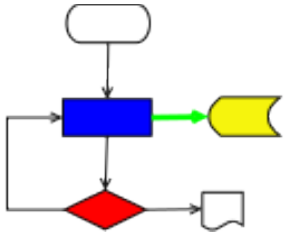
VICI



VISUAL CHART INTERPRETER

Requirements Analysis

Publication History

Date	Who	What Changes
3 November 2012	Brenton Ross	Initial version.

Table of Contents

1 Introduction....4
1.1 Scope....4
1.2 Audience....4
2 Use Case Descriptions....5
2.1 Administration Use Case....5
2.1.1 UC-001: Prepare a Command....5
2.1.2 UC-002: Edit a Prepared Command....6
2.1.3 UC-018: Import and Export Commands....7
2.1.4 UC-003: Create an Alias Command....9
2.2 Editing Uses Cases....10
2.2.1 UC-004: Create Simple Script....10
2.2.2 UC-007: Modify a Script....11
2.2.3 UC-008: Layout of a Flowchart....13
2.2.4 UC-009: Create a Script Function....14
2.2.5 UC-014: Create a Script....15
2.2.6 UC-015: Create a Complex Script....16
2.2.7 UC-010: Search with Tags....17
2.2.8 UC-011: Classify Commands....18
2.2.9 UC-012: Observe Script Operation....20
2.2.10 UC-016: Test a Script....21
2.3 Installation Use Cases....23
2.3.1 UC-005: Install a Script....23
2.3.2 UC-006: Uninstall a Script....25
2.3.3 UC-013: Schedule a Script....26
2.4 Runtime Use Cases....27
2.4.1 UC-017: Run a Script....27
Appendix A – Use Case Template....29

1 Introduction

This is the requirements analysis document for the VICI project.

The usual purpose is to the review the requirements and to highlight any discrepancies, inconsistencies and omissions. This is followed by the generation of any implied requirements and the creation of a set of use case diagrams that provide some evidence that the analysts have understood the requirements.

This information is normally passed back to the authority responsible for creating the requirements so that the problems might be resolved before development gets under way.

1.1 Scope

For the VICI project we are initially limiting this to the use case descriptions since it is a one person project for now. At some later point, if the team ever gets user representatives and analysts the full scope may be added.

1.2 Audience

This document is designed to be used by the designers of the VICI project or anyone else wanting an idea of how the system was intended to be used.

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

2.1.1 UC-001: Prepare a Command

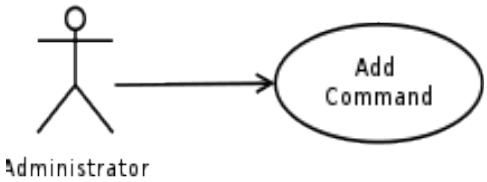
Version

Application design.

Goal

A UNIX/Linux command line program is made available for the users of VICI to easily incorporate into a script.

Summary



Actors

An administrator or someone with good UNIX skills.

Stakeholders

Users of VICI.

Preconditions

1. VICI has been installed.

Triggers

A user wishes to include a new command in one of their scripts.

Normal Sequence of Events

1. Open the VICI-admin program.
2. Enter the command.
3. Enter a short description of the command.
4. Enter the command to use for help on the command.
5. Enter the EBNF of the command options.
6. Press the 'Add' button.
7. The system displays a syntax chart of the command options.
8. The user confirms the operation.

9. The system adds the command to its database.
10. The user closes the program.

Alternative Sequence of Events

1. The system reports that the command is already in the database.
2. The system reports that the EBNF is not valid.

Postconditions

1. The command is installed in the database.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.1.2 UC-002: Edit a Prepared Command

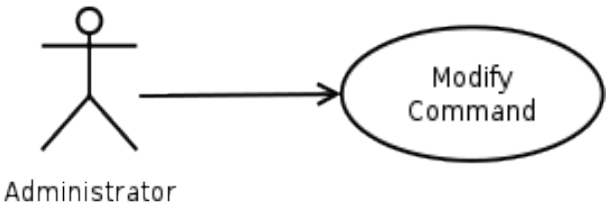
Version

Application design.

Goal

Some aspect of a prepared command needs to be changed, usually to make it easier to use.

Summary



Actors

An administrator, or other experienced Linux user.

Stakeholders

Preconditions

1. The command has already been prepared.

Triggers

A user expresses a desire to have the command improved.

