Chapter 1 - Accelerate

BOOK SUMMARY

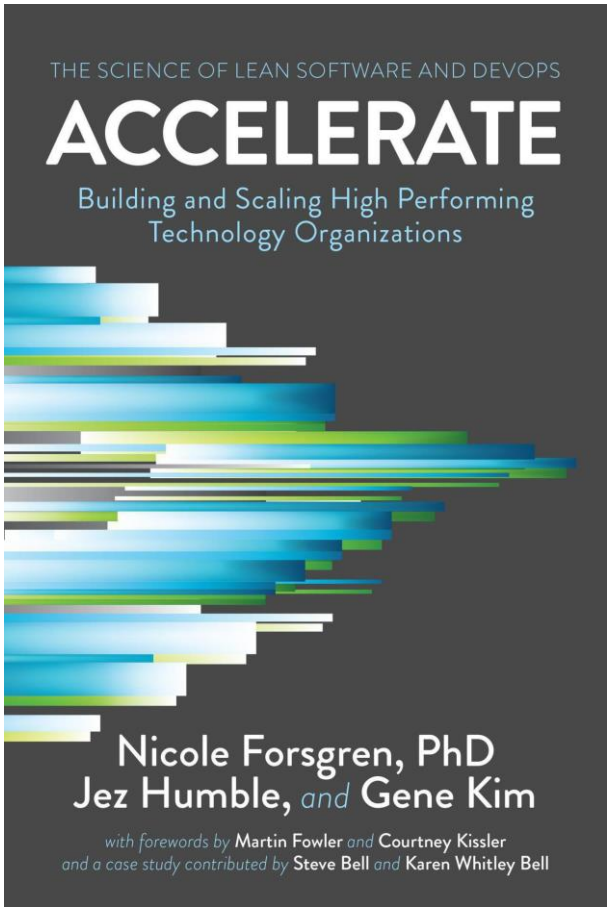
Key Takeaway

Small Teams, Short Cycles with Feedback
Leads to
Delighted Customers and Rapid Value

Survey Based Results
- 23,000 Results
- 2,000 Organizations
- Large and Small Companies

Software and Technology
Are therefore
Key Business Differentiator

Focus on Capabilities, not Maturity, by:
- Measure for Capabilities
- Continually Improve and Progress
- Maturity can vary between Teams
- Capability Model allows Team to increase their Skills
- 24 Key Capabilities in 5 Categories to improve Performance



DevOps

High Performers Know it is

NOT Quality vs Speed

But

Quality Enables Speed

Chapter 2 – Measuring Performance

Developers should solve Business Problems with Minimal Code
- But that is understandable

Velocity is a Capability Planning Tool
- But its Relative
- If Pushed, it gets Gamed

Utilization only Works until it Doesn't
- Approach 100% -> Lead Time becomes Infinite

Measuring Software Delivery Performance
- Globally focused
- Focus on Outcome not Output

Software Delivery Performance

1. Delivery Lead Time
2. Deployment Frequency
3. Time to Restore Service
4. Change Rate Failure

Lead Time

- From Lean Theory
- Time from Customer making Request to Request being Satisfied
- 2 Parts
    1. Time to Design and Validate (Hard to Measure)
    2. Time to Deliver Feature (Implement, Test and Deliver)
        - Delivery Lead Time = Code committed to Code in Production
- Timeframes: < 1 hr, < 1 day, < 1 wk, < 1 mo, < 6 mos, > 6 mos

Restore Service

- From Lean Theory
- Also Mean Time To Restore (MTTR)
- Similar to Time between Failures
- How quickly can a service be restored
    - Timeframes: < 1 hr, < 1 day, < 1 wk, < 1 mo, < 6 mos, > 6 mos

	Cohort Results		
	High	**Medium**	**Low**
Leadtime to Change	< 1 hour	> 1 week < 1 month	> 1 month < 6 months
Deployment Frequency	On Demand	> 1 week < 1 month	> 1 month < 6 months
Restore Service	< 1 hour	< 1 day	> 1 day < 1 week
Change Failure Rate	0-15%	31-45%	31-45%

