Module Interface Specification for Measuring Microstructure Changes During Thermal Treatment

Team #30, ReSprint
Edwin Do
Joseph Braun
Timothy Chen
Abdul Nour Seddiki
Tyler Magarelli

January 18, 2023

1 Revision History

Date	Name	Notes
Jan 17, 2023	Timothy Chen	Added Modules to Module Decomposition
Jan 18, 2023	Timothy Chen	Added Current State Module
Jan 18, 2023	Timothy Chen	Added File Output Module
Jan 18, 2023	Timothy Chen	Added Graphical Output Module
Jan 18 2023	Edwin Do	Added MIS info for UserInputValidation, HardwareInput-Validation, and Calculation Modules
Jan 18 2023	Edwin Do	Added state invariants

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

1 Revision History i

2 Symbols, Abbreviations and Acronyms ii

3 Introduction 1

4 Notation 1

5 Module Decomposition 1

6 MIS of Current State Module 3
- 6.1 Module . . . 3
- 6.2 Uses . . . 3
  - 6.2.1 Imported Types . . . 3
  - 6.2.2 Imported Access Programs . . . 3
- 6.3 Syntax . . . 3
  - 6.3.1 Exported Constants . . . 3
  - 6.3.2 Exported Access Programs . . . 3
- 6.4 Semantics . . . 3
  - 6.4.1 State Variables . . . 3
  - 6.4.2 State Invariant . . . 3
  - 6.4.3 Environment Variables . . . 4
  - 6.4.4 Assumptions . . . 4
  - 6.4.5 Access Routine Semantics . . . 4
  - 6.4.6 Local Functions . . . 4

7 MIS of FileOutput Module 5
- 7.1 Module . . . 5
- 7.2 Uses . . . 5
  - 7.2.1 Imported Types . . . 5
  - 7.2.2 Imported Access Programs . . . 5
- 7.3 Syntax . . . 5
  - 7.3.1 Exported Constants . . . 5
  - 7.3.2 Exported Access Programs . . . 5
- 7.4 Semantics . . . 5
  - 7.4.1 State Variables . . . 5
  - 7.4.2 State Invariant . . . 5
  - 7.4.3 Environment Variables . . . 6
  - 7.4.4 Assumptions . . . 6
  - 7.4.5 Access Routine Semantics . . . 6
  - 7.4.6 Local Functions . . . 6

8 MIS of Graphical Output Module **7**
8.1 Module . . . 7
8.2 Uses . . . 7
8.2.1 Imported Types . . . 7
8.2.2 Imported Access Programs . . . 7
8.3 Syntax . . . 7
8.3.1 Exported Constants . . . 7
8.3.2 Exported Access Programs . . . 7
8.4 Semantics . . . 7
8.4.1 State Variables . . . 7
8.4.2 State Invariant . . . 7
8.4.3 Environment Variables . . . 7
8.4.4 Assumptions . . . 8
8.4.5 Access Routine Semantics . . . 8
8.4.6 Local Functions . . . 8

9 MIS of Calculation Module **9**
9.1 Module . . . 9
9.2 Uses . . . 9
9.2.1 Imported Types . . . 9
9.2.2 Imported Access Programs . . . 9
9.3 Syntax . . . 9
9.3.1 Exported Constants . . . 9
9.3.2 Exported Access Programs . . . 9
9.4 Semantics . . . 9
9.4.1 State Variables . . . 9
9.4.2 State Invariants . . . 10
9.4.3 Environment Variables . . . 10
9.4.4 Assumptions . . . 10
9.4.5 Access Routine Semantics . . . 10
9.4.6 Local Functions . . . 10

10 MIS of UserInputValidation Module **12**
10.1 Module . . . 12
10.2 Uses . . . 12
10.2.1 Imported Types . . . 12
10.3 Syntax . . . 12
10.3.1 Exported Constants . . . 12
10.3.2 Exported Access Programs . . . 12
10.4 Semantics . . . 12
10.4.1 State Variables . . . 12
10.4.2 State Invariants . . . 12
10.4.3 Environment Variables . . . 12

