Class Notes

Tuesday Nov 5

Valid Criteria for postal code

1. Alphabetic
    1. Numeric
2. Exists (not empty)
    1. No input
3. Two chars long
    1. One char input
    2. thre char input
4. Postal code
    1. Non postal code
5. On the list of valid inputs
    1. Postal code not on the

[1,2,3,4]

[0,1,2,3]

A shipping system routes items by code which as follows

1. Start with a three letter prefix CHI or DET indicating Chicago or Detroit. These are the only two destinations, these are handled differently by the system.
    1. ~~Three letters long~~
    2. ~~Upper or lower case~~
    3. ~~DET~~
    4. ~~CHI~~
    5. ~~Two letters long~~
    6. ~~Four letters long~~
    7. ~~Three but not all letters~~
    8. ~~Thee letter code other than DET or CHI~~
2. ~~The fourth position is either a * or /, they are treated the same. This is to accommodate legacy codes.~~
    1. Character not * or /
3. ~~The fifth position is a product category indicated by a letter from ASCII A-T and all processed the same~~
    1. {A B ... T, a, b...t} categorical data.
    2. Any letter not on the list
    3. Ascii character
    4. Ordinal data {A-T} {a-t}
4. The last three position are either 1, 2 or 3 numeric digits long represent a sales code and all processed the same
    1. ~~exactly one~~
    2. exactly two
    3. ~~exactly three~~
    4. They are digits
5. All codes must 6- 8 characters long

~~1. 6-8~~

6. There are no embedded blanks
7. If the code is invalid and error should be printed in the log and the code ignore.
8. The code should not be case sensitive chi = CHI?

1. Critique the spec.
2. Develop a set of equivalence class for test input
    - Listing the valid criteria
3. Choose test cases using boundary value analysis.
    - Choose the valid test cases first
    - Break each test case to get an invalid

Test Cases

1. DET*a879
2. chi/T7
3. DE*a879
4. chic/T7
5. D3T*a879
6. DAL*t89
7. DET\879
8. no input

“The Similarity Principle.”

1. Make the valid cases as dissimilar as possible
2. Make each invalid test case exactly like a valid case but differing in one criteria

copy(src, dest)

copy(dest,src)

User Story Examples

Atm withdraw user story

Example: A user story that is testable with actual data

Example 1: main succes

Rod goes to the bank atm, swipes card and enters pin 78987 and selects withdraw from option, selects checking account, and specifies $100 dollars. The money is dispensed and Rod goes away.

Example 2: no fund

Rod goes to the bank atm, swipes card and enters pin 78987 and selects withdraw from option, selects checking account, and specifies $100 dollars but only has $80 in his account. The request is declined and session ends.

An automated parking attendant on the street

1. user presses start to activate display
    1. Display
    2. Doesn’t start
2. Selects amount of time wanted for parking
    1. user selects max time
    2. user selects min time
    3. User does not respond
3. Offers payment options
    1. user selects option
    2. user can’t make a selection
4. User present payment card
5. Use gets ticket to place in dash

is there a way to cancel the transaction?

Does the transaction time out?

Integration testing

interface independent

Automated tool

black box test cases against the build

Interface testing

Testing interfaces against a mockup of the application

Full acceptance

Application plus interfaces

Scenario based testing.

Beta test

UAT in situ in the operational environment

UAT;

1. Usability – Qualitative
2. Domain driven design – what is there and what thought to be there.
3. Qualitative testing, personas

Scenario based testing:

1. Scenarios have goals because we are looking at the system as a black box, from the user perspective.
2. Goal attainment is not the same as success or failure.
3. Incorrect logic. IEEE “Correct” consistent with business process
4. Incorrect requirement -> bad spec -> broken test “false negative”
5. Manual by the domain users but mostly automated
6. We have done the integration tests and the interface test.

Integration testing (smoke test)

1. Each feature in the build passes the acceptance tests
2. Don’t care about user goals and usability
3. Just running functional tests – no scenarios
4. Automated.

5. This should be passed before we go to UAT.
6. Tested without a minimal test interface

Interface testing

1. Run the scenarios against a mockup application and a real interfaces

November 13

Recap

- Software testing is a special case of testing in the large
- We use the same protocols adapted for software
- Need to control for bias – subjective errors that are made.
- Protocols remove the opportunity to make judgment errors
- Systematic approach to test development and execution
    - Comprehensive – nothing falls through the cracks (hopefully)
    - Repeatable – checked by others
    - Can be automated (GIGO)
    - Auditable – contract compliance, regulatory compliance
    - Reusable
    - Validation and verification of the tests

Black Box

- Requires a specification
- Best practices for specs
- Review the spec for quality
- Equivalence classes
    - Assume that the programmer is implementing the logic in a competent
    - Reduce testing – test types of data rather than individual data points.
    - Cause and effect graphing - equivalence classing the output
- Boundary value analysis
    - Where bugs are mostly likely to occur
    - Sensitive to the type of data

- Has gaps and holes

White Box Testing

- Exercise the logic of the application
- Coverage here is defined in terms of executing code/logic
- Surprise functionality
- Primary use is to get logic coverage
    - Diagram and test business logic
        - Develop test cases
    - Diagram out algorithms used in the app
        - Run the test cases to verify the logic
    - Run the test cases against
- Representing use cases and user stories
    - Apply black and white box testing

Early Dev Tools

- Sonarqube code quality
- Tell you how good your test cases are compared to the code
- Unit testing tools
    - Automated tools used by developers
    - And acceptance tests

Integration test

- Not done through an interface
- Always automated
- The testing of the individual features

UAT

- Usability testing
- Qualitative Testing – exploratory testing
- E2E testing
- Interface testing
    - Running our e2e tests through the interface on a mockup of the application
    - Mockup just returns the correct value for each test case
- Combine for E2E testing
    - Scripting automation
        - Selenium
        - Rational Functional Tester
        - Protractor Appium
- Develop a test suite based on user scenarios
    - user stories and examples
    - our E2E is used for regression testing