CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

CoffeeMaker.jar

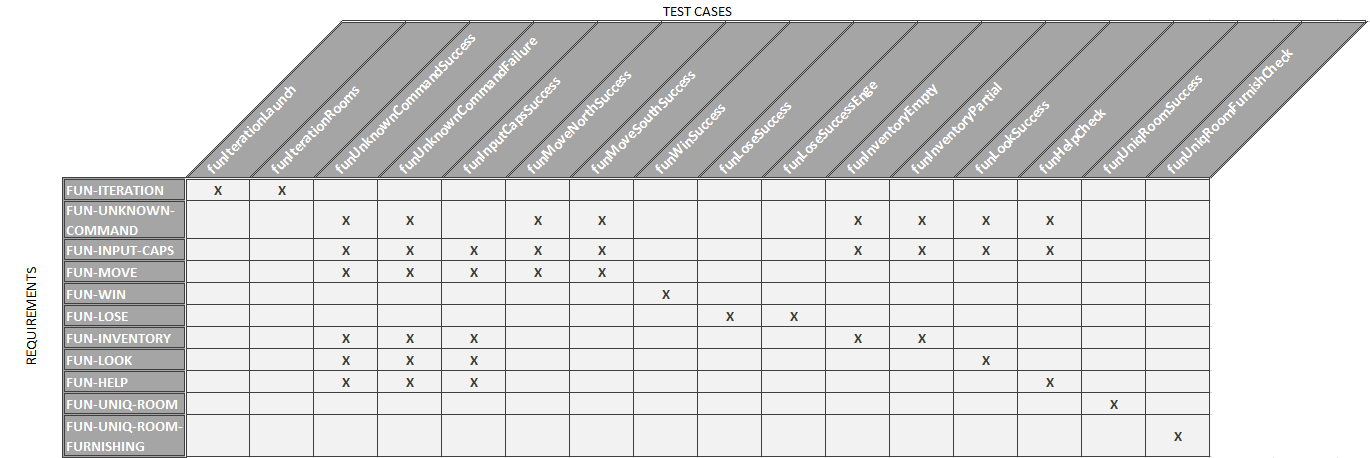
Jeffreys Josephs

Matthew Jeffreys

Summary:

Coffee Maker Quest 1.0 provided us with an adequate introduction to both the planning, and execution aspects of black box software testing. In the early stages, many of our concerns involved the specificity of our test cases. For example, the requirement FUN-ITERATION described a program that should allow for 6 distinct user inputs. It was unclear whether we should approach these as separate tests, or group them together into one test.

Many of the defects that we uncovered involved user input errors. The parsing mechanism failed to accept a variety of inputs at multiple points, throughout the game. A document detailing the tests we devised, and defects we discovered, can be found below.

Traceability Matrix

FUN-ITERATION - At each iteration of the game, the user shall be able enter one of six commands - "N" to go North, "S" to go South, "L" to Look for items, "I" for Inventory, "H" for Help, or "D" to Drink.

**IDENTIFIER**: funIterationLaunch

**TEST CASE**: Tests the program’s user input acceptance functionality, at launch. The program should always prompt the user for 6 possible valid user inputs.

**PRECONDITIONS**:

A new coffeemaker.jar process must be started.

No prior user interaction or commands entered.

**INPUT VALUES**:

java –jar coffeemaker.jar

**EXECUTION STEPS**:

Launch a new game with java –jar coffeemaker.jar, before each input;

Enter one of the instructions (N,S,L,I,D,H) type enter.

**Output values**:

Coffee Maker Quest 1.0

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North"

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**:

The program should wait for user prompt. (Type “D” then press ENTER to exit)

**IDENTIFIER**: funIterationRooms

**TEST CASE**: Tests the program’s user input acceptance functionality. The program should always prompt the user for 6 possible valid user inputs, in all rooms.

**PRECONDITIONS**:

A new coffeemaker.jar process must be started.

