CSE 403, Winter 2007
Homework #2: Software Requirements Specification ("SRS"); 50 points

Due: **Sunday, January 28, 2007, 11:59 PM electronically,**
AND/OR **Monday, January 29, 2007, at start of lecture**

Assignment Description:

Your team has been funded to produce the software project outlined in your previous project proposal. The "customer" hiring you to write the product is the wealthy venture capital firm, SteppCo. The CEO of SteppCo intends to compensate your team for its services by awarding grade points. SteppCo has three upper-level managers (Marty Stepp, Saleema Amershi, and Brian Harris), one of whom will meet regularly with your group to discuss its progress.

In this assignment, your group will specify some of the requirements for your software project, as well as providing rough sketches of a partial user interface (UI) prototype. This document is one of several submissions and fits into the larger scope of your project as follows (this is a rough outline that may change):

- Software Requirements Specification (SRS)
- Software Design Specification (SDS)
- Initial prototype implementation
- Second, more featureful implementation
- Test plan, test cases, and quality-tested implementation
- Retrospective documentation, product documentation, and maintenance information

External Requirements:

The customers do not know exactly what they want, but they do have the following requests:

- The product should display a company logo prominently on the UI.
- Your product should have a clear means of generating (fictional) "revenue."
  A possible way to do this would be an ad-based approach, by providing space on your UI for an advertisement image.
- The product should be as usable as possible, even for people who are not expert computer users (with the exception of projects that are designed specifically for experts, such as development tools).
- The product must be robust against errors that can reasonably be expected to occur, such as invalid user input, lost network connections, etc.
- As before, the product must involve communication between two or more computers. In other words, it should be network enabled, or connect to a remote database back-end, or be a client-server application, etc. If your original proposal did not carefully take this into account, please do so now.
- Your project must be usable and/or installable for a person on a standard computer. If it is web-based, it must have a public URL that others can use to access it. If it is a stand-alone application, you will be expected to provide a reasonable means for others to install and run it. It is not reasonable to expect the user to download and compile your source code to run your application. You can expect that the user will have installed any necessary libraries and tools (such as a Java Virtual Machine, or a .NET framework runtime), but after that, the user should be able to download and run your system easily.

Beyond these requirements, you are largely free to make decisions of your own. You should, however, talk to your customers as you plan this project in order to make sure your product meets their needs. For full credit, you should discuss your proposed requirements in some way with your customers before submitting them.

Software Requirements Specification (SRS)

The first milestone artifact you will submit for this project is a set of documents related to your project requirements and high-level UI design. This project description can be considered a partial requirement specification, but much information is intentionally left out. This is to encourage your group to come up with questions to ask "the customer." You may ask these questions in lecture, by email, or on the course message board. A submitted SRS that does not reflect questions and answers with the "customer" will not receive full credit.

Specifically, your SRS must contain four (4) kinds of documents as outlined on the following pages.

1. **Requirements Outline**
   Submit a requirements outline document, containing brief descriptions of each of the following areas. Many of these can be heavily influenced by the content of your original project proposal, but they should be discussed with all group members to ensure agreement. This document should be greater than two (2) pages in length but no more than five (5) pages in length. Address at least the following categories of information:

   a. **Software Toolset**
      What programming languages, tools, and/or data sources (roughly) do you intend to use? This can be identical to what was submitted on your project proposal, if you still intend to use the same tools. You should specifically address what major programming languages you expect to use, as well as how you plan to share documents and files between group members (perhaps in a version control system such as CVS).

      If some of this information is unknown to you, describe potential choices and a specific plan for who will investigate them, when, and how, to determine which to choose.

   b. **Group Dynamics**
      Now that you know who your project team members are, what do you believe will be their approximate roles? Will everyone share in the development, or will you have designated project managers, testers, etc.? Why have you chosen these roles for these people? If a conflict or disagreement arises (such as two members with conflicting opinions about how to design a certain feature), how will these be resolved? Be specific in all cases.

   c. **Documentation**
      What external documentation will you provide that will enable users to understand and use your product? (Comments in source code do not constitute external documentation, even if they generate API web pages such as Javadoc. Those are developer documents, not user documents.) This could take the form of help files, a written manual or reference card, integrated help text throughout various UI screens and pages, etc.

   d. **Deployment**
      Roughly how will the user access and run your product? If it is web-based, presumably the user will type your product's URL to load the main page... but is there anything else the user must do? If it is a client application, how will the user download, install, and run your product?

