Surviving Complexity

CS 428 – WINTER 2017 – BRUCE F. WEBSTER

Intractable Issues in IT

- The Wetware Crisis: Human Factors
- Only a Few Brass Slugs: Process Factors
- The Acid of Dissemination: Knowledge Factors
- The Tarnished Age of Wireless: Technology Factors

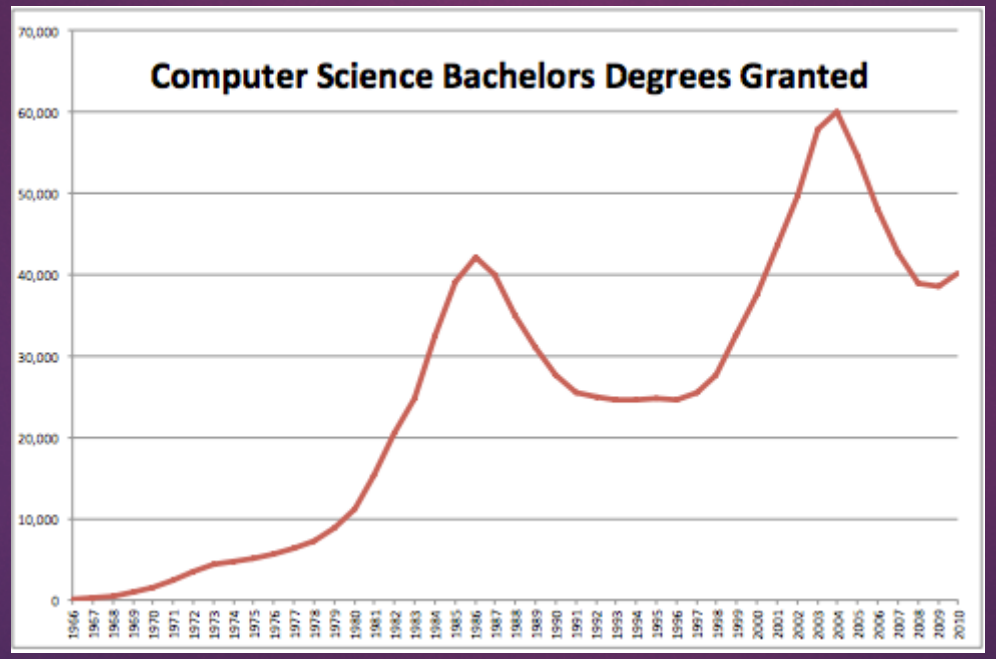
The Wetware Crisis: Human Factors

- Too Many Concertos, Not Enough Mozarts: lack of truly qualified IT workers
- All Our Sins Remembered: human nature and project failure
- Negotiations and Love Songs: game theory and project management
- Lies, Damned Lies, and Metrics: misunderstandings, mis-measurments, and misinformation

Too Many Concertos, Not Enough Mozarts: lack of qualified IT workers

- From CS 404 class 40 years ago: “If you knew which ones to choose, you could eliminate 50% of all data processing [IT] employees, and not only would no deadlines slip, many would be advanced.”
- Study after study over 50 years shows that the single biggest factor in IT project success is the quality of the people involved
  - Productivity can vary from 3:1 up to 28:1
- However, demand far outstrips supply and will for a long time (read: The Real Software Crisis (1996)]
- Needed: talent, education, professionalism, experience, skill (read: TEPES)
- Hard to screen/hire, hard to retain (read: Dead Sea Effect)
- Problem: we treat IT engineers as modules (read: Longest Yard)

Demand for software engineers is subject to economic conditions



All Our Sins Remembered: Personal Nature and Project Failure

- "Indeed, when asked why so many IT projects go wrong in spite of all we know, one could simply cite the seven deadly sins: avarice, sloth, envy, gluttony, wrath, lust, and pride. It is as good an answer as any and more accurate than most."

  – Testimony given by Bruce F. Webster before the Subcommittee on Government Management, Information, and Technology, United States House of Representatives, June 22, 1998.

