Steam-Boiler Controller System (SBCS) Specification Using the SCR Approach

A. John van Schouwen

An SCR specification of the Steam-boiler control specification problem by Jean-Raymond Abrial

May 22, 1996

1.0 SBCS Computer

The computer consists of the CPU and its I/O hardware and communications links. ??

1.1 Data Transfer Definitions

All input and output data items are transceived over a single transmission link between the computer and the physical (peripheral) units ??. Communication is via a sequence of input messages on input and a sequence of output messages on output.

The named data transfer mechanisms are defined as follows:

1) IN(x), where x is a sequence of input messages.
2) OUT(x), where x is a sequence of output messages.

{* This section needs revisiting. ?? *}

2.0 Environmental State Variables

This section describes the environmental state variables relevant to the Steam-Boiler Controller System (SBCS).

2.1 Application State Variables

Figure 1 and Figure 2 show the context of the system components and their associated environmental state variables with respect to the SBCS.

2.1.1 Monitored State Variables

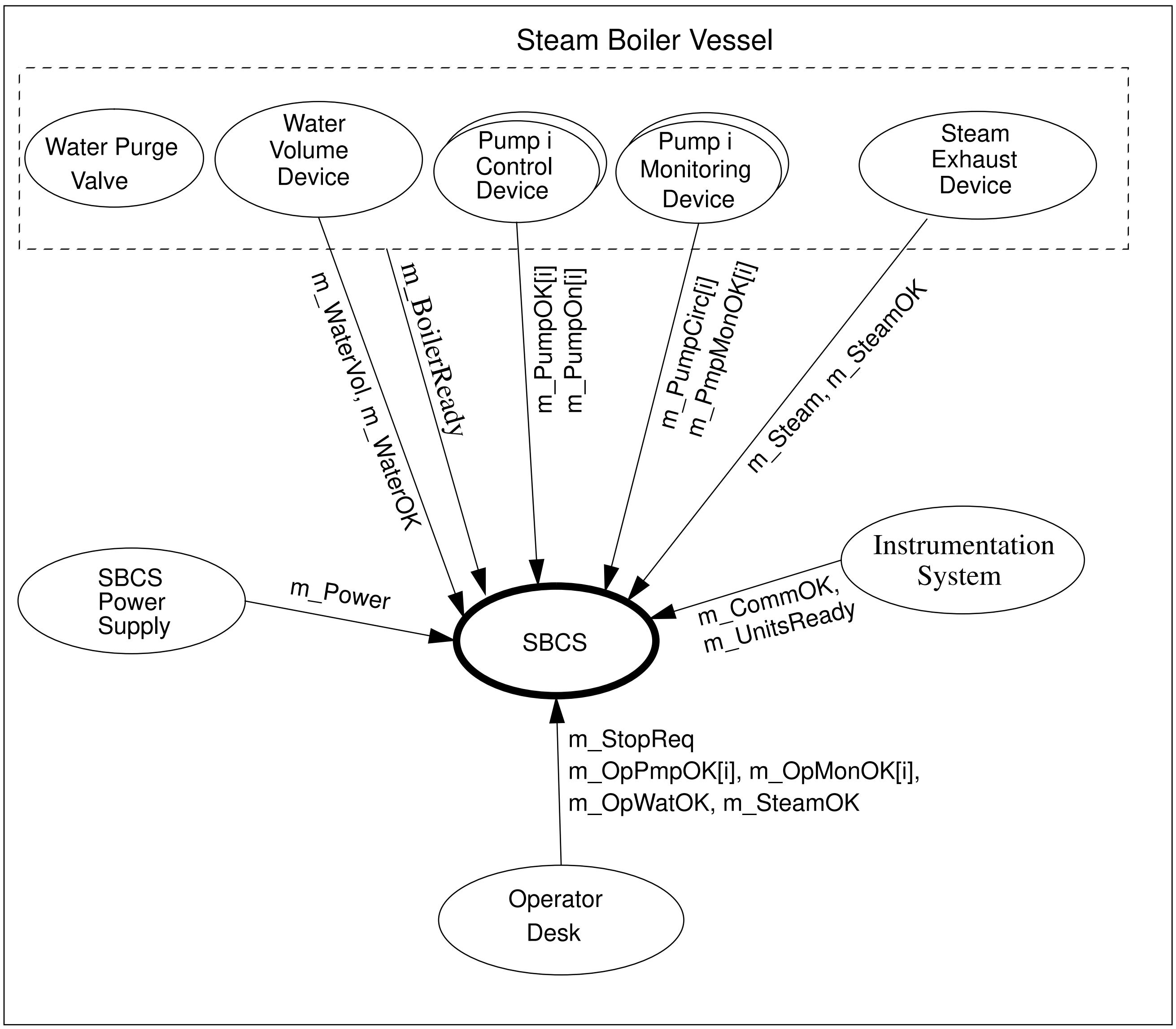

FIGURE 1. The monitored application state variables of the boiler system components.

m_BoilerReady :Boolean:
= true iff the steam boiler vessel is ready to operate.

m_CommOK :Boolean:
= true iff the communications link between the control unit and the instrumentation system is functioning correctly.

m_OpMonStatus[i]
m_OpPmpStatus[i]
m_OpSteamStatus
m_OpWatStatus

in {k_failed, k_repaired, k_none}
Indicates that the operator desk wants to notify the control unit that the associated device (see Table 2) is acknowledged as having failed (k_failed), or has been repaired (k_repaired); k_none indicates that no further notification is needed.

