Best Practices of Agile Teams

•••

v3.0.5

Today:
Which practices separate great agile teams from others?

Today:

Which practices separate great agile teams from others?

Top 5 things the best agile teams get right, every time!

Bonus: the 5 things everyone gets wrong!!

Practicalities

A barely organized list of good and bad things

Not actually a Scrum talk; it's just common.

Clarifying questions welcome!

Who am I

Jakob Buis

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

Professional team builder

www.jakobbuis.nl (now with blogging!)

Should you listen to me?

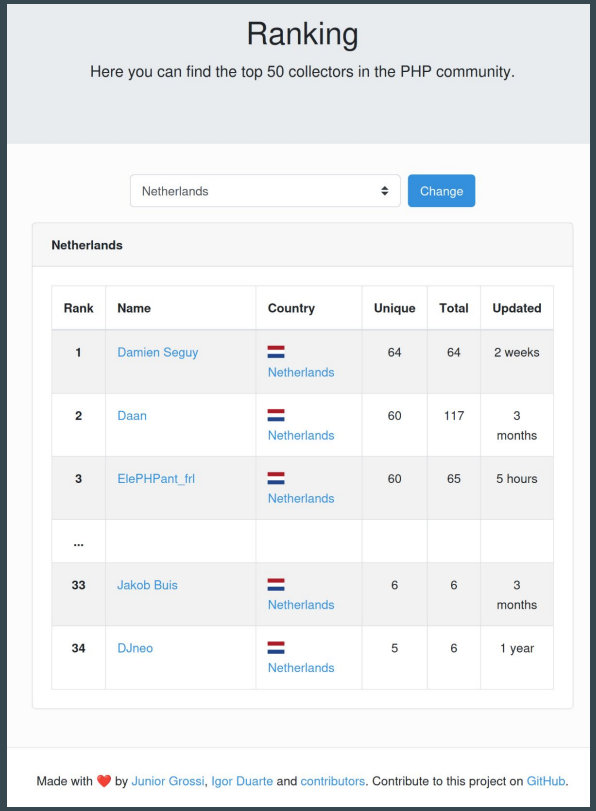
Yes, because:

Never been fired

Herd of 7 elePHPants

Worked with 15+ agile teams in various companies & industries

Professional Scrum Master II



elephpant.me

Should you listen to me?

Yes, because:

Never been fired

Herd of 7 elePHPants

Worked with 15+ agile teams in various companies & industries

Professional Scrum Master II

No, because:

I fuck up, a lot

Worked with 16 teams

Mostly in smaller companies
(< 300 people, < 30 engineers)

Most of my ideas come from other people
(links included!)

