

Module Interface Specification for Solar Water Heating Systems Incorporating Phase Change Material

Brooks MacLachlan and Spencer Smith

November 9, 2017

Contents

1	Introduction	4
2	Notation	4
3	Module Decomposition	5
4	MIS of Control Module	6
4.1	Module	6
4.2	Uses	6
4.3	Syntax	6
4.3.1	Exported Access Programs	6
4.4	Semantics	6
4.4.1	State Variables	6
4.4.2	Access Routine Semantics	6
5	MIS of Input Parameters Module	8
5.1	Module	8
5.2	Uses	8
5.3	Syntax	8
5.4	Semantics	9
5.4.1	Environment Variables	9
5.4.2	State Variables	9

5.4.3	Assumptions	10
5.4.4	Access Routine Semantics	10
5.5	Considerations	12
6	MIS of Temperature ODEs Module	13
6.1	Module	13
6.2	Uses	13
6.3	Syntax	13
6.3.1	Exported Access Programs	13
6.4	Semantics	13
6.4.1	State Variables	13
6.4.2	Assumptions	13
6.4.3	Access Routine Semantics	13
7	MIS of ODE Solver Module	15
7.1	Template Module	15
7.2	Uses	15
7.3	Syntax	15
7.3.1	Exported Access Programs	15
7.4	Semantics	15
7.4.1	State Variables	15
7.4.2	Access Routine Semantics	15
8	MIS of Energy Module	17
8.1	Module	17
8.2	Uses	17
8.3	Syntax	17
8.3.1	External Access Programs	17
8.4	Semantics	17
8.4.1	State Variables	17
8.4.2	Assumptions	17
8.4.3	Access Routine Semantics	17
9	MIS of Output Verification Module	19
9.1	Module	19
9.2	Uses	19
9.3	Syntax	19
9.3.1	Exported Access Programs	19

9.4	Semantics	19
9.4.1	State Variables	19
9.4.2	Environment Variables	19
9.4.3	Local Variables	19
9.4.4	Assumptions	19
9.4.5	Access Routine Semantics	20
9.4.6	Local Functions	20
10	MIS of Plotting Module	22
10.1	Module	22
10.2	Uses	22
10.3	Syntax	22
10.3.1	Exported Access Programs	22
10.4	Semantics	22
10.4.1	State Variables	22
10.4.2	Environment Variables	22
10.4.3	Assumptions	22
10.4.4	Access Routine Semantics	23
11	MIS of Output Module	24
11.1	Module	24
11.2	Uses	24
11.3	Syntax	24
11.3.1	Exported Constants	24
11.3.2	Exported Access Program	24
11.4	Semantics	24
11.4.1	State Variables	24
11.4.2	Environment Variables	24
11.4.3	Access Routine Semantics	25
12	MIS of Specification Parameters	26
12.1	Module	26
12.2	Uses	26
12.3	Syntax	26
12.4	Semantics	26
12.4.1	State Variables	26
12.4.2	Assumptions	27
12.4.3	Access Routine Semantics	27

13 MIS of Types	29
13.1 Template Module	29
13.2 Uses	29
13.3 Exported Types	29
14 Appendix	30

1 Introduction

The following document details the Module Interface Specifications for the implemented modules in a program simulating a Solar Water Heating System with Phase Change Material. It is intended to ease navigation through the program for design and maintenance purposes.

Complementary documents include the System Requirement Specifications and Module Guide.

2 Notation

The structure of the MIS for modules comes from [Hoffman and Strooper \(1995\)](#), with the addition that template modules have been adapted from [Ghezzi et al. \(2003\)](#). The mathematical notation comes from Chapter 3 of [Hoffman and Strooper \(1995\)](#). For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SWHS.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of SWHS uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type.

Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SWHS uses functions, which are defined by the data types of their inputs and outputs. Functions are described by showing their input data types separated by multiplication symbols on the left side of an arrow, and their output data type on the right side.

