6.033 2012 Design Project 2

old / last year

I. Due Dates and Deliverables

You will be working on the second design project in teams of three students who share the same recitation instructor. There are four deliverables for this design project:

1. A list of team members emailed to your TA by April 12, 2012.
2. One copy of a design proposal (not exceeding 1,200 words), due at 5pm on April 26, 2012.
3. One copy of a design report (not exceeding 5,000 words), due at 5pm on May 10, 2012.
4. A five-minute in-recitation presentation, on May 15, 2012. In consultation with the chair of the faculty we have determined that the assignment follows the spirit of the end-of-term rules.

All deliverables should be submitted via the online submission site.

As with real life system designs, 6.033 design projects are under-specified, and it is your job to complete the specification in a sensible way given the overall requirements of the project. As with designs in practice, the specifications often need some adjustment as the design is fleshed out. We recommend that you start early so that you can evolve your design over time. A good design is likely to take more than just a few days to put together.

II. The Problem

Ben Bitdiddle is collaborating with several of his friends on DP2. He wants to build a text editor in which he and his group members can edit the DP2 report at the same time. Ben and his friends often use laptops without internet access, and they don't like relying on Athena, so they would like to come up with a design that allows disconnected operation and does not require a central server.

Ben is worried that he or his friends might make a mistake when editing their report, and not notice that mistake until much later. Ben wants his text editor to allow undoing changes from the past. For example, if Ben accidentally said his design requires 10MB of memory but in reality it requires 100MB, he should be able to go back, find this change, and revert just that change, without reverting all of the other changes he made since then. Furtheromre, if one of Ben's friends copied that text to another place in the document, that change should be identified (automatically) and fixed (perhaps with Ben's approval).

At some point before the deadline, Ben expects to finish his DP2 design. He wants to make sure that the design he submits reflects everyone's opinion. To help all group members to agree on a single version of the document to submit, the text editor should allow Ben to commit a certain version of the document, for example, the one he wants to submit to the 6.033 staff.

Your job is to design a collaborative text editor that meets Ben's requirements.

III. Requirements

The challenges you should address in your design project are as follows:

1. Your design must support disconnected operation. If two group members edit the same document,

but make changes to different parts of the document, their changes should eventually be merged together when they re-connect to the Internet. If they edit the same part of the document, the text editor should flag the conflicting parts of their changes, and ask the users to resolve this conflict. A third member of the group should not have to resolve this conflict once it has already been resolved.

2. Your design must support selective undo. If a user wants to undo any operation from the past, your design must revert the effects of that operation while preserving subsequent changes from that user, to the extent possible. If some subsequent operation depends on the change being undone, the effects of that operation should be updated as if the undone operation never happened. Of course, this may not work in all cases, it's fine to ask the user for input in resolving conflicts when the system cannot do it automatically (as above).
3. Your design must support commit points. A user should be able to initiate commit on a document with a given name, such as ``Ben's final submission". Once the user's commit returns, all users must agree on the document that corresponds to that commit name, even if all machines crash after that point. Additionally, the committed document must reflect the changes from each user at the time the commit was initiated.
4. Your design can assume that the membership of Ben's group does not change for the lifetime of the document.

After you have designed your system, you should evaluate how usable your system is, in terms of how many conflicts have to be resolved when an old change is undone, or when two users make concurrent changes to the document. Your design should not ask users to resolve conflicts that don't matter in the current version of the document. Your design should also not ask users to resolve conflicts that can be reasonably resolved automatically, by, for example, keeping more precise dependencies.

For the purpose of this analysis, you can assume that Ben has 3 students in his group. The document they are editing is a DP2 report. Group members edit the document heavily, writing about 10 times as much text as they eventually submit.

One specific scenario that must not require resolving conflicts is as follows. Ben changes a sentence in the beginning of the document, as one of the first few edits, and that sentence is not modified by anyone since then. That sentence is, however, copied to the conclusion paragraph in the document. When another student undoes Ben's sentence change, no conflicts should arise, and the sentence in the conclusion should be adjusted accordingly (perhaps notifying the user in the process to make sure they're not surprised by the propagation of effects).

