DREAMING IN CODE ANSWERS
Kelbe Faast

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?
   Donald Knuth, Author of *The Art of Computer Programming*
2. Programmers start counting at what number?
   Programmers start counting at 0 because computers count from 0.
3. What was the original sense of a "hacker?"
   The original sense of hacker was "obsessive programming tinkerer"
4. According to a 2002 NIST study what % of software came in significantly late, over budget, or was canceled?
   According to NIST, two out of three, or 66.66%.
5. Who wrote the 1987 essay entitled "No Silver Bullet?"
   No Silver Bullet was written by Frederick P. Brooks Jr.

Chapter 1

1. What roles in the Chandler project did Michael Toy, John Anderson, Ted Burgess, Mitchell Kapor, and Lou Montulli hold.
   They are all programmers in making Chandler, with Toy being the manager, and who all work under the Open Source Applications Foundation.
2. What is "Bugzilla?"
   A collection of difficult to fix bugs.
3. What is OSAF?
   OSAF stands for Open Source Application Foundation
4. What is the projects name?
   The project's name is Chandler
5. What will the software do?
   Chandler's goal is to be the ultimate personal information manager (PIM).
6. What is Toy's keyword for "black hole" bugs?
   Bugzilla
7. What scared Toy so much about Bug 44?

Besides being a black-hole bug, it was also a bug that came about from one of the building blocks of the code, not the code they themselves had created. So either they had to wait for a fix from the original developers or create a work around of their own.

8. What did Toy refer to as a "snake?"

   A snake was an 'important problem we don't know how to attack'.

9. In the software world, what does "slippage" mean?

   Slippage is a synonym for lateness.

10. Fredrick Brooks was a programming manager for what software project?

    Brooks was a programming manager for IBM.

11. What is Brooks's Law?

    Brooks's Law is the saying that "adding manpower to a software project makes it later".

12. Brooks found what % of project time was spent writing code?

    ⅙ of the time was spent writing code.

13. Brooks found what % of project time was for testing and fixing bugs?

    50% of the time was spent fixing bugs.

14. Brooks observed that the unit of effort named "man-month" only applied under what conditions?

    A man-month can only be applied when the work is distributed between many workers with no communication between them.

15. What is the difference between source code and the program you install (.exe) on your computer?

    Source code is the human written language that makes the code work, but it is often kept secret to prevent reverse engineering. One of the ways this is done is through computer compiling which translates the human code into something only other computers could read. Exe files on a computer are computer compiled code lines.

16. What is the one "article of faith" that all "open source" or "free" software advocates share?

    Software that anyone can mess with is going to improve over time.

17. What is the difference between a "good" programmer and a "great" programmer?

    Good programmers know what to write, great programmers know what to rewrite.

18. Eric Raymond's book "The Cathedral and the Bazaar" made a distinction between two important project development ideas, briefly contrast them.

    Linus's law, in contrast to Brook's Law, states that adding more people to a project does not hinder it when the communication is fast and reliable(via the internet), and would be easy to access pools of information to be shared.

19. Has "open source" software project development refuted Brooks's "mythical man-month" concerns?

No, Brooks's concerns still hold merit as more workers meant more bugs goud which inevitably delayed progress.

20. What was Andy Hertzfeld's input when the Chandler project appeared to have stalled?

    Hertzfeld said that once you get exciting work going, the rest will take care of itself.

Chapter 2

1. Linus Torvalds used a "science" and "witchcraft" analogy referring to software, explain.

   Science is the amalgamation of multiple people building upon a project together while witchcraft is a guarded secret, which is bound to die out over time. Software must not be kept secret in order to stay relevant.

2. People often refer to starting their computer as "booting" their computer. What was the origin of this term?

   The origin of booting up a computer came from the phase 'pulling one's self up by the bootstraps'.

3. Where was the graphical user interface (GUI) developed?

   GUI was developed at Xerox's Palo Alto Research Center.