No prior user interaction or commands entered.

**INPUT VALUES**:

Type N and press ENTER

**EXECUTION STEPS**:

Launch game with java –jar coffeemaker.jar

(Wait for room and program info)

(Check for instruction prompt)

(repeat 6 times)

Type N press ENTER

**Output values**:

At launch:

“Coffee Maker Quest 1.0”

At each room:

(Room info)

“INSTRUCTIONS (N,S,L,I,D)>”

**POSTCONDITIONS**:

The program should wait for user prompt. (Type “D” then press ENTER to exit)

FUN-INVENTORY - Upon entering "I" for inventory, the player shall be informed of the items that he/she has collected (consisting of Coffee, Sugar, and Cream).

**IDENTIFIER**: funInventoryEmpty

**TEST CASE**: This test will confirm that coffeemaker.jar will display the correct inventory when prompted by the user command. Specifically, it will test for the empty inventory condition.

**PRECONDITIONS**: Tester must launch the process via the command “java -jar coffeemaker.jar” .

**INPUT VALUES**: [I; ENTER]

**EXECUTION STEPS**:

Launch program

Type I→ ENTER

**Output values**:

YOU HAVE NO COFEE!

YOU HAVE NO CREAM!

YOU HAVE NO SURAR!

You see a small room.

It has a Quaint sofa.

A magenta door leads North.

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**:

The program should wait for user prompt. (Type “D” then press ENTER to exit)

**IDENTIFIER**: funInventoryPartial

**TEST CASE**: This test will determine if coffeemaker.jar displays the correct inventory when prompter by the tester.

**PRECONDITIONS**: User must launch coffeemaker.jar

User must have Cream and Coffee in inventory

**INPUT VALUES**: [I; ENTER]

**EXECUTION STEPS**:

Satisfy preconditions:

Type L → ENTER

Type N → ENTER (twice)

Type L → ENTER

Specific to test:

Type I → ENTER

**Output values**:

You have a cup of delicious coffee.

You have some fresh cream.

YOU HAVE NO SUGAR!

You see a Refinanced room.

It has Tight pizza.

A Dead door leads North.

A Smart door leads South.

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**:

The program should wait for user prompt. (Type “D” then press ENTER to exit)

FUN-UNKNOWN-COMMAND - If a player enters a command not specified by FUN-ITERATION, the system shall respond with the phrase "What?".

**IDENTIFIER**: funUnknownCommandSuccess

**TEST CASE**: Enter value not listed as a valid move

**PRECONDITIONS**: Start game

**INPUT VALUES**: [Q]

**EXECUTION STEPS**:

Type Q → ENTER

**Output values**:

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North"

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Expect to see the input value followed by "What?"

**IDENTIFIER**: funUnknownCommandFailure

**TEST CASE**: Enter value listed as a valid move

**PRECONDITIONS**: Start game

**INPUT VALUES**: [L]

**EXECUTION STEPS**:

Type L → ENTER

**Output values**:

There might be something here...

You found some creamy cream!

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Expect to execute the entered valid move

FUN-INPUT-CAPS - The system shall be case-insensitive in regards to input values; that is, it shall accept capital and lower-case letters and treat them as equivalent.

**IDENTIFIER**: funInputCapsSuccess

**TEST CASE**: Enter lower-case letter of a valid move

**PRECONDITIONS**: Start game

**INPUT VALUES**: [I]

**EXECUTION STEPS**:

Type I → ENTER

**Output values**:

There might be something here...

You found some creamy cream!

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Expect to look in the current room for an item and find out if there was anything in the room or not.

FUN-MOVE - The system shall allow a player to move North only if a door exists going North, and South only if a door exists going South.

**IDENTIFIER**: funMoveNorthSuccess

**TEST CASE**: Enter move to go North where door exists

**PRECONDITIONS**: Start game

**INPUT VALUES**: [N]

**EXECUTION STEPS**:

Type N → ENTER

**Output values**:

You see a Funny room.

