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

Brooks MacLachlan and Spencer Smith

November 8, 2016

Contents

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 $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

??? Have to Revise ??? 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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

SWHS also uses some derived data types: arrays, strings, and structures. Arrays are lists filled with elements of the same data type. Strings are arrays of characters. Structures contain pairs of keys and values, where keys are unique variable names used to identify their corresponding value, and values can be of any data type. 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 Format Module Input Parameters Module Input Verification Module Output Format Module Output Verification Module Temperature ODEs Module Energy Equations Module Control 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 (5), load_params (6), verify_params (7), temperature (8), ODE Solvers Module (9), energy (10), verify_output (11), plot (12), output (13)

4.3 Syntax

4.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	string	-	-

4.4 Semantics

4.4.1 State Variables

time: array of reals
tempW: array of reals
tempP: array of reals
latHeat: array of reals

eW: array of reals
eP: array of reals
$eTot$: array of reals

4.4.2 Environment Variables

win: 2D array of pixels displayed on the screen

4.4.3 Access Routine Semantics

main(s):	transition:	$time$, $tempW$, $tempP$, $latHeat$, eW, eP, $eTot$, win := $results[0]$, $results[1]$, $results[2]$, $results[3]$, $eW1 \| eW2 \| eW3$, $eP1 \| eP2 \| eP3$, $(\forall i \in [0..\|post(eW)\| - 1])$ $(post(eW[i]) + post(eP[i]))$, Prints information about the melting of PCM.
	exception:	none

5 MIS of Input Parameters Module

5.1 Template Module

parameters

5.2 Uses

N/A

5.3 Syntax

5.3.1 Exported Data Types

paramT = ?

5.3.2 Exported Access Programs

Name	In	Out	Exceptions
new paramT	-	paramT	-
getL	-	real	-
get_diam	-	real	-
getVp	-	real	-
...	-	real	-
get_tau_w_noPCM	-	real	-
setL	real	-	-
set_diam	real	-	-
setVp	real	-	-
...	real	-	-
set_tau_w_noPCM	real	-	-

5.3.3 Assumptions

An instance of paramT will not be used before it has been initialized.

5.4 Semantics

5.4.1 State Variables

L: real
diam: real
Vp: real
Ap: real
rho_p: real
Tmelt: real
C_ps: real
C_pl: real
Hf: real
Ac: real
Tc: real
rho_w: real
C_w: real
hc: real
hp: real
Tinit: real
tstep: real
tfinal: real
AbsTol: real
RelTol: real

ConsTol: real
Vt: real
Mw: real
tau_w: real
eta: real
Mp: real
tau_ps: real
tau_pl: real
Epmelt_init: real
Ep_melt3: real
Mw_noPCM: real
tau_w_no_PCM: real

5.4.2 Access Routine Semantics

new paramT():

- transition: L, diam, Vp, Ap, ..., tau_w_no_PCM := 0, 0, 0, 0, ..., 0
- output: *out* := self
- exception: none

getL():

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

get_diam():

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

getVp():

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

getAp():

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

...
get_tau_w_no_PCM():

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

setL(x):

- transition: L := x
- exception: none

set_diam(x):

- transition: diam := x
- exception: none

setVp(x):

- transition: Vp := x
- exception: none

setAp(x):

- transition: Ap := x
- exception: none

...

set_tau_w_no_PCM(x):

- transition: tau_w_no_PCM := x
- exception: none

5.5 Considerations

As implied above, there is a getter and a setter for each state variable. All state variables are initially initialized to zero. Some programming language implementations will likely use the dot notation to access the fields of this template module, rather than actually write out all of the getters and setters.

6 MIS of Input Format Module

6.1 Module

load_params

6.2 Uses

parameters (5)

6.3 Syntax

6.4 Exported Access Programs

Name	In	Out	Exceptions
load_params	string	parameters	-

6.5 Semantics

6.5.1 State Variables

params: parameters
param: array of reals

6.5.2 Assumptions

The input string corresponds to an existing filename in the current directory. The input file is formatted correctly. It should contain 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 should be denoted with a '#' symbol.

