Best Practices of Agile Teams

• • •

v5.0.6

Today:
Which practices separate great
agile teams from others?

Practicalities

Not actually a Scrum talk: it's just common

Practicalities

Not actually a Scrum talk: it's just common

Questions welcome

Practicalities

Not actually a Scrum talk: it's just common

Questions welcome

QR-code for slides at the end

Who am I

Jakob Buis

~~Developer~~
~~Team lead~~
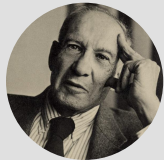~~Engineering Manager~~
Management consultant

Professional team builder

www.jakobbuis.nl (now with blogging!)

1. Measure actual usage

There is **nothing so useless** as doing with great efficiency that which **should not be done** at all.
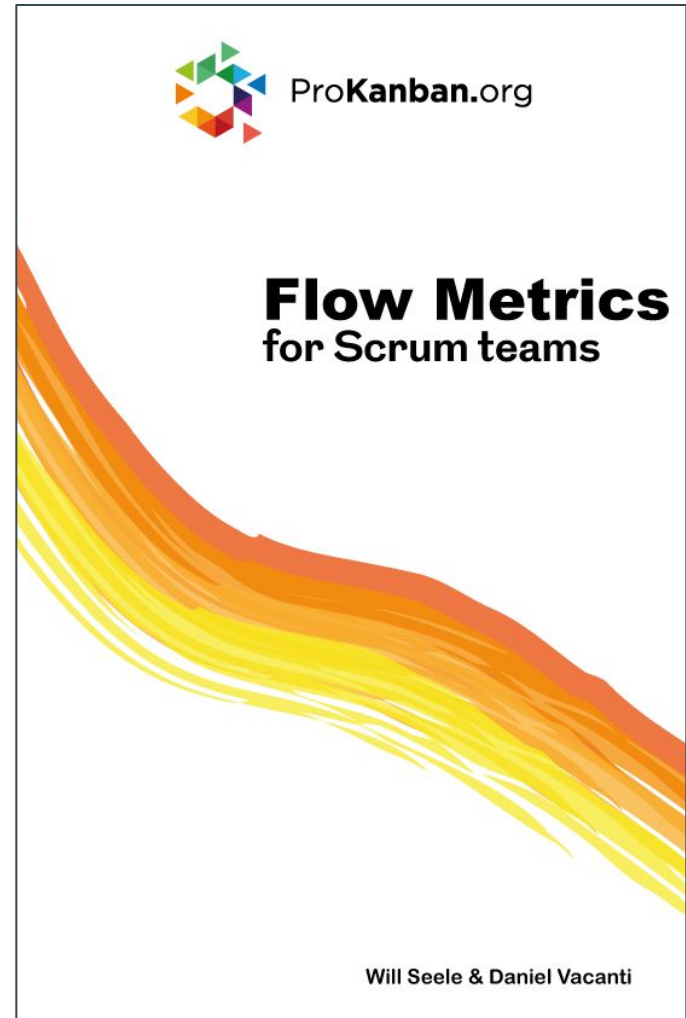
Peter Drucker

Add tracking tables

feature_foo_clicks		
id	user_id	timestamp
1	1	2025-03-10T14:30:10Z
2	2	2025-03-10T14:31:23Z
3	1	2025-03-11T09:16:00Z
4	3	2025-03-12T04:10:59Z

