A Case Study on the Formal Development of a Reactor Safety System

Terje Sivertsen

Institute for Energy Technology, OECD Halden Reactor Project
P.O. Box 173, 1751 Halden, Norway
e-mail: Terje.Sivertsen@hrp.no

Abstract. The EvalFM project was initiated in order to investigate the applicability of formal methods in the development of safety-critical software-based systems. The overall goal was to explore the strengths and limitations of these methods through practical experience on a realistic example. The present paper presents the main results from the project, related to a case study on the applicability of algebraic specification in the development of a reactor safety system.

1 Introduction

The recent years have witnessed the replacement of many conventional electromechanical process control systems with computer-based systems. This also includes the use of computers in safety-related tasks, e.g. in nuclear power plants and traffic control. The motivation behind this shift towards the use of programmable equipment is manifold. Important benefits are the possibilities for implementing more accurate trip criteria, the improved means for automatic surveillance, as well as simplification of calibration and functional testing during operation. There are however also more pragmatic concerns relating to decreasing availability of equipment and spare parts for the conventional systems and of personnel with appropriate technological expertise. Nevertheless, there has been a certain reluctance to use programmable equipment in safety systems. One reason for that has been the complexity of safety assessment and licensing of these systems, in particular of the embedded software.

Since 1977 the OECD Halden Reactor Project has been actively working in the field of software verification and validation [5]. The Halden Project is an international institution with participants from 19 countries. A main objective of the research activities is that the experiences and results from these activities can be utilized by member countries in real applications. An important part of the research activities within software verification and validation has been the applicability of formal methods in development of safety-related software systems [8]. One concrete achievement has been the establishment of a methodology for the practical application of algebraic specification in formal software development. The methodology allows for using the same language, tool and proof techniques both in specification and design, even down to a "concrete" specification. In the specification phase, the HRP Prover is used to verify and validate the specification, while in the design phase the same tool is used to verify

the correctness of the design steps. The HRP Prover has been developed at the Halden Project to facilitate exploration of animation and theorem proving techniques in formal software development. A guiding principle in the development of the method and tool has been to avoid using heuristics which would complicate the comprehensibility of proofs and specifications. The adequacy of this principle becomes particularly clear when considering the possibilities of accomplishing a formal development of the theorem prover itself. In such an undertaking it is important that the proof techniques can be clearly stated in unambiguous terms and that it can be stated *a priori* how the proof will proceed, and why this indeed will be a sound way to proceed.

Only recently the application of formal software development methods have been seriously considered within the nuclear society. Much discussion and, to a certain degree, controversy arised from the verification and validation of the computer-based Darlington shutdown system [4]. The Darlington Nuclear Generating Station is a new station, consisting of four units 50 miles east of Toronto, Canada. The Darlington shutdown system design includes two independent triplicated shutdown systems. The development used the traditional four-step waterfall diagram, with verification and validation of the final implementation. The Atomic Energy Control Board (Canada) did however not achieve a sufficient level of confidence in the ability of the code to conform to the requirements, and eventually required that the existing functional specification should be replaced by a mathematically rigorous specification. It turned out that with the steps taken to meet the requirements, it was possible to achieve a licensable product. On the negative side, these steps involved considerable effort and difficulty. Following [6], formal methods were in effect used *a posteriori* to model what existed and to demonstrate that the code met the requirements.

There is today a growing consensus within the nuclear society that more practice on the use of formal methods is needed in order to evaluate their applicability [10]. Several independent studies suggest that there is a need for a systematic, rigorous effort in establishing design requirements to minimize errors in the final product [1]. Much of this motivation comes from the limited value of traditional methods. Following [7], "traditional software-development techniques usually do not provide the levels of dependability demanded by safety-critical systems, and the quality criteria are usually such that the amount of testing that is feasible cannot demonstrate that the desired goals have been achieved". There are several important aspects which make the application of software in safety-critical applications fundamentally different from their application in other areas. Safety-critical applications must work when needed, and it is not appropriate to wait for evaluation during use to bring the reliability up to an acceptable level. The realization of the potential benefits of computer-based control and safety systems for nuclear power plants[1] therefore requires *verifying* the reliability of these systems. Traditionally this has been done by means of simulation of the hardware design and exhaustive software testing. It appears however that the use of formal mathematics, in some form, is necessary in order to achieve substantial improvements