6.5.3 Access Routine Semantics

load_params(s): transition: $params.L$, $params.diam$, $params.Vp$, $params.Ap$, $params.rho\_p$, $params.Tmelt$, $params.C\_ps$, $params.C\_pl$, $params.Hf$, $params.Ac$, $params.Tc$, $params.rho\_w$, $params.C\_w$, $params.hc$, $params.hp$, $params.Tinit$, $params.tstep$, $params.tfinal$, $params.AbsTol$, $params.RelTol$, $params.ConsTol$, $params.Vt$, $params.Mw$, $params.tau\_w$, $params.eta$, $params.Mp$, $params.tau\_ps$, $params.tau\_pl$, $params.Epmelt\_init$, $params.Ep\_melt3$, $params.Mw\_noPCM$, $params.tau\_w\_noPCM$:= $param$[0], $param$[1], $param$[2], $param$[3], $param$[4], $param$[5], $param$[6], $param$[7], $param$[8], $param$[9], $param$[10], $param$[11], $param$[12], $param$[13], $param$[14], $param$[15], $param$[16], $param$[17], $param$[18], $param$[19], $param$[20], calcVt(post($params.L$), post($params.diam$)), calcMw(post($params.Vp$), post($params.rho\_w$), post($params.Vt$)), calcTauw(post($params.Mw$), post($params.C\_w$), post($params.hc$), post($params.Ac$)), calcEta(post($params.hp$), post($params.Ap$), post($params.hc$), post($params.Ac$)), calcMp(post($params.rho\_p$). post($params.Vp$)), calcTaups(post($params.Mp$), post($params.C\_ps$), post($params.hp$), post($params.Ap$)), calcTaupl(post($params.Mp$), post($params.C\_pl$), post($params.hp$), post($params.Ap$)), calcEpmeltinit(post($params.C\_ps$), post($params.Mp$), post($params.Tmelt$), post($params.Tinit$)), calcEpmelt3(post($params.Hf$), post($params.Mp$)), calcMwnoPCM(post($params.rho\_w$), post($params.Vt$)), calcTauwnoPCM(post($params.Mw\_noPCM$), post($params.C\_w$), post($params.hc$), post($params.Ac$)), where $param$ is the array of parameters obtained from the input file s

exception: none

6.5.4 Local Functions

calcVt: real × real → real
calcVt(L, $diam$) ≡ $\pi \times L \times (\frac{diam}{2})^2$

calcMw: real × real × real → real
calcMw(Vp, rho_w, Vt) ≡ $rho\_w \times (Vt - Vp)$

calcTauw: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcTauw(Mw, C_w, hc, Ac) $\equiv \frac{Mw \times C\_w}{Ac \times hc}$

calcEta: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcEta(hp, Ap, hc, Ac) $\equiv \frac{hp \times Ap}{hc \times Ac}$

calcMp: real $\times$ real $\rightarrow$ real
calcMp(rho_p, Vp) $\equiv rho\_p \times Vp$

calcTaups: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcTaups(Mp, C_ps, hp, Ap) $\equiv \frac{Mp \times C\_ps}{hp \times Ap}$

calcTaupl: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcTaupl(Mp, C_pl, hp, Ap) $\equiv \frac{Mp \times C\_pl}{hp \times Ap}$

calcEpmeltinit: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcEpmeltinit(C_ps, Mp, $Tmelt$, $Tinit$) $\equiv C\_ps \times Mp \times (Tmelt - Tinit)$

calcEpmelt3: real $\times$ real $\rightarrow$ real
calcEpmelt3(Hf, Mp) $\equiv Hf \times Mp$

calcMwnoPCM: real $\times$ real $\rightarrow$ real
calcMwnoPCM(rho_w, Vt) $\equiv rho\_w \times Vt$

calcTauwnoPCM: real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
calcTauwnoPCM(Mw_noPCM, C_w, hc, Ac) $equiv$ $\frac{Mw\_noPCM \times C\_w}{hc \times Ac}$

