Journal of the Korean Nuclear Society
Volume 32, Number 6, pp.537~548, December, 2000

A Comparative Study of Formal Methods for Safety Critical Software in Nuclear Power Plant

Sedo Sohn and PoongHyun Seong
Korea Advanced Institute of Science and Technology
373-1, Kusong-dong Yusong-gu, Taejon 305-701, Korea
sdsohn@ns.kopec.co.kr

(Received August 11, 1999)

Abstract

The requirement of ultra high reliability of the safety critical software can not be demonstrated by testing alone. The specification based on formal method is recommended for safety system software. But there exist various kinds of formal methods, and this variety of formal method is recognized as an obstacle to the wide use of formal method. In this paper six different formal method have been applied to the same part of the functional requirements that is calculation algorithm intensive. The specification results were compared against the criteria that is derived from the characteristics that good software requirements specifications should have and regulatory body recommends to have. The application experience shows that the critical characteristics should be defined first, then appropriate method has to be selected. In our case, the Software Cost Reduction method was recommended for internal condition or calculation algorithm checking, and statechart method is recommended for the external behavioral description.

Key Words : formal method, safety critical software, CPCS, software specification

1. Introduction

Digital systems are used widely and the software is performing critical functions that were previously performed by human or analog devices. Some complicated control functions are not implementable in some applications without sophisticated control softwares. Thus digital systems are used more and more in the safety applications, such as flight control systems, train control systems, and nuclear power plant control and protection systems. The applications of software in these safety critical applications require ultra high reliability. For reliable software, the testing of the software has been emphasized. And a lot of test scheme was developed to eliminate errors or to measure the reliability of the software. But it is said that the ultra high reliability required in safety critical system is not feasible with testing. Software with reliability of 1.0 E-4 for an hour takes almost 1 year for a fault to occur, and extending these data by statistical methods is not sound. Most of the software errors were known to occur from the errors in the requirements phase. Also errors in requirement phase is costly to fix than errors detected in later phase of software life

cycle. Thus a lot of the resources of software developments are spent in the requirements specification phase. The formal method was devised to show mathematically that the software is error free. Formal method is defined as the use of mathematical modeling, calculation, and prediction in the specification, design, analysis, construction, and assurance of computer systems and software. There are many ways to classify formal methods. It can be classified as analytic methods or descriptive methods. It can be classified, depending on the level of formality, as level 1, level 2, or level 3, with level 1 mixed use of natural language and diagrams, with level 2 employing fixed specification language, and level 3 accompanying mechanized analysis. The formal methods need not be applied to the project from the beginning to the end or to the all products. It can be applied to the selected component which is critical, can be applied to investigate specific properties, or can be applied to the early requirements specification phase. The descriptive method utilizes the model-oriented specifications and the desired properties or behaviors are specified by giving a mathematical model that has those properties. Z, Vienna Development Method (VDM), and Communicating Sequential Processes (CSP) belong to this classification. Analytic methods utilize property-oriented specifications. They use axiomatic style to state the properties and relationships that are required to hold of the component being described. Prototype Verification System (PVS) and Larch belong to this classification. The specifications with formal method can be analyzed for consistency and completeness. The formal specification can be validated with animation or by posing and proving putative theorems. The specification with formal method should be analyzed to get maximum benefits of modeling. But the applications of formal method without analysis can be beneficial in understanding the requirements. There exist so many formal methods available with different mathematical background, and this was pointed out to be an obstacle to the wide use of formal method. There have been efforts to evaluate formal methods from the point of practical applications. The steam boiler controller problem controlling the level of the boiler with pumps and valves was set for this purpose. There have been more than 30 different applications using formal methods for this problem. The Production Cell problem, a comparative study featuring more than 30 contributions, was performed to serve as a testbed for formal approaches to reactive system design and verification. For nuclear power plant safety systems of Wolsung Nuclear Power Plant Unit 2,3, and 4, the shutdown system (SDS) number 1 and 2 were designed with formal methods. SDS 1 was designed with Integrated Approach (IA) which simplifies the software design process with graphical requirement specification and design. SDS 2 was designed with Rigorous Design Method which adopts the tabular method of Software Cost Reduction(SCR) method. For safety critical application, the IEC880 reads the use of formal specification language may be a help to show coherence and completeness of the software functional requirements and automatic tools may be used for this purpose [1]. The United States Nuclear Regulatory Committee (USNRC) refers the formal methods as techniques that are still under development and encourages the informed use of formal methods as part of a applicant/license' s software engineering process in Standard Review Plan [2].

