DevOps
It's About How We Work

Randy Shoup
@randyshoup
linkedin.com/in/randyshoup

Background

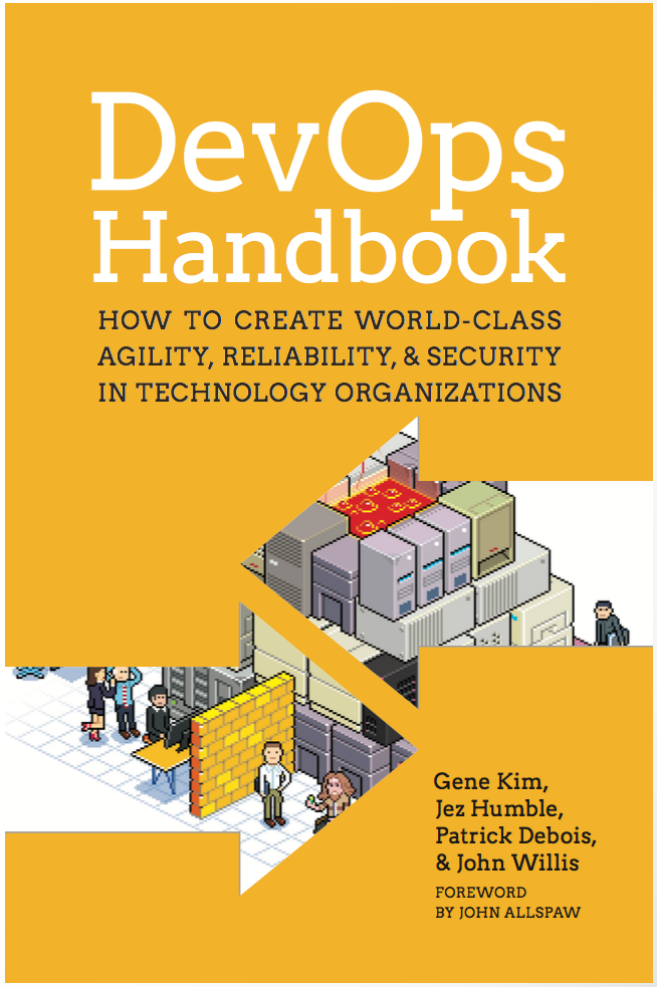
- VP Engineering at Stitch Fix
  - Using technology and data science to revolutionize clothing retail
- Consulting "CTO as a service"
  - Helping companies move fast at scale ☺
- Director of Engineering for Google App Engine
  - World's largest Platform-as-a-Service
- Chief Engineer at eBay
  - Evolving multiple generations of eBay's infrastructure

Time to Value

Speed vs. Stability?

High-Performing Organizations

- **Multiple deploys per day** vs. one per month
- **Commit to deploy in less than 1 hour** vs. one week
- **Recover from failure in less than 1 hour** vs. one day
- **Change failure rate of 0-15%** vs. 31-45%