7 MIS of Input Verification Module

7.1 Module

verify_params

7.2 Uses

parameters (5)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_valid	parameters	-	badLength, badDiam, badPCMVolume, badPCMAndTankVol, badPCMArea, badPCMDensity, badMeltTemp, badCoilAndInitTemp, badCoilTemp, badPCMHeatCapSolid, badPCMHeatCapLiquid, badHeatFusion, badCoilArea, badWaterDensity, badWaterHeatCap, badCoilCoeff, badPCMCoeff, badInitTemp, badFinalTime, badInitAndMeltTemp
verify_recommended	parameters	-	-

7.4 Semantics

7.4.1 Environment Variables

win: 2D array of pixels displayed on the screen.

7.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value.

7.4.3 Access Routine Semantics

verify_valid(*params*):	transition:	*win*: (error is thrown ⇒ Prints error message)
	exceptions:	*exc* := (*params.L* ≤ 0 ⇒ badLength | *params.diam* ≤ 0 ⇒ badDiam | *params.Vp* ≤ 0 ⇒ badPCMVolume | *params.Vp* ≥ *params.Vt* ⇒ badPCMAndTankVol | *params.Ap* ≤ 0 ⇒ badPCMArea | *params.rho_p* ≤ 0 ⇒ badPCMDensity | *params.Tmelt* ≤ 0 ∨ *params.Tmelt* ≥ *params.Tc* ⇒ badMeltTemp | *params.Tc* ≤ *params.Tinit* ⇒ badCoilAndInitTemp | *params.Tc* ≥ 100 ∨ *params.Tc* ≤ 0 ⇒ badCoilTemp | *params.C_ps* ≤ 0 ⇒ badPCMHeatCapSolid | *params.C_pl* ≤ 0 ⇒ badPCMHeatCapLiquid | *params.Hf* ≤ 0 ⇒ badHeatFusion | *params.Ac* ≤ 0 ⇒ badCoilArea | *params.rho_w* ≤ 0 ⇒ badWaterDensity | *params.C_w* ≤ 0 ⇒ badWaterHeatCap | *params.hc* ≤ 0 ⇒ badCoilCoeff | *params.hp* ≤ 0 ⇒ badPCMCoeff | *params.Tinit* ≤ 0 ∨ *params.Tinit* ≥ 100 ⇒ badInitTemp | *params.tfinal* ≤ 0 ⇒ badFinalTime | *params.Tinit* ≥ *params.Tmelt* ⇒ badInitAndMeltTemp) See Appendix (14) for the complete list of exceptions and associated error messages.

verify_recommended(*params*):	transition:	*win*: (Warning is thrown ⇒ Prints warning message)
	exceptions:	$exc := (params.L < 0.1 \vee params.L > 50 \Rightarrow$ warnLength | $params.diam/params.L < 0.002 \vee params.diam/params.L > 200 \Rightarrow$ warnDiam | $params.Vp < params.Vt \times 10^{-}6 \Rightarrow$ warnPCMVol | $params.Vp > params.Ap \vee params.Ap > (2/0.001) \times params.Vp \Rightarrow$ warnVolArea | $params.rho\_p \leq 500 \vee params.rho\_p \geq 20000 \Rightarrow$ warnPCMDensity | $params.C\_ps \leq 100 \vee params.C\_ps \geq 4000 \Rightarrow$ warnPCMHeatCapSolid | $params.C\_pl \leq 100 \vee params.C\_pl \geq 5000 \Rightarrow$ warnPCMHeatCapLiquid | $params.Ac > \pi \times (params.diam/2)^2 \Rightarrow$ warnCoilArea | $params.rho\_w \leq 950 \vee params.rho\_w > 1000 \Rightarrow$ warnWaterDensity | $params.C\_w \leq 4170 \vee params.C\_w \geq 4210 \Rightarrow$ warnWaterHeatCap | $params.hc \leq 10 \vee params.hc \geq 10000 \Rightarrow$ warnCoilCoeff | $params.hp \leq 10 \vee params.hp \geq 10000 \Rightarrow$ warnPCMCoeff | $params.tfinal \leq 0 \vee params.tfinal \geq 86400 \Rightarrow$ warnFinalTime) None of these exceptions terminate the program. See Appendix (14) for the complete list of exceptions and associated warning messages.