This paper is to investigate the appropriate methods for the application of the formal methods in the design of the software for safety critical system in nuclear power plants which is calculation algorithm intensive. Six formal methods, IA, SCR, CPN, Statechart, Z, and PVS have been selected

for evaluation. IA and SCR are the lightest method among six, supporting only graphical specification. These methods have been applied to the safety systems of nuclear power plants. CPN and Statechart are intermediate method supporting graphical specification and simulation. These methods are widely used at applications of computer systems. Z and PVS are the heaviest methods supporting theorem proving. Thus the methods selected represent from the lightest to the heaviest formal methods, and were those that have been used at safety systems of nuclear power plants or those that are used widely for real time systems, or those that are widely used and tools are freely available.

2. Application Example of Nuclear Safety System

The safety system in nuclear power plant performs safety function either cutting off the power source to drop control rods that absorb the neutron to stop nuclear chain reaction or start or stop the pump and valves to supply emergency cooling water into the reactor. The design principle for the safety system is defense in depth and diversity. The design should be simple and deterministic to predict its behavior. The Core Protection Calculator System (CPCS) performs protective action when the conditions at reactor core are challenging the nuclear design criteria. It monitors the coolant temperature, system pressure, coolant pump speed, excore neutron flux, and CEA position signals to monitor Departure from Nucleate Boiling Ratio (DNBR), Local Power Density (LPD). The functional algorithm for this DNBR and LPD calculation is quite complex. It is a good candidate for the rigorous formal software method application. In this paper, a small part of the functional algorithm is presented and various formal methods are applied to investigate which one is better for the CPCS software application.

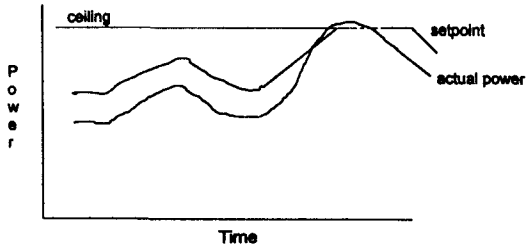

Fig. 1. Behavior of Setpoint of VOPT

2.1. Problem Description

The Variable Over-Power Trip (VOPT) is to generate protection signal when the calculated reactor power is greater than the pre-set value or the increase rate of the reactor power is excessive. The protection signal is generated when the actual power is greater than the calculated trip setpoint. The trip setpoint is a variable, and is calculated to follow the actual power with preset margin within maximum ceiling. But the increase rate of the trip setpoint is limited by pre-specified value. Thus the trip setpoint is changing as the measured actual power is changing, but as the actual power increases at a greater rate than the setpoint can change, the actual power becomes greater than the trip setpoint resulting in trip signal generation. The VOPT is based on auctioneered power from excore detector signals, temperature shadowing corrected excore detector signals, neutron flux power, and thermal power. A variable, FOLLOW, is calculated which follows changes in the auctioneered power within rate limits. FOLLOW cannot be changed from its previous value by more than an amount depending on the data base limits and the computing interval. The VOPT setpoint is computed by adding a fixed power bias, ΔSP_V, to the variable FOLLOW. The VOPT

setpoint is limited by a minimum allowable value, SP_{VMIN}, and a maximum allowable value, SP_{VMAX}. The trip is set by adding 40 to the auxiliary trip flag J_{TRP}. The dynamic behavior of setpoint that varies based on actual power is shown in the Figure 1.