High-Performing Organizations

- **Multiple deploys per day** vs. one per month

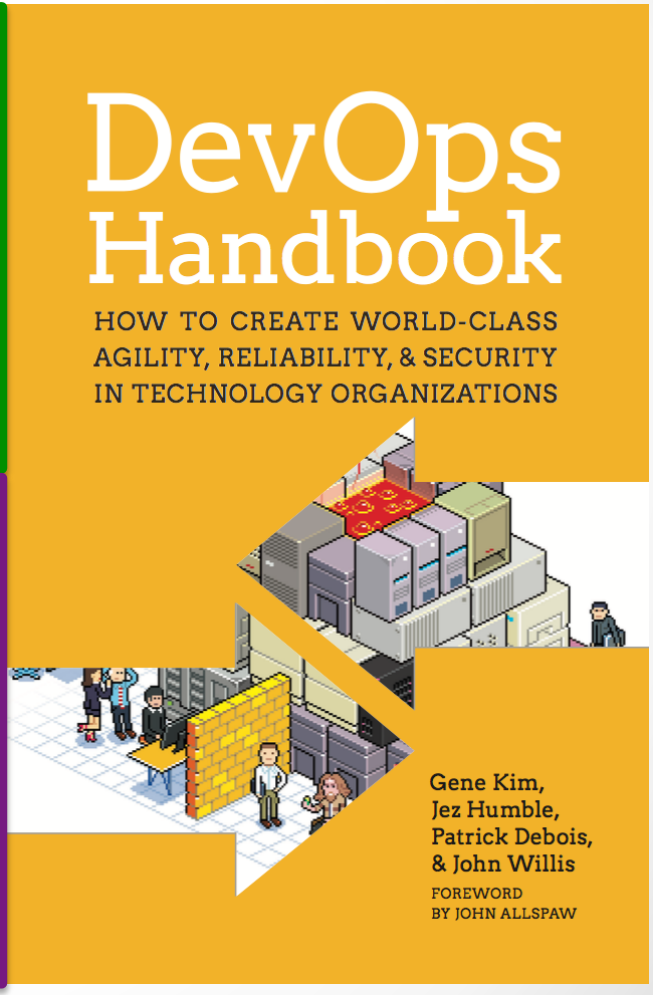
Speed

- **Commit to deploy in less than 1 hour** vs. one week

- **Recover from failure in less than 1 hour** vs. one day

Stability

- **Change failure rate of 0-15%** vs. 31-45%

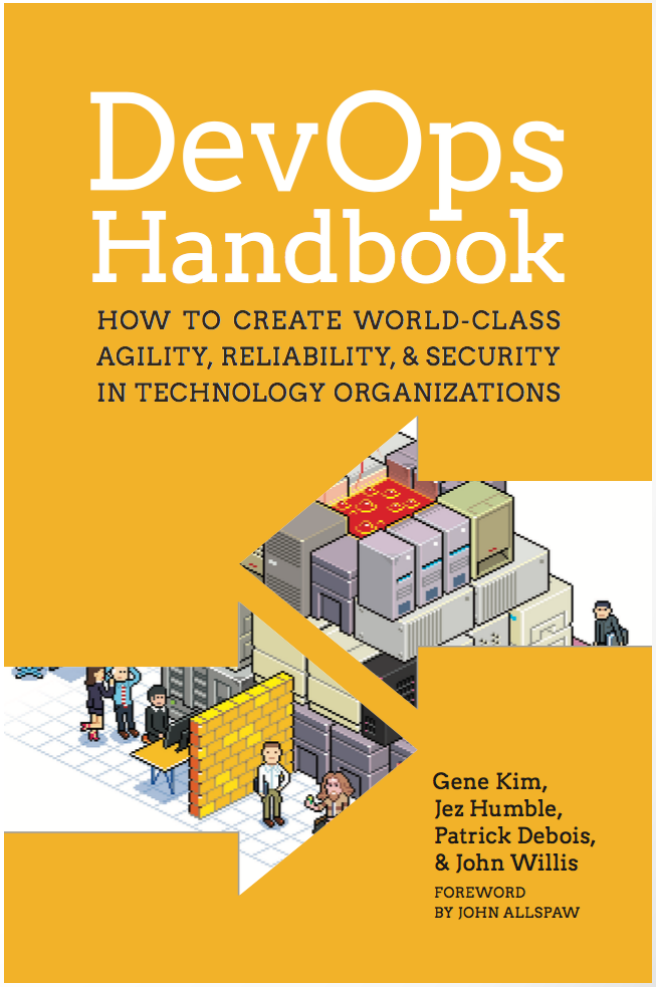


Speed AND Stability!

High-Performing Organizations

➔**2.5x more likely to exceed goals**

- Profitability
- Market share
- Productivity



DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

Conway's Law

- Organization determines architecture
  - Design of a system will be a reflection of the communication paths within the organization
- Modular system requires modular organization
  - Small, independent teams lead to more flexible, composable systems
  - Larger, interdependent teams lead to larger systems
- We can engineer the system we want by engineering the organization

Small "Service" Teams

- Full-Stack, "2 Pizza" Teams
  - No team should be larger than can be fed by 2 large pizzas
  - Typically 4-6 people
  - All disciplines required for the team to function
- Aligned to Business Domains
  - Clear, well-defined area of responsibility
  - Single service or set of related services
  - Deep understanding of business problems
- Growth through "cellular mitosis"

Ideally, 80% of project work should be within a team boundary.

DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

What problem are you trying to solve?

"Building the wrong thing is the biggest waste in software development."

-- Mary and Tom Poppendieck,
Lean Software Development

What Problem Are You Trying to Solve?

- Focus on what is important for your business
- Problem might be solved without any technology at all
  - Redefine the problem
  - Change the business process
  - Continue manually before automating in an application

"A problem well-stated is a problem half-solved."

-- Charles Kettering, former head of research for General Motors

Buy, Not Build

- Use Cloud Infrastructure
  - Faster, cheaper, better than we can do ourselves
  - Stitch Fix has no owned physical infrastructure anywhere in the world
- Prefer Open Source
  - Kubernetes, Docker, Istio
  - MySQL, Postgres, Redis, Elastic Search
  - Machine learning models
  - Etc.
  - Usually better than the commercial alternatives (!)

Buy, Not Build

- Third-Party Services
  - Stitch Fix uses >50 third party services
  - Logging, monitoring, alerting
  - Project management, bug tracking
  - Payments, billing, fraud detection
  - Etc.

- Focus on your core competency
  - Consider third-party services for **everything else** (!)

Soon it will be just as common to run your own data center as it is to run your own electrical power generation.

Experimental Discipline

- State your hypothesis
  - What metrics do you expect to move and why
  - Understand your baseline
- Run a real A | B test
  - Sample size
  - Isolated treatment and control groups
  - No peeking or quitting early!
- Obsessively log and measure
  - Understand customer and system behavior
  - Understand why this experiment worked or did not
  - Develop insights for next experiment

eBay Machine-Learned Ranking

- Ranking function for search results
  - Which item should appear 1st, 10th, 100th, 1000th
  - Before: Small number of hand-tuned factors
  - Goal: Thousands of factors
- Incremental Experimentation
  - Predictive models: query->view, view->purchase, etc.
  - Hundreds of parallel A | B tests
  - Full year of steady, incremental improvements

➔ 2% increase in eBay revenue (~$120M / year)

eBay
Site Speed

- Reduce user-experienced latency for search results
- Iterative Process
  - Implement a potential improvement
  - Release to the site in an A | B test
  - Monitor metrics –time to first byte, time to click, click rate, purchase rate

➔ 2% increase in eBay revenue (~$120M / year)