Board expansion
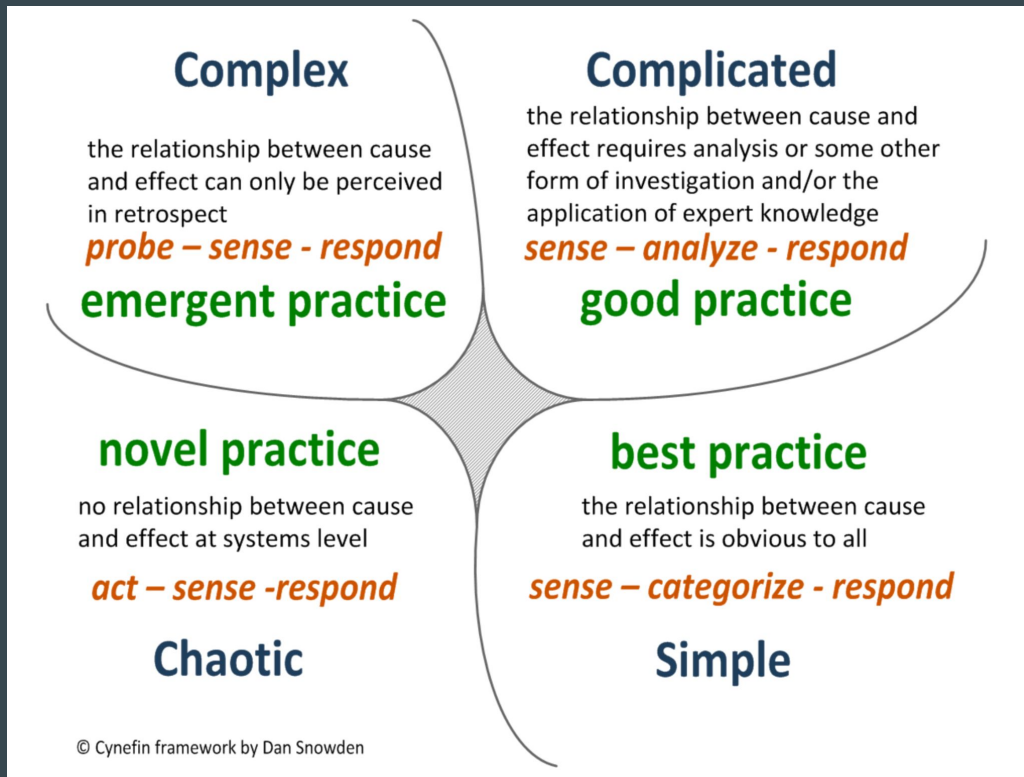
1. Options (Backlog)
2. Discovery
3. Building
   a. Not started
   b. Coding
   c. Code Review
   d. Ready for release
4. Validating
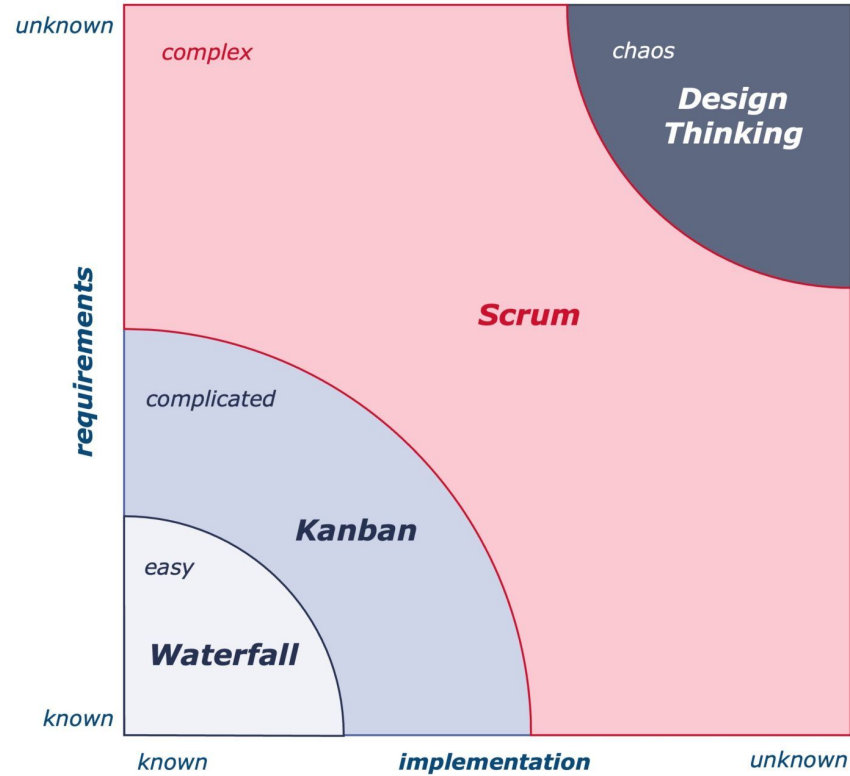5. Done

