Information and Computer Science 52
Introduction to Software Engineering
Fall Quarter 2004

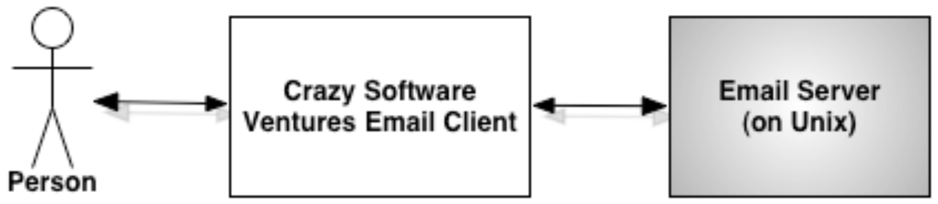
Name	______________________________
Student ID	______________________________
Signature	______________________________
SEAT #	______________________________

Instructions

1. This exam is closed book, closed lecture notes, and closed personal notes. No PDAs, cell phones, SMS devices, … You may use an English/non-English language dictionary.
2. This exam consists of 7 questions on 5 pages; answer all questions on all pages.
3. The questions are **not** equally weighted. The relative point value of each question is given at the beginning of the question. The total numbers of points that can be earned on this exam is 90.
4. Write your name, student ID, seat number, and sign the exam where indicated above. In addition, write your name on **each** page of the exam. Failure to do so results in a loss of points.
5. Before you cheat, consider the ramifications: you will score an F for **all** of ICS 52, not just the midterm.
6. Before you cheat, consider the chances: multiple eyes will be watching you from multiple locations in the room.
7. All answers are to be legible. Write clearly.
8. All answers are to be given on these pages. Use the backsides of the pages **if necessary**, with a reference as to where any extra portion of an answer is located.
9. You have fifty minutes for this exam – that is, until 10:50 a.m. Plan your time accordingly.

Questions 1 to 4 are with respect to the following scenario. Your company, Crazy Software Ventures (CSV), has heard a lot about the many problems that users have as a result of using Microsoft Outlook Express as their email client. They have also noted that the Eudora mail application is looking a little dated. As a result they have decided to create and sell a new email client, code named GPO, to compete in the marketplace against Outlook, Entourage, Eudora, Mailsmith, Pine, and so on.

The basic architecture is shown below:



You have been put in charge of the project to create this new application (GPO is the white box shown above). Even though you don't know a lot about how email systems work (in fact, you're not even sure what the difference is between a POP server and an IMAP server, but you do know that, for GPO, once a user has received his email, the mail is stored on the same machine as the GPO client — it is *not* left on the server) but your management assures you that those issues can wait for a while. The critical need from management's perspective is that the user will like the appearance and functionality of the email client. And of course that the first version of the product be delivered very soon (no more than 6 months), and for only a small investment of funds on their part (enough to fund 5 relatively inexperienced developers full time for 6 months). The initial product must run on Windows XP, but management is also eyeing the Macintosh market, and your boss just bought a new, hand-held Blackberry, which he thinks is very cool…..

1. (10 points) What software development process ("lifecycle model") will you adopt, and why?

Any development process that incorporates facets of rapid prototyping: rapid prototyping, incremental, and spiral is a strong choice. (Incremental is probably the best choice given the constraints.) Waterfall is not likely to be a good process due to the short development time (although several did successfully argue its benefits). Sync-and-stabilize could be justified as providing benefits in the long run, but you need to understand additional costs incurred. Build-and-fix could be a stretch, but could be successfully argued because the developers are familiar with the product already, yet the developers are likely not experienced enough to pull this out without planning.

2. (15 points) What will you emphasize in the requirements phase(s), and why?

The key points to emphasize are core functionality and appearance since those are the declared critical needs of the project. It was stated that you don't know a lot about the difference is between POP and IMAP - so the specification of those sections need a lot of attention. (Other emphasis points for functionality could be: stores message locally, composes messages, works on XP, etc, etc.) The other need is to determine the requirements for GPO's appearance. There are a lot of competing projects (Outlook, Eudora, Pine, etc.) to look at: determine what works well and doesn't. Can you identify points of excellence where GPO can beat (err, learn from) the competition? Could certainly focus on features competitors do poorly.

