Software Design and Architectures
SE-2 / SE426 / CS446 / ECE426
Fall 2003

Assignment 1 : Design Processes

This assignment is due in class (1pm RCH 205) on Monday September 15. Each student should submit his own work, not that of a group.

Identify the last Software Design course you took, or the last course with a significant software development component. Identify the last or a typical programming assignment in that course. Recall the history of how you developed a solution to that assignment. If you worked in groups for the assignment, the other group members might help you remember: but do your own analysis.

a) Identify the steps of development you followed. Give a few sentences of description of the task(s) being performed in each step.
b) Identify the external specifications, work products, and deliverables you produced. Give a few sentences description of each.
c) Describe the development process you followed, as a states-and-transition diagram, showing the start state, work products of each transition, and the final state.
d) In the same manner as above for (c), describe the design process you followed.

Sample Solution

This solution is written with respect to an assignment in CS246 as last taught by me (Malton) and for the process I imagine I would follow. The assignment required implementing a reader of mbox files in two different ways.

A good answer will be simple (because the process is simple) and mention most of the steps and deliverables below. There is no need to distinguish deliverables from work products.

a) Development steps.
There shouldn't be any 'understand' steps or 'think' steps, unless they terminate with concrete work products. There isn't any work product called 'knowledge'!

S1. **Obtain test data**
Some test data were provided as part of the assignment, other must be found and hand-simulated from mboxes in one's own directory.

S2. **Design classes**
Classes were designed by fixing their names, attributes, and method signatures. May involve informal diagramming and coding to discover what works.

S3. **Write code**
Translating the method signatures and intent into implementations. Since by no means everything was in the class design, the given requirements are still input to this task.

S4. **Write makefiles**
It was required to build all the final executables with *make* procedures, designed, coded, and tested in this step, since now we know which classes are going to exist.

S5. **Test**
This step is applying each of the tests and saving results.

S6. **Debug**
The usual task of finding errors in the design or coding, and repairing them and producing revised code.

S7. **Finalize**
Writing the final writeup from all the available deliverables.

S8. **Submit**
Final deliverables were submitted using the electronic *submit* procedure.

b) Deliverables and documents.

D1. **Assignment description**
Provided by the instructor.

D2. **Test data**
Each test was an mbox editing command in a file, plus a pair of mbox files being respectively the input and output for the test.

D3. **Class stubs**
The stub design was recorded as .h files, one per class. According to the requirements of the course, attributes were private and methods were public; there was no use of inheritance. Algorithms were suggested by comments on methods. May be revised.

D4. **C++ source**
One .cc file to match each .h file, and the versions of main procedure. May be revised.

D5. **Make-files**
One make file containing build procedures for all classes and for the final executables.

D6. **Test results**
Concretely a *diff -s* listing comparing the result of a test to the expected result. What we want is *Files results.txt and expected.txt are identical*.

D7. **Writeup**
An informal two-page description of the design and construction of the programs.

D8. **Testing summary**
After testing is complete and successful this summarizes the results of the last tests.

c) Process. I've described this process as a transition table, with the completion of a work product as the transition event. Work should start at S1 and S2. Note that there is no mechanism for revising the design; in keeping with the usual methods of doing assignments, I've supposed that design revision is handled by code revision. This inevitably causes the design to become obsolete!

Start S1 on receiption of D1.
Start S2 on reception of D1 and given D2s.
Move from S2 to S3 on completion of D3.
Move from S2 to S4 on completion of D3.
Move from S3 to S5 on completion of D4 and D5.
Move from S4 to S5 on completion of D4 and D5.
Move from S5 to S6 on completion of D6.
Move from S6 to S5 on revision of D4.
Move from S5 to S7 on completion of D4 and D8.
Move from S7 to S8 on completion of D7.
Finish with S8.

c) Design. Practically the design revision process involved two work products, namely the class stub and the class diagram. These were produced a few at a time, discussed, compiled for syntax checking, and revised, until complete. The 'discussing' state produced either new stubs or revisions of old ones, continuously, until the whole design (at set of stubs) was considered deliverable as D3 above.