in the development of dependable software. On this background, the Nuclear Regulatory Commission (USA) and other instances persuaded the OECD Halden Reactor Project to investigate the applicability of formal methods and in particular algebraic specification and the HRP Prover on a real example related to the control of a nuclear reactor. It was considered important to perform research which could contribute to a clarification of the role and limitations of formal methods when applied on the development of safety-critical software systems. Licensing authorities in general have a particular interest in representative applications of existing formal methods to make decisions on whether the use of formal methods should be required, which formal methods should be used, what is the appropriate way to use them, and what to require to be formally verified.

An example system was selected, based on the following criterion: it should be a realistic, preferably a real, safety-critical system related to nuclear power plant operation. Furthermore, the system should be of reasonable size to keep the effort needed reasonable. After consultation with Sydkraft (Barsebäck NPP) and ABB Atom in Sweden, it was decided to use the computer-based power range monitoring (PRM) system installed at Barsebäck NPP as an example system. The case study did not address ABB's implementation of the example system, but the development of a similar system using formal methods. The formal specification is based on the original customer's requirements document for the system, and is independent of ABB's implementation. The purpose of the PRM system that was of particular interest in this case study is the monitoring of the average power emission of the core. When high power emission is monitored, the system must trip the high level alarms. Based on the requirements document, the EvalFM project produced a formal algebraic specification of one out of four similar subsystems of the PRM system, utilizing a general mathematical tool-kit defined for the method. Finally, the subsystem was designed and implemented in a subset of Pascal.

2 The Structure of the PRM System

The PRM system belongs to class 1E, i.e. systems which automatically effectuate actions in relation to events that may jeopardize safety. PRM stands for Power Range Monitoring, which indicates that the main purpose of the system is to monitor the power emission of the core, represented by the neutron flux. The functioning of such a system is complicated by the fact that the neutron flux varies substantially between different parts of the core. In order to adequately monitor the power emission it is

1. Systems to control critical functions in NPPs are commonly called *reactor safety systems* or *reactor protection systems.*

the requirements document of the example system, it was necessary to abstract away irrelevant details, and otherwise make the requirements applicable as a basis for the specification and design of a software-based implementation. It is however important to note a requirements document typically is used as a basis for the derivation of other specifications for software, hardware, configuration, etc. In this process, experienced developers are usually well trained on treating the words and symbols in the requirements document as specifying functions on a more abstract level.

4 The Identification of Primitive Objects and State Variables

The first step in the specification of the PRM system is to identify the sets of "primitive" objects in the system. The requirements document provides a table describing the interface between the existing equipment and the quoted system:

LPRM Detectors - The components in the existing equipment are first of all 88 analog outputs, i.e. transducers which convert a physical measurement to an electrical signal with voltage between 0 and 10V. It is required that a sampler and A/D converter transforms this voltage to a number in the scale 0 - 125 (%). At this stage in the specification process, it is sufficient to identify these detectors as primitive objects, represented by a new type LPRM.

Probes - In the existing equipment there are also 22 probes, each including 4 specific LPRM detectors. At this stage, it is sufficient to identify the probes as another class of primitive objects, represented by a type PROBE.

For the purpose of the specification of the PRM system, the types LPRM and PROBE need not be further specified. The specification does not need to refer to any particular LPRM detector or probe, and an abstraction from the coded names can therefore be achieved. Such names do however make it easier to communicate the specification to the customer. This can be utilized by using coded names in the *animation* of the specification. The identification of instances might be necessary when directing the design toward a particular implementation, and is definitively necessary in the final implementation of the system.

When the primitive data types have been identified, the next step in the specification process is to identify the "state variables" and their types. These are entities which are essential to an adequate description of a particular state of the specified system, including adjustable parameters and entities representing parts of the "history" of the system. It might be difficult to identify all the variables initially; the need for some of them might be evidenced first in the process of defining the functionality of the system. Fortunately, the need to do some specification first does not jeopardize the formal development.