Table 1:

Controlled State Variable	Device
m_OpMonStatus[i]	The monitoring device for the i-th pump.
m_OpPmpStatus[i]	The i-th pump.
m_OpSteamStatus	The steam exhaust flow device.
m_OpWatStatus	The water volume measuring device.

m_PmpMonOK[i] :Boolean:; i in {1, ..., d_NumPumps}
m_PmpMonOK[i] = true iff the flow monitoring device of the i-th pump is functioning correctly.

m_Power in {k_on, k_off}
= k_on iff power is applied to the control unit.

m_PumpCirc[i] :Boolean:; i in {1, ..., d_NumPumps}
m_PumpCirc[i] = true iff water is flowing from the i-th pump to the boiler vessel.

m_PumpOK[i] :Boolean:; i in {1, ..., d_NumPumps}
m_PumpOK[i] = true iff the i-th pump is functioning correctly.

m_PumpOn[i] :Boolean:; i in {1, ..., d_NumPumps}
m_PumpOn[i] = true iff the i-th pump is running.

m_Steam :Flow:
= rate of steam flow, in litres/second (l/s), at the steam exit of the boiler vessel. (0 <= m_Steam <= d_W)

m_SteamOK :Boolean:
= true iff the steam exhaust flow measuring device is functioning correctly.

m_StopReq :Boolean:
= true iff the operator desires the pumpts to be shut off.

m_UnitsReady :Boolean:
= true iff the instrumentation system wants to acknowledge that it is aware of the control system being ready to begin normal control of the boiler.

m_WaterOK :Boolean:
= true iff the water volume measuring unit is functioning correctly.

m_WaterVol :Volume:
= volume of water in the boiler vessel, in **litres** (*l*). (0 <= m_WaterVol <= d_C)

2.1.2 Controlled State Variables

c_CtrlUnitRdy in {k_ready, k_none}
= k_ready, when the control unit wants to notify the instrumentation system that it is ready to begin normal control of the boiler.

c_Mode :MC_Operating:
= the current indication of the control unit's operating mode

c_OpMonStatus[i]
c_OpPmpStatus[i]
c_OpSteamStatus
c_OpWatStatus

in {k_failed, k_repaired, k_none}; i in {1, ..., d_NumPumps}
Indicates that the control unit wants to notify the operator desk that the associated device (see Table 2) has failed (k_failed), or is acknowledged as having been repaired (k_repaired); k_none indicates no further notification needed.

Table 2:

Controlled State Variable	Device
c_OpMonStatus[i]	The monitoring device for the i-th pump.
c_OpPmpStatus[i]	The i-th pump.
c_OpSteamStatus	The steam exhaust flow device.
c_OpWatStatus	The water volume measuring device.

c_PumpOn[i] :Boolean:; i in {1, ..., d_NumPumps}
c_PumpOn[i] = true iff the i-th pump is supposed to be running.

c_Valve in {k_open, k_closed}
= k_open iff the water evacuation valve is open.

