Software Engineering Project

Part 4 – The Design and Implementation

The objective of this assignment is to proceed with the ***System Design*** and to prepare for the presentation of your project. The basic OO paradigm serves equally well in modeling the real-world requirements and the software implementation. However more detail is required for a working implementation than it was for during domain analysis.

Here are the principles of good design. Follow them.

1. *It is important to define a good architecture*
   The architecture is fundamental to producing and maintaining many releases of the system.
2. *Separate domain and implementation*
   Keep the logical and physical aspects of a system independently.
3. *Take a progressive iterative approach*
   The advantage of the Object Oriented approach is that during the design there is no need to create a model that is different from the one created during analysis.
4. *Design for Reuse*
   Possible levels of reuse form a hierarchy. The higher the form of reuse is in the hierarchy, the more significant is its benefit.

During the design a series of small executable releases are defined that will eventually be integrated into a full-scale system. Each executable release is a piece of a system built to test or to expose the design of a specific area. It is implemented, revised, and then refined to fit into the final model.

Here are the steps that you should follow:

1. ***Define the initial architecture***
2. ***Plan executable releases***
3. ***Develop executable releases***.

Object oriented design is also an iterative process of incrementally adding more details. The design may highlights inconsistencies in the analysis model you have created and make it necessary to iterate through analysis one more time. In fact, do not be surprised if you find yourself iterating through analysis and design several times before you are through.

1 - Define the initial architecture: (20% of the grade)

A good architecture is organized in layers. Each layer provides the basis for the layer above it. The lower layer provides logical interfaces to system-dependent hardware and software services. This allows the system to be ported to other platforms by rewriting a single layer. Layers are divides in loosely coupled partitions, each providing one kind of service. These partitions are called **class categories.** Each class category is designed and implemented separately. As the system evolves the implementation can be changed without affecting the rest of the system.

It is essential to maintain a balance between over-emphasizing future maintainability and extensibility requirements of the system and under-defining the structure of the architecture. A bad architecture results in systems that over time are difficult to maintain and extend.

The architecture includes:

1. the partitioning of system classes into groups (class categories) that will be encapsulated to allow parts of the system to be developed in details without having to look at all the system at one time.

To do list:

- Create a *class-category diagram* that shows the chosen categories, or high-level organizational structures, and their visibility to each other. The class category diagram represents the architecture of the system.

    - Create the class categories. A class category encapsulates classes similar to how a class encapsulates interfaces. Class categories are groups around major architectural functions. Typically, the major grouping of class categories includes different high-level architectural structures such as the database, the GUI, data-structure libraries, and so on. If the domain is large it can be broken into more than one class category
    - Define the visibility to indicate the exposure of the content of one class category to another class category. In a good design, some categories do not need to see others. However there has to be some visibility. Connect the categories if there is some visibility. This implies the ability to use classes from that target category. Classes in a class category that can be accessed by other class categories are said to be **exported**. Classes for exclusive use by a class category are **local**. In the class diagram you show the use of a class (an **import)** within another class category by using the "uses' relationship between them.
    - The goal is to define class categories with high degree of cohesion and weakly coupling to other class categories.

Deliverable #1 (20%) - Create a *class-category diagram* showing the categories to be used.

2 - Planning executable releases: (20% of the grade)

Do not try to design and implement the entire system in one large leap. Rather build the system in a series of identifiable groups of related functions and tasks, or executable releases.

An executable release is like a "mini-system" with a predefined goal. Each release contains actual working code that will be used in the complete system or in other executable releases. An executable release must be tested.

Set up a good design plan to identify potential executable releases and their needs. You also set up an incremental plan to build the system using successive executable releases that you will integrate to provide the complete system.

The planning must be based on **reducing development risk.** Here is an example of a ***Plan*** for an executable release. Add a timetable to the plan.

Executable Release: Registering a Guest

Goal:

Verification and successful use of the navigational paths and registration logic for the registration of a person to a conference.

Classes to be implemented:

class_name1, *class_name2*, and *class_name3*.

