EECS4312 Isolette Assignment

Hovhannes Khachikyan (hovokhc@cse.yorku.ca)
Varsha Ragavendran (varshar@cse.yorku.ca)

November 7, 2017

Prism account used for submission: hovokhc

- You may work on your own or in a team of no more than two students. You must provide both names and Prism logins in the specified space above this.
- **Submit only one document under one Prism account.**
- Keep track of your revisions in the table below.

Requirements Document:
Temperature control for an Isolette

Revisions

Date	Revision	Description
22 October 2017	1.0	Initial requirements document
4 November 2017	2.0	Included function tables for abstract variables
4 November 2017	3.0	Included function tables for controlled variables
5 November 2017	4.0	Included E Descriptions
5 November 2017	5.0	Included R Descriptions
5 November 2017	6.0	Included Context Diagram
5 November 2017	7.0	Included Mode Diagram
5 November 2017	8.0	Included isolette source code in the Appendix
5 November 2017	9.0	Corrected the function table for c_ms
5 November 2017	10.0	Corrected the function table for c_hc
5 November 2017	11.0	Included rationale for state chart
6 November 2017	12.0	Included rationales for all E/R descriptions
6 November 2017	13.0	Included acceptance test

Contents

1. System Overview **5**

2. Goals **6**

3. Context Diagram **7**

4. Monitored Variables **8**

5. Controlled Variables **9**

6. Mode Diagram **10**

7. R-Descriptions **11**

8. E-descriptions **15**

9. Abstract variables needed for the Function Table **17**
9.1. Abstract variable: lo 17
9.2. Abstract variable: hi 17
9.3. Abstract variable: alarm 17

10. Function Tables **18**
10.1. Function Table for heat control: *c_hc* 18
10.2. Function Table for modes: *c_md* 19
10.3. Function Table for temperature displayed: *c_td* 20
10.4. Function table for alarm: *c_al* 21
10.5. Function table for: c_ms 22

11. Validation **23**

12. Use Cases **25**
12.1. Informal Use Case 25
12.2. PVS Use Case 26

13. Acceptance Tests **27**

14. Traceability **28**

15. Glossary **28**

A. Source code **29**
A.1. Time.pvs 29
A.2. isolette.pvs 30
A.3. Top.pvs 38

B. E-descriptions **39**

C. R-descriptions **41**

List of Figures

1. Isolette 5
2. Incubator Safety Problems [2, p98] 6
3. Context Diagram for the Isolette Thermostat 7
4. Statechart for the modes variable c_md 10

List of Tables

1. Monitored Variables 8
2. Controlled Variables 9
3. Function table for the abstract variable: lo 17
4. Function table for abstract variable: hi 17
5. Function table for abstract variable alarm (combines lo and hi) 17
6. Function table for heat control: c_hc 18
7. Function table for modes of the isolette: c_md 19
8. Function table for displayed temperature c_td 20
9. Function table for alarm: c_al 21
10. Function table for c_ms (messages to be displayed to the nurse in order of priority) 22
11. Message descriptions 22

1. System Overview

The System Under Development (SUD) is a computer controller for the thermostat of an Isolette.[1] An Isolette is an incubator for for an infant that provides controlled temperature, humidity and oxygen (Fig. 4). Isolettes are used extensively in Neonatal Intensive Care Units for the care of premature infants.

This requirements document is specifically for the control of temperature. The purpose of the Isolette computer controller is to maintain the air temperature of an Isolette within a desired range. It senses the current temperature of the Isolette and turns the heat source on and off to warm the air as needed. If the temperature falls too far below or rises too far above the desired temperature range, it activates an alarm to alert the nurse. The system allows the nurse to set the desired temperature range and to set the alarm temperature range outside the desired temperature range of which the alarm should be activated. This requirements documents follows the specification in [1] (Appendix A) except where noted.

Figure 1: Isolette

Many babies have died due to faulty incubators. There is thus a standard that manufacturers must satisfy. Modern incubators are equipped with alarms for air temperature, skin temperature, oxygen concentration and humidity. The alarms are both visual such

[1]The image in Fig 4 is from: `www.nufer-medical.ch`.

as red warning lamps, and audio such as beep signals. Once measured measured values exceed permitted limits as well as when faults occur in sensors. For one such incident leading to death see "Medical Devices: Use and Safety" shown in Fig. 3.

CASE 6:2 Baby dies through overheating in incubator

An underdeveloped baby was being treated in an incubator with skin temperature control. When the baby was being washed, the skin sensor was removed and left hanging outside the incubator after the washing. Thus the sensor started measuring the room temperature (approx. 25°C). The control circuits therefore increased the heat to maximum level, and the temperature in the incubator rose to more than 45°C. The baby died.

For increased safety, incubators must be constructed with an extra control circuit that prevents overheating in case the skin sensor is misplaced. The incubator in question was indeed equipped with such a safety circuit, but the circuit was defective.

Figure 2: Incubator Safety Problems [2, p98]

2. Goals

The high-level goals (G) of the system are:

- G1—The Infant should be kept at a safe and comfortable temperature.
- G2—The Nurse should be warned if the Infant becomes too hot or too cold.
- G3—The cost of manufacturing the computer controller for the thermostat should be as low as possible.

3. Context Diagram

See Fig. A-1 in [1]. The System Under Description (SUD) is a computer *controller* to regulate the temperature of the Isolette. Everything else including the Operator Interface (described in [1]) is in the ecosystem (i.e. in the environment of the controller). The monitored variables and controlled variables for the controller are in Table 1 and Table 2, respectively. For clarity, simplicity and safety, there are some differences between the specifications in this document and the descriptions in [1].[2]