Deployment Frequency

- From Lean Theory
- Metric measured is Batch Size
- Reduce Batch Size
    - Reduces Cycle Times
    - Reduces Variability in Flow
    - Accelerates Feedback
    - Reduces Risk and Overhead
    - Improves Efficiency
    - Increases Motivation and Urgency
    - Reduces Cost and Schedule Growth
- How often is code deployed
    - Timeframes: On Demand, < 1 day, < 1 wk, < 1 mo, < 6 mos, > 6 mos

Change Failure Rate

- Percent of Changes in Production that Fail
- Includes
    - Degraded Service
    - Requires Remediation

Organizational Performance

- High Performers Twice as likely to exceed Organizational Goals
    - Including
        - Quality and Quantity
        - Efficiency
        - Customer Satisfaction
        - Mission Goals
    - Therefore a Competitive Advantage
- A/B Testing and Experimentation is Key
- Reason to NOT Outsource Software Development
- Keep Strategic Software
    - Wardley Mapping Method (2020)

Chapter 3 – Measuring and Changing Culture

Model and Measure Culture

- 3 Levels of Organizational Culture
  1. **Basic Assumptions**
     - Set over time
     - Based on Events and Relationships
  2. **Values**
     - Visible and can be discussed/debated
     - "Culture" is usually focused on Values
  3. **Artifacts**
     - Mission Statement
     - Creeds and Procedures
     - Heroes and Rituals

Typology of Organizational Culture

- From Ron Westrum (1988)

1. **Pathological**
   - Power Oriented
   - Fear Based
   - Information Hoarded
2. **Bureaucratic**
   - Rule Oriented
   - Department Based
   - Maintain "Turf"
3. **Generative**
   - Performance Oriented
   - Mission Focused

Organizational Culture predicts how information flows in company

Pathological	Bureaucratic	Generative
Low Cooperation	Modest Cooperation	High Cooperation
Messenger "Shot"	Messenger Neglected	Messenger Trained
Responsibility Shirked	Narrow Responsibility	Risks are Shared
Bridging Discouraged	Bridging Tolerated	Bridging Encouraged
Failure -> Scapegoating	Failure – Justice	Failure -> Inquiry
Novelty Crushed	Novelty causes Problems	Novelty Implemented

Characteristics of Good Information Flow

1. Provides Answers to Questions
2. Timely
3. Presented in a way that can be consumed

Survey Questions

1. Information is Actively Sought
2. Messengers are not punished when they Deliver news of failure or other bad news
3. Responsibilities are Shared
4. Cross Functional Collaboration is Encouraged and Rewarded
5. Failure causes Inquiry
6. New Ideas are welcomed
7. Failures are Treated Primarily as an Opportunity to Improve the System

Likert Scale

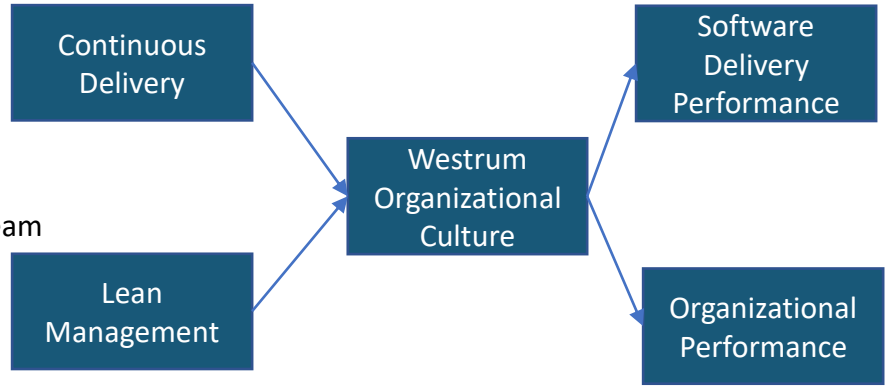
- Strongly Disagree
- Disagree
- Somewhat Disagree
- Neither Agree or Disagree
- Somewhat Agree
- Agree
- Strongly Agree

