Module Interface Specification for Truss Tool

Maryam Valian

April 19, 2023

1 Revision History

Date	Version	Notes
16/03/2023	1.0	Initial Draft
17/03/2023	1.1	Update Control module
19/03/2023	1.2	Update formulas
20/03/2023	1.3	Update other modules
23/03/2023	1.4	Update based on first Reviewer Feedback
30/03/2023	1.5	Update based on second Reviewer Feedback
18/04/2023	1.6	Update based on Supervisor feedback

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at here.

Contents

3 Introduction

The following document details the Module Interface Specifications for Truss Tool is software designed for engineers and students to analyze a truss.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at Truss Tool 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 $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$. Capital letters are used to indicate sequenced data type.

The following table summarizes the primitive data types used by Truss Tool.

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 Truss Tool 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, Truss Tool 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 Module	
Behaviour-Hiding Module	Input parameters module Input verification module Specification parameters module Output format module Output verification module Support reactions module. Force decomposing module Internal force module Control module
Software Decision Module	Sequence data structure module linear equation solver module

Table 1: Module Hierarchy

6 MIS of Control module

6.1 Module

Main

6.2 Uses

Inputs Module, Internals Module, Outputs Module

6.3 Syntax

6.3.1 Exported Constants

Not applicable.

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
Main	-	-	-

6.4 Semantics

6.4.1 State Variables

from R3:
$px : \mathbb{R}$
$py : \mathbb{R}$
$ry : \mathbb{R}$

from R4:
$I_m : \mathbb{R}$

6.4.2 Environment Variables

Not Applicable.

6.4.3 Assumptions

Not Applicable.

6.4.4 Access Routine Semantics

Main():

- transition: Modifies the state of the Input module and the environment variables for the Output modules by following steps:

  $(n, m, J, M, F, S_p, S_r) = Inputs(FileName)$

  $(P_x, P_y, R_y, I) = Internals(n, m, J, M, F, S_p, S_r)$

  $Outputs(P_x, P_y, R_y, I)$

6.4.5 Local Functions

None.

7 MIS of Input Module

7.1 Module

Inputs

7.2 Uses

Input Verification Module

7.3 Syntax

Name	In	Out	Exceptions
load_params	string	-	Input File Error
verify_params	-	-	Input Parameters Error
n	-	$\mathbb{N}$	
m	-	$\mathbb{N}$	
$J[j]$	-	$\mathbb{R}$	
M_m	-	$\mathbb{N}$	
F_m	-	$\mathbb{R}$	
sp	-	$\mathbb{N}$	
sr	-	$\mathbb{N}$	

7.3.1 Exported Constants

Not applicable.

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
Inputs	input plain text file	Parameters	FileError

7.4 Semantics

7.4.1 State Variables

from R1 and R2:
n : Integer
m : Integer
$J[\,j\,]$: Sequence of pair of Real numbers (x,y)
$M[\,i\,]$: Sequence of pair of Integers (end1, end2)
$F[\,j\,]$: Sequence of pair of Real numbers
S_p : Integer
S_r : Integer

7.4.2 Environment Variables

InputFile: sequence of strings in the text file.

7.4.3 Assumptions

- Load parameters 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. The order is important.

7.4.4 Access Routine Semantics

Load-Parameter():

- transition: Modifies the state variables from the input file:
  - Read sequentially from file and loads state variables from R1 and R2.
  - verify-input()
- output: None
- exception: file-Error, input-type-mismatch-Error

Verify-Input():

- transition: Modifies the state variables from the input file:

Table 2: Constraint check table

Constraint	**Error**
$n_{min} \leq$n$\leq$n$_{max}$	InvalidJonits
$m \leq$m$_{max}$	InvalidMembers
$m < n$	InvalidMembers
$J_{min} \leq J[j] \leq J_{max}$	InvalidLocation
$F_{min} \leq$F[j] $\leq$F$_{max}$	InvalidForce
$Sp > n, Sp < 0$	InvalidIndex
$Sr > n, Sr < 0$	InvalidIndex
$Sp = Sr = 0$	NoSupportDefined

  - Read constants from specification module.
  - Check constraints given in Table 2.

