The Realities of Software Testing

[Reading assignment: Chapter 3, pp. 37-50]
Software process models are an ideal ... not reality

- No software development effort follows a process model perfectly. Why?
  - The specification never corresponds to the customers needs perfectly.
  - There is never enough time to perform all of the testing.
- Nevertheless, an ideal model is helpful to make progress.
  - Trade offs and concessions are inevitable.
Plato had some interesting things to say about the ideal

- Physical objects and physical events are shadows of their ideal or perfect forms.
- They exist only to the extent that they instantiate the perfect versions of themselves.
- However, real physical objects are fleeting phenomena, the ideals of which they are mere instances.
- Plato believed in perfect justice, perfect chairs, etc. He would have also believed that the Waterfall model (or some variant of it) was a perfect process model :-(
Software testing axioms

1. It is impossible to test a program completely.
2. Software testing is a risk-based exercise.
3. Testing cannot show the absence of bugs.
4. The more bugs you find, the more bugs there are.
5. Not all bugs found will be fixed.
6. It is difficult to say when a bug is indeed a bug.
7. Specifications are never final.
8. Software testers are not the most popular members of a project.
9. Software testing is a disciplined and technical profession.
Parenthesis:
What is an axiom anyway?

• An **axiom** is a sentence or proposition that is not proved or demonstrated and is considered as obvious or as an initial necessary consensus for a theory building or acceptation.

• Therefore, it is taken for granted as true, and serves as a starting point for deducing and inferring other (theory dependent) truths.
Peano’s axioms for the structure of natural numbers

1. 1 is a natural number.
2. Every natural number is equal to itself (equality is reflexive).
3. For all natural numbers \(a\) and \(b\), \(a = b\) if and only if \(b = a\) (equality is symmetric).
4. For all natural numbers \(a\), \(b\), and \(c\), if \(a = b\) and \(b = c\) then \(a = c\) (equality is transitive).
5. If \(a = b\) and \(b\) is a natural number then \(a\) is a natural number.
6. If \(a\) is a natural number then \(\text{successor}(a)\) is a natural number.
7. If \(a\) and \(b\) are natural numbers then \(a = b\) if and only if \(\text{successor}(a) = \text{successor}(b)\).
8. If \(a\) is a natural number then \(\text{successor}(a)\) is not equal to 1.
9. For every set \(K\), if 1 is in \(K\), and \(\text{successor}(x)\) is in \(K\) for every natural number \(x\) in \(K\), then every natural number is in \(K\).
Discussion …

• Are software testing axioms really axioms?
• If not, what would you call them?
Axiom 1

*It is impossible to test a program completely*

- How many test cases do you need to exhaustively test:
  - Powerpoint
  - A calculator
  - MS Word
  - Any interesting software!
- The only way to be absolutely sure software works is to run it against all possible inputs and observe all of its outputs …
- Oh, and the specification must be correct and complete.
Axiom 1 (cont’d)

*It is impossible to test a program completely*

- The number of possible inputs is very large.
- The number of possible outputs is very large.
- The number of paths through the software is very large.
- The software specification open to interpretation.
Axiom 2

Software testing is a risk-based exercise

• If you do not test the software for all inputs (a wise choice) you take a risk.
• Hopefully you will skip a lot of inputs that work correctly.
• What if you skip inputs that cause a fault?
  – Risk: financial loss, security, loss of money, loss of life!
  – That is a lot of pressure for a tester!
• This course is all about techniques and practices to help reduce the risk without breaking the bank.
Axiom 2 (cont’d)
Software testing is a risk-based exercise

- If you try to test too much, the development cost becomes prohibitive.
- If you test too little, the probability of software failure increases and as we discussed … software failures can cost us big time!
Axiom 2 (cont’d)

Software testing is a risk-based exercise

What about Murphy’s Law?

• "If there's more than one possible outcome of a job or task, and one of those outcomes will result in disaster or an undesirable consequence, then somebody will do it that way."

• The law's name stems from an attempt to use new measurement devices developed by one Edward Murphy.

• The phrase was coined in reaction to something said when his devices failed to perform eventually cast into its present form prior conference months later.
Axiom 3

Testing cannot show the absence of bugs

• “Program testing can be used to show the presence of bugs, but never to show their absence!”
  - Edsger Wybe Dijkstra

• Dijkstra received the 1972 ACM Turing Award for fundamental contributions in the area of programming languages
Discussion ...

• What is the ACM?
• What is the ACM Turing Award?
• Who is Alan Turing?
Axiom 4

The more bugs you find, the more bugs there are

- Bugs appear in groups, where you see one you will likely find more … Why?
  - Programmers can have bad days
  - Programmers tend to make the same mistakes
  - Some bugs are just the tip of the iceberg.

