CS 458 - Exam 1 Review Suggestions - Fall 2016

last modified: 2016-10-12

- You are responsible for material covered in assigned reading, class sessions, and homeworks; but, here's a quick overview of especially important material.
- You are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have **handwritten** whatever you wish on one or both sides. This paper must include your name, it must be handwritten by you, and it will **not** be returned.
  - Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer.
- This will be a pencil-and-paper exam, primarily involving concepts.

"Mythical Man-Month" - Chapter 16 - "No Silver Bullet"

- What is Brooks' major contention in this paper? What are his major arguments in support of this contention?
- What are inherent properties of modern software systems that make developing them difficult?
- What is the difference between "essential" and "accidental" tasks, in this paper's terms? What are examples of "essential" tasks? What are examples of "accidental" tasks? How does this relate to this paper's major contention?
- What are some of the "past breakthroughs" that helped whittle down the percentage of the software building task that is accidental tasks?
- What are some of the "hopes for the silver" that are advanced as possible "silver bullets", but are likely incremental improvements at best? Why are these not "silver bullets"?
- Even though they are *not* "silver bullets", what *are* some promising attacks on the conceptual essence, with a potential for steady, if unspectacular, progress?
  - Which of these could be said to overlap with some of the software process models from Jalote Chapter 2?

"Mythical Man-Month" - Chapter 1 - "The Tar Pit"

- How does Brooks think that a program and a programming system product compare? What are some of the reasons for this?
- What is compared to a tar pit, in this chapter? Why?
- What are some of the joys of the craft of programming? What are some of the woes of the craft of programming?

"Mythical Man-Month" - Chapter 2 - "The Mythical Man-Month"

- Know Brooks Law!! Be able to state it, and be able to explain the reasoning behind it.
- What is a man-month/person-month? Why is the person-month a dangerous and deceptive myth, according to Brooks?

- More software projects have gone awry for lack of what than for all other causes combined, according to Brooks?
- Why does Brooks claim that all programmers are optimists?
- What is Brooks' rule of thumb for scheduling a software task? How does it compare to "conventional"/customary thought on this?

"Mythical Man-Month" - Chapter 3 - "The Surgical Team"

- what have studies shown is the difference in productivity between very good and poor professional programmers, at the same training and experience level?
- According to Brooks, what is the problem with the small, sharp team concept?
- What have most experiences with using a "brute-force" approach to scaling up for developing really large systems resulted in?
- What is Brooks' proposal to address this problem?

"Mythical Man-Month" - Chapter 4 - "Aristocracy, Democracy, and System Design"

- Jalote will give a software-engineering view of software architecture in Chapter 5 of the course text, and you've had a course in computer architecture. In this chapter/in "The Mythical Man-Month", what does Brooks mean by the architecture of a system?
  - What should this make you think of from Jalote Chapter 3?
- What is conceptual integrity? How important does Brooks consider it to be in system design? Why does he consider it this important?
- What does Brooks argue should be the case with regard to architectural effort (design) and implementation in this chapter? What experience of his helped lead him to this conclusion?

"Mythical Man-Month" - Chapter 5 - "The Second-System Effect"

- What does Brooks claim is the most dangerous system that a person ever designs? Why?
- If one follows Brooks' advice with regard to architectural effort (design) and implementation from MMM Chapter 4, then what does Brooks further advise to keep the architect in check?
- Be aware both of the tendency to over-design in terms of functional embellishment and of the tendency "to refine techniques whose very existence has been made obsolete by changes in basic system assumptions".
- Be familiar with his suggestion here that a feature's cost -- not just in terms of memory, but also in terms of how it affects other features, and how it affects usability in general -- is important and worthy of consideration in system design;

Why Software Engineering? and The Software Problem

- Includes Jalote Chapter 1, as well as the posted 1995 Standish Group "Chaos" report.
- In the Standish Group "Chaos" report...

  - of the projects studied, about how many were successfully finished on-time and on-budget (and on-spec)?
  - About how many were never even completed (cancelled during the development cycle)?
  - About how many were completed, but not within time or not within budget or lacking some of the desired features?
- According to Jalote, what are the three main forces that drive an "industrial strength" software project?
- What does Jalote cite as being the most commonly-used measure of software size in the industry? What are some advantages of this measure? What are some drawbacks?
- Be familiar with the attributes of software quality as suggested by the International Standards Organization (the International Standards Organization (ISO) Software Engineering -- Product Quality -- Quality Model)
- Which Brooks' rule of thumb does Jalote reference in this chapter? How does Jalote characterize the two kinds of software being compared (what does he call those "kinds" of software)?

Jalote - Chapter 2 - "Software Processes"

- You need to be very familiar with the 6 software development process models discussed in Chapter 2 of Jalote.
  - You may need to describe them; you will very likely need to be able to distinguish them from one another, and to compare/contrast them with one another in various aspects.
  - What are some advantages of each? ...disadvantages of each?
  - Also consider: for what kinds of projects is each most suitable, and why? For what kinds of projects is each less suitable, and why?
  - What are some typical deliverables from the waterfall model stages?
  - Be especially aware of the listed limitations/concerns about the waterfall model.
  - You should also know the difference between the two Iterative development models discussed: iterative enhancement, and iterative delivery.
  - Which waterfall limitations are addressed by the prototyping model? ...by the iterative development model?
  - What is a cycle in the Rational Unified Process (RUP)? What are the four consecutive phases in each cycle? Why are these phase names deliberately different from the "engineering tasks" to be done in the project? Why is RUP considered to be particularly flexible?
  - What is distinctive about the Timeboxing model? What is the basic unit of development in this model? What is unusual about how its teams are organized? How do the durations of each time box, and each stage within a time box, compare?
- You need to be **very** familiar with the practices of Extreme Programming discussed both in Chapter 2 of Jalote and in the readings posted from the public course web site.
  - You may need to list or describe them; you may need to discuss them, and may need to discuss how they complement or support one another.
  - What is a user story? What is its purpose? What should it include? In what sort/style of language

should it be written? What are some desirable characteristics for user stories? (See also the Week 5 - Lecture 1 and Week 5 - Lecture 2 projected notes.)

