TUCS

Yuliya Prokhorova | Elena Troubitsyna | Linas Laibinis

A Case Study in Refinement-Based Modelling of a Resilient Control System

Turku Centre *for* Computer Science

TUCS Technical Report
No 1086, June 2013

A Case Study in Refinement-Based Modelling of a Resilient Control System

Yuliya Prokhorova
TUCS – Turku Centre for Computer Science,
Åbo Akademi University, Department of Information Technologies
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
yuliya.prokhorova@abo.fi
Elena Troubitsyna
Åbo Akademi University, Department of Information Technologies
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
elena.troubitsyna@abo.fi
Linas Laibinis
Åbo Akademi University, Department of Information Technologies
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
linas.laibinis@abo.fi

TUCS Technical Report
No 1086, June 2013

Abstract

In this paper, we present a case study in modelling a resilient control system in Event-B. We demonstrate how to formally define the basic safety properties and fault tolerance mechanisms, as well as the system modes describing the system behaviour under different execution and fault conditions. Our formal development helps us to identify the diagnosability conditions for resilience, i.e., identify the limitations to be imposed on possible component changes to guarantee its controllability and hence dependability.

Keywords: Event-B, formal modelling, refinement, resilient control systems, steam boiler.

TUCS Laboratory
Distributed Systems Laboratory

1 Introduction

Resilience is a property of a system to remain dependable despite changes [1]. Often changes are introduced in the design to incorporate new functionality, new components as well as change the existing components. How to ensure that the introduced changes do not lead to unsafe or unreliable behaviour?

In this paper, we undertake a formal development of a control system. We apply a system-level modelling technique – Event-B [2] – to formally derive a system specification and represent its behaviour in different operational modes. Our formalisation allows us to identify diagnosability conditions – the restrictions imposed on the system design to ensure that the controller can deduce the state of the controlled environment. The diagnosability conditions introduce the constraints on possible changes in the system requirements that would allow us to preserve system dependability. Our approach is presented via a case study – development of a steam boiler control system [3].

The paper is organised as follows. In Section 2, we briefly describe the Event-B framework and the refinement-based approach to modelling systems in Event-B. Section 3 presents our case study – a steam boiler control system – and outlines the proposed modelling strategy. Section 4 presents a formal development of the steam boiler and illustrates the specification and verification process of the system requirements. In Section 5, we assess our contributions by describing lessons learned. Finally, in Section 6, we review related work and conclude the paper.

2 Refinement-Based Modelling in Event-B

Resilience is a system-level property that requires the techniques supporting system-level modelling and analysis. In this paper, we use Event-B [2, 4] as a framework for system level modelling and proof-based verification.

In Event-B, system models are defined using the notion of an *abstract state machine*. The abstract machine encapsulates the state (the variables) of a model and defines operations (events) on its state. Each machine is uniquely identified by its name *MachineName*. The state variables of the machine are declared in the *Variables* clause and initialised in the *INITIALISATION* event. The variables are strongly typed by the constraining predicates given in the *Invariants* clause. The data types and constants of the model are given in *context* that also postulated their properties as axioms. The behaviour of the system is defined by a number of atomic *events*. An event has the following form:

$$evt \mathrel{\hat{=}} \textbf{any}\ lv\ \textbf{where}\ g\ \textbf{then}\ R\ \textbf{end},$$

where lv is a list of local variables, the guard g is the conjunction of predicates defined over the model (local and state) variables, and the action R is a parallel composition of assignments over the variables.

A guard defines when an event is enabled. If several events are enabled simultaneously, then any of them can be chosen for execution non-deterministically. If none of the events is enabled, then the system deadlocks. In general, the action of an event is a composition of assignments executed simultaneously. Variable assignments can be either deterministic or non-deterministic. The deterministic assignment is denoted as $x := E(v)$, where x is a state variable and $E(v)$ is an expression over the state variables v. The non-deterministic assignment can be denoted as $x :\in S$ or $x :\mid Q(v, x')$, where S is a set of values and $Q(v, x')$ is a predicate. As a result of the non-deterministic assignment, x gets any value from S or it obtains a value x' such that $Q(v, x')$ is satisfied.

Event-B promotes top-down approach to correct-by-construction system development. It relies on the top-down refinement-based approach to formal development. The development starts from an abstract specification of the system that defines essential behaviour and properties of the system. In a number of correctness-preserving transformations – refinements – we introduce implementation details and arrive at the detailed system specification closely resembling an eventual implementation. Usually refinement steps result in introducing new variables and events into the model.

We can also perform data refinement allowing us to replace some abstract variables of the model with their concrete counterparts. In this case, the invariant of a refined model formally defines the relationship between the abstract and concrete variables.

Event-B relies on theorem proving to verify correctness. Via discharging proof obligations we formally verify the essential correctness conditions: the events preserve the invariant; whenever the event is enabled, there exists some reachable after-state (i.e., each event is feasible); the model is well-formed; refinement does not introduce additional deadlocks. The detailed discussion of the Event-B proof obligations can be found in [2].

The Rodin platform [4] provides an integrated modelling environment that includes automated theorem proving environment. In general, the Rodin platform achieves a high degree of automation – usually over 80% of proof obligations are discharged automatically.

In the next section, we present our case study – a formal development of the steam boiler control system.

3 The Steam Boiler Control System

The steam boiler control system (Fig. 1) is a resilient control system that produces steam and adjusts the quantity of water in the steam boiler to maintain it within the predefined safety boundaries [3].

The system consists of the following units: a chamber, a pump, a valve, a sensor to measure the quantity of water in the chamber, a sensor to measure the quantity of steam which comes out of the steam boiler chamber, a sensor to measure water input through the pump, and a sensor to measure water output through the valve. The system parameters are given in Table 1.

The considered system has several execution modes. After being powered on, the system enters the **Initialisation** mode. At each control cycle, the system reads sensors and performs failure detection. Then, depending on the detection result, the system may enter either an operational mode or a non-operational mode. There are three operational modes in the system:

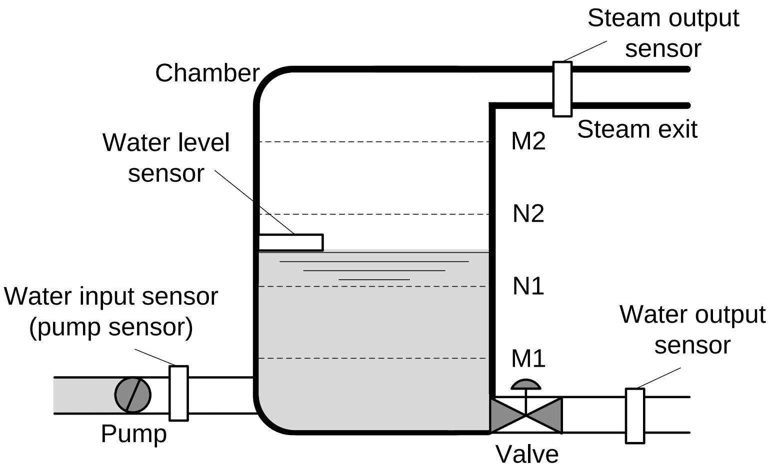

Figure 1: Steam boiler

Table 1: Parameters of the steam boiler

Label	Description	Unit
C	the total capacity of the steam boiler chamber	litre
M1	the minimal quantity of water, i.e., the lower safety boundary	litre
M2	the maximal quantity of water, i.e., the upper safety boundary	litre
N1	the minimal normal quantity of water to be maintained during regular operation	litre
N2	the maximal normal quantity of water to be maintained during regular operation	litre
W	the maximal quantity of steam produced	litre/sec
U1	the maximal gradient of increase of the quantity of steam	litre/sec/sec
U2	the maximal gradient of decrease of the quantity of steam	litre/sec/sec
P	the maximal capacity of the pump	litre/sec

Normal, **Degraded**, and **Rescue**. In the **Normal mode**, the system attempts to maintain the water level in the chamber between the normal boundaries N1 and N2 (here N1 $<$ N2), providing that no failures of the system units have occurred. In the **Degraded mode**, the system tries to maintain the water level within normal boundaries despite failures of some physical non-critical units. In the **Rescue mode**, the system attempts to maintain the normal water level in the presence of a failure of the critical system unit – the water level sensor. If failures of the system units and water level sensor occur simultaneously or the water level is outside of the safety boundaries M1 and M2 (here M1 $<$ M2), the system enters the non-operational mode **Emergency_Stop**.

The fault tree [5] shown in Fig. 2 gives a systematic representation of safety requirements. The main hazard of the system is associated with the overflow or lack of water in the chamber, i.e., when the water level exceeds the safety boundaries. Next, we list both the functional (Table 2) and safety requirements (Table 3) and show how they are incorporated in an Event-B formal specification.

In our development, we consider the following failures of the system and its units.

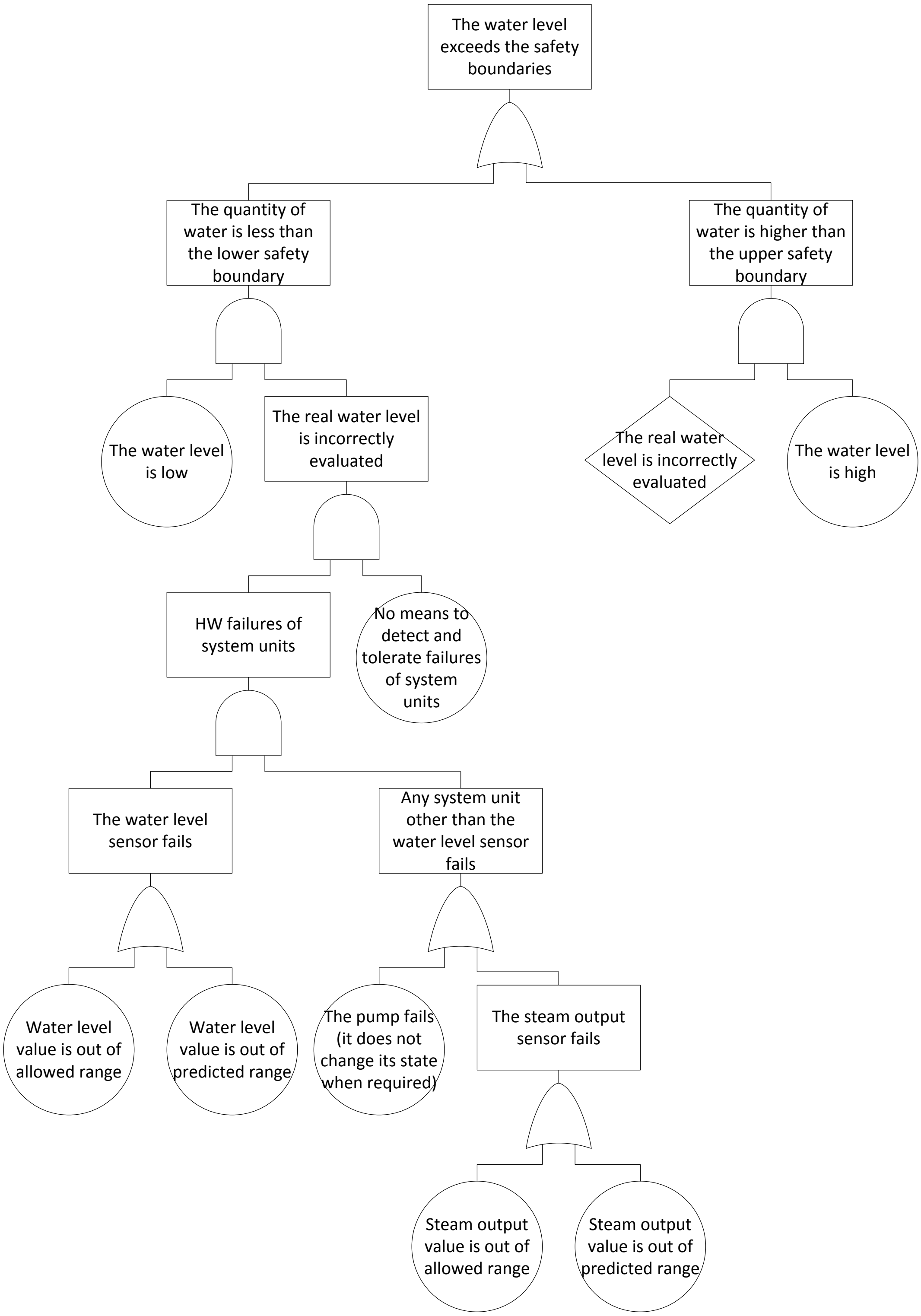

Figure 2: Fault tree of the steam boiler

Table 2: Functional requirements of the steam boiler control system

ID	Requirement
FR-01	The system shall rely on the minimal and maximal predicted values of the water level to detect whether the water level is within the normal and safety boundaries
FR-02	The pump and the valve shall not be operated simultaneously
FR-03	The valve shall be switched on in the pre-operational phase only
FR-04	When the water level is between N1 and M1, the pump shall be switched on
FR-05	When the water level is between N2 and M2, the pump shall be switched off

