**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?
2. Donald Knuth. Author of “The Art of Computer Programing”. Professor emeritus at Stanford University. Page 0.
3. Programmers start counting at what number?
4. Zero. Page 6.
5. What was the original sense of a “hacker?”
6. “Obsessive programming tinkerer.” Page 7.
7. According to a 2002 NIST study what % of software came in significantly late, over budget, or was canceled?
8. 66.7% or 2/3. Page 9.
9. Who wrote the 1987 essay entitled “No Silver Bullet?”
10. Fredrick P Brooks Jr.

**Chapter 1**

1. What roles in the Chandler project did Michael Toy, John Anderson, T[J]ed Burgess, Mitchell Kapor, and Lou Montulli hold.
2. Michael Toy is the manager of the software team for Chandler. John Anderson is the systems architect and effectively the lead coder. Jed Burges is a young programmer on the software team. Mitchell Kapor is the founder of the Chandler project and OSAF. Lou Montulli is a Chandler programmer who also wrote key parts in the Netscape browser.
3. What is “[Bugzilla](http://www.bugzilla.org)?”
4. A program that tracks open bugs and other programming tasks. Page 12.
5. What is [OSAF](http://www.osafoundation.org)?
6. Open Source Applications Foundation. Founded by Kapor. Page 12.
7. What is the projects name?
8. Chandler.
9. What will the software do?
10. It will be a personal information manager (PIM). It will manage and track things like people’s calendars, emails, etc. Page 13.
11. What is Toy’s keyword for “black hole” bugs?
12. Scary. Page 14.
13. What scared Toy so much about Bug 44?
14. The bug took much longer than expected and it represents an unknown amount of time to resolve. That in itself is not the scary part. That scary part is when one imagines that the issue could be multiplied times the number of bugs on file and there is no way to know how many if these instances will occur. Page 15.
15. What did Toy refer to as a “snake?”
16. An “important problem that we don’t have consensus on how to attack.” Page 15.
17. In the software world, what does “slippage” mean?
18. Lateness of a project. Page 15.
19. Fredrick Brooks was a programming manager for what software project?
20. Operating system for IBM System/360. Page 16.
21. What is [Brooks's Law](http://scottberkun.com/2006/exceptions-to-brooks-law/)?
22. “Adding manpower to a late project makes it later.” Page 16.
23. Brooks found what % of project time was spent writing code?
24. About 17% or 1/6. Page 17.
25. Brooks found what % of project time was for testing and fixing bugs?
26. 50% or one half. Page 17.
27. Brooks observed that the unit of effort named “man-month” only applied under what conditions?
28. “When a task can be portioned among many workers *with no communication among them.*” Page 17.
29. What is the difference between source code and the program you install (.exe) on your computer?
30. Source code is the precompiled code written by programmers in a programming language and possessed solely by those who wrote it and those they choose to share it with. The .exe is the product of the compiler and contains only zeros and ones readable only by the computer (and few very talented programmers). Page 20.
31. What is the one “article of faith” that all “open source” or “free” software advocates share?
32. “Software anyone can tinker with is bound to improve over time in ways that ‘closed’ software can’t match.” Page 22.
33. What is the difference between a “good” programmer and a “great” programmer?
34. “Good programmers know what to write. Great ones know what to rewrite (and reuse).” Page 24.
35. 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.
36. Raymond describes the “Cathedral” approach to software development as a small group of developers working together in an organized fashion in isolation. The “Bazaar” approach, on the other hand, he describes as an infinitely large group of seemingly unorganized people coming together with different agendas and somehow producing a product, and doing it well. Page 23.
37. Has “open source” software project development refuted Brooks’s “mythical man-month” concerns?
38. Raymond provides a possible transcendence to Brooks law under certain conditions using “Linus’s Law” which says, “Given enough eyeballs, all bugs are shallow.” He states, “To Brooks Law I counterpropose the following: Provided the development coordinator has a communications medium at least as good as the internet, and knows how to lead without coercion, many heads are inevitably better than one.” However, later, the implication is that Brooks did not live in the same programming world that we do now and, in his particular context, Brook’s law applied more obviously than it does in today’s diversity of programming situations.
39. What was [Andy Hertzfeld](http://andy.hertzfeld.usesthis.com)’s input when the Chandler project appeared to have stalled?
40. “Stop designing and start coding.” He also states that his style is to “get something going really quick and then turn it into the great thing that is the reason you’re doing it.” Page 30.