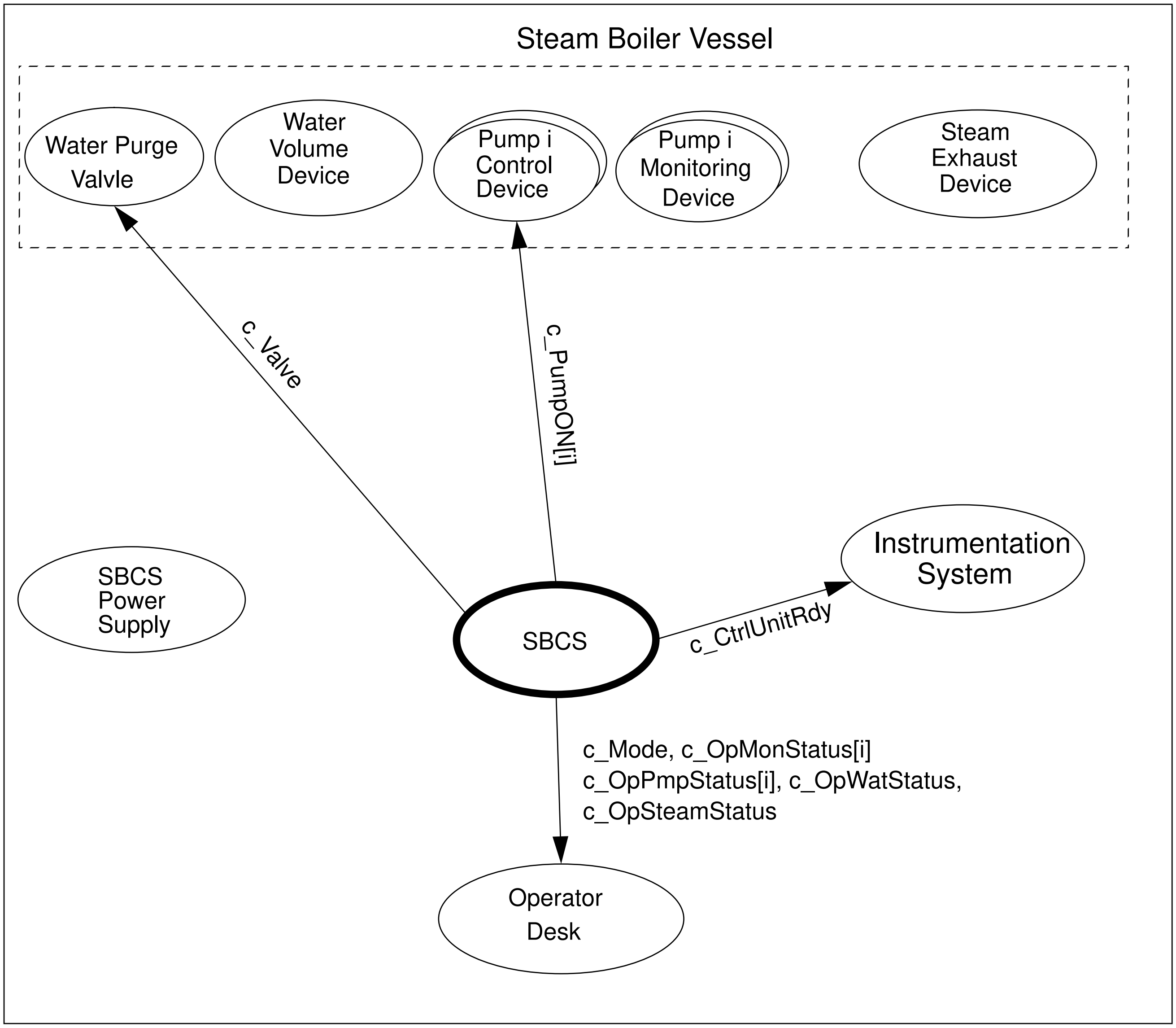

FIGURE 2. The controlled application state variables of the boiler system components.

2.1.3 Environmental Restrictions

m_PumpOn[i] = c_PumpOn[i]

d_U2 <= d m_Steam / dt <= d_U1

c_Valve = k_closed, @T(m_WaterVol = 0)

m_BoilerReady => (c_Valve = k_closed) {design assumption}

2.2 Established State Variables

None.

2.3 Established State Variables

None.

3.0 Input Data Items

3.1 Input Data Item Classes

??

3.2 Input Data Item Definitions

3.2.1 Invalid Instrumentation System Message

Acronym: i_anonymous

Hardware: None

Characteristics of Values: (i_anonymous <> k_Nil) => ~m_CommOK

Value Encoding: k_Nil (nil)

Data Transfer: IN(x), where i_anonymous is a leftover bitstring in x that could not be parsed as part of a valid message.

3.2.2 Physical Units Ready

Acronym: i_PhysURdy

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_PhysURdy = k_Ready) & ('c_CtrlUnitRdy = k_ready)
=> m_UnitsReady

(i_PhysURdy = k_Ready) & ('c_CtrlUnitRdy <> k_ready)
=> ~m_CommOK

Value Encoding:

k_Ready	(??)
k_Nil	(nil)

Data Transfer: IN(x), where i_PhysURdy in x

Data Representation: ??

Message Mnemonic: PHYSICAL_UNITS_READY

3.2.3 Pump Failure Acknowledgement

Acronym: i_PmpFailACK[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_PmpFailACK[x] = k_ACK) & ('c_OpPmpStatus[x] = k_failed) =>
(m_OpPmpStatus[x] = k_failed)

ForSome x ((i_PmpFailACK[x] = k_ACK) & ('c_OpPmpStatus[x] <> k_failed)) =>
~m_CommOK

Value Encoding:	k_ACK	(??)
	k_Absent	(nil)

Data Transfer: IN(y), where i_PmpFailACK[x] in y

Data Representation: ?? (??)

Message Mnemonic: PUMP_FAILURE_ACKNOWLEDGEMENT(x)

3.2.4 Pump Flow Indicator

Acronym: i_PmpFlow[x], where x in {1, ..., d_NumPumps}

Hardware: Pump monitors 1 thru d_NumPumps

Characteristics of Values:

Ideal Values:

Table 3:

i_PmpFlow[x] =	Conditions	=> m_PumpCirc[x] =	m_PmpMonOK[x] =	m_CommOK =
k_flowing	m_PumpOK[x] & ~m_PumpOn[x]	(undefined)	false	X
	m_PumpOK[x] & m_PumpOn[x]	true	true	X
	~m_PumpOK[x]	true	true	X
k_idle	m_PumpOK[x] & ~m_PumpOn[x]	false	true	X
	m_PumpOK[x] & (Drtn(m_PumpOn[x] >= 5s)	(undefined)	false	X
	m_PumpOK[x] & (Drtn(m_PumpOn[x]) < 5s) & 'm_PumpCirc[x]	(undefined)	false	X
	m_PumpOK[x] & (Drtn(m_PumpOn[x]) < 5s) & ~'m_PumpCirc[x] & (d m_WaterVol / dt > 0)	(undefined)	false	X
	m_PumpOK[x] & (Drtn(m_PumpOn[x]) < 5s) & ~'m_PumpCirc[x] & (d m_WaterVol / dt <= 0)	false	true	X
	~m_PumpOK[x] & (d m_WaterVol / dt > 0)	(undefined)	false	X
	~m_PumpOK[x] & (d m_WaterVol / dt <= 0)	false	true	X
k_error	X	'm_PumpCirc[x]	'm_PmpMonOK[x]	true