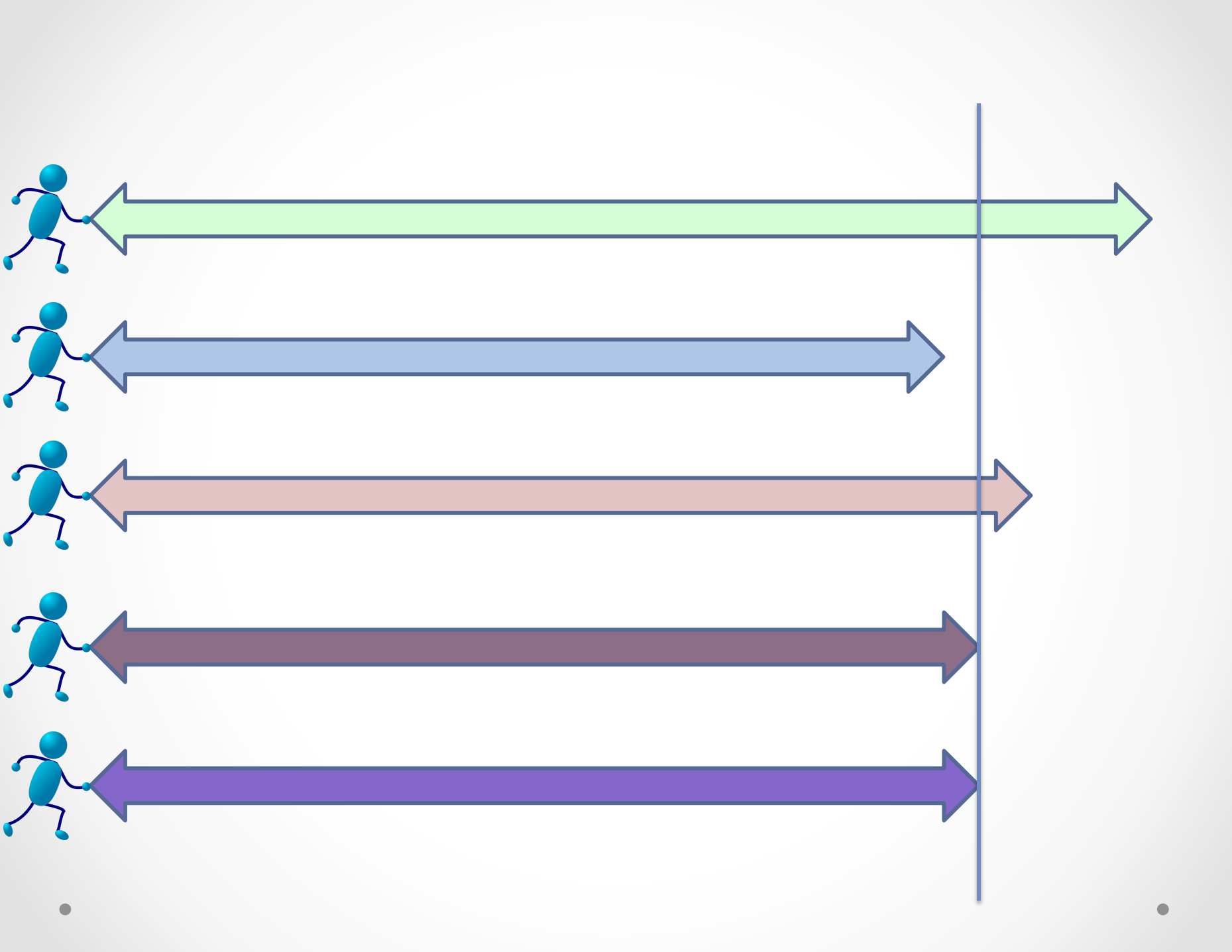
DevOps
How We Work

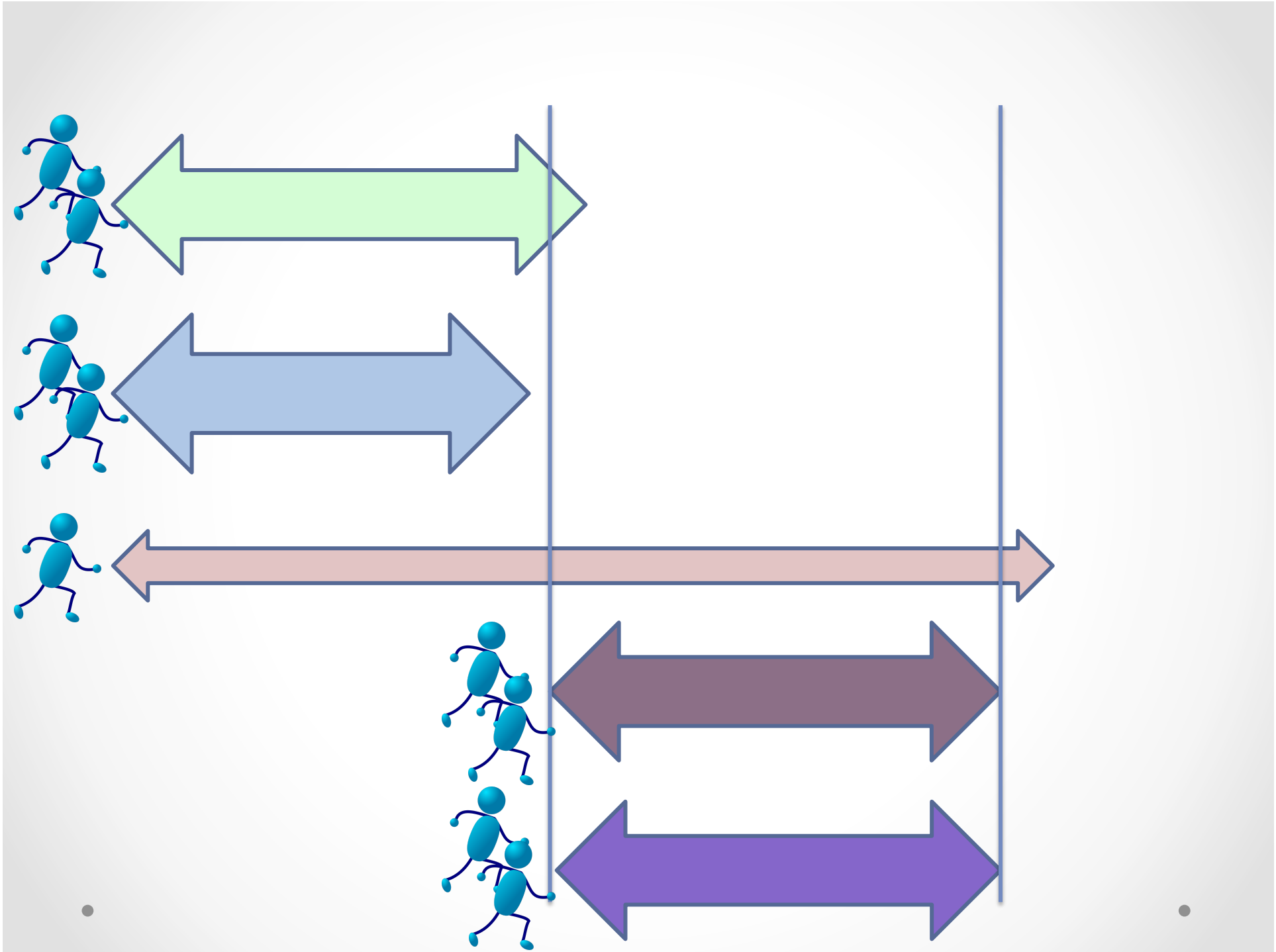
- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

Prioritization

- We always have more to do than resources to do it
- Scarce resources require prioritization
  - Opportunity cost -- deciding to do X means deciding not to do Y
  - Every decision is a tradeoff
- Priority ← Return on Investment
  - Business Value / Effort

Fewer Things, More Done





Fewer Things, More Done

- Deliver Full Value Earlier
  - Time Value of Money
  - Benefit now is worth more than benefit in the future
- Incremental Delivery
  - Deliver increments along the way instead of everything at the end
- Tasks often take less time
  - Multiple engineers can unblock one another

“When you solve problem one, problem two gets a promotion.”

DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

Quality Discipline

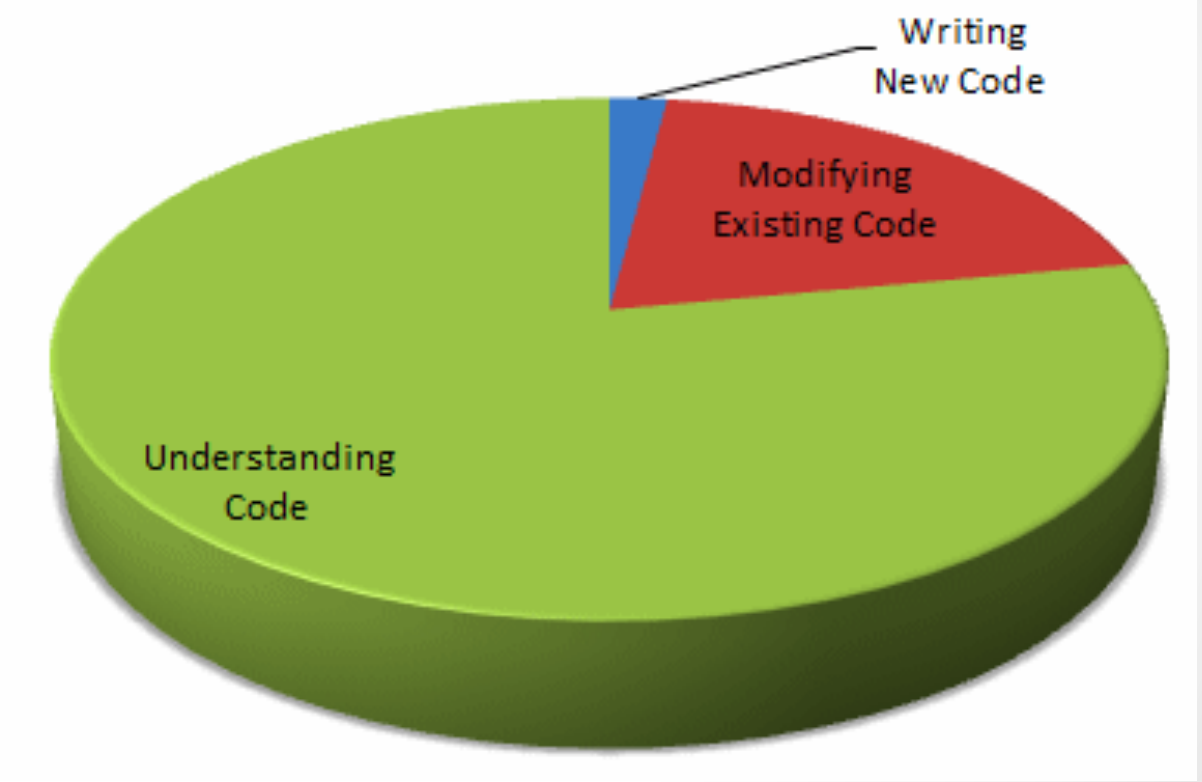
- Quality and Reliability are "Priority-0 features"
  - Equally important to users as product features and engaging user experience
- Developers responsible for
  - Features
  - Quality
  - Performance
  - Reliability
  - Manageability

Test-Driven Development

- Tests make better code
  - Confidence to break things
  - Courage to refactor mercilessly
- Tests make better systems
  - Catch bugs earlier, fail faster

Optimizing Developer Effort

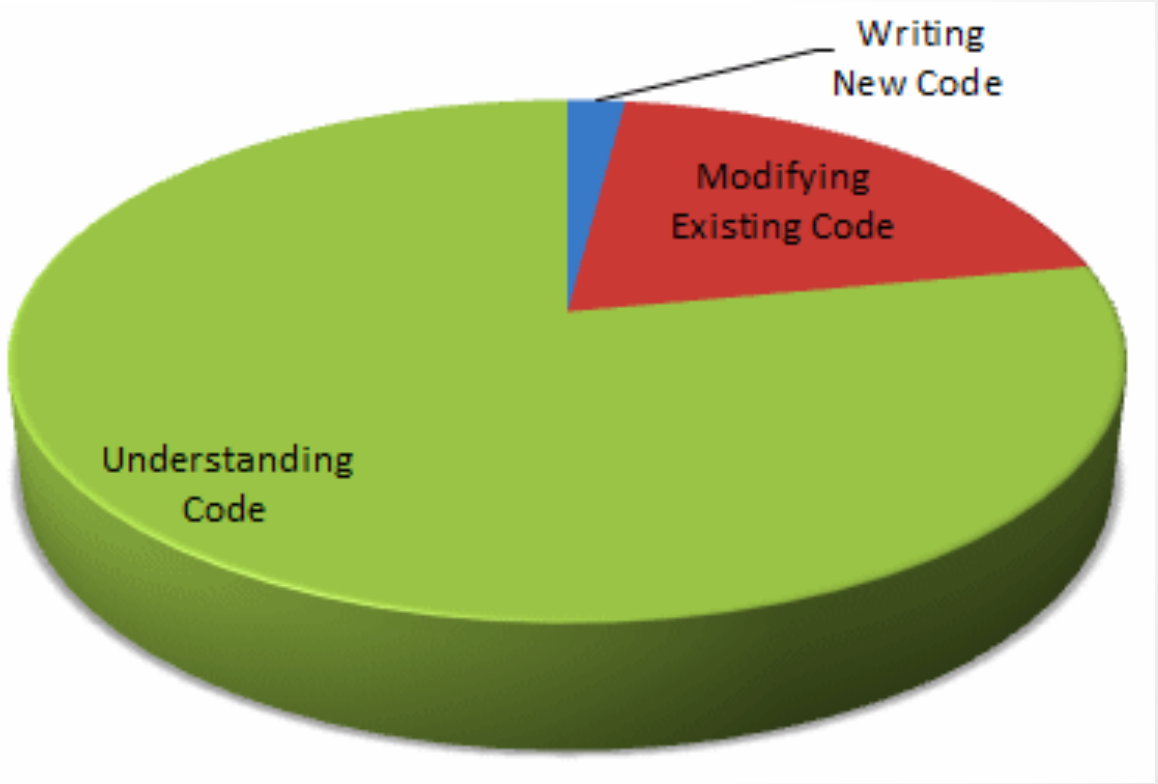
- 75% reading existing code
- 20% modifying existing code
- 5% writing new code