Do this

Avoid this

Experiment before complete methodology


SCRUM POLICE

Daily Scrum on Mon & Wed
Demo work not completed
Retrospective every 3rd sprint

Daily Scrum on Mon & Wed
Demo work not completed
Retrospective every 3rd sprint

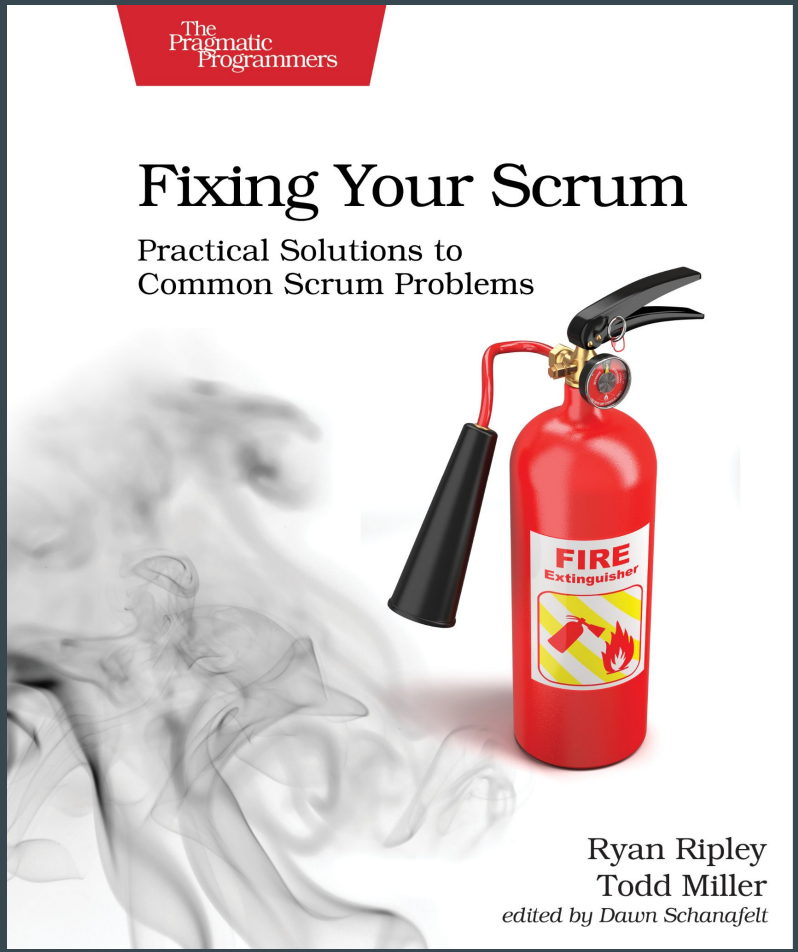
Scrum doesn't work for us

Learn your method

Read the Scrum Guide

Read a book

Take a course



Do this

Avoid this

Experiment before complete methodology

Do this

Working tested software, every sprint

Avoid this

Experiment before complete methodology

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

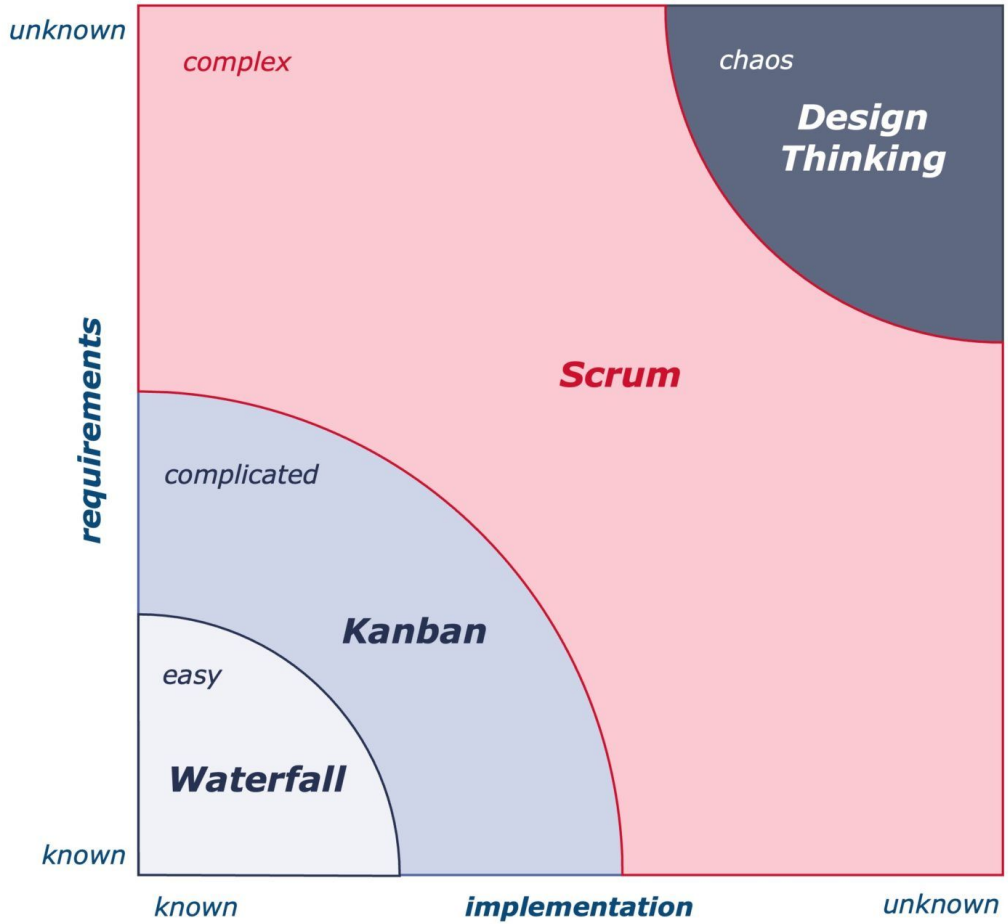
About the Authors
About the Manifesto

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.

