Software Requirements Specification

for

NifflerTM: A Command-line Turing Machine

Author:	Devan Farrell
Semester:	Spring 2017
Course:	CPT_S 322 Software Engineering

March 27th 2017

Initial Specification

List of Figures

Revision History

Revision Date	Description	Rational
3/10/2017	Specification as of initial release	Initial release
3/27/2017	Corrections to initial release	Multiple errors, some missing information, as well as some organization restructuring

1. Introduction

1.1 Purpose and Use

NifflerTM is a command-line interface Turing Machine simulator designed for the Ubuntu operating system.

1.2 Intended Audience

NifflerTM was designed for academic purposes a a personal project for a software engineering course.

1.3 Document Layout

The **background** section of the document describes what a Turing machine is in general along with the formal specification and how a Turing machine works theoretically.

The **overview** section of the document describes the purpose and software environment for the development and installation of the application. It will also describes how a user will trace the operation of the Turing machine.

The **environment** section of the document describes the input and output devices to that the NifflerTM uses. It also shows examples of proper formatting of these input files and erroneous files.

The **operation** section of the document describes the how to use the application in general. It includes the use of commands and settings.

2. Background

A Turing machine is an abstract mathematical model of a computer that can perform functions infinitely fast. In this abstract model there is a tape head that reads from and writes on an infinitely long tape. The Turing Machine has a list of functions and states associated with it. Based on what letter the tape head reads and what state the machine currently is in, a function is chosen that will determine the write letter, what state to move to, and whether to move right or left on the tape head.

The formal definition of a Turing machine is as follows

A Turing machine is a 7-tuple

$M = \{ Q, \Sigma, \Gamma, \delta, q_0, B, F \}$

Q is a finite set of states

$\Sigma \subseteq (\Gamma - \{B\})$ a finite input alphabet

Γ is a finite tape alphabet

δ is a transition function from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

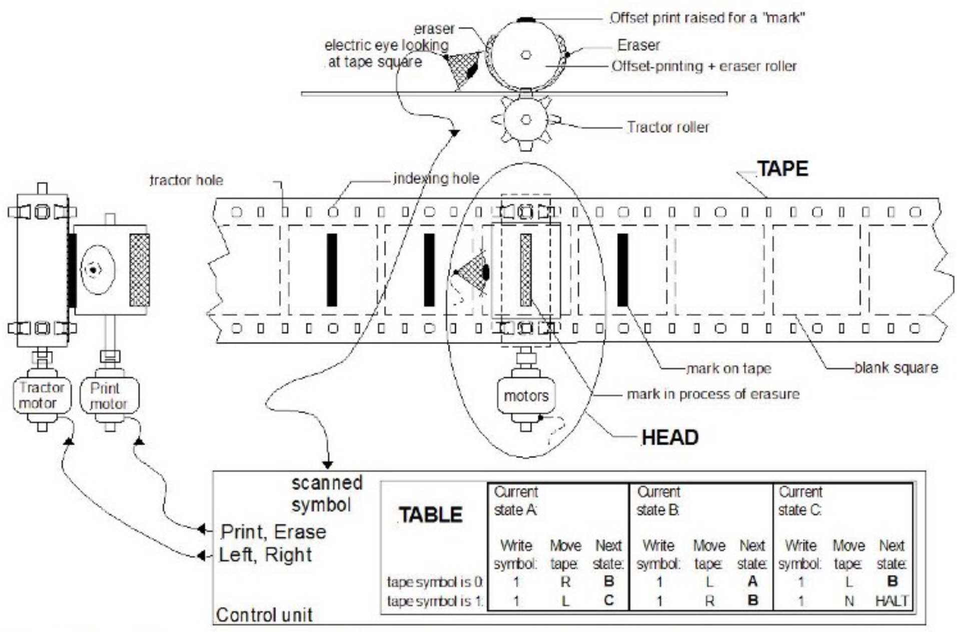
$q_0 \in Q$ is the initial state

$B \in \Gamma$ is the blank symbol

$F \subseteq Q$ is the set of final states

By the very definition a Turing machine with all its requirements is impossible to make. Mechanical representations of the concept are possible and might look something like the following figure.

