**Dreaming in Code Questions**

**Overarching Themes**

**Pay attention / record** the various roles that software engineers have on the Chandler Project.

**Pay attention / record** the many scheduling issues related to Chandler Project.

**Chapter 0**

1. Who wrote “software is hard?” Who is that guy?
   1. Donald Knuth the author of the programming field's most respected textbooks.
2. Programmers start counting at what number?
   1. Zero.
3. What was the original sense of a “hacker?”
   1. An obsessive programming tinkerer.
4. According to a 2002 NIST study what % of software came in significantly late, over budget, or was canceled?
   1. 66.67%
5. Who wrote the 1987 essay entitled “No Silver Bullet?”
   1. Frederick P. Brooks Jr. The expert who in many ways founded the modern field of software studies.

**Chapter 1**

1. What roles in the Chandler project did Michael Toy, John Anderson, Ted Burgess, Mitchell Kapor, and Lou Montulli hold.
   1. Toy was the manager.
   2. Anderson was the systems architect and projects leader. He has also written for Macintosh and has been a manager for Steve Jobs.
   3. Burgess was a young programmer at the time.
   4. Kapor was the father of Chandler and the found and funder of the Open Source Application Foundation.
   5. Montulli wrote key parts of the Netscape browser.
2. What is “[Bugzilla](http://www.bugzilla.org/)?”
   1. A program that holds lists of bugs and used to track all programming tasks that must be completed before the project can be released.
3. What is [OSAF](http://www.osafoundation.org/)?
   1. Open Sources Application Foundation.
4. What is the projects name?
   1. Chandler
5. What will the software do?
   1. Will be a Personal Information Manager (PIM). It will store calendars, emails, and to-do lists.
6. What is Toy’s keyword for “black hole” bugs?
   1. "Scary."
7. What scared Toy so much about Bug 44?
   1. The impossibility of knowing how long it would take to fix.
8. What did Toy refer to as a “snake?”
   1. A difficult and important problem that we don’t have a consensus on how to attack.
9. In the software world, what does “slippage” mean?
   1. The peculiar resistance of software projects to routine scheduling.
10. Fredrick Brooks was a programming manager for what software project?
    1. The creation of the operating system for the IBM system 360
11. What is [Brooks's Law](http://scottberkun.com/2006/exceptions-to-brooks-law/)?
    1. That adding manpower to a late software project makes it even later.
12. Brooks found what % of project time was spent writing code?
    1. 16.67%
13. Brooks found what % of project time was for testing and fixing bugs?
    1. 50%
14. Brooks observed that the unit of effort named “man-month” only applied under what conditions?
    1. He observed that men and months are interchangeable commodities only when a task can be partitioned among many workers with no communication among them and that they provide equal quality work.
15. What is the difference between source code and the program you install (.exe) on your computer?
    1. Commercial programs are typically binary. Companies do this to protect their secrets.
    2. Source code is actual human written code that you can look at, copy, and build on top of.
16. What is the one “article of faith” that all “open source” or “free” software advocates share?
    1. That software anyone can tinker with is bound to improve over time in ways that "closed" software can't match.
17. What is the difference between a “good” programmer and a “great” programmer?
    1. Good programmers know what write, great programmers know what to rewrite.
18. Eric Raymond’s book “[The Cathedral and the Bazaar](http://www.catb.org/esr/writings/cathedral-bazaar/)” made a distinction between two important project development ideas, briefly contrast them.
    1. Raymond believed that there needed to be structure, peace, serenity, time, and a few good men to have an outstanding project development.
    2. Whereas Torvald's project development relied on being open, releasing early and releasing often.
19. Has “open source” software project development refuted Brooks’s “mythical man-month” concerns?
    1. Yes, but only to an extent. It will only work with cheap and widespread access to a network with reliable communication among developers. In addition, it would need storage of common knowledge and code with a rise of a cooperative group ethos built around a leadership style.
20. What was [Andy Hertzfeld](http://andy.hertzfeld.usesthis.com/)’s input when the Chandler project appeared to have stalled?
    1. That they need to get going and modify later. He believes the key is getting exciting work going and the rest will follow.

**Chapter 2**

1. What was the lifetime as a supported product, of Lotus 123? When did Kapor walk away from it? Why did he walk away from it?
   1. Lotus was built for IBM PC in 1982. It was a spreadsheet to show the speed and power of the machine. Kapor walked away in July 1986 because he didn’t like his own success. He felt his success was overwhelming on a personal level.
2. What does it mean for a program to “fork?”
   1. Means to respond to technical disagreements by splitting into rival camps.
3. Linus Torvalds used a “science” and “witchcraft” analogy referring to software, explain.
   1. It's a metaphor. Science (open source) is when you learn, build on top of, and can look at other results. Witchcraft (traditional software) is when someone has a secret and guards it. Like in history witchcraft dies out.
4. Who, where, when demonstrated one of the first PIM software programs?
   1. Douglas Engelbart in1968 at the San Francisco convention center.
5. People often refer to starting their computer as “booting” their computer. What was the origin of this term?
   1. From the term pulling oneself up by the bootstraps.
6. Where was the graphical user interface (GUI) developed?
   1. At the Xerox research center in Palo Alto.
7. List three software project “train wrecks.”
   1. Trilogy (FBI)
   2. IRS system upgrade
   3. Pentagon's "Future Combat System"