The raw neutron flux power and the temperature corrected neutron flux power are calculated by the following algorithms [3],

$$\Phi_{RAWD} = K_{CAL}(D1 + D2 + D3) \cdot 100.0$$
$$\Phi_{TCORD} = \Phi_{RAWD} \cdot T_F$$

The maximum power is then selected.

$$\Phi_{MAX} = \max(\Phi_{RAWD}, \Phi_{TCORD}, \Phi_{CAL}, B_{\Delta T})$$

Where,

D1,D2,D3 = scaled, excore neutron flux detector values

KCAL = addressable neutron flux power calibration constant

Φ_{CAL} = calibrated neutron flux power(% of rated power)

$B_{\Delta T}$ = calibrated static component of thermal power (% of rated power)

Φ_{RAWD} = raw neutron flux power (% rated power)

Φ_{TCORD} = temperature corrected neutron flux power (% rated power)

T_F = coolant temperature shadowing factor

Φ_{MAX} = maximum power (% rated power)

The FOLLOW is calculated by,

$$FUP = \min(\Phi_{MAX}, FOLLOW_p + SUPMAX)$$
$$FDN = FOLLOW_P - SDNMAX$$
$$FOLLOW = \max(FUP, FDN)$$

The VOPT setpoint is then calculated:

$$SP_{VOP1} = \min((FOLLOW + \Delta SP_V), SP_{VMAX})$$
$$SP_{VOPT} = \max(SP_{VOP1}, SP_{VMIN})$$
$$If\ \Phi_{MAX} \geq SP_{VORT} \cdot then \cdot J_{TRP} = 40 + J_{TRP}$$

Above, the subscript p denotes value from previous execution

SUPMAX = maximum increase for FOLLOW within an execution of UPDATE(% rated power)

SDNMAX = maximum decrease for FOLLOW within an execution of UPDATE(% rated power)

FOLLOW = rate limited maximum of auctioneered power(% rated power)

FUP, FDN = intermediate variables used to calculated FOLLOW

ΔSP_V = amount in % power, by which VOPT setpoint is above FOLLOW

SP_{VMAX} = maximum allowable value of VOPT setpoint(% rated power)

SP_{VOP1} = intermediate variable used to calculate VO_{PT} setpoint

SP_{VOPT} = VOPT setpoint (% rated power)

J_{TRP} = Auxiliary trip flag

3. Applied Formal Methods to the Example

3.1. Integrated Approach (IA) Method

The IA consists of two parts, one is the graphical programming language and the other is the procedure that combines the software requirement specification stage and the software design stage. The requirement specification is defined with the functional graphical language and design is performed by adding more information on this requirement specification. This method is very much dependent on tools. The graphical language tool is an indispensable ingredient for this method. The significant advantage of this

