COMPUTING SCIENCE

Formal Techniques for Requirements Analysis for Safe Reactor Control

A. Saeed, R. de Lemos and T. Anderson

TECHNICAL REPORT SERIES

No. 430 May, 1993
Formal Techniques for Requirements Analysis for Safe Reactor Control

A. Saeed, R. de Lemos and T. Anderson

Abstract

Requirements analysis plays a vital role in software development, since defective requirements specifications may introduce faults during subsequent phases of software development which can lead to accidents. In this paper we discuss a methodology, based on formal notations, to support requirements analysis for safety-critical systems. The proposed methodology consists of a framework whose phases are based on levels of abstraction which follow from a general structure for process control systems, a set of notations appropriate for the issues to be analysed at each phase of the framework, and a hierarchical structure for the product of the analysis. The intention is to locate and remove faults during the requirements phase, rather than later in development during the operational lifetime of the system. Some aspects of the methodology are illustrated using an example based on a nuclear reactor system.
Bibliographical details

SAEED, Amer

Formal Techniques for Requirements Analysis for Safe Reactor Control
[By] A. Saeed, R. de Lemos and T. Anderson


(University of Newcastle upon Tyne, Computing Science, Technical Report Series, no.430)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
DE LEMOS, Rogerio Sergio Neves
ANDERSON, Thomas

Abstract

Requirements analysis plays a vital role in software development, since defective requirements specifications may introduce faults during subsequent phases of software development which can lead to accidents. In this paper we discuss a methodology, based on formal notations, to support requirements analysis for safety-critical systems. The proposed methodology consists of a framework whose phases are based on levels of abstraction which follow from a general structure for process control systems, a set of notations appropriate for the issues to be analysed at each phase of the framework, and a hierarchical structure for the product of the analysis. The intention is to locate and remove faults during the requirements phase, rather than later in development during the operational lifetime of the system. Some aspects of the methodology are illustrated using an example based on a nuclear reactor system.

About the author

Dr. A. Saeed is a Research Associate in the Department of Computing Science.

Mr. R. de Lemos is a Research Associate in the Department of Computing Science.

Professor T. Anderson is a Professor in the Department of Computing Science.

Suggested keywords

COMPUTING SYSTEMS  FORMAL NOTATIONS  REACTOR CONTROL
REQUIREMENTS ANALYSIS  SAFETY-CRITICAL SYSTEMS

Suggested classmarks (primary classmark underlined)
Dewey (18th): 001.6425  658.47
U.D.C. 681.322.06  519.718
Abstract

Requirements analysis plays a vital role in software development, since defective requirements specifications may introduce faults during subsequent phases of software development which can lead to accidents. In this paper we discuss a methodology, based on formal notations, to support requirements analysis for safety-critical systems. The proposed methodology consists of a framework whose phases are based on levels of abstraction which follow from a general structure for process control systems, a set of notations appropriate for the issues to be analysed at each phase of the framework, and a hierarchical structure for the product of the analysis. The intention is to locate and remove faults during the requirements phase, rather than later in development or during the operational lifetime of the system. Some aspects of the methodology are illustrated using an example based on a nuclear reactor system.

Keywords: safety-critical systems, computing systems, requirements analysis, formal notations, reactor control.

Introduction

As the use of computers in critical applications becomes more widespread and the level of criticality of the roles performed by the computers increases, a significant factor in the reliability and safety of such applications will be the quality of the software. The reliability and safety requirements for these systems are orders of magnitude higher than can be achieved, even using the best current methods, for complex software systems /Moser 90/. Improved methods for the software development of such systems are urgently needed. Within the context of software development this paper is only concerned with the phase of requirements analysis. This phase plays a vital role in software development, since a defective requirements specification will corrupt subsequent stages of software development, introducing faults in the final system which can lead to accidents /Leveson 86/.