8 MIS of Temperature ODEs Module

8.1 Module

temperature

8.2 Uses

parameters (5)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
temperature1	array of reals, array of reals, array of reals, parameters	array of functions	-
temperature2	array of reals, array of reals, array of reals, array of reals, parameters	array of functions	-
temperature3	array of reals, array of reals, array of reals, parameters	array of functions	-
event1	array of reals, array of reals, array of reals, parameters	function	-
event2	array of reals, array of reals, array of reals, array of reals, parameters	function	-

8.4 Semantics

8.4.1 State Variables

t: array of reals
$Tw1$: array of reals
$Tw2$: array of reals
$Tw3$: array of reals
$Tp1$: array of reals
$Tp2$: array of reals
$Tp3$: array of reals
$Qp2$: array of reals

8.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value. The values have been properly constrained.

8.4.3 Access Routine Semantics

temperature1(t, $Tw1$, $Tp1$, $params$):	output:	$out := \{dTw : real \times real \times real \rightarrow real, dTp : real \times real \times real \rightarrow real\}$
	exception:	none
temperature2(t, $Tw2$, $Tp2$, $Qp2$, $params$):	output:	$out := \{dTw : real \times real \times real \times real \rightarrow real, dTp : real \times real \times real \times real \rightarrow real, dQp : real \times real \times real \times real \rightarrow real\}$
	exception:	none
temperature3(t, $Tw3$, $Tp3$, $params$):	output:	$out := \{dTw : real \times real \times real \rightarrow real, dTp : real \times real \times real \rightarrow real\}$
	exception:	none
event1(t, $Tw1$, $Tp1$, $params$):	output:	$out := Ev : real \times real \times real \rightarrow real$
	exception:	none
event2(t, $Tw2$, $T2p$, $Qp2$, $params$):	output:	$out := Ev : real \times real \times real \times real \rightarrow real$
	exception:	none

9 MIS of ODE Solver Module

9.1 Module

ODE Solver Module

9.2 Uses

N/A

9.3 Syntax

9.3.1 Exported Constants

$MaxStep$: natural number
N: natural number

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
solve	function, array of reals, array of reals, function, real, real	array of reals (N of them)	ODE_BAD_INPUT, ODE_MAXSTEP, ODE_ACCURACY

9.4 Semantics

9.4.1 State Variables

results: array of reals (N of them)

9.4.2 Access Routine Semantics

solve(f, *domain*, *ics*, *events*, *abstol*, *reltol*)

output: *out* := *results*, where *results* holds the solution to the ODE system generated by the solver.

exceptions: *exc* := (Invalid input parameters ⇒ ODE_BAD_INPUT | *MaxStep* steps taken and no solution found ⇒ ODE_MAXSTEP | *reltol* and *abstol* not satisfied for a step ⇒ ODE_ACCURACY)

10 MIS of Energy Module

10.1 Module

energy

10.2 Uses

parameters (5)

10.3 Syntax

10.3.1 External Access Programs

Name	In	Out	Exceptions
energy1Wat	array of reals, parameters	array of reals	-
energy1PCM	array of reals, parameters	array of reals	-
energy2Wat	array of reals, parameters	array of reals	-
energy2PCM	array of reals, parameters	array of reals	-
energy3Wat	array of reals, parameters	array of reals	-
energy3PCM	array of reals, parameters	array of reals	-

10.4 Semantics

10.4.1 State Variables

$eW1$: array of reals
$eP1$: array of reals
$eW2$: array of reals
$eP2$: array of reals
$eW3$: array of reals
$eP3$: array of reals

10.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value. The values have been properly constrained.

