Accelerate Book Summary

Accelerate – Building and scaling high performing technology organisations – Nicole Forsgren, Jez Humble and Gene Kim

Contents

Short Summary

The full title and author is: Accelerate – Building and scaling high performing technology organisations – Nicole Forsgren, Jez Humble and Gene Kim

My top 10 takeaways from this book are below.

1. There is a clear rationale for adopting DevOps:
    a. 46 times more frequent code deployment,
    b. 440 times faster lead time from commit to deploy
    c. 170 times faster mean time to recover from downtime
    d. 5 times lower change failure rate
2. Software delivery performance is measured by four metrics:
    a. Lead time
    b. deployment frequency
    c. mean time to restore (MTTR)
    d. change failure percentage
3. There are three types of culture (power based, rule based and performance based). Culture impacts both software delivery performance and organisational performance
4. Continuous delivery is built on: (1) Building quality in, (2) Working in small batches, (3) Computers perform repetitive tasks, people solve problems, (4) Relentlessly pursue continuous improvement and (5) Everyone is responsible.
5. The following things were seen to lead to continuous delivery:
    a. Version control
    b. Deployment automation
    c. Continuous integration
    d. Trunk based development - higher delivery performance was seen if the team developed off the trunk master rather than on long lived feature branches. GitHub

flow is good for open source where contributors check out code for long periods of time.

e. Test automation - testing can be done without an integrated environment. We can deploy or release independently from other applications / services.

f. Test data management

g. Shift left on security

h. Loosely coupled architecture

i. Empowered teams

j. Monitoring

k. Proactive notification

6. The rugged movement - “Rugged” describes software development organizations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software. → Rugged Software | “Rugged” describes software development organizations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software.

    a. I am rugged and, more importantly, my code is rugged.

    b. I recognize that software has become a foundation of our modern world.

    c. I recognize the awesome responsibility that comes with this foundational role.

    d. I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.

    e. I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.

    f. I recognize these things - and I choose to be rugged.

    g. I am rugged because I refuse to be a source of vulnerability or weakness.

    h. I am rugged because I assure my code will support its mission.

    i. I am rugged because my code can face these challenges and persist in spite of them.

    j. I am rugged, not because it is easy, but because it is necessary and I am up for the challenge.

7. Lean software management practices: (Limit WIP, Visual management, Feedback from production and lightweight change approvals) This leads to better culture, software delivery performance and less burnout.

8. Lead product development: Work in small batches, make flow of work visible, gather and implement customer feedback, team experimentation.

9. Making work sustainable

a. Reduce deployment pain: (1) Build systems that are designed to be deployed easily, (2) ensure that state of production systems can be reproduced, (3) Build intelligence into the application and the platform so that the deployment process can be as simple as possible.

b. Reduce burnout with org culture, reducing deployment pain, having effective leaders, investing in DevOps and driving org performance.

10. Transformational leadership including vision, inspirational comms, intellectual stimulation, supportive leadership and personal recognition leads to:

    a. Test automation

    b. Deployment automation

    c. Trunk based development

    d. Shift left of security

    e. Loosely coupled architecture

    f. Empowered teams

    g. Continuous integration

    h. Team experimentation

    i. Work in small batches

    j. Gather and implement customer feedback

Detailed Summary

Contents

Quick reference: Capabilities to drive improvement.

- Continuous delivery capabilities
    - Version control
    - Deployment automation
    - Continuous integration
    - Trunk based development.
    - Test automation
    - Test data management
    - Shift left on security.
    - Continuous delivery
- Architecture capabilities
    - Loosely coupled architecture.
    - Empowered teams
- Product ND PROCESS CAPABILITIES
    - Customer feedback
    - Value stream
    - Working in small batches
    - Team experimentation
- Lean management and monitoring capabilities.
    - Change approval process.
    - Monitoring
    - Proactive notification
    - WIP limits
    - Visualisation work
- Culture capabilities
    - Westrum organisation culture
    - Supporting learning
    - Collaboration among teams
    - Job satisfaction
    - Transformational leadership

1. Accelerate
    a. To remain competitive and excel, orgs must accelerate….
        i. Delivery of goods and services
        ii. Engagement with the market to detect and understand customer demand.
        iii. Respond to potential risks such as security threats or changes in the economy.
    b. Focus on capabilities not maturity.
    c. Evidence based transformations focus on key capabilities.
    d. Value of adopting DevOps
        i. 2017 found that high performers in DevOps have:
            1. 46 times more frequent code deployment
            2. 440 times faster lead time from commit to deploy.
            3. 170 times faster mean time to recover from downtime.
            4. 5 times lower change failure rate (15 as likely for a change to fail)
2. Measuring performance
    a. Flaws in previous attempts to measure performance.
    b. Measuring software delivery performance

i. Product design and development
    1. Create new products and services that solve customer problems using hypotheses driven delivery, modern UX and design thinking.
    2. Feature design and implementation may require work that has never been performed before
    3. Estimates are highly uncertain
    4. Outcomes are highly variable
ii. Product delivery (Build, test, deploy)
    1. Enable fast flow from development to production and reliable releases by standardising work and reducing variability and batch sizes
    2. Integration test and deployment must be performed continuously as quickly as possible
    3. Cycle times should be well known and predictable
    4. Outcomes should have low variability

c. There are four measures for software delivery performance
    i. Lead time
    ii. Deployment frequency
    iii. Mean time to restore (MTTR)
    iv. Change fail percentage
d. Software delivery performance for 2016 in table 1 and 2017 in table 2

