CS 428 Week #8 Readings

WINTER 2020, WEEK #8

BRUCE F. WEBSTER

MMM Ch 16: No Silver Bullet – Essence and Accident in Software Engineering (1986)

- Probably one of the single most important essays ever written about IT
  - Core argument: “Building software will always be hard. There is inherently no silver bullet [to slay the monsters of software development].”
- Four inescapable essential difficulties in software development
  - Complexity: increases non-linearly with program size, both technically and managerially
  - Conformity: code must “work with” its ever-more-complex environment
  - Changeability: constant pressure to improve or fix existing systems
  - Invisibility: software is extremely hard to inspect and examine (vs., say, a building)

MMM Ch 16: No Silver bullet (cont.)

- Things that do help
  - Buy vs. build
    - Buy and adapt (or adapt to) an existing solution that someone else had built and maintains
  - Requirements refinement and rapid prototyping
    - "…it is really impossible for clients, even those working with software engineers, to specify completely, precisely, and correctly the exact requirements of a modern software product before having built and tried some versions of the product they are specifying."
  - Incremental development
    - "A large, complex system that works is inevitably found to have evolved from a small, simply system that works." – John Gall, Infomatics
  - Great designers
    - "The very best designers produce structures that are faster, smaller, simpler, cleaners, and produced with less effort. . . . Those software systems that have excited passionate fans are the products of one or a few designing minds, great designers."
- Analysis and observations?

MMM Ch 17: "No Silver Bullet" Refired

- "I can't help noticing that the nostrums published so vigorously in 1986 and 1987 have not had the dramatic effects claimed."
- Brad Cox in 1990: "The reusable, interchangeable component approach [is] an attack on the conceptual essence of the problem." This lead to the 'reuse' push of the 1990s, which failed utterly.
- David Harel in 1992 offers "The Vanilla Framework". Ever heard of it?
- Object-oriented development: also another brass slug (hence my book "Pitfalls of Object-Oriented Development" [1995])
- Brooks says his analysis stands; 30 years later, I agree with him.
- Analysis and observations?

PW Ch 7: The Furniture Police

- "[Those making workplace decisions] are not themselves doing the kind of work that is likely to suffer from a poor environment."
- Goals are focused on ease and flexibility of setting up the physical workspace, not on productivity of those who work there.
- Attitude: If everyone can't have a window, then no one can.
- "Almost without exception, the work space given to intellect workers is noisy, interruptive, un-private, and sterile."
- Observations and feedback?

PW Ch 8: "You never get anything done around here between 9 and 5"

- "…overtime is not so much a means to increase the *quantity* of work time as to improve its average *quality*."
  - Fewer interruptions/disturbances outside of regular work hours or at home
- Individual differences (best outperform worse by 10:1)
- Productivity non-factors: language, years of experience, defects, salary
- There is also a 10:1 difference in productivity among software organizations
  - Cf. "Dead Sea Effect"
- Top performers' space is quieter, more private, better protected from interruption, larger
- Observations and feedback?

PW Ch 9: Saving Money on Space

- Cost-saving trend towards less privacy, less dedicated space, more noise
- But cost of work space is small fraction of cost of developer – false economy
- Claims of greater productivity & interaction for open space aren't supported
- Correlations between perceived noise level and defects in work
  - Zero-defect workers: 66% reported noise level ok
  - 1-or-more defects: 8% reported noise level ok
- Noise is generally proportional to workplace density
- Worker response is often to "hide out" where it's quieter
- Observations and feedback?

- So, why isn't this all obvious and followed? Because of how few firms know how to or are willing to measure impact of environment on productivity
- But: "Given that there are 10:1 differences from one organization to another in productivity, you simply can't afford to remain ignorant of where you stand."
- Observations and feedback?

