In Class:
Requirements and Library Interviews

Bring To Class:

- Bring your favorite prepared title hand-written on a piece of paper so that they will be legible from 5 feet away. Use a dark, thick pen. Write large. Get a big piece of paper. Put your name on the paper.

Hand In:

- Personal Background
  - A list of 3-5 project titles that you are considering. For each title, list the potential project user(s) that you would interact with in developing and testing the project. For each idea, evaluate it according to the criteria in the project handout and indicate pros and cons of the project ideas. At least two of your choices must be from the list of outside proposed projects. Any troubles with topics, come talk to a TA and brainstorm.

Read:

- Read the Syllabus and Project Handout. We’ll go over requirements on Friday, so pay attention to that section. Note any questions for class.
- Review the evaluation form that will be used for requirements documents to get an additional sense of the criteria for a good project.
- Read the Library handout and prepare questions to ask during class so that we can begin to formulate a requirements document in class.
- Read “The Phases of Software Engineering” through the end of the “Requirements” section in Reiss, pp. 404-406, plus the outline on 407.
- Read Reiss, pp. 425-427.
- Pursue some example requirements documents from past years. These examples are not perfect, they are intended to give you a feel for what was proposed in previous years. GeoEvents and Biaquest are both projects that were selected for implementation for their respective years.

Mon 1/29

In Class:
In Class:
Requirements and Software Engineering

Bring To Class:
- Bring your favorite proposed titles handwritten on a piece of paper so that they will be legible from 5 feet away. Use a dark, thick pen. Write large. Get a big piece of paper. Put your name on the paper.

Hand In:
- A list of three discussion points, questions, or topics for class based on the reading. What are some of the themes that appear across more than one source? What do you agree with? What do you disagree with?
- What topics intuitive and seem counterintuitive? What surprised you?
- Hand in the name of the project idea that you will proceed with, the names of at least two clients that you will interview, and the times that you have set up for those interviews (which ideally have already passed).

Remember that you must have these interviews before you can write your requirements document.

Read:
- Read Preface to the 20th Anniversary Edition, Chapter 1, and Chapter 2 of The Mythical Man Month.
- Read chapter 9 of Lattics.
- Read tables of contents and introduction to Writing Solid Code.
- Read tables of contents and introduction to Debugging the Development Process.
- Read beginning of "Software Engineering" in Reiss, pp. 397-401.
- Read "The Software Development Process in Reiss, pp. 416-422.
- skim "Techniques for Requirements Elaboration" to get a sense of the complexity of understanding how to ask for requirements and what the answers might mean. This should help in your interviewing.

Hands Up:
- Work on Requirements document. Check out everyone’s project titles handling ideas, if you wish.
Preface to the 20th Anniversary Edition

in the literature than has The Mythical Man-Month. Chapter 17, therefore, comments on some of the published critiques and updates the opinions set forth in 1966.

In preparing my retrospective and update of The Mythical Man-Month, I was struck by how few of the propositions asserted in it have been criticized, proven, or disproven by ongoing software engineering research and experience. It proved useful to me now to catalog these propositions in raw form, stripped of supporting arguments and data. I hope that these bold statements will invite arguments and facts to prove, disprove, update, or refine those propositions. I have included this outline as Chapter 18.

Chapter 19 is the updating essay itself. The reader should be warned that the new opinions are not nearly so well informed by experience in the trenches as the original book was. I have been at work in a university, not industry, and on small-scale projects, not large ones. Since 1966, I have only taught software engineering, not done research in it at all. My research has rather been on virtual environments and their applications.

In preparing this retrospective, I have sought the current views of friends who are indeed at work in software engineering. For a wonderful willingness to share views, to comment thoughtfully on drafts, and to resuscitate me, I am indebted to Barry Boehm, Ken Bache, Dick Case, James Coggin, Tom DeMarco, Jim McCrackin, David Purcell, Earl Wheeler, and Edward Yourdon. Fay Ward has superbly handled the technical production of the new chapters.

I thank Gordon Bell, Bruce Buchanan, Rick Hayes-Roth, my colleagues on the Defense Science Board Task Force on Military Software, and, most especially, David Ferrus for their insights and stimulating ideas for, and Betleach Beley for technical production of, the paper printed here as Chapter 16. Analyzing the software problem into the categories of essence and accident was inspired by Nancy Greenwood Brooks, who used such analysis in a paper on Suzuki visits pedagogy.