Normal Sequence of Events

1. Open the VICI-admin program
2. Enter the command
3. The system loads and displays the command options, the EBNF, and the syntax chart, along with the command to get help.
4. The user modifies the vales as required.
5. The user presses the 'Update' button.
6. The system updates the display of the syntax chart.
7. The user confirms the update.
8. The system saves the file.
9. The user closes the program.

Alternative Sequence of Events

Postconditions

An updated version of the command is available for users to use.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.1.3 UC-018: Import and Export Commands

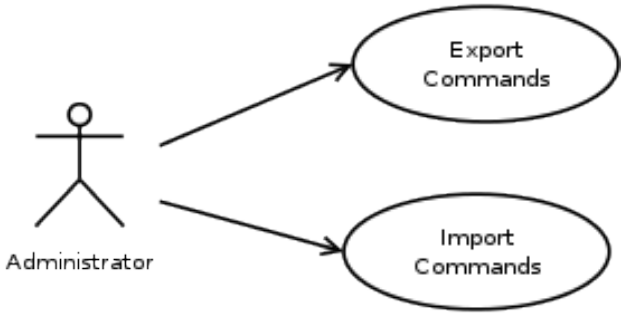
Version

Application design.

Goal

Demonstrate that prepared commands can be imported into and exported from the database of prepared commands. This will allow prepared commands to be shared.

Summary



Actors

An administrator or experienced Linux user.

Stakeholders

Preconditions

1. Some commands have already been created.

Triggers

A need to share prepared commands.

Normal Sequence of Events

1. Select one or more commands.
2. The system highlights the selected commands.
3. Select the “Export” option.
4. The system displays a file dialog.
5. The user enters the name and location of a file into which the commands will be written.
6. The user closes this instance and opens a new instance that uses a different database.
7. The user selects the “Import” option.
8. The system displays a file dialog.
9. The user selects the file to import, and presses OK.
10. The system merges the selected commands into its database.

Alternative Sequence of Events

1. The system detects that imported commands are already present in the database.
2. The system notifies the user that there are duplicate commands and assigns unique new names to the commands.

Postconditions

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.1.4 UC-003: Create an Alias Command

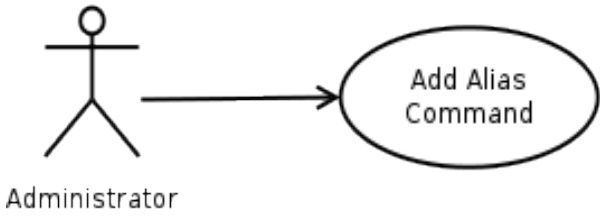
Version

Application design.

Goal

A command is created that does not correspond to a system program, but rather to a system program and some set of parameters and options.

Summary



Actors

An administrator or an experienced Linux user.

Stakeholders

Preconditions

1. VICI has been installed.

Triggers

A command is frequently used with a particular set of parameters and options.

Normal Sequence of Events

1. Open the VICI-admin program.
2. Enter the command.
3. Enter a short description of the command.
4. Enter the command to use for help on the command.
5. Enter an alias name for the command.
6. Enter the options and parameters that are constant for the alias command.
7. Enter the EBNF of the command options.
8. Press the 'Add' button.
9. The system displays a syntax chart of the command options.
10. The user confirms the operation.
11. The system adds the command to its database.
12. The user closes the program.

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

Notes

Author
Brenton Ross

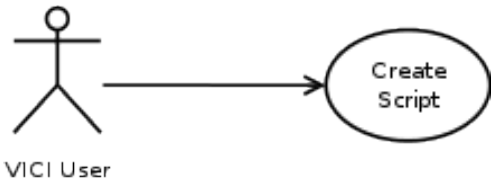
2.2 Editing Uses Cases

2.2.1 UC-004: Create Simple Script

Version
Application design.

Goal
Create a simple script that runs a few commands without user interaction when run.

Summary



Actors
Linux desktop user.

Stakeholders

Preconditions
1. VICI has been installed.
2. All necessary commands have been prepared.

Triggers
The user needs some action performed which can be achieved by running several command line programs.

