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

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.

![Figure 1. Software V&V Tasks along Life Cycle](image-url)

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 does 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 straightforward 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.

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.

![Figure 2. Overall Scheme for an Integrated Environment](image)

![Figure 3. Inspection View of SIS-RT](image)
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.

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.

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.

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.

Figure 6. NuSRS
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.

<table>
<thead>
<tr>
<th>Design Properties</th>
<th>Step 1: Design Specification</th>
<th>Step 2: Design Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>I/O information from NuSRS</td>
<td>I/O completeness check</td>
</tr>
<tr>
<td></td>
<td>Define DB fields for PLC code</td>
<td>Fields consistency check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB consistency check</td>
</tr>
<tr>
<td>S/W architecture</td>
<td>S/W module decomposition</td>
<td>Formal analysis based on AoA</td>
</tr>
<tr>
<td></td>
<td>SD top-down design method</td>
<td>(Model checking using SMV)</td>
</tr>
<tr>
<td></td>
<td>(Decomposing transform supporting structure)</td>
<td></td>
</tr>
<tr>
<td>System behavior</td>
<td>Basic FBD from NuSRS</td>
<td>Design consistency check</td>
</tr>
<tr>
<td></td>
<td>FBD specification based on S/W module</td>
<td>Design traceability analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Model checking using VS)</td>
</tr>
<tr>
<td>H/W configuration</td>
<td>Layout diagram for PLC</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>H/W configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assign S/W module</td>
<td></td>
</tr>
</tbody>
</table>

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.

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.

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 on the table, the integrated environment supports various features throughout the software development life cycle. NuSRS and NuSDS are tools designed for specific nuclear software fields. NuSRS supports System Requirement Specification (SRS), while NuSDS supports System Design Specification (SDS). 

### Table 1. Summary of Each Tool

<table>
<thead>
<tr>
<th>S/W Development Life Cycle</th>
<th>Main Functions</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>NuSRS - System Requirement Phase</td>
<td>Document inspection supporting, System structure analysis supporting</td>
<td>Systematic checklist management, Reducing time of inspection work</td>
</tr>
<tr>
<td>NuSRS - Software Requirement Phase</td>
<td>Formal method (NuSCR) supporting, Theorem proving (PVS) supporting</td>
<td>Formal method for nuclear fields, Effective system formal specification</td>
</tr>
<tr>
<td>NuSDS - Software Design Phase</td>
<td>Model checking supporting, Traceability analysis supporting</td>
<td>Formal method for nuclear fields, Effective system design specification</td>
</tr>
<tr>
<td>NuSNSM - Whole Phase</td>
<td>Project control, configuration management supporting, Change request form in nuclear fields supporting</td>
<td>CM techniques for nuclear fields, Various documents style supporting</td>
</tr>
</tbody>
</table>

### 7. Acknowledgement

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

### References