cs3307a – Object oriented analysis and design

Design Inspection Instrument

Instructions:

- The purpose of this document is to assist in the inspection of object-oriented design.
- Under each question is a choice of answers; please choose one (either replace the box with a checkmark or highlight it)
  ☐ yes ☐ no ☐ partly, could be improved
- Two types of comments are required under each question. One is your analysis. The other is your finding (in the form of a comment). The analysis would typically show how you arrived at the finding.
- Add new lines as necessary for your analysis or findings.

Scope of the system to be considered for inspection:

- With reference to Appendix B – Dashboard Screens, take Demo 1 feature, focusing on that part of the code that produces one Dashboard summary.
- Visualisation code is out of scope of this inspection.

++++++++++++++++++++

Structural correspondence between Design and Code:
Are all the classes and interrelationships programmed in the application explicitly represented in the class diagram of the system?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
Yes the code implemented follows the use case diagram very well. The relationships between the respective classes are displayed in a UML diagram with the proper cardinality and associations.
Comment on your findings:
Followed the "flow" of a program execution of publication.csv data. Checked the cardinality and relationship status of each class as I stepped through the program.

Functionality:
Do all the programmed classes perform their intended operations as per the requirements?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
The classes all work as defined, no class has extenuating functions not related to the purpose of that class.
Comment on your findings:
Checked the function declarations and function prototypes of the class and determined whether or not it was appropriate for the declared class.

Cohesion:
Do the methods encapsulated in each programmed class, together perform a single, well defined, task of the class? (High-Cohesion: the functionalities embedded in a class, accessed through its methods, have much in common, e.g., access common data)

☐ Yes ☐No ☐**Partly (Can be increased)**

Comment on your analysis:
Our program has high cohesion because each page works on the same data pointer passed through from the initial load screen.
Comment on your findings:
I made a list of all the classes that used the shared_ptr throughout the program. Most classes required it in some sort of way to work. I.e. the verify classes needed to access the data to verify it was working. The analyze classes needed to access the data in order to generate the data structures to store the data.

Coupling:
Do the programmed classes have excessive inter-dependency? (High Coupling: In this case a class shares a common variable with another, or relies on, or controls the execution of, another class.)

☐ Yes ☐No ☐**Partly (Can be reduced)**

Comment on your analysis:
Our program is slightly coupled but not in the conventional sense. The flow of the program is specified in a linear flow whereby the next classes are called once the previous classes are finished. This is more of a design metric as it is used to ensure the user is using the program as intended. For example, the data must first be loaded, and the classes required for data verification are called in order to allow the user to finish the data. Afterwards, the analyze page will generate the data structures for the visualization classes. In these cases the classes require on the execution of the previous classes but this is as intended.
Comment on your findings:
Followed the execution through the program with the debug menu.

Separation of concerns:
Is the scoped problem decomposed into separate concerns where each concern is encapsulated in a construct such as a class with well-defined interface and cohesive functions with minimal of connections with other concerns?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
Each part of the program is encapsulated into a class with a defined function with very minimal connections to other concerns. I.e. the concern of data verification does not concern the generation of the data variables for visualization.
Comment on your findings:
Checked the defined function of each program and checked their roles.

Do the classes contain proper access specifications (e.g.: public and private methods)?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
Yes most classes have proper accessors and getters.
Comment on your findings:
Checked the function header declarations.

Reusability:
Are the programmed classes reusable in other applications or situations?

☐ Yes, most of the classes ☐No, none of the classes ☐ **Partly, some of the classes** ☐Don't know

Comment on your analysis:
This program is very specific. Most classes are designed for the analysis and verification of csvs. However, the data structures used to store the data for visualization and the functions used to generate the error lists are all modular and reusable.
Comment on your findings:
Checked the usage of each class. Some classes are used by multiple others.

Simplicity:
Are the functionalities carried out by the classes easily identifiable and understandable?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
Each class is named intuitively and behaves as defined.
Comment on your findings:
Checked the usage of each class.

Do the complicated portions of the code have /*comments*/ for ease of understanding?

☐ **Yes** ☐No ☐Partly (Can be improved)

Comment on your analysis:
Yes, designers did a good job of labeling their code and explaining how it works and why.
Comment on your findings:
Checked the most complicated functions for comments and notes.

Maintainability:

Does the application provide scope for easy enhancement or updates? (e.g., enhancement in the code is not anticipated to require too many changes in the original code)

☐ **Yes** ☐No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis:
The code is well written to accommodate changes and maintenance in the future. It has lots of general usage templates for incorporation of different data types later on.
Comment on your findings:
Checked the amount of Templates and void pointers in the code.

Efficiency:
Does the design introduce inefficiency in code (e.g., causes too many nested loops or delays in concurrent processing)?

☐ Yes ☐**No** ☐Partly (Can be improved) ☐Don't know

Comment on your analysis:
No, the linear flow is very efficient for data processing. The nested loops used in the data structure generation can be parallelized.
Comment on your findings:
No inefficiencies found when loading/running analysis. All csvs load in less than 1 second.

Depth of inheritance:
Do the inheritance relationships between the ancestor/decendent classes go too deep in the hierarchy? (The deeper a class in the hierarchy, the greater the number of methods it will probably inherit from its ancestors, making it harder to predict its behaviour).

☐ Yes ☐**No** ☐Partly (Can be improved)

Comment on your analysis:
No the inheritance is not deep. Can't find evidence of deep hierarchy.
Comment on your findings:
N/A

Children:
Does a parent class have too many children classes? (This could possible suggest an abstraction problem.)

☐ Yes ☐**No** ☐Partly (Can be improved)

Comment on your analysis:
No, each class has few if any children classes.

Comment on your findings:
N/A

Behavioural analysis:
From the system's requirements, **create several scenarios** starting from the **user's** point of view: consider identifying one or more **typical** scenarios (e.g., those expected to be used with high frequency) and one or more **low-frequency** scenarios .

Each scenario is described as follows:

i) Title of scenario
ii) Anticipated frequency of use (high, normal, low)
iii) End-user trigger (starting point) for the scenario.
iv) Expected type of outputs.
v) List of bullet points linking end-user inputs and identifying all the key features of the system expected to be "touched" by the scenario and producing the anticipated outputs.

Follow the code (structured walkthrough) to ascertain whether this scenario is properly implemented both in terms of logic and design.

Scenario 1:
Title: Analyze publication.csv
Frequency of Use: high
End-User trigger: loading a publication csv file
Expected outputs: tree list display and graph display
End-user inputs:

i) Open Program
ii) Hit publication button
iii) Go to Verify Page
iv) Click ignore all errors
v) Click Confirm Changes
vi) Click Analyze
vii) Graphs are now displayed along with Tree List
viii) (Optional) User changes the date range and clicks Filter

Comment on your findings, with specific references to the design/code elements/file names/etc.:
The first stage is very simple. There really is only 1 scenario that the user can conduct. The program is rather linear and the user cannot go back and make any changes without restarting the whole flow from step ii).

(Note: expand here as necessary for each scenario)

END.