- Human nature and individual choices account for the vast majority of IT project delays and failures; technology is rarely the issue
- Once two or more people are involved, software engineering becomes primarily a sociological activity, not a technological one

Negotiations and Love Songs: Game Theory and Project Management

- We think tools & process will solve our software development problems, but we're wrong
- We actually have an 3-player game, where each player has voiced and unvoiced goals, many of which are incompatible with those of others: Developers ("Geeks"), Management/Sales ("Suits"), and Customers ("Users")
- Geeks think they're playing "Overwatch", Suits think they're playing poker, and Users just want to balance their checkbooks & go home
- Each tends to view the other two groups with suspicious and disdain
- Without recognizing and understanding those very different viewponts, the groups will have a hard time aligning goals & effort

Lies, Damned Lies, and Metrics:

Misunderstanding, Mismeasurements & Misinformation

- Most organizations lack IT project metrics that are objective, automated, and useful and that can predict project completion or success (read Part I, Part II, Part III)
- Estimation models are highly sensitive to inputs (e.g., COCOMO II model with 2500:1 variance depending upon choices)
- Classic industry joke: first 90% of the project takes 90% of the schedule; remaining 10% takes other 90% of the schedule
- "Thermocline of Truth" syndrome in large organizations (read here)
- Deliberate underestimation to get project approved
- Mismanagement of expectations
- Managing to metrics almost always results in project distortion

Only a Few Brass Slugs: Process Factors

- Are We There Yet? The essential complexity of software development methodologies
- The End of Quality: can't ship with it, can't ship without it
- Time Out of Joint: mismatches in business, development, and technology cycles
- The Most Perilous Task: organizational change and process re-engineering

Are We There Yet?
Essential Complexity of SLDC

- Methodologies have come and gone for decades; none have really solved the core issues in software creation
- Fast, lightweight methods (XP/agile) tend to settle prematurely on limiting architectures
- Ponderous methodologies (waterfall-ish) tend to grind and grind in analysis and requirements
- Success tends to come because of the people involved, not the particular methodology chosen

The End of Quality:
Can't Ship With It, Can't Ship Without It

- Organizations tend to treat SQA as a second-class task assigned to second-class workers
- SQA is seen as "testing just before shipping" rather than a pervasive and continual set of iterative activities involving all project members and all internal and external deliverables, and one that should be as highly automated as possible
- Most projects failures I have reviewed in fact failed because of an inability to stabilize the number of defects in the system and "glide down" to completion with an ever-decreasing find/fix ratio
- Focus on quality – early and pervasive – usually leads to project success

Time Out of Joint:

Mismatches in Business, Development, and Technology Cycles

- Frequent conflicts and mismatches among:
  - Business objectives and funding
  - Market opportunities
  - IT development time
  - Technology availability, readiness, or obsolescence
- Different types of technology lifecycles (read here)
  - Firefly: promising tech that tantalizes but never quite arrives
  - Underdone: version N.0 tech release that isn't quite ready (vs N.0.1)
  - Conveyor belt: tech with clear end of life requiring transition off
  - Landfill: new tech that's dead on arrival

The Most Perilous Task:
Organizational Change and Process Re-engineering

- *And it ought to be remembered that there is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things. Because the innovator has for enemies all those who have done well under the old conditions and lukewarm defenders in those who may do well under the new.* (Machiavelli, *The Prince*, Chapter VI)
- People hate change; they want the new system to work "just like the old one, but better."
- "In architecting a new [software] program all the serious mistakes are made in the first day." (Spinrad, cited in Maier & Rechtin)
- "[There are] four questions that need to be answered as a self-consistent set if the system is to succeed economically; namely, who benefits? who pays? who provides? and, as appropriate, who loses?" (Maier & Rechtin)

The Acid of Dissemination: Knowledge Factors

- Storing Water in a Sieve: information security
- The Demolished Firm: privacy, security, and discovery
- The IP Civil Wars: who owns this slide's title?
- What Is Reality? And Can I Have a Copy? Blurring the line between bits and atoms

