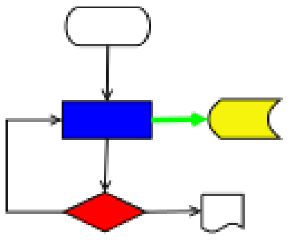
VICI



VISUAL CHART INTERPRETER

Use Cases

Publication History

Date	Who	What Changes
4 October 2012	Brenton Ross	Initial version.

Table of Contents

1 Introduction

This is part of the system design document for the VICI project.

1.1 Scope

This document covers the use cases for the project. There will be use cases for all actions that the user would normally take during the development of a script.

1.2 Overview

The detailed design includes:

- Interface Stubs: A framework of facade classes for the modules.
- Use Case Descriptions: A description of how a user is expected to interact with the application. (This document.)
- Application Design: The classes and their relationships.
- User Interface Design: The design and layout of the graphical components of the system.
- Persistent Storage Design: The specifications for the XML files used to store configuration and scripts.

1.3 Audience

This document is intended to be used by the designers and developers, and later the maintainers, of the VICI project.

2 Use Case Descriptions

This section describes how a user will interact with the application programs.

Every Responsibility listed in the Architecture document should be represented by at least one Use Case.

2.1 Administration Use Case

This section describes the interactions with the administration program that are used to add commands to the VICI system.

2.1.1 UC-100: Open the Administration Program

Version
Application design. Increment #3.

Goal
To demonstrate that the vici-adm program is operational. This use case will be a precondition for subsequent use cases.

Summary
<diagram goes here>

Actors
An administrator or someone with good UNIX skills.

Stakeholders
Users of VICI.

Preconditions
1. VICI has been installed.

Triggers
There is some requirement to add or modify a command.

Normal Sequence of Events
1. The user opens the “System Tools” application menu.
2. The user starts the vici-adm program.
3. The vici-adm program opens a window displaying various text entry fields for entering the command name, its description, the command for help, and the EBNF for the command's options and parameters.
4. *Refer to other use cases.*
5. The user presses the “Exit” menu or the “Exit” icon on the toolbar.
6. The vici-adm program verifies that all changes have been saved.
7. The vici-adm program terminates.

Alternative Sequence of Events

1. On exit the program determines that some changes have not been saved.
2. The program displays a warning dialog allowing the user to save and exit, exit without saving or cancel the exit action.
3. The user selects one of the above options.
4. The program either continues, saves and exits, or exits without saving, according to the selected option.

Postconditions

The command database is in a consistent state.

Business Rules

Responsibilities

None.

Notes

No explicit responsibilities are covered by this use case, however other administration use cases will use this one for their opening and closing actions.

Author

Brenton Ross

2.1.2 UC-101: Prepare a Command

Version

Application design. Increment #3.

Goal

Add a command to the VICI command database.

Summary

<diagram goes here>

Actors

An administrator or experienced UNIX user.

Stakeholders

VICI users.

Preconditions

Use case UC-100 has been used to open the vici-adm program.

Triggers

A user has requested that a new command be added to the database.

Normal Sequence of Events

1. *Refer UC-100 for opening vici-adm.*
2. Enter the command.
3. The system checks to confirm a new command.
4. Enter a short description of the command.
5. Enter the command to use for help on the command.
6. Enter the EBNF of the command options.
7. *Refer UC-102 for the system response.*
8. *Refer UC-100 for closing vici-adm*

Alternative Sequence of Events

Postconditions

A new command is added to the VICI command database.

Business Rules

Responsibilities

T1.1, T1.2, T1.3, T1.4.

Notes

Author

Brenton Ross

2.1.3 UC-102: Validate a Command

Version

Application design. Increment #3.

Goal

Confirm that the syntax entered for a command's options and parameters is valid.

Summary

<diagram goes here>

Actors

An administrator or experienced UNIX user.

Stakeholders

Preconditions

1. UC-101 has been used to enter the details for a new command.

Triggers

Normal Sequence of Events

1. *Refer to UC-101 and UC-103 for data entry*.
2. The system reports an error in the EBNF.
3. The user corrects the EBNF.
4. The system validates the EBNF.
5. The system displays the syntax diagram for the EBNF.
6. The system enables the “Save” button.
7. The user presses the “Save” button.
8. The system adds the new command to the VICI command database.
9. *Refer to UC-101 for completion steps.*

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T1.5, T1.6, T1.9, T1.10.

Notes

Author

Brenton Ross

2.1.4 UC-103: Edit a Command

Version

Application design. Increment #3.

Goal

Modify the options or parameters of a command, or amend its description or help command.

Summary

<diagram goes here>

Actors

An administrator or experienced UNIX user.

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer UC-100 for starting vici-adm.*
2. Enter the command name.
3. The system retrieves the command from the database.
4. The system displays the values for the syntax, description and help commands.
5. The user updates the appropriate fields.
6. *Refer to UC-102 for validation.*

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T1.7, T1.8.