PW Part II Intermezzo: Productivity Measurement

PW Chapter 10: brain time vs body time

- During single-minded work time, people are ideally in “flow” state
  - Deep, nearly meditative involvement
  - Sense of euphoria
  - Unaware of passage of time
- It takes time to enter “flow” state, and interruptions force you to restart
  - Constant interruptions keep us in a state of “no-flow” and far less productive
- E-Factor: uninterrupted hours / body-present hours
  - Boss: “Can’t you do [your thinking] at home?”
- Observations and feedback?

PW Chapter 11: The Telephone

- Chapter is a touch dated – younger generation has learned to ignore phones
- But now: various messaging feeds and apps, social media, e-mail, etc., can all interrupt our flow
- To achieve and preserve flow, *we* have to be willing to shut off these distractions
- Observations and feedback?

- Like windows, doors are frequently a status symbol – and therefore, if everyone can't have no, nobody can have one
- Workers aren't inspired or made more productive because the (open) workplace has "fashionable" or "daring" or "amusing" design
- Piping music into an open workplace doesn't help either
- It's great to have "vital" space for spontaneous interaction w/others, but most IT production is solitary, flow-based intellectual work
- Observations and feedback?

PW Chapter 12: Bring back the door

- Christopher Alexander's *The Timeless Way of Building* and design pattern
  - Alexander on workspaces (pp. 82-83)
  - Cubicles are almost the direct opposite of what Alexander points out
- Tailored workspaces from a pattern
- Use of windows
- Indoor and outdoor space
- Public space
- "No two people have to have exactly the same work space."
- Observations and feedback?

PW Chapter 13: Taking Umbrella Steps

ACC Chapter 9: Making Work Sustainable

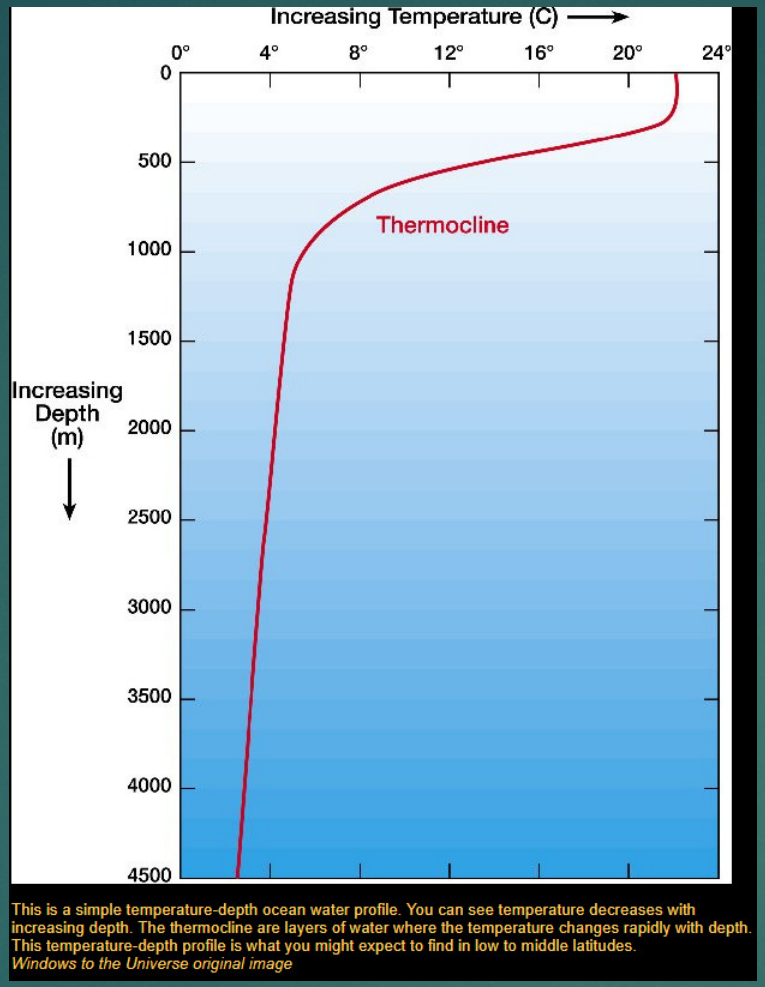
- Where code deployments are most painful => poorest software delivery performance, organizational performance, culture
- Detecting deployment pain:
  - Are deployments feared?
  - Are deployments disruptive to work?
- To reduce deployment pain, build systems that:
  - Are designed to be deployed easily into multiple environments
  - Can detect and tolerate failures in their environments
  - Can have various components of the systems updated independently (loose coupling)
  - Also: ensure state of production systems can be reproduced automatically from version control
  - And: build intelligence into the app & platform so that deployment is simple as possible

AC Chapter 9 (cont.)