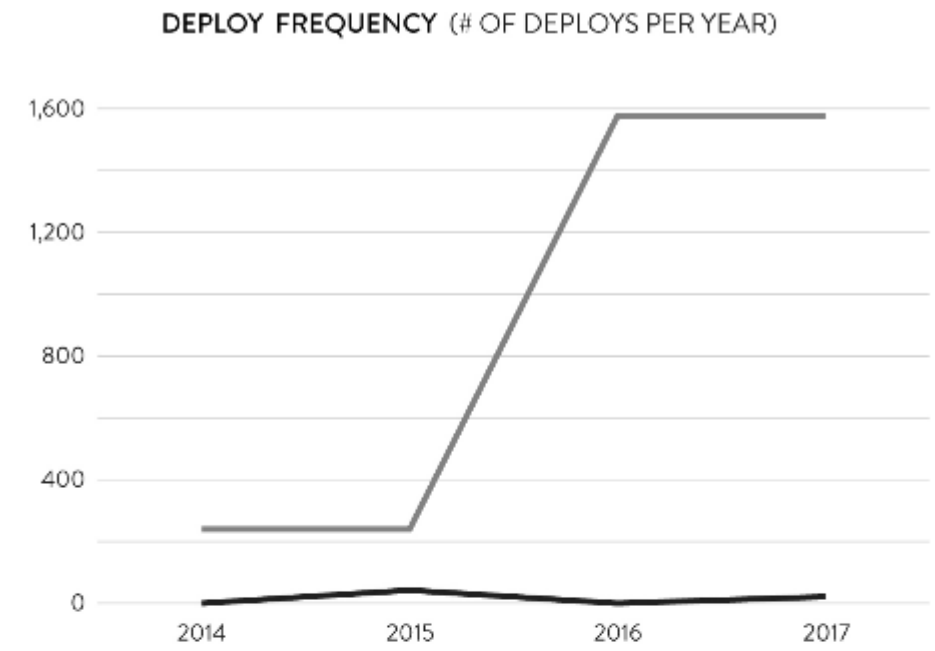
Table 2.2 Software. Delivery Performance for 2016

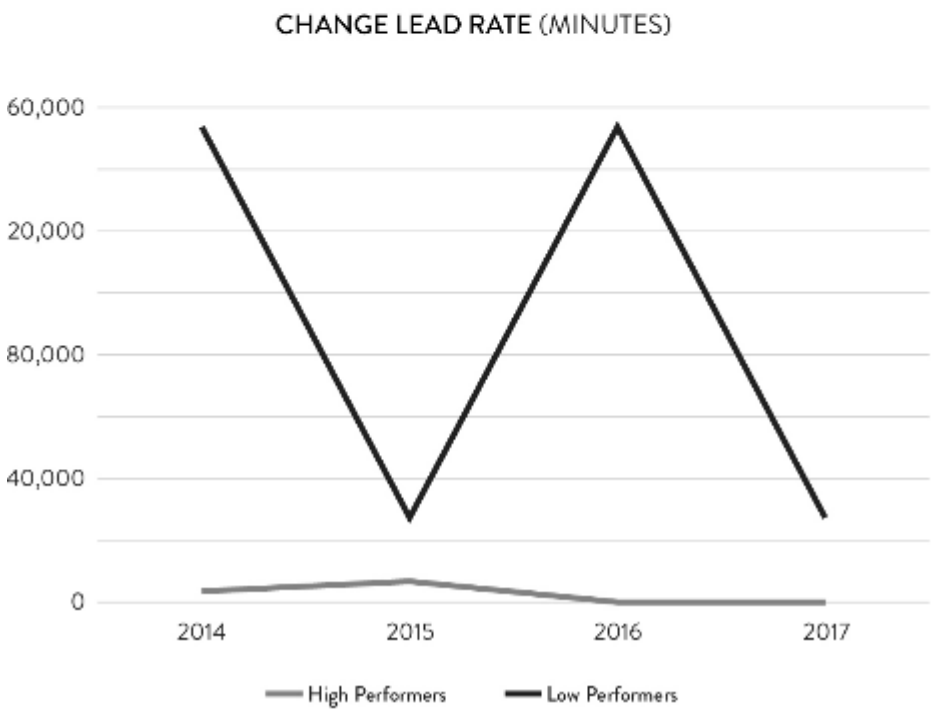
2016	High Performers	Medium Performers	Low Performers
Deployment Frequency	On demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every six months
Lead Time for Changes	Less than one hour	Between one week and one month	Between one month and six months
MTTR	Less than one hour	Less than one day	Less than one day*
Change Failure Rate	0-15%	31-45%	16-30%