Notes

Author

Brenton Ross

2.1.5 UC-104: Select a Command

Version

Application design. Increment #3.

Goal

Allow the user to select a command to edit for the VICI command database.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-100 for opening vici-adm.*
2. The user presses the "..." button adjacent to the command field.
3. The system displays a list of commands that have been prepared.
4. The user selects a command, and presses OK.
5. The system places the selection in the command field
6. *Refer to other use cases for remaining actions.*

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T1.1

Notes

Author

Brenton Ross

2.1.6 UC-105: Create an Alias Command

Version

Application design. Increment #3.

Goal

Create a command that actions an alternative command usually with some fixed set of parameters and options.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The command has been previously prepared.

Triggers

A command is often used with the same set of options and parameters.

Normal Sequence of Events

1. *Refer UC-100 for starting vici-adm.*
2. Enter the command name.
3. The system retrieves the command from the database.
4. The system displays the values for the syntax, description and help commands.
5. Enter the name for an alias.
6. The system confirms that the name is not a duplicate of another alias or a command.
7. The user enters the commands and options that are unchanging for the alias.
8. The system confirms that the options are valid for the command.
9. The user presses “Save”.
10. The system adds the alias to the command in the VICI command database.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T1.11

Notes

Author

Brenton Ross

2.2 *Editing Use Cases*

These use cases refer to the VICI editor that is used to create a script.

2.2.1 UC-106: Starting the Editor

Version

Application design. Increment #1.

Goal

Start the editor so that it is ready to create or modify a VICI script.

Summary

<diagram goes here>

Actors

A user wanting to create a script.

Stakeholders

Preconditions

The VICI system has been installed.

Triggers

The need to create or modify a script.

Normal Sequence of Events

1. The user navigates to the “System Tools” menu.
2. The user clicks on the vici-ed program.
3. The system launches the vici-ed program.
4. The system creates a new empty diagram.
5. The system displays a set of flowchart symbols.
6. The system displays a palette of text attributes.
7. *Refer to other use cases for the details of preparing and modifying a script.*

Alternative Sequence of Events

Postconditions

The vici-ed program is ready to be used.

Business Rules

Responsibilities

T3.1, T3.2, T3.4, T9.1

Notes

Author

Brenton Ross

2.2.2 UC-107: Single Command Script

Version

Application design. Increment #1.

Goal

Create a simple script that has a single command.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-106 for starting vici-ed.*
2. The user clicks on the command box in the symbol palette.
3. The system marks the command box as the default symbol.
4. The user clicks on the canvas.
5. The system draws a command box centred on the location of the mouse click.
6. The system shows the command selection window.
7. The user selects a command.
8. The system labels the command box with the name of the command.
9. The system shows the syntax chart for the command.
10. The user enters the parameters and options for the command.
11. *Refer to UC-108 for saving a script.*

Alternative Sequence of Events

Postconditions

A script is ready for saving and installing.

Business Rules

Responsibilities

T3.2, T3.3, T3.5, T4.1, T4.2, T4.3, T4.5, T6.1, T6.2, T6.4, T6.5

Notes

Author

Brenton Ross

2.2.3 UC-108: Save a Script

Version

Application design. Increment #1.

Goal

The script is saved prior to further editing or installation.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-106 and UC-107 for actions prior to this use case.*
2. The user presses the “Save” button.
3. The system records the user's name in the script.
4. The system adds a signature to the script.
5. The system saves the script as an XML file.

Alternative Sequence of Events

Postconditions

The script has been saved, but is still in the editor for further work, or for installation.

Business Rules

Responsibilities

T3.9, T3.12, T23.1

Notes

Author

Brenton Ross

2.2.4 UC-109: Three Command Script

Version

Application design. Increment #1.

Goal

Create a script which runs three commands in sequence.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *The user performs Use Case 107 three times.*
2. The user selects the unconditional flow of control arrow on the symbol palette.
3. The system displays it as the default symbol.
4. The user clicks on the first command box, and holding down the mouse button, drags to the second command box.
5. The system draw an unconditional flow arrow between the boxes.
6. The user clicks on the second command box, and holding down the mouse button, drags to the third command box.
7. The system draw an unconditional flow arrow between the boxes.
8. *Refer to UC-108 for saving the script.*

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

None. This use case sets up the situation for other use cases that demonstrate responsibilities.

Notes

Author

Brenton Ross

2.2.5 UC-110: Reposition a Symbol

Version

Application design. Increment #1.

Goal

Change the location of a symbol in the flowchart diagram..