Stacey Matrix




SPRINT REVIEW
DONE WORK
FEEDBACK
SCRUM TEAM
STAKEHOLDERS

Working tested software, every sprint

Get really good at vertical slicing

Start here: https://www.youtube.com/watch?v=urZ1TIycedU

Working tested software, every sprint

Get really good at vertical slicing

Start here: https://www.youtube.com/watch?v=urZ1TIycedU

Erase all dependencies

- decoupling architecture & operations
- team changes (Team Topologies)
- incur (some) technical debt

Working tested software, every sprint

Get really good at vertical slicing

Start here: https://www.youtube.com/watch?v=urZ1TIycedU

Erase all dependencies

- decoupling architecture & operations
- team changes (Team Topologies)
- incur (some) technical debt

Avoid big-design up-front

Do this

Working tested software, every sprint

Avoid this

Experiment before complete methodology

Do this

Working tested software, every sprint

Know how the product is used

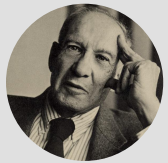
Avoid this

Experiment before complete methodology


SPRINT REVIEW
DONE WORK
FEEDBACK
SCRUM TEAM
STAKEHOLDERS

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

Peter Drucker

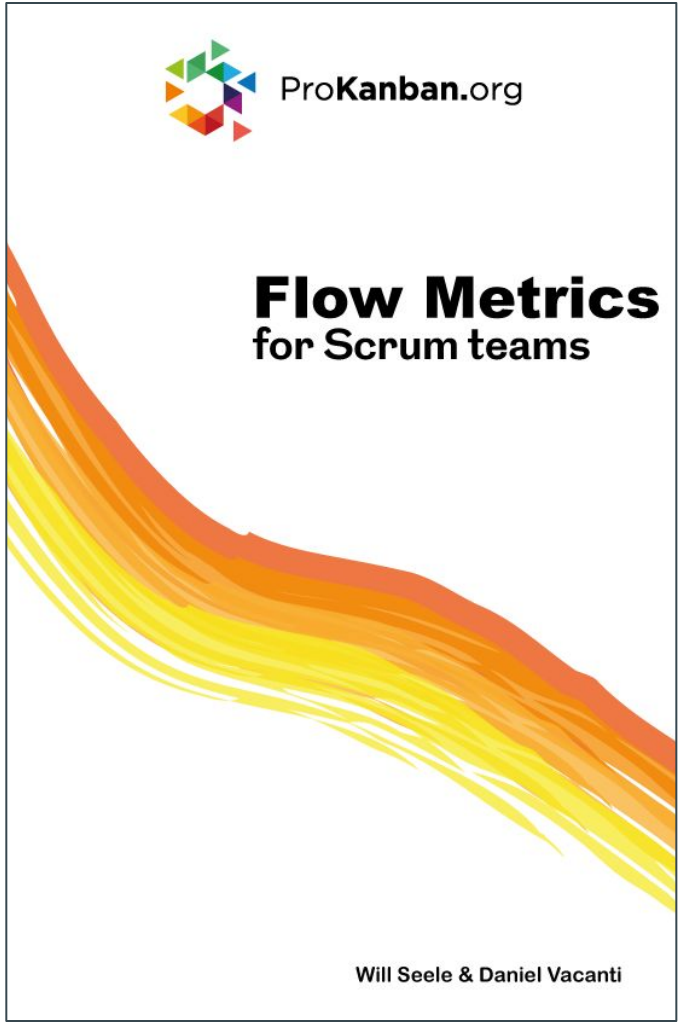


Add tracking tables

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



Do this

Working tested software, every sprint

Know how the product is used

Avoid this

Experiment before complete methodology

Do this

Working tested software, every sprint

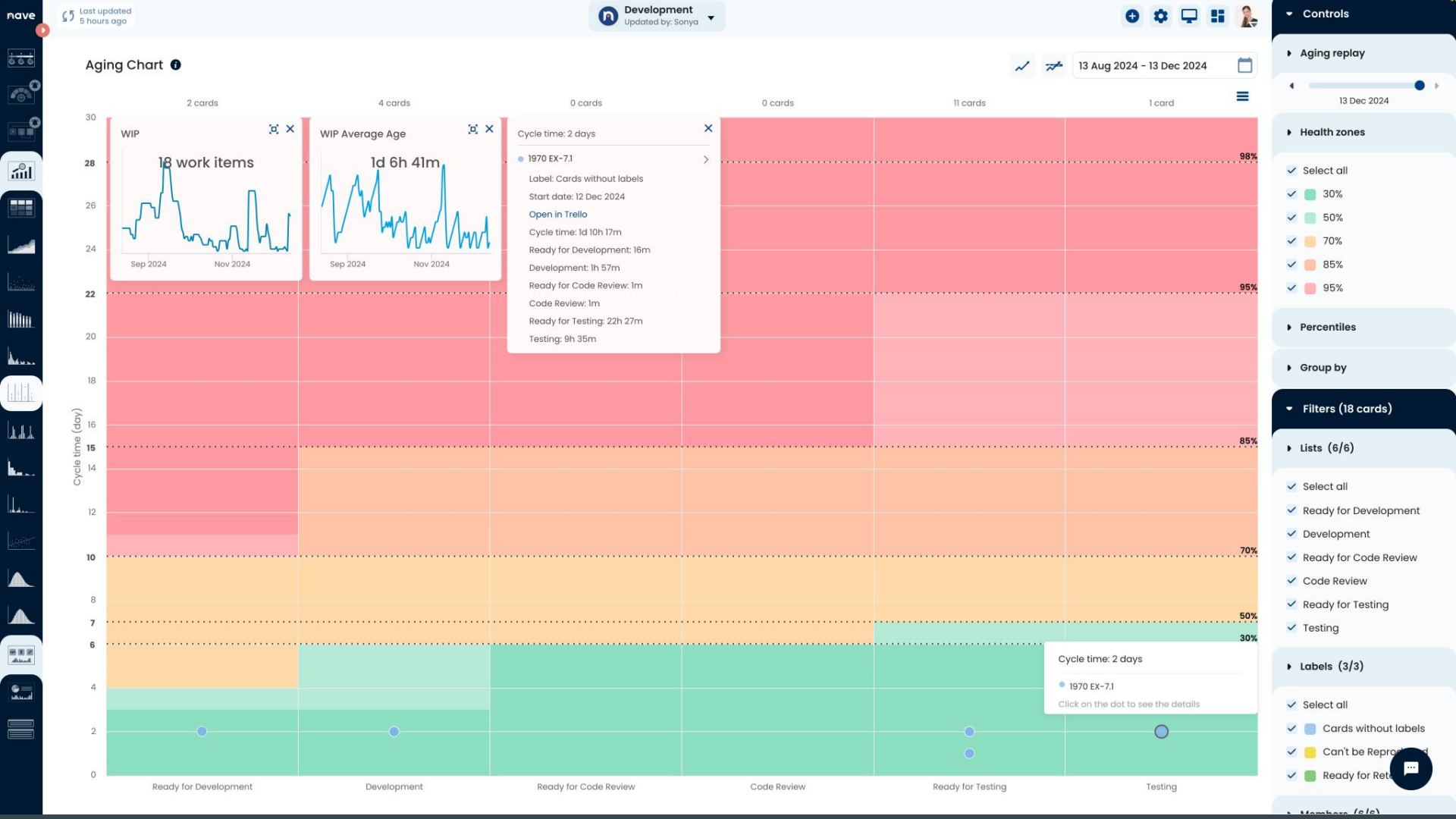
Know how the product is used

Do all work on the board

Avoid this

Experiment before complete methodology


鉄道改変予定


nave
Last updated
5 hours ago
Development
Updated by: Sonya
Aging Chart
13 Aug 2024 - 13 Dec 2024
2 cards
4 cards
0 cards
0 cards
11 cards
1 card
WIP
18 work items
Sep 2024
Nov 2024
WIP Average Age
1d 6h 41m
Sep 2024
Nov 2024
Cycle time: 2 days
1970 EX-7.1
Label: Cards without labels
Start date: 12 Dec 2024
Open in Trello
Cycle time: 1d 10h 17m
Ready for Development: 16m
Development: 1h 57m
Ready for Code Review: 1m
Code Review: 1m
Ready for Testing: 22h 27m
Testing: 9h 35m
Cycle time (day)
98%
95%
85%
70%
50%
30%
Cycle time: 2 days
1970 EX-7.1
Click on the dot to see the details
Ready for Development
Development
Ready for Code Review