3 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Parameters Module Output Format Module Output Verification Module Temperature ODEs Module Energy Equations Module Control Module Specification Parameters Module
Software Decision Module	Sequence Data Structure Module ODE Solver Module Plotting Module

Table 1: Module Hierarchy

4 MIS of Control Module

4.1 Module

main

4.2 Uses

parameters (Section 5), load_params (Section ??), verify_params (Section ??), temperature (Section 6), ODE Solvers Module (Section 7), energy (Section 8), verify_output (Section 9), plot (Section 10), output (Section 11)

4.3 Syntax

4.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

4.4 Semantics

4.4.1 State Variables

None

4.4.2 Access Routine Semantics

main():

- transition: Modify the state of other modules and their environment variables by following these steps

[Steven – this gives an idea of how function calls will be used to specify the behaviour of the control module —SS]

Get filename: string from user

load_params(filename)

$$t_{\text{melt}}^{\text{init}}, [T_W^{\text{Solid}}, T_P^{\text{Solid}}]^T := \text{solve}(\text{ODE_SolidPCM}, 0.0, [T_{\text{init}}, T_{\text{init}}]^T, \text{event_StartMelt})$$

$$t_{\text{melt}}^{\text{final}}, [T_W^{\text{Melting}}, T_P^{\text{Melting}}, Q_p]^T := \text{solve}(\text{ODE_MeltingPCM}, t_{\text{melt}}^{\text{init}}, [T_W^{\text{Solid}}(t_{\text{melt}}^{\text{init}}), T_P^{\text{Solid}}(t_{\text{melt}}^{\text{init}}), 0.0]^T, \text{event_EndMelt})$$

$$[T_W^{\text{Liquid}}, T_P^{\text{Liquid}}]^T := \text{solveNoE}(\text{ODE_LiquidPCM}, t_{\text{melt}}^{\text{final}}, [T_W^{\text{Melting}}(t_{\text{melt}}^{\text{final}}), T_P^{\text{Melting}}(t_{\text{melt}}^{\text{final}})]^T, t_{\text{final}})$$

$$E_W^{\text{Solid}} := \text{energyWater}(T_W^{\text{Solid}})$$

$$E_W^{\text{Melting}} := \text{energyWater}(T_W^{\text{Melting}})$$

$$E_W^{\text{Liquid}} := \text{energyWater}(T_W^{\text{Liquid}})$$

$$E_P^{\text{Solid}} := \text{energySolidPCM}(T_P^{\text{Solid}})$$

$$E_P^{\text{Melting}} := \text{energyMeltingPCM}(Q_P)$$

$$E_P^{\text{Liquid}} := \text{energyLiquidPCM}(T_P^{\text{Liquid}})$$

etc.

5 MIS of Input Parameters Module

The secrets of this module are the data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

5.1 Module

Param

5.2 Uses

SpecParam (Section 13)

5.3 Syntax

Name	In	Out	Exceptions
load_params	string	-	FileError
verify_params	-	-	badLength, badDiam, badPCMVolume, badPCMAndTankVol, badPCMArea, badPCMDensity, badMeltTemp, badCoilAndInitTemp, badCoilTemp, badPCMHeatCapSolid, badPCMHeatCapLiquid, badHeatFusion, badCoilArea, badWaterDensity, badWaterHeatCap, badCoilCoeff, badPCMCoeff, badInitTemp, badFinalTime, badInitAndMeltTemp
L	-	\mathbb{R}	
D	-	\mathbb{R}	
V_P	-	\mathbb{R}	
A_P	-	\mathbb{R}	
...	
m_W^{noPCM}	-	\mathbb{R}	
τ_W^{noPCM}	-	\mathbb{R}	

Should verify_params have a boolean output?