10.4.4 Assumptions . . . 13
10.4.5 Access Routine Semantics . . . 13
10.4.6 Local Functions . . . 13

11 MIS of HardwareInputValidation Module **14**
11.1 Module . . . 14
11.2 Uses . . . 14
11.2.1 Imported Types . . . 14
11.3 Syntax . . . 14
11.3.1 Exported Constants . . . 14
11.3.2 Exported Access Programs . . . 14
11.4 Semantics . . . 14
11.4.1 State Variables . . . 14
11.4.2 State Invariants . . . 14
11.4.3 Environment Variables . . . 14
11.4.4 Assumptions . . . 14
11.4.5 Access Routine Semantics . . . 15
11.4.6 Local Functions . . . 15

12 Appendix **17**

3 Introduction

The following document details the Module Interface Specifications for [Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at .... [provide the url for your repo —SS]

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

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

The following table summarizes the primitive data types used by Measuring Microstructure Changes During Thermal Treatment.

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 (-∞, ∞)

The specification of Measuring Microstructure Changes During Thermal Treatment 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, Measuring Microstructure Changes During Thermal Treatment uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

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

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Communication Module Output Communication Module Remote Access Module Current State Module FileOutput Module Graphical Output Module
Software Decision	Calculation Module User Input Validation Module Hardware Input Validation Module

Table 1: Module Hierarchy

6 MIS of Current State Module

6.1 Module

Current State Module

6.2 Uses

6.2.1 Imported Types

HardwareInput: ($Voltage : real$; $Time : real$; $Temperature : real$; $Current : real$)
UserInput: ($SamplingRate : real$; $SampleLengthgth : real$; $SampleWidth : real$; $Filename : string$; $Name : string$; $SampleName : string$; $Date : string$)

6.2.2 Imported Access Programs

GetUserInput(): UserInput
GetHardwareInput(): HardwareInput

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
StateInit()			
DisplayUserInfo()	string, string, string, string, real, real, real		INVALID
DisplayHardwareState()	real, real, real, real		INVALID

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 State Invariant

N/A

6.4.3 Environment Variables

ApplicationWindow: the screen inferface where the information displayed to the user

6.4.4 Assumptions

StateInit() is called before any other access program

6.4.5 Access Routine Semantics

StateInit():

- transition: State Display is initialized on ApplicationWindow
- exception: none

DisplayUserInfo(Name, SampleName, Date, Filename, SamplingRate, SampleLength, SampleWidth):

- transition: Display Name, SampleName Date, Filename, SamplingRate, SampleLength, and SampleWidth on the ApplicationWindow
- exception: $exc := SamplingRate \notin \mathbb{R} \vee SamplingRate < 0 \vee SampleLength \notin \mathbb{R} \vee SampleLength < 0 \vee SampleWidth \notin \mathbb{R} \vee SampleWidth < 0 \Rightarrow INVALID$

DisplayHardwareState(Voltage, Current, Time, Temperature):

- transition: Display Voltage, Current, and Time on the ApplicationWindow
- exception: $exc := Voltage \notin \mathbb{R} \vee Voltage < 0 \vee Current \notin \mathbb{R} \vee Current < 0 \vee Time \notin \mathbb{R} \vee Time < 0 \vee Temperature \notin \mathbb{R} \Rightarrow INVALID$

6.4.6 Local Functions

N/A

7 MIS of FileOutput Module

7.1 Module

FileOutput Module

7.2 Uses

7.2.1 Imported Types

HardwareInput: ($Voltage : real$; $Time : real$; $Temperature : real$; $Current : real$)
UserInput: ($SamplingRate : real$; $SampleLength : real$; $SampleWidth : real$; $Filename : string$ $Name : string$; $SampleName : string$; $Date : string$)

7.2.2 Imported Access Programs

GetResistivity(): Real
GetResistance(): Real
GetUserInput(): UserInput
GetHardwareInput(): HardwareInput

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
FileInit()			
WriteUserInput()	string, string, string, real real, real		INVALID
WriteSampleOutput()	real, real, real, real, real, real	record	INVALID

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 State Invariant

N/A

7.4.3 Environment Variables

OutputFile: a file used to store data such as the user inputs and hardware outputs

7.4.4 Assumptions

FileInit() is called before any other access program.