Figure 2-1 Mechanical Turing Machine



	Current state A:			Current state B:			Current state C:		
	Write symbol:	Move tape:	Next state:	Write symbol:	Move tape:	Next state:	Write symbol:	Move tape:	Next state:
tape symbol is 0:	1	R	**B**	1	L	**A**	1	L	**B**
tape symbol is 1:	1	L	**C**	1	R	**B**	1	N	HALT

A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

3. Overview

NifflerTM was designed to allow the user input a custom Turing machine and trace their operation on input strings. It is developed exclusively in C++ and the display is optimized for the Ubuntu command terminal. The user traces the operation of the Turing machine on an input string with the run command and the instantaneous description. (See 5.2.9 for details on the run command)

The Instantaneous description has three essential parts. The first part is the number of transitions currently performed on the input string. It is in integer form and followed immediately by a period. The tape head is displayed with the current state in square brackets in its proper location on the tape head. The input string is displayed to the left and right of the tape head appropriately and is truncated with <> when the length exceeds the setting limit. (See 5.2.8 for details on the truncate command)

Figure 3-1 Instantaneous Description Examples
0. [s0]aabaab
10. <baabaa[s4]baabaa>
1000. <XYYXYY[s5]

If the Turing machine successfully transitions through the input string to a final state (F) then a message will display that it was accepted with the number of transition and string inputted. If at any point the Turing machine can't transition the operation stops on the input string and a message will display stating that the input string was rejected with the number of transition and the string inputted. Alternatively, operations can be stopped with the quit command. (See 3.2.10 for more details on the quit command)

Figure 3-2 Input String Accept and Reject Messages
'aaabbb' was accepted in 17 transitions
'aaaaab' was rejected in 12 transitions

4. Environment

4.1 Input and Output Devices

The input devices for the application are the keyboard, a plain-text TM definition file (See 4.2), and an optional plain-text input string file (See 4.3). Output devices include the monitor and a plain-text input string file. Strings can be inputted after successfully reading a definition file with the keyboard using the insert command (See 5.2.5 for more details). All other commands can also invoked from the command line using a keyboard. (See 5.2 for more details)

4.2 Turing Machine Definition File

The Turing machine definition file is a plain text file with an extension "*.def*". It has 7 mandatory keywords and an optional description portion. The description portion, if included, must be at beginning of the file. There is no other comment lines possible. The other 7 sections must be in the order listed in the figure 4-2-1 below and are declared with their associated keyword. The keywords must end with a colon as indicated in the figure, many require an underscore, are not case sensitive, must not be duplicated, and don't need to be on their own line. If any portion of the definition file is rejected than the application will display appropriate error messages and terminate operation.

Figure 4-2-1 Keywords

Keyword	Meaning
states:	Q
input_alphabet:	Σ
tape_alphabet:	Γ
transition_function:	δ*
initial_state:	q_0
blank_character:	B
final_states:	F

* transition_function: is a composite of functions, not a single function

STATES:

States: is the equivalent to Q in the definition of a Turing machine. Everything between the "states:" and the "input alphabet:" keywords are considered as a part of Q. Each state must be separated by blank space. Each state is a string and can be composed of uppercase and lowercase letters, digits, or underscores. All other symbols are illegal and will be rejected. Each state is case sensitive and must be unique. All non-unique states in Q will be rejected. There is no limitation to the length of each string. Q must include q_0, therefore Q must include at least one state; there is no limitation in the number of states.