- Boris Beizer coined the term **pesticide paradox** to describe the phenomenon that the more you test software the more immune it becomes to your test cases.
  - **Remedy:** continually write new and different tests to exercise different parts of the software.
Axiom 5
Not all bugs found will be fixed

• Why wouldn’t you fix a bug you knew about?
  – There’s not enough time
    • Some deadlines cannot be extended (e.g., Y2K)
  – It’s not really a bug
    • Specifications can be wrong
  – It’s too risky to fix
    • “I’m not touching Murphy’s code!”
  – It’s just not worth it
    • Bugs in fringe features may have to wait
    • Why not charge the customer for bug fixes in the next release (sound familiar?) :-)

Axiom 6

*It is difficult to say when a bug is indeed a bug*

- If there is a problem in the software but no one ever discovers it … is it a bug?
  - Parody of “if a tree falls in the forest … does it really make a noise?”
- What is your opinion? Does a bug have to be observable in order for it to me a bug?
- Bugs that are undiscovered are called *latent bugs.*
Axiom 7

Specifications are never final

• Building a product based on a “moving target” specification is fairly unique to software development.
  – Competition is fierce
  – Very rapid release cycles
  – Software is “easy” to change

• Not true in other engineering domains
  – E.g., the Brooklyn Bridge could not be adjusted to allow train traffic to cross it once its construction started.
Dear Mr. Architect,

Please design and build me a house. I am not quite sure of what I need, so you should use your discretion.

My house should have between two and forty-five bedrooms. Just make sure the plans are such that bedrooms can be easily added or deleted. When you bring the blueprints to me, I will make the final decision of what I want. Also bring me the cost breakdown for each configuration so that I can arbitrarily pick one.
A Story … (Cont’d)

Keep in mind that the house I ultimately chose must cost less than the one I am currently living in. Make sure, however, that you correct all the deficiencies that currently exist in my house (the floor of my kitchen vibrates when I walk across it, and the walls don’t have nearly enough insulation in them).

Also keep in mind as you design this house that I wish to keep yearly maintenance cost as low as possible. This should mean the incorporation of extra-cost features like aluminum or vinyl siding. If you chose not to specify aluminum, be prepared to explain in detail.
Please take care that modern design practices and the latest materials are used in construction of the house. The house should be really nice. However, be alerted that the kitchen should be designed to accommodate among other things, my 1952 Gibson refrigerator.

To assure that you are building the correct house for our family, make sure that you contact each of the children and also the in-laws. My mother-in-law will have very strong feelings about how the house ought to be designed since she visits with us at least once a year. Make sure that you weigh all these options carefully and make the right decision. I, however, retain the right to override any decision you come up with.
Please don’t bother me with small details right now. Your job is to develop the overall plans for this house. Get the big picture. It is not appropriate at this time to be choosing the color of the carpet. However, keep in mind that my wife likes green.

Also do not worry at this time about acquiring resources to build this house. Your first priority is to develop detailed plans and specifications. However, once I accept these plans, I will expect to have the house under roof within 48 hours.
A Story … (Cont’d)

While you are designing this house specifically for me, keep in mind that sooner or later I will have to sell this house. It should have appeal to potential buyers. Please make sure that before you finalize the plans, there is a consensus of the population in my area that they like the features this house has.

You are advised to run up and look at my neighbor’s house he had constructed last year. We like it a great deal. It has many features that we would like to have in our new home, particularly the 75-foot swimming pool. With careful engineering I believe that you can design this into our new house without impacting the construction cost.
A Story … (Cont’d)

Please prepare a complete set of blueprints. It is not necessary at this time to do the real design since these blueprints will be used only for construction bids. Please be advised however, that any increase of cost in the future as a result of design changes will result in you getting your hands slapped.

You must be thrilled to be working on such an interesting project such as this. To be able to use new kinds of construction and to be given such freedom in your designs is something that doesn’t happen very often. Contact me as rapidly as possible with your design ideas. I am enthusiastic about seeing what you can come up with.
A Story … (Cont’d)

P.S. My wife has just told me that she disagrees with many on the instructions I’ve given you in this letter. As architect it is your responsibility to resolve these issues. I have tried in the past and have been unable to accomplish this. If you can’t handle this, I’ll have to look for a new architect.

P.P.S. Perhaps what I need is not a house at all, but a travel trailer. Please advise me as early as possible if that is the case.
Axiom 8

Software testers are not the most popular members of a project

• Goal of a software tester:
  – Find bugs
  – Find bugs early
  – Make sure bugs get fixed

• Tips to avoid becoming unpopular:
  – Find bugs early
  – Temper your enthusiasm … act in a professional manner
  – Don’t report just the bad news
Axiom 9

Software testing is a disciplined and technical profession

• When software was simpler and more manageable software testers were often untrained and testing was not done methodically.
  – Contrary to the urban legend, if you hire a million moneys and have them test for a million years … you will not find all of the bugs in your software.
  – Sorry Mr. Burns :-)

• It is now too costly to build buggy software. As a result testing has matured as a discipline.
  – Sophisticated techniques
  – Tool support
  – Rewarding careers
Some Terminology

• Verification
  – “Are we building the product right?”
  – Does the software meet the specification?

• Validation
  – “Are we building the right product?”
  – Does the software meet the user requirements?

• Software Quality Assurance
  – Not the same as software testing …
  – Create and enforce standards and methods to improve the development process and to prevent bugs from occurring.
You now know …

• … the 9 axioms of software testing
• … what is software verification
• … what is software validation
• … the relationship between software quality and reliability
• … the relationship between testing and quality assurance