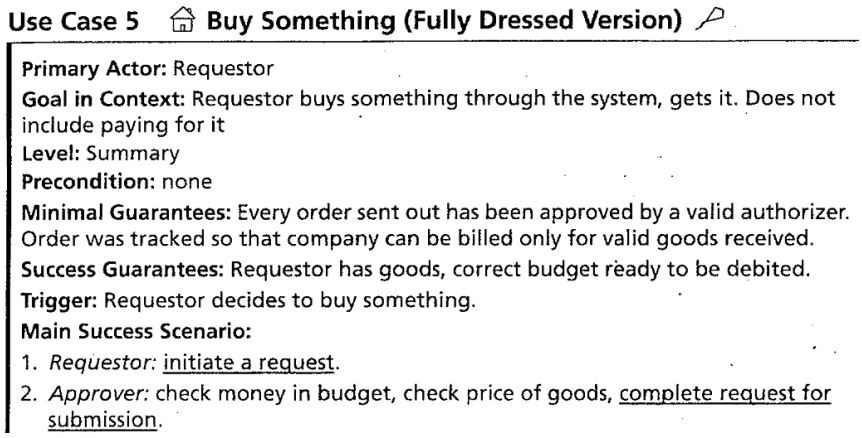
   e. **Risk Summary**
      Describe at least three (3) specific adjustments you are willing and able to make, if the project begins to fall behind schedule. No more than two (2) of the adjustments you list can be feature cuts; at least one must be some other change or cutback, such as changing specific areas of testing, adjusting your group dynamics or time schedule, etc.

2. **Use cases**
   Submit the following use case documents:

   a. **One (1) use case summary diagram** (the one with the stick-men, lines, and circles) that depicts several of the major use cases and actors that are important in your system. It should include at least the three use cases described below, as well as a few other relevant use cases. The use cases are described simply as bubbles, without actual detailed information inside. See the lecture slides about requirements / use cases for examples.

   b. **Two (2) formal use cases** for scenarios you think are two of the most important to your product. They should be similar to Use Cases 1, 2, 3, and 5 from Cockburn's paper, and should include: primary actor, level, preconditions, minimal/success guarantees, a list of steps to the success scenario, a list of properly numbered extensions, and a failure-handling remedy for each extension as appropriate.

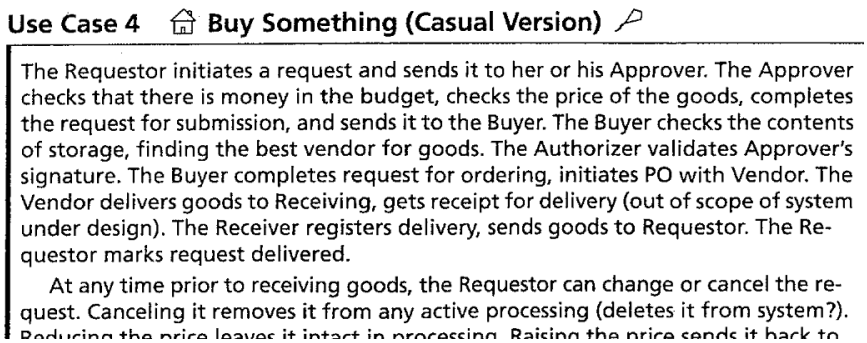
      It is impossible to think of every possible failure case and how to solve it ahead of time. Therefore it is understood that you may miss a few extension cases, and that you may not have a completely detailed remedy for each extension. However, your list of extensions should reflect extensive thought about the subject, and you should have failure remedies if reasonable ones exist. If you do not have a known remedy for an extension, your use case should state this clearly and explain why this is the case and what will be done to investigate possible remedies.