5.4 Semantics

5.4.1 Environment Variables

inputFile: sequence of string $\#f[i]$ is the i th string in the text file f

5.4.2 State Variables

From R1

$L: \mathbb{R}$

$D: \mathbb{R}$

$V_P: \mathbb{R}$

$A_P: \mathbb{R}$

$\rho_P: \mathbb{R}$

$T_{\text{melt}}^P: \mathbb{R}$

$C_P^S: \mathbb{R}$

$C_P^L: \mathbb{R}$

$H_f: \mathbb{R}$

$A_C: \mathbb{R}$

$T_C: \mathbb{R}$

$\rho_W: \mathbb{R}$

$C_W: \mathbb{R}$

$h_C: \mathbb{R}$

$h_P: \mathbb{R}$

$T_{\text{init}}: \mathbb{R}$

$t_{\text{step}}: \mathbb{R}$

$t_{\text{final}}: \mathbb{R}$

$AbsTol: \mathbb{R}$

$RelTol: \mathbb{R}$

$ConsTol: \mathbb{R}$

From R2

$V_{\text{tank}}: \mathbb{R}$

$m_W: \mathbb{R}$

$m_P: \mathbb{R}$

From R3

$\tau_W: \mathbb{R}$

$\eta: \mathbb{R}$

$\tau_P^S: \mathbb{R}$

$\tau_P^L: \mathbb{R}$

To Support IM4

$E_{P_{\text{melt}}}^{\text{init}}: \mathbb{R}$

$E_{P_{\text{melt}}}^{\text{all}}: \mathbb{R}$

To Support Testing

$m_W^{\text{noPCM}}: \mathbb{R}$

$\tau_W^{\text{noPCM}}: \mathbb{R}$

5.4.3 Assumptions

- `load_params` will be called before the values of any state variables will be accessed.
- The file contains the string equivalents of the numeric values for each input parameter in order, each on a new line. The order is the same as in the table in R1 of the SRS. Any comments in the input file should be denoted with a '#' symbol.

5.4.4 Access Routine Semantics

Param.*L*:

- output: *out* := *L*
- exception: none

Param.*D*:

- output: $out := D$
- exception: none
- ...

Param. m_W^{noPCM} :

- output: $out := m_W^{\text{noPCM}}$
- exception: none

Param. τ_W^{noPCM} :

- output: $out := \tau_W^{\text{noPCM}}$
- exception: none

load_params(s):

- transition: The filename s is first associated with the file f. inputFile is used to modify the state variables using the following procedural specification:

1. Read data sequentially from inputFile to populate the state variables from R1 (L to $ConsTol$).
2. Calculate the derived quantities (all other state variables) as follows:

$$\begin{aligned}
& - V_{\text{tank}} := \pi \times L \times \left(\frac{D}{2}\right)^2 \\
& - m_W := \rho_w (V_t - V_p) \\
& - m_P := \rho_p V_p \\
& - \tau_W := \frac{m_w C_w}{A_c h_c} \\
& - \eta := \frac{h_p A_p}{h_c A_c} \\
& - \tau_P^S := \frac{M_p C_{ps}}{h_p A_p} \\
& - \tau_P^L := \frac{M_p C_{pl}}{h_p A_p} \\
& - E_{P_{\text{melt}}}^{\text{init}} := C_{ps} M_p (T_{\text{melt}} - T_{\text{init}}) \\
& - E_{P_{\text{melt}}}^{\text{all}} := H_f m_p \\
& - m_W^{\text{noPCM}} := \rho_w V_t
\end{aligned}$$

$$- \tau_W^{\text{noPCM}} := \frac{m_W^{\text{noPCM}} C_w}{h_c A_c}$$

3. `verify_params()`

- exception: `exc` := a file name `s` cannot be found OR the format of `inputFile` is incorrect \Rightarrow `FileError`

`verify_params()`:

- out: `out` := none
- exception: `exc` :=

$\neg(L > 0)$	\Rightarrow <code>badLength</code>
$\neg(L_{\min} \leq L \leq L_{\max})$	\Rightarrow <code>warnLength</code>
$\neg(D > 0)$	\Rightarrow <code>badDiam</code>
$\neg(\frac{D}{L}_{\min} \leq \frac{D}{L} \leq \frac{D}{L}_{\max})$	\Rightarrow <code>warnDiam</code>
$\neg(V_P > 0)$	\Rightarrow <code>badPCMVVolume</code>
$\neg(V_P \geq \text{minfrac} \cdot V_{\text{tank}}(D, L))$	\Rightarrow <code>warnPCMVVol</code>
$\neg(V_P < V_{\text{tank}}(D, L))$	\Rightarrow <code>badPCMAndTankVol</code>
$\neg(A_P > 0)$	\Rightarrow <code>badPCMArea</code>
$\neg(V_P \leq A_P \leq \frac{2}{h_{\min}} V_P)$	\Rightarrow <code>warnVolArea</code>
$\neg(\rho_P > 0)$	\Rightarrow <code>badPCMDensity</code>
$\neg(\rho_P^{\min} < \rho_P < \rho_P^{\max})$	\Rightarrow <code>warnPCMDensity</code>

etc. See Appendix (Section 14) for the complete list of exceptions and associated error messages.

5.5 Considerations

The value of each state variable can be accessed through its name (getter). An access program is available for each state variable. There are no setters for the state variables, since the values will be set and checked by load params and not changed for the life of the program.

6 MIS of Temperature ODEs Module

6.1 Module

Temperature

6.2 Uses

Param (Section 5), Types n

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
ODE_SolidPCM	–	systFuncT 2	-
ODE_MeltingPCM	–	systFuncT 3	-
ODE_LiquidPCM	–	systFuncT 2	-
event_StartMelt	–	stateBasedFuncT 2	-
event_EndMelt	–	stateBasedFuncT 3	-

6.4 Semantics

6.4.1 State Variables

none

6.4.2 Assumptions

none

6.4.3 Access Routine Semantics

ODE.SolidPCM():

$$\bullet \text{ output: } out := \frac{d}{dt} \begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W} [(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau_P^S} (T_W(t) - T_P(t)) \end{bmatrix}$$

- exception: none

ODE_MeltingPCM():

- output: $out := \frac{d}{dt} \begin{bmatrix} T_W \\ T_P \\ Q_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ 0 \\ h_P A_P(T_W(t) - T_{\text{melt}}^P) \end{bmatrix}$

- exception: none

ODE_LiquidPCM():

- output: $out := \frac{d}{dt} \begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau_P}(T_W(t) - T_P(t)) \end{bmatrix}$

- exception: none

event_StartMelt():

- output: $out := g([T_W, T_P]^T) = T_{\text{melt}}^P - T_P$

- exception: none

event_EndMelt():

- output: $out := g([T_W, T_P, Q_P]^T) = 1 - \phi$, where $\phi = \frac{Q_P}{E_{P\text{melt}}^{\text{all}}}$

- exception: none

7 MIS of ODE Solver Module

7.1 Template Module

Solver(Types n)

7.2 Uses

Types n

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
solve	$\text{systFuncT } n, \mathbb{R}, \text{stateT } n, \text{stateBasedFuncT } n$	$\mathbb{R}, \text{seqFuncT } n$	ODE_BAD_INPUT, ODE_MAXSTEP, ODE_ACCURACY
solve (alternative notation)	$(\mathbb{R}^n \rightarrow \mathbb{R})^n, \mathbb{R}, \mathbb{R}^n, \mathbb{R}^n \rightarrow \mathbb{R}$	$\mathbb{R}, (\mathbb{R} \rightarrow \mathbb{R})^n$	ODE_BAD_INPUT, ODE_MAXSTEP, ODE_ACCURACY
solveNoE	$\text{systFuncT } n, \mathbb{R}, \text{stateT } n, \mathbb{R}$	$\text{seqFuncT } n$	ODE_BAD_INPUT, ODE_MAXSTEP, ODE_ACCURACY

Need to add final times

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Access Routine Semantics

$\text{solve}(f, t_0, y_0, g):$

- output: $out := t_1, y(t)$ where

$$y(t) = y_0 + \int_{t_0}^t f(y(s))ds$$

with t_1 determined by the first time where $g(y(t_1)) = 0$. $y(t)$ is calculated from $t = t_o$ to $t = t_1$.