Normal Sequence of Events

1. Open VICI-editor
2. Locate command in search panel
3. Select a symbol
4. Place symbol on canvas
5. Assign command to symbol
6. System displays syntax chart of command
7. User enters options and parameters
8. <Repeat from 2>
9. User adds links between commands.
10. User saves script

Alternative Sequence of Events

Postconditions

A script h/as been saved in the user's script collection.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.2 UC-007: Modify a Script

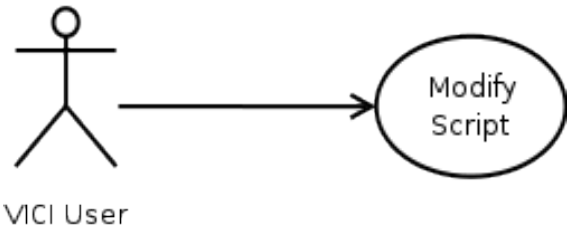
Version

Application design.

Goal

The user wants to make some changes to an existing script.

Summary



Actors

Linux desktop user

Stakeholders

Preconditions

1. VICI is installed.
2. All necessary commands have been prepared.
3. An existing script exists.

Triggers

The user notices some alteration is necessary in the script.

Normal Sequence of Events

1. The user opens VICI-editor
2. The user selects an existing script in the file open dialog.
3. The system displays the script as a flowchart.
4. The user selects a command box.
5. The system displays the command and its options.
6. The user modifies the options and parameters.
7. The user selects another command box and presses 'Delete'.
8. The system removes the command box and associated links from the diagram.
9. The user saves the script.

Alternative Sequence of Events

Postconditions

The modified script replaces the original version in the user's script collection.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.3 UC-008: Layout of a Flowchart

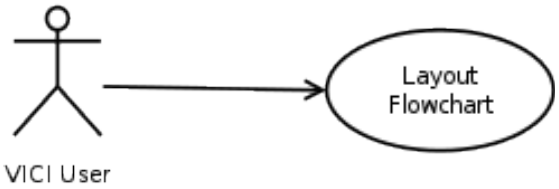
Version

Application Design.

Goal

A flowchart will be laid out using the automated capability of VICI-ed.

Summary



Actors

A Linux desktop user.

Stakeholders

Preconditions

1. A script has already been created and loaded.

Triggers

The user considers that their manual layout is a mess and needs to be fixed.

Normal Sequence of Events

1. The user presses the 'Auto Layout' button.
2. The system calculates a new layout and presents it to the user.
3. The user saves the script.

Alternative Sequence of Events

1. The user rejects the new layout and presses the 'Revert' button.
2. The system restores the layout to its previous configuration.

Postconditions

The layout of the script is re-organised.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.4 UC-009: Create a Script Function

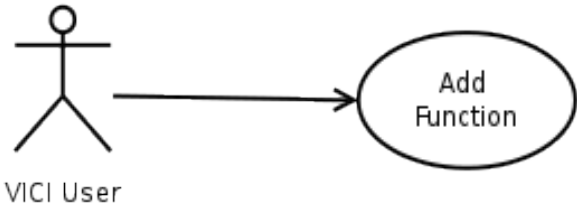
Version

Application design.

Goal

Create a script function that can be accessed from a menu option.

Summary



Actors

Linux desktop user.

Stakeholders

Preconditions

A script already exists and is open for editing.

Triggers

The user wishes to add a secondary function to the script.

Normal Sequence of Events

1. The user selects a set of command boxes.
2. The system displays a text field for the function name.
3. The user enters a function name for the selection.
4. The user enters a name for the menu entry.
5. The user positions the menu entry in the menu hierarchy for the script.
6. The user presses the 'Add Function' button.
7. The system creates a new function containing the selected command boxes, and replaces the command boxes in the original function with a function command box.
8. The user saves the script.

Alternative Sequence of Events

1. The user does not enter a menu name.
2. The system does not create a separate menu, but still creates the function and replaces the boxes.

Postconditions

A menu entry is presented when the script is run.

Business Rules

Responsibilities

Notes

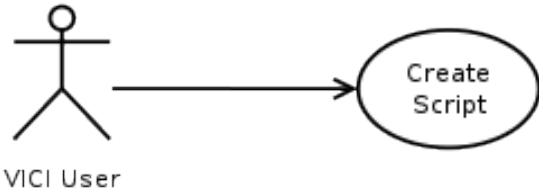
Author
Brenton Ross

2.2.5 UC-014: Create a Script

Version
Application design.

Goal
Create a VICI script that demonstrates built in commands.

Summary



Actors
Linux desktop user.