The values of the different variables are in the specification collected in terms sub(t_1, ..., t_n), where each t_i, $1 \le i \le n$, represents the value of a particular parameter or group (in the form of a compound term) of parameters. Our first task is to identify the parameters which should be represented, and to define the type of the representing terms. The adjustable parameters include the amplification factors for the LPRM signals, the selection of LPRM detectors to be used in the calculation of the APRM signal, the amplification of the APRM signal, and the various alarm limits. In addition to the adjustable parameters, we need to include parameters representing the previous values of the outputs from the subsystem, i.e. the alarms, the calculated APRM value, and values of relevance to the filtering of the APRM value and of the HC-flow (core cooling). The alarms of interest will be the low level and high level alarms associated to the LPRM detectors and the probes, and the alarms L1, H1, ..., H6 associated to the APRM and FLOW/FLUX values (see sections 5.3 and 5.4). We will treat all these alarms as a unit by specifying a data type 'SUB_ALARMS' with the following generator:

```
type SUB_ALARMS
generators
    sub_alarms: FUN(LPRM,BOOL)**2 x FUN(PROBE,BOOL)**2 x
                BOOL**7 -> SUB_ALARMS
```

The specification utilizes the parameterized type FUN(X,Y) of functions, specified in a mathematical tool-kit associated to the method.

In the case study, all functions common to the four subsystems are collected in the specification of a datatype SUB. For each of the four subsystems, it is straight forward to specify an enrichment providing additional functions specific to that subsystem. When the state variables common to the description of the state of the four subsystems have been identified, a generator 'sub' corresponding to the "record" of these variables can be declared:

```
type SUB
generators
    sub: FUN(LPRM,FLOAT) x SET(LPRM) x FLOAT**3 x
         SUB_ALARM_LIMITS x SUB_ALARMS x FUN(LPRM,FLOAT) x
         FLOAT**6 -> SUB
```

The specification utilizes the parameterized tool-kit types SET(X) of sets and FUN(X,Y) of functions. To select the different state variables, an appropriate number of selector functions are introduced:

The current amplification factors are given by the state variable selected by the function 'amp'. These factors are regularly changed in the calibration routines for the LPRM detectors. The amplification of a LPRM signal can be specified by a function which takes the current state variables, the given signals, and the identifier for the detector of interest, and returns the amplified signal from this detector. This gives a function 'amplified_lprm' declared

amplified_lprm: SUB x FUN(LPRM,FLOAT) x LPRM -> FLOAT

and defined

amplified_lprm(S,L,ID) == float_product(apl(L,ID),apl(amp(S),ID)).

The amplified signal from '531K810' is then given by

> amplified_lprm(initSUB,signals,'531K810').

... = *83.6217*

(Note that this would be a very high value for the amplified LPRM signal, since a value of about 60 would correspond to full power).

5.2 LPRM Trip Alarms

To each LPRM detector there is associated an adjustable low and high alarm limit, see Fig. 2.

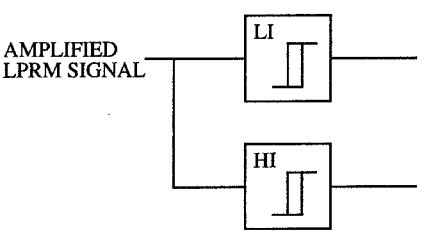

Fig. 2. Low and high LPRM alarms.

These limits can be represented as floating point numbers, and thus the relationship between the detectors and the alarm limits are represented by two selector functions yielding values of type FUN(LPRM,FLOAT).

In order to accomplish hysteresis, we need to compare the amplified LPRM signals not only with the given alarm levels but must also take into account the previous alarm output. Hysteresis about a high alarm limit is accomplished by not resetting the output value of the alarm to 'false' until the input value has dropped below the alarm limit minus the deadband value. Vice versa for low alarm limits. It is convenient to intro-

duce two Boolean functions 'low_alarm' and 'high_alarm', each taking three arguments of type FLOAT and one argument of type BOOL:

- the present signal value;
- the alarm level;
- the deadband value;
- the previous alarm value.