Generative Culture

- Has more Trust
- Emphasizes Mission over Personal Issues
- Minimizes Hierarchy while ensuring Fairness

Better Culture causes

- Better Information for Decision Making
- Poor Decisions are more easily Reversed
- Teams are more Open and Transparent
- Impacts Software Delivery
- Team members less important than how the team members interact
- Failure is seen as an Emergent property of a Complex System with no single cause



Chapter 4 – Technical Practices

Continuous Delivery

Move to Production Safety, Quickly and Sustainably

5 Key Principles

1. Build Quality In
   - Eliminate Inspection
   - Use Tools and People to find issues quickly
2. Work in Small Batches
   - Faster to Check'
   - Faster Feedback
3. Computers perform Repetitive Tasks; People solve Problems
4. Relentlessly Pursue Continuous Improvement
   - Strive to get better
5. Everyone is Responsible
   - Collaboration

Outcomes

1. Strong Identification with Organization
2. Higher Software Delivery Performance
3. Lower Change Failure Rates
4. Leads to Generative Culture (Performance Oriented)

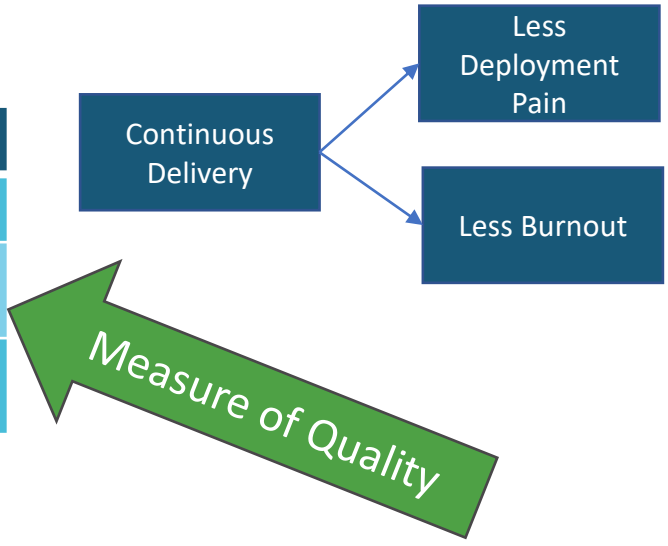
	High Performers	Low Performers
New Work	49%	38%
Unplanned Work (Rework)	21%	27%
Other Work (Meetings, etc)	30%	35%

Foundations

1. Comprehensive Configuration Management
   - Environments are Version Controlled
2. Continuous Integration (CI)
   - Short Branches
   - Commits lead to Tests
3. Continuous Testing
   - Part of Development
   - Automated

Additional Benefits

1. Decrease Deployment Pain and Team Burnout
2. Deployment during Business Hours
3. Developers can accept responsibility for Global Outcomes (Quality)



Continuous Development Practices

1. Version Control
2. Test Automation
   1. Need to be Reliable
   2. Developers responsible to create and maintain
   3. Every Commit should trigger Automated Tests
3. Test Data Management
   1. Able to acquire test data
4. Trunk-Based Development
   1. Fewer Branches (<3)
   2. Shorter Branches (< 1 day)
   3. Don't use Git Workflow
5. Information Security
   1. High Performing teams incorporate Information Security into their Delivery Process
   2. Increases Software Delivery Performance

Chapter 5 - Architecture

Loosely Coupled leads to High Performance

Focus on Deployability and Testability

- Local Testing
- Independent Releases
- Design Systems so that they are Loosely Coupled

To Achieve this:

- Create Cross Functional Teams that can Design, Develop, Test, Deploy and Operate System

Decouple Large Domains with

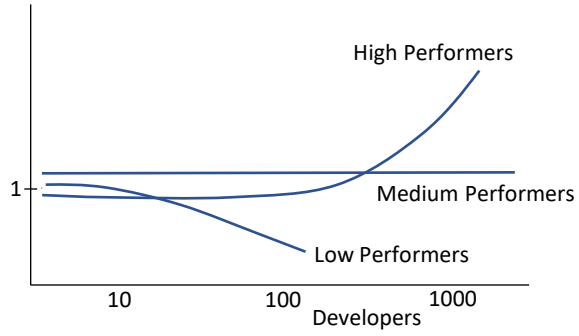
- Bounded Contexts
- APIs

Ensure communication bandwidth isn't overwhelmed

"Design Systems…are copies of communication structures of the organization" Conway 1968