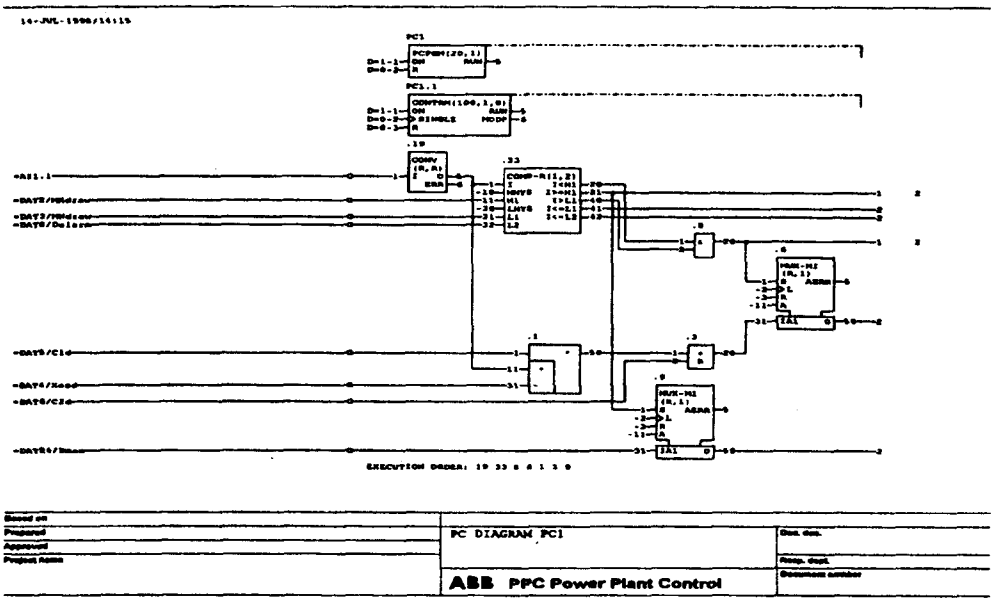

Fig. 2. VOPT Specification with IA

method is the simplification of the design steps by reducing interfaces between design steps or by removing design steps. With this method, software requirements are easier to write, understand, and review. The design stage is significantly simplified. The implementation stage is practically eliminated. With this reduced design step interfaces or removed design steps, the design process is simplified making the design verification and traceability very easy. But the application of this method is limited to the simple system application like logic programming. This Integrated Approach was used in the design of the Wolsung shutdown system design of SDS 1. The IA is well suited for small and simple applications. To apply to large and complex problems, it looks necessary to have higher level description of the software. This method does not support any specification executions or verification. This is the lightest formal method of the methods reviewed in this paper. In case of ABB PLC AC160, the system comes with graphical tools for application programming language called AMPL (Application Modular Programming Language). There are about 150 programming elements that can be used for programming. The design information is provided into the programming with appropriate system information. The programming is done with two parts, DB and PC element. The input and output information are stored in the DB element. The application program logic is entered in the PC element. Part of VOPT specification with AMPL language is shown in Figure 2. The square in the figure represents the logic program element, the lines between these square represents the data flow paths. The description in the left side of the square denotes either analog inputs or the intermediate values from other functional square. The execution order of the logic is from top to bottom, left to right.

3.2. Software Cost Reduction(SCR) Method

As one of the techniques for making requirements specifications precise, unambiguous, and easy to check for completeness and

consistency, K. Heninger proposed techniques for notations based on data item [4]. The symbolic names for data items and values are like /input-data-items/, //output-data-items//, and $nonnumeric-values$ to reduce confusion and for systematic cross-referencing. Also she recommends use of special tables for precision and completeness [4]. The SCR uses four variable model for description of the system; monitored variables, controlled variable, input variables, and output variables. The relationship between input variables and monitored variables are described by REQ. The relationships are described in the specification using tabular notation [5]. The conditions can be checked for its completeness and consistency. Other characteristics that can be verified are deterministic nature, and lack of circularity in specification. The method has been applied to the Wolsung SDS 2 design. The specification was written manually and a lot of tables were created. The problem was that the checking of the consistency and completeness was very tedious and was done manually. The automatic tools are under development and are argued that more functionalities are added into the tool. It has the specification editor which supports generation of tables for defining terms, tables for defining controlled variables, and mode transition tables. It has a consistency checker that can perform syntax and type errors, instances of incompleteness in the variable definition and dictionaries, missing initial values, unreachable modes, circular definitions, and the missing cases (coverage errors) and non-determinism (disjointness error) [6]. Part of the VOPT specification that adopts the SDS 2 notation is shown in Figure 3. The variable name for calculated outputs are denoted with a prefix f_. The structured decision table denotes what the f_Jtrp should have from the values of condition statements. With this form of tabular notation, conditions can be easily checked for completeness and consistency.

Variable Name : f_Jtrp

Description: VOPT trip flag determination
Function Inputs: *f_Spvopt, f_Phimax, f_Jtrp*
Value Encoding : N/A

Condition Macros: N/A
Structured Decision Table:

CONDITION STATEMENTS			
f_Phimax > f_Spvopt		T	F
f_Phimax <= f_Spvopt		F	T
Action Statements			
f_Jtrp = f_Jtrp + 40		X	
f_Jtrp = f_Jtrp			X

Fig. 3. VOPT Specfication with SCR

3.3. Colored Petri Net (CPN)

Colored Petri Net have been developed from ordinary Petri net by adding colors to the token and adding hierarchical description. In CPN, the system is modeled with places that have tokens, and transitions that generates or removes tokens from the places when conditions are met. The state of the system is determined by the distribution of tokens in places. A Colored Petri Net is a tuple CPN = (Σ, P, T, A, N, C, G, E, I). Σ is a finite set of non-empty types color sets, P is places, T is transition, A is arc, N is node, C is color function, G is guard, E is arc expression, and I is initialization function. CPN is Graphical Oriented Language for design, specification, simulation, and verification of system. This method has been applied for the analysis of synchronization, communication and resource sharing between concurrent processes communication protocols, distributed systems, automated production systems, work flow analysis, and VLSI chips [7]. The specification in CPN is verified with Simulation of the specification, Occurrence Graph (Reachability graph) analysis, or place invariant. For the application that is sequential execution of algorithms like VOPT, the