Use Case 5 **Buy Something (Fully Dressed Version)**

Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it
Level: Summary
Precondition: none
Minimal Guarantees: Every order sent out has been approved by a valid authorizer. Order was tracked so that company can be billed only for valid goods received.
Success Guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main Success Scenario:
1. *Requestor:* initiate a request.
2. *Approver:* check money in budget, check price of goods, complete request for submission.

   c. **One (1) casual use case** in paragraph form for one other scenario that you think is important to your product, perhaps of lower importance than the two you chose to depict formally. This use case should be similar to Use Case 4 from Cockburn's paper, describing the main success scenario first in paragraph form, then listing each extension and its remedy (if known) in a second paragraph.



Use Case 4 **Buy Something (Casual Version)**

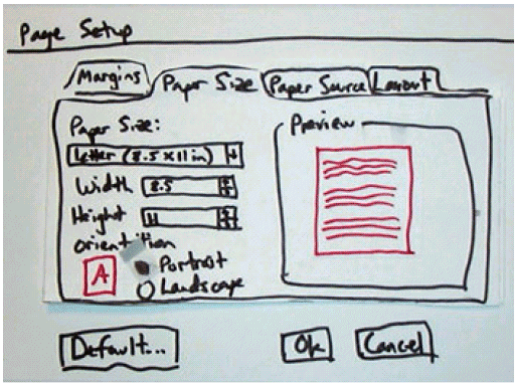
The Requestor initiates a request and sends it to her or his Approver. The Approver checks that there is money in the budget, checks the price of the goods, completes the request for submission, and sends it to the Buyer. The Buyer checks the contents of storage, finding the best vendor for goods. The Authorizer validates Approver's signature. The Buyer completes request for ordering, initiates PO with Vendor. The Vendor delivers goods to Receiving, gets receipt for delivery (out of scope of system under design). The Receiver registers delivery, sends goods to Requestor. The Requestor marks request delivered.

At any time prior to receiving goods, the Requestor can change or cancel the request. Canceling it removes it from any active processing (deletes it from system?). Reducing the price leaves it intact in processing. Raising the price sends it back to

3. **UI prototype**
   Submit diagrams containing rough sketches of your product's user interface. These diagrams should depict the major UI used to complete the use cases you are submitting. For example, if one of your use cases is to Purchase Stocks, you should draw the initial UI that is presented when the user wishes to purchase a stock, along with any other potential major windows, message boxes, etc. that appear as the user navigates through this use case.

   You should submit no less than **two (2) UI diagrams**. The diagrams can be hand-drawn, drawn by computer, or can come from screenshots of an actual programmed prototype if you like. If a window leads to a dialog box, drop-down box, etc., perhaps this should be included as a sub-diagram. If you draw your diagrams by hand, you may later wish to scan them or otherwise digitize them so that they can be used in your group's presentation as described below.

   Your diagrams do not need to be pretty to get full credit, but they should be legible and reflect some forethought about what options will need to be shown and how the user will use the software.



4. **Presentation**
   Submit a set of 5-10 slides that represent a brief presentation of the above material to the rest of the class. Each group will be given approximately **9-10** minutes to present. At least two group members must participate in some way in the presentation, but not every group member needs to be involved in it.

   Your presentation slides should summarize important parts of the preceding elements for your product. Specifically, talk about your group member roles, at least one of your use cases, and at least one UI prototype screenshot. Include a title slide that states your project's name and authors. You should have at least two figures or diagrams in your slides to receive full credit; this can be a diagram taken directly from your other SRS documents described previously.

Submission and Grading:

If you choose to turn in your SRS documents 1-3 electronically, please submit them in Word (.doc) or PDF format; otherwise turn them in as printed pages in lecture. Submit your presentation slides online in PowerPoint (.ppt) or PDF format. Each document's file name should begin with your project's name and reflect what it contains. For example, if your project is called "SuperAwesome", acceptable file names might be `SuperAwesome_use_cases.pdf` , `SuperAwesome_items_1-3.pdf` or `SuperAwesome_presentation.ppt`. You may receive a deduction if you turn in clumsily named or organized files.

Make sure that your project's name and all group members' names appear clearly atop each document. Only one copy of the documents should be submitted for each group.