INPUT_ALPHABET:

Input_alphabet: is the equivalent to Σ in the definition of a Turing machine. Everything between the "input_alphabet:" and the "tape_alphabet:" keywords are considered as a part of Σ. Letters in Σ must only be in the ascii printable character set. White space is used to separate letters. Each letter must be separated from one another. Each character must be unique. Each character in Σ must be in Γ but the character in B must not be in Σ. There are only 5 reserved characters which are: \ [ ] < >

TAPE_ALPHABET:

Tape_alphabet: is the equivalent to Γ in the definition of a Turing machine. Everything between the “tape_alphabet:” and the “transition_function:” keywords are considered as a part of Γ. Letters in Γ must only be in the ascii printable character set. White space is used to separate letters. Each letter must be separated from one another. Each character must be unique. There are only 5 reserved characters which are: \ [ ] < >. Characters included in Σ and B must be included in Γ.

TRANSITION_FUNCTION:

Transition_function: is a composite of multiple functions which are the equivalent to δ in the definition of a Turing machine. Everything between the “transition_function:” and the “initial_state:” keywords are considered as a part of the section. Each function has 5 components, each of which are mandatory and each have specific rules.

1. The current state. This is the state that the Turing Machine is in when reading a character from the tape.
   RULES: State must be in Q and must not be an element of F.

   NOTE: current state must not be an element of F because you can not transition out of a final state.

2. The read letter. This is the letter that the Turing Machine is reading.
   RULES: Letter must be in Γ.

3. The next state. This is the state that the Turing Machine will go to if the current state and the read letter match the function.
   RULES: State must be in Q.

4. The write letter. This is the character that the Turing machine will write the the tape if the current state and the read letter match the function.
   RULES: Letter must be in Γ.

5. Move tape. This is the direction that the Turing machine will move the tape head if the current state and the read letter match the function and the write letter has already been written.
   RULES: Letter must be a ‘R’ or ‘L’ in uppercase or lowercase representing right and left respectively.

INITIAL_STATE:

Initial_State: is the equivalent to q_0 in the definition of a Turing machine. q_0 must only contain one state which must be between the “initial_state:” and the “blank_character:” keywords. The state must be separated by blank space. The state is a string and can be composed of uppercase and lowercase letters, digits, or underscores. All other symbols are illegal and will be rejected. The state is case sensitive and must be included in Q.

BLANK_CHARACTER:

Blank_Character: is the equivalent to B in the definition of a Turing machine. B must only contain one letter which must be between the "blank_character:" and the "final_states:" keywords. The letter must be separated by blank space. The letter must only be in the ascii printable character set. There are only 5 reserved characters which are: \ [ ] < >. The letter is case sensitive and must be included in Γ.

FINAL_STATES:

Final_states: is the equivalent to F in the definition of a Turing machine. Everything between the "final_states:" keyword and the end of the file are considered as a part of F. Each state must be separated by blank space. Each state is a string and can be composed of uppercase and lowercase letters, digits, or underscores. All other symbols are illegal and will be rejected. Each state is case sensitive and must be unique. All non-unique states in F will be rejected. There is no limitation to the length of each string. F must be included in Q. There is no limitation in the number of states, neither minimum or maximum.

Figure 4-2-2 Valid Definition File

```
This Turing machine accepts the language of one or more a's followed by the same number of b's.

STATES: s0 s1 s2 s3 s4

INPUT_ALPHABET: a b

TAPE_ALPHABET: a b X Y -

TRANSITION_FUNCTION:

s0 a s1 X R
s0 Y s3 Y R
s1 a s1 a R
s1 b s2 Y L
s1 Y s1 Y R
s2 a s2 a L
s2 X s0 X R
s2 Y s2 Y L

INITIAL_STATE: s0

BLANK_CHARACTER: -

FINAL_STATES: s4
```

Figure 4-2-3 Erroneous Definition File 1

STATES:	States declared in δ, q_0, and F not included in Q
INPUT_ALPHABET: a b	
TAPE_ALPHABET: a X Y -	‘b’ is declared in Σ but not in Γ
TRANSITION_FUNCTION:	
s0 a s1 X R	
s0 Y s3 Y	Function is missing a move direction
s1 a s1 a R	
s1 b s2 Y L	
s1 Y s1 Y R	
s2 a s2 a L	
s2 X s0 X R	
s2 Y s2 Y L	
INITIAL_STATE: s0	
BLANK_CHARACTER: -	
FINAL_STATES:	Not an error that no final states are defined