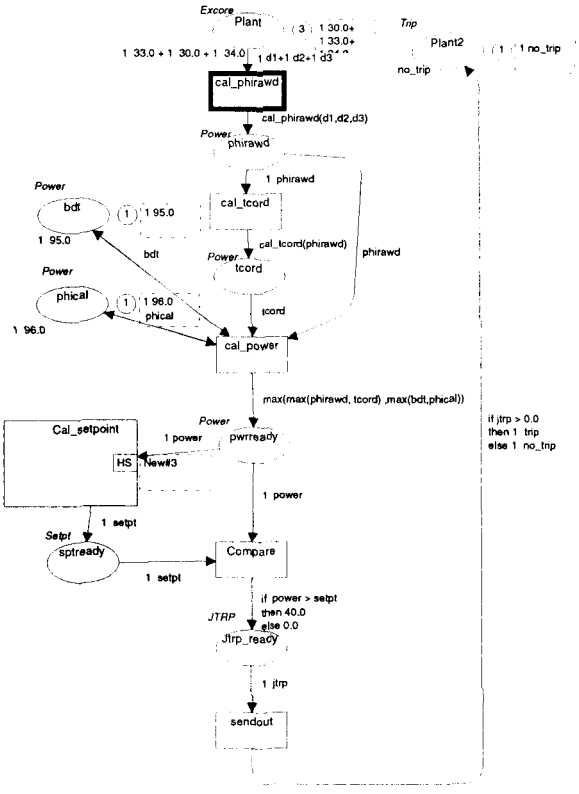

Fig. 4. VOPT Specfication in CPN

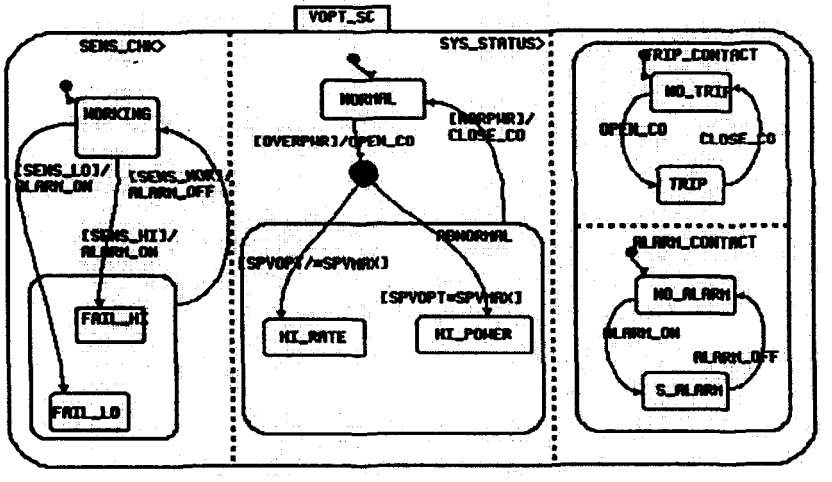

Fig. 5. VOPT Specfication in Statechart

specification with CPN resulted in a diagram similar to flow chart. The calculated output was treated as a token and calculation is described as a transition. The transformation was performed to

this token according to the algorithm. This resulted in lots of tokens making the analysis of the system not feasible. Thus the tokens should be defined for abstracted component of the model. Part of the VOPT specification with CPN is shown in Figure 4 with simulations being executed. The input signals and calculated results are represented as tokens in the corresponding places. These tokens are processed as they move through the transitions.