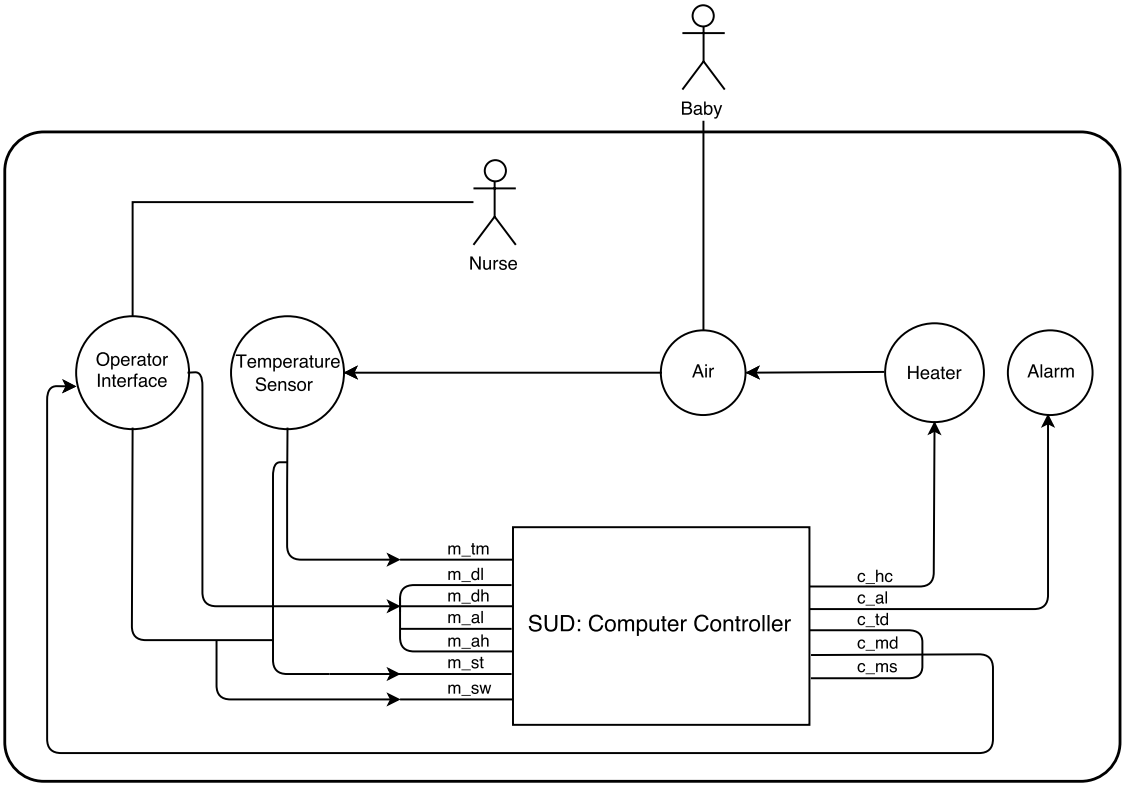

Figure 3: Context Diagram for the Isolette Thermostat

[2]Documented in the write-up to this assignment: `assign1-spec.pdf`.

4. Monitored Variables

The monitored variables are a subset of those described in [1].[3] There is a single status variable *m_st* that is *invalid* whenever any one of the operator inputs or temperature sensor are in a failed state. Otherwise types and ranges are as in [1].

Name	Type	Range	Units	Physical Interpretation
m_tm	$\mathbb{R}$	68.0 .. 105.0	°F	actual temperature of Isolette air temperature from sensor
m_dl	$\mathbb{Z}$	97 .. 99	°F	desired lower temperature set by operator
m_dh	$\mathbb{Z}$	98 .. 100	°F	desired higher temperature set by operator
m_al	$\mathbb{Z}$	93 .. 98	°F	lower alarm temperature set by operator
m_ah	$\mathbb{Z}$	99 .. 103	°F	higher alarm temperature set by operator
m_st	Enumerated	{valid, invalid}		status of sensor and operator settings
m_sw	Enumerated	{on, off}		switch set by operator

Table 1: Monitored Variables

[3] With some change of nomenclature. Monitored variables have an "m" prefix.

5. Controlled Variables

The controlled variables are a subset of those described in [1].[4] In addition, there is a mode display *c_md* and a message display *c_ms*.[5]

Name	Type	Range	Units	Physical Interpretation
c_hc	Enumerated	{on, off}		heat control: command to turn heat source on or off
c_td	$\mathbb{Z}$	$\{0\} \cup \{68 \,..\, 105\}$	°F	displayed temperature of Isolette (zero when Isolette is off)
c_al	Enumerated	{off, on}		sound alarm to call nurse
c_md	Enumerated	{off, init, normal, failed}		mode of Isolette operation (failed if $m\_st = invalid$)
c_ms	Enumerated	{ok, err1, err2 err3, err4, err5, err6}		messages to display to nurse

Table 2: Controlled Variables

[4] With some change of nomenclature. Controlled variables have a "c" prefix.

[5] The mode "off" is added to that of Fig. A-4 in [1], and the mode transitions have been changed.