Table 3: Safety requirements of the steam boiler control system

ID	Requirement
SR-01	When the system failure is detected, the steam boiler control system shall be shut down and an alarm shall be raised (the system shall enter the emergency stop mode)
SR-02	During the system operation the water level shall not exceed the predefined safety boundaries
SR-03	If either the water level exceeds the safety boundaries or there is a failure of the water level sensor and there is a failure of the pump or the steam output sensor, the system failure shall be detected
SR-04	When the water level value is out of the allowed range or the water level value is out of the predicted range, the water level sensor failure shall be detected
SR-05	When the pump does not change its state if required, the pump actuator failure shall be detected
SR-06	When the steam output value is out of the allowed range or the steam output value is out of the predicted range, the steam output sensor failure shall be detected
SR-07	When the water level sensor fails, the minimal and maximal predicted values of the water level shall be computed independently of the sensor readings
SR-08	When the steam output sensor fails, the minimal and maximal predicted values of the steam output shall be computed independently of the sensor readings
SR-09	When the pump actuator fails, the system shall rely on the pump sensor readings and shall not switch the pump actuator
SR-10	When the pump or the steam output sensor failure is detected, the steam boiler control system shall enter the degraded mode
SR-11	When the water level sensor failure is detected, the steam boiler control system shall enter the rescue mode

A failure of the steam boiler control system is detected if either the water level in the chamber is out of the predefined safety boundaries (i.e., if it is lower than M1 or higher than M2) or a combined failure of the water level sensor and any other system unit (the pump or the steam output sensor) occurs. The water level sensor fails if it indicates a value which exceeds the allowed range (i.e., the range in which a non-failed sensor operates) or a value which exceeds the predicted range. The pump fails if it does not change its state when required. The steam output sensor fails if it indicates a value which is out of the allowed range or the value which is out of the predicted range.

The water level sensor failure by itself does not lead to the system failure. The steam boiler contains the information redundancy, i.e., the controller is able to estimate the water level in the chamber based on the amount of water produced by the pump and the amount of the released steam. Similarly, the controller is able to maintain the acceptable level of efficiency based on the water level sensor readings if either the pump or the steam output sensor fail. Furthermore, the system has an intrinsic resilience mechanism: it can cope with both the physical pump failure and failed water supply due to clogged pipes.

We design a formal specification of the steam boiler control system incrementally, i.e., by gradually unfolding the system functionality and architecture. This allows us to structure complex functional (**FR-01..FR-05**) and safety (**SR-01..SR-11**) requirements and also simplifies verification.

Let us now shortly outline the proposed development strategy (Fig. 3). The abstract model (the machine **M0**) implements a basic control loop. The first refinement (**M1**) introduces an abstract representation of the activities performed after the system is powered on and during system operation (in nominal and failure conditions). At the second refinement step (**M2**), we introduce a detailed representation of the conditions leading to a system failure. We model the physical environment of the system and refine its failure detection procedures at the third refinement step (**M3**). Finally, at the fourth refinement step (**M4**), we introduce a representation of the required execution modes. Each machine **M0..M4** has an associated context **C0..C3** (i.e., a machine *sees* a context) where the corresponding properties of the model are postulated as axioms. While each subsequent machine *refines* the previous one, each subsequent context *extends* the previous one. Additionally, several machines can see the same context (e.g., both machines **M1** and **M2** see the context **C1**).

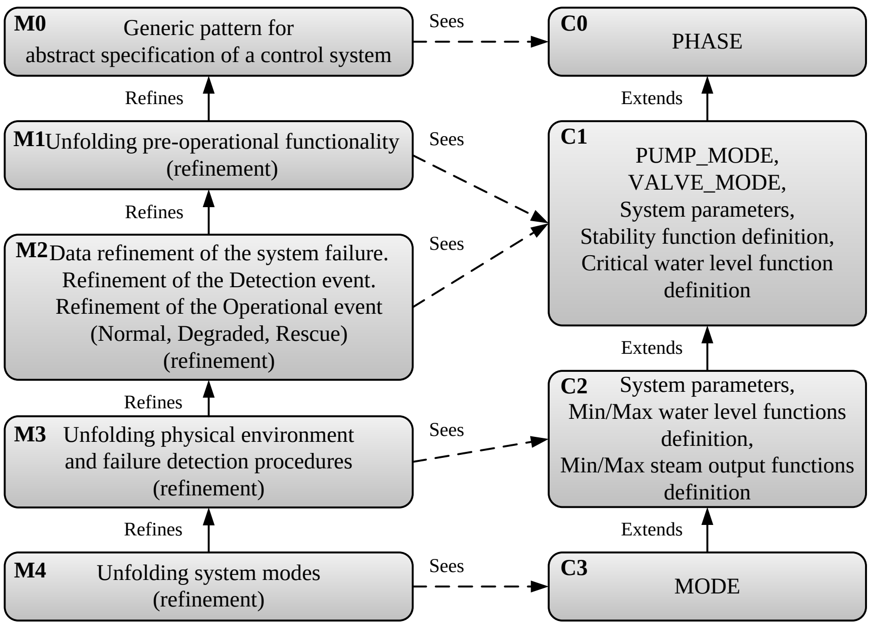

Figure 3: Refinement strategy

4 Development of Steam Boiler by Refinement in Event-B

In this section, we give an overview of the formal development of the steam boiler. The complete models of our formal development can be found in Appendix.

4.1 The Abstract Model

In the abstract model, the system behaviour is defined as interleaving between the events modelling the environment and the controller as proposed in our previous work [6]. The behaviour of the controller has the following stages: Detection, Control (Normal Operation or Error Handling), Prediction as modelled by the variable *phase*: {*ENV, DET, CONT, PRED*}. The events *Environment*, *Operational* and *Prediction* abstractly model the environment, controller reaction and computation of the next expected states of the system units correspondingly. The event *Detection* non-deterministically models an outcome of the error detection. A reaction on errors is abstractly modelled by the event *EmergencyStop*.

In the abstract specification, we start to abstractly model the safety requirement **SR-01**. The variable *stop* abstractly models system shutdown and raising the alarm. The safety invariant:

inv0.1: $failure = TRUE \wedge phase \neq CONT \Rightarrow stop = TRUE$

ensures that, if a failure has been detected and processed (*phase* $\neq$ *CONT*), the shutdown will be initiated. Let us note that the invariant **inv0.1** covers only a part of the requirement **SR-01**. Modelling **SR-01** will be completed when we introduce a representation of execution modes in our model.

4.2 The First Refinement: Unfolding Pre-operational Functionality

At our first refinement step, we introduce a representation of system components. We define the variables representing the water level sensor and the actuators – the pump and the valve. The variable *water_level* models the sensor readings, while *min_water_level* and *max_water_level* represent the estimated interval for a *sensed* water level (which corresponds to the functional requirement **FR-01**). The values of the variables *min_water_level* and *max_water_level* are assigned in the event *Prediction*.

The variables *pump_ctrl* and *valve_ctrl* model the steam boiler actuators – the pump and the valve respectively. If the pump is switched on, the value of the variable *pump_ctrl* equals to *ON*. It is *OFF* otherwise. The valve can be open (*valve_ctrl = OPEN*) or closed (*valve_ctrl = CLOSED*). At this refinement step, we introduce the following invariants:

inv1.1: $valve\_ctrl = OPEN \Rightarrow pump\_ctrl = OFF,$

inv1.2: $failure = FALSE \wedge phase \neq ENV \wedge phase \neq DET \Rightarrow min\_water\_level \geq M1 \wedge max\_water\_level \leq M2.$

The invariant **inv1.1** corresponds to the system functional requirement **FR-02**, while the invariant **inv1.2** ensures another main system safety requirement (**SR-02**). The event guards ensure that the minimal and maximal water levels are within the nominal interval [M1..M2].

Moreover, at this stage we refine the event *Operational* to single out the system initialisation stage. The *PreOperational* events (Fig. 4) are executed only at the beginning of the system operation to equalize the amount of water in the boiler chamber. Once the water level reaches the normal boundaries, the *PreOperational* events are disabled. Then the *Operational* event can be executed.

```
event PreOperational1  refines Operational
  where
    phase = CONT
    failure = FALSE
    stop = FALSE
    preop_flag = TRUE
    max_water_level > N2 ∧ min_water_level > N1
    min_water_level ≥ M1 ∧ max_water_level ≤ M2
  then
    pump_ctrl ≔ OFF
    valve_ctrl ≔ OPEN
    phase ≔ PRED
end
```

```
event PreOperational2  refines Operational
  where
    phase = CONT
    failure = FALSE
    stop = FALSE
    preop_flag = TRUE
    min_water_level ≥ M1 ∧ max_water_level ≤ M2
    max_water_level ≤ N2
  then
    pump_ctrl :| pump_ctrl' ∈ PUMP_MODE ∧
        (((min_water_level ≥ N1 ⇒ pump_ctrl' = OFF) ∧
        (min_water_level < N1 ⇒ pump_ctrl' = ON)) ∨
        pump_ctrl' = pump_ctrl)
    valve_ctrl ≔ CLOSED
    preop_flag ≔ FALSE
    phase ≔ PRED
end
```

Figure 4: Pre-operational events

Since the valve is only used to adjust the water level in the chamber at the pre-operational phase, the requirement (**FR-03**) is formalised as follows:

$$\textbf{inv1.3:}\ preop\_flag = FALSE \ \Rightarrow\ valve\_ctrl = CLOSED,$$

where *preop_flag* indicates whether the system is in the pre-operational stage. Furthermore, we refine the event *Detection* to separate three cases: (1) the water level is within [M1..M2] and no failure is detected; (2) the water level is out of [M1..M2]; (3) the water level is within [M1..M2], but some failures are detected.

4.3 The Second Refinement: Introducing Failure Assumptions

To introduce the required operational modes presented in Section 3, we split the event *Operational* into three events: *Normal_Operational*, *Degraded_Operational* and *Rescue_Operational* (see Fig. 5).

At this refinement step, we also elaborate on failure detection procedures. We establish a relation between a system failure and component failures via the following invariant:

inv2.1: $(phase \neq DET \wedge phase \neq ENV) \Rightarrow$
$(failure = TRUE \Leftrightarrow ((wl\_sensor\_failure = TRUE \wedge$
$(pump\_failure = TRUE \vee so\_sensor\_failure = TRUE)) \vee$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE)),$

where *wl_sensor_failure* stands for a failure of the water level sensor, the variable *pump_failure* models a failure of the pump, while the variable *so_sensor_failure* represents a failure of the steam output sensor. *WL_critical* is a function that returns *TRUE*, if the water level exceeds the safety limits, and *FALSE* otherwise. Let us point out that the pump is a complex device which includes both a sensor and an actuator. However, we consider only an actuator failure and assume that the pump sensor (the water input sensor) never fails. We also assume that the valve and water output sensor never fail.

