Fall 2018

Bruce F. Webster

CS 428
Accelerate, Chs 1-6

Accelerate: Forsgren, Humble, Kim

- Data collected by surveying software development teams and organizations over 4 years (2013-2017), 2,000+ organizations, 23,000+ responses
- Goal: discover practices and factors that distinguish “high-performing” software organizations from the other
- Five categories of capabilities
  - Continuous delivery
  - Architecture
  - Product and process
  - Lean management and monitoring
  - Cultural

Chapter 1: Accelerate

- "High-performing organizations" must accelerate:
  - Delivery of goods and services "to delight their customers"
  - Engagement with market to detect and understand customer demand
  - Anticipation of compliance and regulatory changes
  - Responses to potential risks (security, market changes, etc.)
- Focus on key transformations based on actual results (evidence)
- Adopt DevOps (not a silver bullet, though)
  - Merging of the development and operations sides of an organization
  - Improving communication, visualizing work, eliminating bottlenecks, and continually automating and improving procedures
  - Culture that an engineering group creates which promotes the outcomes mentioned above

Chapter 2: Measuring Performance

- Focus on software delivery performance (build, test, deploy)
  - Lead time (=> less than one hour)
  - Deployment frequency (=> on demand/multiple times per day)
  - Mean time to restore (MTTR) (=> less than one hour)
  - Change fail percentage (=> 0-15%)
- Software delivery performance impacts
  - Organizational performance (profitability, market share, productivity)
  - Non-commercial performance (quantity & quality of goods/services, operating efficiency, customer satisfaction, achieving organization/mission goals)
- Bring software delivery in-house; buy COTS/SaaS for commodity functions

Chapter 3: Measuring and Changing Culture

- Westrum organizational culture models (see table next slide)
  - Pathological (power-oriented) – based on fear & threat – information withholding
  - Bureaucratic (rule-oriented) – based on group protection and turf battles
  - Generative (performance-oriented) – everything subordinated to good performance & goals
  - Impacts both software delivery performance and organizational performance
- Elements of "good information"
  - Provides needed answers
  - Timely (i.e., can be acted upon to make a difference)
  - Presented so that it can be understood and used
- Continuous delivery and lean management both impact (for good) organizational culture

Chapter 3 (cont.)

Pathological (power-oriented)	Bureaucratic (rule-oriented)	Generative (performance)
Low cooperation	Modest cooperation	High cooperation
Messengers “shot”	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novely crushed	Novelty leads to problems	Novelty implemented

Chapter 4: Technical Practices

- Major focus: continuous delivery (CD)
  - Goal: make changes safely, quickly, sustainably
- Key principles of CD:
  - Build quality in
  - Work in small batches
  - Use computers for repetitive tasks, humans to solve problems
  - Relentless pursue continuous improvement
  - *Everyone* is responsible
- Foundations of CD:
  - Comprehensive configuration management (fully automated)
  - Continuous integration (CI) – short-lived branches
  - Continuous testing (fully or highly automated)

Chapter 4: Technical Practices (cont.)

- Technologies/practices that support CD
  - Version control
  - Deployment automation
  - Continuous integration
  - Trunk-based development (merge branches back daily)
  - Test automation
  - Test data management
  - Shift left on security
  - Loosely coupled architecture
  - Empowered teams
  - Monitoring
  - Proactive notification

Chapter 4: Technical Practices (cont.)

- Observed results of adopting continuous delivery
  - Higher quality
  - Better organizational culture
  - Higher software delivery performance
  - Strong team identity
  - Lower levels of deployment pain
  - Reduced team burnout

Chapter 5: Architecture

- 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

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."

Chapter 6: Integrating Infosec

- "Shift left on security": design and build it into the software delivery process instead of making it a separate downstream phase (parallels SQA issue)
- "Rugged Manifesto" (selected items)
  - I recognize software has become a foundation of our modern world.
  - I recognize my code will be used in ways I cannot anticipate, in ways it was not designed for, and for longer than it was ever intended.
  - I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.
  - I refuse to be a source of vulnerability and weakness.
  - My code will support its mission.
  - My code can face these challenges and persist in spite of them.
  - I am rugged not because it is easy, but because it is necessary, and I am up for the challenge.

Assignments for next class (11/12)

- By midnight on Saturday (11/10)
  - Submit your latest status report
  - Watch one podcast
- By start of next class period (11/12):
  - Read *Accelerate*, chapters 7-11, 16
  - Finish Webster #7
  - Be ready with your demos
- Reminders
  - 11/19: Midterm