- According to the IEEE standard glossary of software engineering terminology, what is a process?
  - What, then, is a process model?
  - How might you use a knowledge of software development process models to then choose/select/craft a process for a particular project? Why might this be beneficial?

Jalote - Chapter 3 - "Software Requirements Analysis and Specification"

- **Note: some of the following were not described in class, but WERE included in the required reading.**
- What is IEEE's definition of a requirement? (see projected notes from Week 6 - Lecture 1)
- What is a Software Requirements Specification (SRS)? What are some typical contents of an SRS?
  - Understand that, ideally, it should describe what the proposed software should do without describing how the software will do it.
  - What are some of the advantages of a "good" SRS, accordingly to Jalote?
- We discussed six desirable characteristics for an SRS. You should know what these are, should be able to describe them, should be able to discuss why they are desirable, and be able to compare/contrast them.
- Be familiar with what Jalote describes as the three basic tasks that result in an SRS:
  - problem analysis
  - requirements specification
  - requirements validation
  - (although note that these tasks are not in a linear sequence, but rather should have considerable overlap and feedback between them)
- What is meant by problem analysis? What is its basic purpose?
- What part of an SRS specifies the expected behavior of a system? What is our preferred mechanism for this part?
- Jalote also discusses some basic components of an SRS -- these include functional requirements, performance requirements, design constraints, and external interface specification. What is meant by each of these? What is the purpose of each within an SRS, and why might it be considered to be beneficial?
- Be familiar with use cases -- what they are, and their purpose within an SRS;
  - Know the expected and optional parts within use cases.
  - What are some advantages of use cases?
  - What is the relationship between use cases (as described in Jalote Chapter 3) and UML use case diagrams? Which, if either, is preferred? Why?
  - Know what primary actors, preconditions, main success scenarios, and exception scenarios are, and know their purposes within use cases.

- What are two other diagrams described in Jalote Chapter 2 that can also be very useful in the problem analysis task during software requirements analysis and specification?
  - What does each of these seek to show?
- What are the four most common types of errors that typically occur in an SRS, according to Jalote?
- What is the most commonly-used method for validating an SRS, according to Jalote?
  - What is this?
  - Who generally is involved in this?

Jalote - Chapter 4 - Planning a Software Project (up to Quality Control)

- (that is, for Exam 1, you are responsible for the first part of this chapter -- the parts on Quality Control and Risk Planning/Risk Management will be considered fair game for the Final Exam)
- This chapter notes that one of planning's basic objectives is to establish reasonable cost, schedule, and quality goals for the project. Which of these is considered by Jalote to be the hardest in terms of quantified goal setting and planning?
- What are the two basic objectives of planning, according to this chapter?
- Be familiar with both top-down and bottoms-up estimation methods (for effort estimation) as discussed in this chapter.
  - What are considered to be potential advantages of each? What are are considered to be potential pitfalls of each? What are some reasons/project characteristics that might make one of these preferred to the other?
  - What is a potential risk of a bottoms-up estimation method?
- Consider COCOMO 81. How are cost driver attributes used in this? Given an appropriate chart and formulas, you should be able to apply this approach.
  - What are some examples of cost driver attributes (what they are, not the particular multipliers)? Why might these effect one's estimates?
- What are some of the differences between COCOMO 81 and COCOMO II? (Consider the web site at USC and the links to it given in the Homework 5 handout as additional references for this.)
- What is a typical personnel ramp-up in a typical project? What are some of the approaches discussed for trying to determine an appropriate duration schedule and appropriate staffing levels?
  - Are all possible combinations of people and number of months possible? Why or why not? (Consider Brooks' argument in this area!)

Git and GitHub

- What is Git? What is GitHub? How are they related? How might they be used?
- What are some of the features/capabilities of Git?
- What are some of the features/capabilities of GitHub?

- What are some of the benefits of using Git?
- What are some of the benefits of using GitHub?
- You should be able to perform common actions with `git` that should be necessary for work on the course project -- for example, (and note that all of these are covered in the Week 3 and Week 4 Lab projections):
  - What command can you type to get the newest updates from the main repo?
  - What command can you type to check the status of your `git` repo (for example, to see if a file has been modified)?
  - Say that you have modified a given file. What command can you type to stage it for a commit?
  - ...what command can you then type the commit this modified file? What is the purpose of the string literal at the end of this command?
  - What command can you type to push changes you have made to the main repository?
- Recall that you tried out branches in git in Homework 4. What is a branch? What are some reasons one might want to create a branch?
  - What command allows you to create a branch? ...to switch to/check out a particular branch, so that you will now be working in that branch?
- What command allows you to list your current branches? ...to show that last commit on each branch?
- What command allows you to merge two branches? ...to delete a branch?
  - What are conflict resolution markers? When are they added to a file?