3.4. Statechart

Finite state machine (FSM) describes a system with initial states, possible incoming input events, resulting possible new states, and action or events from this state transitions. The FSMs are not used because it is flat providing no hierarchy, or modularity, and transition involves many states resulting in state explosion problem, and it is not appropriate for concurrency. D. Harel introduced hierarchy in state, orthogonality between states, and broadcast communication among states. The hierarchy of the state is described by defining sub-state which is depicted inside the parent state. Orthogonality of the state is used to describe the state in short notation and denotes the states that can not happen at the same time. The events in the statechart are communicated to other statecharts in broadcasting manner. The statechart is modeling specification language with graphical method. Thus application engineers can easily understand the specification with basic training. And from the hierarchical nature of the specification, the specification can be understood from overview into details of the system. Statemate developed by D. Harel contains the design tool for activity chart, statechart, and module chart. The activity chart is used for functional description of the system, statechart is used to model behavior view of the system, and

```
--- setalarm ----------
| Delta VoptModel;
| Xi internal
|-----------
| sensoralarm' = if dalarm = 1 then   alarm else   noalarm;
| vopttrip' =vopttrip;
| d1'=d1; d2'=d2; d3'=d3
-------------------------


--- Setpoint ----------
| Delta internal;
| Xi constant;
| spvop1, fdn, fup, followp, follow : Z;
|----------
| fup = min {power, (followp + supmax)};
| fdn = followp- sdnmax;
| follow = max {fup, fdn};
| spvop1 = min { (follow+delspv), spvmax};
| spvopt' = max {spvop1, spvmin};
| followp = follow;
| power'=power
-------------------------
```

Fig. 6. VOPT Specfication with Z

the module chart is used to describe the structural view of the system [8]. The statechart of the VOPT function using the Statemate toolset is shown in Figure 5. The SENS_CHK, SYS_STATUS, and state with RTP_CONTACT and ALARM_CONTACT substates are three independent states. The state name with '>' denotes the state is defined with detailed algorithms. The SYS_STATUS consists of two states NORMAL and ABNORMAL. ABNORMAL state is further divided into two HI_RATE and HI_POWER, which is orthogonal. The state transition is denoted by combination of events, conditions, and action with notations of Event[Condition]/Action. Thus [OVERPWR] condition generates the action OPEN_CO which commands the contacts to open.

3.5. Z

Z is based on mathematics of set theory and first order predicate calculus. Z was originated by J.R. Abrial and developed at the Oxford University Computing laboratory. It is widely used with

```
% Power Calculation

Phirawd(inp): real = kcal * inp
PowerCal(inp, phical, bdt) : real =
  IF (SensorSet(inp) >=phical AND SensorSet(in) >= bdt)
  THEN SensorSet(inp)
  ELSE IF(phical >= SensorSet(inp) AND phical >= bdt)
    THEN phical
    ELSE bdt
    ENDIF
  ENDIF

% Setpoint Calculation

Fup(inp, phical, bdt, followp):real =
  IF (PowerCal(inp, phical, bdt) >= (followp + Supmax))
    THEN (followp + Supmax)
    ELSE PowerCal(inp, phical, bdt)
  ENDIF

Fdn(followp): real = followp – Sdnmax
Follow(inp, phical, bdt, followp): real =
IF (Fup(inp, phical, bdt, followp) >= Fnd(followp) )
  THEN Fup(inp, phical, bdt, followp) ELSE Fdn(followp)
ENDIF
```

Fig. 7. VOPT Specfication with PVS

governments, academics and some industry. It is standardized by International standardization under ISO/IEC. Set is collection of objects or elements of some type, predicate is a general form for expressing properties of an environment, which may be expressed as truth-valued relationships between values of variables in the environment, or between other predicates. Z method is modeling based language, thus need to select appropriate level of abstraction. But it can not model timing property, it is suitable for sequential algorithm. Program verification is performed using Hoare triple notation. Verification is performed by showing existence of initial conditions, proving putative theorem, or perseverance of precondition. The Z specification uses schema notation, schema is a two dimensional graphical notation for grouping together all the relevant information that belongs to a state description [9]. The declaration part, upper part of schema contains the declaration of variables, and formula in the predicate part, lower part, contains the state invariant or operations on declared variables. All state components and associated constraining predicates are included into the schema. This allows information hiding, and enables emphasis on important details. The properties are described with universal quantifier or existential quantifier. State space is derived first, then the invariant, and operation on these state variables are defined at each schema. The specification can be type checked and type checking is based on set theory and validated by executing the specifications. Part of the VOPT specification with Z is shown in Figure 6. In this figure, schema setalarm declares that it changes the parameters in the VoptModel schema with notation Delta. And it uses another schema internal in its operational parts. The ' in the Vopttrip' denotes the state after the execution of the schema.

