Steam-boiler control specification problem

Jean-Raymond Abrial

August 10, 1994

Abstract

The following specification problem is suggested to the participants of the Dagstuhl Meeting *Methods for Semantics and Specification*, organized jointly with Egon Börger (Pisa) and Hans Langmaack (Kiel) for the week from June 4-9, 1995.

1 Introduction

This text constitutes an informal specification of a program which serves to control the level of water in a steam-boiler. It is important that the program works correctly because the quantity of water present when the steam-boiler is working has to be neither too low nor to high; otherwise the steam-boiler or the turbine sitting in front of it might be seriously affected.

The proposed specification is derived from an original text that has been written by LtCol. J.C. Bauer for the Institute for Risk Research of the University of Waterloo, Ontario, Canada. The original text has been submitted as a competition problem to be solved by the participants of the International Software Safety Symposium organized by the Institute for Risk Research. It has been given to us by the Institut de Protection et de Sureté Nucléaire, Fontenay-aux-Roses, France. We would like to thank the author, the Institute for Risk Research and the Institut de Protection et de Sureté Nucléaire for their kind permission to use their text.

The text to follow is severly biased to a particular implementation. This is very often the case with industrial specifications that are rarely independent from a certain implementation people have in mind. In that sense, this specification is realistic. Your first formalization steps could be *much more abstract* if that seems important to you (in particular if your formalism allows you to do so). In other words, you are encouraged to *structure* your specification in a way that is not necessarily the same as the one proposed in what follows. But in any case, you are asked to demonstrate that your specification can be refined to an implementation that is close enough to the functional requirements of the "specification" proposed below.

You might also judge that the specification contains some loose ends and inconsistencies. Do not hesitate to point them out and to take yourself some appropriate decisions. The idea, however, is that such inconsistencies should be solely within the *organization* of the system and *not within its physical properties*.

We are aware of the fact that the text to follow does not propose any precise model of the physical evolution of the system, only elementary suggestions. As a consequence, you may have to take some simple, even simplistic, abstract decisions concerning such a physical model.

2 Physical environment

The system comprises the following units

- the steam-boiler
- a device to measure the quantity of water in the steam-boiler
- four pumps to provide the steam-boiler with water
- four devices to supervise the pumps (one controller for each pump)
- a device to measure the quantity of steam which comes out of the steam-boiler
- an operator desk
- a message transmission system

2.1 The steam-boiler

The steam-boiler is characterized by the following elements:

- A valve for evacuation of water. It serves only to empty the steam-boiler in its initial phase.
- Its total capacity C (indicated in litres).
- The minimal limit quantity M_1 of water (in litres). Below M_1 the steam-boiler would be in danger after five seconds, if the steam continued to come out at its maximum quantity without supply of water from the pumps.
- The maximal limit quantity M_2 of waters (in litres). Above M_2 the steam-boiler would be in danger after five seconds, if the pumps continued to supply the steam-boiler with water without possibility to evacuate the steam.
- The minimal normal quantity N_1 of water in litres to be maintained in the steam-boiler during regular operation ($M_1 < N_1$).
- The maximal normal quantity N_2 of water (in litres) to be maintained in the steam-boiler during regular operation ($N_2 < M_2$).
- The maximum quantity W of steam (in litres/sec) at the exit of the steam-boiler.
- The maximum gradient U_1 of increase of the quantity of steam (in litres/sec/sec).
- The maximum gradient U_2 of decrease of the quantity of steam (in litres/sec/sec).

2.2 The water level measurement device

The device to measure the level of water in the steam-boiler provides the following information

- the quantity q (in litres) of water in the steam-boiler.

2.3 The pumps

Each pump is characterized by the following elements

- Its capacity P (in litres/sec)
- Its functioning mode: on or off
- it's being started: after having been switched on the pump needs five seconds to start pouring water into the boiler (this is due to the fact that the pump does not balance instantaneously the pressure of the steam-boiler).
- it's being stopped: with instantaneous effect

2.4 The pump control device

Each pump controller provides the following information:

- the water circulates from the pump to the steam-boiler or, in the contrary, it does not circulate.

2.5 The steam measurement device

The device to measure the quantity of steam which comes out of the steam-boiler provides the following information:

- a quantity of steam v (in litres/sec).

2.6 Summary of constants and variables

The following table summerizes the various constants or physical variables of the system:

	Unit	Comment
		Quantity of water in the steam-boiler
C	litre	Maximal capacity
M_1	litre	Minimal limit
M_2	litre	Maximal limit
N_1	litre	Minimal normal
N_2	litre	Maximal normal
		Outcome of steam at the exit of the steam-boiler
W	litre/sec	Maximal quantity
U_1	litre/sec/sec	Maximum gradient of increase
U_2	litre/sec/sec	Maximum gradient of decrease
		Capacity of each pump
P	litre/sec	Nominal capacity
		Current measures
q	litre	Quantity of water in the steam-boiler
p	litre/sec	Throughput of the pumps
v	litre/sec	Quantity of steam exiting the steam-boiler

3 The overall operation of the program

The program communicates with the physical units through messages which are transmitted over a number of dedicated lines connecting each physical unit with the control unit. In first approximation, the time for transmission can be neglected.