6. Mode Diagram

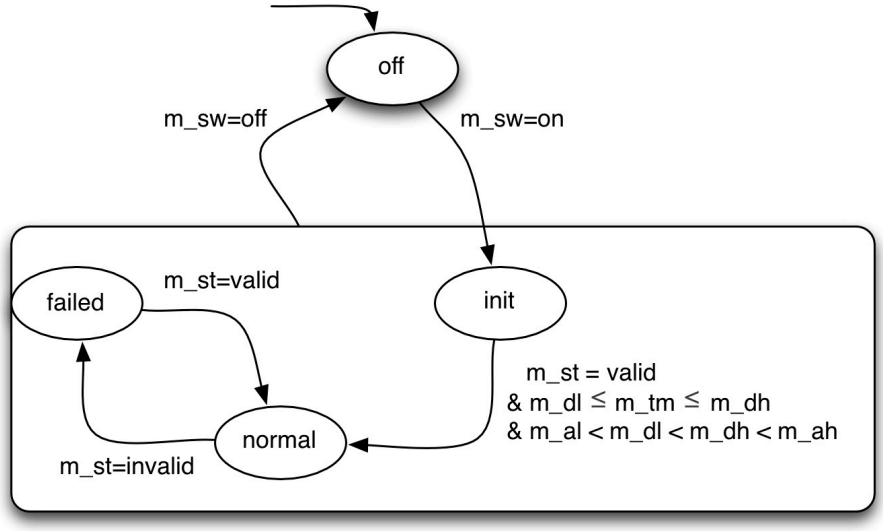

Figure 4: Statechart for the modes variable c_md

The isolette has four main modes of operation *off, init, normal, failed*. The purpose of the *off* mode is to ensure that the isolette cannot operate unless physically switched on by the operator. The rationale as to why the *off* mode is outside of the *****, is to show that the isolette can be in any of the other modes while operating provided that the switch is on. If the switch is off, the isolette can only be in the *off* mode. In the *init* mode, the operator will be able to enter the *desired low and high* and *lower alarm and higher alarm* temperatures. The isolette shall only switch into the normal mode of operation if and only if the operator enters valid temperature ranges, the current temperature in the isolette is within the desired range and the the sensor is valid. Therefore, the significance of the *init* mode is to ensure that the isolette is safe, thereby switching into *normal* mode, in which the operator can place the baby into the isolette. In the *normal* mode, the isolette ensures that baby will be exposed to safe temperatures. However, the system can be faulty at anytime (i.e. hardware circuitry failure, sensor failure, etc). To account for these failures, the system will transition into *failed* mode of operation if the sensor is invalid. The isolette can recover from the *failed* mode, once the sensor is valid and also if the operator changes the settings to the temperature ranges.

7. R-Descriptions

We have already elicited the following R-descriptions.

REQ1	The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*.	See statechart in Fig. **??**.

REQ2	In the *normal* mode, the temperature controller shall maintain current temperature inside the Isolette within a set temperature range (the *desired* range).	The *desired* temperature range is *m_dl*.. *m_dh*. If the current temperature *m_tm* is outside this range, the controller shall turn the heater on or off via the controlled variable *m_hc* to maintain the desired state.

Rationale: The *desired temperature range* will be set by the nurse to the desired range based on the infant's weight and health. The controller shall maintain the current temperature within this range under normal operation.

The following relevant hazard was identified through the safety assessment process:

- **H1**: Prolonged exposure of Infant to unsafe heat or cold;
- *Classification*: catastrophic;
- *Probability*: $< 10^{-9}$ per hour of operation.

To ensure that probability of hazard H1 is 10^{-9} per hour of operation, the following derived safety requirement shall apply to the Isolette controller:

REQ3	In *normal* mode, the controller shall activate an alarm whenever • the current temperature falls outside the *alarm* temperature range (either through temperature fluctuation or a change in the alarm range by an operator), or • a failure is signalled in any of the input devices (temperature sensor and operator settings).	The alarm temperature range is $m\_al .. m\_ah$. Monitored variable m_st shows "invalid" when any of the input signals fail.

REQ4	Once the alarm is activated, it becomes deactivated in one of two ways: • The nurse turns off the Isolette; • The alarm has lasted for 10 seconds, and after 10 seconds or more the alarm conditions are removed.	Refer to the relevant tables of monitored and/or controlled variables and function tables.

REQ5	The *controller* shall turn on the heat control, when: • the isolette is in *init* mode of operation, the operator has entered valid *desired* and *alarm* temperatures and the sensor is valid. • the isolette is in *normal* mode of operation and the *current temperature* falls below the *desired low temperature*.	The controlled variable *c_hc = on* when the *current temperature* falls below the *desired low* temperature (*m_tm <m_dl*)

Rationale: The *controller* shall turn on the heat control to get the current temperature within the *desired low and high* temperatures, so that the operator can place the baby into the isolette. The *controller* shall maintain the current temperature within the range under normal operation *m_dl*≤*m_tm*≤*m_dh*. If this requirement is not satisfied, the baby will be exposed to unsafe temperatures which will affect the baby negatively.

REQ6	The *controller* shall deactivate the heat control, in one of the following ways: • the isolette mode of operation switches from *normal* to *failed* • the isolette is in *normal* mode of operation and the *current temperature* rises above the *desired high temperature*	The controlled variable *c_hc = off* when the isolette is in *failed* mode of operation (*c_md=failed*), or when the *current temperature* rises above *desired high temperature* (*m_tm>m_dh*)

Rationale: The isolette cannot regulate the *current temperature* of the isolette in *off* and *failed* modes of operation. Therefore, the *controller* will turn the heat control off.

REQ7	In the *failed* mode: • the alarm shall be on • the heat control shall be turned off • the displayed temperature is 0°F • an error message shall be displayed	Refer to Figure 4, REQ-MHS-5, REQ-MA-5 and table 10

Rationale: In the event of equipment failure, the system can no longer be safe for the baby, hence an alarm raised, the heat control and temperature display is disabled, and an appropriate message is displayed to the nurse.

See Appendix C for more R-descriptions

8. E-descriptions

ENV1	The current temperature received from the sensor is a a real number in the range 68.0 to 105.0°F.	The current temperature range is between 68.0 ..105.0°F. The monitored variable is within this range, 68.0 ≤ $m\_tm \leq 105.0$

Rationale: This is the specified range of operation for the temperature sensor. The Computer controller will not deal with temperatures above or below the stated range.

ENV2	A single message is displayed to the nurse at a given time.	Refer to table 2 and 10

Rationale: In order to avoid confusion, a single message will be displayed to the nurse at a given time. For priority of messages refer to table 10

ENV3	At any unit of time, the sensor can be invalid as a result of hardware failure in the temperature sensors or in the operator interface unit.	Refer to table 1

Rationale: The temperature sensor, and the operator interface are part of the environment, hence may fail at any time. When such failure occurs, there is some hardware circuitry (also in the environment) that sets m_st = invalid.

ENV4	The desired and alarm temperatures received from the operator are all in increments of 1°F.	The desired temperatures: • *m_dl* – desired low temperature • *m_dh* – desired high temperature The alarm temperatures: • *m_al* – lower alarm temperature • *m_ah* – higher alarm temperature

Rationale: Marketing studies have shown that customers prefer to set temperatures in 1 degree increments. A resolution 1°F is sufficient to be consistent with the functional and performance requirements specified in the rest of the document.