Do NOT focus on miscellaneous features. They are unimportant.

It wasn't enough to just say functionality and UI; you had to somehow indicate or give examples that showed that you were talking about the basic functionality for the system. And, if you talked about emphasizing features beyond what other email clients had, that would also not count that as core functionality.

3. (10 points) How will you validate your requirements?

The most important key here are various forms of 'testing', 'feedback', and 'analysis'. This is why the chosen software development process should strongly incorporate user feedback. Try to have prospective users go over mockups with an eye towards the 'right' software being built. Additionally, a detailed 'acceptance test plan' could be drawn up to ensure that the requirements are met once the implementation is completed. Comparing to other clients directly was NOT acceptable.

Some examples of items in the acceptance test plan are desired. Suggestions: testing on Windows XP, works with IMAP/POP servers, stores mail locally without loss, etc.

4. (15 points) Given your tight schedule and limited resources, it is clear that you won't be able to put everything in your initial product release that you would like to. What are the consequences of this realization for your solution architecture? Provide an illustration (just text is fine, no need to draw a picture) of something that won't be in your initial release but that you will include later, and how this would be supported by your architecture.

The critical insight is that the solution architecture must be extensible so that we can go back to it later and add the missing functionality. There are some functionality that is mandatory (i.e. being able to write emails, read emails, etc.), but there are some that aren't mandatory.

An example of a missing feature could be supporting IMAP but not POP initially. A layered architecture that abstracts out the communication aspects would be essential to adding this later. Another reasonable example is supporting only one platform initially and then adding multiple platforms later.

Note that we were asking about 'architecture' not lifecycles: 'sync and stabilize' (or similar) is NOT correct. Also, the architecture of the overall system is client/server, but that's not a suitable architecture for the email client itself.

5. (24 points – that's 2 points per box) Fill in the following comparison chart.

Lifecycle Model (Process)	Advantages	Disadvantages
Build-and-fix	Cheap; fast	Does not scale to large projects
Waterfall	Document-driven, clear lifecycle phases	May not address user's needs; can not proceed to one phase until down with current phase
Rapid Prototyping	Allows user feedback early; good for when requirements are not known precisely	Throw one away; may not focus on detailed features
Incremental	Allows early releases to increase revenue stream;	Requires an open architecture; may degrade into build and fix.
Synch-and-Stabilize	Allows parallel release tracks; incorporate new features quickly;	Only been used by Microsoft; May require more manpower; May need a monopoly to pull off
Spiral	Incorporates most advantages; explicitly focuses on risk	Need a risk expert; may take longer to get to less 'risky' tasks

6. (10 points) eBay does not use a peer-to-peer architectural style for their auction application. For the purposes of this question assume that their architecture is a simple client-server (it's actually quite a bit more complicated than that.)

(a) What advantages are there, from eBay's perspective, to the client-server approach? You should briefly (i.e. one sentence) describe 3 such advantages.

Some examples: scalable, allows easy entrance to the system, matches with WWW architecture, well known client infrastructure (browser), centralized answer, allows for business model, etc.

(b) What advantages would there be to switching over to a peer-to-peer (p2p) style? Would those advantages be primarily for the users? For eBay?

Some advantages with P2P: anarchic scalability above client/server; more reliable; etc.

Users: Pro: no middleman costs perhaps; Con: how to deal with fraud?
eBay: Pro: Less resources need to be dedicated; Con: Lack of direct control and brand.

7. (6 points) Complete the following definition of the uses relationship: "Module A uses module B ....

Definition: Mi uses Mj if and only if correct execution of Mj is necessary for Mi to complete the task described in its specification.

(...if and only if correct execution of Module B is necessary for Module A to complete the task describe in its (Module A's) specification.)

(Design Slide #33)