Therefore, the given invariant establishes a correspondence between an abstract system failure (represented by the abstract variable *failure*) and the specific failures of the components. It postulates that a failure of the system occurs if and only if either a water level sensor failure is detected in combination with a unit failure (a pump failure or a steam output sensor failure or both) or the critical water level is exceeded. The corresponding events are modified accordingly (Fig. 5).

```
event Normal_Operational  refines Operational
 where
  ...
  wl_sensor_failure = FALSE ∧ pump_failure = FALSE ∧ so_sensor_failure = FALSE
  WL_critical(min_water_level ↦ max_water_level) = FALSE
 then
  // operate normally relying on min/max_water_level
end

event Degraded_Operational  refines Operational
 where
  ...
  wl_sensor_failure = FALSE ∧ (pump_failure = TRUE ∨ so_sensor_failure = TRUE)
  WL_critical(min_water_level ↦ max_water_level) = FALSE
 then
  // if the pump failure is detected, do not modify pump_ctrl,
                              else operate normally relying on min/max_water_level
end

event Rescue_Operational  refines Operational
 where
  ...
  wl_sensor_failure = TRUE ∧ pump_failure = FALSE ∧ so_sensor_failure = FALSE
  WL_critical(min_water_level ↦ max_water_level) = FALSE
 then
  // operate normally relying on min/max_water_level
end
```

Figure 5: Normal, degraded and rescue operational events

We also introduce the notion of failure stability into the model (axioms **axm2.1** and **axm2.2** in the context **C1** given in Fig. 3). The failure stability means that, once a failure occurred, the value of the variable representing this failure remains unchanged until the whole system is restarted:

axm2.1: $Stable \in BOOL \times BOOL \rightarrow BOOL,$

axm2.2: $\forall x, y \cdot x \in BOOL \wedge y \in BOOL \Rightarrow$
$(Stable(x \mapsto y) = TRUE \Leftrightarrow (x = TRUE \Rightarrow y = TRUE)).$

We rely on the stability property to refine the detection events. At this refinement step, we also model the process of switching on and off the pump, i.e., **FR-04** and **FR-05**. An adherence to the corresponding requirements is ensured by the following invariants:

inv2.2: $(pump\_failure = FALSE \;\wedge\; phase = PRED \;\wedge$
$max\_water\_level < N1 \;\wedge\; min\_water\_level \geq M1) \;\Rightarrow$
$pump\_ctrl = ON,$

inv2.3: $(pump\_failure = FALSE \;\wedge\; phase = PRED \;\wedge$
$min\_water\_level > N2 \;\wedge\; max\_water\_level \leq M2) \;\Rightarrow$
$pump\_ctrl = OFF.$

Let us note that the invariants **inv2.2** and **inv2.3** guarantee that the pump is not switched on if a failure is detected (**SR-09**).

4.4 The Third Refinement: Unfolding Physical Environment

The third refinement step elaborates further on the physical behaviour of the steam boiler and failure detection procedures. The new variables *steam_output* and *water_output* stand for readings of the steam output sensor and the water output sensor respectively.

To implement safety requirements associated with failure detection of the system components (i.e., **SR-04..SR-06**), we refine the previously introduced abstract detection events as follows. Firstly, the event *Detection_OK* is decomposed into a set of events modelling detection of different types of failures. Similarly, the event *Detection_NOK* is refined into a set of events modelling combinations of failures of the water level sensor and pump or steam output sensors.

The names of the events reflect the results of failure detection. If a failure or combination of failures does not lead to a system failure, it is called *Detection_OK_**. Otherwise, an event is called *Detection_NOK_**.

Two variables representing the steam output predicted values – *min_steam_output* and *max_steam_output* – are introduced in the same way as the water prediction values. For the sake of simplicity, we assume that the water output sensor never fails.

We refine the event *Prediction* to calculate expected (predicted) values of the minimal and maximal water level and steam output (Fig. 6). In the context of the model (**C2** in Fig. 3), we define functions *WL_min*, *WL_max*, *SO_min* and *SO_max* to compute them.

```
event Prediction  refines Prediction
 where
  phase = PRED
  stop = FALSE
 then
  phase ≔ ENV
  min_water_level, max_water_level :| min_water_level' ∈ 0..C ∧ max_water_level' ∈ 0..C ∧
      ((wl_sensor_failure = FALSE ∧ so_sensor_failure = FALSE) ⇒
      (min_water_level' = WL_min(water_level↦steam_output↦pump↦water_output) ∧
       max_water_level' = WL_max(water_level↦steam_output↦pump↦water_output))) ∧
      ((wl_sensor_failure = TRUE ∧ so_sensor_failure = FALSE) ⇒
      (min_water_level' = WL_min(min_water_level↦steam_output↦pump↦water_output) ∧
       max_water_level' = WL_max(max_water_level↦steam_output↦pump↦water_output))) ∧
       min_water_level' ≤ max_water_level'∧
      ((wl_sensor_failure = FALSE ∧ so_sensor_failure = TRUE) ⇒
      (min_water_level' = WL_min(water_level↦min_steam_output↦pump↦water_output) ∧
       max_water_level' = WL_max(water_level↦max_steam_output↦pump↦water_output)))
  min_steam_output, max_steam_output :| min_steam_output' ∈ 0..W ∧ max_steam_output' ∈ 0..W ∧
      (so_sensor_failure = FALSE ⇒ (min_steam_output' = SO_min(steam_output) ∧
       max_steam_output' = SO_max(steam_output))) ∧
      (so_sensor_failure = TRUE ⇒ (min_steam_output' = SO_min(min_steam_output) ∧
       max_steam_output' = SO_max(max_steam_output))) ∧ min_steam_output' ≤ max_steam_output'
end
```

Figure 6: The event Prediction

WL_min and *WL_max* take the current values of the variables *water_level*, *steam_output*, *pump*, *water_output* as the input and return new predicted values, which are assigned to the respective variables *min_water_level* and *max_water_level*. Similarly, the functions *SO_min* and *SO_max* take the current value of the variable *steam_output* and return new values to be assigned to the respective variables *min_steam_output* and *max_steam_output*.

These calculations are performed with the actual water level and steam output values only in the nominal conditions. In the presence of failures, the minimal and maximal values are used instead. The system behaviour in the presence of a pump failure (**SR-09**) is modelled by the following assignment:

$$\begin{aligned} pump\_ctrl\ :|\ & pump\_ctrl' \in PUMP\_MODE \wedge \\ & (pump\_failure = TRUE \ \Rightarrow\ pump\_ctrl' = pump\_ctrl). \end{aligned}$$

4.5 The Fourth Refinement: Introducing System Modes

To explicitly define the system execution modes, we introduce the variable *mode*, which can take the following values: {*Initialisation, Normal, Degraded, Rescue, Emergency_Stop*}.

The assignments to the variable *mode* reflect the mode changes defined in the requirements (**SR-10**) and **(SR-11)**. The mode changes are modelled in the corresponding detection events. The following invariants ensure a proper mapping between a mode and its entry conditions:

inv4.1: $mode = Normal \;\Rightarrow\; wl\_sensor\_failure = FALSE \;\wedge$
$pump\_failure = FALSE \;\wedge\; so\_sensor\_failure = FALSE,$

inv4.2: $mode = Degraded \;\Rightarrow\; wl\_sensor\_failure = FALSE \;\wedge$
$(pump\_failure = TRUE \;\vee\; so\_sensor\_failure = TRUE),$

inv4.3: $mode = Rescue \;\Rightarrow\; wl\_sensor\_failure = TRUE \;\wedge$
$pump\_failure = FALSE \;\wedge\; so\_sensor\_failure = FALSE,$

inv4.4: $mode = Emergency\_Stop \;\Rightarrow\; ((wl\_sensor\_failure = TRUE \;\wedge$
$(pump\_failure = TRUE \;\vee\; so\_sensor\_failure = TRUE)) \;\vee$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE).$

After introducing a representation of modes in the model, we complete modelling of the safety requirement **SR-01**:

inv4.5: $phase \neq ENV \;\wedge\; phase \neq DET \;\wedge$
$((wl\_sensor\_failure = TRUE \;\wedge\; (pump\_failure = TRUE \;\vee$
$so\_sensor\_failure = TRUE)) \;\vee$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE) \;\Rightarrow$
$mode = Emergency\_Stop.$

To guarantee that the system will not enter the non-operational mode if the system failure is not detected, we define the invariant:

inv4.6: $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \;\wedge$
$stop = FALSE \;\wedge\; (wl\_sensor\_failure = FALSE \;\vee$
$(pump\_failure = FALSE \;\wedge\; so\_sensor\_failure = FALSE)) \;\Rightarrow$
$mode \;\neq\; Emergency\_Stop.$

Moreover, to guarantee that the predefined reaction on errors (i.e., shutdown of the system and raising the alarm) occurs after execution of the event $EmergencyStop$, we postulate the following theorem:

thm4.1: $\forall p' \cdot p' \in \{stop' \mapsto pump\_ctrl' \mapsto valve\_ctrl' \mid$
$(\exists phase, stop, pump\_ctrl, valve\_ctrl, mode \cdot$
$(phase = CONT \;\wedge\; stop = FALSE \;\wedge\; mode = Emergency\_Stop) \;\wedge$
$(stop' = TRUE \;\wedge\; pump\_ctrl' = OFF \;\wedge\; valve\_ctrl' = CLOSED))\} \;\Rightarrow$
$p' \in \{stop' \mapsto pump\_ctrl' \mapsto valve\_ctrl' \mid stop' = TRUE\},$

where the variable p' is of the type $BOOL \times \{ON, OFF\} \times \{OPEN, CLOSED\}$.

5 Lessons Learned

5.1 Discussion of the Development

Table 4 gives the proof statistics of the formal development of the steam boiler control system. It shows that over 90% of proof obligations were automatically proved by the Rodin platform. Moreover, one can observe the significant increase in the number of proof obligations at the third refinement step. This is caused by the complexity of the model of the physical environment and by a high number of the introduced error detection events to cover all the identified hazardous situations associated with the environment. In general, the number of proof obligations to be discharged at each refinement step does not depend on the number of the proof obligations at the previous refinement step. For instance, since introducing the system modes is a more simple procedure than unfolding the physical environment and error detection, the number of proof obligations in the fourth refinement is lower.

Table 4: Proof statistics

Model	Proof Obligations	Automatically Discharged	Interactively Discharged
Context	15	13	2
Abstract Model	10	10	0
1^{st} *Refinement*	35	33	2
2^{nd} *Refinement*	157	145	12
3^{rd} *Refinement*	231	205	26
4^{th} *Refinement*	193	183	10
Total	641	589	52

The presented formal development in Event-B has facilitated derivation and verification of a complex specification in a highly automated manner. However, the Rodin platform has not coped sufficiently well with the event feasibility proofs and required interactive proving. Moreover, weak support provided by the platform for arithmetic calculations made it hard to instantiate the required abstract functions with the actual physical laws.

5.2 Diagnosability

Our formal modelling has allowed us to formally underpin the diagnosability conditions. The formulated invariants explicitly define the conditions that should be satisfied for an action to take place. These conditions can be seen as restrictions that should be put on the system architecture when changes are introduced. For instance, the changes should ensure that each parameter remains controllable either by the corresponding sensor or via information redundancy. Moreover, an introduction of new operational modes should ensure mode exclusiveness conditions (no two modes are enabled simultaneously). Finally, the mechanisms of monitoring the environment should not be weakened as a result of changes.

6 Related Work and Conclusions

Nowadays, resilient control systems received a notable attention. In spite of the fact that these systems are employed in critical domains, there is a lack in formal techniques to modelling and verification of their crucial safety properties. Variations of resilient control systems are usually verified by simulation [7] and model-checking [8]. Thus, the paper [7] verifies the proposed resilient control strategy by utilising a co-simulation platform based on Matlab/Simulink and EnergyPlus while the authors of [8] perform model-checking of adaptive resilient systems using the AdaCTL logic.