Loosely Coupled Architecture Enables Scaling

- Enables more developers to work on system
- Measure is Deploys/Day/Developer



Allow Teams to Choose their Tools

Standardization Enables

- Reduce Complexity
- Ensure Skills in place
- Increase Purchasing Power
- Ensure correct Licensing

Downside

- Lack Flexibility
- Prevents Teams choosing Best Tool for their Needs
- Prevents Experimentation

Better to Let the Team Decide

Architects should Focus on

- Engineers
- Outcomes

Rather than

- Tools
- Technology

Enable Teams

To make Changes to their Product and Services

- Support Developers

Organizational Structure → Architectural Structure

Chapter 6 – Integrating InfoSec into the Delivery Lifecycle

For DevOps

- Bringing together
  - Development Teams
  - Operations Teams
- For System-Level Goals

Shift Left on Security

- Security Reviews on all Major Features
- Information Security integrated into Software Delivery Lifecycle
- Easy for Developers to do the Right Thing with InfoSec
- Give Developers the means to “build security in”
- High Performers spend 50% less time on Security Bugs

Rugged Manifesto

1. I am Rugged, and more importantly, my code is Rugged
2. I recognize that software is the Foundation of the Modern World
3. I recognize the Awesome Responsibility that comes with this Foundational Role
4. I recognize that my Code will be used in ways I cannot Anticipate, in ways not Designed, and for Longer than intended
5. I recognize that me code will be Attacked by Talented and Persistent Adversaries who Threaten our Physical, Economic and National Security
6. I recognize these – and I choose to be Rugged
7. I am Rugged because I Refuse to be a Source of Vulnerability or Weakness
8. I am Rugged because I assume that my Code will Support its Mission
9. I am Rugged because my Code can face these Challenges and Persist in spite of them
10. I am Rugged, Not because it is easy, but because it is necessary and I am up for the Challenge

Chapter 7 – Management Practices for Software

Lean Philosophy adapted to Software Development in "*Lean Software Development*" Poppendieck (2003)

Lean Management Practices

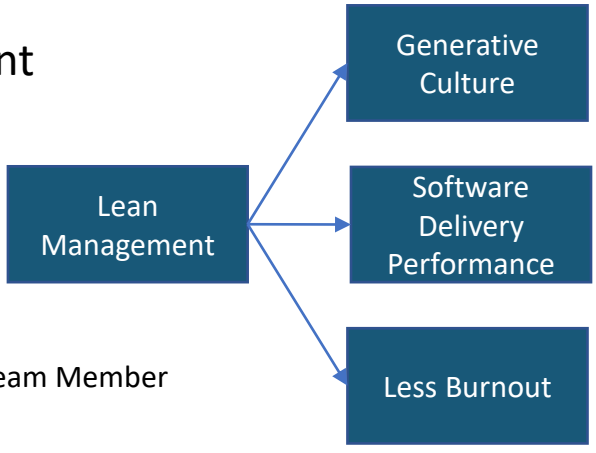
1. Limiting Work In Progress (WIP)
2. Visual Displays for:
   - Key Quality Metrics
   - Key Productivity Metrics
   - Current Status of Work
3. Using Performance and Infrastructure Data
   - To make Daily Business Decisions

Lean Management

- Limit Work-in-Progress (WIP)
- Visual Management
- Feedback from Production
- Lightweight Change Approval

Lean Management for Software Development

- Limit WIP to prevent over burdening
- Visual Displays enable High Quality Communication
- Implement lightweight Change Management Process
  - None or Peer Review works best
  - External (CAB) does not work
  - Pair programming or intra-Team review enough
- Overcome Segregation of Duties
  - Code should be reviewed AFTER commit by another Team Member
  - Deployment to Production should be fully Automated