Table 2.3 Software Delivery Performance for 2017

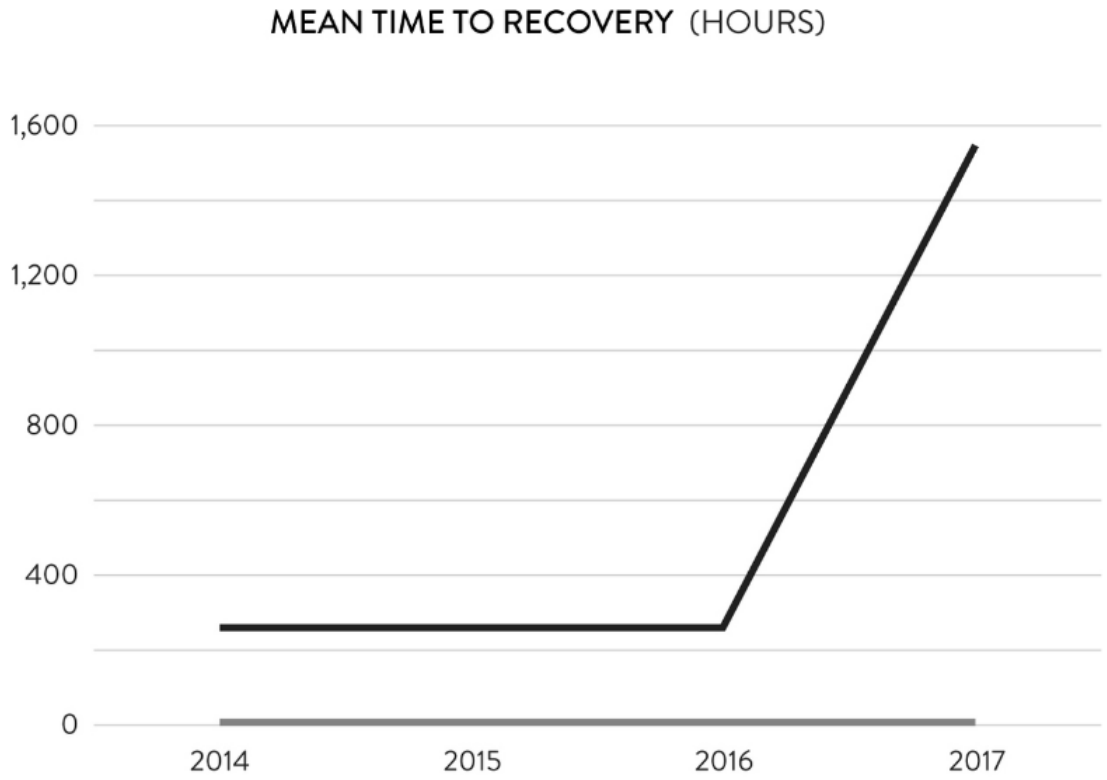
2017	High Performers	Medium Performers	Low Performers
Deployment Frequency	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month*
Lead Time for Changes	Less than one hour	Between one week and one month	Between one week and one month*
MTTR	Less than one hour	Less than one day	Between one day and one week
Change Failure Rate	0-15%	0-15%	31-45%