7.4.5 Access Routine Semantics

FileInit():

- transition: Initializes an empty file
- exception: none

WriteUserInput(Name, SampleName, Date, SamplingRate, SampleLength, SampleWidth):

- transition: Write user input into the first line of the OutputFile
- exception: $exc := SamplingRate \notin \mathbb{R} \vee SamplingRate < 0 \vee SampleLength \notin \mathbb{R} \vee SampleLength < 0 lor SampleWidth \notin \mathbb{R} \vee SampleWidth < 0 \Rightarrow INVALID$

WriteSampleOutput(Time, Temperature, Voltage, Current, Resistance, Resistivity):

- transition: Write each data set into the OutputFile at each time interval
- exception: $exc := Time \notin \mathbb{R} \vee Time < 0 \vee Temperature \notin \mathbb{R} \vee Voltage < 0 \vee Voltage \notin \mathbb{R} \vee Current < 0 \vee Current \notin \mathbb{R} \vee Resistance < 0 \vee Resistance \notin \mathbb{R} \vee Resistivity < 0 \vee Resistance \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

7.4.6 Local Functions

N/A

8 MIS of Graphical Output Module

8.1 Module

File Output Module

8.2 Uses

8.2.1 Imported Types

HardwareInput: ($Voltage : real$; $Time : real$; $Temperature : real$; $Current : real$)

8.2.2 Imported Access Programs

GetResistivity(): Real GetResistance(): Real GetHardwareInput(): HardwareInput

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
GraphInit()			
GraphTimeVResistance()	real, real		INVALID
GraphTimeVResistivity()	real, real		INVALID
GraphVoltageVResistence()	real, real		INVALID
GraphVoltageVResistivity()	real, real		INVALID
GraphTemperatureVResistence()	real, real		INVALID
GraphTemperatureVResistivity()	real, real		INVALID

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 State Invariant

N/A

8.4.3 Environment Variables

ApplicationWindow: the screen inferface where the information displayed to the user

8.4.4 Assumptions

GraphInit() is called before any other access program

8.4.5 Access Routine Semantics

GraphInit():

- transition: Graph is initialized on ApplicationWindow
- exception: none

GraphTimeVResistance(Time, Resistance):

- transition: Disaply graph of Time versus Resistance on ApplicationWindow
- exception: $exc := Time \notin \mathbb{R} \vee Time < 0 \vee Resistance \notin \mathbb{R} \vee Resistance < 0 \Rightarrow INVALID$

GraphTimeVResistivity(Time, Resistivity):

- transition: Display graph of Time versus Resistivity on ApplicationWindow
- exception: $exc := Time \notin \mathbb{R} \vee Time < 0 \vee Resistivity \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

GraphVoltageVResistance(Voltage, Resistance):

- transition: Display graph of Voltage versus Resistance on ApplicationWindow
- exception: $exc := Voltage \notin \mathbb{R} \vee Voltage < 0 \vee Resistance \notin \mathbb{R} \vee Resistance < 0 \Rightarrow INVALID$

GraphVoltageVResistivity(Voltage, Resistivity):

- transition: Display graph of Voltage versus Resistivity on ApplicationWindow
- exception: $exc := Voltage \notin \mathbb{R} \vee Voltage < 0 \vee Resistivity \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

GraphTemperatureVResistance(Temperature, Resistance):

- transition: Display graph of Temperature versus Resistance on ApplicationWindow
- exception: $exc := Temperature \notin \mathbb{R} \vee Resistance \notin \mathbb{R} \vee Resistance < 0 \Rightarrow INVALID$

GraphTemperatureVResistivity(Temperature, Resistivity):

- transition: Display graph of Temperature versus Resistivity on ApplicationWindow
- exception: $exc := Temperature \notin \mathbb{R} < Resistivity \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

8.4.6 Local Functions

N/A

9 MIS of Calculation Module

[Use labels for cross-referencing —SS]
[You can reference SRS labels, such as R??. —SS]
[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

9.1 Module

Calculation

9.2 Uses

9.2.1 Imported Types

HardwareInput:
($Voltage : real$; $Time : real$; $Temperature : real$; $Current : real$)

UserInput:
($SamplingRate : real$; $SampleLength : real$; $SampleWidth : real$; $Filename : string$; $Name : string$; $SampleName : string$; $Date : string$)

