CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

Members: Christian Boni, Tim Glorioso

Project: Coffee Maker Quest

Introduction

It is safe to say that a majority of the concerns and difficulties of this project stemmed from the requirements. All of the requirements seemed very straightforward and easily understable after the first time reading them. However, when it came down to analyzing each requirement individually a lot ambiguity began to arise. Using the FUN-MOVE requirement as an example. This requirement states a player can only move north if and only if a door leading north exists and a player can only move south if and only if a door leading south exists. However, when testing this behavior in the program if the player attempts to move in the direction of a nonexistent door the player enters “magical land” and is teleported back to the starting room. This behavior was obviously wrong as the requirements stated that no movement should occur in the direction with a nonexistent door. Since this was not mentioned anywhere in the requirements we viewed this as unexpected behavior and classified it as a default. Another example would be the both FUN-LOOK and FUN-INVENTORY requirements. These requirements clearly define the behavior on how to collect and display an item when it has not previously been collected. However, there comes some ambiguity when attempting to re-collect an item. Some of the questions that came up: Should the player be able to collect multiple items of the same type? Should the player only be permitted to collect one of each item? Should the inventory display quantities for each item? Should the inventory only show if an item was collected or not? ect. After performing some exploratory testing we came to a conclusion on the issue. We found that while items can be re-collected by a player the item will only be shown as being collected or not in the inventory. We then used this expected behavior as a basis for our test cases.

As we were creating our test plan, we found at times it was hard to distinguish which requirement each test case belonged to. For example, when coming up with test cases for the FUN-WIN requirement we wondered if some of them should actually belong to the FUN-LOSE requirement instead. This is because these two requirements are polar opposites of each other. A failure case for the FUN-WIN requirement would be a passing case for the FUN-LOSE requirement and vice-versa. For simplicity we chose to separate the test cases so that each requirement would only have the passing cases. As a result, we created test cases that would cause a player to win for the FUN-WIN requirement and test cases that would cause a player to lose for the FUN-LOSE requirement. Another requirement that had a lot of overlapping test cases was the FUN-ITERATION requirement. This requirement essentially needed a test case for each and every command. However, the requirements FUN-MOVE, FUN-INVENTORY, FUN-LOOK, and FUN-HELP already required test cases for 5 of the 6 commands. It became obvious that it would be redundant to include all of these test cases again for the FUN-ITERATION requirement. As a result, the only test case that we found was necessary for the TEST-ITERATION requirement was for the drink command. Lastly, we did find it rather important to include edge case testing in our test plan. An example of this would be with the FUN-UNKNOWN-COMMAND requirement. We originally had two test cases, one for a single character specified command and another for a single character unspecified command. We then chose to add additional test cases that would test for a no input command, a multiple character command, and a numerical command.

Test Plan

IDENTIFIER: DRINK-COMMAND-TEST

TEST CASE: Ensure that the command ‘D’ causes the player to drink.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: D

EXECUTION STEPS: Type ‘D’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has drank.

IDENTIFIER: VALID-INPUT-TEST

TEST CASE: Enter a command specified in the FUN-ITERATION requirement.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: N,S,L,I,H,D

EXECUTION STEPS: Type ‘c’ on the command line and press enter, where c is equal to one of the input values.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console displays the normal output for the given command. The console shall not display “What?”.

IDENTIFIER: INVALID-INPUT-TEST

TEST CASE: Enter a command not specified in the FUN-ITERATION requirement.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: Z

EXECUTION STEPS: Type ‘Z’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display “What?”.

IDENTIFIER: NO-INPUT-TEST

TEST CASE: Enter a blank command.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: N/A

EXECUTION STEPS: Press enter without any characters on the command line.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display “What?”.

IDENTIFIER: NUM-INPUT-TEST

TEST CASE: Enter a numerical command.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: 0,1,2,3,4,5,6,7,8,9

EXECUTION STEPS: Type n on the command line and press enter, where n is equal to one of the input values.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display “What?”.

IDENTIFIER: MULTIPLE-INPUT-TEST

TEST CASE: Combine multiple single letter commands specified in the FUN-ITERATION requirement into one multi letter command. Ensure the multi letter command does not attempt to execute single letter commands in succession of one another.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: NSLIHD

EXECUTION STEPS: Type ‘NSLIHD’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display “What?”.