**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?
2. Lotus 1-2-3 made its debut in *Time Magazine* in 1982. In July 1986 Kapor resigned from Lotus “Extracting [himself] from [his] own success” as he put it. However Kapor stuck around to finish the Agenda project that was unveiled in 1988 but was considered unsupported sometime in the 1990s. Kapor states his reasoning for leaving by saying, “My perspective was really never totally shaped by needing to success in building a big company and making lots of money.” Essentially, he felt his purpose was unfulfilled with what he was doing. Page 36. Later, he also states that part of the reason for leaving was because of the open source conflict within the company at Lotus. Page 40.
3. What does it mean for a program to “fork?”
4. “To respond to technical disagreements by splitting into rival camps.” Page 41.
5. Linus Torvalds used a “science” and “witchcraft” analogy referring to software, explain.
6. The “Cathedral” approach (explained earlier) of small groups hoarding their knowledge in isolation is compared to “Witchcraft”, much like the common knowledge that, “a magician never reveals his secrets.” However, the “Bazaar” approach of open source sharing is compared to “Science” because science is usually subject to peer review. The implication is that traditional isolated programming will die out like witchcraft is considered to have died out, while open source will live on like science. Page 41.
7. Who, where, when demonstrated one of the first PIM software programs?
8. Douglas Engelbart in 1968 at the San Francisco Convention Center. Page 43.
9. People often refer to starting their computer as “booting” their computer. What was the origin of this term?
10. “The builders of early computer systems had borrowed the term from the concept of pulling one’s self up by the bootstraps to describe the paradox of getting a computer up and running. When you first turn a computer on, its memory is blank. That sets up a sort of chicken-and-egg paradox:. . .“ Booting a computer contains a process to resolve this paradox (originally named a “bootstrap loader”), thus “pulling one’s self up by the bootstraps.” Page 44.
11. Where was the graphical user interface (GUI) developed?
12. Xerox’s Palo Alto Research Center by some of Engelbarts former colleagues (among others). It was then popularized by Apple Macintosh.
13. List three software project “train wrecks.”
14. The FBI’s Virtual Case File project, the FBI’s Trilogy project that come to be known as “Tragedy”, three failed attempts at upgrading the IRS software. Then there are the disasters that took place in the UK, at McDonalds, and at Ford. Page 49. The software development word has no shortage of train wrecks.
15. What software crisis? Scan this article, what does it suggest?   
    <http://www.drdobbs.com/architecture-and-design/the-non-existent-software-crisis-debunki/240165910>
16. It implies that the “Software Crisis” is no crisis at all. That the situation has remained virtually unchanged and there is not cause for alarm.
17. Scan down this article to the two conclusions (about eight paragraphs down). With two sentences, what is your take away from this?  
    <http://scribblethink.org/Work/Softestim/softestim.html>
18. Software deadline estimates are not impossible but should be made carefully. The programmer should be honest with his/her supervisor and explain that the estimate may be rough (if it is) and that he/she may know more about how long it will take at such and such a date.

**Chapter 3**

1. Keep track of team members: 2001 --> Mitch Kapor, Morgen Sagen, Al Cho, Andy Hertzfeld
2. Kapor (founder of Chandler) and his first two employees on the Chandler project (Sagen and Cho) worked on a way to make their program as flexible as possible for the users. When Hertzfeld joined the team as a full-time volunteer, he brought his expertise in the field of music organization. Hertzfeld’s Vista, in combination with Sagen’s and Cho’s modification of RDF, comprised the first prototype of Chandler. Page 61.
3. When introducing a new technology or design, why did Frederick Brooks advise “plan to throw one away?”
4. Because according to Brooks, “you almost certainly won’t get it right the first time. All you can do is *plan* to get it wrong, or rashly promise to deliver a throwaway to customers.” Page 61.
5. 2002, Katie Parlante joined Chandler. When she first joined what did she do?
6. “She was happily researching user behavior patterns, exploring ways a PIM could help people avoid the tedious game of telephone tag, and compiling the most common complaints users have about their calendar software.” Page 62. She was kind of a secretary/researcher from what I gather.
7. John Anderson joined Chandler as “systems architect.” Briefly, what did that mean?
8. “He held chief responsibility for choosing the building blocks that the programmers would use to create their software.” Programming language, etc. Page 63.
9. What is a “core” dump? Why the use of the word core?
10. A core dump occurs when a computer freezes due to a conflict that cannot be resolved at that moment. The computer stops everything and reports the exact contents of its memory. It’s called a “core” dump because originally computer memory was stored on wire coils known as “ferrite cores.” Page 65.
11. How did Erik Sink, speaking of abstractions, describe what programmers do?
12. “We build piles of abstractions. We design our own abstractions and then pile them up on top of layers we got from somebody else.” Page 66.
13. Rather than writing program statements in binary code, 110101110 1001101111, programmers developed a shorthand language called what?
14. Assembly language. Page 66.
15. 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?
16. A compiler. Page 67.
17. What are the implications behind: “...there is no [Moore’s Law](http://www.mooreslaw.org) for software. Chips may double in capacity every year or two; our brains don’t.”?
18. The time it takes a programmer to produce a product is more valuable than the speed of the program on the computer. This is because computer speed will continue to grow exponentially but brains don’t and the programmer’s time spent is in short demand.
19. The language selected for Chandler was [Python](https://www.python.org). What does it mean that Python was an interpreted language?
20. The program is compiled at the time it is run on the computer rather than before the product is shipped. It requires an interpreter to be run with it at the time the program is executed. Page 70.
21. The [announcement of Chandler](http://seclists.org/interesting-people/2002/Oct/83) occurred when? ([Interesting Paper](http://web.archive.org/web/20030504050437/http://www.riehle.org/vuw-mirror/CS-TR-02-9.pdf)) [Slashdot page from 2002](http://web.archive.org/web/20021021082601/http://slashdot.org/).
22. October 20th 2002 at the urging of Kapor’s friend, Dan Gillmor.