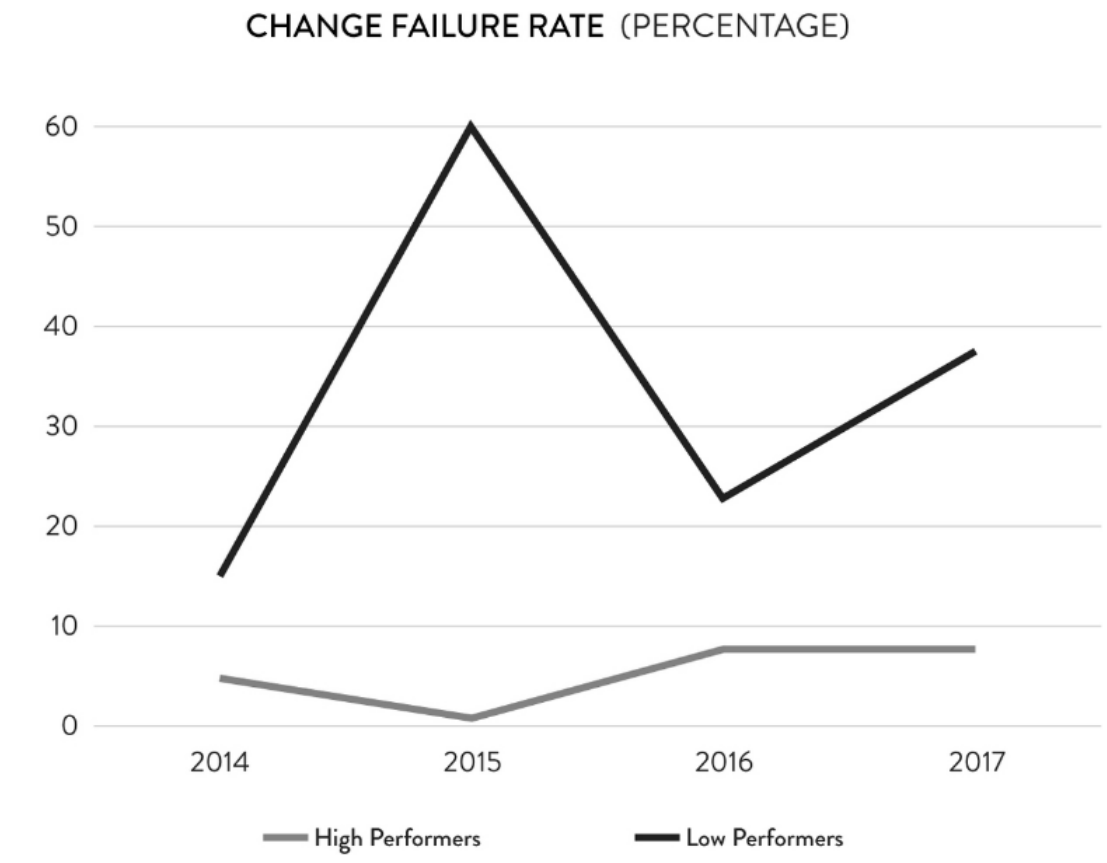
e. Deploy frequency and change lead rate





f. Mean time to recover and change failure rate


MEAN TIME TO RECOVERY (HOURS)
1,600
1,200
800
400
0
2014
2015
2016
2017


CHANGE FAILURE RATE (PERCENTAGE)
60
50
40
30
20
10
0
2014
2015
2016
2017
High Performers
Low Performers

g. In 2016 medium performers did worse than low performers. The hypothesis is that medium performers are doing business transformation and addressing legacy code and that this is the reason they are poorer performing than the low performers.
h. The thinking is that Software Delivery Performance drives both organisational performance and non-commercial performance
i. Driving change

3. Measuring and changing culture
    a. Modelling and measuring culture
    b. There are three types of organisational culture
        i. Pathological – power oriented
        ii. Bureaucratic – rule oriented
        iii. Generative – performance oriented
    c. Characteristics of good information are
        i. Provides answers to the questions that the receiver needs answered
        ii. It is timeline
        iii. It is presented in such a way that it can be effectively used by the received.
    d. Culture and its impact on DevOps performance

Table 3.1 Westrums Typology of Organizational Culture.

Pathological (Power-Oriented)	Bureaucratic (Rule-Oriented)	Generative (Performance-Oriented)
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
Novelty crushed	Novelty leads to problems	Novelty implemented

    a. Survey results
        a. Multiple surveys were created to ensure the results were correct.
        b. Tests were done for:
            i. Discriminant validity – making sure unrelated things are in fact unrelated
            ii. Convergent validity – making sure that related things are actually related
            iii. Reliability – making sure the items are read and interpreted similarly by those who take the survey. This is also referred to as internal consistency (The survey respondent should agree with the results that come out for his / her survey)
    b. What does westrum's organisation culture predict?
        a. Westrum's organisational culture leads to both software delivery performance and organisational performance
    c. How to change culture
        a. It turns out that both continuous delivery and lean management lead to westrum's organisational culture.

4. Technical Practices
    d. What is continuous delivery
        a. Build quality in
        b. Work in small batches
        c. Computers perform repetitive tasks; people solve problems
        d. Relentlessly pursue continuous improvement