Formal modelling of the steam boiler control system has been undertaken in several works [9] by applying various formalisms (e.g., Z, VDM, Action Systems, etc.) and focusing on various properties of the system (e.g., safety properties, real-time behaviour, etc.). Our formalisation is based on state-based modelling. Moreover, it allows us to obtain a more detailed specification. Furthermore, the used formal language (Event-B) has more powerful tool support, which makes it attractive to the industrial practitioners.

In this paper, we have presented a formal refinement-based development of a resilient control system – the steam boiler control system. We formally specified and verified the essential functional and safety requirements of this system. Our formal modelling helped us to define diagnosability conditions that facilitate incorporation of the design changes typical for resilient systems in a dependability-preserving way.

In our future work, we are planning to further elaborate on a taxonomy of diagnosability requirements.

References

[1] J. Laprie, “From Dependability to Resilience,” in *Proceedings of the 38th IEEE/IFIP International Conference on Dependable Systems and Networks*, 2008, pp. G8—-G9.

[2] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*, 1st ed. New York, NY, USA: Cambridge University Press, 2010.

[3] J.-R. Abrial, “Steam-Boiler Control Specification Problem,” in *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control (the book grow out of a Dagstuhl Seminar, June 1995)*. London, UK: Springer-Verlag, 1996, pp. 500–509.

[4] “Event-B and the Rodin Platform,” http://www.event-b.org/, 2013.

[5] N. Storey, *Safety-Critical Computer Systems*. Harlow, UK: Addison-Wesley, 1996.

[6] I. Lopatkin, Y. Prokhorova, E. Troubitsyna, A. Iliasov, and A. Romanovsky, “Patterns for Representing FMEA in Formal Specification of Control Systems,” Tech. Rep. TUCS 1003, 2011.

[7] K. Ji, Y. Lu, L. Liao, Z. Song, and D. Wei, “Prognostics Enabled Resilient Control for Model-Based Building Automation Systems,” in *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, 2011, pp. 286–293.

[8] M. Cordy, A. Classen, P. Heymans, A. Legay, and P.-Y. Schobbens, “Model Checking Adaptive Software with Featured Transition Systems,” in *Assurances for Self-Adaptive Systems*, ser. LNCS, J. Cámara, R. Lemos, C. Ghezzi, and A. Lopes, Eds. Springer Berlin Heidelberg, 2013, vol. 7740, pp. 1–29.

[9] J.-R. Abrial, E. Börger, and H. Langmaack, Eds., *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control (the book grow out of a Dagstuhl Seminar, June 1995)*, ser. LNCS, vol. 1165. London, UK, UK: Springer-Verlag, 1996.

Appendix

Event-B Development of the Steam Boiler Control System

CONTEXT C0
SETS

$PHASE$

CONSTANTS

ENV
DET
$CONT$
$PRED$

AXIOMS

`axm1` : $partition(PHASE, \{ENV\}, \{DET\}, \{CONT\}, \{PRED\})$

END

MACHINE M0 // The abstract model
SEES C0
VARIABLES

$phase$
$failure$
$stop$

INVARIANTS

`inv1` : $phase \in PHASE$
`inv2` : $failure \in BOOL$
`inv3` : $stop \in BOOL$
`inv4` : $failure = FALSE \Rightarrow stop = FALSE$
`inv5` : $failure = TRUE \wedge phase \neq CONT \Rightarrow stop = TRUE$ // inv0.1

EVENTS

Initialisation

begin

act1 : $phase := ENV$
act2 : $failure := FALSE$
act3 : $stop := FALSE$

end

Event *Environment* $\widehat{=}$

when

grd1 : $phase = ENV$
grd2 : $stop = FALSE$

then

act1 : $phase := DET$

end

Event *Detection* $\widehat{=}$

when

grd1 : $phase = DET$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$

then

act1 : $phase := CONT$
act2 : $failure :\in BOOL$

end

Event *Operational* $\widehat{=}$

when

grd1 : $phase = CONT$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$

then

act1 : $phase := PRED$

end

Event *EmergencyStop* $\widehat{=}$

when

grd1 : $phase = CONT$
grd2 : $failure = TRUE$
grd3 : $stop = FALSE$

then

act1 : $stop := TRUE$

end

Event *Prediction* $\widehat{=}$

when

grd1 : $phase = PRED$
grd2 : $stop = FALSE$

then

act1 : $phase := ENV$

end

END

CONTEXT C1

EXTENDS C0

SETS

$PUMP\_MODE$
$VALVE\_MODE$

CONSTANTS

ON
OFF
$OPEN$
$CLOSED$
$N1$
$N2$
$M1$
$M2$
C
$WL\_critical$
$Stable$

AXIOMS

axm1 : $partition(PUMP\_MODE, \{ON\}, \{OFF\})$
axm2 : $partition(VALVE\_MODE, \{OPEN\}, \{CLOSED\})$
axm3 : $N1 \in \mathbb{N}_1$
axm4 : $N2 \in \mathbb{N}_1$
axm5 : $M1 \in \mathbb{N}_1$

axm6 : $M2 \in \mathbb{N}_1$

axm7 : $C \in \mathbb{N}_1$

axm8 : $0 < M1 \wedge M1 < N1 \wedge N1 < N2 \wedge N2 < M2 \wedge M2 < C$

axm9 : $0 \notin \mathbb{N}_1$

axm10 : $WL\_critical \in \mathbb{N} \times \mathbb{N} \rightarrow BOOL$

axm11 : $\forall x, y \cdot x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge ((x < M1 \vee y > M2) \Leftrightarrow WL\_critical(x \mapsto y) = TRUE)$

axm12 : $\forall x, y \cdot x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge ((x \geq M1 \wedge y \leq M2) \Leftrightarrow WL\_critical(x \mapsto y) = FALSE)$

axm13 : $Stable \in BOOL \times BOOL \rightarrow BOOL$ //axm2.1

axm14 : $\forall x, y \cdot x \in BOOL \wedge y \in BOOL \Rightarrow (Stable(x \mapsto y) = TRUE \Leftrightarrow (x = TRUE \Rightarrow y = TRUE))$ //axm2.2

axm15 : $\exists n1, n2, m1, m2, c \cdot n1 \in \mathbb{N}_1 \wedge n2 \in \mathbb{N}_1 \wedge m1 \in \mathbb{N}_1 \wedge m2 \in \mathbb{N}_1 \wedge c \in \mathbb{N}_1 \wedge 0 < m1 \wedge m1 < n1 \wedge n1 < n2 \wedge n2 < m2 \wedge m2 < c$

axm16 : $\forall x, y, z \cdot x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge z \in \mathbb{N} \wedge x \leq y \wedge y < z \Rightarrow x < z$

END

MACHINE M1 // The first refinement: unfolding pre-operational functionality

REFINES M0

SEES C1

VARIABLES

phase

failure

stop

water_level

pump_ctrl

valve_ctrl

preop_flag

min_water_level

max_water_level

INVARIANTS

inv1 : $water\_level \in \mathbb{Z}$

inv2 : $pump\_ctrl \in PUMP\_MODE$

inv3 : $valve\_ctrl \in VALVE\_MODE$

inv4 : $preop\_flag \in BOOL$

inv5 : $valve\_ctrl = OPEN \Rightarrow pump\_ctrl = OFF$ // inv1.1

inv6 : $pump\_ctrl = ON \Rightarrow valve\_ctrl = CLOSED$

inv7 : $min\_water\_level \in \mathbb{N}$

inv8 : $max\_water\_level \in \mathbb{N}$

inv9 : $min\_water\_level \leq max\_water\_level$

inv10 : $failure = FALSE \land phase \neq ENV \land phase \neq DET \Rightarrow min\_water\_level \geq M1 \land max\_water\_level \leq M2$ // inv1.2

inv11 : $preop\_flag = TRUE \Rightarrow pump\_ctrl = OFF$

inv12 : $preop\_flag = FALSE \Rightarrow valve\_ctrl = CLOSED$ // inv1.3

EVENTS

Initialisation

begin

act1 : $phase := ENV$

act2 : $failure := FALSE$

act3 : $stop := FALSE$

act4 : $water\_level, min\_water\_level, max\_water\_level :|$
$water\_level' \in M1 .. M2 \land min\_water\_level' \in M1 .. M2 \land$
$max\_water\_level' \in M1 .. M2 \land$
$min\_water\_level' \leq max\_water\_level' \land$
$min\_water\_level' = water\_level' \land max\_water\_level' = water\_level'$

act5 : $pump\_ctrl := OFF$

act6 : $valve\_ctrl := CLOSED$

act7 : $preop\_flag := TRUE$

end

Event *Environment* $\widehat{=}$
refines *Environment*

when

grd1 : $phase = ENV$

grd2 : $stop = FALSE$

then

act1 : $phase := DET$

act2 : $water\_level :\in \mathbb{Z}$

end

Event *Detection_OK* $\widehat{=}$
refines *Detection*

when

grd1 : $phase = DET$

grd2 : $failure = FALSE$

grd3 : $stop = FALSE$

grd4 : $min\_water\_level \geq M1 \wedge max\_water\_level \leq M2$

then

act1 : $phase := CONT$

end

Event *Detection_NOK_1* $\widehat{=}$
refines *Detection*

when

grd1 : $phase = DET$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$
grd4 : $min\_water\_level < M1 \vee max\_water\_level > M2$

then

act1 : $phase := CONT$
act2 : $failure := TRUE$

end

Event *Detection_NOK_2* $\widehat{=}$
refines *Detection*

when

grd1 : $phase = DET$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$
grd4 : $min\_water\_level \geq M1 \wedge max\_water\_level \leq M2$

then

act1 : $phase := CONT$
act2 : $failure :\in BOOL$

end

Event *PreOperational1* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$
grd4 : $preop\_flag = TRUE$
grd5 : $max\_water\_level > N2 \wedge min\_water\_level > N1$
grd6 : $min\_water\_level \geq M1 \wedge max\_water\_level \leq M2$

then

act1 : $pump\_ctrl := OFF$
act2 : $valve\_ctrl := OPEN$
act3 : $phase := PRED$

end

Event *PreOperational2* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$
grd4 : $preop\_flag = TRUE$
grd5 : $min\_water\_level \geq M1 \wedge max\_water\_level \leq M2$
grd6 : $max\_water\_level \leq N2$

then

act1 : $pump\_ctrl : |pump\_ctrl' \in PUMP\_MODE \wedge$
$(((min\_water\_level \geq N1 \Rightarrow pump\_ctrl' = OFF) \wedge$
$(min\_water\_level < N1 \Rightarrow pump\_ctrl' = ON)) \vee$
$pump\_ctrl' = pump\_ctrl)$
act2 : $valve\_ctrl := CLOSED$
act3 : $preop\_flag := FALSE$
act4 : $phase := PRED$

end

Event *Operational* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $failure = FALSE$
grd3 : $stop = FALSE$
grd4 : $preop\_flag = FALSE$
grd5 : $min\_water\_level \geq M1 \wedge max\_water\_level \leq M2$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl : |pump\_ctrl' \in PUMP\_MODE \wedge$
$(((min\_water\_level \geq M1 \wedge max\_water\_level < N1 \Rightarrow$
$pump\_ctrl' = ON) \wedge$
$(min\_water\_level > N2 \wedge max\_water\_level \leq M2 \Rightarrow$
$pump\_ctrl' = OFF) \wedge$
$(min\_water\_level \geq N1 \wedge max\_water\_level \leq N2 \Rightarrow$
$pump\_ctrl' = pump\_ctrl)) \vee$
$pump\_ctrl' = pump\_ctrl)$

end

Event *EmergencyStop* $\widehat{=}$
refines *EmergencyStop*

when

grd1 : $phase = CONT$
grd2 : $failure = TRUE$
grd3 : $stop = FALSE$

then

act1 : $stop := TRUE$
act2 : $pump\_ctrl := OFF$
act3 : $valve\_ctrl := CLOSED$

end

Event *Prediction* $\widehat{=}$
refines *Prediction*

when

grd1 : $phase = PRED$
grd2 : $stop = FALSE$

then

act1 : $phase := ENV$
act2 : $min\_water\_level, max\_water\_level :|$
$min\_water\_level' \in 0 \,..\, C \wedge max\_water\_level' \in 0 \,..\, C \wedge$
$min\_water\_level' \leq max\_water\_level'$

end

END

MACHINE M2 // The second refinement: introducing failure assumptions
REFINES M1
SEES C1
VARIABLES