Two key characteristics of the requirements phase are: it is a multidisciplinary activity that necessitates the analysis of complementary views of a system, and it must deal with a large volume and wide diversity of information which necessitate the employment of an organizing principle (i.e. an approach to structure the information to promote traceability, verification and validation) and the use of many different notations. When developing safety-critical systems, a vital concern is to ensure that the probability that an accident will occur is kept below an acceptable level. The usual approach to ensuring safety is to strive to identify all the hazards (circumstances from which an accident might ensue) and then apply techniques to reduce the probability of these hazards occurring. The level of effort that should be employed to handle a
hazard depends on the risk associated with that hazard. Identification of hazards and risk analysis is performed using standard system safety techniques, such as HAZOPS /Lawley 73/ and Fault Tree Analysis /Vesely 81/. The requirements analysis phase forms the link between such system safety activities and software design; therefore a practical approach to requirements analysis must be able to incorporate the results of traditional safety and hazard analysis and promote traceability between system safety and software safety.

One approach that has been advocated for improvement in the dependability of safety—critical systems is the utilization of formal methods /MoD 91/. Formal is a term applied to a notation or technique if it is amenable to mechanical manipulation according to some calculus that can be understood mathematically. A method is a set of procedures and guidelines for the application of a notation or technique to system development. When the notation or technique is formal, the method is referred to as a formal method.

A general advantage of formal methods in the development of a specification is that they contribute to improving the understanding of the specification by applying rigour to the reasoning process. During the requirements phase, formal methods provide support in a number of specific ways: the expression of unambiguous specifications, the development of requirements specifications by formal refinement, and the opportunity to check for consistency and completeness. The application of formal methods to requirements analysis aims to locate and remove faults (which can lead to accidents) introduced in this phase, that are typically identified in subsequent phases of development or during the operational lifetime of the system.

**Principles of the Methodology**

The proposed methodology consists of a framework with distinct phases of analysis, a set of notations appropriate for the issues to be analysed at each phase of the framework, and a hierarchical structure for the product of the analysis. The intention is to define abstraction levels in which the analysis of requirements can be performed in terms of specific issues of the system, thereby providing a systematic approach. The methodology reflects the following four key principles.

- **Process Control Structure.** The phases of the framework, based on abstraction levels for the analysis of the requirements, follow directly from a general structure adopted for process control systems. In this structure, these systems are partitioned into three different components: the operator, the controller, and the physical process (or plant). The controller has two interfaces, the plant interface and the operator interface; the plant interface is made up of sensors and actuators. The system exists in an environment, i.e. that part of the rest of the world which may affect, or be affected by, the system.

- **Formality.** Formal support for the different activities performed during requirements analysis demands the utilisation of a set of formal notations and techniques whose features and expressive power should match the characteristics of the activities. Selecting an appropriate formal notation for the activity allows emphasis to be placed on pertinent characteristics of the system, enabling a notation to work to its own strengths. In the
proposed methodology we employ formal notations in accordance with the characteristics of the system to be analysed, during the different phases of the framework, and define the linkage between the different notations, resulting in a coherent approach.

- **Separation.** The methodology adopts the approach of separating the mission from the safety requirements; the latter being those requirements which focus on the elimination and control of hazards, and the limitation of damage in case of an accident. This distinction during the requirements phase is essentially a logical separation, and has the following benefits: the resolution of potential conflicts, detection of omissions and inconsistencies between the mission and safety issues, the ability to focus on the safety issues, and the simplification of safety certification. The separation of mission from safety can be enforced at the physical process level (as for example in the shut down systems of nuclear reactors), at the controller level by implementing separate mission and safety controllers, or by the application of design techniques, such as “firewalls”, which are intended to prevent the logical mission controller from interfering with the logical safety controller. However, it is not the intention of this principle to force a particular process structure on the implementation. Rather, the aim is to enable formal notations to be targeted at the safety—critical behaviour of a system.