Chapter 8 – Product Development

Faux Agile Practices

- Follow Common Practices
- Ignore Organizational Culture and Practices
- Grouping work into Larger Projects
- Not gathering Customer Feedback until the end

Focus on Lean Startup

- Frequently test Product with User Research
  - Its Design
  - Its Business Model
- Emphasis Experimentation into the product development process

- Work in small batches
- Make Flow of Work visible
- Gather and Implement Customer Feedback
- Team Experimentation

Lean Product Development Practices

1. Focus on working on Features that can be Completed within less than 1 week
   - Including MVPs
2. Ensure that the Team has good Visibility in the Flow of Work through to the Customer
3. Seek Customer Feedback and incorporate it into Product Design
4. Give Teams Authority to make changes to the Product

Virtuous Cycle

- Improved Software Delivery enables Better Lean Management Practices
- Working in Small Batches is key
  - Reduced cycle times leads to faster learning

Chapter 9 – Making Work Sustainable

Deployment Pain

Common Issues

- Fear of Pushing Code to Production
- Link to Software Delivery and Culture
- Making Deployments a "Black Box" to Developers doesn't help

Deployment Plan

1. Design Systems that can be easily Deployed to multiple Environments and can be updated independently
2. Ensure that the system can be reproduced (without Production Data) in an Automated fashion from Version Control
3. Build Intelligence into Application and Platform to make Deployments as simple as possible

Reduce Deployment Pain

- Extensive Test and Deployment Automation
- Continuous Integration, including Truck-Based Development
- Shift Left on Security
- Manage Test Data
- Use Loosely-Coupled Architecture
- Version Control Everything
- Enable Independent Teams

Therefore:

- Overcome Complex, Brittle Deployment Process

Burnout

Symptoms:

- Feeling exhausted, cynical and ineffective
- Little/no sense of accomplishment
- Feeling that Work negatively effects Life

To Prevent Burnout

1. Foster Respect and Emphasis Learning from Failures, not Blame
2. Communicate a strong sense of Purpose
3. Invest in Employee Development
4. Ask Employee What is Preventing them achieving their Objectives and help solve any issues
5. Give Employees Time and Space to Experiment and Learn

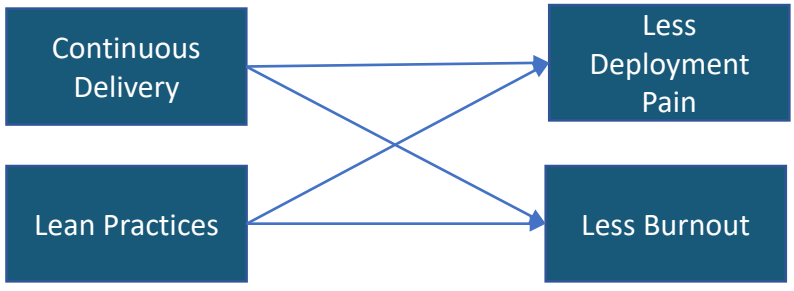
- Lean Management is giving Employees Time and Resources to Improve their Work
- Aligning Organizational and Personal Values reduces Burnout
  - Alignment → Employee Thrives

Factors that Predict Burnout

- Workload
- Lack of Control
- Insufficient Rewards
- Breakdown of Community
- Absence of Fairness
- Conflict of Values

Correlated Factors to Burnout

1. Organizational Culture (Sense of Purpose)
2. Deployment Pain
3. Effectiveness of Leadership
4. Investment in DevOps
5. Organizational Performance



Chapter 10 – Employee Satisfaction, Identity and Engagement

Employee Loyalty

Use:

- Net Promoter Score (NPS)

  "How likely is it that you would recommend our Company/Product/Service to a Friend or Colleague?"