- exception: $exc :=$ (Invalid input parameters \Rightarrow ODE_BAD_INPUT | $MaxStep$ steps taken and no solution found \Rightarrow ODE_MAXSTEP | $reltol$ and $abstol$ not satisfied for a step \Rightarrow ODE_ACCURACY)

$solve(f, t_0, y_0, t_{fin})$:

- output: $out := t_1, y(t)$ where

$$y(t) = y_0 + \int_{t_0}^{t_{fin}} f(y(s))ds$$

$y(t)$ is calculated from $t = t_o$ to $t = t_{fin}$.

- exception: $exc :=$ (Invalid input parameters \Rightarrow ODE_BAD_INPUT | $MaxStep$ steps taken and no solution found \Rightarrow ODE_MAXSTEP | $reltol$ and $abstol$ not satisfied for a step \Rightarrow ODE_ACCURACY)

8 MIS of Energy Module

8.1 Module

Energy

8.2 Uses

Param (Section 5)

8.3 Syntax

8.3.1 External Access Programs

Name	In	Out	Exceptions
energyWater	functT	functT	-
energySolidPCM	functT	functT	-
energyMeltingPCM	functT	functT	-
energyLiquidPCM	functT	functT	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Assumptions

None

8.4.3 Access Routine Semantics

energyWater(T_W):

- output: $out := C_W m_W (T_W - T_{\text{init}})$
- exception: none

energySolidPCM(T_P):

- output: $out := C_P^S m_P (T_P - T_{\text{init}})$
- exception: none

energyMeltingPCM(Q_P):

- output: $out := E_{P_{\text{melt}}}^{\text{init}} + Q_P$
- exception: none

energyLiquidPCM(T_P):

- output: $out := E_{P_{\text{melt}}}^{\text{init}} + H_f m_p + C_P^L m_P (T_P(t) - T_{\text{melt}}^P)$
- exception: none

9 MIS of Output Verification Module

9.1 Module

verify_output

9.2 Uses

Param (Section 5)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_output	array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , parameters	-	-

9.4 Semantics

9.4.1 State Variables

expEPCM: array of \mathbb{R}

expEWat: array of \mathbb{R}

errorWater: \mathbb{R}

errorPCM: \mathbb{R}

9.4.2 Environment Variables

win: 2D array of pixels displayed on the screen

9.4.3 Local Variables

9.4.4 Assumptions

All of the fields of the input parameters structure have been assigned a value. The values have been properly constrained. The input arrays are not empty.

9.4.5 Access Routine Semantics

$\text{verify_output}(t, Tw, Tp, Ew, Ep, \text{params})$: transition: $\text{expEPCM}, \quad \text{expEWat},$
 $\text{errorWater}, \quad \text{errorPCM},$
 $\text{win} := (\forall i \in [1..|t| - 1])$
 $(\text{expectedEp}(\text{traprule}(\text{delta}(t[i - 1], t[i]), Tw[i], Tp[i],$
 $Tw[i - 1], Tp[i - 1]), \text{params}))),$
 $(\forall i \in [1..|t| - 1]) (\text{expectedEw}$
 $(\text{expectedEc}(\text{traprule}(\text{delta}(t[i - 1], t[i]),$
 $\text{params.Tc}, Tw[i], \text{params.Tc}, Tw[i - 1]),$
 $\text{params}), \text{post}(\text{expEPCM}))),$
 $\text{error}(\text{sum}(\text{post}(\text{expEWat})),$
 $Ew[|Ew| - 1]),$
 $\text{error}(\text{sum}(\text{post}(\text{expEPCM})),$
 $Ep[|Ep| - 1]), (\text{errorWater} >$
 $\text{ConsTol} \vee \text{errorPCM} >$
 $\text{ConsTol} \Rightarrow \text{Prints warning}$
 $\text{message(s)})$

exception: $(\text{errorWater} > \text{ConsTol} \Rightarrow$
 $\text{warnWaterError} \mid \text{errorPCM} >$
 $\text{ConsTol} \Rightarrow \text{warnPCMError})$
These exceptions do not terminate the program.