See Appendix B for more E-descriptions

9. Abstract variables needed for the Function Table

9.1. Abstract variable: lo

		lo(i)
i=0		off
i>0	m_tm(i) < m_al(i)	on
	m_tm(i) ≥ m_al(i) ∧ m_tm(i) < (m_al(i) + EPS)	lo(i-1)
	m_tm(i) ≥ (m_al(i) + EPS)	off

Table 3: Function table for the abstract variable: lo

9.2. Abstract variable: hi

		hi(i)
i=0		off
i>0	m_tm(i) >m_ah(i)	on
	m_tm(i) ≤ m_ah(i) ∧ m_tm(i) ≥ (m_ah(i) - EPS)	hi(i-1)
	m_tm(i) ≤ (m_ah(i) - EPS)	off

Table 4: Function table for abstract variable: hi

Where EPS = 0.5

9.3. Abstract variable: alarm

	alarm(i)
lo(i) = on ∨ hi(i) = on	on
¬ (lo(i) = on ∨ hi(i) = on)	off

Table 5: Function table for abstract variable alarm
(combines lo and hi)

10. Function Tables

10.1. Function Table for heat control: c_hc

				c_hc(i)
i=0				off
i>0	c_md(i) = off ∨ c_md(i) = failed			off
	c_md(i) = init	m_st(i) = valid	env1	on
			¬ env1	off
		m_st(i) = invalid		off
	c_md(i) = normal	m_tm(i) < m_dl(i)		on
		m_tm(i) > m_dh(i)		off
		m_dl(i) ≤ m_tm(i) ∧ m_tm(i) ≤ m_dh(i)		c_hc(i-1)

Table 6: Function table for heat control: c_hc

$$\text{env1} \triangleq \text{m\_al(i)} < \text{m\_dl(i)} \wedge \text{m\_dl(i)} < \text{m\_dh(i)} \wedge \text{m\_dh(i)} < \text{m\_ah(i)}$$

10.2. Function Table for modes: c_md

				c_md(i)
i=0				off
i>0	m_sw(i) = off			off
	m_sw(i) = on	c_md(i-1) = off		init
		c_md(i-1) = init	m_st(i) = valid ∧ m_dl(i) ≤ m_tm(i) ≤ m_dh(i) ∧ env1	normal
			¬ (m_st(i) = valid ∧ m_dl(i) ≤ m_tm(i) ≤ m_dh(i) ∧ env1)	c_md(i-1)
		c_md(i-1) = normal	m_st(i) = valid	c_md(i-1)
			m_st(i) = invalid	failed
		c_md(i-1) = failed	m_st(i) = invalid	c_md(i-1)
			m_st(i) = valid	normal

Table 7: Function table for modes of the isolette: c_md

$$\text{env1} \triangleq \text{m\_al(i)} < \text{m\_dl(i)} \wedge \text{m\_dl(i)} < \text{m\_dh(i)} \wedge \text{m\_dh(i)} < \text{m\_ah(i)}$$

10.3. Function Table for temperature displayed: c_td

	c_td(i)
c_md(i) = normal	$\lfloor m\_tm + 0.5 \rfloor$
c_md(i) = off $\vee$ c_md(i) = init $\vee$ c_md(i) = failed	0

Table 8: Function table for displayed temperature c_td

10.4. Function table for alarm: c_al

				c_al(i)
i=0				off
i>0	c_md(i) = off ∨ c_md(i) = init			off
	c_md(i) = normal	env1 ∧ m_st(i) = valid	alarm(i) = on ∨ ¬ held_for(c_al_pred, 10)(i-1)	on
			alarm(i) = off ∧ held_for(c_al_pred, 10)(i-1)	off
		¬ (env1 ∧ m_st(i) = valid)		on
	c_md(i) = failed			on

Table 9: Function table for alarm: c_al

$$\text{env1} \triangleq \text{m\_al(i)} < \text{m\_dl(i)} \wedge \text{m\_dl(i)} < \text{m\_dh(i)} \wedge \text{m\_dh(i)} < \text{m\_ah(i)}$$
$$\text{c\_al\_pred} \triangleq \text{c\_al(i)} = \text{on}$$

10.5. Function table for: c_ms

		c_ms(i)
i=0	c_md(i) = off	ok
i>0	m_st(i) = invalid	err1
	m_tm(i) > m_ah(i)	err2
	m_tm(i) < m_al(i)	err3
	m_al(i) ≥ m_dl(i)	err4
	m_dl(i) ≥ m_dh(i)	err5
	m_dh(i) ≥ m_ah(i)	err6
	ELSE	ok

Table 10: Function table for c_ms (messages to be displayed to the nurse in order of priority)

c_ms	Description
ok	All good
err1	Failure in temperature sensor or control interface
err2	Actual temperature of the isolette exceeds high alarm temperature
err3	Actual temperature of the isolette is less than the lower alarm temperature
err4	Lower alarm temperature must be less than the lower desired temperature
err5	Lower desired temperature must be less than high desired temperature
err6	High desired temperature must be less than high alarm temperature