**Chapter 4**

1. What do “front ends” and “back ends” mean to software developers?
2. The front end is what the user sees and interacts with. The back end is where all the input from the user is processed. “The front end talks to people; the back end talks to bits.” In the Chandler prototype, the front end is Vista and the backend in Shimmer. Page 86.
3. The terms RAP, SQL, object persistence, all relate to a problem Chandler’s developers needed to solve at the earliest stage. What was the problem?
4. They need a repository storage system that is inclined to “object persistence” and lends itself to flexibility and speed. It also must allow for mobility of the user and function well across a network. Page 91.
5. What did the Lego Hypothesis refer to?
6. It refers to a notion proposed by James Noble and Robert Biddle saying that, “In the future, programs will be built out of reusable parts. Software parts will be available worldwide. Software engineering will be set free from the mundane necessity of programming.” Supposedly the programmers will be able to, “Pull some pieces off the shelf, snap them together, and presto – working software, with no painful coding.” Page 94.
7. Give one reason why the Lego Hypothesis seems to not work so well.
8. The software components are not like Legos in the fact that the components are more different from each other than they are alike. Page 94.
9. How many “modules” does CPAN currently host?
10. 8,107 at the time this was written. Page 98
11. After Chandler was announced in Oct. 2002, when was the first public release?
12. April 21st 2003 at 3:07 PST.

**Chapter 5**

1. What is the three-way trade-off that many software projects struggle to overcome.
2. The quality triangle” or “impossible triangle” is the principle that software can be made “fast, cheap, or good. Pick any two.” Page 119.
3. What is the more recent definition of “geek?”
4. “Geek: a person who has chosen concentration rather than conformity; one who pursues skill (especially technical skill) and imagination, not mainstream social acceptance.” Page 131.
5. What does “refactoring” mean to programmers?
6. “To programmers, refactoring means to rewrite a chunk of code to make it briefer, clearer, and easier to read without changing what it actually does. Refactoring is often compared to gardening; it is never finished.” Page 140.
7. What is “yak-shaving?”
8. “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.’” Page 144.

**Chapter 6**

1. What is term “edge cases” referring to in software development?
2. “Edge cases” are instances where users engage a program in a series of highly strange actions that the program was not directly built for. The idea is that every program will eventually be used for even the most unorthodox methods if in existence long enough (a kind of programmer’s Murphy’s Law). Edge cases often end up revealing hidden bugs. Page 148.
3. Summarize briefly Linus Torvalds advice about “large projects” give in 2004
4. Don’t try to start a “large project”. Start small. Design a small program that addresses an immediate need. Focus on details, not the big picture and expect it to take time. Page 174.

**Chapter 7**

1. Briefly describe Hungarian notation
2. “In Hungarian notation, the programmer appends a prefix to every variable name that gives anyone reading the code important clues about what sort [type] of variable it is.” - “Type” was originally meant in a loose sense, to give more information about the variable, not in the more rigid, syntax oriented, sense that Microsoft took Hungarian notation to mean. Page 198.
3. What does the author state is the “...single most challenging demand of software development.”
4. “Communicating abstractions unambiguously – from programmer to machine, from programmer to programmer, and from program to user – is the single most challenging demand of software development.” Page 198.

**Chapter 8**

1. What does “eat your own dogfood” mean?
2. “That software developers must themselves use the products they are in the process of building” Page 208.
3. Quote: “When people ask for numbers that far out, the traditional thing that engineers do ....” When discussing the timeline for Chandler, how was the quote above completed?
4. “. . . is make them up.” Page 219.

**Chapter 9**

1. After about two years with little to show for it, the author suggests the project had something else important. What?
2. Structured programming evolved to address what programming practice?
3. Was structured programming a solution to the problem of software development?
4. Have any techniques shown to improve the software development process?
5. Give a few software failures the author lists. Did anyone die due to the errors?
6. The “waterfall model” of programming was/is popular. What were some problems with this model?
7. What are the four tenets of Agile Software Development?
8. What did a 2004 study find about the development practices of some two hundred software team leaders?
9. What is the “Joel Test” and what did he say about Microsoft and the Joel Test.
10. What is Rosenberg’s Law?

**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.

**Remaining Pages**

Complete the reading reflecting on the Chandler Projects **scheduling** issues and the various **project roles** that were important on the project.