$phase$
$stop$
$water\_level$
$pump\_ctrl$
$valve\_ctrl$
$wl\_sensor\_failure$
$pump\_failure$
$so\_sensor\_failure$
$preop\_flag$
min_water_level
max_water_level

INVARIANTS

inv1 : $wl\_sensor\_failure \in BOOL$
inv2 : $pump\_failure \in BOOL$
inv3 : $so\_sensor\_failure \in BOOL$
inv4 : $(phase \neq DET \land phase \neq ENV) \Rightarrow (failure = TRUE \Leftrightarrow ((wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)) \lor WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE))$ // inv2.1
inv5 : $stop = FALSE \land phase = PRED \Rightarrow \neg(wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
inv6 : $wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE) \land phase = PRED \Rightarrow stop = TRUE$
inv7 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE \land phase = PRED \Rightarrow stop = TRUE$
inv8 : $phase = PRED \land valve\_ctrl = OPEN \Rightarrow WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land \neg(wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
inv9 : $phase = PRED \land pump\_ctrl = ON \Rightarrow WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land \neg(wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
inv10 : $phase \neq CONT \Rightarrow (wl\_sensor\_failure = FALSE \lor (pump\_failure = FALSE \land so\_sensor\_failure = FALSE))$

`inv11` : $(pump\_failure = FALSE \land phase = PRED \land$
$max\_water\_level < N1 \land min\_water\_level \geq M1) \Rightarrow$
$pump\_ctrl = ON$ // inv2.2

`inv12` : $(pump\_failure = FALSE \land phase = PRED \land$
$min\_water\_level > N2 \land max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl = OFF$ // inv2.3

`inv13` : $stop = FALSE \land phase = PRED \Rightarrow$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

`inv14` : $phase \neq ENV \Rightarrow$
$((WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE) \Leftrightarrow$
$(min\_water\_level \geq M1 \land max\_water\_level \leq M2))$

`inv15` : $phase \neq ENV \Rightarrow$
$((WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE) \Leftrightarrow$
$(min\_water\_level < M1 \lor max\_water\_level > M2))$

EVENTS
Initialisation

begin

`act1` : $phase := ENV$
`act2` : $stop := FALSE$
`act3` : $water\_level, min\_water\_level, max\_water\_level :|$
$water\_level' \in M1 \,..\, M2 \land min\_water\_level' \in M1 \,..\, M2 \land$
$max\_water\_level' \in M1 \,..\, M2 \land$
$min\_water\_level' \leq max\_water\_level' \land$
$min\_water\_level' = water\_level' \land max\_water\_level' = water\_level'$
`act4` : $pump\_ctrl := OFF$
`act5` : $valve\_ctrl := CLOSED$
`act6` : $wl\_sensor\_failure := FALSE$
`act7` : $pump\_failure := FALSE$
`act8` : $so\_sensor\_failure := FALSE$
`act9` : $preop\_flag := TRUE$

end

Event *Environment* $\widehat{=}$
refines *Environment*

when

`grd1` : $phase = ENV$
`grd2` : $stop = FALSE$

then

`act1` : $phase := DET$
`act2` : $water\_level :\in \mathbb{Z}$

end

Event *Detection_OK* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $\neg(wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
grd3 : $stop = FALSE$
grd4 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure :| Stable(wl\_sensor\_failure \mapsto wl\_sensor\_failure') = TRUE \land Stable(pump\_failure \mapsto pump\_failure') = TRUE \land Stable(so\_sensor\_failure \mapsto so\_sensor\_failure') = TRUE \land \neg(wl\_sensor\_failure' = TRUE \land (pump\_failure' = TRUE \lor so\_sensor\_failure' = TRUE))$

end

Event *Detection_NOK_safety_bounds* $\widehat{=}$
refines *Detection_NOK_1*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE$

then

act1 : $phase := CONT$

end

Event *Detection_NOK* $\widehat{=}$
refines *Detection_NOK_2*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

with

failure' : $failure' = TRUE$

then

act1 : $phase := CONT$

act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure :|$
$((Stable(wl\_sensor\_failure \mapsto wl\_sensor\_failure') = TRUE \land$
$Stable(pump\_failure \mapsto pump\_failure') = TRUE \land$
$Stable(so\_sensor\_failure \mapsto so\_sensor\_failure') = TRUE) \land$
$(wl\_sensor\_failure' = TRUE \land (pump\_failure' = TRUE \lor$
$so\_sensor\_failure' = TRUE)))$

end

Event *PreOperational1* $\widehat{=}$
refines *PreOperational1*

when

grd1 : $phase = CONT$
grd2 : $\neg(wl\_sensor\_failure = TRUE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
grd3 : $stop = FALSE$
grd4 : $max\_water\_level > N2 \land min\_water\_level > N1$
grd5 : $preop\_flag = TRUE$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $valve\_ctrl := OPEN$
act2 : $phase := PRED$
act3 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = FALSE \Rightarrow pump\_ctrl' = OFF) \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl)$

end

Event *PreOperational2* $\widehat{=}$
refines *PreOperational2*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = TRUE$
grd4 : $max\_water\_level \leq N2$
grd5 : $\neg(wl\_sensor\_failure = TRUE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq N1 \Rightarrow$
$pump\_ctrl' = OFF) \land$
$(pump\_failure = FALSE \land min\_water\_level < N1 \Rightarrow$
$pump\_ctrl' = ON)$

act2 : $phase := PRED$
act3 : $valve\_ctrl := CLOSED$
act4 : $preop\_flag := FALSE$

end

Event *Normal_Operational* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = FALSE \wedge pump\_failure = FALSE \wedge so\_sensor\_failure = FALSE$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \wedge ((min\_water\_level \geq M1 \wedge max\_water\_level < N1) \Rightarrow pump\_ctrl' = ON) \wedge ((min\_water\_level > N2 \wedge max\_water\_level \leq M2) \Rightarrow pump\_ctrl' = OFF) \wedge ((min\_water\_level \geq N1 \wedge max\_water\_level \leq N2) \Rightarrow pump\_ctrl' = pump\_ctrl)$

end

Event *Degraded_Operational* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = FALSE \wedge (pump\_failure = TRUE \vee so\_sensor\_failure = TRUE)$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$

act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq M1 \land$
$max\_water\_level < N1 \Rightarrow pump\_ctrl' = ON) \land$
$(pump\_failure = FALSE \land min\_water\_level > N2 \land$
$max\_water\_level \leq M2 \Rightarrow pump\_ctrl' = OFF) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq N1 \land$
$max\_water\_level \leq N2 \Rightarrow pump\_ctrl' = pump\_ctrl)$

end

Event *Rescue_Operational* $\widehat{=}$
refines *Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = TRUE \land$
$pump\_failure = FALSE \land so\_sensor\_failure = FALSE$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$((min\_water\_level \geq M1 \land max\_water\_level < N1) \Rightarrow$
$pump\_ctrl' = ON) \land$
$((min\_water\_level > N2 \land max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl' = OFF) \land$
$((min\_water\_level \geq N1 \land max\_water\_level \leq N2) \Rightarrow$
$pump\_ctrl' = pump\_ctrl)$

end

Event *EmergencyStop* $\widehat{=}$
refines *EmergencyStop*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $(wl\_sensor\_failure = TRUE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)) \lor$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE$

then

act1 : $stop := TRUE$
act2 : $pump\_ctrl := OFF$
act3 : $valve\_ctrl := CLOSED$

end

Event *Prediction* $\widehat{=}$
refines *Prediction*

when

grd1 : $phase = PRED$
grd2 : $stop = FALSE$

then

act1 : $phase := ENV$
act2 : $min\_water\_level, max\_water\_level :|$
$min\_water\_level' \in 0 \,..\, C \land max\_water\_level' \in 0 \,..\, C \land$
$min\_water\_level' \leq max\_water\_level'$

end

END

CONTEXT C2
EXTENDS C1
CONSTANTS

$U1$
$U2$
P
W
T
E
WL_min
WL_max
SO_min
SO_max
WL
SO

AXIOMS

axm1 : $U1 \in \mathbb{N}_1$
axm2 : $U2 \in \mathbb{N}_1$
axm3 : $P \in \mathbb{N}_1$

axm4 : $W \in \mathbb{N}_1$

axm5 : $T \in \mathbb{N}_1$

axm6 : $E \in \mathbb{N}_1$

axm7 : $WL\_min \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

axm8 : $\forall x, y, m, n \cdot x \in \mathbb{N} \land y \in \mathbb{N} \land m \in \mathbb{N} \land n \in \mathbb{N} \Rightarrow$
$WL\_min(x \mapsto y \mapsto m \mapsto n) \in 0 \,..\, C$

axm9 : $WL\_max \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

axm10 : $\forall x, y, m, n \cdot x \in \mathbb{N} \land y \in \mathbb{N} \land m \in \mathbb{N} \land n \in \mathbb{N} \Rightarrow$
$WL\_max(x \mapsto y \mapsto m \mapsto n) \in 0 \,..\, C$

axm11 : $\forall x1, x2, y1, y2, m, n \cdot x1 \in \mathbb{N} \land x2 \in \mathbb{N} \land x1 \leq x2$
$\land\, y1 \in \mathbb{N} \land y2 \in \mathbb{N} \land m \in \mathbb{N} \land n \in \mathbb{N} \Rightarrow$
$WL\_min(x1 \mapsto y1 \mapsto m \mapsto n) \leq$
$WL\_max(x2 \mapsto y2 \mapsto m \mapsto n)$

axm12 : $SO\_min \in \mathbb{N} \rightarrow \mathbb{N}$

axm13 : $\forall x \cdot x \in \mathbb{N} \Rightarrow SO\_min(x) \in 0 \,..\, W$

axm14 : $SO\_max \in \mathbb{N} \rightarrow \mathbb{N}$

axm15 : $\forall x \cdot x \in \mathbb{N} \Rightarrow SO\_max(x) \in 0 \,..\, W$

axm16 : $\forall x1, x2 \cdot x1 \in \mathbb{N} \land x2 \in \mathbb{N} \Rightarrow SO\_min(x1) \leq SO\_max(x2)$

axm17 : $WL \in 0 \,..\, C \times 0 \,..\, W \times 0 \,..\, P * T \times 0 \,..\, E * T \rightarrow \mathbb{P}_1(0 \,..\, C)$

axm18 : $\forall x, y, m, n \cdot x \in 0 \,..\, C \land y \in 0 \,..\, W \land m \in 0 \,..\, P * T \land n \in 0 \,..\, E * T \Rightarrow$
$WL(x \mapsto y \mapsto m \mapsto n) \subseteq 0 \,..\, C$

axm19 : $SO \in 0 \,..\, W \rightarrow \mathbb{P}_1(0 \,..\, W)$

axm20 : $\forall x \cdot x \in 0 \,..\, W \Rightarrow SO(x) \subseteq 0 \,..\, W$

axm21 : $\forall x, y, m, n \cdot x \in 0 \,..\, C \land y \in 0 \,..\, W \land m \in 0 \,..\, P * T \land n \in 0 \,..\, E * T \Rightarrow$
$(WL(x \mapsto y \mapsto m \mapsto n) = \{0\} \Rightarrow SO(y) = \{0\})$

axm22 : $\forall x, y, m, n \cdot x \in 0 \,..\, C \land y \in 0 \,..\, W \land$
$m \in 0 \,..\, P * T \land n \in 0 \,..\, E * T \Rightarrow$
$((WL\_min(x \mapsto y \mapsto m \mapsto n) = 0 \,\land$
$WL\_max(x \mapsto y \mapsto m \mapsto n) = 0) \Rightarrow$
$(SO\_min(y) = 0 \land SO\_max(y) = 0))$