https://blogs.msdn.microsoft.com/peterhal/2006/01/04/what-do-programmers-really-do-anyway-aka-part-2-of-the-yardstick-saga/

Optimizing Developer Effort

- 75% reading existing code
- 20% modifying existing code
- 5% writing new code



https://blogs.msdn.microsoft.com/peterhal/2006/01/04/what-do-programmers-really-do-anyway-aka-part-2-of-the-yardstick-saga/

"We don't have time to do it right!"

"Do you have time to do it **twice**?"

The more constrained you are on time or resources, the more important it is to build it right the first time.

Build It Right (Enough) The First Time

- Build one great thing instead of two half-finished things
- Right ≠ Perfect (80 / 20 Rule)
- ➔ Basically no bug tracking system (!)
  - Bugs are fixed as they come up
  - Backlog contains features we want to build
  - Backlog contains technical debt we want to repay

DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

You Build It, You Run It.

-- Werner Vogels

End-to-End Ownership

- All disciplines required for the team's function
  - Design
  - Development
  - Quality and Performance
  - Maintenance
  - Operations
- Teams take long-term ownership
  - Team owns service from design to deployment to retirement
  - No separate maintenance or sustaining engineering team

Continuous Delivery

- Repeatable Deployment Pipeline
  - Low-risk, push-button deployment
  - Rapid release cadence
  - Rapid rollback and recovery
- Most applications deployed multiple times per day
- More solid systems
  - Release smaller units of work
  - Smaller changes to roll back or roll forward
  - Faster to repair, easier to understand, simpler to diagnose

Blameless Post-Mortems

- Post-mortem After Every Incident
  - Document exactly what happened
  - What went right
  - What went wrong

- Open and Honest Discussion
  - What contributed to the incident?
  - What could we have done better?

➔ Engineers compete to take personal responsibility (!)

“Finally we can prioritize fixing that broken system!”

Blameless Post-Mortems

- Action Items
  - How will we change process, technology, documentation, etc.
  - How could we have automated the problems away?
  - How could we have diagnosed more quickly?
  - How could we have restored service more quickly?
- Follow up (!)

Failure is not falling down, but refusing to get back up.

-- Theodore Roosevelt

DevOps
How We Work

- Organizing for DevOps
- What to Build / What NOT to Build
- When to Build
- How to Build
- Delivering and Operating

High-Performing Organizations

➔2.5x more likely to exceed business goals

- Profitability
- Market share
- Productivity

https://puppet.com/resources/whitepaper/state-of-devops-report

Time to Value

Merci vielmal!

- @randyshoup
- linkedin.com/in/randyshoup

DevOps Resources

- Books
  - *The Phoenix Project*, 2013
  - *The DevOps Handbook*, 2015
  - *Making Work Visible*, 2017
  - *Leading the Transformation*, 2015
  - *Continuous Delivery*, 2010
  - *Lean Software Development*, 2003
- Inspirations
  - *The Goal*, 1984
  - *Toyota Production System*, 1978
  - *Toyota Kata*, 2009
- Conferences
  - DevOpsDays (*everywhere*; Zürich, May 2-3 2018)
  - DevOps Enterprise Summit (London, June 25-26 2018)
- Podcasts
  - DevOps Café
  - Arrested DevOps