- **Hierarchical Organization.** The safety specifications produced at each phase of the framework are organised through a safety specification hierarchy (SSH), which facilitates the quality assessment of the requirements specifications. The SSH records the different options and facilitates a comparison of these options with respect to predefined criteria. The assessment is performed by qualitative and quantitative means in order to obtain high confidence (assurance) that the level of risk is acceptable. Qualitative analysis aims to increase assurance in the safety specifications by establishing consistency between the specifications across each level of the hierarchy and checking the maintenance of safe behaviour down the hierarchy, by employing verification and validation techniques. The quantitative assessment evaluates the impact of violations in assumptions on the validity of the safety specifications, by the construction of statistical models relating the violations to the probability of hazards.

**Behaviour Description**

In order to describe the behaviour of systems at different levels of abstraction, we adopt an event/action model (E/A model) /Anderson 92/ as a common foundation for models of system behaviour. The E/A model is based on primitive concepts such as events, actions and states, and the concept of a timeline. The utilization of these concepts provides flexibility, enabling descriptions to be given of system behaviour ranging from the activities of the physical entities of the plant to the temporal ordering of the computational tasks of the controller. In general terms, the main features of the E/A model are its primitive concepts can be expressed in different notations, it supports both discrete and dense time structures, and it can depict timing constraints graphically.
The primitive concepts of the E/A model are described in terms of a set of variables representing the state of the system, and are defined as follows. A condition is a change in the value of a system predicate (a predicate over the system variables). An event is a temporal marker of no duration which marks the time point at which a condition becomes true or false; it is represented as a cut in the timeline. A state is a mapping from a subset of system variables to a set of values from the ranges of the variables in the subset. An action is a basic unit of activity, with an implied duration for its execution. Apart from the events, actions and states to describe the behaviour of systems, the E/A model also takes into account the timing uncertainties associated with them.

These primitive concepts are related to the timeline by three types of primitive functions: event—occurrence, state—holding and action—execution. However, to formally express the primitive concepts of the E/A model and to facilitate formal analysis, we employ a Predicate Event/Action notation (PEA notation). In the PEA notation the primitive functions are related to predicates over system variables.

![Diagram](image.png)

**Figure 1. General structure for process control systems.**

The set of variables of the system, system variables \( (V) \), through which the system behaviour is described, are classified into three subsets matching the components of a process control system as shown in figure 1: physical variables \( (V_P) \), controller variables \( (V_C) \), and operator variables \( (V_O) \). Behaviour at the plant interface comprises the physical variables that are being directly monitored \( (V_{PM}) \) or controlled \( (V_{PC}) \) and the controller variables that provide input to the controller \( (V_{CI}) \) or generate output from the controller \( (V_{CO}) \). Each variable \( (v_i) \) of the set of system variables \( (V=\{v_1, v_2, v_3, \ldots\}) \) can be represented as a function which maps the timeline \( (T) \) into the range of values of that variable \( (VR_{v_i}) \). That is, \( v_i(t): T \rightarrow VR_{v_i} \). The range
of values \( VR_v \) is partitioned into two subranges: the anticipated values (\( VR^a_v \)) and unanticipated values (\( VR^u_v \)): \( VR_v = VR^a_v \cup VR^u_v \). Defining these two subranges for each of the system variables enables the specification of the standard, exceptional and failure behaviours of the system and its components.

**Case Study: Controlling the Temperature in a Nuclear Reactor**

In order to clarify some aspects of the methodology, an example based on a simplified nuclear reactor control system was selected as a case study. Specifically, the case study will serve to illustrate how the levels of abstraction partition the analysis into smaller domains, and how the E/A model can be used to describe system behaviour.

The case study involves a system used to control the temperature of a nuclear reactor by moving two independent control rods; lowering a control rod slows down the nuclear reaction, decreasing the temperature by at least a predefined amount. A complete shut-down of the reactor is required under two circumstances: if the temperature of the reactor is not decreased by (at least) a specified amount as the result of lowering a control rod, or if temperature of the reactor exceeds a high temperature threshold. The relative position of the two control rods should always be within specified bounds, they should not be moved simultaneously, and there is a minimum time between any two movements of the rods. This example is a modified version of two examples discussed in the literature /Jaffe 91; Jahanian 87/.