Your design must handle a situation where two users add lots of text to the document in different paragraphs, and also make different changes to a single sentence in the introduction. Once these users re-connect to the network, your design must not require resolving conflicts for the changes to different paragraphs.

Your design report must discuss how you handle failures during commit, where Ben's system crashes (when Ben initiated commit) at any point in the commit process, what happens if another student in the group has outstanding edits that Ben has not seen yet, and what happens if another student's computer crashes while Ben is in the middle of committing. Under any failures (including the ones mentioned here), once a user sees that some version was committed under a given name, no other version can ever be committed under that name. If a user does not see that a version was committed, they must be able to either commit a new version *or* find out the version that *was* committed under that name.

Your design must correctly handle concurrent commits, when multiple users try to commit a version with

the same name.

Your design must never silently drop changes, except if the computer of the user making the change crashes just after the user made that change, *and* that change has not been sent out to other users yet.

Optional challenge problem:

- Handle dynamic group membership, where any group member can leave the group at any time, and a group member can add a new member at any time as well.

IV. Design proposal

The design proposal should summarize your design in 1,200 words or fewer. It should outline the protocol and algorithms the nodes use to publish messages, to subscribe to topics, and to route messages from publishers to subscribers.

You do not have to present a detailed rationale or analysis in your proposal. However, if any of your design decisions are unusual (particularly creative, experimental, or risky) or if you deviate from the requirements, you should explain and justify those decisions in your proposal.

You will receive feedback from your TA in time to adjust your final report.

V. Design report

Your report should explain your design. It should discuss the major design decisions and tradeoffs you made, and justify your choices. It should discuss any limitations of which you are aware. You should assume that your report is being read by someone who has read this assignment, but has not thought carefully about this particular design problem. Give enough detail that your project can be turned over successfully to an implementation team. Your report should convince the reader that your design satisfies the requirements in Section III.

Use this organization for your report:

- Title page: Give your report a title that reflects the subject and scope of your project. Include your names, email address, recitation instructor, section time(s), and the date on the title page.
- No table of contents.
- Introduction: Summarize what your design is intended to achieve, outline the design, explain the major trade-offs and design decisions you have made, and justify those trade-offs and decisions.
- Design: Explain your design. Identify your design's main components, state, algorithms, and protocols. You should sub-divide the design, with corresponding subsections in the text, so that the reader can focus on and understand one piece at a time. Explain why your design makes sense as well as explaining how it works. Use diagrams, pseudo-code, and worked examples as appropriate.
- Analysis: Explain how you expect your design to behave in different scenarios. What scenarios might pose problems for performance or correctness? What do you expect to be the scalability limits of your design?
- Conclusion: Briefly summarize your design and provide recommendations for further actions and a list of any problems that must be resolved before the design can be implemented.
- Acknowledgments and references: Give credit to individuals whom you consulted in developing your design. Provide a list of references at the end using the IEEE citation-sequence system

("IEEE style") described in the Mayfield Handbook.
- Word count. Please indicate the word count of your report at the end of the document.

Here are a few tips:

- Use ideas and terms from the course notes when appropriate; this will save you space (you can refer the reader to the relevant section of the notes) and will save the reader some effort.
- Before you explain the solution to any given problem, say what the problem is.
- Before presenting the details of any given design component, ensure that the purpose and requirements of that component are well described.
- It's often valuable to illustrate an idea using an example, but an example is no substitute for a full explanation of the idea.
- You may want to separate the explanation of a component's data structures (or packet formats) from its algorithms.
- Explain all figures, tables, and pseudo-code; explain what is being presented, and what conclusions the reader should draw.

While the Writing Program will not be grading DP2, you should feel free to ask them for help.

VI. Presentation

You will have only about five minutes for your presentation. The audience will be very familiar with the problem, so you can get right to the guts of your solution.

VII. How we evaluate your work

Your recitation instructor will assign your report a grade that reflects both the design itself and how well your report presents the design. These are the main high-level grading criteria:

- **Clarity.** Is the design described well enough to be understood, evaluated, and implemented?
- **Correctness.** Does the design achieve the requirements laid out in Section III? Does your report give a convincing analysis that your design works under the described scenario?
- **Simplicity.** Is the level of complexity in the design justified?

VII. Clarifications

- None yet.