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-109 for setting up the use case.*
2. The user clicks on a command box.
3. The system indicates that the command box has been selected by highlighting it (in some way).
4. The user drags the mouse to a new location.
5. The system repositions the command box to the mouse coordinates.
6. The system maintains the connections between the boxes.
7. The user releases the mouse.
8. The user clicks somewhere else.
9. The system removes the highlighting indicating the symbol has been selected.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T3.5, T3.6, T3.7

Notes

Author

Brenton Ross

2.2.6 UC-111: Reposition Several Symbols

Version

Application design. Increment #1.

Goal

Change the position of several symbols in a flowchart diagram.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-109 for setting up the use case.*
2. The user clicks on a command box.
3. The system indicates that the command box has been selected by highlighting it (in some way).
4. The user holds down the control key and clicks on another command box.
5. The system indicates that both symbols have been selected.
6. The user drags the mouse to a new location.
7. The system repositions the command boxes to the mouse coordinates.
8. The system maintains the connections between the boxes.
9. The user releases the mouse.
10. The user clicks somewhere else.
11. The system removes the highlighting indicating the symbols have been selected.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T3.5, T3.6, T3.7

Notes

Author

Brenton Ross

2.2.7 UC-112: Remove a Command

Version

Application design. Increment #1.

Goal

Remove a command from the script.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. *Refer to UC-109 for setting up the use case.*
2. The user clicks on a symbol.
3. The system indicates that the symbol has been selected by highlighting it.
4. The user presses “Delete”.
5. The system presents a confirmation dialog.
6. The user selects OK.
7. The system removes the symbol, and for command boxes it also removes connecting symbols.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T3.5, T3.8.

Notes

Author

Brenton Ross

2.2.8 UC-113: Reload a Script

Version

Application design. Increment #1.

Goal

Load an existing script for further editing..

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script has been saved.

Triggers

Normal Sequence of Events

1. *Refer to UC-106 for starting the editor.*
2. The user selects "Open".
3. The system displays a file selection dialog showing the user's scripts.
4. The user selects a script and presses "Open" on the dialog.
5. The system loads the XML file.
6. The system confirms the signature is valid for this user.
7. The system makes a backup copy of the script.
8. The system displays the flowchart for the script.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T3.10, T3.13, T23.2.

Notes

Author

Brenton Ross

2.2.9 UC-114: Automatic Layout

Version

Application design. Increment TBA.

Goal

Allow the system to layout the flowchart for the script..

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The user has a script loaded.

Triggers

Normal Sequence of Events

1. The user presses the “Auto Layout” option.
2. The system records the current location of the symbols.
3. The system generates a new layout.
4. The system displays the new layout.

Alternative Sequence of Events

1. The user presses the “Revert” option.
2. The system reverts to the saved layout.
3. The system displays the previous layout for the flowchart.

Postconditions

Business Rules

Responsibilities

T8.1, T8.2, T8.3, T8.4

Notes

Author

Brenton Ross

2.2.10 UC-115: Add Comments to a Script

Version

Application design. Increment #1.

Goal

The user wishes to annotate the script with comments and other text.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The script is open for editing.

Triggers

Normal Sequence of Events

1. The user selects the attributes for the text.
2. The system indicates the style of text that has been selected.
3. The user selects the “Text” option.
4. The user clicks on the canvas.
5. The system displays a text insertion cursor at the mouse coordinates.
6. The user enters the desired text.
7. The user saves the script.
8. The user reloads the script.
9. The system display the script flow chart and the comment text.
10. The user selects some of the text.
11. The system highlights the selection.

12. The user selects attributes for the selected text.
13. The system redisplays the text with the new attributes.
14. The user selects some of the text and presses “Delete”.
15. The text is redisplayed without the selected text.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T9.1, T9.2, T9.3, T9.4, T9.5, T9.6

Notes

Author

Brenton Ross

2.2.11 UC-116: Show Help

Version

Application design. Increment #3.

Goal

Display the help available for a command.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The script is open for editing.

Triggers

Normal Sequence of Events

1. The user selects a command.
2. The user opens the help window.
3. The system displays the prepared help text for the command.
4. The user presses the “Man” option.

5. The system displays the man page for the command in the help window.
6. The user presses the “Info” option.
7. The system displays the Info page for the command in the help window.

Alternative Sequence of Events

1. The user enters a non-prepared command.
2. The user opens the help window.
3. *As above.*

Postconditions

Business Rules

Responsibilities

T5.1, T5.2, T5.3, T5.4, T5.5

Notes

Author

Brenton Ross

2.2.12 UC-117: Use of Variables

Version

Application design. Increment #1.

Goal

Create a command which references a variable.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A new script is ready for editing.

