Module Interface Specification for Measuring Microstructure Changes During Thermal Treatment

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

April 5, 2023

1 Revision History

Date	Developer	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
Jan 18 2023	Tyler Magarelli	Added Input Communication Module
Jan 18 2023	Tyler Magarelli	Added Output Communication Module
Jan 18 2023	Tyler Magarelli	Added Remote Access Module
Apr 5 2023	Edwin Do	Updated with revised modules

2 Symbols, Abbreviations, and Acronyms

See SRS Documentation at here.

Contents

3 Introduction

The following document details the Module Interface Specifications for Measuring Microstructure Changes During Thermal Treatment. This project will allow the Materials Engineering lab at McMaster University, led by Dr. Zurob, to use software capable of providing data on thermally treated metals. The data includes measurements of the resistivity of the material as well as graphical representations and analysis.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at our GitHub repository.

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

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	Remote Access Module (6) Current State Module (7) FileOutput Module (8) Graphical Output Module (9)
Software Decision	Calculation Module (10) User Input Validation Module (11) Instrument Input Validation Module (12)

Table 1: Module Hierarchy

6 MIS of Remote Access Module

6.1 Module

Remote Access Module

6.2 Uses

CurrentState Module
Microsoft.Extensions.Hosting

6.2.1 Imported Types

IHost

6.2.2 Imported Access Program

RemoteGetCurrentStatus()
RemoteGetExperimentStatus()
RemoteTurnCurrentOn()
RemoteTurnCurrentOff()
RemoteStartCapture()
RemoteStopCapture()

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
StartServer(host)	IHost	-	INVALID

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 State Invariant

N/A

6.4.3 Environment Variables

MainWindow ServerStatus

6.4.4 Assumptions

Hosting will occur only on port 5100
StartServer() is called before any other routine

6.4.5 Access Routine Semantics

StartServer():

- transition: ServerStatus := FALSE $\rightarrow$ ServerStatus := TRUE
- exception: exc := PortNotAvailable $\vee$ PortAlreadyInUse $\Rightarrow INVALID$

6.4.6 Local Functions

N/A

7 MIS of Current State Module

7.1 Module

Current State Module (MainWindow)

7.2 Uses

File Output Module
Calculation Module
Instrument Input Validation Module
User Input Validation Module
Remote Access Module

7.2.1 Imported Types

HardwareInput: ($Voltage : \mathbb{R}$; $Time : \mathbb{R}$; $Temperature : \mathbb{R}$; $Current : \mathbb{R}$, $Resistance : \mathbb{R}$; $Resistivity : \mathbb{R}$;)

InstrumentInput: ($CurrentLevel : \mathbb{R}$; $Compliance : \mathbb{R}$; $SampleRate : \mathbb{R}$; $JuncTemperature : \mathbb{R}$; string: Range; string ThType)

UserInput: (UserSampleThickness: $\mathbb{R}$; UserSampleLength: $\mathbb{R}$; UserSampleWidth: $\mathbb{R}$; UserName: *string*; UserSampleName: *string*)

7.2.2 Imported Access Programs

StartServer(): void
CheckSampleRate(r) : bool
CheckCurrentLevel(c) : bool
CheckCompliance(c): bool
CheckJuncTemperature(t): bool
CalcResistance(): $\mathbb{R}$
CalcResistivity(): $\mathbb{R}$
CalcTemperature(): $\mathbb{R}$
CalcAperture(): string
checkUserInput(): UserInput
checkInputBox(): bool
validateUserData(): bool

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
RemoteGetCurrentStatus()	-	bool	INVALID
RemoteGetExperimentStatus()	-	bool	INVALID
RemoteTurnCurrentOn()	-	-	INVALID
RemoteTurnCurrentOff()	-	-	INVALID
RemoteStartCapture()	-	-	INVALID
RemoteStopCapture()	-	-	INVALID

7.4 Semantics

7.4.1 State Variables

ExperimentStatus: bool

7.4.2 State Invariant

N/A

7.4.3 Environment Variables

DeviceVoltageRate: The sampling rate setting on the connected nanovoltmeter
DeviceTemperature: The temperature setting on the connected multimeter
DeviceCurrentLevel: The current level setting on the connected current source
DeviceCompliance: The compliance setting on the connected current source
DeviceCurrentOutput: The output setting on the connected current source