**Methodology for Requirements Analysis**

In this section, we present an overview of our methodology, focusing on an exposition of the framework. Our general framework for the requirements analysis of safety-critical systems is shown in figure 2. This is obtained by first, applying the principle of separating the mission requirements from the safety requirements, and second, subdividing the analysis of the safety requirements according to levels of abstraction matching the structure adopted for process control systems.

The phases of analysis are convenient notions when describing the framework and show that the level of abstraction changes from descriptive to operational as the analysis proceeds. The results of a phase can be influenced by feedback from subsequent phases, since these can take additional information into account. For each phase of the framework, we present the features that a suitable formal notation should possess, and suggest some candidate notations.

**Conceptual Analysis**

The objective of this phase is to produce an initial and informal statement of the aim and purpose of the system — particularly the relationship between the system and environment. What is meant by “safety” for the system must also be determined during this phase (i.e. the failure behaviours of the system which constitute accidents must be identified). As a product of the Conceptual Analysis phase, we obtain the Safety Requirements, enumerating the potential accidents. The accidents of a system would be identified using standard techniques (e.g. Preliminary Hazard Analysis). The identified accidents provide a basis from which a
distinction between the mission and safety issues of the system can be drawn. Another activity
to be performed during this phase is the identification of the modes of operation of the system;
these are classes of states that group together related operational functions.

For the conceptual phase, we are seeking notations which would allow the reasoning and
representation of a very high level description of the preferred behaviour of the system within
its enclosing environment. Preliminary work suggests that it may be possible to introduce
additional formal support by employing notations representing causal relationships between
states of the system and environment.

Figure 2. A framework for requirements analysis.

**Safety Plant Analysis**

The objective of this phase is the identification of those real world properties which are
relevant to the Safety Requirements. An essential starting point for the plant analysis is to
construct a model of the physical process that captures its characteristics as dictated by physical
laws and rules of operation. The plant analysis must also support the identification of hazards
and then express these as a logical condition. As the product of the plant analysis we obtain the
*Safety Plant Specification* which contains the *safety constraints* (a safety constraint is a condition
over the physical process that is the negation of a hazard modified to incorporate safety
margins) and the *safety strategies* (a safety strategy is a scheme to maintain a safety constraint
and is defined as a set of conditions, in terms of controllable factors, over the physical process. The conjunction of all of the safety constraints of a system characterises the safe behaviour for that system.

During the plant analysis phase, we have to model and represent systems which contain both continuous and discrete variables and hence both differential equations and discrete mathematics are needed. After the analysis of the continuous aspects of the system is performed, and the critical thresholds defined, a model of the plant is constructed and the safety specifications developed in an incremental manner. Both the modelling and specification activities suggest that a descriptive formalism that enables behaviour to be specified in terms of axioms over a model of the system, for example Temporal Logic or THL /Saeed 90/, is most appropriate. Such formalisms allow specifications to have a conjunctive nature in the sense that new requirements can be added without the need to reconstruct the full specification.

**Nuclear Reactor Example: Safety Plant Analysis**

**Physical Variables**

The physical process consists of a reactor containing two control rods — Rod1 and Rod2. To model the reactor behaviour, three physical variables (which are functions from a time-domain T) are identified: Temp(t) — temperature of the reactor (°k); PosRod1(t), PosRod2(t) — position of Rod1 and Rod2 (cm).