9.4.6 Local Functions

$\text{delta}: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

$\text{delta}(t1, t2) \equiv t2 - t1$

$\text{traprule}: \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

$\text{traprule}(t, A1, B1, A2, B2) \equiv t \times (A1 - B1 + A2 - B2)/2$

$\text{expectedEc}: \mathbb{R} \times \text{parameters} \rightarrow \mathbb{R}$

$\text{expectedEc}(c, \text{params}) \equiv \text{params.hc} \times \text{params.Ac} \times c$

$\text{expectedEp}: \mathbb{R} \times \text{parameters} \rightarrow \mathbb{R}$
 $\text{expectedEp}(p, \text{params}) \equiv \text{params.hp} \times \text{params.Ap} \times p$

$\text{expectedEw}: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
 $\text{expectedEw}(Ec, Ep) \equiv Ec - Ep$

$\text{sum}: \text{array of } \mathbb{R}\text{s} \rightarrow \mathbb{R}$
 $\text{sum}(a) \equiv \sum_{i=0}^{|a|-1} a[i]$

$\text{error}: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
 $\text{error}(\text{exp}, \text{act}) \equiv \frac{|\text{exp} - \text{act}|}{\text{act}} \times 100$

10 MIS of Plotting Module

10.1 Module

plot

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , string	-	-

10.4 Semantics

10.4.1 State Variables

plotFilename: string

10.4.2 Environment Variables

directory: The current directory of files from which the program is run.

10.4.3 Assumptions

The input arrays are all of the same size.

10.4.4 Access Routine Semantics

`plot(t , Tw , Tp , Tw , Ep , $filename$)`: transition: *directory*: writes a .png file
named *plotFilename* containing
the graphs of the simulation re-
sults.
exception: none

11 MIS of Output Module

11.1 Module

output

11.2 Uses

Param (Section 5)

11.3 Syntax

11.3.1 Exported Constants

max_width: integer

11.3.2 Exported Access Program

Name	In	Out	Exceptions
output	string, array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , array of \mathbb{R} , parameters	-	-

11.4 Semantics

11.4.1 State Variables

outFilename: string

11.4.2 Environment Variables

directory: The current directory of files from which the program is run.

11.4.3 Access Routine Semantics

output(*params*, *t*, *Tw*, *Tp*, *Ew*, *Ep*, *ETot*, *filename*): transition: *directory*: writes
a .txt file named
outFilename con-
taining the input
parameters, calcu-
lated parameters,
and results of the
simulation.

exception: none

References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995.

12 MIS of Specification Parameters

The secrets of this module is the value of the specification parameters.

12.1 Module

SpecParam

12.2 Uses

N/A

12.3 Syntax

Name	In	Out	Exceptions
L_{\min}	-	\mathbb{R}	
L_{\max}	-	\mathbb{R}	
...	
t_{final}^{\max}	-	\mathbb{R}	

12.4 Semantics

12.4.1 State Variables

From Table 2 in SRS

$L_{\min} := 0.1$

$L_{\max} := 50$

$\frac{D}{L}_{\min} := 0.002$

$\frac{D}{L}_{\max} := 200$

$\text{minfrac} := 10^{-6}$

$h_{\min} := 0.001$

$\rho_P^{\min} := 500$

$\rho_P^{\max} := 20000$

$C_{P_{\min}}^S := 100$
 $C_{P_{\max}}^S := 4000$
 $C_{P_{\min}}^L := 100$
 $C_{P_{\max}}^L := 5000$
 $A_C^{\max} := \pi(\frac{D}{2})^2$
 $\rho_W^{\min} := 950$
 $\rho_W^{\max} := 1000$
 $C_W^{\min} := 4170$
 $C_W^{\max} := 4210$
 $h_C^{\min} := 10$
 $h_C^{\max} := 10000$
 $h_P^{\min} := 10$
 $h_P^{\max} := 10000$
 $t_{\text{final}}^{\max} := 86400$

#A_C^{\max} shouldn't be in this table of constants

12.4.2 Assumptions

None

12.4.3 Access Routine Semantics

SpecParam. L_{\min} :

- output: $out := L_{\min}$
- exception: none

SpecParam. L_{\max} :

- output: $out := L_{\max}$
- exception: none

...