IDENTIFIER: UPPER-INPUT-TEST

TEST CASE: Enter a command specified in the FUN-ITERATION requirement as uppercase.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: N,S,L,I,H,D

EXECUTION STEPS: Type ‘c’ on the command line and press enter, where c is equal to one of the input values.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console displays the normal output for the given command. The console shall not display “What?”.

IDENTIFIER: LOWER-INPUT-TEST

TEST CASE: Enter a command specified in the FUN-ITERATION requirement as lowercase.

PRECONDITIONS: The player must currently be in a room.

INPUT VALUES: n,s,l,i,h,d

EXECUTION STEPS: Type ‘c’ on the command line and press enter, where c is equal to one of the input values.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console displays the normal output for the given command. The console shall not display “What?”.

IDENTIFIER: VALID-MOVE-NORTH-TEST

TEST CASE: Ensure that the command ‘N’ causes the player to move north when in a room with a door leading north..

PRECONDITIONS: The player must currently be in a room with a door leading north.

INPUT VALUES: N

EXECUTION STEPS: Type ‘N’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has entered the northern room.

IDENTIFIER: INVALID-MOVE-NORTH-TEST

TEST CASE: Ensure that the command ‘N’ does not cause the player to move when in a room without a door leading north.

PRECONDITIONS: The player must currently be in a room with no door leading north.

INPUT VALUES: N

EXECUTION STEPS: Type ‘N’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The player shall not move and remain in the same room.

IDENTIFIER: VALID-MOVE-SOUTH-TEST

TEST CASE: Ensure that the command ‘S’ causes the player to move south when in a room with a door leading south..

PRECONDITIONS: The player must currently be in a room with a door leading south.

INPUT VALUES: S

EXECUTION STEPS: Type ‘S’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has entered the southern room.

IDENTIFIER: INVALID-MOVE-SOUTH-TEST

TEST CASE: Ensure that the command ‘N’ does not cause the player to move when in a room without a door leading south.

PRECONDITIONS: The player must currently be in a room with no door leading south.

INPUT VALUES: S

EXECUTION STEPS: Type ‘S’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The player shall not move and remain in the same room.

IDENTIFIER: ALL-ITEMS-TEST

TEST CASE: Ensure that when the player has collected all three items (Coffee, Sugar, Cream) the command ‘D’ causes the player to win the game.

PRECONDITIONS: The player must have already collected Coffee, Sugar, and Cream.

INPUT VALUES: D

EXECUTION STEPS: Type ‘D’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has won the game.

IDENTIFIER: TWO-ITEMS-TEST

TEST CASE: Ensure that when the player has collected any two of the three items (Coffee, Sugar, Cream) the command ‘D’ causes the player to lose the game.

PRECONDITIONS: The player must have already collected only two of following three items: Coffee, Sugar, or Cream.

INPUT VALUES: D

EXECUTION STEPS: Type ‘D’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has lost the game.

IDENTIFIER: ONE-ITEM-TEST

TEST CASE: Ensure that when the player has collected any one of the three items (Coffee, Sugar, Cream) the command ‘D’ causes the player to lose the game.

PRECONDITIONS: The player must have already collected only one of following three items: Coffee, Sugar, or Cream.

INPUT VALUES: D

EXECUTION STEPS: Type ‘D’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has lost the game.

IDENTIFIER: NO-ITEM-TEST

TEST CASE: Ensure that when the player has not collected any of the three items (Coffee, Sugar, Cream) the command ‘D’ causes the player to lose the game.

PRECONDITIONS: The player must have not collected any of following three items: Coffee, Sugar, or Cream.

INPUT VALUES: D

EXECUTION STEPS: Type ‘D’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has lost the game.

IDENTIFIER: EMPTY-INVENTORY-TEST

TEST CASE: Ensure that when the player has not collected any of the three items (Coffee, Sugar, Cream) the command ‘I’ shows an empty inventory with no items collected.

PRECONDITIONS: The player must have not collected any of following three items: Coffee, Sugar, or Cream.

INPUT VALUES: I

EXECUTION STEPS: Type ‘I’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate to the player that he/she has not collected any of the items yet.