Value Encoding:

k_flowing	(??)
k_idle	(??)
k_error	(nil)

Data Transfer: IN(y), where i_PmpFlow[x] in y

Data Representation: ??

Message Mnemonic: PUMP_CONTROL_STATE(x, i_PmpFlow[x])

3.2.5 Pump Monitor Device Repaired

Acronym: i_PmpMonRep[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_PmpMonRep[x] = k_Repaired) & ('c_OpMonStatus[x] = k_none) =>
(m_OpMonStatus[x] = k_Repaired)

ForSome x ((i_PmpMonRep[x] = k_Repaired) & ('c_OpMonStatus[x] <> k_none)) =>
~m_CommOK

Value Encoding:

k_Repaired	(??)
k_Absent	(nil)

Data Transfer: IN(y), where i_PmpMonRep[x] in y

Data Representation: ??

Message Mnemonic: PUMP_CONTROL_REPAIRED(x)

3.2.6 Pump Monitor Failure Acknowlegement

Acronym: i_MonFailACK[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_MonFailACK[x] = k_ACK) & ('c_OpMonStatus[x] = k_failed) =>
(m_OpMonStatus[x] = k_failed)

ForSome x ((i_MonFailACK[x] = k_ACK) & ('c_OpMonStatus[x] <> k_failed)) =>
~m_CommOK

Value Encoding:

k_ACK	(??)
k_Absent	(nil)

Data Transfer: IN(y), where i_MonFailACK[x] in y

Data Representation: ?? (??)

Message Mnemonic: PUMP_CONTROL_FAILURE_ACKNOWLEDGEMENT(x)

3.2.7 Pump Repaired

Acronym: i_PmpRepair[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_PmpRepair[x] = k_Repaired) & ('c_OpPmpStatus[x] = k_none) =>
(m_OpPmpStatus[x] = k_Repaired)

ForSome x ((i_PmpRepair[x] = k_Repaired) & ('c_OpPmpStatus[x] <> k_none)) =>
~m_CommOK

Value Encoding: k_Repaired (??)
k_Absent (nil)

Data Transfer: IN(y), where i_PmpRepair[x] in y

Data Representation: ??

Message Mnemonic: PUMP_REPAIRED(x)

3.2.8 Pump State

Acronym: i_Pump[x], where x in {1, ..., d_NumPumps}

Hardware: Pumps 1 thru d_NumPumps

Characteristics of Values:

Ideal Values:

Table 4:

i_Pump[x] =	'c_PumpOn[x] = =>	m_Pump[x] =	m_PumpOK[x] =	m_CommOK =
k_on	true	true	true	X[†]
	false	(undefined)	false	X
k_off	true	(undefined)	false	X
	false	false	true	X
k_error[‡]	X[††]	'm_Pump[x]	'm_PumpOK[x]	false

† Means "cannot draw an implication in this case".
‡ ?? Use the statement below instead of this row?
†† Means "don't care".

{* ForSome x (i_Pump[x] = k_error) => ~m_CommOK *}

Value Encoding:

k_on	(??)
k_off	(??)
k_error	(nil)

Data Transfer: IN(y), where i_Pump[x] in y

Data Representation: ??

Message Mnemonic: PUMP_STATE(x, i_Pump[x])

3.2.9 Steam Boiler Ready Indicator

Acronym: i_StmBlWait

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_StmBlWait = k_Ready) & Inmode(k_Initialisation) => m_BoilerReady

(i_StmBlWait = k_Ready) & ~Inmode(k_Initialisation) => ~m_CommOK

Value Encoding: k_Ready (??)
k_NotReady (nil)

Data Transfer: IN(x), where i_StmBlWait in x

Data Representation: ??

Message Mnemonic: STEAM_BOILER_WAITING

3.2.10 Steam Flow Indicator

Acronym: i_Flow

Hardware: Steam Flow Measurement Device

Characteristics of Values:

Ideal Values:

f(i_Flow) = m_Steam +/- k

(f(i_Flow) ~in [0, d_W]) OR (d f(i_Flow) / dt ~in [d_U2, d_U1])
=> ~m_SteamOK

(i_Flow = nil) => ~m_CommOK

Tolerance: k = ??

Data Transfer: IN(x), where i_Flow in x

Data Representation: ?? (??)