Figure 4-2-3 Erroneous Definition File 2

STATES: s0 s1 s2 s3 s4	
INPUT_ALPHABET: a b	
TAPE_ALPHABET: a b X Y	Character in B not included in Γ
TRANSITION_FUNCTION:	
s1 X R	Current state and read state not included. May cause cascading errors throughout the transition_function: section of the document
s0 Y s3 Y R	
s1 a s1 a R	
s1 b s2 Y L	
s1 Y s1 Y R	
s2 a s2 a L	
s2 X s0 X R	
s2 Y s2 Y L	
	No initial state declared
INITIAL_STATE:	
BLANK_CHARACTER: -	Is an error that final_states: keyword is not included

4.3 Input String File

The input string file is a plain text file with an extension "*.str*". Each line of an input string file is considered a different input string. A string is added to the input string list if it is valid. To be considered a valid input string it has to pass a series of tests. All characters in the string must be in sigma (Σ, the input alphabet) or must be the empty string "\". If this test fails display "*ERROR: <input string> contains letters not in Σ*". There must be no extra white space in the input string. If this test fails display "*ERROR: <input string> contains extra white space*". The input string must not be a duplicate of another input string. If this test fails display "*ERROR: <input string> is a duplicate*"

Figure 4-3 Input String File Examples
In the case that: Σ = { a, b }
Valid input string aaaaaabbbababababa
Valid input string, is the empty string \
Invalid, contains characters not in sigma aacb
Invalid, contains extra blanks aabb\t
Invalid, contains characters not in sigma \\

5. Operation

5.1 Invocation

5.1.1 Command Line

To open a Turing Machine one must execute the application with the name of the Turing machine without the extension “*.def*”. If there is an input file matching the Turing machine then it also is inputted without the extension “*.str*”.

Figure 5-1-1 Example Application Execution
User wants to input myFirstTM.def and myFirstTM.str
./TM myFirstTM

Once executed, the application reads and loads the definition file. If the definition is valid, it attempts to read the input string file. If the input string file exists then the valid it is also read and the valid strings are loaded into the application. After these operations are complete then the application prompts for a command (See 5.1.3)

5.1.2 Configuration Settings

To application has two manipulatable settings, the number of transitions to perform and the truncation of the instantaneous description.

The number of transition setting lets the user perform multiple transitions on an input string at a time by changing the maximum. The default is set to 1 transition and if the setting is changed it isn’t saved upon closing the application.

NOTE: For invocation and input information see section 5.2.7

The truncation setting changes maximum number of characters to the left and to the right of the tape head to display at a time. The default is set to 32 characters and if the setting is changed it isn’t saved upon closing the application. If there is more than the specified number of characters in the instantaneous description to the right or left of the tape head, then the string is truncated with a "<" on the left or ">" to the right.

NOTE: For invocation and input information see section 5.2.8

Figure 5-1-2 Truncation examples
Truncation set to 6, more than 6 characters to the right of the tape head and 0 on the left: 0. [s0]aabaab>
Truncation set to 6, more than 6 characters to the right and left of the tape head: 10. <baabaa[s4]baabaa>
Truncation set to 6, more than 6 characters to the left of the tape head and 0 on the right: 1000. <XYYXYY[s5]

5.1.3 Opening Turing Machine

After a Turing machine has been successfully loaded into the application, the user must be presented with the command prompt "*Command:* ". There are 11 possible commands: delete, exit, help, insert, list, quit, run, set, show, truncate, and view. Each command is invoked with a single character in upper or lower case (see section 5.2 - commands for details on each command). If anything other than the possible commands are inputted, display *"ERROR: invalid command, enter h for a list of commands*". There must be no message on a blank response. After an invalid or blank response, the application must return back to the command prompt.