Table 11: Message descriptions

11. Validation

Proof of completeness and disjointness and validation of the requirements using PVS.

```
*** top (19:47:43 11/5/2017)
*** Generated by proveit - ProofLite-6.0.9 (3/14/14)
*** Trusted Oracles
***   MetiTarski: MetiTarski Theorem Prover via PVS proof
***   rule metit

 Proof summary for theory top
   Theory totals: 0 formulas, 0 attempted, 0 succeeded (0.00 s)

 Proof summary for theory Time
   r2d_TCC1 ..... proved - complete
   d2r_TCC1 ..... proved - complete
   held_for_TCC1 ..... proved - complete
   Theory totals: 3 formulas, 3 attempted, 3 succeeded (0.34 s)

 Proof summary for theory isolette
   mode_TCC1 ..... proved - complete
   mode_TCC2 ..... proved - complete
   mode_TCC3 ..... proved - complete
   mode_TCC4 ..... proved - complete
   mode_TCC5 ..... proved - complete
   mode_TCC6 ..... proved - complete
   mode_TCC7 ..... proved - complete
   mode_TCC8 ..... proved - complete
   mode_TCC9 ..... proved - complete
   mode_TCC10 ..... proved - complete
   mode_TCC11 ..... proved - complete
   mode_TCC12 ..... proved - complete
   display_temperature_TCC1 ..... proved - complete
   display_temperature_TCC2 ..... proved - complete
   heat_control_TCC1 ..... proved - complete
   heat_control_TCC2 ..... proved - complete
   heat_control_TCC3 ..... proved - complete
   heat_control_TCC4 ..... proved - complete
   heat_control_TCC5 ..... proved - complete
```

```
    heat_control_TCC6 ..... proved - complete
    heat_control_TCC7 ..... proved - complete
    low_alarm_TCC1 ..... proved - complete
    low_alarm_TCC2 ..... proved - complete
    low_alarm_TCC3 ..... proved - complete
    high_alarm_TCC1 ..... proved - complete
    high_alarm_TCC2 ..... proved - complete
    high_alarm_TCC3 ..... proved - complete
    alarm_check_TCC1 ..... proved - complete
    alarm_TCC1 ..... proved - complete
    alarm_TCC2 ..... proved - complete
    alarm_TCC3 ..... proved - complete
    alarm_TCC4 ..... proved - complete
    alarm_TCC5 ..... proved - complete
    alarm_TCC6 ..... proved - complete
    inv1 ..... proved - complete
    inv2 ..... proved - complete
    inv3 ..... proved - complete
    inv4 ..... proved - complete
    alarm_req_entails_inv ..... proved - complete
    usecase1 ..... proved - complete
    usecase2 ..... proved - complete
    usecase3 ..... proved - complete
    usecase4 ..... proved - complete
    usecase5 ..... proved - complete
Theory totals: 44 formulas, 44 attempted, 44 succeeded (4.15 s)

Grand Totals: 47 proofs, 47 attempted, 47 succeeded (4.48 s)
```

Listing 1: Disjointedness and completeness proves for Time.pvs and isolette.pvs

12. Use Cases

See Section A2 of [1] for some use cases.

12.1. Informal Use Case

Use Case: Alarm Deactivation

Primary Actor: Thermostat controller

Precondition:

- Thermostat controller is in the *normal* mode
- Alarm is turned off

Postcondition:

- Thermostat controller is in the *normal* mode
- Alarm is turned off

Main Success Scenario:

1. Thermostat controller detects an internal failure (temperature sensor, or operator interface)
2. Thermostat controller enters the *failed* mode
3. Thermostat controller activates the alarm
4. Internal failure is resolved
5. Thermostat controller enters *normal* mode
6. Alarm continues to sound for 10 seconds
7. Alarm is deactivated

12.2. PVS Use Case

PVS implementation for the above informal use case.

```
usecase5: CONJECTURE
    m_sw(1) = on AND m_sw(2) = on AND
    m_sw(3) = on AND m_sw(4) = on AND
    m_sw(5) = on AND m_sw(6) = on AND
    m_sw(7) = on AND m_sw(8) = on AND
    m_sw(9) = on AND m_sw(10) = on AND
    m_sw(11) = on AND m_sw(12) = on AND
    m_sw(13) = on AND

    m_al(2) = 93 AND m_ah(2) = 103 AND
    m_dl(2) = 94 AND m_dh(2) = 100 AND m_tm(2) = 96 AND

    m_st(2) = valid AND m_st(3) = invalid AND
    m_st(4) = valid AND m_st(5) = valid AND
    m_st(6) = valid AND m_st(7) = valid AND
    m_st(8) = valid AND m_st(9) = valid AND
    m_st(10) = valid AND m_st(11) = valid AND
    m_st(12) = valid AND

    mode(0) AND mode(1) AND mode(2) AND
    mode(3) AND mode(4) AND mode(5) AND mode(6) AND
    mode(7) AND mode(8) AND mode(9) AND mode(10) AND
    mode(11) AND mode(12) AND mode(13) AND

    alarm(3) AND alarm(4) AND alarm(5) AND alarm(6) AND
    alarm(7) AND alarm(8) AND alarm(9) AND alarm(10) AND
    alarm(11) AND alarm(12) AND alarm(13)

    IMPLIES

    c_al(3) = on AND c_al(4) = on AND c_al(5) = on AND
    c_al(6) = on AND c_al(7) = on AND c_al(8) = on AND
    c_al(9) = on AND c_al(10) = on AND c_al(11) = on AND
    c_al(12) = on AND c_al(13) = off
```

Listing 2: Alarm deactivation use case in pvs

13. Acceptance Tests

See section 12.1 for the informal use case.

Precondition

- m_sw(i) = on
- m_al(i) < m_ah(i) < m_dl(i) < m_dh(i)
- c_al(i) = off
- c_md(i) = normal

Intermediate step:

- m_st(i) = invalid
- c_md(i) = failed
- c_hc(i) = off
- c_al(i) = on (kept for 10 seconds)
- m_st(i+1) = valid
- c_md(i+1) = normal

Postcondition:

- m_st(i) = valid
- c_al(i) = off
- c_md(i) = normal

14. Traceability

Matrix to show which acceptance tests passed, and which R-descriptions they checked. No need to do this for this assignment.

15. Glossary

The definition of important terms is placed in this section. You are not required to complete this.

References

[1] US FAA. Requirements Engineering Management Handbook. Technical Report DOT/FAA/AR-08/32, U.S. Department of Transportation Federal Aviation Administration, June 2009.

[2] Bertil Jacobson and Alan Murray. *Medical Devices: Use and Safety*. Elsevier, 2007.

