

Guidelines for Detailed Design of an Online Bookshop

For COMP2110/2510 Assignment 3, 2009

Department of Computer Science

College of Engineering and Computer Science
The Australian National University

1 Deliverables

The submission should comprise of the following document:

- ass3.pdf (This should include the cover sheet)

We expect this design to be **around 26 pages** (excluding the cover sheet and table of contents). There is an absolute upper limit of 34 pages (excluding the cover sheet and table of contents). Choose your diagrams carefully: text in diagrams must be legible on the printed page.

2 Basic Specification

The assignment is to do a complete design (both high-level and detailed design) for an online bookshop. Your application must satisfy the requirements set out in your assignment 1.

The basic requirements for the assignment are two-fold:

1. the technical design of a solution to the problem (classes, methods, etc.)
2. the description of this solution in a well-structured, well-formatted document.

Both aspects are important. Make sure that you put a good amount of time into the document, and make sure that you put a good amount of time into the solution.

You can choose your document format. It may follow the format of the Aqualush Detailed Design [Fox, 2007] (this is the preferred format). The requirement is that the document should be well-structured and that it communicates to a human reader. If you follow a template or another example you must not include trivial details that swamp the essential message, but you **must not** make it all a narrative blog-style journal of personal discovery. **This is a formal design document.**

But:

- Your design document does not have to contain as many detailed sections as there are in the templates. Tailor accordingly. Many of the sections included in the templates may not be necessary here.
- Your design document must identify any uses of standard design patterns by noting the roles in the pattern that are played by specific classes and methods and relationships in your class diagrams, as shown in the lectures. Using design patterns is encouraged, but not absolutely required.
- As you can see in [Fox, 2007], the user interfaces/screen designs are described in the software requirements section. No specific screen designs have been included in the SRS in your assignment 1. This time, you must therefore state what kinds of screen views are needed, and the information content of each view, and the navigational/control functions associated with each one: showing a design for a graphical layout format is not required,
- but your description should be clear and concise (think of using tables).

3 Goals of this design

What is required is a single document containing a clearly stated description of the software system. This description must be suitable for two distinct classes of readers:

- A knowledgeable and experienced reviewer or project manager. (This reader will be reading your design while asking questions like: Can this design be implemented? How long will it take? How much will it cost? Should I go ahead with this project? Will the finished software meet the requirements?)
- Teams of programmers. (They need to be able to read the individual subsections of the detailed design without reading the rest, and need to be able to find enough information there, and sufficiently precise, that they can implement the individual components and have someone else put them together and have them work, without having to make lots of additional design and interface decisions – these people are not paid to make such decisions, and probably will not be good at making them either. These decisions are the job of the designer, and you must describe the resulting design for these people to implement.)

So you have to see the big picture and describe it (for Type 1 readers) and also work out the details and describe them (for Type 1 and Type 2 readers).

4 High-Level Design

You **must** include a high-level design. It does not have to be the same as the one you submitted for Assignment 2. In most cases, it should be much better than your Assignment 2. This is partly because you have been through a feedback cycle with Assignment 2. It is also because you will be preparing it at the same time as your detailed design, and so the high-level design will partly serve as an introduction to or executive summary of your detailed design. You will know how your components work and how they interact – in detail – and so your high-level design will be built on firm ground, not guesswork.

We will not be comparing the two high-level designs. So you are free to use your old one unchanged (not recommended) or start again from scratch (probably not recommended for most of you) or to take the good parts of your old design as a starting point and replace or revise the bits that were not so good (recommended).

The high-level design will need to include an overview: a module guide with a statement of module responsibilities. If you are using a known architecture (such as Model-View-Controller or Repository-Based) then you should say so and indicate the correspondence between this architecture and your modules, preferably with labelled diagrams. You should also include a description of the interactions between these modules.

Do not waste time on alternative viewpoints that do not add to the reader's understanding. Do not just include diagrams because you know how to: ask yourself whether they are necessary, and whether they make the design easier to understand. Sometimes a state diagram is simply confusing and is better left out. Sometimes it is essential. The high-level design should probably be around 5–7 pages.

5 Detailed Design

For the detailed design you need to give a complete specification of the entire software: every package, class, attribute, procedure, function. It will probably make sense to organise this according to the modules/packages identified in your high-level design.