ProKanban.org

Flow Metrics
for Scrum teams

Will Seele & Daniel Vacanti

1. Measure actual usage

2. Working, tested software every sprint

Software development is complex work

Complex

the relationship between cause and effect can only be perceived in retrospect

probe – sense - respond

emergent practice

Complicated

the relationship between cause and effect requires analysis or some other form of investigation and/or the application of expert knowledge

sense – analyze - respond

good practice

novel practice

no relationship between cause and effect at systems level

act – sense -respond

Chaotic

best practice

the relationship between cause and effect is obvious to all

sense – categorize - respond

Simple

© Cynefin framework by Dan Snowden

Stacey Matrix

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

[Twelve Principles of Agile Software](#)

[View Signatories](#)

[About the Authors](#)

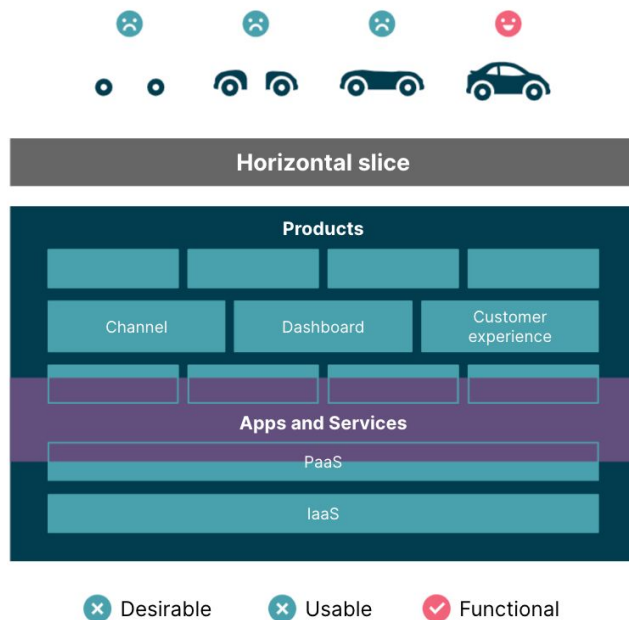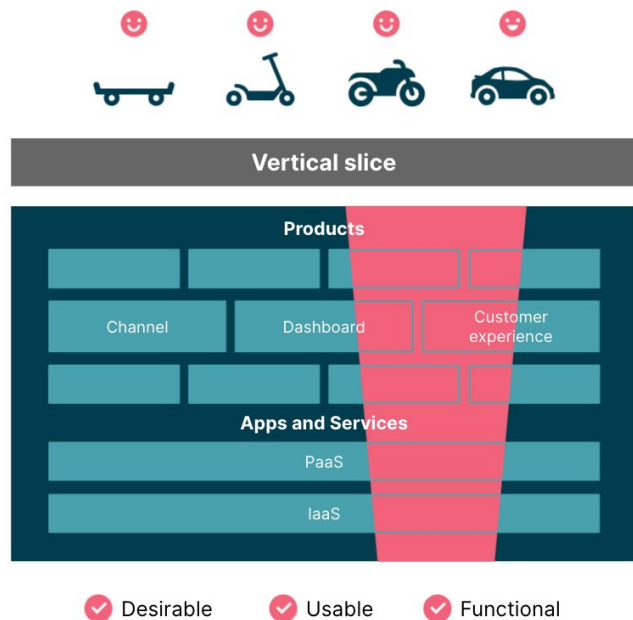About the Manifesto

Principle 1:
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Principle 7:
Working software is the primary measure of **progress**.

Deliver functional slices

Delivering early with "Thin Vertical Slices"

Working tested software, every sprint

Erase all dependencies
- encode manual stage-gates earlier & shift left
- decoupling architecture
- release yourself
- team layout changes (Team Topologies)

Working tested software, every sprint

Erase all dependencies
- encode manual stage-gates earlier & shift left
- decoupling architecture
- release yourself
- team layout changes (Team Topologies)

Better habits
- Avoid big-design up-front
- Incur (some) technical debt
- Don't optimize for personal productivity

2. Working, tested software every sprint

3. Data-driven estimation

This guy is a software engineer, you can tell by his awesome estimation skills

Subject to biases

Optimism bias

Confirmation bias

Group-think / bandwagon

Flaw of averages

Re-estimation bias

#NoEstimates

#NoEstimates (Allen Holub)
https://www.youtube.com
/watch?v=QVBInCTu9Ms

Improving estimation

Good:

> make items smaller
> multi-point estimates
> same-sizing everything: "1 story point" and "too big"
> https://mdalmijn.com/p/roman-estimation-a-simple-easy-and

Improving estimation

Good:

    make items smaller
    multi-point estimates
    same-sizing everything: "1 story point" and "too big"
    https://mdalmijn.com/p/roman-estimation-a-simple-easy-and

Better:

    use data

Monte Carlo simulation

Record throughput per day:

0   7   2   6   6   3   7   2   9   1   13   0   0   2   4

Monte Carlo simulation

Record throughput per day:

0   7   2   6   6   3   7   2   9   1   13   0   0   2   4

Sample next 5 days:

2   0   2   7   0

Monte Carlo simulation

Record throughput per day:

0   7   2   6   6   3   7   2   9   1   13   0   0   2

Sample next 5 days:

2   0   2   7   0   =   11

Next week, we'll finish 11 stories

Pitfalls

The future is dependent on the past

Same-same, but different.

Pitfalls

The future is dependent on the past

100% certainty assholes

Pitfalls

The future is dependent on the past

100% certainty assholes

Weighted monte carlo

3. Data-driven estimation

4. Effective retrospectives

Make retrospectives effective

Inspect & adapt
  1-2 high priority improvements,
  implemented next sprint

Make retrospectives effective

Inspect & adapt
     1-2 high priority improvements,
     implemented next sprint

Escalate what you cannot solve

Make retrospectives effective

Inspect & adapt
       1-2 high priority improvements,
       implemented next sprint

Escalate what you cannot solve

Data-driven decision making

Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

https://cloud.google.com/blog/products/devops-sre/
using-the-four-keys-to-measure-your-devops-performance

nave

- Process Improvement Dashboard
- Planning & Forecasting
- Prioritization & Dependency Management
- Flow Metrics & Analytics
  - Home
  - Cumulative Flow Diagram
  - Cycle Time Scatterplot
  - Cycle Time Breakdown
  - Cycle Time Histogram
  - Aging Chart
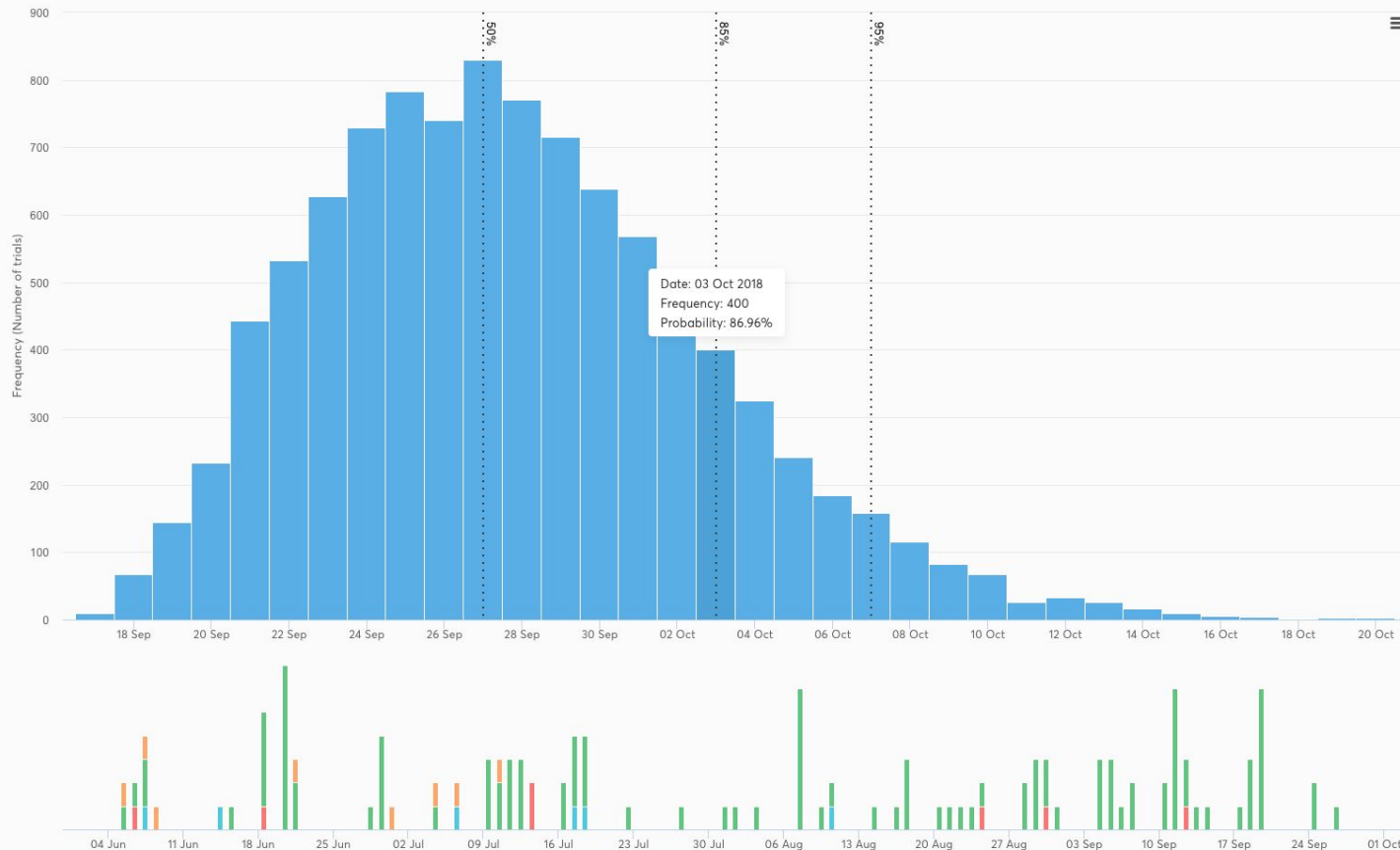 - Throughput Run Chart
  - Throughput Histogram
  - Flow Efficiency Chart
  - Due Date Performance Chart
  - Monte Carlo: Delivery Date
  - Monte Carlo: Number Of Tasks
  - Executive View
  - Executive Dashboard
  - Executive Report