A. Source code

A.1. Time.pvs

```
Time[delta: posreal]  : THEORY
BEGIN
  % digital time
  DTIME: TYPE = nat
  init(i: DTIME) : bool = i = 0

  % psuedo, digitized real time
  RTIME: TYPE = {t : nnreal | (EXISTS (i : DTIME) : t = i * delta)}

  % actual time
  TIME: TYPE = nnreal

  % Positive DTIME
  POS_DTIME: TYPE = posnat

  % conversions
  r2d(t: RTIME): DTIME = t / delta
  d2r(i: DTIME): RTIME = i * delta

  % held_for is defined for negative times
  % so that we don't need to guard those function applications
  % in tables. If the period of held_for covers any
  % negative time instant, it is automatically false
  DURATION: TYPE = nnreal

  held_for(p: pred[DTIME], d: DURATION)(i:DTIME): bool =
    (FORALL (j: int):
      i - (d / delta) <= j AND j <= i IMPLIES 0 <= j AND p(j))

END Time
```

Listing 3: Time theory source code

A.2. isolette.pvs

```
isolette : THEORY

  BEGIN

  delta: posreal
  IMPORTING Time[delta]

  i: VAR DTIME

  % Valid Ranges and Modes of Monitored and Controlled Variables
  M_AL_RANGE: TYPE = {t:integer | t>=93 AND t<=98}  CONTAINING 98
  M_AH_RANGE: TYPE = {t:integer | t>=99 AND t<=103} CONTAINING 103
  M_DL_RANGE: TYPE = {t:integer | t>=97 AND t<=99}  CONTAINING 99
  M_DH_RANGE: TYPE = {t:integer | t>=98 AND t<=100} CONTAINING 100
  M_TM_RANGE: TYPE = {t:integer | t>=65 AND t<=105} CONTAINING 105
  ON_OFF_MODES: TYPE = {on, off}
  HEAT_CONTROL_MODES: TYPE = {on, off}
  SENSOR_MODES: TYPE = {valid, invalid}
  ISOLETTE_MODES: TYPE = {off, init, normal, failed}
  ERROR_MODES: TYPE = {ok, err1, err2, err3, err4, err5, err6}

  %Timing Resolution of Alarm
  EPS: real = 0.5

  %MONITORED Variables
  m_ah: [DTIME -> M_AH_RANGE] % higher alarm temperature
  m_tm: [DTIME -> M_TM_RANGE] % actual temperature
  m_al: [DTIME -> M_AL_RANGE] % lower alarm temperature
  m_dl: [DTIME -> M_DL_RANGE] % desired low temperature
  m_dh: [DTIME -> M_DH_RANGE] % desired high temperature
  m_st: [DTIME -> SENSOR_MODES] % status of sensor
  m_sw: [DTIME -> ON_OFF_MODES] % switch

  %CONTROLLED Variables
  lo : [DTIME -> ON_OFF_MODES] % low alarm temp [abstract var]
  hi : [DTIME -> ON_OFF_MODES] % high alarm temp [abstract var]
  alarm : [DTIME -> ON_OFF_MODES] % internal alarm considering
                                  % low & high [abstract var]
```

```
c_al: [DTIME -> ON_OFF_MODES] % Main Alarm of Isolette
c_md: [DTIME -> ISOLETTE_MODES] % mode of isolette operation
c_hc: [DTIME -> HEAT_CONTROL_MODES] % heat control
c_ms: [DTIME -> ERROR_MODES] % messages to display
c_td: [DTIME -> integer] % temperature display

%ENVIRONMENT CONDITIONS
env1: bool =
FORALL(i:DTIME):
    m_al(i) < m_dl(i) AND m_dl(i) < m_dh(i) AND m_dh(i) < m_ah(i)

env2 : bool =
c_md(0) = off

env3 : bool =
FORALL(i:DTIME) :
    (m_ah(i)-EPS) > (m_al(i)+EPS) AND m_ah(i) < (m_al(i)-EPS)

%-----------------------------------------------------------%
%Mode Function:
% Determines the mode of isolette operation
% {off, init, normal, failed}
mode(i:DTIME) : bool =
COND
    i = 0 -> c_md(i) = off,
    i > 0 ->
    COND
        m_sw(i) = off -> c_md(i) = off,
        m_sw(i) = on ->
        COND
            c_md(i-1) = off -> c_md(i) = init,
            c_md(i-1) = init ->
            COND
                m_st(i) = valid
                  AND m_dl(i) <= m_tm(i) <= m_dh(i)
                  AND env1 -> c_md(i) = normal,
                not(m_st(i) = valid
                  AND m_dl(i) <= m_tm(i) <= m_dh(i)
                  AND env1) -> c_md(i) = c_md(i-1)
            ENDCOND,
            c_md(i-1) = normal ->
```

```
            COND
                m_st(i) = valid -> c_md(i) = c_md(i-1),
                m_st(i) = invalid -> c_md(i) = failed
            ENDCOND,
            c_md(i-1) = failed ->
            COND
                m_st(i) = invalid -> c_md(i) = c_md(i-1),
                m_st(i) = valid -> c_md(i) = normal
            ENDCOND
        ENDCOND
    ENDCOND
ENDCOND
%-------------------------------------------------------------%
%Display Messages Function:
%      Determines the message to be displayed
display_messages(i:DTIME) : bool =
COND
    i = 0 -> c_ms(i) = ok,
    i > 0 ->
    IF m_st(i) = invalid THEN
        c_ms(i) = err1
    ELSIF m_tm(i) > m_ah(i) THEN
        c_ms(i) = err2
    ELSIF m_tm(i) < m_al(i) THEN
        c_ms(i) = err3
    ELSIF m_al(i) >= m_dl(i) THEN
        c_ms(i) = err4
    ELSIF m_dl(i) >= m_dh(i) THEN
        c_ms(i) = err5
    ELSIF m_dh(i) >= m_ah(i) THEN
        c_ms(i) = err6
    ELSE
        c_ms(i) = ok
    ENDIF
ENDCOND
%-------------------------------------------------------------%
%Display Temperature Function:
%     Displays the temperature of the isolette.
display_temperature(i:DTIME) : bool =
COND
    c_md(i) = normal -> c_td(i) = floor(m_tm(i)+0.5),
```

```
    c_md(i) = off OR c_md(i) = init OR c_md(i) = failed
      -> c_td(i) = 0
ENDCOND
%-----------------------------------------------------------%
%Heat Control Function:
%     Determines the mode of heat control (thermostat) {on,off}.
heat_control(i:DTIME) : bool =
COND
    i = 0 -> c_hc(i) = off,
    i > 0 ->
    COND
        c_md(i) = off OR c_md(i) = failed -> c_hc(i) = off,
        c_md(i) = init ->
        COND
            m_st(i) = valid ->
            COND
                env1 -> c_hc(i) = on,
                NOT env1 -> c_hc(i) = off
            ENDCOND
            ,m_st(i) = invalid -> c_hc(i) = off