Storing Water in a Sieve: Information Security

- Even as we rush towards greater interconnectedness (including the 'Internet of Things'), our security practices are falling behind
- Security must be designed in, it cannot be added as an afterthought
- Increasing complexity both within software and among interacting software components & systems makes protection more and more difficult
- In all this, humans remain the weakest security link (e.g., Podesta phishing attack, password of "p@ssw0rd")

The Demolished Firm: Privacy, Security, and Discovery

- We leave digital traces of ourselves and our actions all over the place
- This is compounded within organizations such as businesses due to the constant nature and high volume of electronic interactions
- At the same time, businesses are now generating terabytes or even petabytes of documents, databases, spreadsheets and other information
- When legal issues arise (civil or criminal), these traces are subject to court-ordered discovery and must be preserved
- It usually falls upon IT workers within the firm to ensure the preservation and appropriate production

The IP Civil Wars: Who Owns This Slide's Title?

- Monetization of intellectual property (IP) has set off an IP land rush among organizations and individuals
- When you work as a software engineer, how much of your solutions belong to you as personal/professional skills vs. how much belong to your employer?
- The Gordian Knot of software patents has yet to be unraveled (or cut in two)
- What constitutes fair use?
- What are the pitfalls of open source software?

What Is Reality? And Can I Have a Copy? Blurring the Line between Bits and Atoms

- What is a ‘real’ photo?
- What is an ‘accurate’ database?
- What are the implications of 3-D printers?
- How much do you trust the readouts on your mobile devices?
  - Classic study from decades ago about sabotaged calculator

The Tarnished Age of Wireless: Technology Factors

- Cheap, Fast and Out of Control: keeping up with tech
- All Our Sins Encoded: software bloat, complexity, and inertia
- Commanding the Tide: legal and regulatory collision with technology
- Welcome to the Slow Zone: entrenchment of standards, format, code, and flaws

Cheap, Fast, and Out of Control: Keeping Up with Tech

- Moore's Law keeps marching on, in spite of predictions of its demise
- New combinations of technology loom on horizon
- Old technologies die and yet leave legacy requirements
- Disruptive technologies require four key aspects to supplant existing solutions:
  - Clearly superior, enough to overcome resistance to change
  - Has a path to sufficient standardization
  - Able to be used in parallel with existing solutions
  - Expands in utility and function/decrease in cost until it replaces existing solutions

All Our Sins Encoded:
Software Bloat, Complexity, and Inertia

- Many software systems succumb to the "arc of engineering" [read here] and begin to decline in reliability as they grow in functionality
- Key factors include:
  - The developer loses conceptual control of the system
  - Software rot sets in
  - The enhanced system finally outgrows its original foundation
  - Market or business needs shift beyond the product's fundamental design
  - The firm begins to add "blue sky"/"kitchen sink" enhancements
  - Backward compatibility is maintained at all costs

Commanding the Tide:
Legal and Regulatory Collision with Technology

- National and regional governments keep trying to regulate, control, prohibit, or mandate use of technology
  - Europe: “right to be forgotten”
  - US: DCMA, HIPAA, Sarbanes-Oxley
- Usually fail to recognize the complexity and adaptive nature of technology and often get things wrong
- On-going legal struggles with concept of ‘software patent’ illustrate this as well

Welcome to the Slow Zone:
Entrenchment of Standards, Format, Code and Flaws

- Success of technology makes it hard to replace or retire
- Article from 1996: “Windows Forever and Ever” [read here]
  - “As impossible as it might seem, Windows may still be dominant in 2025.” (Yes, but on desktops/laptops, not on mobile devices)
- Re-engineering/replacement of legacy systems in organizations is very difficult and prone to overruns/failures
- Data locked in proprietary file formats can be lost or may require special efforts to retrieve
- Even known defects in systems and applications may become ‘relied upon’

Conclusion

- Software engineering – and IT itself – is a chaotic collision of human, process, knowledge, and technological factors
- All indications – based on 40+ years of observation – is that things will get more chaotic, not less
- Awareness of these factors will help you better navigate through individual projects, specific employment positions, your professional career, and life in general
- Have fun, and be careful out there. ☺