It has a Sad record player.

A Beige door leads North.

A Massive door leads South.

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Player should have moved to the room to the North

**IDENTIFIER**: funMoveSouthSuccess

**TEST CASE**: Enter move to go South where door exists

**PRECONDITIONS**: Start game and then already have moved North where a door to the South exists

**INPUT VALUES**: [N;S]

**EXECUTION STEPS**:

Type N → ENTER

Type S → ENTER

**Output values**:

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Player should have moved back to the starting room.

FUN-WIN - The player shall win the game if and only if Coffee, Sugar, and Cream have been collected by the player and then drunk.

**IDENTIFIER**: funWinSuccess

**TEST CASE**: Test will confirm that the program coffeemaker.jar will exit in a win game scenario, correctly.

**PRECONDITIONS**:

Tester must launch program via the command “java -jar coffeemaker.jar”. Tester must obtain cream, coffee, and sugar.

**INPUT VALUES**: [“D”; ENTER]

**EXECUTION STEPS**:

1. Tester should launch coffeemaker.jar via the command java -jar coffeemaker.jar

2. Repeat following steps 5 times:

Type “L” press ENTER.

Type “N” press ENTER.

3. Type “L” to collect any item in the 6th room.

4. Type “D” to drink coffee and end game.

**Output values**: You drink the coffee beverage and are ready to study.

You win!

**POSTCONDITIONS**: The program should simply terminate and exit to command line.

FUN-LOSE - The player shall lose the game if and only if the player Drinks but has not collected all of the items (Coffee, Sugar, and Cream).

**IDENTIFIER**: funLoseSuccess

**TEST CASE**: This test will determine if the program coffeemaker.jar will exit in a losing game scenario, in the correct manner. Specifically, it will test the condition the results when the user attempts to “Drink” before obtaining the sugar.

**PRECONDITIONS**:

Tester must launch program via the command “java -jar coffeemaker.jar”.

Tester must obtain cream (inventory item) in the first room.

Tester must obtain coffee (inventory item) in the third room.

**INPUT VALUES**: [“D”; ENTER]

**EXECUTION STEPS**:

1. Tester should launch coffeemaker.jar via the command java -jar coffeemaker.jar

2. Repeat following steps 2 times:

• Type “L” press ENTER.

• Type “N” press ENTER.

3. Type “L” to collect any item in the 3rd room.

4. Type “D” to drink coffee and end game.

Without sugar, the coffee is to bitter. You cannot study.

You lose!

**POSTCONDITIONS** The program should simply terminate and exit to command line.

**IDENTIFIER**: funLoseSuccessEnge

**TEST CASE**: This test will confirm that coffeemaker.jar will produce a losing scenario, in the event that that tester fails to pick the “cream” inventory item, but attempts to pick up the sugar twice. This may expose programming error, pertaining to the inventory system.

**PRECONDITIONS**: Tester must launch the process via the command “java -jar coffeemaker.jar”

Inventory must contain coffee and cream.

**INPUT VALUES**: [L; D; ENTER]

**EXECUTION STEPS**:

Launch program

(set pre conditions)

Type N → ENTER (twice)

Type L → ENTER

Type N → ENTER (3 TIMES)

Type L → ENTER

Specific test

Type L → ENTER

Type D → ENTER

**Output values**: Without cream you get an ulcer, and cannot study.

You lose!

**POSTCONDITIONS**: After the loading scenario notification, the program should terminate and return to to command prompt without incident.

FUN-LOOK - Upon entering "L" for Look, the player shall collect any items in the room and those items will be added to the player's inventory.

**IDENTIFIER**: funLookSuccess

**TEST CASE**: Look into inventory where cream has been collected.

**PRECONDITIONS**: Start game and iterate through rooms and have collected cream.

**INPUT VALUES**: [L;I]

**EXECUTION STEPS**:

Type L → ENTER

Type I → ENTER

**Output values**:

YOU HAVE NO COFFEE!