5.2 Commands

5.2.1 Help User

Invocation: h or H when application is prompting for a command

The help user command will describe the list of possible commands to the user and a very brief description of each purpose. After invocation and display, application must return to the TM command prompt.

Figure 5-2-1 Help Example Display

(D)ELETE:	Delete input string from list
E(X)IT:	Exit application
(H)ELP:	Displays Commands
(I)NSERT:	Insert input string into list
(L)IST	List input strings
(Q)UIT	Quit operation of TM on input string
(R)UN:	Run TM on input string
S(E)T	Set maximum number of transitions to perform
SHO(W)	Show status of application
(T)RUNCATE:	Set truncation length of instantaneous descriptions
(V)IEW:	View TM

5.2.2 Show Status

Invocation: w or W when application is prompting for a command

The show status command will display details related to the the application itself, the configuration settings, and status of the application.

Details related to the application to be displayed must include: course, semester, year, instructor, author, and application version.

Details related to configuration settings to be displayed must include: the maximum number of transitions and the maximum number of cells to the left and right of the tape head to be displayed in an instantaneous description.

Details related to the status of the application include:

The Turing machine name, which is given when the file name is passed upon running the application. This must be displayed without the ".def" extension

The status of the TM, choose one: {"*TM has not been run on an input string*", "*TM is currently running on an input string*", "*TM has completed operation on an input string*"}

If the TM is running on an input string display the input string and the number of transitions performed up to now on that input string.

If TM has completed operation on input string display last input string and whether it was accepted, rejected or terminated by user and display the number of transitions performed up to that point.

After invocation and display, application must return to the TM command prompt.

Figure 5-2-2 Show Example Display

```
COURSE: Computer Science 322: Software Engineering

SEMESTER: Spring

YEAR: 2017

INSTRUCTOR: Neil Corrigan

AUTHOR: Devan Farrell

VERSION: 0.0.1


CONFIGURATION SETTINGS:

        Max number of transitions: 1

        Max number of cells to the left and right of the tape head: 32


TM name: ANBR


Status of TM:

        TM has completed operation on an input string

        Input string ‘aab’ was not accepted or rejected in 15 transitions
```

5.2.3 View Turing Machine

Invocation: v or V when application is prompting for a command

Display definition of the currently loaded TM on the monitor in a form that is readable by the user. Format should be similar to traditional formatting and not to to input file. After invocation and display, application must return to the TM command prompt.

Figure 5-2-3 View Example Display

This is a description

M	=	{ Q, Σ, Γ, δ, q_0, B, F }
Q	=	{ s0, s1, s2, s3, s4 }
Σ	=	{ a, b }
Γ	=	{ a, b, Y , X, - }
δ(s0, a)	=	{ s1, X, R }
δ(s0, Y)	=	{ s3, Y, R }
δ(s1, a)	=	{ s1, a, R }
q_0	=	s0
B	=	-
F	=	{ s4 }

5.2.4 List Input Strings

Invocation: l or L when application is prompting for a command

Displays the list of valid input strings that are currently being stored. Each input string display must be on its own line and numbered sequentially starting at '1'. These numbers associated with the strings are the numbers used as inputs in the run and delete commands. If the list is empty a message must display stating "*The list is empty!*". After invocation and display, application must return to the TM command prompt.

Figure 5-2-4 List Example Display

```
[1] aaa

[2] bbb

[3] \

[4] aaab
```

5.2.5 Insert Input String

Invocation: i or I when application is prompting for a command

The insert command allows the user to create a string and input it into the stored list. Upon invoking the command, the user will type a string and press return to confirm it. The string will go through the same validation as the input string file; the string must not already be in the list, there must not be extra white space, all characters must be in the input alphabet (Σ). If the string passes all the validation requirements it is appended to the end of the list. If the string is already in the list, display "*ERROR: duplicate string*" and discard the string. If the string contains a character not in the input alphabet (Σ), display "*ERROR: string contains invalid character(s)*" and discard the string. If the string contains extra white space, display "*ERROR: string contains extra white space(s)*" and discard the string. After any input, valid or invalid, and corresponding application action, application must return to the TM command prompt.

