SYST 101: Intro to Systems

Lecture 18

Mar 25, 2004

C. Wells, SEOR Dept.

Agenda

- Problem resolution

Failure Analysis In Lego Robots

- Like many real systems, we have a combination of hardware and software.
- When behavior is not as expected, where is the problem?
- Typically, on large project, you have your hardware team and your software team.

Typical Team Failure Analysis

Software Can Be Tricky

- Validation and verification of software is problematic at best
  - Impossible to test all conditions
- Failure modes are harder to identify
  - Failure sources can be computer hardware, logic, or coding
- Compilation and optimization can exacerbate the problem

A Basic Problem

- Knowing you’re right
- Leads to
  - Dead ends
  - Being stuck
  - Conflict with others who know they’re right too.

The Pencil Point Analysis

- All the steps were right, so why isn't the answer right?
  - An assumption was invalid, but was overlooked.
  - An inadvertent error exists but is repeatedly missed because the mind sees what it expects to see

Human Nature

- It’s human nature, so it happens to all of us
- Not really feasible to avoid it happening
- Learn to recognize and learn how to get unstuck.

You May Be Stuck If...

- You’re sure you’ve done everything right, and it’s not working
- You blame the device for not doing what it is supposed to do (cheap parts, etc.)
- You know there’s this hidden thing that is messing you up that you can’t get to.

Mental Models

- You have a mental model of your system in your head
  - Especially when *you* designed it
- As it operates, your interpret what it is doing as matching the mental model
- Your assumptions about your mental model may lead you to overlook the actual behavior of your system

Getting Unstuck

- Become an independent observer
  - Observe your system operating, without following your mental model
  - Watch what it does, and only what it does
  - Preconceptions about what is wrong can keep your attention away from the real difficulty

Using Your Team

- Appoint parts of your team as developers, other parts as testers
- Example:
  - Team A designs mechanical, tests program
  - Team B tests mechanical, writes programs

“Pride of Ownership”

- A phrase often heard in the real world.
- Meaning: “I built this, and if you criticize what I built, you’re criticizing me, and I will take it personally.”
- Effects: Criticisms dismissed as “stupid” or “ignorant”. “They really don’t understand what’s going on if they can make comments like that”.

“Pride of Ownership”

- Leads to rigid thinking, no adaptation, bad designs.
- Lesson: Don’t take it personally!!

"NIH" - Not Invented Here

- Translation: "If we didn't invent it, then it can't be any good".
- Allows a team to dismiss others' ideas and criticisms without truly considering them.

Hardware Designs

- Symmetric Vs. Asymmetric?
  - Symmetric means that it's the same on either side of a line (usually left-right symmetry in our case)
- Asymmetric designs may be harder to make perform exactly as you want.
  - Propeller aircraft turn one way better than the other.

Separate Testing

- Disconnect the hardware from the software:
- Is is supposed to roll in a straight line? Does it? Can it turn if it needs to?
  - A Test” Disconnect the motors from the wheels (remove gears) and roll it down a hill.
  - Try to make it turn

Assignments

- Homework
  - Give an example of an instance where you solved a problem or made a decision, outside of class, using system engineering processes, procedures, and techniques scope of SE consistent with the problem at hand
    - not end to end process, just part of the SE process
    - ***must*** include the observations that initiated the process and the observations made during the process
    - (to see if you use, and recognize you are using, SE tools)