10.4.3 Access Routine Semantics

energy1Wat($Tw1$, $params$):	transition:	$(\forall i \in [0..\lvert Tw1\rvert - 1])$ $(eW1[i] :=$ watEnergy($Tw1[i]$, $params$))
	output:	$out := eW1$
	exception:	none
energy1PCM($Tp1$, $params$):	transition:	$(\forall i \in [0..\lvert Tp1\rvert - 1])$ $(eP1[i] :=$ pcmEnergy1($Tp1[i]$, $params$))
	output:	$out := eP1$
	exception:	none
energy2Wat($Tw2$, $params$):	transition:	$(\forall i \in [0..\lvert Tw2\rvert - 1])$ $(eW2[i] :=$ watEnergy($Tw2[i]$, $params$))
	output:	$out := eW2$
	exception:	none
energy2PCM($Qp2$, $params$):	transition:	$(\forall i \in [0..\lvert Qp2\rvert - 1])$ $(eP2[i] :=$ pcmEnergy2($Qp2[i]$, $params$))
	output:	$out := eP2$
	exception:	none
energy3Wat($Tw3$, $params$):	transition:	$(\forall i \in [0..\lvert Tw3\rvert - 1])$ $(eW3[i] :=$ watEnergy($Tw3[i]$, $params$))
	output:	$out := eW3$
	exception:	none
energy3PCM($Tp3$, $params$):	transition:	$(\forall i \in [0..\lvert Tp3\rvert - 1])$ $(eP3[i] :=$ pcmEnergy3($Tp3[i]$, $params$))
	output:	$out := eP3$
	exception:	none

10.4.4 Local Functions

watEnergy: real $\times$ parameters $\rightarrow$ real
watEnergy(Tw, $params$) $\equiv params.C\_w \times params.Mw \times (Tw - params.Tinit)$

pcmEnergy1: real $\times$ parameters $\rightarrow$ real
pcmEnergy1(Tp, $params$) $\equiv params.C\_ps \times params.Mp \times (Tp - params.Tinit)$

pcmEnergy2: real $\times$ parameters $\rightarrow$ real
pcmEnergy2(Qp, $params$) $\equiv params.Epmelt\_init + Qp$

pcmEnergy3: real $\times$ parameters $\rightarrow$ real

pcmEnergy3(Tp, $params$) $\equiv$ $params.Epmelt\_init + params.Ep\_melt3 + params.C\_pl \times params.Mp \times (Tp - params.Tmelt)$

11 MIS of Output Verification Module

11.1 Module

verify_output

11.2 Uses

parameters (5)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_output	array of reals, array of reals, array of reals, array of reals, array of reals, parameters	-	-

11.4 Semantics

11.4.1 State Variables

$expEPCM$: array of reals
$expEWat$: array of reals
$errorWater$: real
$errorPCM$: real

11.4.2 Environment Variables

win: 2D array of pixels displayed on the screen

11.4.3 Local Variables

11.4.4 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.

11.4.5 Access Routine Semantics

verify_output(t, Tw, Tp, Ew, Ep, $params$):	transition:	$expEPCM$, $expEWat$, $errorWater$, $errorPCM$, $win := (\forall i \in [1..\|t\| - 1])$ (expectedEp(traprule(delta($t[i-1]$, $t[i]$), $Tw[i]$, $Tp[i]$, $Tw[i-1]$, $Tp[i-1]$), $params$)), $(\forall i \in [1..\|t\| - 1])$ (expectedEw (expectedEc(traprule(delta($t[i-1]$, $t[i]$), $params.Tc$, $Tw[i]$, $params.Tc$, $Tw[i-1]$), $params$), post($expEPCM$))), error(sum(post($expEWat$)), $Ew[\|Ew\| - 1]$), error(sum(post($expEPCM$)), $Ep[\|Ep\| - 1]$), $(errorWater > ConsTol \vee errorPCM > ConsTol \Rightarrow$ Prints warning message(s))
	exception:	$(errorWater > ConsTol \Rightarrow$ warnWaterError | $errorPCM > ConsTol \Rightarrow$ warnPCMError) These exceptions do not terminate the program.