- Burnout: "physical, mental, or emotional exhaustion caused by overwork or stress"
  - Work overload: job demands exceed human limits
  - Lack of control: inability to influence decisions that affect your job
  - Insufficient rewards: financial, institutional, social
  - Breakdown of community: unsupportive work environment
  - Absence of fairness in decision-making process
  - Value conflict between organization and individual

ACC Chapter 9 (cont.)

- How to fight/reduce burnout
  - Improve organizational culture (generative rather than bureaucratic or pathological)
  - Reduce or eliminate deployment pain
  - Hire and empower effective leaders
  - Invest in best practices wisely (including training and pilot projects for developments)
  - Adopt changes to improve organizational performance
- Continuous Delivery + Lean Practices => Less Deployment Pain + Less Burnout



WEB #6: The Thermocline of Truth (2008) [Link]

The Thermocline of Truth (cont.)

- A line drawn across the organizational chart that represents a barrier to accurate information regarding the project's progress
  - Those below this level tend to know how well the project is actually going
  - Those above it tend to have a more optimistic (if unrealistic) view
- Why does it form?
  - Lack of true metrics (objective, automated, predictive) on project status
  - Excessive optimism on part of engineers
  - Self-protection on the part of managers going up the chain
  - Top management tends to reward good news and punish bad news

The Thermocline of Truth (cont.)

- Consequence: as the deadline draws near, the actual project status tends to move upward in the management chain
  - Hence the classic “slip the project schedule three weeks before delivery” pattern
- How to avoid it
  - Honesty and outspokenness on the part of engineers and managers
  - Rewarding that honesty
  - Upper management actively seeking out from lower levels realistic feedback on project
  - Avoiding the temptation of the “quick fix to ship”

- Quality of work and effort
- Project planning and execution
- Quality assurance and process
- Architecture
- Application performance
- Staffing
- Management principles
- Intellectual honesty

WEB #6: Anatomy of a Runaway IT project (2008) [Link]

- Septic code is why some large IT projects never go live
  - Some portion of the source code created to date is so bad and has such a negative impact on other code that relies upon it that the project will never stabilize
  - Only solution: cut that course code out of the project and throw it away; write brand new source code in its place.
  - Sometimes requires complete reboot of project from scratch
- Reasons:
  - Use of un- or underqualified software engineers and architects
  - Poor hiring techniques and bad management
  - Doing too much too quickly
  - Lack of conceptual unity (solid architecture)
  - Lack of effective software quality assurance

Web #6: Septic Code (2013) [link]

- Temptation: the appearance (illusion, really) of progress
  - Prototyping user interface
  - Use of third-party libraries, engines, utilities
  - Getting important modules to “80% completion” and then moving on
- Finishing that last 10-20% is where things drag on forever
  - All the hardest problems have been deferred to the end
  - Can find yourself in “solution deadlock” among remaining hard problems
- Solution: courage to actively identify and tackle hardest problems first
  - Initial progress will be slow, but you will be more likely to be able to predict completion

Web #6; Do not Defer the Difficult in IT Projects (Baseline, 2008) [Link]

FOR NEXT WEEK (03/02)

- By midnight on Saturday (02/28):
  - Latest team status report
  - Individually: watch next podcast (#7)
- By class on Monday (03/02):
  - Read Peopleware parts III and IV
  - Read Accelerate, chapter 10
  - Start working through Webster #7 (will cover on 03/16)
- Reminder: first demos on 03/16
- Reminder: midterm on 03/23