END

MACHINE M3 // The third refinement: unfolding physical environment
REFINES M2
SEES C2
VARIABLES

$phase$
$stop$
$water\_level$
$steam\_output$
$pump$
$water\_output$
$pump\_ctrl$
$valve\_ctrl$
$wl\_sensor\_failure$
$pump\_failure$
$so\_sensor\_failure$
min_water_level
max_water_level
min_steam_output
max_steam_output
$preop\_flag$

INVARIANTS

inv1 : $pump \in (0 \mathrel{..} P * T)$
inv2 : $water\_level \in \mathbb{Z}$
inv3 : $water\_output \in 0 \mathrel{..} E * T$
inv4 : $steam\_output \in \mathbb{Z}$
inv5 : $min\_water\_level \in 0 \mathrel{..} C$
inv6 : $max\_water\_level \in 0 \mathrel{..} C$
inv7 : $min\_steam\_output \in 0 \mathrel{..} W$
inv8 : $max\_steam\_output \in 0 \mathrel{..} W$
inv9 : $min\_water\_level \leq max\_water\_level$
inv10 : $min\_steam\_output \leq max\_steam\_output$
inv11 : $phase = DET \Rightarrow (valve\_ctrl = OPEN \Rightarrow water\_output > 0)$
inv12 : $phase = DET \Rightarrow (valve\_ctrl = CLOSED \Rightarrow water\_output = 0)$
inv13 : $phase = PRED \Rightarrow$
$(min\_water\_level < N1 \Rightarrow valve\_ctrl = CLOSED)$
inv14 : $(phase = PRED \land pump\_failure = FALSE) \Rightarrow$
$(min\_water\_level > N2 \Rightarrow pump\_ctrl = OFF)$

inv15 : $((min\_water\_level > water\_level \lor max\_water\_level < water\_level \lor water\_level \notin 0\,..\,C) \land phase = CONT \land WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE) \Rightarrow wl\_sensor\_failure = TRUE$

inv16 : $((min\_steam\_output > steam\_output \lor max\_steam\_output < steam\_output \lor steam\_output \notin 0\,..\,W \lor (pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)) \land phase = CONT \land WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE) \Rightarrow (so\_sensor\_failure = TRUE \lor pump\_failure = TRUE)$

inv17 : $wl\_sensor\_failure = FALSE \land phase \neq DET \land WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \Rightarrow water\_level \in 0\,..\,C$

inv18 : $so\_sensor\_failure = FALSE \land phase \neq DET \land WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \Rightarrow steam\_output \in 0\,..\,W$

EVENTS

Initialisation

begin

act1 : $phase := ENV$

act2 : $stop := FALSE$

act3 : $pump := 0$

act4 : $water\_output := 0$

act5 : $pump\_ctrl := OFF$

act6 : $valve\_ctrl := CLOSED$

act7 : $wl\_sensor\_failure := FALSE$

act8 : $pump\_failure := FALSE$

act9 : $so\_sensor\_failure := FALSE$

act10 : $steam\_output, min\_steam\_output, max\_steam\_output :| steam\_output' \in 0\,..\,W \land min\_steam\_output' \in 0\,..\,W \land max\_steam\_output' \in 0\,..\,W \land min\_steam\_output' \leq max\_steam\_output' \land min\_steam\_output' = steam\_output' \land max\_steam\_output' = steam\_output'$

act11 : $preop\_flag := TRUE$

act12 : $water\_level, min\_water\_level, max\_water\_level :| water\_level' \in M1\,..\,M2 \land min\_water\_level' \in M1\,..\,M2 \land max\_water\_level' \in M1\,..\,M2 \land min\_water\_level' \leq max\_water\_level' \land min\_water\_level' = water\_level' \land max\_water\_level' = water\_level'$

end

Event *Environment* $\widehat{=}$
refines *Environment*

when

grd1 : $phase = ENV$
grd2 : $stop = FALSE$

then

act1 : $phase := DET$
act2 : $water\_level :\in \mathbb{Z}$
act3 : $pump :\in (0 \mathinner{..} P * T)$
act4 : $steam\_output :\in \mathbb{Z}$
act5 : $water\_output : |water\_output' \in 0 \mathinner{..} E * T \land$
$(valve\_ctrl = OPEN \Rightarrow water\_output' = E * T) \land$
$(\neg(valve\_ctrl = OPEN) \Rightarrow water\_output' = 0)$

end

Event *Detection_OK_no_F* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \mathinner{..} C$
grd5 : $steam\_output \in 0 \mathinner{..} W$
grd6 : $min\_water\_level \leq water\_level \land water\_level \leq max\_water\_level$
grd7 : $min\_steam\_output \leq steam\_output \land$
$steam\_output \leq max\_steam\_output$
grd8 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd9 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd10 : $wl\_sensor\_failure = FALSE \land pump\_failure = FALSE \land$
$so\_sensor\_failure = FALSE$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure : |$
$(wl\_sensor\_failure' = FALSE \land pump\_failure' = FALSE \land$
$so\_sensor\_failure' = FALSE)$

end

Event *Detection_OK_new_F_WL_NoF_p_so* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd4 : $steam\_output \in 0 \mathrel{..} W$
grd5 : $min\_water\_level > water\_level \lor$
$water\_level > max\_water\_level \lor water\_level \notin 0 \mathrel{..} C$
grd6 : $min\_steam\_output \leq steam\_output \land$
$steam\_output \leq max\_steam\_output$
grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd9 : $pump\_failure = FALSE \land so\_sensor\_failure = FALSE$
grd10 : $wl\_sensor\_failure = FALSE$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure :\mid$
$wl\_sensor\_failure' = TRUE \land pump\_failure' = FALSE \land$
$so\_sensor\_failure' = FALSE$

end

Event *Detection_OK_det_F_WL_NoF_p_so* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $steam\_output \in 0 \mathrel{..} W$
grd5 : $min\_steam\_output \leq steam\_output \land$
$steam\_output \leq max\_steam\_output$
grd6 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd7 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd8 : $pump\_failure = FALSE \land so\_sensor\_failure = FALSE$
grd9 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $pump\_failure, so\_sensor\_failure :\mid$
$pump\_failure' = FALSE \land so\_sensor\_failure' = FALSE$

end

Event *Detection_OK_NoF_WL_F_p* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \mathrel{..} C$
grd5 : $min\_water\_level \leq water\_level \land water\_level \leq max\_water\_level$

grd6 : $min\_steam\_output \leq steam\_output \wedge$
$steam\_output \leq max\_steam\_output \wedge steam\_output \in 0 \,..\, W$
grd7 : $(pump\_ctrl = ON \wedge pump = 0) \vee (pump\_ctrl = OFF \wedge pump > 0)$
grd8 : $wl\_sensor\_failure = FALSE$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$

end

Event *Detection_OK_NoF_WL_F_so* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \,..\, C$
grd5 : $min\_water\_level \leq water\_level \wedge water\_level \leq max\_water\_level$
grd6 : $(min\_steam\_output > steam\_output) \vee$
$(steam\_output > max\_steam\_output) \vee steam\_output \notin 0 \,..\, W$
grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd9 : $wl\_sensor\_failure = FALSE$

then

act1 : $phase := CONT$
act2 : $so\_sensor\_failure := TRUE$

end

Event *Detection_OK_NoF_WL_F_p_so_both* $\widehat{=}$
refines *Detection_OK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \,..\, C$
grd5 : $min\_water\_level \leq water\_level \wedge water\_level \leq max\_water\_level$
grd6 : $(pump\_ctrl = ON \wedge pump = 0) \vee (pump\_ctrl = OFF \wedge pump > 0)$
grd7 : $wl\_sensor\_failure = FALSE$
grd8 : $(min\_steam\_output > steam\_output) \vee$
$(steam\_output > max\_steam\_output) \vee steam\_output \notin 0 \,..\, W$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$

act3 : $so\_sensor\_failure := TRUE$

end

Event *Detection_NOK_new_F_WL_F_p* $\widehat{=}$

refines *Detection_NOK*

when

grd1 : $phase = DET$

grd2 : $stop = FALSE$

grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor water\_level \notin 0 \mathinner{..} C$

grd4 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $wl\_sensor\_failure = FALSE$

grd7 : $min\_steam\_output \leq steam\_output \land steam\_output \leq max\_steam\_output \land steam\_output \in 0 \mathinner{..} W$

then

act1 : $phase := CONT$

act2 : $wl\_sensor\_failure := TRUE$

act3 : $pump\_failure := TRUE$

end

Event *Detection_NOK_new_F_WL_F_so* $\widehat{=}$

refines *Detection_NOK*

when

grd1 : $phase = DET$

grd2 : $stop = FALSE$

grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor water\_level \notin 0 \mathinner{..} C$

grd4 : $(min\_steam\_output > steam\_output) \lor (steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \mathinner{..} W$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $wl\_sensor\_failure = FALSE$

grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$

grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$

then

act1 : $phase := CONT$

act2 : $wl\_sensor\_failure := TRUE$

act3 : $so\_sensor\_failure := TRUE$

end

Event *Detection_NOK_new_F_WL_F_p_so_both* $\widehat{=}$

refines *Detection_NOK*

when

grd1 : $phase = DET$

grd2 : $stop = FALSE$

grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor water\_level \notin 0 \,..\, C$

grd4 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $wl\_sensor\_failure = FALSE$

grd7 : $(min\_steam\_output > steam\_output) \lor (steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$

then

act1 : $phase := CONT$

act2 : $wl\_sensor\_failure := TRUE$

act3 : $pump\_failure := TRUE$

act4 : $so\_sensor\_failure := TRUE$

end

Event *Detection_NOK_det_F_WL_F_p* $\widehat{=}$
refines *Detection_NOK*

when

grd1 : $phase = DET$

grd2 : $stop = FALSE$

grd3 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$

grd4 : $min\_steam\_output \leq steam\_output \land steam\_output \leq max\_steam\_output \land steam\_output \in 0 \,..\, W$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$

act2 : $pump\_failure := TRUE$

end

Event *Detection_NOK_det_F_WL_F_so* $\widehat{=}$
refines *Detection_NOK*

when

grd1 : $phase = DET$

grd2 : $stop = FALSE$

grd3 : $(min\_steam\_output > steam\_output) \lor (steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$

grd4 : $(pump\_ctrl = ON \Rightarrow pump > 0)$

grd5 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$

grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd7 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $so\_sensor\_failure := TRUE$

end

Event *Detection_NOK_det_F_WL_F_p_so_both* $\widehat{=}$
refines *Detection_NOK*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd4 : $(min\_steam\_output > steam\_output) \lor$ $(steam\_output > max\_steam\_output) \lor steam\_output \notin 0\,..\,W$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$
act3 : $so\_sensor\_failure := TRUE$

end

Event *Detection_NOK_safety_bounds_WL* $\widehat{=}$
refines *Detection_NOK_safety_bounds*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE$

then

act1 : $phase := CONT$

end

Event *PreOperational1* $\widehat{=}$
refines *PreOperational1*

when

grd1 : $phase = CONT$
grd2 : $\neg(wl\_sensor\_failure = TRUE \land$ $(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
grd3 : $stop = FALSE$
grd4 : $max\_water\_level > N2 \land min\_water\_level > N1$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $preop\_flag = TRUE$

then

act1 : $valve\_ctrl := OPEN$
act2 : $phase := PRED$

act3 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = FALSE \Rightarrow pump\_ctrl' = OFF) \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl)$

end

Event *PreOperational2* $\widehat{=}$
refines *PreOperational2*

when

grd1 : $phase = CONT$
grd2 : $\neg(wl\_sensor\_failure = TRUE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE))$
grd3 : $stop = FALSE$
grd4 : $max\_water\_level \leq N2$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $preop\_flag = TRUE$

then

act1 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq N1 \Rightarrow$
$pump\_ctrl' = OFF) \land$
$(pump\_failure = FALSE \land min\_water\_level < N1 \Rightarrow$
$pump\_ctrl' = ON)$
act2 : $phase := PRED$
act3 : $valve\_ctrl := CLOSED$
act4 : $preop\_flag := FALSE$