Triggers

Normal Sequence of Events

1. The user places a symbol representing a constant on the canvas.

2. The user enters the text value of the constant.
3. The user places a symbol representing a variable on the canvas.
4. The user enters the name of the variable.
5. The system adds the variable name to the list of defined variables.
6. The user connects the constant to the variable using the the stdout data pipe.
7. The user places a symbol representing a variable on the canvas.
8. The user enters the name of the variable.
9. The user indicates that it is a drag-n-drop field.
10. The system enables the name field.
11. The user enters the display name of the variable.
12. The user places a command box on the canvas and connects the variable to it using the unconditional flow symbol.
13. The system displays the syntax chart for the command.
14. The user adds the variable name as a parameter to the command.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T6.1, T6.2, T6.3, T6.4, T6.5,
T24.1, T24.2, T24.7.

Notes

Author

Brenton Ross

2.2.13 UC-118: Create a Function

Version

Application design. Increment #1.

Goal

Split out a set of commands into a separate function.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Some part of the script is to be reused.

Normal Sequence of Events

1. *Refer to UC-109 for setting up the use case.*
2. The user selects the commands to be grouped into a function.
3. The system indicates that the commands have been selected.
4. The user presses the “Create Function” option.
5. The system prompts for a name for the function.
6. The user enters a function name and presses OK.
7. The system replaces the selected commands with a single function box.
8. The system creates a second flowchart diagram containing the selected commands.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T7.1, T7.2

Notes

Author

Brenton Ross

2.2.14 UC-119: Menu Entries

Version

Application design. Increment #1.

Goal

Make a function accessible from the menu on the run-time program.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A function has been created.

Triggers

Normal Sequence of Events

1. The user selects a function flow chart.
2. The system indicates that the function has been selected.
3. The user presses the “Menu” option.
4. The system prompts for a menu name, using the function name as its default.
5. The user enters the name for the menu item.
6. The system registers the function against the menu item and places the menu item in the top level menu.
7. The user moves the menu item to another menu item.
8. The system relocates the menu item as a sub menu.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T7.3, T7.4

Notes

Author

Brenton Ross

2.2.15 UC-120: Search Prepared Commands

Version

Application design. Increment #4.

Goal

Search for a key word in the help text for the prepared commands.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The editor is running.

Triggers

Normal Sequence of Events

1. The user presses the “Search” option.
2. The system provides alternative areas to search.
3. The user selects Prepared Commands Help Text
4. The user enters a word to search for.
5. The system scans the prepared commands and returns references to those that have the search word.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T11.1

Notes

Author

Brenton Ross

2.2.16 UC-121: Search using apropos

Version

Application design. Increment #4.

Goal

Search for a key word in the short descriptions of man pages.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user presses the “Search” option.
2. The system provides alternative areas to search.
3. The user selects Man Pages
4. The user enters a word to search for.
5. The system runs the apropos command and returns the result in the search window.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T11.2

Notes

Author

Brenton Ross

2.2.17 UC-122: Configure Symbols

Version

Application design. Increment #7.

Goal

Allow the user to modify the colours and patterns used for the symbols.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user selects a symbol for command boxes.
2. The system indicates that the symbol is now the default symbol.
3. The user selects a colour for the symbol.
4. The system sets the colour for this symbol.
5. The user selects a texture for the symbol.
6. The system sets the texture for the symbol.
7. The user selects a symbol for stderr.
8. The system highlights the symbol.
9. The user selects a pattern for the lines.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T17.1, T17.2, T17.3, T17.4.

Notes

Author

Brenton Ross

2.2.18 UC-123: Configure Windows

Version

Application design. Increment #7.

Goal

Allow the user to arrange the windows according to their preferences.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

The vici-ed program is running.

Triggers

Normal Sequence of Events

1. The user drags a window handle outside the program's main window.
2. The system moves the window to a new separate window.
3. The user drags the window handle of a separate window onto the main window.
4. The system moves the window to the main window, and redoes the window layout.
5. The user moves a window handle to a new location within the main window.
6. The system repositions the windows.
7. The user resizes a window.
8. The system redoes the window layout.
9. The user presses the "Default Layout" option.
10. The system resets the window layout.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T18.1, T18.2, T18.3, T18.4.

Notes

Author

Brenton Ross

2.2.19 UC-124: Demonstrate cd Command

Version

Application design. Increment #1.

Goal

Demonstrate the cd command to confirm that it changes the working directory.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A simple script has been created.

Triggers

Normal Sequence of Events

1. The user inserts a command box with the “cd” command to a test directory.
2. The user adds a command that references a local file in the test directory.

Alternative Sequence of Events

Postconditions

The run time is able to access the file in the test directory.

Business Rules

Responsibilities

T19.1

Notes

Author

Brenton Ross

2.2.20 UC-125: Demonstrate for-each Command

Version

Application design. Increment #1.

Goal

Demonstrate the operation of the for-each command that processes a set of commands for each symbol of its input.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A simple script has been created.