IDENTIFIER: FILLED-INVENTORY-TEST

TEST CASE: Ensure that when the player has collected any one of the three items (Coffee, Sugar, Cream) the command ‘I’ shows an inventory with the item marked as collected.

PRECONDITIONS: The player must have already collected any one of following three items: Coffee, Sugar, or Cream.

INPUT VALUES: I

EXECUTION STEPS: Type ‘I’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate to the player that he/she has collected the item mentioned above.

IDENTIFIER: DUPLICATE-INVENTORY-TEST

TEST CASE: When in a room containing a previously collected item ensure that the command ‘L’ followed by the command ‘I’ shows an inventory with the item marked as being collected only once

PRECONDITIONS: The player must be in a room containing an item. The item in the room must have already been collected by the player.

INPUT VALUES: L,I

EXECUTION STEPS: Type ‘L’ on the command line and press enter. Type ‘I’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate to the player that he/she has collected the item mentioned above. The console shall not indicate to the player that he/she has collected the item multiple times.

IDENTIFIER: VALID-LOOK-TEST

TEST CASE: Ensure that the command ‘L’ causes the player to collect an item when in a room containing an item.

PRECONDITIONS: The player must be in a room containing an item.

INPUT VALUES: L

EXECUTION STEPS: Type ‘L’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate to the player that an item has been found.

IDENTIFIER: INVALID-LOOK-TEST

TEST CASE: Ensure that the command ‘L’ does not cause the player to collect an item when in a room not containing an item.

PRECONDITIONS: The player must be in a room not containing an item.

INPUT VALUES: L

EXECUTION STEPS: Type ‘L’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display “You don't see anything out of the ordinary.”

IDENTIFIER: DUPLICATE-LOOK-TEST

TEST CASE: When in a room containing a previously collected item ensure that the command ‘L’ allows the player to re-collect the item.

PRECONDITIONS: The player must be in a room containing an item. The item in the room must have already been collected by the player.

INPUT VALUES: L

EXECUTION STEPS: Type ‘L’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate to the player that an item has been found. The console shall not indicate the player has already found the item.

IDENTIFIER: HELP-TEST

TEST CASE: Ensure that the command ‘H’ shows the player a list of possible commands and what their effects are.

PRECONDITIONS: The player must be in a room.

INPUT VALUES: H

EXECUTION STEPS: Type ‘H’ on the command line and press enter.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall display a list of possible commands and what their effects are.

IDENTIFIER: UNIQ-ADJ-TEST

TEST CASE: Ensure that each room in the house has a unique adjective describing it.

PRECONDITIONS: The player must be in the first room (Small Room).

INPUT VALUES: N

EXECUTION STEPS: Type ‘N’ on the command line and press enter. Repeat this process until there are no doors left leading north.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has reached the last room in the house (Rough Room). Over the course of the execution steps the user shall notice that no two rooms had the same adjective describing them.

IDENTIFIER: UNIQ-FURN-TEST

TEST CASE: Ensure that each room in the house has one and only one unique furnishing visible to the user.

PRECONDITIONS: The player must be in the first room (Small Room).

INPUT VALUES: N

EXECUTION STEPS: Type ‘N’ on the command line and press enter. Repeat this process until there are no doors left leading north.

OUTPUT VALUES: N/A

POSTCONDITIONS: The console shall indicate the player has reached the last room in the house (Rough Room). Over the course of the execution steps the user shall notice that each room had one and only one unique furnishing visible.

Traceability Matrix

Requirements	Test Cases
FUN-ITERATION	DRINK-COMMAND-TEST
FUN-UNKNOWN-COMMAND	VALID-INPUT-TEST
	INVALID-INPUT-TEST
	NO-INPUT-TEST
	NUM-INPUT-TEST
	MULTIPLE-INPUT-TEST
FUN-INPUT-CAPS	UPPER-INPUT-TEST
	LOWER-INPUT-TEST
FUN-MOVE	VALID-MOVE-NORTH-TEST
	INVALID-MOVE-NORTH-TEST
	VALID-MOVE-SOUTH-TEST
	INVALID-MOVE-SOUTH-TEST
FUN-WIN	ALL-ITEMS-TEST
FUN-LOSE	TWO-ITEM-TEST
	ONE-ITEM-TEST
	NO-ITEM-TEST
FUN-INVENTORY	EMPTY-INVENTORY-TEST
	FILLED-INVENTORY-TEST
	DUPLICATE-INVENTORY-TEST
FUN-LOOK	VAILD-LOOK-TEST
	INVAILD-LOOK-TEST
	DUPLICATE-LOOK-TEST
FUN-HELP	HELP-TEST
FUN-UNIQ-ROOM	UNIQ-ADJ-TEST
FUN-UNIQ-ROOM-FURNISHING	UNIQ-FURN-TEST