NPS Score are Impacted by:

- Use of Customer Feedback to Design Product/Service
- Ability to Visualize Development Workflow
- Identify with Organizational Values and Goals

Employee Engagement Leads to:

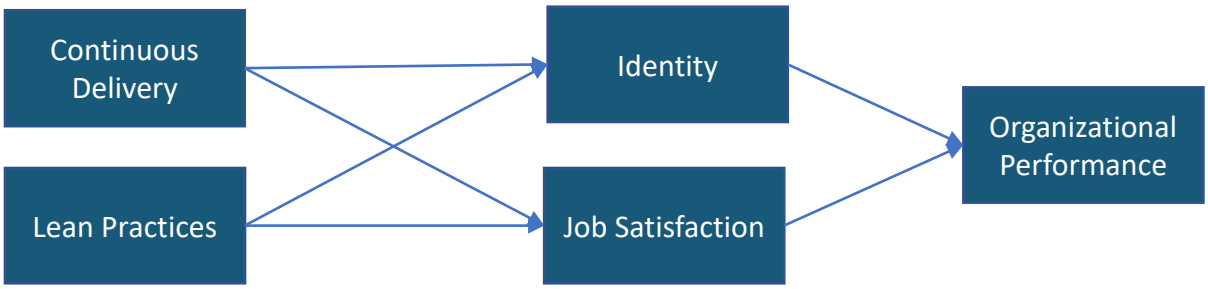
- Loyalty and Identity
- Reduce Burnout
- Drive Organizational Outcomes
  - Profitability and Market Share

9,10 = Promoters
7,8 = Passives
0-6 = Detractors

NPS = Promoters - Detractors

Can be performed on Organization or Team

Changing Culture and Identity



Questions to Measure Identity

- from Kankauhalli (2005)

1. I am glad I chose to work for this organization rather than another company
2. I talk of this organization to my friends as a great company to work for
3. I am willing to put a great deal of effort beyond what is normally expected to help my organization be successful
4. I find that my values and my organization's values are very similar
5. In general, the people employed by my organization are working towards the same goals
6. I feel me organization cares about me

Continuous Delivery + Experimental Approach = Better Products

Key Take Aways:

- Matching Values = Less Burnout
- Experimental Approach = Investment in People

Impact of DevOps on Job Satisfaction

- Having the tools and resources to do the work
- Work that uses your skills and abilities
- Automation is good because it has the computers do thing they are good at
- Experimentation lets people make decisions and use their skills

Continuous Improvement + Learning = Success

Chapter 11 – Leaders and Managers

Transformation Leadership

Leadership = Inspiring & Motivating Others

Leadership is needed to:

- Establish and Support Generative and High Trust Cultural Norms
- Creating technologies that enable Developer productivity and reliability
- Support Team Experimentation
- Achieve Strategic Alignment
- Set Tone and Reinforce Cultural Norms

Dimensions of Transformational Leadership

- From Rafferty and Griffin (2004)

1. **Vision**—Has a Clear Vision of where the organization is going and where it will be in 5 years
2. **Inspirational Communication**—Communicates in a way that inspires and motivates, even in uncertain and changing environments
3. **Intellectual Stimulation**—Challenges followers to think about problems in new ways
4. **Supportive Leadership**—Demonstrates care and consideration of follower's personal needs and feelings
5. **Personal Recognition**—Praises and acknowledges achievement of goals and improvements in work quality, personally complimenting others when they do outstanding work

- Similar to a Servant Leadership but with a focus on Results rather than Followers
- Strong correlation with eNPS scores

Role of Managers

Primary Roles

1. Connecting Strategic Objectives to the work of their Team
2. Create environment where Employees feel Safe
3. Investing in developing the Capabilities of their People
4. Removing Obstacles to Work (including Deployment Pain)

Enable Cross-Functional Collaboration by:

- Building trust within your counterparties on other teams—building Trust between Teams
- Encouraging practioners to move between Departments—Lateral movements are valuable
- Seek and Reward work that facilitates collaboration

Create a Climate of Learning by:

- Create a Training budget and advocating for it internally
- Ensure the resources of your team to engage in informal learning and the space to explore ideas
- Create opportunities and spaces to share information
- Encourage sharing with Demo Days and Forums

Make Effective use of Tools by:

- Make sure the team can choose their Tools—Commitment outweighs the benefits of standardization
- Make monitoring a Priority—Proactive monitoring leads to less Burnout and greater Job Satisfaction