4. List three software project "train wrecks."

   Salon site launch, Virtual Case File, Advanced Automation System

Chapter 3

1. When introducing a new technology or design, why did Frederick Brooks advise "plan to throw one away?"

   Because you probably won't get it right on the first try.

2. What is a "core" dump? Why the use of the word core?

   A core dump is a file as a result of a system crash that details the memory contents at the moment of failure. The word core is used because memory banks used to be made out of wire coils called ferrite cores.

3. Rather than writing program statements in binary code, 110101110 1001101111, programmers developed a shorthand language called what?

   The shorthand is called 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?

   A program called a compiler changed the source code into binary.

Chapter 4

1. What do “front ends” and “back ends” mean to software developers?

   The front end is the user, or the code the user interacted with while the back end is where the results of the front end code goes so the computer can interpret it.
2. What did the Lego Hypothesis refer to?

   The programmers dream to streamline the software making process by use of plug-in parts.
3. Give one reason why the Lego Hypothesis seems to not work so well.

   While legos relate to uniformity, software is hardly ever similar to another piece of related content, so its hard to make them easy to piece together.

Chapter 5

1. What is the three-way trade-off that many software projects struggle to overcome.

   The things that make up the three-way trade-off, or quality triangle is fast, cheap, or good.
2. What is the more recent definition of “geek?”

   To be a geek is to have easier relations with computers than with other humans.
3. What does “refactoring” mean to programmers?

   Refactoring means to rewrite code so that it is shorter and easier to read without changing its function.
4. What is “yak-shaving?”

   Something that seems useless but is actually necessary to solve a problem.

Chapter 6

1. What is term “edge cases” referring to in software development?

   Edge cases refer to far out scenarios that could happen and result in bugs in the programming.
2. Summarize briefly Linus Torvalds advice about “large projects” give in 2004

   Never start a large project. Start smaller projects and focus on the details and if it grows larger over time that is fine as long as you don’t start treating it like something large.

Chapter 7

1. Briefly describe Hungarian notation

   Hungarian Notation is giving every variable a prefix that hints towards what it is, even if at first it looks like a foreign language.
2. What does the author state is the “...single most challenging demand of software development.”

   The single most challenging demand of software making is communicating in a way with only one possible way to interpret it.

Chapter 8

1. What does “eat your own dogfood” mean?

Programmers must also use the thing they are building.

2. 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?

   “... make them up.”

Chapter 9

1. Structured programming evolved to address what programming practice?

   Structure programming evolved to eliminate the use of GOTO and to keep developers from writing messy code, called spaghetti code.

2. Was structured programming a solution to the problem of software development?

   No because despite its innovation software will continue to fail in ways never before imagined.

3. Have any techniques shown to improve the software development process?

   Rapid Application Development, Patterns and the waterfall model all helped to improve the software development process.

4. The “waterfall model” of programming was/is popular. What were some problems with this model?

   Some problems that came from the waterfall model was how slow the process was, and that would lead to people sitting around waiting for the requirements to arrive, while another group would skip waiting and just start working ahead without the correct instructions. Then writing the code separately and putting it together later would end up crashing the system when the codes did not mesh well.

5. What are the four tenets of Agile Software Development?

   The four tenets of Agile Software Development are:

   Individuals and interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan.

6. What did a 2004 study find about the development practices of some two hundred software team leaders?

   The study found the the most dominant practice amongst the team leaders was no practice at all.

7. What is the “Joel Test” and what did he say about Microsoft and the Joel Test.

   The Joel Test is similar to CMM in that it judges development organizations and how well their products turn out. Spolsky says that Microsoft gets a perfect score continuously, and that smaller companies need to step up their game.

8. What is Rosenberg’s Law?

   Rosenberg's Law states that software is easy to make unless you want it to do 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.

   The most important lesson a programmer can get out of the tale *Yurtle the Turtle* is that even the smallest of bumps in a stack of code can make the whole thing crash. Software should be made to bend, not break at the slightest crack.

Remaining Pages

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