Triggers

Normal Sequence of Events

1. The user adds a command that creates a sequence of white space separated symbols.
2. The user adds a for-each command, and connects it via a stdout pipe from the command described in item 1.
3. The user adds a command box that takes a input from stdin, and connects it via a stdin pipe from the for-each command.
4. An unconditional flow connection is made from the command of item 3 back to the for-each command.

Alternative Sequence of Events

Postconditions

The run-time is able to process each symbol.

Business Rules

Responsibilities

T19.3

Notes

Author

Brenton Ross

2.2.21 UC-126: Demonstrate for-each-line Command

Version

Application design. Increment #1.

Goal

Demonstrate the operation of the for-each-line command that processes a set of commands for each line of its input.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions
A simple script has been created.

Triggers

Normal Sequence of Events

1. The user adds a command that creates a sequence of lines.
2. The user adds a for-each-line command, and connects it via a stdout pipe from the command described in item 1.
3. The user adds a command box that takes a input from stdin, and connects it via a stdin pipe from the for-each-line command.
4. An unconditional flow connection is made from the command of item 3 back to the for-each command.

Alternative Sequence of Events

Postconditions
The run-time is able to process each line.

Business Rules

Responsibilities
T19.2.

Notes

Author
Brenton Ross

2.2.22 UC-127: Demonstrate Background Process

Version
Application design. Increment #1.

Goal
Create a script which demonstrates a background process..

Summary
<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script containing a background command and a command which generates a signal to terminate the process.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T13.7, T19.5, T24.4

Notes

Author

Brenton Ross

2.2.23 UC-128: Demonstrate Pipes

Version

Application design. Increment #1.

Goal

Create a script which demonstrates the use of pipes.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script which includes the standard pipes between commands, and also uses a named pipe.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T19.4, T20.1.

Notes

Author

Brenton Ross

2.2.24 UC-129: Demonstrate File Handling

Version

Application design. Increment #1.

Goal

Create a script which demonstrates how files can be manipulated.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script which includes connecting standard pipes to both permanent and temporary files. The script shall demonstrate both append and overwrite capability. The script shall also demonstrate an in-line file with embedded variables.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities
T20.2, T21.1, T22.1.

Notes

Author
Brenton Ross

2.2.25 UC-130: Demonstrate Arithmetic Operations

Version
Application design. Increment #1.

Goal
Create a script which demonstrates the range of arithmetic available in the VSL.

Summary
<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events
Create a script which uses addition, subtraction, multiplication, division and remainder operations on a variety of variables.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T24.3.

Notes

Author

Brenton Ross

2.2.26 UC-131: Demonstrate Exit Status & Process Id

Version

Application design. Increment #1.

Goal

Create a script that demonstrates the use of the exit status and process id built in variables.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script that has a background process, and use its process id to send it a signal. Display the exit status of the process when it completes.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T24.4, T24.5.

Notes

Author

Brenton Ross

2.2.27 UC-132: Demonstrate Function Parameters

Version

Application design. Increment #1.

Goal

Create a script that demonstrates passing parameters to a function.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script that has a function defined and call it passing a parameter to it with a different value each time.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T24.6

Notes

Author

Brenton Ross

2.2.28 UC-133: Demonstrate Mutexes

Version

Application design. Increment #1.

Goal

Create a script the demonstrates the use of mutexes to synchronize access to a variable.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a function which runs in the background and accesses a variable that is also used by the main function. Include a mutex variable to protect the access so that only one thread can access it at a time.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T24.8.

Notes

Author

Brenton Ross

2.2.29 UC-134: Demonstrate Semaphores

Version

Application design. Increment #1.

Goal

Create a script that demonstrates how a semaphore can be used to synchronize processing between threads.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Create a script with a background function and which includes a semaphore that prevents access until its value is incremented by the main thread. Include a command in the main thread to increment the semaphore.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T24.9.

Notes

Author

Brenton Ross

2.3 Command Searching Use Cases

2.3.1 UC-135: Search with Tags

Version

Application design. Increment #4.

Goal

Use tags to find commands that perform a desired function.

Summary

<diagram goes here>

Actors

Linux desktop user.

Stakeholders

Preconditions

The VICI system is installed.

Triggers

The user wants to include a command but cannot remember what it is called.

Normal Sequence of Events

1. The user open the search window.
2. The user selects some tags and specifies unions and intersections.
3. The system displays the list of matching commands.
4. The user selects a command.

Alternative Sequence of Events

Postconditions

The user has selected a command of interest.

Business Rules

Responsibilities

T12.4, T12.6, T12.7, T31.1

Notes

Author

Brenton Ross

2.3.2 UC-136: Classify Commands

Version

Application design. Increment #4.

Goal

The commands are classified according to the user's own scheme.

Summary

<diagram goes here>

Actors

Linux desktop user.

Stakeholders

Preconditions

1. VICI has been installed.
2. All desired commands have been prepared.

Triggers

The user's desire to classify the commands.