Message Mnemonic: STEAM(i_Flow)

3.2.11 Steam Flow Sensor Failure Acknowledgement

Acronym: i_SteamFailACK

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_SteamFailACK = k_ACK) & ('c_OpSteamStatus = k_failed) =>
(m_OpSteamStatus = k_failed)

(i_SteamFailACK = k_ACK) & ('c_OpSteamStatus <> k_failed)) =>
~m_CommOK

Value Encoding: k_ACK (??)
k_Absent (nil)

Data Transfer: IN(x), where i_SteamFailACK in x

Data Representation: ?? (??)

Message Mnemonic: STEAM_OUTCOME_FAILURE_ACKNOWLEDGEMENT

3.2.12 Steam Flow Sensor Repaired

Acronym: i_SteamRepair

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_SteamRepair = k_Repaired) & ('c_OpSteamStatus = k_none) =>
(m_OpSteamStatus = k_Repaired)

(i_SteamRepair = k_Repaired) & ('c_OpSteamStatus <> k_none)) =>
~m_CommOK

Value Encoding: k_Repaired (??)
k_Absent (nil)

Data Transfer: IN(x), where i_SteamRepair in x

Data Representation: ??

Message Mnemonic: STEAM_REPAIRED

3.2.13 Stop Program

Acronym: i_Stop

Hardware: Operator Desk

Characteristics of Values:

Ideal Values: (Drtn(i_Stop = k_present) >= 15s + k) => m_StopReq

Tolerance: k = +/- ??

Value Encoding: k_present (??)
k_absent (nil)

Data Transfer: IN(x), where i_Stop in x

Data Representation: ??

Message Mnemonic: STOP

3.2.14 Water Volume

Acronym: i_Vol

Hardware: Water Volume Measuring Device

Characteristics of Values:

Ideal Values:

f(i_Vol) = m_WaterVol +/- k

(f(i_Vol) ~in [0, d_C]) OR (d f(i_Vol) / dt > (d_NumPumps * d_P))
=> ~m_WaterOK

(i_Flow = nil) => ~m_CommOK

Tolerance: k = ??

Data Transfer: IN(x), where i_Vol in x

Data Representation: ??

Message Mnemonic: LEVEL(i_Vol)

3.2.15 Water Volume Sensor Failure Acknowlegement

Acronym: i_VolFailACK

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_VolFailACK = k_ACK) & ('c_OpWatStatus = k_failed) =>
(m_OpWatStatus = k_failed)

(i_VolFailACK = k_ACK) & ('c_OpWatStatus <> k_failed)) =>
~m_CommOK

Value Encoding:		
	k_ACK	(??)
	k_Absent	(nil)

Data Transfer: IN(x), where i_VolFailACK in x

Data Representation: ?? (??)

Message Mnemonic: LEVEL_FAILURE_ACKNOWLEDGEMENT

3.2.16 Water Volume Sensor Repaired

Acronym: i_VolRepair

Hardware: Operator Desk

Characteristics of Values:

Ideal Values:

(i_VolRepair = k_Repaired) & ('c_OpWatStatus = k_none) =>
(m_OpWatStatus = k_Repaired)

(i_VolRepair = k_Repaired) & ('c_OpWatStatus <> k_none)) =>
~m_CommOK

Value Encoding:		
	k_Repaired	(??)
	k_Absent	(nil)

Data Transfer: IN(x), where i_VolRepair in x

Data Representation: ??

Message Mnemonic: LEVEL_REPAIRED

4.0 Output Data Items

4.1 Output Data Item Classes

??

4.2 Output Data Item Definitions

4.2.1 Control Unit Ready

Acronym: o_CtrlRdy

Hardware: Operator Desk

Characteristics of Values:

Table 5: Selector Table

o_CtrlRdy =	c_CtrlUnitRdy =
k_Ready	k_ready
k_None	k_none

Value Encoding: k_Ready (??)
k_None (nil)

Data Transfer: OUT(x), where o_CtrlRdy in x

Data Representation: ??

Message Mnemonic: PROGRAM_READY

4.2.2 Evacuation Value Control

Acronym: o_Valve

Hardware: Water Evacuation Valve

Characteristics of Values: c_Valve = k_open, @T(o_Valve = k_open)

Value Encoding: k_open (??)
k_absent (nil)

Data Transfer: OUT(x), where o_Valve in x

Data Representation: ??

Message Mnemonic: VALVE

4.2.3 Mode Indication

Acronym: o_Mode

Hardware: Operator Desk

Characteristics of Values: c_Mode = o_Mode

Value Encoding:

k_Initialisation	(??)
k_Normal	(??)
k_Degraded	(??)
k_Rescue	(??)
k_EmergStop	(??)

Data Transfer: OUT(x), where o_Mode in x

Data Representation: ??

Message Mnemonic: MODE(o_Mode)

4.2.4 Pump Control

Acronym: o_PmpCtrl[x], where x in {1, ..., d_NumPumps}

Hardware: Pumps 1 thru d_NumPumps

Characteristics of Values:

Table 6: Event Table for c_PumpOn[x]

Condition	Triggering Event	
true	@T(o_PmpCtrl[x] = k_open)[1]	@T(o_PmpCtrl[x] = k_close)[1]
c_PumpOn[x] =	true	false

Value Encoding:

k_open	(??)
k_close	(??)
k_Absent	(nil)

Data Transfer: OUT(y), where o_PmpCtrl[x] in y

Data Representation: ??