end

Event *Normal_Operational* $\widehat{=}$
refines *Normal_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = FALSE \land$
$pump\_failure = FALSE \land so\_sensor\_failure = FALSE$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$

act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$((min\_water\_level \geq M1 \land max\_water\_level < N1) \Rightarrow$
$pump\_ctrl' = ON) \land$
$((min\_water\_level > N2 \land max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl' = OFF) \land$
$((min\_water\_level \geq N1 \land max\_water\_level \leq N2) \Rightarrow$
$pump\_ctrl' = pump\_ctrl)$

end

Event *Degraded_Operational* $\widehat{=}$
refines *Degraded_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = FALSE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq M1 \land$
$max\_water\_level < N1 \Rightarrow pump\_ctrl' = ON) \land$
$(pump\_failure = FALSE \land min\_water\_level > N2 \land$
$max\_water\_level \leq M2 \Rightarrow pump\_ctrl' = OFF) \land$
$(pump\_failure = FALSE \land min\_water\_level \geq N1 \land$
$max\_water\_level \leq N2 \Rightarrow pump\_ctrl' = pump\_ctrl)$

end

Event *Rescue_Operational* $\widehat{=}$
refines *Rescue_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $wl\_sensor\_failure = TRUE \land$
$pump\_failure = FALSE \land so\_sensor\_failure = FALSE$
grd5 : $valve\_ctrl = CLOSED$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

then

act1 : $phase := PRED$

act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \land$
$((min\_water\_level \geq M1 \land max\_water\_level < N1) \Rightarrow$
$pump\_ctrl' = ON) \land$
$((min\_water\_level > N2 \land max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl' = OFF) \land$
$((min\_water\_level \geq N1 \land max\_water\_level \leq N2) \Rightarrow$
$pump\_ctrl' = pump\_ctrl)$

end

Event *EmergencyStop* $\widehat{=}$
refines *EmergencyStop*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $(wl\_sensor\_failure = TRUE \land$
$(pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)) \lor$
$WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE$

then

act1 : $stop := TRUE$
act2 : $pump\_ctrl := OFF$
act3 : $valve\_ctrl := CLOSED$

end

Event *Prediction* $\widehat{=}$
refines *Prediction*

when

grd1 : $phase = PRED$
grd2 : $stop = FALSE$

then

act1 : $phase := ENV$

act2 : $min\_water\_level, max\_water\_level :|$
$min\_water\_level' \in 0\,..\,C \wedge max\_water\_level' \in 0\,..\,C \wedge$

$((wl\_sensor\_failure = FALSE \wedge so\_sensor\_failure = FALSE) \Rightarrow$
$(min\_water\_level' = WL\_min(water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output) \wedge$
$max\_water\_level' = WL\_max(water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output))) \wedge$

$((wl\_sensor\_failure = TRUE \wedge so\_sensor\_failure = FALSE) \Rightarrow$
$(min\_water\_level' = WL\_min(min\_water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output) \wedge$
$max\_water\_level' = WL\_max(max\_water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output))) \wedge$

$((wl\_sensor\_failure = FALSE \wedge so\_sensor\_failure = TRUE) \Rightarrow$
$(min\_water\_level' = WL\_min(water\_level \mapsto min\_steam\_output \mapsto$
$pump \mapsto water\_output) \wedge$
$max\_water\_level' = WL\_max(water\_level \mapsto max\_steam\_output \mapsto$
$pump \mapsto water\_output \wedge$

$min\_water\_level' \leq max\_water\_level'$

act3 : $min\_steam\_output, max\_steam\_output :|$
$min\_steam\_output' \in 0\,..\,W \wedge max\_steam\_output' \in 0\,..\,W \wedge$

$(so\_sensor\_failure = FALSE \Rightarrow$
$(min\_steam\_output' = SO\_min(steam\_output) \wedge$
$max\_steam\_output' = SO\_max(steam\_output))) \wedge$

$(so\_sensor\_failure = TRUE \Rightarrow$
$(min\_steam\_output' = SO\_min(min\_steam\_output) \wedge$
$max\_steam\_output' = SO\_max(max\_steam\_output))) \wedge$

$min\_steam\_output' \leq max\_steam\_output'$

end

END

CONTEXT C3
EXTENDS C2
SETS

$MODE$

CONSTANTS

$Initialisation$
$Normal$
$Degraded$
$Rescue$
$Emergency\_Stop$

AXIOMS

axm1 : $partition(MODE, \{Initialisation\}, \{Normal\},$
$\{Degraded\}, \{Rescue\}, \{Emergency\_Stop\})$

END

MACHINE M4 // The fourth refinement: introducing system modes
REFINES M3
SEES C3
VARIABLES

$phase$
$stop$
$water\_level$
$steam\_output$
$pump$
$water\_output$
$pump\_ctrl$
$valve\_ctrl$
$wl\_sensor\_failure$
$pump\_failure$
$so\_sensor\_failure$
$mode$
$preop\_flag$

min_water_level

max_water_level

min_steam_output

max_steam_output

INVARIANTS

inv1 : $mode \in MODE$

inv2 : $preop\_flag \in BOOL$

inv3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land wl\_sensor\_failure = FALSE \land pump\_failure = FALSE \land so\_sensor\_failure = FALSE \Rightarrow mode = Initialisation \lor mode = Normal$

inv4 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land wl\_sensor\_failure = TRUE \land pump\_failure = FALSE \land so\_sensor\_failure = FALSE \Rightarrow mode = Initialisation \lor mode = Rescue$

inv5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land wl\_sensor\_failure = FALSE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE) \Rightarrow mode = Initialisation \lor mode = Degraded$

inv6 : $mode = Normal \Rightarrow wl\_sensor\_failure = FALSE \land pump\_failure = FALSE \land so\_sensor\_failure = FALSE$ // inv4.1

inv7 : $mode = Degraded \Rightarrow wl\_sensor\_failure = FALSE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)$ // inv4.2

inv8 : $mode = Rescue \Rightarrow wl\_sensor\_failure = TRUE \land pump\_failure = FALSE \land so\_sensor\_failure = FALSE$ // inv4.3

inv9 : $mode = Emergency\_Stop \Rightarrow ((wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)) \lor WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE)$ // inv4.4

inv10 : $phase \neq ENV \land phase \neq DET \land ((wl\_sensor\_failure = TRUE \land (pump\_failure = TRUE \lor so\_sensor\_failure = TRUE)) \lor WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE) \Rightarrow mode = Emergency\_Stop$ // inv4.5

inv11 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE \land stop = FALSE \land (wl\_sensor\_failure = FALSE \lor (pump\_failure = FALSE \land so\_sensor\_failure = FALSE)) \Rightarrow mode \neq Emergency\_Stop$ // inv4.6

`inv12` : $\forall p' \cdot p' \in \{stop' \mapsto pump\_ctrl' \mapsto valve\_ctrl' |$
$stop' \in BOOL \land pump\_ctrl' \in PUMP\_MODE \land$
$valve\_ctrl' \in VALVE\_MODE \land$
$(\exists phase, stop, pump\_ctrl, valve\_ctrl, mode \cdot$
$phase \in PHASE \land stop \in BOOL \land pump\_ctrl \in PUMP\_MODE \land$
$valve\_ctrl \in VALVE\_MODE \land mode \in MODE \land$
$(phase = CONT \land stop = FALSE \land mode = Emergency\_Stop) \land$
$(stop' = TRUE \land pump\_ctrl' = OFF \land valve\_ctrl' = CLOSED))\} \Rightarrow$

$p' \in \{stop' \mapsto pump\_ctrl' \mapsto valve\_ctrl' | stop' \in BOOL \land$
$pump\_ctrl' \in PUMP\_MODE \land valve\_ctrl' \in VALVE\_MODE \land$
$(stop' = TRUE)\}$ // thm4.1

EVENTS
Initialisation
begin

`act1` : $phase := ENV$
`act2` : $stop := FALSE$
`act3` : $pump := 0$
`act4` : $water\_output := 0$
`act5` : $pump\_ctrl := OFF$
`act6` : $valve\_ctrl := CLOSED$
`act7` : $wl\_sensor\_failure := FALSE$
`act8` : $pump\_failure := FALSE$
`act9` : $so\_sensor\_failure := FALSE$
`act10` : $steam\_output, min\_steam\_output, max\_steam\_output : |$
$steam\_output' \in 0 \,..\, W \land min\_steam\_output' \in 0 \,..\, W \land$
$max\_steam\_output' \in 0 \,..\, W \land$
$min\_steam\_output' \leq max\_steam\_output' \land$
$min\_steam\_output' = steam\_output' \land$
$max\_steam\_output' = steam\_output'$
`act11` : $preop\_flag := TRUE$
`act12` : $mode := Initialisation$
`act13` : $water\_level, min\_water\_level, max\_water\_level : |$
$water\_level' \in M1 \,..\, M2 \land min\_water\_level' \in M1 \,..\, M2 \land$
$max\_water\_level' \in M1 \,..\, M2 \land$
$min\_water\_level' \leq max\_water\_level' \land$
$min\_water\_level' = water\_level' \land max\_water\_level' = water\_level'$

end

Event *Environment* $\widehat{=}$
extends *Environment*
when

`grd1` : $phase = ENV$

grd2 : $stop = FALSE$
grd3 : $mode = Initialisation$

then

act1 : $phase := DET$
act2 : $water\_level :\in \mathbb{Z}$
act3 : $pump :\in (0 \,..\, P * T)$
act4 : $steam\_output :\in \mathbb{Z}$
act5 : $water\_output : |water\_output' \in 0 \,..\, E * T \wedge$
$(valve\_ctrl = OPEN \Rightarrow water\_output' = E * T) \wedge$
$(\neg(valve\_ctrl = OPEN) \Rightarrow water\_output' = 0)$

end

Event *Detection_OK_no_F* $\widehat{=}$
extends *Detection_OK_no_F*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \,..\, C$
grd5 : $steam\_output \in 0 \,..\, W$
grd6 : $min\_water\_level \leq water\_level \wedge water\_level \leq max\_water\_level$
grd7 : $min\_steam\_output \leq steam\_output \wedge$
$steam\_output \leq max\_steam\_output$
grd8 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd9 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd10 : $wl\_sensor\_failure = FALSE \wedge pump\_failure = FALSE \wedge$
$so\_sensor\_failure = FALSE$
grd11 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure : |$
$(wl\_sensor\_failure' = FALSE \wedge$
$pump\_failure' = FALSE \wedge so\_sensor\_failure' = FALSE)$
act3 : $mode := Normal$

end

Event *Detection_OK_new_F_WL_NoF_p_so* $\widehat{=}$
extends *Detection_OK_new_F_WL_NoF_p_so*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $steam\_output \in 0 \,..\, W$

grd5 : $min\_water\_level > water\_level \vee water\_level > max\_water\_level \vee water\_level \notin 0 \,..\, C$
grd6 : $min\_steam\_output \leq steam\_output \wedge steam\_output \leq max\_steam\_output$
grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd9 : $pump\_failure = FALSE \wedge so\_sensor\_failure = FALSE$
grd10 : $wl\_sensor\_failure = FALSE$
grd11 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure, pump\_failure, so\_sensor\_failure :| wl\_sensor\_failure' = TRUE \wedge pump\_failure' = FALSE \wedge so\_sensor\_failure' = FALSE$
act3 : $mode := Rescue$

end

Event *Detection_OK_det_F_WL_NoF_p_so* $\widehat{=}$
extends *Detection_OK_det_F_WL_NoF_p_so*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $steam\_output \in 0 \,..\, W$
grd5 : $min\_steam\_output \leq steam\_output \wedge steam\_output \leq max\_steam\_output$
grd6 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd7 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd8 : $pump\_failure = FALSE \wedge so\_sensor\_failure = FALSE$
grd9 : $wl\_sensor\_failure = TRUE$
grd10 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $pump\_failure, so\_sensor\_failure :| pump\_failure' = FALSE \wedge so\_sensor\_failure' = FALSE$
act3 : $mode := Rescue$

