LAB 3: Design and Implementation

1. OBJECTIVES

1.1 Transform the analysis model into a design model
1.2 Determine proper system architecture
1.3 Address key design issues and use appropriate design patterns
1.4 Start implementing your design model

2. INTRODUCTION

2.1 During software design, the analysis model is transformed into the corresponding design model. In object-oriented methodology, the important classes in the analysis model can be mapped onto the design model.

2.2 The boundary between analysis and design stages in the SDLC is fuzzy. At the tail end of analysis, you were already beginning to think about some of the design issues listed below. You will continue to refine sequence diagrams and the dialog map from Lab #2.

2.3 Make design decisions to incorporate good principles such as encapsulation, loose coupling and high cohesion. The use of appropriate design patterns will improve reusability, extensibility and maintainability.

2.4 Once your design model becomes stable, you can start mapping your design model onto code. That is, start implementing your system.

If you implement your design using Java or other Object-Oriented languages

- each UML class is mapped onto a Java class
- the generalization relationship is mapped to an *extends* keyword
- the interface realization relationship is mapped onto an *implements* keyword
- an x-to-many association is mapped onto a Collection.

2.5 As you start implementation, keep in mind the traceability from requirements to design, then to implementation and test cases.

2.6 In addition to software tools provided by the SWLAB, you can choose any technologies that you deem relevant to the design and implementation of your applications. You need to justify your choice in the project demo and final documentations. Also, remember that software tools not provided from the School come with no School support so be sure that they will work for you before committing to them.

3. PROCEDURE

3.1 *Transform analysis model into design model*

3.1.1 Modify your conceptual model from Lab #2 to make a draft design class model.

3.1.2 Determine appropriate system architecture.

3.1.3 Follow the heuristics from page 341 ~ 345 of Fox:

- Add a start-up class (which performs system initialization)
- Add control classes and distribute the logic and coordination responsibilities amongst them

- Add container classes to hold collection of objects, e.g. List
- Apply design patterns where appropriate

3.1.4 Re-visit your sequence diagrams and dialog map from Lab #2. Add detail e.g. attributes and operations to your class model.

3.2 *Address key design issues*

3.2.1 In your design, address key design issues listed in Chapter 7.4 of Bruegge:
- Identifying and storing persistent data
- Providing access control

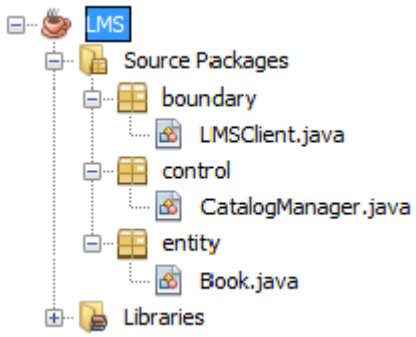
3.2.2 Apply design patterns where appropriate.

3.2.3 Add detail to the class diagram and sequence diagrams of your design model.

3.3 *Implement your design in code*

3.3.1 From the class diagram, generate the skeleton code. Implement the behavior as documented in the Sequence Diagrams and Dialog Map.

3.3.2 You may wish to organize the classes into packages or folders, following the stereotypes in the design class model.



3.3.3 Document the classes and the **key public methods** to convey design intent and usage using Javadoc or similar techniques for other programming languages.

```
/**
 * This class implements the Book entity with
 * the attributes title, author, ISBN
 *
 * @author jane doe
 */
public class Book {
```

3.3.4 To facilitate team collaboration, upload the source code and libraries into the SVN. Your teammates can then check out the code.
Note that SVN is an industry practice for collaborative software development. It is very different from document sharing service such as Dropbox.

Similar techniques include github, bitbucket, etc. You may use other repository. But you must regularly upload your work products to your team's SVN repository.

4. DELIVERABLES

- Complete Use Case model
- Design Model
  - Class diagram
  - Sequence diagrams
  - Dialog map
- System architecture
- application skeleton

Please submit the deliverables to your SVN repository (under the folder "lab3") before the Lab#4 starts. Your Lab Supervisor will need to see and discuss these deliverables with you during Lab #4.

5. REFERENCES

- Introduction to Software Engineering Design: Processes, Principles and Patterns with UML2 – C. Fox
- Object-Oriented Software Engineering: Using UML, Patterns and Java – B. Bruegge and A. Dutoit