BP

(d) H/W configuration of BP

Fig. 13. Software Design Specification of the BP

NuFDS, basic verification is possible. For example, during the design specification of the BP, I/O errors and some missed SAs were found. Among the I/O errors, there was inconsistency between the natural language SRS and the formal SRS, and some I/O variables, such as heartbeat operation related data, were missing. In addition, there were some ambiguities concerning initiation variables that had been declared in the formal SRS. Since the communication module and the hardware check module were not included in the SRS, we newly defined the software architectures in the design phase.

3.4. NuSCM

Software configuration management (SCM) is an activity that configures the form of a system (documents, programs, and hardware) and systematically manages and controls modifications used to compile plans, development and maintenance. Many kinds of documents for system development and V&V processes are produced during the software lifecycle 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 valued highly in both the development phase as well as in the modification and maintenance phases. Even while operating the software, requests in modification continue to be received; hence, in order to confront these requests, specific corresponding plans should be established. If modification requests are not properly processed in the software maintenance phase, deterioration in quality and declination in the life of the software will result. Particularly in systems where safety is seriously valued, the risk of accidents due to software may increase. For this reason, many research institutes and companies are currently making attempts to automate systematic document management in an effort to satisfy high quality and reliability.

NuSCM is a project-centered software configuration management system especially designed for nuclear safety systems. In our integrated environment, NuSCM systematically supports management of all system development documents, V&V documents and codes throughout the lifecycle. Additionally, for the interface between NuSCM and other tools, NuSCM manages all result files produced from NuSISRT, NuSRS and NuSDS. Recently, since most software systems are compatible regardless of location and users can easily approach, web-based systems are being developed . NuSCM was also designed and embodied using the web. Figure 14 shows a document management view and a change request view in NuSCM.

4. CONCLUSIONS

An IE approach, which is an integrated environment of software specification and V&V processes for safety-critical systems, was proposed in this paper. The IE approach systematically supports a formal based specification technique according to the lifecycle and also various kinds of effective V&V techniques based on the proposed specifications, specifically to nuclear fields. For tool support, NuSEE, integrated with various CASE tools including NuSISRT, NuSRS, NuSDS and NuSCM, was developed for support of each software lifecycle phase. NuSISRT is a special tool for software inspection and traceability analysis, and thus can be used in the concept phase as well as all documents based phases. For specific nuclear software fields based on PLC applications, NuSRS and NuSDS support generation of documents such

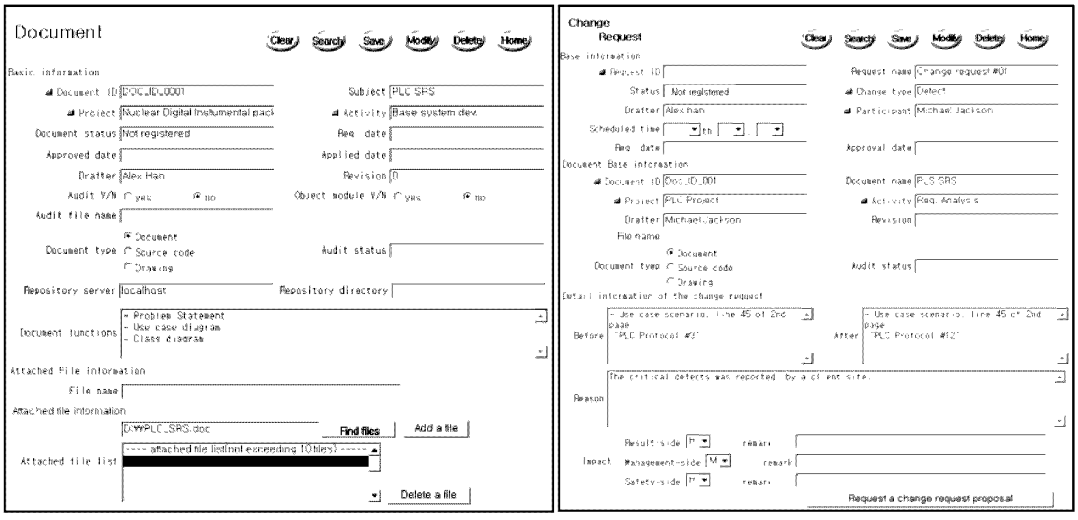


Fig. 14. The NuSCM

Table 5. Summary of Each Tool

	S/W development life cycle	Main functions	Advantages
NuSISRT	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, System 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