<table>
<thead>
<tr>
<th>Variables</th>
<th>Class</th>
<th>VR(^a)</th>
<th>VR(^a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td>(V_{PM})</td>
<td>(Temp (\in) (R_+)</td>
<td>minTemp (\leq) Temp (\leq) maxTemp)</td>
</tr>
<tr>
<td>PosRod1, PosRod2</td>
<td>(V_{PC})</td>
<td>(PosRod (\in) (R_+)</td>
<td>minPos (\leq) PosRod (\leq) maxPos)</td>
</tr>
</tbody>
</table>

**E/A Model of Plant**

For the variable Temp(t) we identify three states: StabTemp(i)\(\uparrow\), LowTemp(i)\(\uparrow\) and HighTemp(i)\(\uparrow\) (StabTemp(i)\(\uparrow\) is shorthand for StabTemp(i)\(\uparrow\)StabTemp, \(\downarrow\)StabTemp, where the index i refers to the occurrence number and the terms \(\uparrow\)StabTemp, \(\downarrow\)StabTemp refer to the conditions that mark the boundaries of the state). These states characterise the temperature: within a stable range, lower than the stable range and higher than the stable range, respectively. For the variables PosRod1(t) and PosRod2(t) we identify the state InitPRods(i)\(\uparrow\) in which the rods are at their minimum positions (in the sequel, the index i will be suppressed when not required). These states can be related to the physical variables by the PEA notation. The bar operator ("|"") is applied to a system predicate to denote the corresponding event, that is \(|(sp(t))|\) (or, \(|(sp)@t|\) denotes the event for system predicate sp(t) and \(|i(sp(t))|\) denotes the ith occurrence of the event. As an example, the definition of InitPRods\(\uparrow\) is presented below.

\[
\uparrow\text{InitPRods}@t \equiv |(\text{PosRod1} = \text{minPos} \land \text{PosRod2} = \text{minPos})|@t.
\]

\[
\downarrow\text{InitPRods}@t \equiv |(\text{PosRod1} \neq \text{minPos} \land \text{PosRod2} \neq \text{minPos})|@t.
\]
Where $\uparrow Init PRods @ t$ and $\downarrow Init PRods @ t$ are respectively the true and false events for the state $Init PRods()$.

For $Rod 1$ we define two actions: $DMov Rod 1(i)( )$ and $UMov Rod 1(i)( )$; these actions model the rod being moved down and up by a specified displacement $Res Mov Rods$. These actions can be used to define the action $Mov Rod 1(i)( )$, modelling either movement of $Rod 1$, as follows:

$$\forall i \in I^+: \forall t_1, t_2 \in T: \ Mov Rod 1(i) @ (t_1, t_2) \iff \exists j \in I^+: DMov Rod 1(j) @ (t_1, t_2) \lor UMov Rod 1(j) @ (t_1, t_2)$$

where $Mov Rod 1(i) @ (t_1, t_2)$ means that the $i$th execution of $Mov Rod 1(i)( )$ starts at time point $t_1$ and finishes at time point $t_2$.

Similarly for $Rod 2$ we define the actions: $DMov Rod 2(i)( )$, $UMov Rod 2(i)( )$ and $Mov Rod 2(i)( )$. In addition we define the following actions: $DMov Rods(i)( )$, $UMov Rods(i)( )$ and $Mov Rods(i)( )$; these represent up movement, down movement and movement, respectively, of either rod. We also define the action $Rel Rods()$ which represents the release of both rods.

**Rules of Operation**

Two examples of how the above actions can be used to express rules of operation are presented.

**ro1.** The time interval between two consecutive movements of any of the two rods should be at least 30 time units.

$$\forall i \in I^+: \forall t_1, t_2 \in T: [\uparrow Mov Rods(i) @ t_1 \land \uparrow Mov Rods(i+1) @ t_2 \Rightarrow t_2 \geq t_1 + 30]$$

where $\uparrow Mov Rods(i) @ t$ means that the $i$th execution of $Mov Rods(i)( )$ starts at time point $t$.