11.4.6 Local Functions

delta: real $\times$ real $\rightarrow$ real
delta($t1$, $t2$) $\equiv t2 - t1$

traprule: real $\times$ real $\times$ real $\times$ real $\times$ real $\rightarrow$ real
traprule(t, $A1$, $B1$, $A2$, $B2$) $\equiv t \times (A1 - B1 + A2 - B2)/2$

expectedEc: real $\times$ parameters $\rightarrow$ real
expectedEc(c, $params$) $\equiv params.hc \times params.Ac \times c$

expectedEp: real $\times$ parameters $\rightarrow$ real
expectedEp(p, $params$) $\equiv params.hp \times params.Ap \times p$

expectedEw: real $\times$ real $\rightarrow$ real
expectedEw(Ec, Ep) $\equiv Ec - Ep$

sum: array of reals $\rightarrow$ real
sum(a) $\equiv \sum_{i=0}^{|a|-1} a[i]$

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

12 MIS of Plotting Module

12.1 Module

plot

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	array of reals, array of reals, array of reals, array of reals, array of reals, string	-	-

12.4 Semantics

12.4.1 State Variables

$plotFilename$: string

12.4.2 Environment Variables

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

12.4.3 Assumptions

The input arrays are all of the same size.

12.4.4 Access Routine Semantics

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

13 MIS of Output Module

13.1 Module

output

13.2 Uses

parameters (5)

13.3 Syntax

13.3.1 Exported Constants

max_width: integer

13.3.2 Exported Access Program

Name	In	Out	Exceptions
output	string, array of reals, array of reals, array of reals, array of reals, array of reals, array of reals, parameters	-	-

13.4 Semantics

13.4.1 State Variables

outFilename: string

13.4.2 Environment Variables

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

13.4.3 Access Routine Semantics

output(*params*, t, Tw, Tp, Ew, Ep, $ETot$, $filename$):	transition:	*directory*: writes a .txt file named *outFilename* containing the input parameters, calculated 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. URL http://citeseer.ist.psu.edu/428727.html.

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: rho_p must be > 0
badMeltTemp	Error: Tmelt must be > 0 and $< Tc$
badCoilAndInitTemp	Error: Tc must be $>$ Tinit
badCoilTemp	Error: Tc must be > 0 and < 100
badPCMHeatCapSolid	Error: C_ps must be > 0
badPCMHeatCapLiquid	Error: C_pl must be > 0

badHeatFusion	Error: Hf must be > 0
badCoilArea	Error: Ac must be > 0
badWaterDensity	Error: rho_w must be > 0
badWaterHeatCap	Error: C_w must be > 0
badCoilCoeff	Error: hc must be > 0
badPCMCoeff	Error: hp must be > 0
badInitTemp	Error: Tinit must be > 0 and < 100
badFinalTime	Error: tfinal must be > 0
badInitAndMeltTemp	Error: Tinit must be < Tmelt
ODE_ACCURACY	*reltol* and *abstol* 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 $MaxStep$ steps and did not find solution
warnLength	Warning: It is recommended that 0.1 <= L <= 50
warnDiam	Warning: It is recommended that 0.002 <= D/L <= 200
warnPCMVol	Warning: It is recommended that Vp be >= 0.0001% of Vt
warnVolArea	Warning: It is recommended that Vp <= Ap <= (2/0.001) * Vp
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 Ac <= pi * (D/2) ∧ 2
warnWaterDensity	Warning: It is recommended that 950 < rho_w <= 1000
warnWaterHeatCap	Warning: It is recommended that 4170 < C_w < 4210
warnCoilCoeff	Warning: It is recommended that 10 < hc < 10000
warnPCMCoeff	Warning: It is recommended that 10 < hp < 10000
warnFinalTime	Warning: It is recommended that 0 < tfinal < 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 $ConsTol$)
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 $ConsTol$)