SpecParam. t_{final}^{\max} :

- output: $out := t_{\text{final}}^{\text{max}}$
- exception: none

13 MIS of Types

This modules provides types that are used throughout the specification.

13.1 Template Module

Types($n : \mathbb{N}$)

13.2 Uses

N/A

13.3 Exported Types

$\text{functT} = \mathbb{R} \rightarrow \mathbb{R}$

$\text{seqFuncT } n = (\text{functT})^n = (\mathbb{R} \rightarrow \mathbb{R})^n$

$\text{stateT } n = \mathbb{R}^n$

$\text{stateBasedFuncT } n = \text{stateT } n \rightarrow \mathbb{R} = \mathbb{R}^n \rightarrow \mathbb{R}$

$\text{systFuncT } n = (\text{stateBasedFuncT } n)^n = (\mathbb{R}^n \rightarrow \mathbb{R})^n$

14 Appendix

Table 2: Possible Exceptions

Message ID	Error Message
badLength	Error: Tank length must be > 0
badDiam	Error: Tank diameter must be > 0
badPCMVolume	Error: PCM volume must be > 0
badPCMAndTankVol	Error: PCM volume must be $<$ tank volume
badPCMArea	Error: PCM area must be > 0
badPCMDensity	Error: ρ_p must be > 0
badMeltTemp	Error: T_{melt} must be > 0 and $< T_c$
badCoilAndInitTemp	Error: T_c must be $> T_{init}$
badCoilTemp	Error: T_c must be > 0 and < 100
badPCMHeatCapSolid	Error: C_{ps} must be > 0
badPCMHeatCapLiquid	Error: C_{pl} must be > 0
badHeatFusion	Error: H_f must be > 0
badCoilArea	Error: A_c must be > 0
badWaterDensity	Error: ρ_w must be > 0
badWaterHeatCap	Error: C_w must be > 0
badCoilCoeff	Error: h_c must be > 0
badPCMCoeff	Error: h_p must be > 0
badInitTemp	Error: T_{init} must be > 0 and < 100
badFinalTime	Error: t_{final} must be > 0
badInitAndMeltTemp	Error: T_{init} must be $< T_{melt}$
ODE_ACCURACY	<i>reltol</i> and <i>abstol</i> were not satisfied by the ODE solver for a given solution step.
ODE_BAD_INPUT	Invalid input to ODE solver
ODE_MAXSTEP	ODE solver took <i>MaxStep</i> steps and did not find solution
warnLength	Warning: It is recommended that $0.1 \leq L \leq 50$
warnDiam	Warning: It is recommended that $0.002 \leq D/L \leq 200$

warnPCMVOL	Warning: It is recommended that V_p be $\geq 0.0001\%$ of V_t
warnVolArea	Warning: It is recommended that $V_p \leq A_p \leq (2/0.001) * V_p$
warnPCMDensity	Warning: It is recommended that $500 < \rho_p < 20000$
warnPCMHeatCapSolid	Warning: It is recommended that $100 < C_{ps} < 4000$
warnPCMHeatCapLiquid	Warning: It is recommended that $100 < C_{pl} < 5000$
warnCoilArea	Warning: It is recommended that $A_c \leq \pi * (D/2) \wedge 2$
warnWaterDensity	Warning: It is recommended that $950 < \rho_w \leq 1000$
warnWaterHeatCap	Warning: It is recommended that $4170 < C_w < 4210$
warnCoilCoeff	Warning: It is recommended that $10 < h_c < 10000$
warnPCMCoeff	Warning: It is recommended that $10 < h_p < 10000$
warnFinalTime	Warning: It is recommended that $0 < t_{final} < 86400$
warnWaterError	Warning: There is greater than $x\%$ relative error between the energy in the water output and the expected output based on the law of conservation of energy. (Where x is the value of <i>ConsTol</i>)
warnPCMError	Warning: There is greater than $x\%$ relative error between the energy in the PCM output and the expected output based on the law of conservation of energy. (Where x is the value of <i>ConsTol</i>)