**ro2.** A down movement of either control rod should have the effect that the temperature decreases by at least a pre-defined value ($Res Temp$).

$$\forall i \in I^+: \forall t_1, t_2 \in T: [DMov Rods(i) @ (t_1, t_2) \Rightarrow (Temp(t_1) \geq Temp(t_2) + Res Temp)].$$

**Safety Strategy**

As an example of a safety specification, we present a safety strategy which is based on shutting down the reactor by releasing the rods according to the following rules. Each rule is sufficient for the release of rods.

**ss1.** If the temperature does not decrease by a pre-defined value ($Res Temp$) following a down movement of the rods, both rods must be released within a duration of $\Delta Release$.

$$\forall i \in I^+: \forall t_1, t_2 \in T: [DMov Rods(i) @ (t_1, t_2) \land (Temp(t_2) > Temp(t_1) - Res Temp) \Rightarrow \exists t_3 \in T: \uparrow Rel Rods @ t_3 \land t_3 \leq t_2 + \Delta Release].$$

**ss2.** If the temperature of the reactor exceeds a threshold ($SD Temp$), both rods must be released within a duration of $\Delta Release$.

$$\forall t_1 \in T: [(Temp \geq SD Temp) @ t_1 \Rightarrow \exists t_2 \in T: \uparrow Rel Rods @ t_2 \land t_2 \leq t_1 + \Delta Release].$$
Safety Interface Analysis

The objective of this phase is to delineate the plant interface, and specify the behaviour that must be exhibited at that interface. In this phase we perform an analysis of the safety strategies to determine their robustness in the presence of imperfections (such as inaccuracies in sensors) or failures in the components of the plant interface. This phase leads to the production of the Safety Interface Specification, containing the robust safety strategies (a robust safety strategy is a modification of a safety strategy, incorporating properties of sensors and actuators). As for the plant phase, these characteristics suggest the employment of a descriptive formalism.

Nuclear Reactor Example: Safety Interface Analysis

Interface Variables

The interface consists of a thermometer \( \text{SenTemp} \) to measure the temperature of the reactor, and two types of actuators: \( \text{ActRel} \) to release the rods and \( \text{ActMov} \) to move the rods. The behaviour at the interface is modelled by four variables: \( \text{CiTemp}(t) \) — temperature of the reactor as measured by \( \text{SenTemp} \); \( \text{CoMov1}(t), \text{CoMov2}(t) \) — inputs to \( \text{ActMov} \) to control the movement of the rods and \( \text{CoRel}(t) \) — input to \( \text{ActRel} \) to control the release of the rods. The input/output variables for \( \text{SenTemp} \) and \( \text{ActRel} \) are illustrated in figure 3.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Class</th>
<th>( \text{VR} )</th>
<th>( \text{VR} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{CiTemp} )</td>
<td>( V_{CI} )</td>
<td>( { \text{CiTemp} \in \mathbb{R}_+ \mid \text{minCiTemp} \leq \text{CiTemp} \leq \text{maxCiTemp} } )</td>
<td>( R_+ - \text{VR} )</td>
</tr>
<tr>
<td>( \text{CoMov1}, \text{CoMov2} )</td>
<td>( V_{CO} )</td>
<td>( { \text{idle, up, down} } )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( \text{CoRel} )</td>
<td>( V_{CO} )</td>
<td>( { \text{on, off} } )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

![Diagram](image)

Figure 3. Plant interface variables.

E/A Model of Interface

For \( \text{CoMov1}(t) \) we define the action \( \text{CoDMovRod1}(i) \) which represents the controller action that performs (via \( \text{ActMov} \)) a down movement of \( \text{Rod1} \).
$\uparrow CoDMovRod1(i)@t \equiv I_i(\text{CoMov1=down})@t.$

$\downarrow CoDMovRod1(i)@t \equiv I_i(\text{CoMov1=idle})@t.$

Where $\uparrow CoDMovRod1(i)@t$ and $\downarrow CoDMovRod1(i)@t$ are respectively the start and finish events for the action $CoDMovRod1(i)()$.