The program follows a cycle and a priori does not terminate. This cycle takes place each five seconds and consists of the following actions:

- Reception of messages coming from the physical units.
- Analysis of informations which have been received.
- Transmission of messages to the physical units.

To simplify matters, and in first approximation, all messages coming from (or going to) the physical units are supposed to be received (emitted) *simultaneously* by the program at each cycle.

4 Operation modes of the program

The program operates in different modes, namely: *initialization, normal, degraded, rescue, emergency stop.*

4.1 *Initialization* mode

The *initialization* mode is the mode to start with. The program enters a state in which it waits for the message STEAM-BOILER_WAITING to come from the physical units. As soon as this message has been received the program checks whether the quantity of steam coming out of the steam-boiler is really zero. If the unit for detection of the level of steam is defective—that is, when v is not equal to zero—the program enters the *emergency stop* mode. If the quantity of water in the steam-boiler is above N_2 the program activates the valve of the steam-boiler in order to empty it. If the quantity of water in the steam-boiler is below N_1 then the program activates a pump to fill the steam-boiler. If the program realizes a failure of the water level detection unit it enters the *emergency stop* mode. As soon as a level of water between N_1 and N_2 has been reached the program can send continuously the signal PROGRAM_READY to the physical units until it receives the signal PHYSICAL_UNITS_READY which must necessarily be emitted by the physical units. As soon as this signal has been received, the program enters either the mode *normal* if all the physical units operate correctly or the mode *degraded* if any

physical unit is defective. A transmission failure puts the program into the mode *emergency stop*.

4.2 *Normal* mode

The normal mode is the standard operating mode in which the program tries to maintain the water level in the steam-boiler between N_1 and N_2 with all physical units operating correctly. As soon as the water level is below N_1 or above N_2 the level can be adjusted by the program by switching the pumps on or off. The corresponding decision is taken on the basis of the information which has been received from the physical units. As soon as the program recognizes a failure of the water level measuring unit it goes into *rescue* mode. Failure of any other physical unit puts the program into *degraded* mode. If the water level is risking to reach one of the limit values M_1 or M_2 the program enters the mode *emergency stop*. This risk is evaluated on the basis of a maximal behaviour of the physical units. A transmission failure puts the program into *emergency stop* mode.

4.3 *Degraded* mode

The *degraded* mode is the mode in which the program tries to maintain a satisfactory water level despite of the presence of failure of some physical unit. It is assumed however that the water level measuring unit in the steam-boiler is working correctly. The functionality is the same as in the preceding case. Once all the units which were defective have been repaired, the program comes back to *normal* mode. As soon as the program sees that the water level measuring unit has a failure, the program goes into mode *rescue*. If the water level is risking to reach one of the limit values M_1 or M_2 the program enters the mode *emergency stop*. A transmission failure puts the program into *emergency stop* mode.

4.4 *Rescue* mode

The *rescue* mode is the mode in which the program tries to maintain a satisfactory water level despite of the failure of the water level measuring unit. The water level is then estimated by a computation which is done taking into account the maximum dynamics of the quantity of steam coming out of the steam-boiler. For the sake of simplicity, this calculation can suppose that exactly n liters of water, supplied by the pumps, do account for exactly the same amount of boiler contents (no thermal expansion). This calculation can however be done only if the unit which measures the quantity of steam is itself working and if one can rely upon the information which comes from the units for controlling the pumps. As soon as the water measuring unit is repaired, the program returns into mode *degraded* or into mode *normal*. The program goes into *emergency stop* mode if it realizes that one of the following cases holds: the unit which measures the outcome of steam has a failure, or the units which control the pumps have a failure, or the water level risks to reach one of the two limit values. A transmission failure puts the program into *emergency stop* mode.

4.5 *Emergency stop* mode

The *emergency stop* mode is the mode into which the program has to go, as we have seen already, when either the vital units have a failure or when the water level risks to reach one of its two limit values. This mode can also be reached after detection of an erroneous transmission between the program and the physical units. This mode can also be set directly from outside. Once the program has reached the *Emergency stop* mode, the physical environmente is then responsible to take approrpiate actions, and the program stops.

5 Messages sent by the program

The following messages can be sent by the program:

- MODE(m): The program sends, at each cycle, its current mode of operation to the physical units.