Defect Tracking

DESCRIPTION: The character ‘H’ is not a recognized command.
SUMMARY: When entering an uppercase ‘H’ command the console responds with “What?” instead of showing a list of possible commands and their effects to the user. This behavior conflicts with the requirements FUN-ITERATION and FUN-HELP which both indicate that a help command exists. This defect was found while performing the test case HELP-TEST
REPRODUCTION STEPS:

1. Run Coffee Maker Quest.
2. Type ‘H’ on the command line.
3. Press enter.

EXPECTED BEHAVIOR: The console should show a list of possible commands and their effects.
OBSERVED BEHAVIOR: Console displays “What?”.

DESCRIPTION: The character ‘h’ is not a recognized command.
SUMMARY: When entering an lowercase ‘h’ command the console responds with “What?” instead of showing a list of possible commands and their effects to the user. This behavior conflicts with the requirements FUN-ITERATION and FUN-HELP which both indicate that a help command exists. This defect was found while performing the test case HELP-TEST
REPRODUCTION STEPS:

1. Run Coffee Maker Quest.
2. Type ‘h’ on the command line.
3. Press enter.

EXPECTED BEHAVIOR: The console should show a list of possible commands and their effects.
OBSERVED BEHAVIOR: Console displays “What?”.

DESCRIPTION: Player is able to move north when no door leading north exists.

SUMMARY: When in the “Rough room” with no door leading north, if the player enters the “N” command to move north he/she is taken to “magical land” and teleported back into the “Small room”. This behavior conflicts with the requirement FUN-MOVE which states that a player shall only be able to move north if and only if a door exists going north. This defect was found while performing the test case INVALID-MOVE-NORTH-TEST.

REPRODUCTION STEPS:

1. Run Coffee Maker Quest.
2. Type ‘N’ on the command line.
3. Press enter.
4. Repeat Steps 2 & 3 until player enters the “Rough room”.
5. Type ‘N’ on the command line.
6. Press enter.

EXPECTED BEHAVIOR: The player shall remain in the “Rough room”.

OBSERVED BEHAVIOR: The console displays “You are in a magical land! But you are returned to the beginning!” and the player is teleported back into the “Small room”.

DESCRIPTION: Player is able to move south when no door leading south exists.

SUMMARY: When in the “Small room” with no door leading south, if the player enters the “S” command to move south he/she is taken to “magical land” and teleported back into the “Small room”. This behavior conflicts with the requirement FUN-MOVE which states that a player shall only be able to move south if and only if a door exists going south. This defect was found while performing the test case INVALID-MOVE-SOUTH-TEST.

REPRODUCTION STEPS:

1. Run Coffee Maker Quest.
2. Type ‘S’ on the command line.
3. Press enter.

EXPECTED BEHAVIOR: The player shall remain in the “Small room”.

OBSERVED BEHAVIOR: The console displays “You are in a magical land! But you are returned to the beginning!” and the player is teleported back into the “Small room”.

DESCRIPTION: The character ’n’ is not a recognized command.

SUMMARY: When entering a lowercase ‘n’ command the console responds with “What?” instead of moving the player to the northern room. This behavior conflicts with the requirement FUN-INPUT-CAPS which states that commands shall all be case insensitive. This defect was found while performing the test case LOWER-INPUT-TEST.

REPRODUCTION STEPS:

1. Run Coffee Maker Quest.
2. Type ‘N’ on the command line.
3. Press enter.

EXPECTED BEHAVIOR: Player would move north into the “Funny room”.

OBSERVED BEHAVIOR: Player remains in the same room. Console displays “What?”.