Similarly, we define $CoDMovRod2(i)()$ for the down movement of Rod2; and $CoDMovRods(i)()$ for the down movement of any rod.

**Properties of Sensors/Actuators**

$srl$. If the temperature is within the anticipated range, the difference between the thermometer reading and the actual temperature of the reactor is bounded by $\Delta Temp$.

$$\forall t \in T: [\text{Temp}(t) \in VR^u_{temp} \Rightarrow |\text{Temp}(t) - \text{CiTemp}(t)| \leq \Delta Temp].$$

Temperature constants for the controller (adjusted to allow for sensor inaccuracy).

- $\text{CiResTemp} = \text{ResTemp} + 2\Delta Temp.$
- $\text{CiSDTemp} = \text{SDTemp} - \Delta Temp.$

$ar1$. The delay between a request to release the rods and the start of $\text{RelRods}$ is bounded by $\Delta Rel$.

$$\forall t_1 \in T: [((\text{CoRel} = \text{on})@t_1 \Rightarrow \exists t_2 \in T: \uparrow \text{RelRods}@t_2 \land t_2 - t_1 \leq \Delta Rel].$$

**Robust Safety Strategy**

The robust safety strategy activates $\text{ActRel}$ to release the rods according to the following rules.

**rss1.** If the observed temperature is not decreased by a significant value ($\text{CiResTemp}$) following a controller action for the down movement of any rod, then $\text{ActRel}$ is activated.

$$\forall i \in I^+: [\forall t_1, t_2 \in T: CoDMovRods(i)@t_1 \land (\text{CiTemp}(t_2) > \text{CiTemp}(t_1) - \text{CiResTemp}) \Rightarrow ((\text{CoRel} = \text{on})@t_2)]$$

**rss2.** If the temperature observed by the controller exceeds a predefined threshold ($\text{CiSDTemp}$), then $\text{ActRel}$ is activated.

$$\forall t \in T: [(\text{CiTemp} \geq \text{CiSDTemp})@t \Rightarrow ((\text{CoRel} = \text{on})@t].$$

**Safety Controller Analysis**

During this phase we establish a top level organization for the controller in terms of the properties of its components, and their interactions. This phase leads to the production of the Safety Controller Specification, containing the safety controller strategies (a safety controller strategy is a refinement of a robust safety strategy incorporating the components of the controller). The Safety Controller Specification is the final product of the requirements analysis and provides the basis for subsequent development of the software.
The activities to be performed in this phase include an analysis of the high level architectural design of the controller, and modelling of the interactions between its components and the operations that must be performed by these components. To perform these activities we need an operational formalism which enables concurrency and non-determinism to be explicitly specified, such as Predicate—Transition nets (PrT nets) /Genrich 87/ and Statecharts /Harel 87/. The general approach followed in the modelling of the system is to maintain a clear separation between the models of the physical process, interface and controller. The advantage of adopting this approach is that three models can be separately developed and modified, and the behaviour of the physical process can be seen to correspond to the sequence of control commands issued by the controller.

**Nuclear Reactor Example: Safety Controller Analysis**

The PrT net model of the nuclear reactor system is shown in figure 4. The model of the physical process includes only those states which are relevant from the safety viewpoint. The model of the interface represents logical sensors and actuators. Finally, the model of the controller represents the safety related activities which implement the robust safety strategies, and the mission activities related to the downwards movement of the rods, since these are fundamental for the realization of the safety controller strategies.

