CS 428
WEEK #6 READINGS

Fall 2019, Week #6

Bruce F. Webster

- Brooks: conceptual integrity is *the* most important consideration in system design (I agree)
- Simplicity, straightforwardness, unity of design are necessary
- The design must proceed from one mind or a very small number of agreeing resonant minds
- The conceptual integrity of a system determines its ease of use
- A consistent architecture enhances the creative style of implementers
- A well-thought-out architecture increases the robustness and adaptability of the resulting software system

MMM CH 4: ARISTOCRACY, DEMOCRACY, AND SYSTEM DESIGN

- Key goal: loosely-coupled architecture
  - Standard mantra: tight cohesion within modules, loose coupling among modules
  - Goal: we can do most of our testing without requiring an integrated environment
  - Goal: we can deploy our application independent of other applications/services it relies upon
- Expanded list to evaluate for loosely-coupled architecture
  - Make large-scale app design changes w/o permission of someone outside of team
  - Make large-scale app design changes w/o depending upon or creating work for other teams
  - Complete work w/o communicating/coordinating with people outside of team
  - Deploy/release app on demand, regardless of external dependencies
  - Do most testing on demand w/o requiring an integrated test environment
  - Perform deployments during business hours with only negligible downtime

ACCELERATE CHAPTER 5: ARCHITECTURE

CHAPTER 5: ARCHITECTURE (CONT.)

- Conway's Law (again!): organizations should evolve their teams and organizational structure to achieve the desired (loosely-coupled) architecture
  - Teams should be able to get work done (design through deployment) without requiring high-bandwidth communications between teams
- A loosely-coupled architecture enables organizational scaling (inverts Brooks' Law)
- Teams should be allowed to choose their own tools (within reason)
- Architects should focus on engineers and outcomes, not tools or technologies
  - "What tools or technologies you use is irrelevant if the people who must use them hate using them, or if they don't achieve the outcomes and enable the behaviors we care out."

- Problem: costs and difficulties of maintaining existing systems (50% to 80%)
  - Maintenance often used for entry-level personnel and old-timers
  - Software entropy sets in
- Possible solution: appoint a maintenance architect
  - Learn (and document) essential architecture of all production systems
  - Review all proposed changes to any given system (bug fix, enhancement, replacement)
  - Issue 'environmental impact statement' on consequences of such proposals
  - Oversee actual work on existing systems
  - Great training to become a chief software architect and/or CTO

WEB #4: "CONTROLLING IT COSTS: USING A MAINTENANCE ARCHITECT" (BASELINE, 2008) [LINK]

- (Co-authored with Ruby Raley, long-time colleague and friend)
- Many corporate organizational practices originated with the rise of the Industrial Age and into early 20th century corporations
- They bear little resemblance to what is needed to effectively recruit and manage IT workers in the 21st century
- Suggested idea: approach IT recruitment, training, and management from a sports team perspective rather than a manufacturing perspective
- Offered as a thought-experiment to reconsider current approaches
- We used football since Ruby & I know that game better than others, but remember these are just analogies

WEB #4: "THE LONGEST YARD: REORGANIZING IT FOR SUCCESS" (CUTTER IT JOURNAL, 2006) [LINK]

PERSONNEL: RECRUITING

- IT recruiting often focuses on checkbox items, puzzle solving, # of bodies
- Sports teams focus on finding, evaluating, recruiting small number of highly talented professionals to fill specific needs of the team
- How might this look in IT?
  - Recruit based on evaluation, recommendation and reputation
  - Focus on TEPES: talent, education, professionalism, experience, skills
  - Work within a team size limit and an overall salary cap
- Don't hire just for the sake of hiring someone
- Avoid overpaying new hires and underpaying proven performers

- Quarterback: chief software architect
  - Most visible player and directs efforts
  - Needs to be skilled at her/his position, but not at all positions
- Mutually-supporting team members with a common goal
  - Great quarterback can compensate some for a weaker team, and vice versa, but you really need a great quarterback and a great team
  - Team achievements need to be met with team recognition and team rewards
  - Team goals need to support and meet individual goals

PERSONNEL: ORGANIZING THE TEAM

COACHING

- We spend lot of money on IT engineers but very little on improving their skills, education, and performance via active coaching
  - Managers do not necessarily see themselves as coaches
  - Organizations usually lack ‘specialty’ coaches
- Likewise, sports teams have philosophies (overall approach) and playbooks (specific approaches for specific situations) – IT organizations usually don’t
  - Have seen successful use of “team rules” in IT organizations
    - Cf. Agile Manifesto and Twelve Principles of Agile Development
  - Playbooks can include development heuristics and maxims, design patterns, language standards and guidelines

- Adaptive (West Coast offense: fast moving, pass-heavy)
  - Business drivers require rapid development and close interaction with end users
  - Business drivers change on a rapid basis (due to competition
  - Fast, tight development cycle is required.
- Predictive (East Coast offense: “three yards and a cloud of dust”)
  - Business drivers and technological considerations require a significant investment in analysis, architecture, and design up front
- Iterative (Balanced offense: even pass/run mix)
  - Up-front investment in analysis and architecture required due to the scale and complexity of the systems under development
  - But can’t wait months or years for initial deployment

PERFORMANCE: OFFENSE = DEVELOPMENT

- Adaptive (Blitz defense)
  - Agile approach: test-before-code, pair-programming, strong built-in testing, etc.
  - Ensure zero defects before the code is ever released to higher-level integration and testing
- Adaptive/Predictive (Man-to-Man defense)
  - Integration, interface, and end-to-end testing, configuration management
  - Ensure that complete scenarios and uses cases can be carried out
- Predictive/Adaptive (Zone defense)
  - category-based testing, such as performance and stress testing, computational verification, scheduling testing, and regulatory testing, along with classic defect and change control management
  - ensure applications meet system and business requirements
- Predictive (Prevent defense)
  - compatibility/parallel testing, restart recovery testing, user acceptance testing, and production readiness testing; and performing release management
  - ensure that all stakeholders agree that it is both safe and desirable for the new system to go into production

PERFORMANCE: DEFENSE = QUALITY ASSURANCE

- Often inherent conflicts between business and IT goals
- Key Business goals
  - B1 — provide required and sufficient functionality to allow the firm to operate and compete on a level playing field
  - B2 — provide superior or unique functionality to allow the firm to beat its competition
  - B3 — provide efficiencies in productivity to allow the firm to free up funds for investment, expansion, and/or profits
- Key IT Goals
  - IT1 — maintain or grow its existing systems (and staff) or their equivalent
  - IT2 — migrate off aging, obsolete, or defective technology
  - IT3 — keep the business side (including end users) happy, or at least off its collective back, while getting the funds necessary to accomplish IT1 and IT2
- Essential to call out and understand these goals when negotiating between business and IT

DEALING WITH THE FRONT OFFICE

IT'S NOT JUST ONE GAME – IT'S A SEASON

- Organizations too often take a very short-term view
  - Focus solely on the (next) release of a given software project/product
  - Too often will not care if they injure or drive away key players in the process
  - Short-term choices for "success" can lead to long-term failure for the team and the organization
- Similarly, you need to focus on retaining your best people
  - Appropriate reviews and raises
  - Offer benefits that keep them happy and loyal
    - Excellent tools and equipment
    - Paid attendance at seminars, conferences, training
    - Book budget
    - Side projects exploring new technologies and methodologies