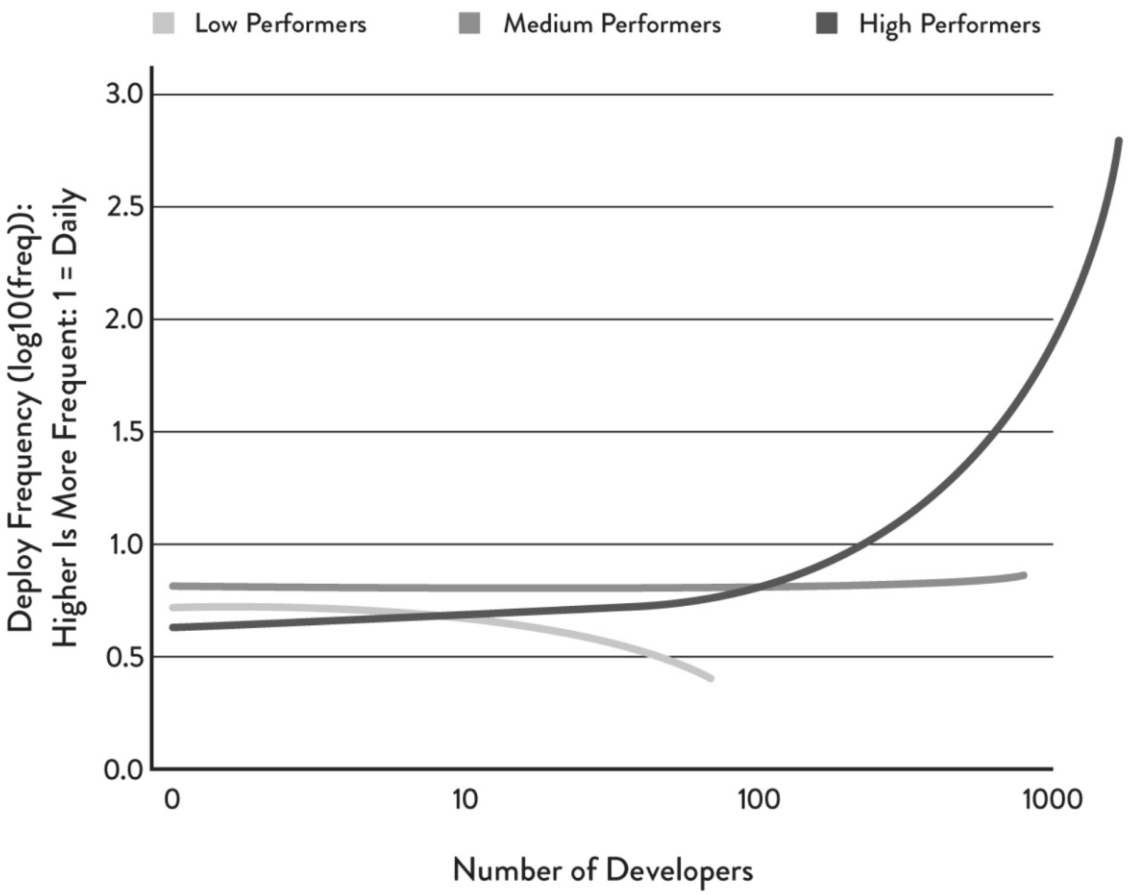
    e. Everyone is responsible
e. The following foundations must be implemented
    a. Comprehensive configuration management
    b. Continuous integration
    c. Continuous testing
f. The impact on continuous delivery
    a. The following capabilities were measured:
        i. Use of version control for application code, system configuration, application configuration and build configuration scripts
        ii. Comprehensive test automation that is reliable, easy to fix, runs regularly
        iii. Deployment automation
        iv. Continuous integration
        v. Shifting left on security: bringing security – and security teams – in process with software delivery rather than as a downstream phase
        vi. Using trunk based development as opposed to long lived feature branches
        vii. Effective test data management
    b. To measure the version control capability, we ask respondents to report on a Likert scale, the extent to which they agree or disagree with the following statements
        i. Our application code is in a version control system
        ii. Our system configurations are in a version control system
        iii. Our application configuration s are in a version control system
        iv. Our scripts for automating build and configuration are in a version control system.
    c. In addition two further capabilities were measured which had a strong and statistically significant impact on continuous delivery
        i. A loosely coupled well encapsulated architecture
        ii. Teams that can choose their own tools based on what is best for the users of those tools
    d. The following things were seen to lead to continuous delivery
        i. Version control
        ii. Deployment automation
        iii. Continuous integration
        iv. Trunk based development
        v. Test automation
        vi. Test data management
        vii. Shift left on security
        viii. Loosely coupled architecture
        ix. Empowered teams
        x. Monitoring
        xi. Proactive notification
    e. Teams that did well at continuous delivery achieved the following outcomes:
        i. Strong identification with the organisation you work for
        ii. Higher levels of software delivery performance
        iii. Lower change fail rates
        iv. A generative, performance-oriented culture

f. Continuous delivery leads to less work, westrum organisational culture, software delivery performance and identity
g. Organisational performance are driven by westrums organisational culture, software delivery performance and identity
h. Continuous delivery leads to less deployment pain and less burnout
i. The impact of continuous delivery on quality
    i. The following variables were tested to measure quality
        1. Quality and performance of applications, as perceived by those working on them
        2. Percentage of time spent on rework or unplanned work
        3. Percentage of time spent working on defects identified by end users
    ii. High performers spent
        1. 49% time on new work
        2. 30% on other work (meetings, routine maintenance)
        3. 21% on unplanned work or rework
    iii. Low performers spent
        1. 38% time on new work
        2. 35% time on other work (meetings, routine maintenance)
        3. 27% on unplanned work or rework
j. Continuous delivery practices what works and what doesn't
k. Version control – comprehensive version control and configuration management
l. Test automation – automate as much as possible and run the tests regularly. Test driven development should be used.
m. Test data management – require adequate test data to run the automated tests.
n. Trunk based development
    i. Higher delivery performance was seen if the team developed off the trunk master rather than on long lived feature branches.
    ii. GitHub flow workflow is suitable for open source projects whose contributors are not working on a project full time. This means the branches being worked on last for a longer amount of time.
o. Information security – to be incorporated into the delivery process.
p. Adopting continuous delivery – many benefits to this which are already described. One limiting factor is enterprise and application architecture.