You must provide complete interfaces for all classes (similar in level of detail to those provided by Javadoc). You must also provide internal details for all classes, including the private attributes and routines, and including pseudo-code for the more complex routines.

You must describe the interactions between the classes in your system. This is probably your largest and most challenging task.

As part of this you must include detailed UML class diagrams showing all classes in your system and what other classes they interact with and how. But do not put this all into one huge diagram for the whole system. Such a diagram is impossible to understand. Instead, include several smaller diagrams (perhaps one for each package). Keep them small enough to understand. These must be proper UML or X UML [Starr, 2002] class diagrams, not vague informal diagrams whose meaning is unclear. You must include all externally visible features of each class. You must include inheritance, aggregation/composition, association and dependency relationships as applicable. If you have used any design patterns, indicate this with an annotation in your class diagram.

Include additional descriptive text as necessary, explaining any special features of your design or anticipating readers' questions about the design (see the Rationale sections in [Fox, 2007]). Be selective and focus on the parts of your design that readers are likely to find difficult to understand. If a particular class or package plays some vital role then perhaps you need to explain its workings in more detail, in particular if it differs in some way from the most obvious way of doing things. Include UML sequence diagrams to clarify complex or difficult interactions. Include object diagrams in situations where showing how multiple instances are connected can show a clearer picture than a class diagram.

6 Components Required

- Your name and student ID. Title of the Document. Date.
- Overview/Introduction
- High-level design/architecture: description and correspondence with the modules, interactions among high-level modules
- Module guide: division of responsibilities among high-level modules/packages
- Class diagrams for each module, class interactions, and sequence diagrams for selected parts
- Complete interfaces for all classes (you do not need pre- and post-conditions for all methods)
- Internal details for all classes
- Possibly some object diagrams, where useful to explain the workings of some specific design aspect

You should not include the Software Requirements Specification, but you should include a proper reference to it (author, title, URL, date accessed etc.). If you have modified the SRS you should include the revised version including explanatory comments for your changes as an Appendix. It will not be counted in the page count.

7 Marking and Assessment

The assignment will be assessed on three criteria:

- the amount of the functional requirements covered by the design
- the quality of the design itself (appropriate use of design patterns, elegance and levels of abstraction, re-usability, generality, quality of modularity, etc.)
- the quality of your description of your design.

No marks will be given for adding features to the specification given; in fact, marks may be subtracted. You are required to meet the requirements, not to build what you think would be better (the client just will not pay). This is important: the requirements specification is there as a specification of what is needed, not as a starting point. If you need to add requirements for completeness or to change requirements because they are inconsistent or ambiguous then you must do so with minimal changes (you must not rewrite the lot just because you have a preferred style), and all your design must be traced to original requirements numbers and separately numbered additions, if any.

Pass Level

Minimum required for a Pass: A detailed design for the functions of starting a book shopping client session, handling the shopping cart and payment, managing shop database, and creating simple actions.

At Pass level you can omit the design of:

1. electronic book sharing
2. searching for books
3. using third party services (e.g. services from a credit card company)

Some inconsistencies and missing functions in the design are acceptable at pass level.

Credit Level – and above

To make the assignment eligible for a Credit or better you must include the design for electronic book sharing, book searching, and using third party services. The design description and quality needs to be at this level as well.

Distinction and High Distinction level

A good design: that is, the right abstractions expressed in classes and functions, and a good description: clear communication, in good order, at the right level for the reader (see earlier discussion), is needed to get into the Distinction and High Distinction categories. The Distinction level also requires you to include more consideration of performance needs (speed and space) in your design. This will not be visible to the reader (marker) unless you discuss it in design rationale sections.

Covering more of the functional requirements does not guarantee a higher mark. Inconsistencies and missing functions will cost marks, and so will poor communication: too much irrelevant detail, missing abstractions, etc.

References

- Braude, E. J., *Software Design: From Programming to Architecture*, Wiley, 2004.
- Chemboli, S., A. Khorev, S. Flint, and L. Nguyen-Hoan, *Requirements for a commodity messaging client application for comp2110/2510 assignment 2 and 3, 2008*, Tech. rep., The Australian National University, 2008.
- Fox, C. J., *Introduction to Software Engineering Design*, chap. Aqualush Detailed Design, pp. 648–687, Addison-Wesley, 2007.
- Starr, L., *Executable UML: How to Build Class Models*, Prentice Hall, 2002.