You have some fresh cream.

YOU HAVE NO SUGAR!

You see a Small room.

It has a Quaint sofa.

A Magenta door leads North

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Inventory should show that the user has collected cream and no other items.

FUN-HELP - Upon entering "H" for Help, the player shall be shown a listing of possible commands and what their effects are.

**IDENTIFIER**: funHelpCheck

**TEST CASE**: Check help menu to view possible views and their effects

**PRECONDITIONS**: Start game

**INPUT VALUES**: [H]

**EXECUTION STEPS**:

Type H → ENTER

**Output values**: Help menu

**POSTCONDITIONS**: Player should see a list of possible moves and what their effects are.

FUN-UNIQ-ROOM - Each room in the house shall have a unique adjective describing it.

**IDENTIFIER**: funUniqRoomSuccess

**TEST CASE**: Iterate through two rooms to the North to see that it has a unique adjective.

**PRECONDITIONS**: Start game and move North through two rooms.

**INPUT VALUES**: [N]

**EXECUTION STEPS**:

Type N → ENTER

Type N → ENTER

**Output values**:

You see a Refinanced room.

It has a Tight pizza.

A Dead door leads North.

A Smart door leads South.

INSTRUCTIONS (N,S,L,I,D)>

**POSTCONDITIONS**: Room should have a unique adjective assigned to it.

FUN-UNIQ-ROOM-FURNISHING - Each room in the house shall have one and only one unique furnishing visible to the user upon entering the room.

**IDENTIFIER**: funUniqRoomFurnishCheck

**TEST CASE**: This test will confirm that the program coffeemaker.jar will display unique furnishing in each room. The tester will be required to make a note of the furnishings.

**PRECONDITIONS**: Tester must launch program via the command “java -jar coffeemaker.jar”.

**INPUT VALUES**: [N, ENTER]

**EXECUTION STEPS**:

Type N → ENTER

Type N → ENTER

**Output values**:

Ignore:

You see a <Room description>

Make note of:

<Furnishing descriptions>

**POSTCONDITIONS**: The program should prompt the user for input. Type “D” and press ENTER to terminate and exit to the command line

Defects:

DESCRIPTION: FUN-HELP - Valid move entered prompts "What?"

SUMMARY: When entering "H", which represents the help menu, an unknown command prompt is listed.

REPRODUCTION STEPS: Type H→ ENTER

EXPECTED BEHAVIOR: A list showing a list of possible commands and what their effects are.

OBSERVED BEHAVIOR: "What?"

IMPACT: Defect will prevent user from accessing the Help menu and thus will be unsure of the functionality of the game if he/she is not familiar with the game.

SEVERITY: MAJOR

DESCRIPTION: FUN-INPUT-CAPS - input case-sensitive rather than case-insensitive

SUMMARY: Upon entering a valid move, the input should be case-insensitive, but when entering "n", which would result in a move North, instead prompts the unknown command error.

REPRODUCTION STEPS: Type n→ ENTER

EXPECTED BEHAVIOR: Move to the next room to the North.

OBSERVED BEHAVIOR: "What?"

IMPACT: User will be unable to use all commands as lower case and thus will cause an annoyance if user depends on only lower case input.

SEVERITY: MINOR

DESCRIPTION: FUN- ITERATION – wrongprompt

SUMMARY: The user prompt fails to list H for the help menu.

REPRODUCTION STEPS: Enter → N

EXPECTED BEHAVIOR: Move to the next room to the North.

(Room description)

INSTRUCTIONS (N,S,L,I,D,H)>

OBSERVED BEHAVIOR:

(Room description)

INSTRUCTIONS (N,S,L,I,D)>

IMPACT: User will be unable to access Help menu and will also not know of its existence. This could cause confusion of the user if the user is unfamiliar with the game.

SEVERITY: NORMAL