5.2.6 Delete Input String

Invocation: d or D when application is prompting for a command

Allows user to delete an input string from the input string list. The string to be deleted must be chosen from the list by number and not by typing the string. Deleting a string will cause a renumbering of any strings in the list following that string. It is possible to delete a string that the TM is currently evaluating. If that takes place the TM should continue to run without any issue on that string but the string should be deleted from the list. If the input is anything but a positive integer display "*Error: input was not a positive integer*". If the input was a positive integer but doesn't correspond to an input string display “*ERROR: string is not in list*”. After any input, valid or invalid, and corresponding application action, application must return to the TM command prompt.

NOTE: to see the list of input strings and their numbering run the **list** command.

5.2.7 Set Transitions

Invocation: e or E when application is prompting for a command

Allows user to change the setting for maximum number of transitions to perform at a time during operation of the Turing Machine on an input string. The value must be a positive integer and the prompt should include the current setting. If the input is anything but a positive integer (e.g. 1.5, ab, 0 , -1) display “*Error: input was not a positive integer*”. After any input, valid or invalid, and corresponding application action, application must return to the TM command prompt. Command must be cancelable to leave the setting unchanged by pressing enter/return without entering an input which must return the application to the command prompt.

5.2.8 Truncate Instantaneous Descriptions

Invocation: t or T when application is prompting for a command

Allows user to change the setting for the maximum number of cells to the left and right of the tape head to display in an instantaneous description during operation of the TM on an input string. The value must be a positive integer and the prompt should include the current setting. If the input is anything but a positive integer (e.g. 1.5, ab, 0 , -1) display “*Error: input was not a positive integer*”. Command must be cancelable to leave the setting unchanged by pressing enter/return without entering an input which must return the application to the command prompt.

5.2.9 Run Turing Machine

Invocation: r or R when application is prompting for a command

The run command has two functions. If the TM is not currently operating on an input string, the run command prompts for an input string to operating on. The string to be operated upon must be chosen from the list by number and not by typing the string. If the input is anything but a positive integer display “*Error: input was not a positive integer*”. If the input was a positive integer but doesn’t correspond to an input string display “*ERROR: string is not in list*”. After an invalid input, and the corresponding error message application must return to the TM command prompt. If the input string is valid the instantaneous description is displayed. If the TM is currently operating on an input string, the application makes a transition on it.

5.2.10 Quit Turing Machine

Invocation: q or Q when application is prompting for a command

Allows user to quit operation on an input string before completion. If the TM is currently operating on a string a message should be displayed stating the input string, the fact that the TM has has not accepted or rejected the input string, and the number of transitions performed on the string before quitting operation.

Figure 5-2-5 Quit Example Display

```
‘aab’ not accepted or rejected in 1 transition
```

If the TM is not currently operating on an input string display “*ERROR: nothing to quit*”. After invocation and appropriate display, application must return to the TM command prompt.

5.2.11 Exit Application

Invocation: x or X when application is prompting for a command

The exit command exits the application, returning the user the the shell command line.

NOTE: for further details on existing the application see section 5.3

5.3 Termination

5.3.1 Closing Turing Machine

Upon invoking the exit application command, the application begins to perform cleanup.

First, it determines if the input list has been modified in any way. Modifications include: discarding invalid strings, insertions, and deletions.

If the list has been modified the application makes a single attempt to replace the input file with the contents of the input string list. If it is successful, display "*Overwrite successful!*". If it unsuccessful, display "*Overwrite failed*".

If the list has not been modified display nothing and move on with cleanup.

All dynamic memory is then freed and the application terminates.

References

Figure 2-1

Wvbailey. Mechanical Turing Machine. Digital image. *Wikipedia*. Free Software Foundation, 4 Aug. 2006. Web. 27 Mar. 2017.