Message Mnemonic: OPEN_PUMP(x), CLOSE_PUMP(x)

Notes: 1) Output this value only once for each event to be signalled.

4.2.5 Pump Monitor Reporting Output

Acronym: o_MonStat[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Table 7: Selector Table

o_MonStat[x] =	c_OpMonStatus[x] =
k_failed	k_failed
k_repaired	k_repaired
k_absent	k_none

Value Encoding:

k_failed	(??)
k_repaired	(??)
k_absent	(nil)

Data Transfer: OUT(y), where o_MonStat[x] in y

Data Representation: ??

Message Mnemonic: PUMP_CONTROL_FAILURE_DETECTION(x), PUMP_CONTROL_REPAIRED_ACKNOWLEDGEMENT(x)

4.2.6 Pump Reporting Output

Acronym: o_PmpStat[x], where x in {1, ..., d_NumPumps}

Hardware: Operator Desk

Characteristics of Values:

Table 8: Selector Table

o_PmpStat[x] =	c_OpPmpStatus[x] =
k_failed	k_failed
k_repaired	k_repaired
k_absent	k_none

Value Encoding:

k_failed	(??)
k_repaired	(??)
k_absent	(nil)

Data Transfer: OUT(y), where o_PmpStat[x] in y

Data Representation: ??

Message Mnemonic: PUMP_FAILURE_DETECTION(x), PUMP_REPAIRED_ACKNOWLEDGEMENT(x)

4.2.7 Steam Flow Device Reporting Output

Acronym: o_SteamStat

Hardware: Operator Desk

Characteristics of Values:

Table 9: Selector Table

o_SteamStat =	c_OpMonStatus =
k_failed	k_failed
k_repaired	k_repaired
k_absent	k_none

Value Encoding:
k_failed (??)
k_repaired (??)
k_absent (nil)

Data Transfer: OUT(x), where o_SteamStat in x

Data Representation: ??

Message Mnemonic: STEAM_FAILURE_DETECTION, STEAM_REPAIRED_ACKNOWLEDGEMENT

4.2.8 Water Volume Device Reporting Output

Acronym: o_VolStat

Hardware: Operator Desk

Characteristics of Values:

Table 10: Selector Table

o_VolStat =	c_OpWatStatus =
k_failed	k_failed
k_repaired	k_repaired
k_absent	k_none

Value Encoding:
k_failed (??)
k_repaired (??)
k_absent (nil)

Data Transfer: OUT(x), where o_VolStat in x

Data Representation: ??

Message Mnemonic: LEVEL_FAILURE_DETECTION, LEVEL_REPAIRED_ACKNOWLEDGEMENT

5.0 Modes and Transitions

When Initialised: d_SystemInit

5.1 Modes

5.1.1 Operating Modes

Class: MC_Operating

Modes in Class: k_Initialisation
k_Normal
k_Degraded
k_Rescue
k_EmergStop

Initial Mode: k_Initialisation

5.2 Mode Transitions

5.2.1 Operating Mode Transitions

Table 11: Event Table

Mode	Triggering Event				
k_Initialisation	X	@T(m_UnitsReady) WHEN (d_PumpsOK & d_MonsOK & m_SteamOK & m_WaterOK & m_CommOK & l_NoStop & d_NormWaterRange)	@T(m_UnitsReady) WHEN (~(d_PumpsOK & d_MonsOK) & (m_SteamOK & m_WaterOK & m_CommOK & l_NoStop & d_NormWaterRange)	X	@F(m_SteamOK & m_WaterOK & m_CommOK) OR @F(l_NoStop)
k_Normal	X	X	@T(~(d_PumpsOK & d_MonsOK & m_SteamOK) & (~d_ExLimits & m_WaterOK & m_CommOK & l_NoStop)) WHEN (m_WaterOK & m_CommOK & ~d_ExLimits & l_NoStop)	@T(~m_WaterOK & (~d_ExLimits & d_MonsOK & m_SteamOK & m_CommOK & l_NoStop)) WHEN (d_MonsOK & m_SteamOK & m_CommOK & ~d_ExLimits & l_NoStop)	@T(d_ExLimits) OR @F(m_CommOK) OR @T(~m_WaterOK & (~d_MonsOK & ~m_SteamOK)) OR @F(l_NoStop)
k_Degraded	X	@T(d_PumpsOK & d_MonsOK & m_SteamOK & m_WaterOK & (m_CommOK & ~d_ExLimits & l_NoStop) WHEN (~d_ExLimits & m_CommOK & l_NoStop)	X	@F(m_WaterOK) WHEN (d_MonsOK & m_SteamOK & m_CommOK & ~d_ExLimits & l_NoStop)	@T(~d_MonsOK OR ~m_SteamOK) OR @F(m_CommOK) OR @T(d_ExLimits) OR @F(l_NoStop)
k_Rescue	X	@T(m_WaterOK) WHEN (d_PumpsOK & d_MonsOK & m_SteamOK & m_CommOK & ~d_ExLimits & l_NoStop)	@T(m_WaterOK) WHEN (~(d_PumpsOK & d_MonsOK & m_SteamOK) & m_CommOK & ~d_ExLimits & l_NoStop)	X	@F(m_SteamOK) OR @F(d_MonsOK) OR @F(m_CommOK) OR @T(d_ExLimits) OR @F(l_NoStop)
k_EmergStop	X	X	X	X	X
MC_Operating =	k_Initialisation	k_Normal	k_Degraded	k_Rescue	k_EmergStop

l_NoStop = (Drtn(m_StopReq) < 5s)