3.6. Prototype Verification System (PVS)

PVS has been developed by Software Research Institute (SRI). It provides an integrated environment for the development and analysis of formal specifications, and support a wide range of activities of creating, analyzing, modifying, managing, and documenting theories and proofs [10]. Specification language of PVS is built on typed higher-order logic. Specification can be constructed using definitions, axioms, or a mixture of two. Basic types include booleans, integers, rational numbers, uninterpreted type, uninterpreted subtype, and interpreted type. Datatypes include arrays, records, lists, sequences, and abstract datatypes. It supports boolean expression, if-then-else, let and where, set, tuple, record, override, and recursive function calls. Thus it supports most of the high level language features. PVS proofer support the construction of readable proofs, provides a collection of powerful proof commands to carry out propositional,

Table 1. Evaluation Results

Criteria \ FM	IA	SCR	CPN	State-chart	Z	PVS
Readability	H	H	M	H	M	M
Implementability	H	H	M	M	M	M
Verifiability	L	M	M	M	H	H
Safety Analysis	L	L	M	M	H	H
Traceability	H	H	M	M	M	M
Timing Analysis	L	L	M	M	L	L
Tool Support	L	L	H	H	H	H

equality, and arithmetic reasoning. Proofs can be edited, saved and rerun. The prover maintains a proof tree, the user must construct a proof tree which is complete, all of the leaves are recognized as true. Design verification is performed by constructing a mapping function that relates the objects of the high level design specification to the objects of the requirements level specification [10]. The PVS provides the mechanized tools to prove that the specifications meet the putative theorems. The important point with PVS is the verification of the specification. The verification is mechanized and performed by calling appropriate command set to prove the specification. Part of the VOPT specification with PVS is shown in Figure 7. The functions are defined with type declaration and definitions. The writing of specification with PVS is somewhat similar to programming with functional programming language.

4. Evaluation

The criteria chosen for evaluation was based on IEEE standard for software requirements specifications and regulatory guidelines for safety system at nuclear power plant. In the IEEE recommended practice for software requirement specifications (SRS), it lists as characteristics of good SRS as correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable [11]. The regulatory guide stresses the software design process and independent people from the designers should perform verification and validation. And the safety analysis activities should be successfully accomplished for each life cycle activity group to ensure that safety requirements are adequately addressed and no new hazards have been introduced [2]. Among the IEEE recommended characteristics, the correctness, unambiguity, completeness, and consistency are conceived to be met by each formal method. For independent verification and validation, the readability is important characteristics to be considered. Traceability, verifiability, and safety analysis was chosen based on the nuclear regulatory body' s requirements. The implementability was taken considering that the formal specification should be fitted into the current practice of design and implementation process. The tool support is absolutely needed for the industry scale application. IA is very efficient method for logic programming. It has high readability and traceability. The requirements and design is done with same functional graphical language, and the code is generated automatically from these design products. Thus the implementation process has been practically removed. But this method is least rigorous method and the automatic verification and safety analysis