7.4.4 Assumptions

InitializeComponent() is called before any other access program

7.4.5 Access Routine Semantics

RemoteGetCurrentStatus():

- output: The status of current output from current source (bool)
- exception: none

RemoteGetExperimentStatus():

- output: The status of the experiment (bool)
- exception: none

RemoteTurnCurrentOn():

- transition: Turns the connected current output on
- exception: none

RemoteTurnCurrentOff():

- transition: Turns the connected current output off
- exception: none

RemoteStartCapture():

- transition: Starts the experiment to capture data
- exception: none

RemoteStopCapture():

- transition: Stops the experiment from capturing data
- exception: none

7.4.6 Local Functions

InitializeComponent():

- transition: State is initialized on MainWindow
- exception: none

InitializeCurrentSource():

- transition: Address of Current Source is found and connected
- exception: none

InitializeNanoVoltmeter():

- transition: Address of Nano Voltmeter is found and connected
- exception: none

InitializeMultimeter():

- transition: Address of Multimeter is found and connected
- exception: none

SetVoltRate(rate):

- transition: DeviceVoltageRate = rate
- exception: if CheckSampleRate = FALSE → INVALID

SetJuncTemp(temp):

- transition: DeviceTemperature = temp
- exception: if CheckJuncTemp = FALSE → INVALID

SetCurrent(current):

- transition: DeviceCurrentLevel = current
- exception: if CheckCurrentLevel = FALSE → INVALID

ToggleCurrentOutput():

- transition: DeviceCurrentOutput = !DeviceCurrentOutput
- exception: N/A

StartCapture():

- transition: ExperimentStatus = TRUE
- exception: N/A

StopCapture():

- transition: ExperimentStatus = FALSE
- exception: N/A

8 MIS of FileOutput Module

8.1 Module

FileOutput Module

8.2 Uses

8.2.1 Imported Types

HardwareInput: ($Voltage : \mathbb{R}$; $Time : \mathbb{R}$; $Temperature : \mathbb{R}$; $Current : \mathbb{R}$)
UserInput: ($SamplingRate : \mathbb{R}$; $SampleLength : \mathbb{R}$; $SampleWidth : \mathbb{R}$; $Filename : string$ $Name : string$; $SampleName : string$; $Date : string$)

8.2.2 Imported Access Programs

GetResistivity(): $\mathbb{R}$
GetResistance():$\mathbb{R}$
GetUserInput(): UserInput
GetHardwareInput(): HardwareInput

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
FileInit()	-	FileOutput	-
GetFilePath()	-	string	-
WriteUserInput()	string, string, string, $\mathbb{R}, \mathbb{R}, \mathbb{R}$		INVALID
WriteSampleOutput()	$\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}$	record	INVALID

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 State Invariant

N/A

8.4.3 Environment Variables

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

8.4.4 Assumptions

FileOutput() is called and initialized before any other access program.

8.4.5 Access Routine Semantics

FileOutput(filePath):

- transition: Initializes a FileOutput instance
- exception: INVALID

GetFilePath():

- Output: string
- exception: N/A

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

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

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

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

8.4.6 Local Functions

N/A

9 MIS of Graphical Output Module

9.1 Module

File Output Module

9.2 Uses

SyncFusion Charts API (3rd Party)

9.2.1 Imported Types

SyncFusion - SfChart
SyncFusion - DataGrid

9.2.2 Imported Access Programs

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

N/A

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 State Invariant

N/A

9.4.3 Environment Variables

MainWindow: The application interface where the information displayed to the user

9.4.4 Assumptions

initializeChart() is called before any other access program

9.4.5 Access Routine Semantics

initializeChart():

- transition: Graph is initialized on MainWindow and data source is binded correctly.
- exception: none

9.4.6 Local Functions

N/A

10 MIS of Calculation Module

10.1 Module

Calculation Module

10.2 Uses

10.2.1 Imported Types

N/A

10.2.2 Imported Access Programs

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
CalcResistance()	$\mathbb{R}$,$\mathbb{R}$	$\mathbb{R}$	-
CalcResistivity()	$\mathbb{R}$,$\mathbb{R}$,$\mathbb{R}$	$\mathbb{R}$	-
CalcTemperature()	$\mathbb{R}$,$\mathbb{R}$, int, $\mathbb{R}$	$\mathbb{R}$	-
CalcAperture()	$\mathbb{R}$	string	-