6.0 Auxiliary State Variable Behaviour

6.1 (None)

7.0 Controlled Application State Variable Behaviour

7.1 c_CtrlUnitRdy

Relevant Mode Classes: MC_Operating

Initial Value: k_none, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Ideal Value:

Table 12: Event Table

Mode	Triggering Event	
k_initialisation	@T(Drtn(|d m_WaterVol / dt| <= k1) = 5s) WHEN (d_NormWaterRange & ~m_UnitsReady)	@T(m_UnitsReady)
k_Normal k_Degraded k_Rescue k_EmergStop	X	X
c_CtrlUnitRdy =	k_ready	k_none

Tolerance: k1 = ??
??

7.2 c_Mode

Relevant Mode Classes: MC_Operating

Sustaining Condition: m_Power = k_on

Ideal Value: c_Mode = MC_Operating ??†

Tolerance: ??

† This would have to be the case unless something else in the environment is monitoring the boiler and takes appropriate action when things go wrong; otherwise there would be no way for the external environment to know that action needs to be taken.

7.3 c_OpMonStatus

Relevant Mode Classes: MC_Operating

Initial Value: k_none, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Ideal Value: ForAll i

Table 13: Event Table

Mode	Triggering Event		
k_initialisation k_Normal k_Degraded k_Rescue	@F(m_PmpMonOK[i]) OR @T(m_OpMonStatus[i] = k_none) WHEN ((m_OpMonStatus[i] = k_repaired) & ~m_PmpMonOK[i])	@T(m_OpMonStatus[i] = k_repaired)	@T(m_OpMonStatus[i] = k_failed) OR @T(m_OpMonStatus[i] = k_none) WHEN ((m_OpMonStatus[i] = k_repaired) & m_PmpMonOK[i])
k_EmergStop	X	X	X
c_OpMonStatus[i] =	k_failed	k_repaired	k_none

Tolerance: ??

7.4 c_OpPmpStatus

Relevant Mode Classes: MC_Operating

Initial Value: k_none, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Ideal Value: ForAll i

Table 14: Event Table

Mode	Triggering Event		
k_initialisation k_Normal k_Degraded k_Rescue	@F(m_PumpOK[i]) OR @T(m_OpPmpStatus[i] = k_none) WHEN ((m_OpPmpStatus[i] = k_repaired) & ~m_PumpOK[i])	@T(m_OpPmpStatus[i] = k_repaired)	@T(m_OpPmpStatus[i] = k_failed) OR @T(m_OpPmpStatus[i] = k_none) WHEN ((m_OpPmpStatus[i] = k_repaired) & m_PumpOK[i])
k_EmergStop	X	X	X
c_OpPmpStatus[i] =	k_failed	k_repaired	k_none

Tolerance: ??

7.5 c_OpSteamStatus

Relevant Mode Classes: MC_Operating

Initial Value: k_none, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Ideal Value:

Table 15: Event Table

Mode	Triggering Event		
k_initialisation k_Normal k_Degraded k_Rescue	@F(m_SteamOK) OR @T(m_OpSteamStatus = k_none) WHEN ((m_OpSteamStatus = k_repaired) & ~m_SteamOK)	@T(m_OpSteamStatus = k_repaired)	@T(m_OpSteamStatus = k_failed) OR @T(m_OpSteamStatus = k_none) WHEN ((m_OpSteamStatus = k_repaired) & m_SteamOK)
k_EmergStop	X	X	X
c_OpSteamStatus =	k_failed	k_repaired	k_none

Tolerance: ??

7.6 c_OpWatStatus

Relevant Mode Classes: MC_Operating

Initial Value: k_none, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Ideal Value:

Table 16: Event Table

Mode	Triggering Event		
k_initialisation k_Normal k_Degraded k_Rescue	@F(m_WaterOK) OR @T(m_OpWatStatus = k_none) WHEN ((m_OpWatStatus = k_repaired) & ~m_WaterOK)	@T(m_OpWatStatus = k_repaired)	@T(m_OpWatStatus = k_failed) OR @T(m_OpWatStatus = k_none) WHEN ((m_OpWatStatus = k_repaired) & m_WaterOK)
k_EmergStop	X	X	X
c_OpWatStatus =	k_failed	k_repaired	k_none

Tolerance: ??

