W. P. CAREY
SCHOOL *of* BUSINESS
ARIZONA STATE UNIVERSITY

CIS 345 – Business Information Systems Development II – Fall 2014

Evaluating Software Projects

Software applications have two components – functionality (user-oriented) and quality of the programming code (developer-oriented). CIS 345 projects will be evaluated on both counts. While functionality is of paramount importance, the structure within the program also counts.

Examples of assessing quality:

- A program could possibly implement the same logic using 15 nested if statements or a switch statement. The second option would be the preferable one since it would be easier to implement, comprehend, neater in both reading/writing, and easier to edit/maintain.
- Commenting should enable another person to read and understand what your code does.
    - Variable and method names should be descriptive, e.g. "userChoice" is much more clearer than "pleasework3"
    - If you have 5 variables declared and all of them have similar data types and/or purpose, have one comment, rather than 5 different ones.
    - The purpose is not to flood the code with comments. It is so that another person can easily understand the logic that you have implemented.
    - If your method names and variable names are focused and self-descriptive, your code will be self-documenting and you often do not need to have as many in-depth descriptions of comments
- Tasks should be distributed across methods. Smaller, focused methods are better than long running methods than do everything and do not have one clearly defined purpose.
    - If you cannot come up with a method name which describes everything that a method does, then it *probably* is doing too much.
    - If you cannot describe in one short line (a comment!) everything that a method does, it is *probably* doing too much.

Following is a listing of the major functionality that needs to be found in the project as well as some of the other factors that determine quality of the source code.

A fully functional project will

- Display a list of flights
- Add a new Flight
- Allow a flight to be selected to display a detail view of one flight, and allow editing.
- Maintain and show passenger information per flight. *While there can be common sample data across flights, the structure of the program should allow for a different instance of passenger array in each flight.*
    - Display a passenger as flagged if its boolean flag is set to true
- Search for a passenger by first and last name.
- Be able to send the passenger manifest to the TSA DLL
    - Process the returned array (which might be of size 0 or more).
    - For each passenger in the returned array, it should mark passengers with the same name on the manifest as flagged.

General rubric for evaluating projects (beyond checking for functionality)

An exemplary ("A+") application:

- Meets all the requirements for an excellent application (see below).
- Has both programming code and comments that are elegant and concise (just right, not overly complex, not redundant and long).
- Does not fail at all in any situation.
- Is hard to improve upon while keeping features constant. It is perfect.

An excellent ("A"/"A-") application:

- Is fully functional
- Handles all exceptions
    - Verifies and validates input
    - Able to handle bad user input
    - Does not crash except in very exceptional situations

- Has excellent commenting standards.
    - Descriptive variable and method names
    - Comments indicate purpose and internal logic
- Functionality is distributed across various classes and methods
    - Methods are focused and do primarily one main thing
- Code is largely concise and non-redundant.
- Program is well done in an object-oriented fashion
    - Very little use, if any, of static variables and methods
    - Uses primarily instance variables and methods
    - Has methods and classes that will be reusable in future projects i.e. they are structured generically and all configuration happens through specifying parameters or properties.

A Good (“B+”/”B”) application does not meet all the standards of an excellent application and:

- Crashes on rare occasions
- Code may be sparsely redundant/repetitive in places
- Internal logic may be unclear in places
- Some variable names or purpose is not clear
- Uses a few static method and variables

A Working (“B-“) application does not meet all the standards of a good application and:

- Crashes more frequently
- Some functionality is absent
- Commenting is less clear
    - Variable and method names are not clear and self-explanatory
- Internal logic is hard to follow in places
- Many static variables and methods used
- Methods are longer and fewer
- Methods are not focused – one method does many things. There is little logical subdivision of logic and tasks.

A Moderately functional (“C+”/”C”) application does not meet prior standards and:

- Has major areas of functionality missing
- Does not do much input validation
- Application is prone to crashing
- Code is inefficient and incorrect at places
- Largely not object-oriented

A non-working application (“D”/”E”):

- Does not properly fulfill most criteria.