        ENDCOND
        ,c_md(i) = normal ->
        COND
            m_tm(i) < m_dl(i) -> c_hc(i) = on,
            m_tm(i) > m_dh(i) -> c_hc(i) = off,
            m_dl(i) <= m_tm(i) AND m_tm(i) <= m_dh(i)
              -> c_hc(i) = c_hc(i-1)
        ENDCOND
    ENDCOND
ENDCOND
%-----------------------------------------------------------%
%Predicate Function:
%    [DTIME -> Bool]
c_al_pred(i:DTIME) : bool = c_al(i) = on

%Low Alarm Function: [Abstract Variable]
%  Determines if actual temp is within the low alarm bandwidth.
%  Low Alarm is turned on if actual temp is below
%  low alarm bandwidth.
low_alarm(i:DTIME) : bool =
COND
```

```
    i = 0 -> lo(i) = off,
    i > 0 ->
    COND
        m_tm(i) < m_al(i) -> lo(i) = on,
        m_tm(i) >= m_al(i) AND m_tm(i) < (m_al(i)+EPS)
          -> lo(i) = lo(i-1),
        m_tm(i) >= (m_al(i)+EPS) -> lo(i) = off
    ENDCOND
ENDCOND

%High Alarm Function: [Abstract Variable]
%  Determines if actual temp is within the high alarm bandwidth.
%  High Alarm is turned on if actual temp is above
%  high alarm bandwidth.
high_alarm(i:DTIME) : bool =
COND
    i = 0 -> hi(i) = off,
    i > 0 ->
    COND
        m_tm(i) > m_ah(i) -> hi(i) = on,
        m_tm(i) <= m_ah(i) AND m_tm(i) >= (m_ah(i)-EPS)
          -> hi(i) = hi(i-1),
        m_tm(i) < m_ah(i)-EPS -> hi(i) = off
    ENDCOND
ENDCOND

%Alarm Check Function: [Abstract Variable]
%    Determines if High Alarm or Low Alarm is on.
alarm_check(i:DTIME) : bool =
COND
    lo(i) = on OR hi(i) = on -> alarm(i) = on,
    NOT(lo(i) = on OR hi(i) = on) -> alarm(i) = off
ENDCOND

%Alarm Function:
%    Controls the main alarm of the isolette to notify nurse
alarm(i:DTIME) : bool =
COND
    i = 0 -> c_al(i) = off,
    i > 0 ->
    COND
```

```
        c_md(i) = off OR c_md(i) = init -> c_al(i) = off,
        c_md(i) = normal ->
        COND
            env1 AND m_st(i) = valid ->
            COND
                alarm(i) = on OR NOT held_for(c_al_pred,10)(i-1)
                  -> c_al(i) = on,
                alarm(i) = off AND held_for(c_al_pred,10)(i-1)
                  -> c_al(i) = off
            ENDCOND,
            NOT(env1 AND m_st(i) = valid) -> c_al(i) = on
        ENDCOND,
        c_md(i) = failed -> c_al(i) = on
    ENDCOND
ENDCOND

% No Simultaneous Alarm Function:
%  Alarm cannot be in on mode at low alarm bandwidth
%  and high alarm bandwidth.
no_simultaneous_alarm(i:DTIME) : bool =
  NOT(lo(i)=on AND hi(i)=on)
 %-----------------------------------------------------------%

%INVARIANTS
inv1 : CONJECTURE
FORALL(i:DTIME) :
    c_md(i) = off OR
    c_md(i) = init OR
    c_md(i) = normal OR
    c_md(i) = failed

inv2 : CONJECTURE
FORALL(i:DTIME) :
    c_ms(i) = ok OR
    c_ms(i) = err1 OR
    c_ms(i) = err2 OR
    c_ms(i) = err3 OR
    c_ms(i) = err4 OR
    c_ms(i) = err5 OR
    c_ms(i) = err6
```

```
inv3 : CONJECTURE
EXISTS(i:DTIME) :
    alarm(i) AND c_md(i) = failed IMPLIES c_al(i) = on
       %c_md(i) = failed OR c_low_high(i)=on IMPLIES c_al(i)=on

inv4 : CONJECTURE
FORALL(i: DTIME) :
    (i > 0 AND alarm(i) AND
    NOT(c_md(i) = init OR  c_md(i) = off OR c_md(i) = failed))
    IMPLIES
    (alarm(i) = on => c_al(i) = on)

alarm_req_entails_inv : THEOREM
FORALL(i:DTIME) :
    env3 AND alarm_check(i) IMPLIES no_simultaneous_alarm(i)
 %-----------------------------------------------------------%
%USE CASES
usecase1: CONJECTURE
m_sw(1) = on
AND m_tm(1) = 95
AND m_ah(1) = 103
AND m_al(1) = 97
AND m_dl(1) = 98
AND m_dh(1) = 100
AND m_st(1) = valid
AND mode(1) AND mode(0)
    IMPLIES
c_md(0)=off AND c_md(1)=init

usecase2: CONJECTURE
m_sw(0) = off
AND m_ah(0) = 99
AND m_al(0) = 94
AND m_dl(0) = 97
AND m_dh(0) = 98
AND alarm(0)
    IMPLIES
c_al(0)=off

usecase3: CONJECTURE
m_sw(1) = on
```

```
 AND m_ah(1) = 99
 AND m_al(1) = 94
 AND m_dl(1) = 97
 AND m_dh(1) = 98
 AND alarm(1)
 AND mode(1) AND mode(0)
     IMPLIES
 c_al(1) = off

usecase4: CONJECTURE
 m_ah(2)=99
 AND m_al(2)=94
 AND m_dl(2)=97
 AND m_dh(2)=99
 AND alarm(2)
 AND mode(2) AND mode(1) AND mode(0)
 AND alarm_check(2)
 AND high_alarm(2)
 AND m_st(2)=valid
 AND m_sw(1)=on AND m_sw(2)=on
 AND m_tm(2)=98
     IMPLIES
 c_al(2)=off

 usecase5: CONJECTURE
   m_sw(1) = on AND m_sw(2) = on AND
   m_sw(3) = on AND m_sw(4) = on AND
   m_sw(5) = on AND m_sw(6) = on AND
   m_sw(7) = on AND m_sw(8) = on AND
   m_sw(9) = on AND m_sw(10) = on AND
   m_sw(11) = on AND m_sw(12) = on AND
   m_sw(13) = on AND

   m_al(2) = 93 AND m_ah(2) = 103 AND
   m_dl(2) = 94 AND m_dh(2) = 100 AND m_tm(2) = 96 AND

   m_st(2) = valid AND m_st(3) = invalid AND
   m_st(4) = valid AND m_st(5) = valid AND
   m_st(6) = valid AND m_st(7) = valid AND
   m_st(8) = valid AND m_st(9) = valid AND
   m_st(10) = valid AND m_st(11) = valid AND
```

```
    m_st(12) = valid AND

    mode(0) AND mode(1) AND mode(2) AND
    mode(3) AND mode(4) AND mode(5) AND mode(6) AND
    mode(7) AND mode(8) AND mode(9) AND mode(10) AND
    mode(11) AND mode(12) AND mode(13) AND

    alarm(3) AND alarm(4) AND alarm(5) AND alarm(6) AND
    alarm(7) AND alarm(8) AND alarm(9) AND alarm(10) AND
    alarm(11) AND alarm(12) AND alarm(13)

    IMPLIES

    c_al(3) = on AND c_al(4) = on AND c_al(5) = on AND
    c_al(6) = on AND c_al(7) = on AND c_al(8) = on AND
    c_al(9) = on AND c_al(10) = on AND c_al(11) = on AND
    c_al(12) = on AND c_al(13) = off

  END isolette
```