9.2.2 Imported Access Programs

getHardwareInput(): HardwareInput
getUserInput(): UserInput

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
getResistance()	-	Real	INVALID
getResistivity()	-	Real	INVALID

9.4 Semantics

9.4.1 State Variables

Resistance : The calculated resistance value (real)
Resistivity : The calculated resistivity value (real)
SampleArea : The calculated area of sample based on the length and width from user's input

9.4.2 State Invariants

Resistance ≥ 0
Resistivity ≥ 0
SampleArea ≥ 0

9.4.3 Environment Variables

N/A

9.4.4 Assumptions

We assume that the user may enter invalid values for inputs such as characters, empty spaces etc.. This will cause the program to throw and INVALID exception. This type of programmer error is also captured in the UserInputValidation Module to improve redundancy.

9.4.5 Access Routine Semantics

getResistance():

- transition: N/A
- output: out:= Resistance
- exception: $exc := Resistance \notin \mathbb{R} \vee Resistance < 0 \Rightarrow INVALID$

getResistivity():

- transition: N/A
- output: out := Resistivity
- exception: $exc := Resistivity \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

9.4.6 Local Functions

findSampleArea(SampleLength, SampleWidth)

- transition: SampleArea := SampleLength x SampleWidth

calcResistance(voltage, current):

- transition: Resistance := voltage/current
- exception: $exc :=$
  $voltage \notin \mathbb{R} \vee voltage < 0$
  $\vee\, current \notin \mathbb{R} \vee current < 0$
  $\Rightarrow INVALID$

calcResistivity(Resistance, SampleArea, SampleLength):

- transition: Resistivity := (resistance x SampleArea)/SampleLength
- exception: $exc :=$
  $voltage \notin \mathbb{R} \vee voltage < 0$
  $\vee\, current \notin \mathbb{R} \vee current < 0$
  $\Rightarrow INVALID$

10 MIS of UserInputValidation Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

10.1 Module

UserInputValidation

10.2 Uses

10.2.1 Imported Types

UserInput:
($SamplingRate : real$; $SampleLength : real$; $SampleWidth : real$; $Filename : string$; $Name : string$; $SampleName : string$; $Date : string$)

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
getUserInput()	-	ADT (UserInput)	INVALID

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 State Invariants

N/A

10.4.3 Environment Variables

N/A

10.4.4 Assumptions

We assume that the user may enter invalid values for inputs such as characters, empty spaces etc.. This will cause the program to throw and INVALID exception.

10.4.5 Access Routine Semantics

getUserInput():

- output: out:= UserInput
- exception: $exc := validateFileData \neq TRUE \vee$
  $validateSampleData \neq TRUE$
  $\Rightarrow INVALID$

10.4.6 Local Functions

valdiateFileData(Filename, Date, Name):

- output: out:= TRUE
- exception: $exc := Filename.type \neq STRING \vee$
  $Date.type \neq STRING \vee$
  $Name.type \neq STRING$
  $\Rightarrow INVALID$

validateSampleData(SamplingRate, SampleLength, SampleWidth):

- output: out:= TRUE
- exception: $exc := SamplingRate \notin \mathbb{R} \vee SamplingRate < 0 \vee$
  $SampleLength \notin \mathbb{R} \vee SampleLength < 0 \vee$
  $SampleWidth \notin \mathbb{R} \vee SampleWidth < 0$
  $\Rightarrow INVALID$

11 MIS of HardwareInputValidation Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R**??**. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

11.1 Module

HardwareInputValidation

11.2 Uses

11.2.1 Imported Types

HardwareInput:
($Voltage : real$; $Time : real$; $Temperature : real$; $Current : real$)

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
getHardwareInput()	-	ADT (HardwareInput)	INVALID

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 State Invariants

N/A

11.4.3 Environment Variables

N/A

11.4.4 Assumptions

N/A

11.4.5 Access Routine Semantics

getHardwareInput():

- output: out:= HardwareInput
- exception: $exc := validateParameters \neq TRUE \ \Rightarrow INVALID$

11.4.6 Local Functions

validateParameters(Voltage, Time, Current):

- output: out:= TRUE
- exception: $exc := Voltage < 0 \vee$
  $Time < 0 \vee Current < 0$
  $\Rightarrow INVALID$

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.

12 Appendix

[Extra information if required —SS]