The function 'low_alarm' is declared

low_alarm: FLOAT**3 x BOOL -> BOOL

and defined

```
low_alarm(V,L,DB,B) ==
        if B=true
        then float_less_than(V,float_sum(L,DB))
        else float_less_than(V,L)
        endif.
```

Particular care must be exercised with regard to the required use of the APRM value in connection with the low level alarms for the probes. The requirements document states that a relay should be falling if any of the detectors in a probe falls below the associated limit, if the APRM value exceeds it's H1 limit at the same time, see Fig. 3. (The input K is true if the stated condition of the APRM value is true).

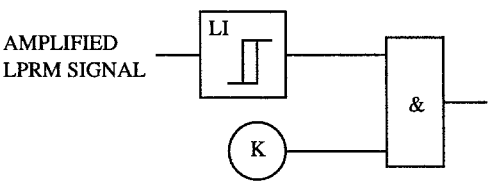

Fig. 3. Condition for low LPRM alarm.

However, the status of this alarm is needed in the determination of the detectors to be used in the calculation of the APRM value (see section 5.3). Apparently, this gives a cyclic definition of the LPRM L1 alarm and the APRM value, because both require the result provided by the other. Evidently, this problem relates to the use of the specification as a basis for the development of a digital system. The solution is easy to find by considering the temporal nature of the system, where the APRM value and other signals are calculated at regular intervals. The problem is therefore resolved by explicitly stating that the APRM value to be used in the calculation of the LPRM L1 alarm is the one calculated in the previous interval. This can be reflected in the specification by using either the previous APRM value or the previous APRM H1 alarm, both of which are represented in the state description of the system.

'aprm_amplific'. The operations of the mean value calculator is therefore one summation, one division, and one multiplication, see Fig. 5.

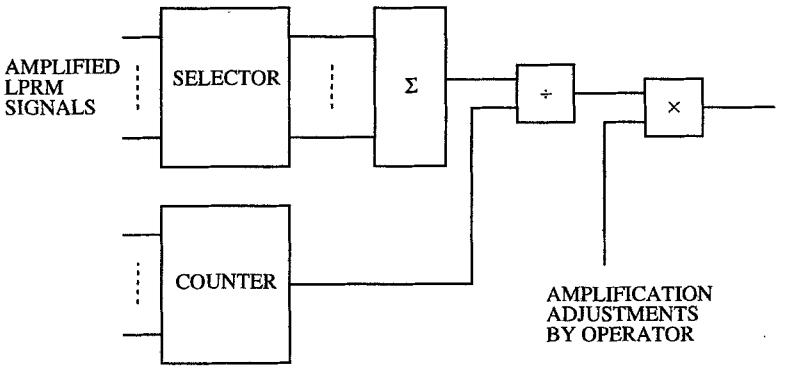

Fig. 5. Calculation of the amplified APRM.

We can specify the operation of the mean value calculator with the function 'amplified_aprm' declared

```
amplified_aprm: SUB x FUN(LPRM,FLOAT) -> FLOAT
```

and defined[1]

```
amplified_aprm(S,L) ==
        float_product(
          float_quotient(
            sum_of_amplified_signals(S,selected(S,L),L),
            integer_float(counter(S,L))),
          aprm_amplific(S)).
```

The function 'sum_of_amplified_signals' calculates the sum of the amplified signals of the elements in a set of detectors, while 'counter' returns an integer representing the number of selected detectors not having too low signal values.

5.4 FLOW/FLUX

The range of the HC-flow (core cooling) is 0-8800 kg/s, which can be expressed in percentages of full operating power. The requirements document does not explicitly provide the transition factor from kg/s to percentages, but indicates that it is a linear transition. In the calculation of the FLOW/FLUX (FF) signal, the HC-flow signal

1. The definition assumes no division by zero. This can be modified by introducing a conditional test on the value of counter(S,L).

should have an upper limit. In the requirements document, the FF signal is defined to be the difference between the APRM and the limited HC-flow. Given a value for the HC-flow in the range 0-8800 kg/s, the FF signal will be specified by introducing a function 'flow_flux' declared

```
flow_flux: SUB x FUN(LPRM,FLOAT) x FLOAT -> FLOAT
```

and defined

```
flow_flux(S,L,HC) ==
        float_difference(
            amplified_aprm(S,L),
            limited_hc(float_product(HC,scale_factor(S)))).
```