g. Architecture
    a. Types of systems and delivery performance
        i. The following types of systems were looked at
            1. Greenfield
            2. Systems of engagement
            3. Systems of record
            4. Customer software developed by another company
            5. Customer soft developed in hose
            6. Packaged commercial off the shelf software
            7. Embedded software that runs on a manufactured hardware device

8. Non mainframe software that runs on servers operated by another company
9. Non mainframe software that runs on our own servers
10. Mainframe software

b. Focus on deploy ability and testability
    i. High performers say….
        1. We can do most of our testing without requiring an integrated environment
        2. We can and do deploy or release our application independently of other applications / services it depends on
c. Teams that can do continuous delivery successfully can:
    i. Make large scale changes to the design of their system without the permission of somebody outside the team
    ii. Make large scale changes to the design of their system without depending on other teams to make changes tin their systems or creating significant work for other teams
    iii. Complete their work without communicating and coordinating with people outside the team
    iv. Deployment and release their product r service on demand, regardless of other services it depends on
    v. Do most of their testing on demand without requiring an integrated the environment
    vi. Performance deployments during normal business hours with negligible downtime
d. Loosely coupled architecture enables scaling



e. Allow teams to choose their own tools which enables
    i. Reducing the complexity of the environment
    ii. Ensuring the necessary skills are in place to manage the technology throughout its lifecycle
    iii. Increasing purchasing power with vendors
    iv. Ensuring all technologies are correctly licensed
f. Architects should focus on engineers and outcomes not tools or technologies

h. Integrating infosec into the delivery lifecycle
    a. Shifting left on security

i. The rugged movement

a. ***I am rugged and, more importantly, my code is rugged.***
b. ***I recognize that software has become a foundation of our modern world.***
c. ***I recognize the awesome responsibility that comes with this foundational role.***

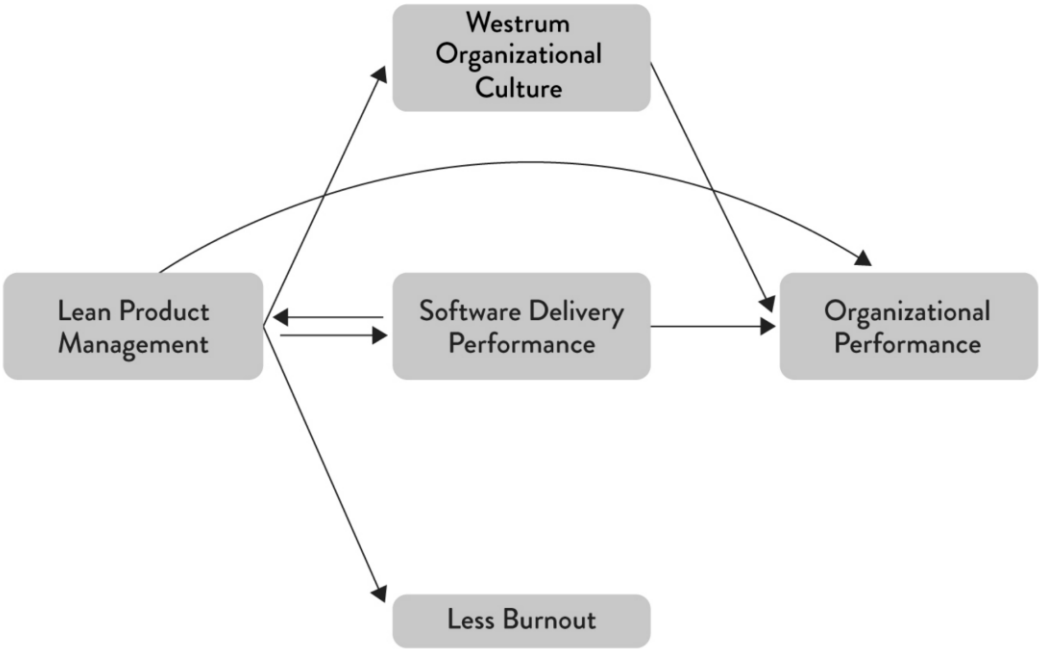
d. ***I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.***
e. ***I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.***
f. ***I recognize these things - and I choose to be rugged.***
g. ***I am rugged because I refuse to be a source of vulnerability or weakness.***
h. ***I am rugged because I assure my code will support its mission.***
i. ***I am rugged because my code can face these challenges and persist in spite of them.***
j. ***I am rugged, not because it is easy, but because it is necessary and I am up for the challenge.***