as SRS in the requirement phase and SDS in the design phase. Also, they support formal based analyses such as theorem proving and model checking. NuSCM is a project-centered software configuration management tool. NuSCM manages and controls the modification of all system development and V&V documents. All result files from NuSISRT, NuSRS and NuSDS can be managed through NuSCM.

In order to integrate the various tools gracefully, this paper considered interfacing functions between each tool an important feature in the NuSEE toolset. The NuSEE toolset achieves optimized integration of work products throughout the software lifecycle of safety-critical systems based on PLC applications. Through the NuSEE toolset, nuclear engineers can reduce the time and cost required for the development of software while user convenience may be enhanced. NuSEE is a tool for building bridges between specialists in system engineering, software engineering, and safety engineering.

In Table 5, each tool in the NuSEE toolset is summarized according to three point of views; S/W development life-cycle support, main functions and special advantages. Upon further development efforts, NuSEE based on an IE approach promises to be a unique and effective software development and analysis tool that will support the entire software lifecycle for the development of NPP safety-critical systems.

ACKNOWLEDGEMENTS

This research was supported by the KNICS (Korea Nuclear I&C System) R&D center and a 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 1012 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] J. Yoo, T. Kim, S. Cha, J. Lee and H. S. Son, "A Formal Software Requirements Specification Method for Digital Nuclear Plants Protection Systems", Journal of Systems and Software, accepted.
[6] S. Koo, P. Seong and S. Cha, "Software Design Specification and Analysis Technique for the Safety Critical Software based on Programmable Logic Controller (PLC)", Eighth IEEE International Symposium on High Assurance Systems Engineering, pp. 283-284, 2004.
[7] S. Koo, P. Seong, J. Jung and S. Choi, "Software Design Specification and Analysis (NuFDS) Approach for the

Safety Critical Software based on Programmable Logic Controller (PLC)", Proceedings of the Korean Nuclear Spring Meeting, 2004.
[8] S. Koo, P. Seong, J. Yoo, S. Cha and Y. Yoo, "An Effective Technique for the Software Requirements Analysis of NPP Safety-Critical Systems, Based on Software Inspection, Requirements Traceability and Formal Specification", Reliability Engineering & System Safety, 2004, in press.
[9] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, vol. 8, pp.231-274, 1987.
[10] Kurt Jensen, Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use Volume 1 ,Springer-Verlag Berlin Heidelberg, 1997.
[11] 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.
[12] C. Heitmeyer and B. Labaw, "Consistency Checking of SCR-style Requirements Specification", International Symposium on Requirements Engineering, March 1995.
[13] 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.
[14] J. Hopcroft and J. Ullman, Introduction to Automata Theory, Language and Computation, Addison-Wesley, 1979.
[15] R. Alur and David L. Dill, "A theory of Timed Automata," Theoretical Computer Science Vol. 126, No. 2, pp. 183-236, April 1994.
[16] Roger S. Pressman, Software Engineering: A practitioner's approach, McGRAW-HILL Book Co, 2001.
[17] IEC, IEC Standard 61131-3: Programmable controllers-Part 3, IEC 61131, 1993.
[18] KOPEC. Functional Requirements for ATWS Mitigation System for KORI NPP UNIT 1, 2001
[19] KOPEC. Software Requirements Specification for ATWS Mitigation System for KORI NPP UNIT 1, 2001.
[20] KNICS (Korea Nuclear Instrumentation and Control System Research and Development Center). http://www.knics.re.kr.