Normal Sequence of Events

1. The user opens the search window.
2. The user selects a command.
3. The system highlights the matching tags in the tag list.
4. The user adds a new tag
5. The system associates the command with the tag.
6. The user selects another tag and indicates that it is the parent type.
7. The user presses the 'save' button.
8. The system saves the tag database.

Alternative Sequence of Events

Postconditions

The tag database is updated to include the new relationships.

Business Rules

Responsibilities

T12.1, T12.2, T12.3, T12.4

Notes

Author

Brenton Ross

2.4 Testing Use Cases

These use cases refer to the user of the program testing the scripts they have written.

2.4.1 UC-137: Observe Script Operation

Version

Application design. Increment #2.

Goal

Observe the operation of a script to verify that it behaves as expected.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script is being edited.

Triggers

Normal Sequence of Events

1. The user selects the “Test” option.
2. The system builds an execution model, and indicates that it is done by placing an execution cursor at the start of the main thread.
3. The user selects a delay interval, and presses “Run”.
4. The system steps through the script with a pause after each command.
5. The system indicates which command is next to be executed by placing the execution cursor next to the command symbol.
6. The system displays the values of variables at each step.
7. The system completes the script, and removes the execution cursor.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T13.1, T13.2, T13.4, T13.7, T13.8, T13.9, T13.10, T14.1, T14.2, T14.3, T14.4, T14.5

Notes

This is the basis for the remaining test use cases, all of which will include the above responsibilities.

Author

Brenton Ross

2.4.2 UC-138: Setting Break Points

Version

Application design. Increment #2.

Goal

Stop the script at some point in its execution and examine the state of the system.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script has been written and loaded into the editor.

Triggers

Normal Sequence of Events

1. *Refer to UC-137 for starting the test.*
2. The user selects a command.
3. The system highlights the selected command.
4. The user presses the “Break” option.
5. The system sets a breakpoint at the selected command.
6. The user presses “Run”.
7. The system steps through the script until it comes to the breakpoint.
8. The user examines the variables.
9. The user presses “Continue”.
10. The system steps through the remainder of the script (or until it reaches another breakpoint).

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T13.6

Notes

Author

Brenton Ross

2.4.3 UC-139: Stepping Through Script

Version

Application design. Increment #2.

Goal

Execute the script one command at a time, and examine variables after each step.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script has been loaded into the editor.

Triggers

Normal Sequence of Events

1. *Refer to UC-137 for starting the test.*
2. The user presses the “Step” option.
3. The system executes the next command.
4. The user views the variables.
5. Repeat from item 2 as required.
6. The user presses “Continue”.
7. The system completes execution of the script.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T13.3

Notes

Author

Brenton Ross

2.4.4 UC-140: Verify File Operations

Version

Application design. Increment #2.

Goal

Allow the user to examine files as the script executes.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script which access files has been created and is loaded into the editor.

Triggers

Normal Sequence of Events

1. *Refer to UC-138 for setting a break point.*
2. The user selects a file that is being written to from the list of open files.
3. The system displays the selected file in a new window, positioned at the end of the file.
4. The user steps through the script.
5. The system updates the display with the contents of the file after each step.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T16.1, T16.2.

Notes

Author

Brenton Ross

2.4.5 UC-141: Repeatable Tests

Version

Application design. Increment #2.

Goal

To thoroughly test a script.

Summary

<diagram goes here>

Actors

Linux desktop user.

Stakeholders

Preconditions

1. A script has been written.
2. The script has been loaded into the vici-editor.

Triggers

The user wants to confirm the correct operation of a script.

Normal Sequence of Events

1. The user opens the testing window.
2. The user places the execution cursor at a command.
3. The user places a break point at a subsequent command.
4. The user presses 'Continue'
5. The system executes the commands between the execution cursor and the break point.
6. The user selects 'Save Snapshot As'

7. The system presents a file save dialog.
8. The user enters a snapshot file name
9. The user selects the files that are to be part of the snapshot.
10. The system saves a snapshot of the variables and files.
11. The user performs further tests.
12. The user reloads the snapshot file.
13. The system restores the variables and files to their saved state.
14. The use repeats the tests from the same initial position.

Alternative Sequence of Events

Postconditions

The user has confirmed correct operation of the script.

Business Rules

Responsibilities

T13.5, T14.6, T15.1, T15.3, T15.4, T15.5, T16.1, T16.2

Notes

Author

Brenton Ross

2.5 Installation Use Cases

2.5.1 UC-142: Install a Script

Version

Application design. Increment #5.

Goal

Put entries into the desktop menu system such that a user's script can be run.

Summary

<diagram goes here>

Actors

A Linux desktop user.

Stakeholders

Preconditions

A script has been created.

Triggers

The user has completed testing the script and considers it ready for use.