Listing 4: Isolette theory source code

A.3. Top.pvs

```
top: THEORY
BEGIN
    IMPORTING Time
    IMPORTING isolette
END top
```

Listing 5: top.pvs source code

B. E-descriptions

ENV5	The lower alarm temperature set by the operator is an integer is in the range of 93 ..98.	The monitored variable *m_al* for the *lower alarm*, is within this range $(93 \leq m\_al \leq 98)$.

Rationale: Exposure to temperatures less than 93°F will result in hypothermia, which can lead to death within a few minutes for severely ill preterm infants.

ENV6	The Lower Alarm Temperature will always be less than or equal to the Lower Desired Temperature of −1°F.	The monitored variable *m_al* for the *lower alarm temperature* will be, *m_al leq m_dl* .

Rationale: If the Lower Alarm Temperature is greater than or equal to the Lower Desired Temperature, the Alarm could be activated while the Current Temperature is still in the Desired Temperature Range.

ENV7	The desired lower temperature set by the operator is an integer in the range of 97..99.	The monitored variable *m_dl* for the *desired lower temperature*, is within this range $(97 \leq m\_dl \leq 99)$.

Rationale: Exposing the Infant to temperatures lower than 97°F may result in excessive heat loss and drop in heart rate secondary to metabolic acidosis.

ENV8	The Lower Desired Temperature will always be less than or equal to the Upper Desired Temperature of −1°F.	The monitored variable *m_dl* for the *desired lower temperature* will be, $m\_dl \leq m\_dh$.

Rationale: If the Lower Desired Temperature is greater than or equal to the Upper Desired Temperature, may result in excessive cycling of the Heat Source.

ENV9	The desired higher temperature set by the nurse is an integer in the range of 98..100.	The monitored variable *m_dh* for the *desired higher temperature*, is within this range $(98 \leq m\_dh \leq 100)$.

Rationale: Exposing the Infant to temperatures greater than 100°F may result in an incorrect diagnosis of fever resulting in aggressive evaluation (blood culture and lumbar puncture) and treatment for infection.

ENV10	The higher alarm temperature set by the nurse is an integer in the range of 99..103.	The monitored variable *m_ah* for the *higher alarm temperature*, is within this range $(99 \leq m\_ah \leq 103)$.

Rationale: Exposure to temperatures greater than 103°F will result in hyperthermia, which can lead to cardiac arrhythmias and febrile seizures within a few minutes.

ENV11	The Upper Alarm Temperature will always be greater than or equal to the Upper Desired Temperature of 1°F.	The monitored variable *m_ah* for the *higher alarm temperature* will be, $m\_ah \geq m\_dh$.

Rationale: If the Upper Alarm Temperature is less than or equal to the Upper Desired Temperature, the Alarm could be activated while the Current Temperature is still in the Desired Temperature Range.

ENV12	The Display Temperature will cover the range of at least 68.0° to 105.0°F.	The controlled variable *c_td* for the *display temperature*, is within this range $(68.0 \leq m\_dh \leq 105.0)$.

Rationale: This is the specified range of operation of the Isolette. The lower end of this range is useful for monitoring an Isolette that is warming to the Desired Temperature Range. The upper end is set to be greater than the maximum Upper Alarm Temperature.

C. R-descriptions

REQ8	Once in the *failed* mode, the isolette can go back to *normal* mode in one of the following ways: • the sensor becomes valid • the nurse turns off the isoltte	Refer to table 10, and figure 4

Rationale: The isolette may return to *normal* mode of operation provided that the conditions signaling the failure are removed.

REQ9	In the *init* mode: • the isolettes switch is on • the heat control shall be on provided that: 1. the operator enters valid settings 2. the sensor is valid • the displayed temperature is 0°F • the alarm shall be off	Refer to table 5, REQ-MA-1 from [1], and table 8

Rationale: The baby is place in the isolette in *normal* mode, hence when the isolette is in *init* mode, the alarm is disabled, and the heat control is turned on provided that the operator set temperatures are valid, and no equipment failure conditions are present.

REQ10	Once in the *init* mode, the isolette shall switch to *normal* mode, provided: • the operator enters valid settings and the sensor is valid	The controlled variable c_md = *normal* when the operator enters valid temperature ranges ($m\_al < m\_dl < m\_dh < m\_ah$) , and the sensor is valid (m_st=*valid*)

Rationale: The baby is placed into the isolette when the isolette is in the normal mode of operation. Operator errors are possible. Exposure to temperatures inconsiderate of the babys health and weight will result in several complications.