end

Event *Detection_OK_NoF_WL_F_p* $\widehat{=}$
extends *Detection_OK_NoF_WL_F_p*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd4 : $water\_level \in 0 \mathrel{..} C$
grd5 : $min\_water\_level \leq water\_level \land water\_level \leq max\_water\_level$
grd6 : $min\_steam\_output \leq steam\_output \land$
$steam\_output \leq max\_steam\_output \land steam\_output \in 0 \mathrel{..} W$
grd7 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd8 : $wl\_sensor\_failure = FALSE$
grd9 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$
act3 : $mode := Degraded$

end

Event *Detection_OK_NoF_WL_F_so* $\widehat{=}$
extends *Detection_OK_NoF_WL_F_so*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \mathrel{..} C$
grd5 : $min\_water\_level \leq water\_level \land water\_level \leq max\_water\_level$
grd6 : $(min\_steam\_output > steam\_output) \lor$
$(steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \mathrel{..} W$
grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd9 : $wl\_sensor\_failure = FALSE$
grd10 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $so\_sensor\_failure := TRUE$
act3 : $mode := Degraded$

end

Event *Detection_OK_NoF_WL_F_p_so_both* $\widehat{=}$
extends *Detection_OK_NoF_WL_F_p_so_both*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd4 : $water\_level \in 0 \mathrel{..} C$
grd5 : $min\_water\_level \leq water\_level \land water\_level \leq max\_water\_level$
grd6 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd7 : $wl\_sensor\_failure = FALSE$

grd8 : $(min\_steam\_output > steam\_output) \lor$
$(steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$
grd9 : $mode = Initialisation$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$
act3 : $so\_sensor\_failure := TRUE$
act4 : $mode := Degraded$

end

Event *Detection_NOK_new_F_WL_F_p* $\widehat{=}$
extends *Detection_NOK_new_F_WL_F_p*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor$
$water\_level \notin 0 \,..\, C$
grd4 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = FALSE$
grd7 : $min\_steam\_output \leq steam\_output \land$
$steam\_output \leq max\_steam\_output \land steam\_output \in 0 \,..\, W$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure := TRUE$
act3 : $pump\_failure := TRUE$
act4 : $mode := Emergency\_Stop$

end

Event *Detection_NOK_new_F_WL_F_so* $\widehat{=}$
extends *Detection_NOK_new_F_WL_F_so*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor$
$water\_level \notin 0 \,..\, C$
grd4 : $(min\_steam\_output > steam\_output) \lor$
$(steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = FALSE$
grd7 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd8 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure := TRUE$
act3 : $so\_sensor\_failure := TRUE$
act4 : $mode := Emergency\_Stop$

end

Event *Detection_NOK_new_F_WL_F_p_so_both* $\widehat{=}$
extends *Detection_NOK_new_F_WL_F_p_so_both*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $min\_water\_level > water\_level \lor water\_level > max\_water\_level \lor water\_level \notin 0 \,..\, C$
grd4 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = FALSE$
grd7 : $(min\_steam\_output > steam\_output) \lor (steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$

then

act1 : $phase := CONT$
act2 : $wl\_sensor\_failure := TRUE$
act3 : $pump\_failure := TRUE$
act4 : $so\_sensor\_failure := TRUE$
act5 : $mode := Emergency\_Stop$

end

Event *Detection_NOK_det_F_WL_F_p* $\widehat{=}$
extends *Detection_NOK_det_F_WL_F_p*

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd4 : $min\_steam\_output \leq steam\_output \land steam\_output \leq max\_steam\_output \land steam\_output \in 0 \,..\, W$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$
act3 : $mode := Emergency\_Stop$

end

Event $Detection\_NOK\_det\_F\_WL\_F\_so \mathrel{\widehat{=}}$
extends $Detection\_NOK\_det\_F\_WL\_F\_so$

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $(min\_steam\_output > steam\_output) \lor$
$(steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$
grd4 : $(pump\_ctrl = ON \Rightarrow pump > 0)$
grd5 : $(pump\_ctrl = OFF \Rightarrow pump = 0)$
grd6 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd7 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $so\_sensor\_failure := TRUE$
act3 : $mode := Emergency\_Stop$

end

Event $Detection\_NOK\_det\_F\_WL\_F\_p\_so\_both \mathrel{\widehat{=}}$
extends $Detection\_NOK\_det\_F\_WL\_F\_p\_so\_both$

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $(pump\_ctrl = ON \land pump = 0) \lor (pump\_ctrl = OFF \land pump > 0)$
grd4 : $(min\_steam\_output > steam\_output) \lor$
$(steam\_output > max\_steam\_output) \lor steam\_output \notin 0 \,..\, W$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $wl\_sensor\_failure = TRUE$

then

act1 : $phase := CONT$
act2 : $pump\_failure := TRUE$
act3 : $so\_sensor\_failure := TRUE$
act4 : $mode := Emergency\_Stop$

end

Event $Detection\_NOK\_safety\_bounds\_WL \mathrel{\widehat{=}}$
extends $Detection\_NOK\_safety\_bounds\_WL$

when

grd1 : $phase = DET$
grd2 : $stop = FALSE$
grd3 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = TRUE$

then

act1 : $phase := CONT$

act2 : $mode := Emergency\_Stop$

end

Event *PreOperational1* $\widehat{=}$

extends *PreOperational1*

when

grd1 : $phase = CONT$

grd2 : $\neg(wl\_sensor\_failure = TRUE \wedge (pump\_failure = TRUE \vee so\_sensor\_failure = TRUE))$

grd3 : $stop = FALSE$

grd4 : $max\_water\_level > N2 \wedge min\_water\_level > N1$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $preop\_flag = TRUE$

grd7 : $mode = Normal$

then

act1 : $valve\_ctrl := OPEN$

act2 : $phase := PRED$

act3 : $pump\_ctrl : |pump\_ctrl' \in PUMP\_MODE \wedge (pump\_failure = FALSE \Rightarrow pump\_ctrl' = OFF) \wedge (pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl)$

act4 : $mode := Initialisation$

end

Event *PreOperational2* $\widehat{=}$

extends *PreOperational2*

when

grd1 : $phase = CONT$

grd2 : $\neg(wl\_sensor\_failure = TRUE \wedge (pump\_failure = TRUE \vee so\_sensor\_failure = TRUE))$

grd3 : $stop = FALSE$

grd4 : $max\_water\_level \leq N2$

grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$

grd6 : $preop\_flag = TRUE$

grd7 : $mode = Normal$

then

act1 : $pump\_ctrl : |pump\_ctrl' \in PUMP\_MODE \wedge (pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \wedge (pump\_failure = FALSE \wedge min\_water\_level \geq N1 \Rightarrow pump\_ctrl' = OFF) \wedge (pump\_failure = FALSE \wedge min\_water\_level < N1 \Rightarrow pump\_ctrl' = ON)$

act2 : $phase := PRED$

act3 : $valve\_ctrl := CLOSED$

act4 : $preop\_flag := FALSE$
act5 : $mode := Initialisation$

end

Event *Normal_Operational* $\widehat{=}$
refines *Normal_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $valve\_ctrl = CLOSED$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $mode = Normal$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \wedge$
$((min\_water\_level \geq M1 \wedge max\_water\_level < N1) \Rightarrow$
$pump\_ctrl' = ON) \wedge$
$((min\_water\_level > N2 \wedge max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl' = OFF) \wedge$
$((min\_water\_level \geq N1 \wedge max\_water\_level \leq N2) \Rightarrow$
$pump\_ctrl' = pump\_ctrl)$
act3 : $mode := Initialisation$

end

Event *Degraded_Operational* $\widehat{=}$
refines *Degraded_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $valve\_ctrl = CLOSED$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $mode = Degraded$

then

act1 : $phase := PRED$

act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \wedge$
$(pump\_failure = TRUE \Rightarrow pump\_ctrl' = pump\_ctrl) \wedge$
$(pump\_failure = FALSE \wedge min\_water\_level \geq M1 \wedge$
$max\_water\_level < N1 \Rightarrow pump\_ctrl' = ON) \wedge$
$(pump\_failure = FALSE \wedge min\_water\_level > N2 \wedge$
$max\_water\_level \leq M2 \Rightarrow pump\_ctrl' = OFF) \wedge$
$(pump\_failure = FALSE \wedge min\_water\_level \geq N1 \wedge$
$max\_water\_level \leq N2 \Rightarrow pump\_ctrl' = pump\_ctrl)$
act3 : $mode := Initialisation$

end

Event *Rescue_Operational* $\widehat{=}$
refines *Rescue_Operational*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $preop\_flag = FALSE$
grd4 : $valve\_ctrl = CLOSED$
grd5 : $WL\_critical(min\_water\_level \mapsto max\_water\_level) = FALSE$
grd6 : $mode = Rescue$

then

act1 : $phase := PRED$
act2 : $pump\_ctrl :| pump\_ctrl' \in PUMP\_MODE \wedge$
$((min\_water\_level \geq M1 \wedge max\_water\_level < N1) \Rightarrow$
$pump\_ctrl' = ON) \wedge$
$((min\_water\_level > N2 \wedge max\_water\_level \leq M2) \Rightarrow$
$pump\_ctrl' = OFF) \wedge$
$((min\_water\_level \geq N1 \wedge max\_water\_level \leq N2) \Rightarrow$
$pump\_ctrl' = pump\_ctrl)$
act3 : $mode := Initialisation$

end

Event *EmergencyStop* $\widehat{=}$
refines *EmergencyStop*

when

grd1 : $phase = CONT$
grd2 : $stop = FALSE$
grd3 : $mode = Emergency\_Stop$

then

act1 : $stop := TRUE$
act2 : $pump\_ctrl := OFF$
act3 : $valve\_ctrl := CLOSED$

end

Event *Prediction* $\widehat{=}$
extends *Prediction*

when

grd1 : $phase = PRED$
grd2 : $stop = FALSE$
grd3 : $mode = Initialisation$

then

act1 : $phase := ENV$
act2 : $min\_water\_level, max\_water\_level :|$
$min\_water\_level' \in 0 \,..\, C \land max\_water\_level' \in 0 \,..\, C \land$

$((wl\_sensor\_failure = FALSE \land so\_sensor\_failure = FALSE) \Rightarrow$
$(min\_water\_level' = WL\_min(water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output) \land$
$max\_water\_level' = WL\_max(water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output))) \land$

$((wl\_sensor\_failure = TRUE \land so\_sensor\_failure = FALSE) \Rightarrow$
$(min\_water\_level' = WL\_min(min\_water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output) \land$
$max\_water\_level' = WL\_max(max\_water\_level \mapsto steam\_output \mapsto$
$pump \mapsto water\_output))) \land$

$((wl\_sensor\_failure = FALSE \land so\_sensor\_failure = TRUE) \Rightarrow$
$(min\_water\_level' = WL\_min(water\_level \mapsto min\_steam\_output \mapsto$
$pump \mapsto water\_output) \land$
$max\_water\_level' = WL\_max(water\_level \mapsto max\_steam\_output \mapsto$
$pump \mapsto water\_output))) \land$
$min\_water\_level' \leq max\_water\_level'$

act3 : $min\_steam\_output, max\_steam\_output :|$
$min\_steam\_output' \in 0 \,..\, W \land max\_steam\_output' \in 0 \,..\, W \land$

$(so\_sensor\_failure = FALSE \Rightarrow$
$(min\_steam\_output' = SO\_min(steam\_output) \land$
$max\_steam\_output' = SO\_max(steam\_output))) \land$
$(so\_sensor\_failure = TRUE \Rightarrow$
$(min\_steam\_output' = SO\_min(min\_steam\_output) \land$
$max\_steam\_output' = SO\_max(max\_steam\_output))) \land$
$min\_steam\_output' \leq max\_steam\_output'$

end

END

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-2924-4
ISSN 1239-1891