Normal Sequence of Events

1. The user opens the vici-editor program.
2. The user selects the script, and flags it for installation.
3. The system confirms that the script can be installed.
4. The user confirms.
5. The desktop menus are modified to include a reference to the script.
6. The system indicates that installation completed.
7. The user closes the program.

Alternative Sequence of Events

Postconditions

The script is available from the desktop menus.

Business Rules

Responsibilities

T10.1, T10.2, T10.3

Notes

Author

Brenton Ross

2.5.2 UC-143: Uninstall a Script

Version

Application design. Increment #5.

Goal

Remove a script from the desktop menu system.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

A script has been installed.

Triggers

The script is no longer required.

Normal Sequence of Events

8. Open the vici-editor program.
9. Select the script.
10. Flag it for removal.
11. The system confirms that the script is currently installed.
12. The user confirms removal.
13. The system removes the script from the desktop menus.
14. The program is closed.

Alternative Sequence of Events

Postconditions

The script is no longer displayed on the desktop menus.

Business Rules

Responsibilities

T10.4

Notes

Author

Brenton Ross

2.5.3 UC-144: Schedule a Script

Version

Application design. Increment #6.

Goal

Schedule a script to be run at specific times.

Summary

<diagram goes here>

Actors

Linux desktop user.

Stakeholders

Preconditions

1. A VICI script has been created.

Triggers

The user decides that the script should be run at predetermined times.

Normal Sequence of Events

1. The user opens the vici-editor program.
2. The user opens the cron window.
3. The user selects a script to run.
4. The user enters the time and date for the script.
5. The presses 'Schedule' button.
6. The system updates crontab to run the script.

Alternative Sequence of Events

The user removes a scheduled run.

Postconditions

The script is scheduled to run at the desired time.

Business Rules

Responsibilities

T26.1, T26.2, T26.3, T26.4

Notes

Author

Brenton Ross

2.6 Runtime Use Cases

2.6.1 UC-145: Non-interactive Script

Version

Application design. Increment #1.

Goal

Run a script which does not require the input from the user once started.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user navigates to the scripts desktop menu.
2. The user runs the script.
3. The system executes the vici program which runs the script.
4. The script completes and the vici program exits.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T23.2, T25.1

Notes

Author

Brenton Ross

2.6.2 UC-146: Run Script in Directory

Version

Application design. Increment #1.

Goal

Run the script in a directory specified by the user.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user runs the script.
2. The system executes the vici program.
3. The user selects the “Directory” option.
4. The system displays a directory selection dialog.
5. The user navigates to the required directory.
6. The user selects “OK” and the “Run”.
7. The system confirms that the script has been signed.
8. The system executes the script.

Alternative Sequence of Events

Postconditions

T23.2, T25.2.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.6.3 UC-147: Script Selection

Version

Application design. Increment #1.

Goal

The script to run is selected after the vici program is running.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user runs the vici program.
2. The system launches the vici program.
3. The user selects “Open”.
4. The system displays a file selection dialog.
5. The user selects the script to run.
6. The system confirms that the script has been signed.
7. The user presses “Run”.
8. The system executes the script.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T23.2, T25.3.

Notes

Author

Brenton Ross

2.6.4 UC-148: Run a Script Function

Version

Application design. Increment #1.

Goal

Run a function defined within the script that has been attached to a menu item.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user opens a script in the vici program.
2. The system creates menu entries for the functions that are attached to menus.
3. The user selects the menu item.
4. The script runs the function attached to the menu.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T25.4.

Notes

Author

Brenton Ross

2.6.5 UC-149: Script with Drag-n-Drop

Version

Application design. Increment TBA.

Goal

Allow the user to start a script, or function within the script, by dropping an item onto a display field.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user opens a script in the vici program.
2. The system creates text entry fields for the drag-n-drop variables.
3. The user drags a file name (for example) from the file browser to the text field.
4. The system starts the function attached to the field.
5. The system completes execution of the script.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T25.5, T25.8.

Notes

Author

Brenton Ross

2.6.6 UC-150: Control Script

Version

Application design. Increment #2.

Goal

The user can pause, restart and terminate a running script.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user opens a script with the vici program.

2. The system confirms that it has been signed.
3. The user presses the “Run” button.
4. The system begins execution of the script.
5. The user presses the “Pause” button.
6. The system suspends execution.
7. The user presses the “Continue” button.
8. The system resumes execution.
9. The user presses the “Terminate” button.
10. The system terminates execution of the script.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T25.6, T25.9.

Notes

Author

Brenton Ross

2.6.7 UC-151: Text Interfaces

Version

Application design. Increment #1.

Goal

Accept user input, and display output and error messages.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

1. The user opens a script in the vici program.
2. The system creates multi-line text fields for user input, standard output and error output.
3. The user selects the “Run” option.
4. The system waits for user input.
5. The user enters the required input.
6. The system displays the results in the output window.
7. The system displays error messages in the error window.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

T25.7

Notes

Author

Brenton Ross

3 Use Cases by Responsibility

This table lists all the responsibilities and for each one lists the use cases that demonstrate it. This can be used to verify that our testing covers the entire set of responsibilities and that nothing has been overlooked.