10.4 Semantics

10.4.1 State Variables

Resistance : The calculated resistance value ($\mathbb{R}$)
Resistivity : The calculated resistivity value ($\mathbb{R}$)
SampleArea : The calculated area of the sample based on the length and width from the user's input

10.4.2 State Invariants

Resistance ≥ 0
Resistivity ≥ 0
SampleArea ≥ 0

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 type of error is captured in the UserInputValidation Module.

10.4.5 Access Routine Semantics

CalcResistance(voltage, current):

- output: out:= (voltage/current) $\mathbb{R}$
- exception: $exc := Resistance \notin \mathbb{R} \vee Resistance < 0 \Rightarrow INVALID$

CalcResistivity(resistance, area, length):

- output: out := (resistance $\cdot$ area /length) $\mathbb{R}$
- exception: $exc := Resistivity \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

CalcTemperature(volt, junc_temp, type, temperature):

- output: out := $\mathbb{R}$
- exception: $exc := Temperature \notin \mathbb{R} \Rightarrow INVALID$

CalcAperture(rate):

- output: out := string
- exception: $exc :=$ Aperture is not string $\Rightarrow INVALID$

10.4.6 Local Functions

N/A

11 MIS of UserInputValidation Module

11.1 Module

UserInputValidation Module

11.2 Uses

11.2.1 Imported Types

UserInput:
(UserSampleThickness: $\mathbb{R}$; UserSampleLength: $\mathbb{R}$; UserSampleWidth: $\mathbb{R}$; UserName: *string*; UserSampleName: *string*)

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
checkUserInput()	string,string,$\mathbb{R}$,$\mathbb{R}$,$\mathbb{R}$	UserInput	INVALID
checkInputBox()	string	bool	INVALID
validateUserData()	string,string,$\mathbb{R}$,$\mathbb{R}$,$\mathbb{R}$	bool	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

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

11.4.5 Access Routine Semantics

checkUserInput(userName, userSampleName, userSampleLength, userSampleWidth, userSampleThickness):

- output: out:= UserInput
- exception: $exc := validateUserData \neq TRUE \Rightarrow INVALID$

checkInputBox(input):

- output: out:= bool
- exception: $exc := input = EMPTY \vee input = NULL \Rightarrow INVALID$

validateUserData(userName, userSampleName, userSampleLength, userSampleWidth, userSampleThickness):

- output: out:= bool
- exception: $exc := (userName \vee userSampleName is NOT string) \vee (userSampleLength \leq 0 \vee userSampleWidth \leq 0 \vee userSampleThickness \leq 0)$
  $\Rightarrow INVALID$

11.4.6 Local Functions

N/A

12 MIS of InstrumentInputValidation Module

12.1 Module

InstrumentInputValidation Module

12.2 Uses

12.2.1 Imported Types

N/A

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
CheckSampleRate(r)	$\mathbb{R}$	bool	INVALID
CheckCurrentLevel(c)	$\mathbb{R}$	bool	INVALID
CheckCompliance(c)	$\mathbb{R}$	bool	INVALID
CheckJuncTemperature(t)	$\mathbb{R}$	bool	INVALID

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 State Invariants

N/A

12.4.3 Environment Variables

N/A

12.4.4 Assumptions

N/A

12.4.5 Access Routine Semantics

CheckSampleRate(r):

- output: out:= bool
- exception: $exc := r > 600 \vee r < 1 \Rightarrow INVALID$

CheckCurrentLevel(c):

- output: out:= bool
- exception: $exc := c < -105 \vee c > 105 \vee c = 0 \Rightarrow INVALID$

CheckCompliance(c):

- output: out:= bool
- exception: $exc := c > 105 \vee c < -0.1 \Rightarrow INVALID$

CheckJuncTemperature(t):

- output: out:= bool
- exception: $exc := t > 9999 \vee t < -9999 \Rightarrow INVALID$

12.4.6 Local Functions

N/A

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.

13 Appendix