Given a HC-flow of 4843 kg/s, the current FF signal is given by:

```
> flow_flux(initSUB,signals,4843.0).
... = 37.9312
```

The requirements document defines a filtered FF signal by means of two Laplace transfer functions. These transfer functions give rise to a two-step and a one-step difference equation. Using bilinear transformation, we find the following difference equation for transfer function 2:

$$c(k+1) = \frac{r(k+1) + r(k) + (11/T - 1)c(k)}{11/T + 1}$$

Using this equation and the state variables representing the previous I/O values, transfer function 2 can be specified by introducing a function 'transfer2' declared

```
transfer2: SUB x FLOAT -> FLOAT
```

and defined in accordance to the one-step difference equation. In a similar way, we can introduce a function 'transfer1' corresponding to the two-step difference equation resulting from transfer function 1. The filtered FF signal can now be specified by introducing a function 'filtered_flow_flux' declared

```
filtered_flow_flux: SUB x FUN(LPRM,FLOAT) x FLOAT -> FLOAT
```

and defined

```
filtered_flow_flux(S,L,HC) ==
    float_difference(
        transfer1(S,amplified_aprm(S,L)),
        transfer2(S,limited_hc(S,float_product(HC,scale_factor(S))))).
```

toward the implementation language until the design is sufficiently close to facilitate a more or less direct transformation. Alternatively, the implementation can be represented as an abstract syntax tree in algebraic specification. In that case, the correctness of the implementation can in principle be reduced to the correctness of the translation from the abstract syntax tree to the actual code. The development of an integrated environment for the methodology would include the establishment of correct transformation rules and a mechanism for performing (parts of) the actual transformation from design to code.

The development method based on algebraic specification and the HRP Prover allows for variation on the design of the specified system, both with respect to the implementation language, and to the efficient and safe use of the data structures provided by this language. This was also demonstrated in the EvalFM project, where two alternative designs were given which put different emphasis on the efficiency of the implementation. The case study demonstrates that the design of the example system could be directed to an efficient implementation in a safe subset of a specific programming language (in this case Pascal).

7 Further Work

The specification of the example system was written in a formal specification language without the support of any graphical presentation techniques. The comprehensibility of this specification could probably be improved by adopting such techniques, an approach which appears to be feasible because of the close correspondence between parts of the specification and the functional descriptions in the customer's requirements document. It is expected that the adoption of graphical techniques would make it easier to inspect the specification by domain experts with limited knowledge about formal methods. This is important, because part of the specification process is the interaction between the domain experts and the software developers. The adoption of graphical techniques would probably also make it easier for industrial organisations to integrate formal methods in their own development projects.

The application of graphical techniques to the case study is currently investigated in a co-operative project between OECD Halden Reactor Project and ENEA, Italy. One of the project assignments is to apply the IPTES methodology [2] on the PRM system. Much of the motivation behind the IPTES project was the observation that software developers are generally reluctant to using formal notations. The idea was to support widely used graphical notations with formal methods. As a concrete graphical notation, the IPTES project has used the SA/RT (Structured Analysis/Real Time), while high-level Petri nets and an executable subset of VDM (Vienna Development Method) have been chosen as the underlying formal notations. The strategy implemented in the

IPTES tools is to translate an SA/RT model into a high-level Petri net, and to use VDM for the formal analysis of the model.

The Pascal-program produced in the EvalFM project is utilized at the OECD Halden Reactor Project in investigations on the PIE method [9]. This method combines random testing with sensitive analysis, i.e. the analysis of how sensitive the revelation of a certain fault is to the choice of test data. A set of artificial faults, called *mutants*, are seeded into the program, which is executed back-to-back with an incorrupted program and with a large number of test data. The aim of the PIE analysis is to increase the confidence in a program in cases where no failures occur during testing.