- output: None
- exception: Bad-parameters-Error

8 MIS of Specification Parameters

8.1 Module

Specification

8.2 Uses

None.

8.3 Syntax

8.3.1 Exported Constants

From Table.2 SRS:
$n_{min} := 3$
$n_{max} := 20$
$m_{max} := 30$
$J_{max} := 200$
$J_{min} := 0$

$F_{max} := 100000$
$F_{min} := -100000$
$error_{value} := 0.1$

8.3.2 Exported Access Programs

8.4 Semantics

8.4.1 State Variables

None.

8.4.2 Environment Variables

None.

9 MIS of Internals Module

9.1 Module

Internals

9.2 Uses

Sequence data structure module
linear equation solver module

9.3 Syntax

9.3.1 Exported Constants

None.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
React	n, m, J, F, S_p, S_r	P_x, P_y, R_y	ReactFailed
Internals	n, m, J, M, F, S_p, S_r	$I[\,i\,]$	InternalFailed

Exception ReactFailed will show to the user if a correct member is not defined by the user so the angle between the member and the x-axis cannot be calculated. Exception InternalFailed will show to the user if the equations cannot be solved.

9.4 Semantics

9.4.1 State Variables

None.

9.4.2 Environment Variables

None.

9.4.3 Assumptions

None.

9.4.4 Access Routine Semantics

- Calculates: $Out := P_x, P_y, R_y, I\,[i\,]$ by following steps:
  $\#Find\ Support\ reactions\ (P_x, P_y, R_y)$
  $P_x = -\sum F_x$

$$A = \begin{bmatrix} 1 & 1 \\ J\,[S_p\,] & J[\,S_r\,] \end{bmatrix}$$

$$B = \begin{bmatrix} -\sum F[\,i\,] \\ -\sum (J[\,i\,] * F[\,i\,]) \end{bmatrix}$$

  $A * [\,P_y, R_y\,] = B$
  $(P_y, R_y) = Solve(A, B)$

  $\#Find\ Internal\ Forces\ I_m$

  $\theta[\,i\,] = Tan^{-}1(J[\,M[i, end1\,]], J[M[i, end2]])$
  $\#Compute\ a[\,i, j\,] Elements of A, and compute b[j]\ elements\ of\ B$ from equilibrium equations where $AI = B$:

  Build Equilibrium equations in the X-axis direction: $a[i, j] = \cos\theta[i]$
  $b_j = \sum All\ forces\ and\ reactions\ at\ j$

  Or Build Equilibrium equations in the Y-axis direction:

  $a[i, j] = \sin\theta[i]$
  $b_j = \sum All\ forces\ and\ reactions\ at\ j$

$I = Solve(A, B)$

9.5 MIS of Output Verification Module

9.6 Module

verify-out

9.7 Uses

None.

9.8 Syntax

9.8.1 Exported Constants

None.

9.8.2 Exported Access Programs

Name	In	Out	Exceptions
Verify-0ut	$check(I_m, J_j) : \mathbb{R} * \mathbb{R} \rightarrow Boolean$	Valid/NotValid	verifyFailed

9.9 Semantics

9.9.1 State Variables

None.

9.9.2 Environment Variables

None.

9.9.3 Assumptions

None.

9.9.4 Access Routine Semantics

- Exception: Exep:=$(\sum (I_m(n) + F_m(n), P_x(n), P_y(n), R_y(n)) \neq 0) \Rightarrow VerifyFailed$

9.9.5 Assumptions

None.

9.10 MIS of Output Module

9.11 Module

Output

9.12 Uses

None.

9.13 Syntax

9.13.1 Exported Constants

output file name.

9.13.2 Exported Access Programs

None.

9.14 Semantics

9.14.1 State Variables

None.

9.14.2 Environment Variables

output file

9.14.3 Assumptions

None.

9.15

Assumptions None.

9.15.1 Access Routine Semantics

- transition: Write to file the following: the input parameters from Param, and the calculated values I_m, P_x, P_y, R_y.

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.

10 Appendix

Not applicable.