j. Management practices for software
    a. Lean management practices
        i. Limit work in progress (WIP)
        ii. Visual management
        iii. Feedback from production
        iv. Lightweight change approvals
    b. Lean management leads to
        i. Westrum organisational culture
        ii. Software delivery performance
        iii. Less burnout
    c. Implement a lightweight change management process
    d. Segregation of duty
        i. Any change committed should be reviewed by someone who wasn't involved in authoring the change
        ii. Changes should only be applied to production using a fully automated process that forms part of a deployment pipeline
k. Product development
    a. Lean product development practices
        i. How teams slice up work
        ii. Whether teams have a good understanding of the flow of work
        iii. Whether organisations actively and regularly seek customer feedback

iv. Whether development teams have the authority to create and change specifications as part of the development process without requiring approval

b. Lean product development
    i. Work in small batches
    ii. Make flow of work visible
    iii. Gather and implement customer feedback
    iv. Team experimentation
c. Effective product management drives performance

Lean product management practices predict organizational performance, measured in terms of productivity, profitability, and market share. The virtuous cycle of increased delivery performance and Lean product management practices drives better outcomes for your organization (see Figure 8.2).



l.

m. Making work sustainable
    a. Deployment pain – if deployed outside of business hours, there is deployment pain.
    b. To reduce deployment pain
        i. Build systems that are designed to be deployed easily into multiple environments, can detect and tolerate failure sin their environments and can have various components of the system updated independently
        ii. Ensure that the state o production systems can be reproduced (with the exception of production data0 in an automated fashion from information in version control
        iii. Build intelligence into the application and the platform so that the deployment process can be as simple as possible.
    c. Burnout
        i. Managers who want to avert employee burnout should concentrate their attention and efforts on:
            1. Fostering a respectful, supportive work environment that emphasis learning from failure rather than blaming,
            2. Communicating a strong sense of purpose
            3. Investing in employee development

4. Asking employees what is preventing them from achieving their objectives and then fixing those things
5. Giving employees time, space and resources to experiment and learn

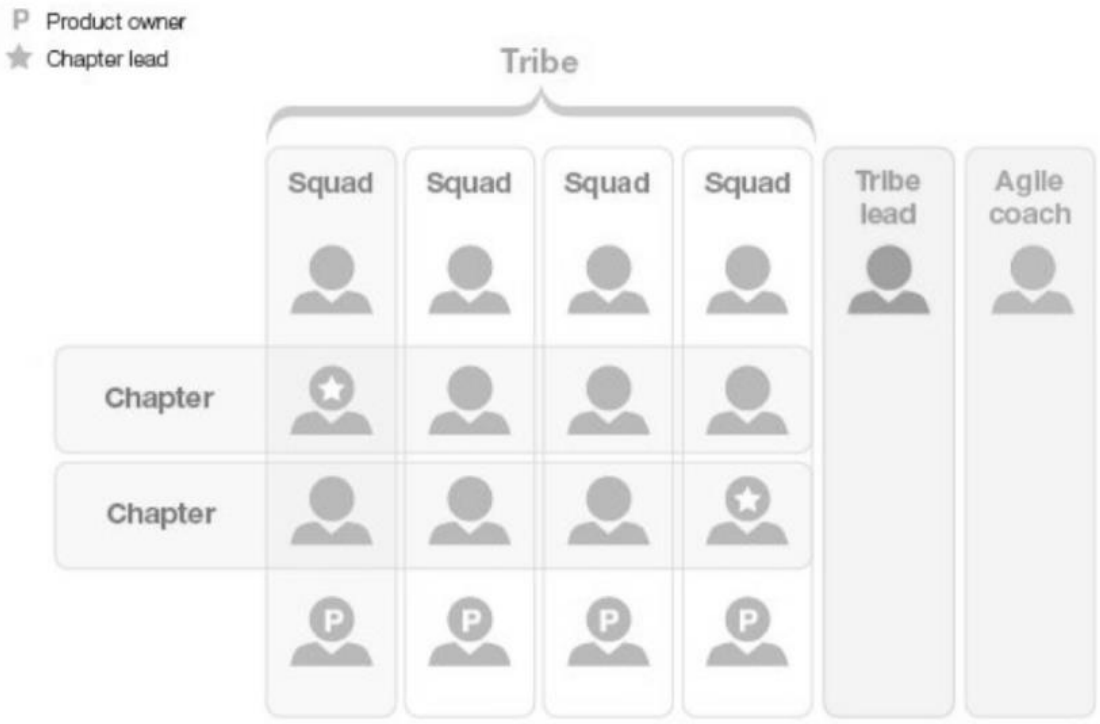
d. Common problems that can lead to burnout
    i. Work overload
    ii. Lack of control
    iii. Insufficient rewards
    iv. Breakdown of community
    v. Absence of fairness
    vi. Value conflicts
e. To measure burnout we asked respondents
    i. If they felt burned out or exhaust
    ii. Ed
    iii. If they felt indifferent or cynical about their work, or if they felt ineffective
    iv. If their work was having a negative effect on their life
f. How to reduce or fight burnout
    i. Organisation culture – supportive, respectful, blame free, learning from failures, communicating a shared sense of purpose. Human error is never the root cause of failure in systems
    ii. Deployment pain
    iii. Effectiveness of leaders – team leaders should limit work in progress and eliminate blockers
    iv. Organisational investments in DevOps
    v. Organizational performance
g. Continuous delivery led to less deployment pain and less burnout.
h. Lean practices led to less burnout