An important extension of the case study would be to study the possibility of raising the abstraction level, combined with a consideration of a larger part of the system context. Through the research activities on formal process models, the Halden Project has been developing methods, principles and tools for formal representation of plant knowledge [3]. The goal of the work on formal process models has been to implement computerised support of the design and verification of control strategies. Examples are automatics and operating procedures. The approach has been to model process plants and plant automatics in a unified way to allow verification and computer aided design of control strategies. The methods are based on transforming as directly as possible the information in P&I diagrams and other plant knowledge into a knowledge base to be used in automated reasoning. The knowledge representation primitives must make it possible to utilise different types of knowledge according to what knowledge is available and to the different levels of abstraction used in solving the control problems. The scope of the EvalFM project was to take a detailed requirements document as a starting point, a situation which appears to be quite common in relation to nuclear power class 1E systems. The proposed extension would be particularly applicable if the project involved the production of a requirements specification, and not merely the formalisation of the existing specification. The practical impact on the development process would appear to be quite different in such an extension of the project.

The EvalFM project and many other case studies on the application of formal methods demonstrate that the practical usefulness of these methods depends on the availability of support tools, including powerful theorem provers for the development of proofs of safety invariants and correctness of design. A common objection to the use of formal methods has been that the supporting tools are themselves susceptible to programming errors. In addition comes the imperfectness of compilers, operating systems and computer hardware. The OECD Halden Reactor Project currently addresses these problems through the application of formal methods in the development of a new version of the HRP Prover.

- Industrial use of algebraic specification, as well as of formal methods in general, would require the provision of a reliable integrated environment.
- A large part of the effort involved in a formal development project is invested in the production and assessment of the specification. In most cases, only a minor part is invested in the actual implementation.

9 REFERENCES

[1] L. Beltracchi, NRC Research Activities, in: D.R. Wallace, B.B. Cuthill, L.M. Ippolito and L. Beltracchi (eds.), *Proc. Digital Systems Reliability and Nuclear Safety Workshop*, Sep. 13-14, 1993, NUREG/CP-0136, United States Nuclear Regulatory Commission, Washington DC (1994) 31-45.

[2] S. Bologna (ed.), *Special Issue: Incremental Prototyping Technology for Embedded Real-Time Systems*, Real-Time Systems **5**, 2/3 (May 1993).

[3] S. Bologna, T. Sivertsen and H. Välisuo, Rigorous Engineering Practice and Formal Reasoning of Deep Domain Knowledge - The Basis of Dependable Knowledge Based Systems for Process Plant Control, International Journal of Software Engineering and Knowledge Engineering **3** (1993) 53-98.

[4] R.H. Crane, Experience Gained in the Production of Licensable Safety-Critical Software for Darlington NGS, in: *Proc. Methodologies, Tools, and Standards for Cost-effective, Reliable Software Verification and Validation*, EPRI, Palo Alto, CA, USA (Jan. 1992).

[5] G. Dahll, T. Sivertsen, Halden Project Activities on Software Dependability, in: D. Ruan (ed.), *Intelligent Technologies for Man-Machine Interaction at the OECD Halden Reactor Project, Special Presentations of FLINS'94* (SCK-CEN Nuclear Research Centre, Belgium, 1994) 27-38.

[6] S. Gerhart, D. Craigen and T. Ralston, Experience with Formal Methods in Critical Systems, IEEE Software (Jan. 1994) 21-39.

[7] J. Knight and B. Littlewood, Critical Task of Writing Dependable Software, IEEE Software (Jan. 1994) 16-20.

[8] T. Sivertsen, Formal Methods and Their Applicability in the Development of Safety Critical Software, *Proc. IAEA Technical Committee Meeting on Advanced Control and Instrumentation Systems in Nuclear Power Plants: Design, Verification and Validation* (Helsinki/Espoo, Finland, 20-23 June 1994).

[9] J.M. Voas, PIE: A Dynamic Failure-Based Technique, IEEE Trans. Soft. Eng. **18** (Aug. 1992) 717-727.

[10] D.R. Wallace, B.B. Cuthill, L.M. Ippolito and L. Beltracchi (eds.), *Proc. Digital Systems Reliability and Nuclear Safety Workshop*, Sep. 13-14, 1993, NUREG/CP-0136, United States Nuclear Regulatory Commission, Washington DC (1994).