SE 3XA3: Test Report
PyCards

Team 2
Aravi Premachandran premaa
Michael Lee leemr2
Nikhil Patel patelna2

December 5, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
December 4	1.0	Initial Revision

1 Functional Requirements Evaluation

1.1 Event Handling

Key Bindings (T??)

1. KB1
   Type: Functional, Dynamic, Manual

   Initial State: Application instance that is capturing user input

   Input: Keyboard press of one of COMMAND_KEYS by user

   Expected Results: The corect response from BOUND_ACTIONS executes

   Success/Failure: Success

1.2 Widget Callbacks (T??)

1. WC1
   Type: Functional, Dynamic, Manual

   Initial State: Menubar widget waiting for user interaction

   Input: User clicks on a label in the menu bar

   Expected Results: If the user clicks or hovers on a cascading menu it will expand to show all contained submenu labels. If the user clicks on a menu label it will perform its associated callback function.

   Success/Failure: Success

2. WC2
   Type: Functional, Dynamic, Manual

   Initial State: Toolbar widget waiting for user interaction

   Input: User clicks on a toolbar button

   Expected Results: The application will execute the appropriate function based upon the toolbar button clicked. There is a bijection between the actions in BOUND_ACTIONS and toolbar buttons

   Success/Failure: Success

3. WC3
   Type: Functional, Dynamic, Manual

   Initial State: Card widgets waiting for user click

   Input: Primary button click on card widget

   Expected Results: If the card selection is valid (see Functional Klondike Requirements in PyCards SRS for selection constraints) the card is highlighted while the mouse button remains pressed and is redrawn to follow the cursor. If the card selection is invalid no visible changes are made to the window.

   Success/Failure: Success

1.3 Game Logic

Card Selection (T??)

1. PC1
   Type: Structural, Dynamic, Automated

   Initial State: Klondike game is loaded

   Input: User selects card with mouse click

   Expected Results: Clicked card is highlighted and tracks mouse cursor if valid selection. If invalid selection the click is ignored. Criteria for valid selection is defined in the SRS

   Success/Failure: Success

2. PC2
   Type: Structural, Static, Manual

   Initial State: The rules of the game written using first-order logic

   Input: A sequence of possible executions that covers the different conditions

   Expected Results: Truth value whether the result of the execution conforms to the rules of the game

   Success/Failure: Success

2 Nonfunctional Requirements Evaluation

2.1 Interoperability

Operating System Compatibility (T??)

1. OSX
   Type: Structural, Dynamic, Manual

   Initial State: Executable for program is on target machine running a standard install of OS X OSX_VERSION.

   Input/Condition: User locates executable and launches program

   Expected Results: Either a successful launch or a message from Gatekeeper alerting user that program was created by an unidentified developer

   Success/Failure: Success

2. OS2
   Type: Structural, Dynamic, Manual

   Initial State: Executable for program is on target machine running a standard install of Ubuntu UBUNTU_VERSION.

   Input/Condition: User locates executable and launches program

   Expected Results: Either a successful launch or a failure caused by incompatible packaging, dependencies, or other causes

   Success/Failure: Success

3. OSWIN
   Type: Structural, Dynamic, Manual

   Initial State: Executable for program is on target machine running a standard install of Windows WIN_VERSION.

   Input/Condition: User locates executable and launches program via double-click

   Expected Results: Depending on the system, either a successful launch or application crash due to missing VC++2008 binaries (required even after building executable)

   Success/Failure: Success

Portability (T??)

1. PR1
   Type: Structural, Dynamic, Manual

   Initial State: Executable for application is located on a removable USB drive

   Input/Condition: User launches program executable from a removable USB drive

   Expected Results: The program launches exactly as if it had been launched on the same system from the internal hard drive

   Success/Failure: Success

2. PR2
   Type: Structural, Static, Manual

   Initial State: The VC++ 2008 binaries required for execution of the program are not found on the host system

   Input: The program fails to launch on a Windows-based machine that does not have the VC++ 2008 redistributable package installed

   Expected Results: All of the users are able to successfully follow the provided instructions to install the VC++ 2008 redistributable package

   Success/Failure: Success

2.2 Usability (T??)

Navigation

1. UN1
   Type: Structural, Dynamic, Manual

   Initial State: The existing implementation is running and idle on the user's machine, and is waiting for user interaction

   Input/Condition: A group of users are asked to perform each of the actions defined in BOUND_ACTIONS

   Expected Results: The majority of users successfully perform the different actions, completing each within in a period of under MAX_FIND_TIME

   Success/Failure: Success

2. UN2
   Type: Structural, Dynamic, Manual

   Initial State: Users have completed and taken note of the results of the previous test. Our application, PyCards, is now running and idle, waiting for user interaction

   Input/Condition: The group of users are asked to perform each of the actions defined in BOUND_ACTIONS

   Expected Results: The majority of users successfully perform the different actions, completing each within in a period of under MAX_FIND_TIME and in time less than or equal to what was required to perform the same action using the existing implementation

   Success/Failure: Success

Playability (T??)

1. UP1
   Type: Structural, Dynamic, Manual

   Initial State: Existing implementation is running, with a game in progress

   Input/Condition: A group of users are asked to play a game and rate it based ease of use using a scale from 1-5 where 5 is very user friendly

   Expected Results: The majority of users give the game an average rating above 3

   Success/Failure: Success

2. UP2
   Type: Structural, Dynamic, Manual

   Initial State: Users have completed the previous test. PyCards is now running, with a game in progress

   Input/Condition: A group of users are asked to play the game and rate it based ease of use using a scale from 1-5 where 5 is very user friendly

   Expected Results: The majority of users give the game an average rating above 3 and greater than or equal to the rating given in the previous test

   Success/Failure: Success

2.3 Product Integrity (T??)

Resource Loading

1. RL1
   Type: Structural, Dynamic, Manual

   Initial State: The 'cardsets' directory is missing or corrupt

   Input/Condition: When loading the application prompts users to either re-download the application or download specifically the cardsets directory

   Expected Results: The majority of users are able to find the repository for the program and re-download the executable or the cardset directory

   Success/Failure: Success

2. RL2
   Type: Structural, Dynamic, Manual

   Initial State: The 'tiles' directory is missing or corrupt

   Input/Condition: Application attempts to load images from the 'tiles' directory

   Expected Results: An exception is thrown and handled by using a solid color tile as the window background

   Success/Failure: Success

2.4 Security (T??)

System Permissions

1. SP1
   Type: Structural, Dynamic, Manual

   Initial State: Application is located on the host machine's file system

   Input/Condition: User launches the application

   Expected Results: The application should launch without asking for administrative (root) privileges

   Success/Failure: Success

3 Unit Testing

4 Changes Due to Testing

5 Automated Testing

Automated testing was done using the unittest.py python module. This allowed us to create test suites for the various methods and use assert statements to specify preconditions and postconditions.

6 Trace to Requirements

Req.	**Modules**
FR1	
FR1	
FR3	
GR1	
GR2	
GR3	
REQS	T??, T??, T??, T??
REQS	T??, T??, T??, T??
REQS	T??

Table 2: Trace Between Requirements and Modules

7 Trace to Modules

8 Code Coverage Metrics

Code coverage was done using the coverage.py python module and running it against our project. The coverage.py module works in three phases: first it executes our code and monitors the statements that were executed, then it examines the source to determine which lines could have run, and finally it provides a report in the desired format (ie. text, html, annotated source).