**Chapter 3**

1. When introducing a new technology or design, why did Frederick Brooks advise “plan to throw one away?”
   1. Is that you are going to throw one way anyways so you might as well plan for it.
2. What is a “core” dump? Why the use of the word core?
   1. When a machine freezes up because of an irreconcilable conflict and the computer drops everything, grinds to a halt. The verbiage is from the fact that the core is a physical entity and not just a metaphor.
3. Rather than writing program statements in binary code, 110101110 1001101111, programmers developed a shorthand language called what?
   1. Assembly language
4. Adding layers of abstraction, new programming languages were created: Lisp, Cobol, Algol, Basic. Fortran was the first widely used. What kind of program converted Fortran to binary?
   1. Formula Translating System

**Chapter 4**

1. What do “front ends” and “back ends” mean to software developers?
   1. You the user are the “front end.” The front end is the the part of the program that deals with you. The back end is where the results of the front end events and inputs go so that the computer can make sense of them, save them, and retrieve them.
2. What did the Lego Hypothesis refer to?
   1. In the future all programs will be built out of reusable parts. Software parts will be available worldwide. Software engineering with be set free from the mundane necessity of programming.
3. Give one reason why the Lego Hypothesis seems to not work so well.
   1. If projects were like like the Lego hypothesis the would be small, invisible, and substitutable. They would be more similar to one another.

**Chapter 5**

1. What is the three-way trade-off that many software projects struggle to overcome.
   1. When creating fast, cheap, or good software you are only able to choose two.
2. What is the more recent definition of “geek?”
   1. a person who has chosen concentration rather than conformity; one who pursues skill, imagination, not mainstream social acceptance.
3. What does “refactoring” mean to programmers?
   1. Means rewriting a chunk of code to make it briefer, cleaner, and easier to read without changing what it actually does.
4. What is “yak-shaving?”
   1. a seemingly pointless activity which is actually necessary to solve a problem which solves a problem which , several levels of recursion later,solves the real problem you’re working on.

**Chapter 6**

1. What is term “edge cases” referring to in software development?
   1. millions of people use a piece of software like windows, sooner or later someone will end up making every possible unlikely choice that the software allows . They are often where bugs lie.
2. Summarize briefly Linus Torvalds advice about “large projects” give in 2004
   1. that nobody should take on a large project. Even if you started small , kept your ambitions in check, thought about details, and never, ever thought about the big picture you still shouldn't expect it to progress fast.

**Chapter 7**

1. Briefly describe Hungarian notation
   1. scheme for naming variables. In this notation the programmer appends a prefix to every variable name that gives anyone reading the code important clues about what type of variable it is. House style at Microsoft.
2. What does the author state is the “...single most challenging demand of software development.”
   1. Communicating abstractions unambiguously from programmer to machine , from programmer to programmer, and from program to user.

**Chapter 8**

1. What does “eat your own dogfood” mean?
   1. Means that software developers must themselves use the products they are in the process of building.
2. Quote: “When people ask for numbers that far out, the traditional thing that engineers do ....” When discussing the time line for Chandler, how was the quote above completed?
   1. … is make them up. Talking about time frame for release date.

**Chapter 9**

1. Structured programming evolved to address what programming practice?
   1. “Procedural pasta” To make a mess of code in a sense
2. Was structured programming a solution to the problem of software development?
   1. no. but it did pave the way for more ambitious software
3. Have any techniques shown to improve the software development process?
   1. Some developers are structured while other are not. However, counting bugs and fixing them asap has proven to help the software development process.
4. The “waterfall model” of programming was/is popular. What were some problems with this model?
   1. Surfaced circa 1970, divides project into an orderly sequence of discrete phases, like requirements definition, design, implementation, integration, testing, and deployment. Each phases ending before the start of the next. Can lead to confusion.
5. What are the four tenets of Agile Software Development?
   1. Individuals and interactions, working software, customer collaboration, responding to change.
6. What did a 2004 study find about the development practices of some two hundred software team leaders?
   1. The research found that the dominant practice is no practice all when it comes to development practices.
7. What is the “Joel Test” and what did he say about Microsoft and the Joel Test.
   1. The Joel test is a list of factors that must be marked off in order to receive points. The company with the most points is the best ranging in the 10 – 12 category. Any company with below 10 points isn’t doing well. Joel claims that Microsoft is getting better but still takes an extended amount of time to produce any type of software.
8. What is Rosenberg’s Law?
   1. Software is easy to make, except when you want it to do something new. And then, of course, there is a corollary: the only software thats worth making is software that does something new.

**Chapter 10**

1. Chapter 10 is about the notion of “Software Engineering” and the difficulty of applying this label to the development of software. The author suggests that Yertle the Turtle provides an important lesson for programmers. Describe it.
   1. Its a way of talking about systems that are internally coherent. About layers of abstractions , that don’t respond well to the failure of even one small part. The brittleness of software development.