Development (CONWIP: 2 Planned + 1 Unplanned)
Last Updated by Sonya

How we calculate your thresholds

Work in Progress HIGH

Your Work In Progress
Low — Stable — High
10 tasks / 8 tasks / 10 tasks

Your WIP is too high! Focus on moving current tasks along, especially the ones that are almost done:
- Under Review [DEV] 2222-738
- Ready for Review [DEV] 5302-730
- Ready for Review [DEV] 9983-283

Cycle Time STABLE

Your Cycle Time
Low — Normal — High
12 days / 20 days / 24 days

Your cycle time is stable! If you're still looking for improvement opportunities, consider reviewing the items with the longest cycle time:
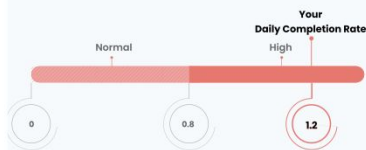- 16 days [DEV] 7382-849
- 12 days [DEV] 6638-839
- 12 days [DEV] 2673-098

Throughput HIGH

Your Daily Completion Rate
Normal — High
0 / 0.8 / 1.2

Your throughput is too high! Review these tasks to identify what factors contributed to this boost and assess if they can be sustained in the long run:
- Aug 4th [DEV] 8293-283
- Aug 4th [DEV] 8983-103
- Aug 4th [DEV] 8112-099

Work in Progress Age AT RISK

Your Total Work in Progress Age
Low — Normal — High
11 days / 23 days / 26 days

Your WIP age is at risk! Consider prioritizing the tasks with the highest work in progress age to bring your workflow back under control:
- 16 days [DEV] 1232-190
- 12 days [DEV] 3320-827

Cycle Time per Process Step HIGH

Your Development Cycle Time
Low — Normal — High
8 days / 16 days / 19 days

Your Development cycle time is too high! Focus on assessing the issue that caused the delay to bring cycle time on that status back in line:
- 16 days [DEV] 9892-772

Flow Efficiency STABLE

Your Flow Efficiency
Low — Normal — High
40% / 47% / 52%

Your flow efficiency is stable! If you want to further reduce wait times, consider analyzing the following items with the lowest flow efficiency and highest cycle time:
- 16 days 23% [DEV] 537-009
- 12 days 26% [DEV] 346-098