n. Employee satisfaction, identity and engagement
    a. Employee loyalty - measured by net promotor score. Employees who recommend their team or their organisation.
    b. Changing organisational culture and identity
        i. Both continuous delivery and lean practices lead to changes in identity. Identity leads to organisational performance
    c. Both continuous delivery and lean practices lead to job satisfaction.
    d. Job Satisfaction leads to organisational performance
    e. Diversity in tech, what our research found.
        i. Of all the surveys that were done in this book. The gender view is as follows
            1. 33% teams had no women
            2. 56% of teams had less than 10% female
            3. 81% of teams had less than 25% female
        ii. On average the statistics were
            1. 91% male
            2. 6% female
            3. 1% non binary
    f. What can be done – use the below online resources
        i. Anita Borg institute

ii. Geek feminism
iii. Project include

o. Leaders and managers
    a. Transformational leadership
    b. Five characteristics of a transformational leader are:
        i. Vision
        ii. Inspirational communication
        iii. Intellectual stimulation
        iv. Supportive leadership
        v. Personal recognition
    c. Transformational leadership including vision, inspirational comms, intellectual stimulation, supportive leadership and personal recognition leads to:
        i. Test automation
        ii. Deployment automation
        iii. Trunk based development
        iv. Shift left of security
        v. Loosely coupled architecture
        vi. Empowered teams
        vii. Continuous integration
        viii. Team experimentation
        ix. Work in small batches
        x. Gather and implement customer feedback
    d. The role of leaders
        i. Ensure that existing resources are available
        ii. Establish a dedicated training budget
        iii. Encourage staff to attend technical conferences
        iv. Set up internal hack days
        v. Encourage teams to organise internal yak days
        vi. Hold regular internal DevOps mini conferences
        vii. Give staff dedicated time such as 20% time on several days after a release to experiment
    e. Enable cross functional collaboration by
        i. Building trust with your counterparts on other teams
        ii. Encouraging practitioners to move between departments
        iii. Actively seeking, encouraging and rewarding work that facilitates collaboration
    f. Create a climate of earning
        i. Training budget
        ii. Informal learning
        iii. Safe to fail
        iv. Share information
        v. Share innovation and have demo days and forums
    g. Make effective use of tools
        i. Make sure your team can choose their own tools
        ii. Make monitoring a priority
p. Part II The research

q. The science behind the book – There was robust effort in getting the facts and data in this book correct.
r. Introduction to psychometrics – Psychometrics were considered
s. Why use a survey – because it gives exactly the data they want
t. The data for the project
u. Part three – Transformation
v. High performance leadership and management



Tribe
(collection of squads with interconnected missions)

- includes on average 150 people
- empowers tribe lead to establish priorities, allocate budgets, and form interface with other tribes to ensure knowledge/insights are shared

Agile coach

- coaches individuals and squads to create high-performing teams

Squad
(basis of new agile organization)

- includes no more than 9 people; is self-steering and autonomous
- comprises representatives of different functions working in single location
- has end-to-end responsibility for achieving client-related objective
- can change functional composition as mission evolves
- is dismantled as soon as mission is executed

Product owner
(squad member, not its leader)

- is responsible for coordinating squad activities
- manages backlog, to-do lists, and priority setting

Chapter
(develops expertise and knowledge across squads)

Chapter lead

- is responsible for one chapter
- represents hierarchy for squad members (re: personal development, coaching, staffing, and performance management)

w.
x.
y. Capabilities to drive improvement
    a. Continuous delivery
        i. Use version control
        ii. Automate your deployment process
        iii. Implement continuous integration

iv. Use trunk based development methods
v. Implement test automation
vi. Support test data management
vii. Shift left on security
viii. Implement continuous delivery

b. Architecture capabilities
    i. Use a loosely coupled architecture
    ii. Architect for empowered teams
c. Product and process capabilities
    i. Gather and implement customer feedback
    ii. Make the flow of work visible through the value stream
    iii. Work in small batches
    iv. Foster and enable team experimentation
d. Lean management and monitoring capabilities
    i. Have a lightweight change approval process
    ii. Monitor across applications and infrastructure to inform business decisions
    iii. Check system health proactively
    iv. Improve processes and manage work with work in progress
    v. Visualise work to monitor quality and communicate throughout the team
e. Cultural capabilities
    i. Support a generative culture
    ii. Encourage and support learning
    iii. Support and facilitate collaboration among teams
    iv. Provide resources and tools that make work meaningful
    v. Support or embody transformational leadership

z. Stats – there are lots. Read the chapter if interested.
    a. High performers in DevOps had 50% high er market cap
    b.