are not supported. The essence of this method is the functional graphical tool, and the tool is offered as part of the hardware platform. Thus the tool is only available when the specific hardware product is selected scoring low in tool support. The SCR method is one of easiest to apply and the reviewer or domain expert can easily understand it. It can be applied without automatic tools, but it will be very tedious. The verification and analysis capability of this method is limited. But it can check missing conditions, inconsistencies, and non-determinism. It is difficult to describe overall view of the behavior. No automatic tool is yet available, but the tool-set SCR* have been described in the paper [6]. CPN is a tool based on graphical notation, and is relatively easy to learn. The verification and analysis of the specification is performed by perseverance of tokens and state occurrence graph. In the case of algorithm intensive application, there can be too many tokens generated making the analysis infeasible. The translation into programming language can be done in hierarchical manner. The timing is supported in the simulated time by giving a token a time stamp. The tool is freely available from the University of Aarhus in Denmark. Statechart describes the external behavior of the system by the event/action and state transitions. Thus the behavioral aspects of the system is well described by this method. And it can be analyzed with reachability analysis for hazard conditions. The finite state transition and the event/action can be translated into tabular format. Statemate is a tool-set that supports the writing of the statechart and support the simulation of statechart. Z is quite readable and is well suited with sequential functions. The mathematical basis for Z makes the specification executable, and makes verification and analysis of the specification possible. The required characteristics can be proved from the putative theorem. The sequential nature of the specification is conceived to be easily translated into programming language. The timing analysis is not supported by this method, but a lot of tools including commercial version are available. PVS is one of the most rigorous specification languages based on logic. Thus it has good verification capability and safety analysis capability. But the specification written in this logic is a little bit difficult to understand and takes long time to learn compared to other method. The translation into programming language is conceived to be an easy task. This method does not support the timing analysis of the specification. The overall evaluation results are shown in Table 1. The H stands high, M stands for Middle, and L stands for Low. The evaluation result is based on the application experience of each method to VOPT function and review of user' s manual and published applications. At some points the evaluation is subjective. The lightest formal methods of IA and SCR have high readability meaning that domain expert can easily learn to apply the method. They have high traceability and high implementability. But they have low verifiability and safety analysis capability. The heaviest methods of Z and PVS have high verifiability and safety analysis capability and tool support. But with logical notations, the domain expert have to spend more time in learning how to write/read specifications. The two intermediate methods of CPN and Statechart have intermediate characteristics among the lighter and heavier methods. It has moderate readability and implementability with moderate verifiability and safety analysis capability.

This work was performed to evaluate the applicability and feasibility of the formal method to safety software of the nuclear power plants. Our experience is that each method has its own strength and one method alone can not cover all the required characteristics. First the

characteristics need to be analyzed should be defined such as concurrency, timing, liveness. Then appropriate formal method should be selected. Depending on the analysis needed, additional method might be applied. To apply formal methods to safety system in nuclear power plant, it should be economical and should contribute in any of the following aspects; smoothly fitted into the current design process, to be part of SRS in natural way, support in testing or reducing testing, support S/W safety analysis, or make verification and validation easy.

We consider best the Statechart description of the system for external behavioral analysis and tabular notation for analysis of the internal conditions and calculations for algorithms(SCR).

5. Conclusions

Six different formal methods have been applied to the same part of the nuclear power plant safety system application. The application is calculation algorithm intensive. The specifications were evaluated against the criteria based on recommended SRS and recommendations from regulatory body. The evaluation result was that the critical characteristics of the problem should be defined first, then appropriate methods for that critical characteristics have to be selected. Each formal method has its own strength against others. The study shows that, in our application, the tabular method has the strength in the analysis of algorithms and easy understandability. But for the behavioral aspect of the system, the statechart method can be applied. With the appropriate method selected for the problem, the software requirement specification can be prepared. And the benefits of formal methods can be found by applying the methods into more practical problems. Thus it is left as further work to prepare and analyze the requirements specification of the CPCS with tabular method and statechart.

References

1. International Electrotechnical Commission, Software for Computers in the Safety System of Nuclear Power Stations, IEC 880 (1986).
2. NRC, Standard Review Plan, NUREG-0800, Sec. 7.1 Instrumentation and Control (1997).
3. KNFC, Functional Design Requirements for Core Protection Calculator, CENPD-335, Rev. 02-P, (1988).
4. K.L. Heninger, Specifying Software Requirements for Complex Systems: New Techniques and Their Application, *IEEE Trans. On S/W engineering*, Vol. SE-6, NO.1, 2-13 (1980).
5. D. L. Parans and J. Madey, Functional Documents for Computer Systems, *Science of Computer Programming*, 25, No. 1, 41-62 (1995).
6. C.Heitmeyer, et al., Tools for Formal Specification, Verification, and Validation of Requirements, COMPASS' 97 (1997).
7. K. Jensen, Colored Petri Nets. A High-Level Language for System Design and Analysis, Springer-Verlag (1991).
8. David Harel, On Visual Formalisms, *Communications of the ACM*, Vol. 31 No. 5, 514-529 (1988).
9. Antoni Diller, Z An Introduction to Formal Methods, John Wiley & Sons (1994).
10. Judy Crow, et al., A tutorial introduction to PVS, WIFP, Florida (1995).
11. IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specification, IEEE 830-1993 (1993).