T1.1	UC-101 UC-104
T1.2	UC-101
T1.3	UC-101
T1.4	UC101
T1.5	UC-102
T1.6	UC-102
T1.7	UC-103
T1.8	UC-103
T1.9	UC-102
T1.10	UC-102
T1.11	UC-105
T1.12	Construction
T1.13	
T1.14	
T2.1	Construction
T2.108	Construction
T3.1	UC-106
T3.2	UC-106 UC-107
T3.3	UC-107
T3.4	UC-106
T3.5	UC-107 UC-110 UC-111 UC-112
T3.6	UC-110 UC-111
T3.7	UC-110 UC-111
T3.8	UC-112
T3.9	UC-108
T3.10	UC-113
T3.11	Construction
T3.12	UC-108
T3.13	UC-113
T4.1	UC-107

T4.2	UC-107
T4.3	UC-107
T4.5	UC-107
T5.1	UC-116
T5.2	UC-116
T5.3	UC-116
T5.4	UC-116
T5.5	UC-116
T6.1	UC-107 UC-117
T6.2	UC-107 UC-117
T6.3	UC-117
T6.4	UC-107 UC-117
T6.5	UC-107 UC-117
T7.1	UC-118
T7.2	UC-118
T7.3	UC-119
T7.4	UC-119
T8.1	UC-114
T8.2	UC-114
T8.3	UC-114
T8.4	UC-114
T9.1	UC-106 UC-115
T9.2	UC-115
T9.3	UC-115
T9.4	UC-115
T9.5	UC-115
T9.6	UC-115
T10.1	UC-142
T10.2	UC-142
T10.3	UC-142
T10.4	UC-143
T11.1	UC-120
T11.2	UC-121
T12.1	UC-136

T12.2	UC-136
T12.3	UC-136
T12.4	UC-135 UC-136
T12.5	Construction
T12.6	UC-135
T12.7	UC-135
T13.1	UC-137
T13.2	UC-137
T13.3	UC-139
T13.4	UC-137
T13.5	UC-141
T13.6	UC-138
T13.7	UC-127 UC-137
T13.8	UC-137
T13.9	UC-137
T13.10	UC-137
T14.1	UC-137
T14.2	UC-137
T14.3	UC-137
T14.4	UC-137
T14.5	UC-137
T14.6	UC-141
T15.1	UC-141
T15.2	Construction
T15.3	UC-141
T15.4	UC-141
T15.5	UC-141
T16.1	UC-140 UC-141
T16.2	UC-140 UC-141
T17.1	UC-122
T17.2	UC-122
T17.3	UC-122
T17.4	UC-122
T18.1	UC-123

T18.2	UC-123
T18.3	UC-123
T18.4	UC-123
T19.1	UC-124
T19.2	UC-126
T19.3	UC-125
T19.4	UC-128
T19.5	UC-127
T20.1	UC-128
T20.2	UC-129
T21.1	UC-129
T22.1	UC-129
T23.1	UC-108
T23.2	UC-113 UC-145 UC-146 UC-147
T24.1	UC-117
T24.2	UC-117
T24.3	UC-130
T24.4	UC-127 UC-131
T24.5	UC-131
T24.6	UC-132
T24.7	UC-117
T24.8	UC-133
T24.9	UC-134
T25.1	UC-145
T25.2	UC-146
T25.3	UC-147
T25.4	UC-148
T25.5	UC-149
T25.6	UC-150
T25.7	UC-151
T25.8	UC-149
T25.9	UC-150
T26.1	UC-144
T26.2	UC-144

T26.3	UC-144
T26.4	UC-144
T27.1	Construction
T28.1	Construction
T29.1	Construction
T30.1	Construction
T31.1	UC-135
T32.1	Construction
T33.1	Construction
T33.2	Construction
T34.1	Construction
T35.1	Construction
T35.2	Construction
T36.1	Construction
T37.1	Construction
T38.1	Construction
T38.2	Construction
T39.1	
T40.1	Construction
T41.1	Construction
T42.1	Construction
T43.1	Construction
T43.2	Construction
T43.3	Construction
T44.1	Construction
T44.2	Construction
T45.1	Construction
T45.2	Construction
T45.3	Construction
T46.1	Construction
T47.1	Construction
T47.2	Construction
T48.1	Construction
T48.2	Construction

Appendix A – Use Case Template

UC-000: Copy Me

Version

Application design.

Goal

.

Summary

<diagram goes here>

Actors

Stakeholders

Preconditions

Triggers

Normal Sequence of Events

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

Notes

Author

Brenton Ross

Appendix A