Controls for all charts

Colors
- Issue Type
- Status
- Priority
- Severity of Impact

Cycle time precision

Filters (70 cards)

Statuses (1/4)
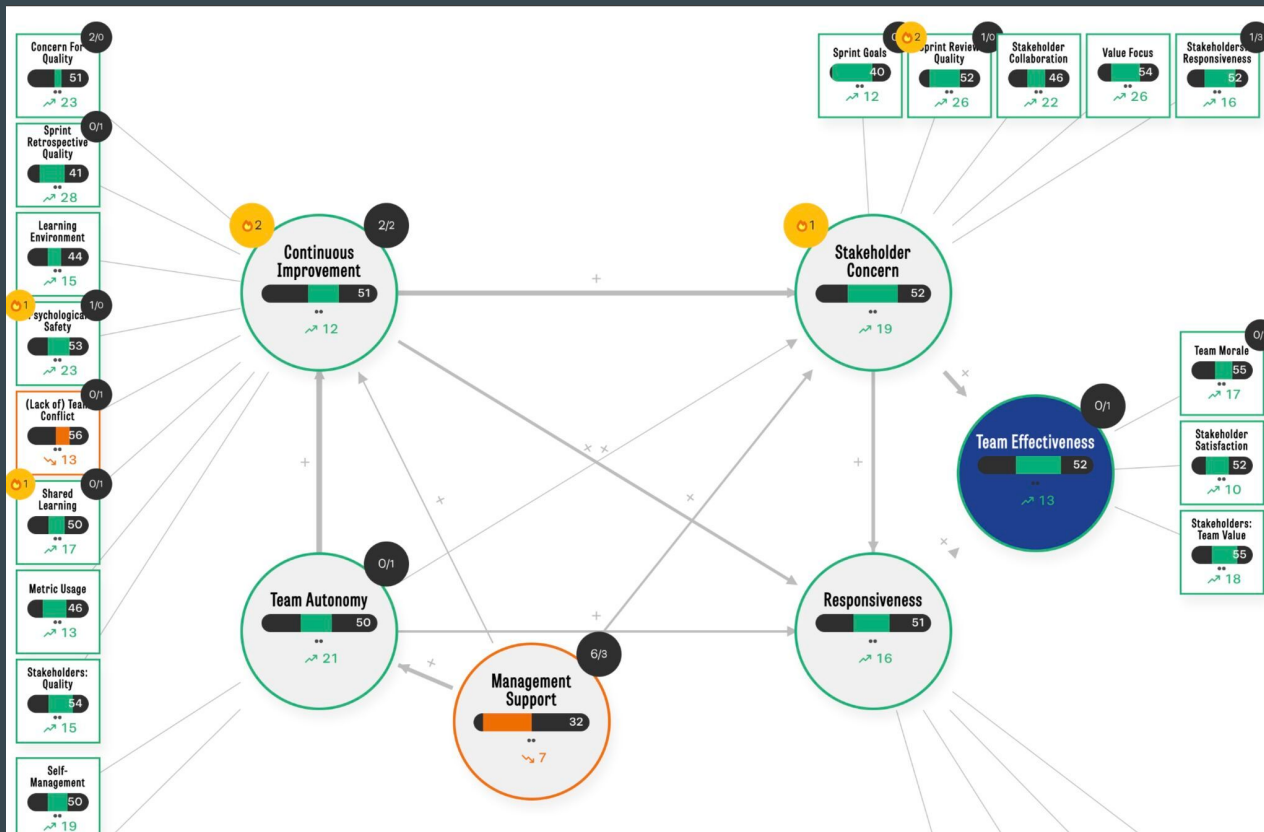- ☑ Development
- ☐ Code Review
- ☐ Testing
- ☐ Deployment

