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

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

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

Contents

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 File Output Module Graphical Output Module
Software Decision	Resistivity Module Resistance 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: ($SamplingRate : real$; $SampleLen : real$; $SampleWidth : real$; $Filename : string$ $Name : string$; $SampleName : string$; $Date : string$)

6.2.2 Imported Access Programs

Name	In	Out	Exceptions
GetUserInput()		ADT	NULL
GetHardwareInput()		ADT	NULL

6.3 Syntax

6.3.1 Exported Constants

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

[Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 State Invariant

6.4.3 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

6.4.4 Assumptions

StateInit() is called before any other access program

6.4.5 Access Routine Semantics

StateInit():

- transition: State Display is Initialized
- exception: none

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

- transition: Display Name, SampleName Date, Filename, SamplingRate, SampleLen, and SampleWidth in user input state section
- exception: $exc := SamplingRate \notin \mathbb{R} \vee SamplingRate < 0 \vee SampleLen \notin \mathbb{R} \vee SampleLen < 0 lor SampleWidth \notin \mathbb{R} \vee SampleWidth < 0 \Rightarrow INVALID$

DisplayHardwareState(Voltage, Current, Time, Temperature):

- transition: Display Voltage, Current, and Time in hardware state section
- exception: $exc := Voltage \notin \mathbb{R} \vee Voltage < 0 \vee Current \notin \mathbb{R} \vee Current < 0 \vee Time \notin \mathbb{R} \vee Time < 0 \vee Temperature \notin \mathbb{R} \Rightarrow INVALID$

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

6.4.6 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of File Output Module

7.1 Module

File Output Module

7.2 Uses

7.2.1 Imported Types

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

7.2.2 Imported Access Programs

Name	In	Out	Exceptions
GetResistivity()		real	NULL
GetResistance()		real	NULL
GetUserInput()		ADT	NULL
GetHardwareInput()		ADT	NULL

7.3 Syntax

7.3.1 Exported Constants

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		INVALID

7.4 Semantics

7.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

7.4.2 State Invariant

7.4.3 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface,

keyboard, file, etc. —SS]

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, SampleLen, SampleWidth):

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

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

- transition: Write each data set into the file until the program stops
- exception: $exc := Time \notin \mathbb{R} \vee Time < 0 \vee Temperature \notin \mathbb{R} \vee Voltage < 0 \vee Voltage \notin \mathbb{R} \vee Current < 0 \vee Current \notin \mathbb{R} \vee Resistance < 0 \vee Resistance \notin \mathbb{R} \vee Resistivity < 0 \vee Resistance \notin \mathbb{R} \vee Resistivity < 0 \Rightarrow INVALID$

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

7.4.6 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

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

Name	In	Out	Exceptions
GetResistivity()		real	NULL
GetResistance()		real	NULL
GetHardwareInput()		ADT	NULL

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
GraphInit()			
GraphTimeVResistence()	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

[Not all modules will have state variables. State variables give the module a memory. —SS]

8.4.2 State Invariant

8.4.3 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface,

keyboard, file, etc. —SS]

8.4.4 Assumptions

GraphInit() is called before any other access program

8.4.5 Access Routine Semantics

GraphInit():

- transition: Graph is Initialize
- exception: none

GraphTimeVResistence(Time, Resistance):

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

GraphTimeVResistivity(Time, Resistivity):

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

GraphVoltageVResistence(Voltage, Resistance):

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

GraphVoltageVResistivity(Voltage, Resistivity):

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

GraphTemperatureVResistence(Temperature, Resistance):

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

GraphTemperatureVResistivity(Temperature, Resistivity):

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

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

8.4.6 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

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.

9 Appendix

[Extra information if required —SS]