Stakeholders

Preconditions
1. VICI has been installed.
2. All necessary commands have been prepared.

Triggers
User desires to create a script to automate some process.

Normal Sequence of Events
1. The user opens the VICI-editor program.
2. The user opens a new script.
3. The user places a “cd” command.
4. The user places a “for each line” command, fed from a file.
5. The user places a “for each” command, from the input lines.
6. The user places other commands as necessary.
7. The user connects the output to a file.
8. The user saves the script.

Alternative Sequence of Events

Postconditions
A new script is saved in the user's script collection.

Business Rules

Responsibilities

Notes

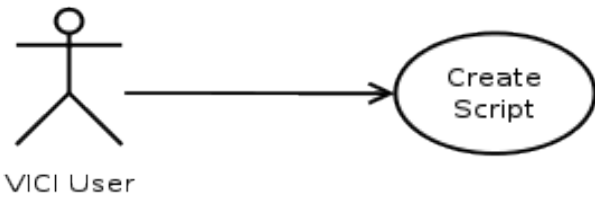
Author
Brenton Ross

2.2.6 UC-015: Create a Complex Script

Version
Application design.

Goal
Create a script which demonstrates most aspects of VICI programming.

Summary



Actors
Linux desktop user.

Stakeholders

Preconditions
1. VICI has been installed.
2. All necessary commands have been prepared.

Triggers
The user desires to create a new script.

Normal Sequence of Events
1. The user opens the VICI-editor.
2. The user opens a new script.
3. The user places command boxes:

1. Background functions, with parameters
2. Global variables
3. Thread specific variables
4. Global and temporary files.
5. Named pipes
6. In-line files with embedded variables
7. Signals, mutexes and semaphores
8. Arithmetic operations
9. Use of exit status, and process ids

Alternative Sequence of Events

Postconditions

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.7 UC-010: Search with Tags

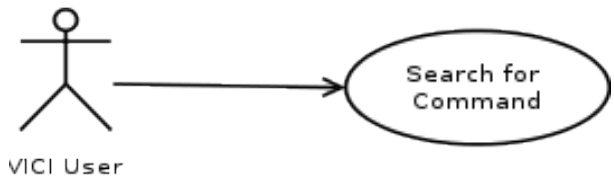
Version

Application design.

Goal

Use tags to find commands that perform a desired function.

Summary



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 starts the VICI-editor program.
2. The user open the search window.
3. The user selects some tags and specifies unions and intersections.
4. The system displays the list of matching commands.
5. The user selects a command.

Alternative Sequence of Events

Postconditions

The user has selected a command of interest.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.8 UC-011: Classify Commands

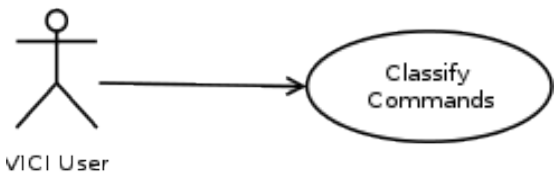
Version

Application design.

Goal

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

Summary



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

Alternative Sequence of Events

Postconditions

The tag database is updated to include the new relationships.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

2.2.9 UC-012: Observe Script Operation

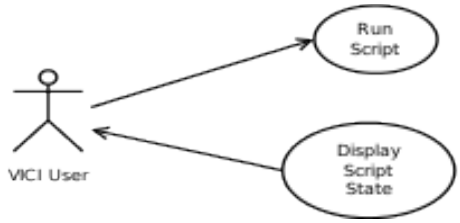
Version

Application design.

Goal

Verify that a script behaves as required.

Summary



Actors

Linux desktop user.

Stakeholders

Preconditions

1. A script has been written.
2. The VICI-editor program has the script loaded.

Triggers

A script has been written.

Normal Sequence of Events

1. The user opens the testing window.
2. The system displays the flowchart with the execution cursor at the start of the script.
3. The user presses 'Step'
4. The system executes the first command, and updates the variables display, and moves the cursor to the next command.
5. The user sets the pause interval at 2 seconds.
6. The user sets a break point in the script.
7. The user presses the 'Continue' button.
8. The system executes the script, waiting 2 seconds after each command, and stops when the breakpoint is reached. Variable values are displayed after each command is executed.

Alternative Sequence of Events

Postconditions

The script is executed and the user is confident of correct behaviour.