Previously implemented classes to use:

class_name0

Use case to be implemented:

Determining the guest identify
Determining if guest is in the list (if necessary)
Determining if payment has been performed
etc..

Inputs:

a dummy list of guest, a list of received payments, etc...

Outputs:

The data is required to compute the 1% of cost to charge the conference organizer which is part of the output of the system.

Time for completion:

To be completed by .....

Deliverable #2 (20%) - Create a *plan of executable releases*. Add your plans to the 3 ring binder folder. Follow the plan during the project development. At the end of the project or at the deadline of each release, write down the effective status of the release (for ex. Not started, Started but Incomplete - Completed by (write here the date of completion), Completed as planned.). Use GitHub or any other software to maintain trace of your plan. Produce the log and add it to your 3 ring binder folder.

3 - Developing executable releases – the steps of design: (50% of the grade)

Develop the executable releases by designing and implementing the classes and the detailed specification of operations. Here are the tasks that you will perform. There is no fixed order in which these tasks must be done.

1. ***Add any control class***
2. ***Define new classes to support the implementation***
3. ***Define operations needed to carry out the implementation***
4. ***Define algorithms to implement operations***
5. ***Provide implementations of the relationships in the analysis model***
    - Observe that when implementing a Physical Containment, the implementation can be specified in the following way:
        - *Containment by value*: contains an object. This choice implies that the lifetime of the target object is equivalent to the lifetime of the source object (i.e. construction and destruction of the target occur as a consequence of the construction and destruction of the source).
          For example a *Event* is contained in the *Olympic Games*. If the *Olympic Games* are cancelled the *Event* will not take place.
        - *Containment by reference*: contains a pointer or a reference to an object. This choice indicates that the lifetime of the target object is independent of the lifetime of the source object (i.e. destruction of the source does not necessary results in the destruction of the target).
          For example *Speed Skating Event* contain a number of *Speed Skating Events*. If one *Speed Skating Event* is cancelled the rest of the *Speed Skating Events* still take place.
6. ***Determine the necessary access control***
    - For example, look at visibility, at the access in a class (public, private, protected, or make a class a friend or not.
7. ***Add navigational paths***
    - For example, associations are bidirectional, but if an association is traversed only in one direction it can be implemented as a pointer in C++

Deliverable #3 (40%) – *Develop and implement the executable releases* that reflect the choices made during the iterative process of analysis and produce the output on the line of the one described in Project#1. Attention, while the content of the output produced by the prototype should be the same, the formatting of the output is irrelevant.

Deliverable #4 (10%) – Complete the template provided in the part 1 of the project and prepare the complete documentation of the system. Submit both electronically as well as printed in your binder.

4 - Project Presentation: (20% of the grade)

Develop any additional material (such as slides, reports, handouts, charts, etc.) required to present your project in class in a professional manner. An additional report text file that indicates how the work has been distributed within the group and the contribution of each participant in the team is also required.

The presentation of your project will be scheduled either on May 1st or May 3rd. Be sure you check the scheduled time for your presentation. Each student of the team must present a part of the project.

ATTENTION

- ***If the project is not presented the entire project (not just this part!) will receive automatically a 0.***
- ***Similarly, if you are not present in the presentation of your classmates' project, your entire project will receive a 0.***

Deliverable #5 (20%) – Prepare any material required for a professional presentation of the project. The 20% of this deliverable will be divided in the following way:

- evaluation in term of the quality of the material produced for the presentation (10%)
- evaluation of the quality of the presentation, i.e. clarity of presentation, competence, professional presentation, etc.(10%).

Submission Instructions –

- Collect all the deliverables in a folder, zip it, and attach it to the project assignment before the deadline.
- Return the 3 ring binder folder to the instructor.

The **grading rubric** of the project has been given in detail in each of the above steps.

NO LATE PROJECT WILL BE ACCEPTED. SO PLAN ACCORDINGLY!

WHATEVER YOU HAVE COMPLETED BY THAT DAY MUST BE TURNED IN AND WILL BE EVALAUTED AS IS.

Deadline: 11:59 pm, April 30, 2018