- PROGRAM_READY: In *initialization* mode, as soon as the program assumes to be ready, this message is continuously sent until the message PHYSICAL_UNITS_READY coming from the physical units has been received.
- VALVE: In *initialization* mode this message is sent to the physical units to request opening and then closure of the valve for evacuation of water from the steam-boiler.
- OPEN_PUMP(n): This message is sent to the physical units to activate a pump.
- CLOSE_PUMP(n): This message is sent to the physical units to stop a pump.
- PUMP_FAILURE_DETECTION(n): This message is sent (until receipt of the corresponding acknowledgement) to indicate to the physical units that the program has detected a pump failure.
- PUMP_CONTROL_FAILURE_DETECTION(n): This message is sent (until receipt of the corresponding acknowledgement) to indicate to the physical units that the program has detected a failure of the physical unit which controls a pump.
- LEVEL_FAILURE_DETECTION: This message is sent (until receipt of the corresponding acknowledgement) to indicate to the physical units that the program has detected a failure of the water level measuring unit.
- STEAM_FAILURE_DETECTION: This message is sent (until receipt of the corresponding acknowledgement) to indicate to the physical units that the program has detected a failure of the physical unit which measures the outcome of steam.
- PUMP_REPAIRED_ACKNOWLEDGEMENT(n): This message is sent by the program to acknowledge a message coming from the physical units and indicating that the corresponding pump has been repaired.
- PUMP_CONTROL_REPAIRED_ACKNOWLEDGEMENT(n): This message is sent by the program to acknowledge a message coming from the physical units and indicating that the corresponding physical control unit has been repaired.
- LEVEL_REPAIRED_ACKNOWLEDGEMENT: This message is sent by the program to acknowledge a message coming from the physical units and indicating that the water level measuring unit has been repaired.
- STEAM_REPAIRED_ACKNOWLEDGEMENT: This message is sent by the program to acknowledge a message coming from the physical units and indicating that the unit which measures the outcome of steam has been repaired.

6 Messages received by the program

The following messages can be received by the program:

- STOP: When the message has been received three times in a row by the program, the program must go into *emergency stop*.
- STEAM_BOILER_WAITING: When this message is received in *initialization* mode it triggers the effective start of the program.
- PHYSICAL_UNITS_READY: This message when received in *initialization* mode acknowledges the message PROGRAM_READY which has been sent previously by the program.
- PUMP_STATE(n, b): This message indicates the state of pump n (open or closed). This message must be present during each transmission.
- PUMP_CONTROL_STATE(n, b): This message gives the information which comes from the control unit of pump n (there is flow of water or there is no flow of water). This message must be present during each transmission.
- LEVEL(v): This message contains the information which comes from the water level measuring unit. This message must be present during each transmission.
- STEAM(v): This message contains the information which comes from the unit which measures the outcome of steam. This message must be present during each transmission.
- PUMP_REPAIRED(n): This message indicates that the corresponding pump has been repaired. It is sent by the physical units until a corresponding acknowledgement message has been sent by the program and received by the physical units.

- PUMP_CONTROL_REPAIRED(n): This message indicates that the corresponding control unit has been repaired. It is sent by the physical units until a corresponding acknowledgement message has been sent by the program and received by the physical units.
- LEVEL_REPAIRED: This message indicates that the water level measuring unit has been repaired. It is sent by the physical units until a corresponding acknowledgement message has been sent by the program and received by the physical units.
- STEAM_REPAIRED: This message indicates that the unit which measures the outcome of steam has been repaired. It is sent by the physical units until a corresponding acknowledgement message has been sent by the program and received by the physical units.
- PUMP_FAILURE_ACKNOWLEDGEMENT(n): By this message the physical units acknowledge the receipt of the corresponding failure detection message which has been emitted previously by the program.
- PUMP_CONTROL_FAILURE_ACKNOWLEDGEMENT(n): By this message the physical units acknowledge the receipt of the corresponding failure detection message which has been emitted previously by the program.
- LEVEL_FAILURE_ACKNOWLEDGEMENT: By this message the physical units acknowledge the receipt of the corresponding failure detection message which has been emitted previously by the program.
- STEAM_OUTCOME_FAILURE_ACKNOWLEDGEMENT: By this message the physical units acknowledge the receipt of the corresponding failure detection message which has been emitted previously by the program.

7 Detection of equipment failures

The following erroneous kinds of behaviour are distinguished to decide whether certain physical units have a failure:

- PUMP: (1) Assume that the program has sent a start or stop message to a pump. The program detects that during the following transmission that pump does not indicate its having effectively been started or stopped. (2) The program detects that the pump changes its state spontaneously.
- PUMP_CONTROLLER: (1) Assume that the program has sent a start or stop message to a pump. The program detects that during the second transmission after the start or stop message the pump does not indicate that the water is flowing or is not flowing; this despite of the fact that the program knows from elsewhere that the pump is working correctly. (2) The program detects that the unit changes its state spontaneously.
- WATER_LEVEL_MEASURING_UNIT: (1) The program detects that the unit indicates a value which is out of the valid static limits–i.e. between 0 and C. (2) The program detects that the unit indicates a value which is incompatible with the dynamics of the system.
- STEAM_LEVEL_MEASURING_UNIT: (1) The program detects that the unit indicates a value which is out of the valid static limits—i.e. between 0 and W. (2) The program detects that the unit indicates a value which is incompatible with the dynamics of the system.
- TRANSMISSION: (1) The program receives a message whose presence is aberrant. (2) The program does not receive a message whose presence is indispensable.