Business Rules

Responsibilities

Notes

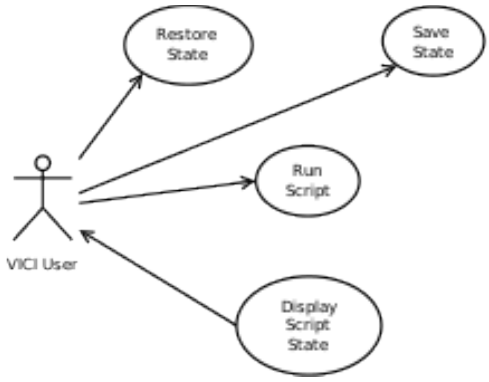
Author
Brenton Ross

2.2.10 UC-016: Test a Script

Version
Application design.

Goal
To thoroughly test a script.

Summary



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 system saves a snapshot of the variables.
10. The user performs further tests.
11. The user reloads the snapshot file
12. 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

Notes

Author
Brenton Ross

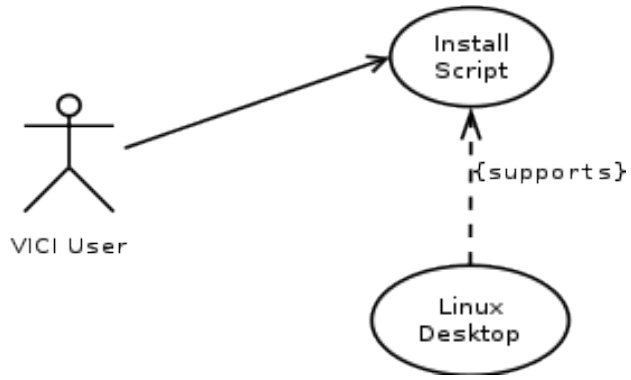
2.3 Installation Use Cases

2.3.1 UC-005: Install a Script

Version
Application design.

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

Summary



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

Notes

Author

Brenton Ross

2.3.2 UC-006: Uninstall a Script

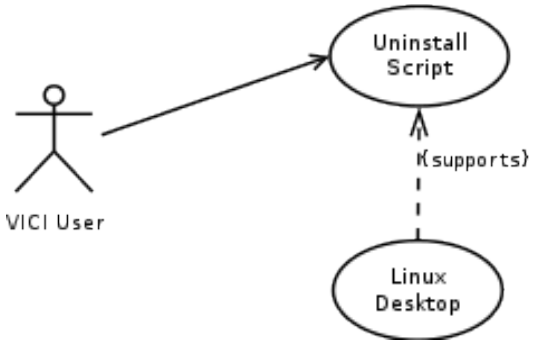
Version

Application design.

Goal

Remove a script from the desktop menu system.

Summary



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

Notes

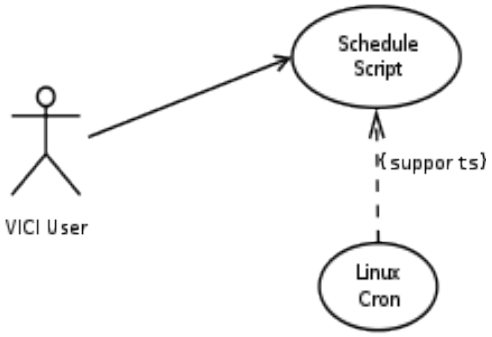
Author
Brenton Ross

2.3.3 UC-013: Schedule a Script

Version
Application design.

Goal
Schedule a script to be run at specific times.

Summary



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

Notes

Author
Brenton Ross

2.4 Runtime Use Cases

2.4.1 UC-017: Run a Script

Version
Application design.

Goal
Run a script in a particular directory.

Summary

Actors
Linux desktop user.

Stakeholders

Preconditions
1. VICI has been installed.
2. A VICI script has been created.
3. The VICI script has been installed.

Triggers
The user desires to run the script.

Normal Sequence of Events

1. The user runs the script from the desktop.
2. The system starts the VICI program and loads the script.
3. The user selects the directory menu
4. The system opens a directory selection dialog.
5. The user enters a directory.
6. The system changes to that directory.
7. The user selects the file menu.
8. The system starts the file manager program.
9. The user selects a file and drops it onto the script.
10. The system executes the script.

Alternative Sequence of Events

Postconditions

The script's actions have been implemented.

Business Rules

Responsibilities

Notes

Author

Brenton Ross

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