Priorities (4/4)
- ☑ Expedite
- ☑ Fixed Date
- ☑ Standard
- ☑ Intangible

Sprints (1/4)
- ☐ Sprint 01 - Launch Prep
- ☑ Sprint 02 - UI Improvements
- ☐ Sprint 03 - Bug Fixes
- ☐ Sprint 04 - UX Updates

getnave.com

Concern For Quality 2/0 — 51 — ↗ 23

Sprint Retrospective Quality 0/1 — 41 — ↗ 28

Learning Environment — 44 — ↗ 15

Psychological Safety 1/0 — 53 — ↗ 23

(Lack of) Team Conflict 0/1 — 56 — ↘ 13

Shared Learning 0/1 — 50 — ↗ 17

Metric Usage — 46 — ↗ 13

Stakeholders: Quality — 54 — ↗ 15

Self-Management — 50 — ↗ 19

Sprint Goals — 40 — ↗ 12

Sprint Review Quality 2 — 52 — ↗ 26

Sprint Review Quality 1/0 — 52 — ↗ 26

Stakeholder Collaboration — 46 — ↗ 22

Value Focus — 54 — ↗ 26

Stakeholders: Responsiveness 1/3 — 52 — ↗ 16

Continuous Improvement 2/2 — 51 — ↗ 12

Stakeholder Concern — 52 — ↗ 19

Team Morale 0/1 — 55 — ↗ 17

Stakeholder Satisfaction — 52 — ↗ 10

Team Effectiveness 0/1 — 52 — ↗ 13

Stakeholders: Team Value — 55 — ↗ 18

Team Autonomy 0/1 — 50 — ↗ 21

Responsiveness — 51 — ↗ 16

Management Support 6/3 — 32 — ↘ 7

4. Effective retrospectives

To do

To do:

1. Measure actual usage
2. Working tested software, every sprint
3. Data-driven estimation
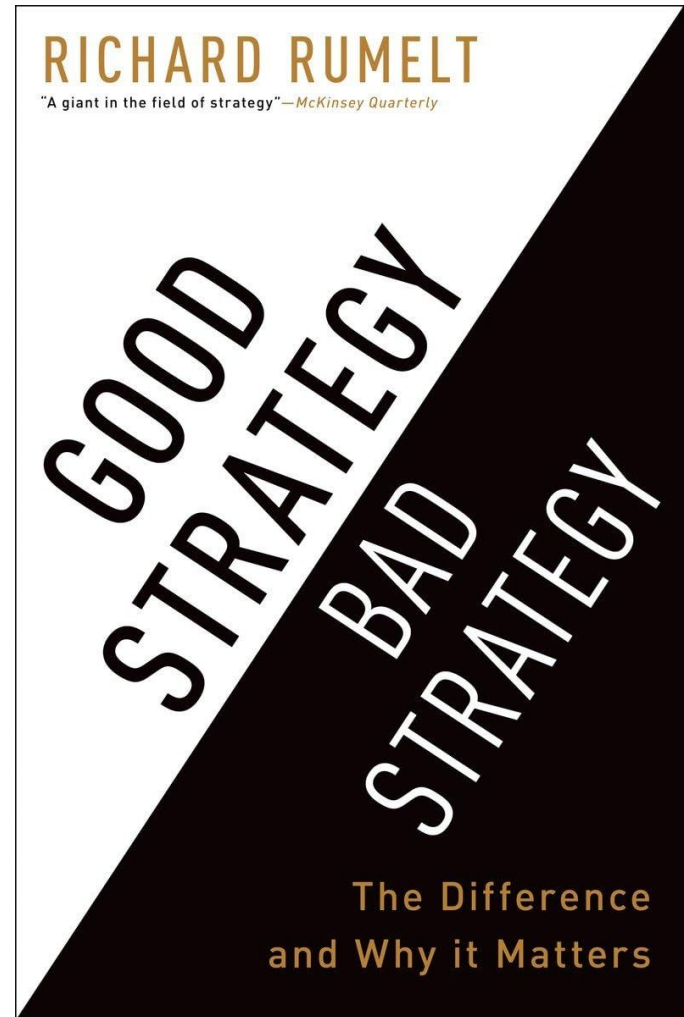4. Effective retrospectives

How to get started

The kernel of a strategy
contains three elements:
a diagnosis,
a guiding policy,
and coherent action.

RICHARD RUMELT

"A giant in the field of strategy" —*McKinsey Quarterly*

GOOD STRATEGY

BAD STRATEGY

The Difference
and Why it Matters

That's all!

Contact, blog & slides @
www.jakobbuis.nl