7.7 c_PumpOn

Relevant Mode Classes: MC_Operating

Initial Value: ForAll i: c_PumpOn[i] = false, @T(m_Power = k_on)

Sustaining Condition: m_Power = k_on

Table 17: Event Table for c_PumpOn[1]

Mode	Triggering Event	
k_initialisation	@T(Enter) WHEN (m_WaterVol < d_N1) OR @T(Drtn(m_WaterVol = 0) = 5s)	@T(m_WaterVol > d_N1)
k_Normal k_Degraded k_Rescue	@T(m_WaterVol < d_N1)	@T(m_WaterVol >= d_N2)
k_EmergStop	X	X
c_PumpOn[1] =	true	false

Table 18: Event Table for c_PumpOn[i],
i in {1, ..., d_NumPumps - 1}

Mode	Triggering Event	
k_initialisation	X	X
k_Normal k_Degraded k_Rescue	@T(m_WaterVol < d_N1)	@T(m_WaterVol >= d_N2)
k_EmergStop	X	X
c_PumpOn[i] =	true	false

Tolerance: ??

7.8 c_Valve

Relevant Mode Classes: MC_Operating

Initial Value: k_closed, d_SystemInit
{assumed to be handled externally in the environment—See section 2.1.3 on page 9.}

Sustaining Condition: m_Power = k_on

Table 19: Event Table

Mode	Triggering Event	
k_initialisation	@T(m_BoilerReady) WHEN m_WaterVol > d_N2	@T(m_WaterVol = 0)
k_Normal k_Degraded k_Rescue k_EmergStop	X	X
c_Valve	k_open	k_closed

Tolerance: ??

8.0 Monitored State Variable Precisions

State Variable	Precision
m_BoilerReady	Delay(m_BoilerReady, 5s)
m_CommOK	Delay(m_CommOK, 5s)
m_OpMonStatus[i]	Delay(m_OpMonStatus[i], 5s)
m_OpPmpStatus[i]	Delay(m_OpPmpStatus[i], 5s)
m_OpSteamStatus	Delay(m_OpSteamStatus, 5s)
m_OpWatStatus	Delay(m_OpWatStatus, 5s)
m_PmpMonOK[i]	Delay(m_PmpMonOK[i], 5s)
m_Power	Delay(m_Power, 500 ms)
m_PumpCirc[i]	Delay(m_PumpCirc[i], 5s)
m_PumpOK[i]	Delay(m_PumpOK[i], 5s)
m_PumpOn[i]	Delay(m_PumpON[i], 5s)
m_Steam	m_Steam +/- (5s * max(d_U1, d_U2))
m_SteamOK	Delay(m_SteamOK, 5s)
m_StopReq	Delay(m_StopReq, 5s)
m_UnitsReady	Delay(m_UnitsReady, 5s)
m_WaterOK	Delay(m_WaterOK, 5s)
m_WaterVol	DelayRng(m_WaterVol, 5s, ??)

9.0 Fundamental Assumptions and Expected Changes

9.1 Fundamental Assumptions

??

9.2 Expected Changes

??

10.0 Dictionary

d_C :Volume:
= ?? {Maximum water capacity of boiler vessel)

d_ExLimits :Boolean:
= (m_WaterVol + (d_NumPumps * d_P * 5s)) >= d_M2) OR
(m_WaterVol - (d_W * 5s) <= d_M1)
{Danger of exceeding the absolute water level limits}

d_MonsOK :Boolean:
= ForAll i (m_PmpMonOK[i])

d_NormWaterRange :Boolean:
= (d_N1 <= m_WaterVol <= d_N2)

d_NumPumps :Integer:
= 4
(d_NumPumps >= 1)

d_M1 :Volume:
= ?? {Minimum safe water volume limit.}
d_M1 < d_M2

d_M2 :Volume:
= ?? {Maximum safe water volume limit.}

d_N1 :Volume:
= ?? {Lower water volume setpoint.}
(d_N1 > d_M1) & (d_N1 < d_N2)

d_N2 :Volume:
= ?? {Upper water volume setpoint.}
(d_N2 < d_M2)

d_P :Flow:
= ?? {Nominal pump flow capacity.}

d_PumpsOK :Boolean:
= ForAll i (m_PumpOK[i])

d_SystemInit = @T(m_BoilerReady) WHEN (m_Power = k_on)

d_U1	:FlowGradient: = ?? {Maximum steam outflow gradient increase}
d_U2	:FlowGradient: = ?? {Maximum steam outflow gradient decrease.}
d_W	:Flow: = ?? {Maximum steam outflow.}

11.0 Glossary

??

12.0 References

[1] Abrial, Jean-Raymond, Steam-boiler control specification problem. In *Dagstuhl Meeting: Method for Semantics and Specifications*, August 1994.

[2] van Schouwen, A.J., *The A-7 Requirements Model: Re-examination for Real-Time Systems and an Application to Monitoring Systems*, M.Sc. dissertation, Department of Computing and Information Science, Queen's University, Kingston, ON, 1990. (Also available as Technical Report 90-276, Queen's University.)

[3] van Schouwen, A.J., Parnas, D.L., Madey, J., *Documentation of Requirements for Computer Systems*, Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, California, January 4-6, 1993, pp. 198-207.

13.0 Indices

13.1 State Variable Value Index

This index shows where the values of environmental state variables are restricted, either by physical limitations, required behaviour, or as described in output data item sections.

C

M

13.2 Monitored State Variable/Input Data Item Index

This index shows the data items that can be used to determine the values of monitored environmental state variables.

13.3 Data Item Index

This index shows the location of each data item definition.

I

O