The predicates which are most important for understanding the PrT net model in figure 4 are the following:

- \( P_1() \) — the controller considers the rods to be moving down;
- \( P_2() \) — the controller considers the rods to be stopped;
- \( P_3() \) — the controller is allowed to move the rods down;
- \( P_9(Temp, tstamp) \) — the rods are moving down;
- \( P_{10}(Temp, tstamp) + (Temp, tstamp) \) — the rods are stopped;
- \( P_{11}() \) — the reactor temperature does not decrease by a significant amount after the rods are moved down;
- \( P_{12}() \) — the temperature of the reactor has exceeded \( SDTemp \);
- \( P_{13}() \) — the rods are down and the reactor is Shut—Down;
- \( AM.in() \) — the controller signals the actuator to start/finish the down movement of the rods;
- \( AM.out() \) — the actuator starts/finishes to move the rods down;
- \( AR.in(CoRel) \) — the controller signals the actuator to release the rods;
- \( AR.out(ReI) \) — the actuator releases the rods;
- \( ST.in(Temp) \) — the temperature of the nuclear reactor is read;
- \( ST.out(CiTemp) \) — the sensor provides the temperature to the controller;
The relations on transitions \( tr_1 \) and \( tr_2 \) model the rules of operation associated with the movement of the rods. From the relations on the transitions \( tr_3 \) and \( tr_4 \) the controller detects when the nuclear reactor should be shut-down. These relations refer to the implementation by the controller, respectively, of rules \( rss1 \) and \( rss2 \) of the robust safety strategy. The relations on transitions \( tr_5, tr_6 \) and \( tr_7 \) model those conditions of the nuclear reactor which can lead to an unsafe state. These relations are the same as the conditions that are part of the rules of the safety strategy. In the model of the physical process we only consider those temperature values...
which are relevant to the safety of the reactor; in the model of the PrT net, these correspond to the predicates $P_{11}$ and $P_{12}$. For example, if the temperature of the reactor exceeds a predefined threshold ($SDTemp$), the reactor enters into the state modelled by the predicate $P_{12}$. Once this state is detected by the controller, the controller issues a request for the rods to be released. This request is also issued if the controller detects that the reactor temperature does not decrease by a significant amount after the rods are moved down, this reactor state is modelled by the predicate $P_{11}$.

**Animation**

Animation of the Safety Controller Specification enables the user of the system to check if the safety controller strategies do indeed capture the intended behaviour, and are consistent with the mission requirements. This can be achieved by executing the specification obtained at the end of the requirements analysis together with a simulation of the physical process and the plant interface. The animation provides a concrete manifestation of the abstract models, thus enabling the user to exercise the specifications in defined scenarios, and visualize different options in the refinement of the specifications. It is certainly possible to envisage a requirements validation system which would support and facilitate user interaction with the formally expressed requirements specification by presenting a user friendly graphical interpretation of a operational formalism in a form similar to that shown in Figure 4.

**Discussion**

This paper aims to present a methodology for the requirements analysis of safety—critical systems, based on the application of formal notations. The analysis is divided into domains of abstraction, separating the analysis into smaller domains and enabling the specifications to be constructed in small stages. With each domain we associate a phase of analysis identifying the activities to be performed, a method to support the analysis and the specifications to be produced /de Lemos 92/. The proposed approach attempts to follow closely the usual practices in system engineering. The utilization of formal notations are viewed as being complementary to current practices and are intended to augment traditional engineering rather than supplant established techniques.

A problem that can often arise in the application of formal notations is that engineers can lose track of the realities of the system, transforming the modelling and analysis into a mathematical exercise as opposed to a phase of system development. This issue is addressed by identifying the properties that must be analysed at each phase, and promoting linkages between the results of the different phases. For example, the framework forces the analysis of the physical process (e.g. physical laws and rules of operation) to be kept separate from the analysis of actuators and sensors, and the E/A model provides simple concepts that can be used to link the safety specifications. A further benefit of the proposed framework is that the use of formal specifications are enforced from the initial stages of the requirements analysis. This has the advantage that once the essential requirements of the system are formalised, e.g. hazards, the subsequent safety specifications can be supported by successive formal refinements. A
simple example based on nuclear reactor control was used to illustrate the support provided for formal analysis by the framework and E/A model.

Acknowledgements

The authors would like to acknowledge the financial support of BAe (DCSC) and CAPES/Brazil.

References