Code Review
Ready for Testing
Testing
Controls
Aging replay
13 Dec 2024
Health zones
Select all
30%
50%
70%
85%
95%
Percentiles
Group by
Filters (18 cards)
Lists (6/6)
Select all
Ready for Development
Development
Ready for Code Review
Code Review
Ready for Testing
Testing
Labels (3/3)
Select all
Cards without labels
Can't be Repro
Ready for Ret

Do all the work on the board

Create item: [title] + [assigned you] + [in progress]

Consider skipping ticket when:

- doing it right now
- takes < 10 minutes (and you're 99% certain)
- is a repeating action (automate it!)

Try a physical board

Bias to having a single board per team.
Items never go back: stuck is preferable.

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Avoid this

Experiment before complete methodology

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Avoid this

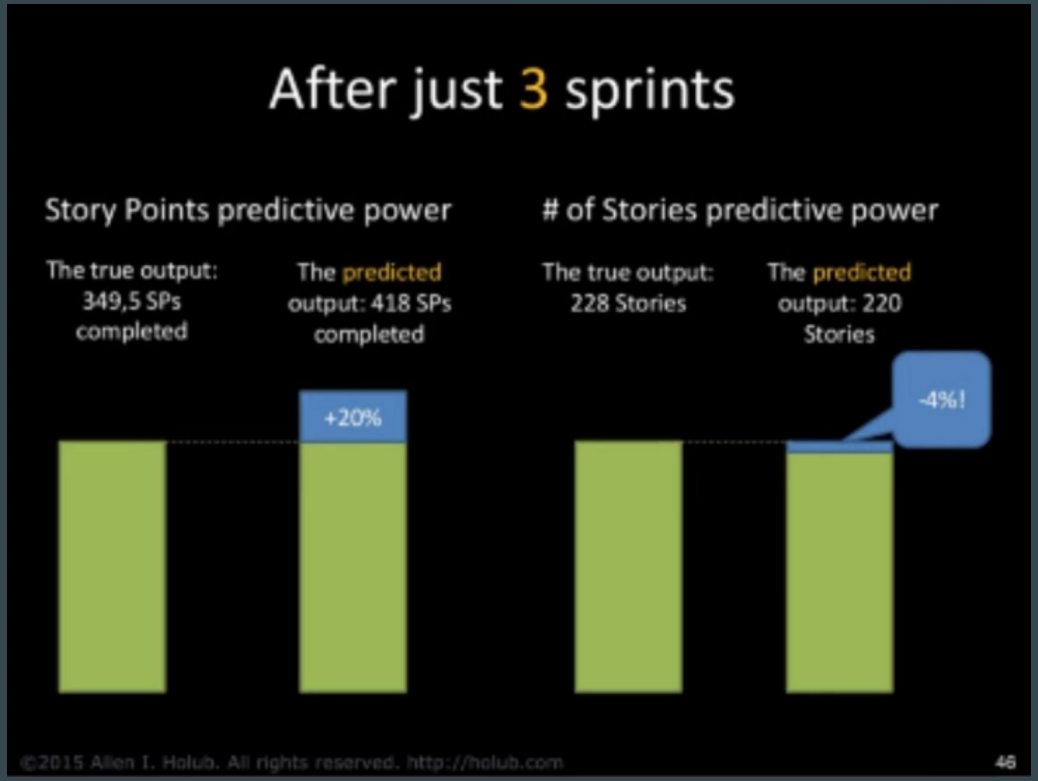
Experiment before complete methodology

You're not smarter than the customer

Estimation

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

#NoEstimates



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

Don't estimate

Good:

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

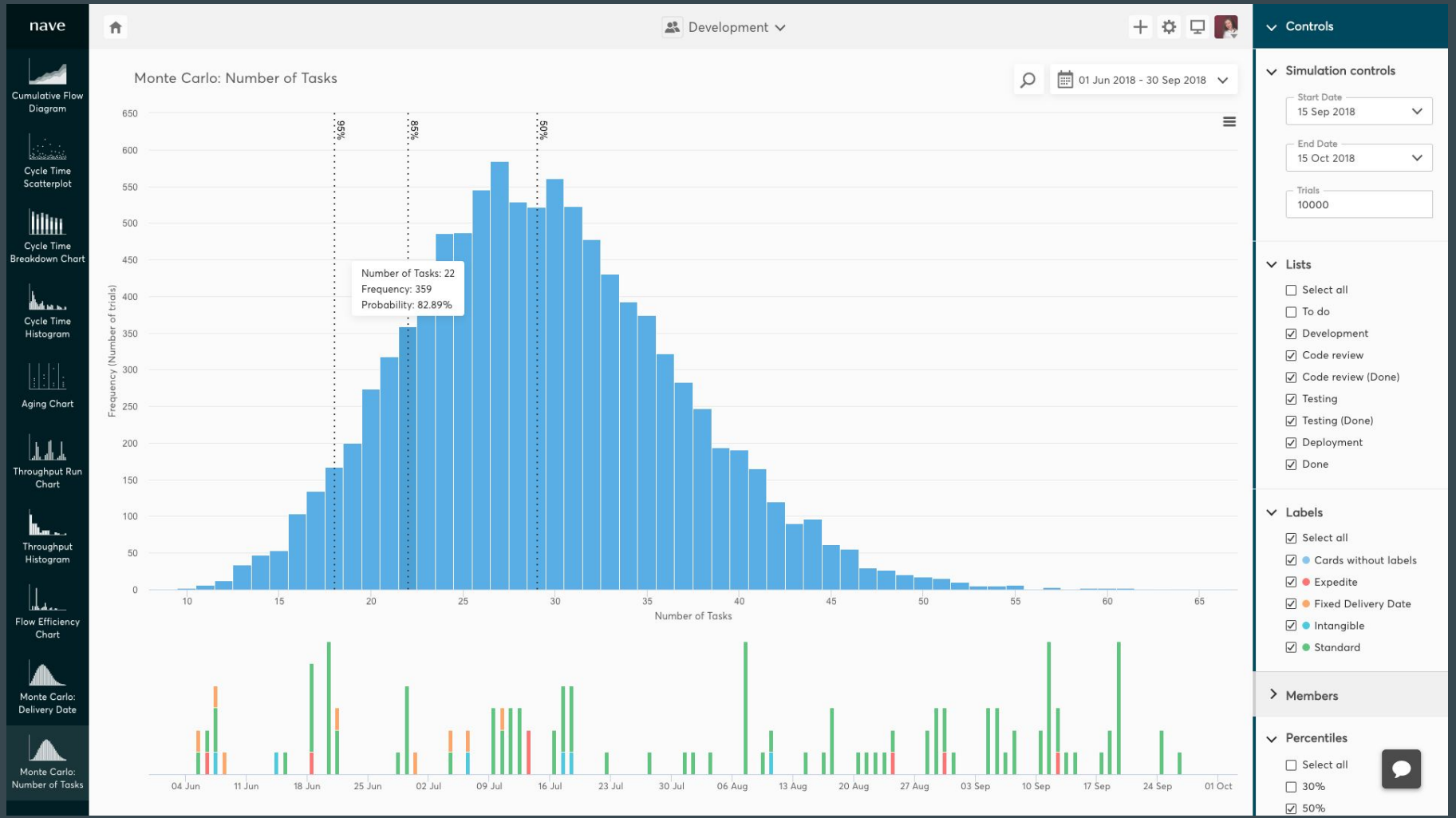
Record throughput per day:

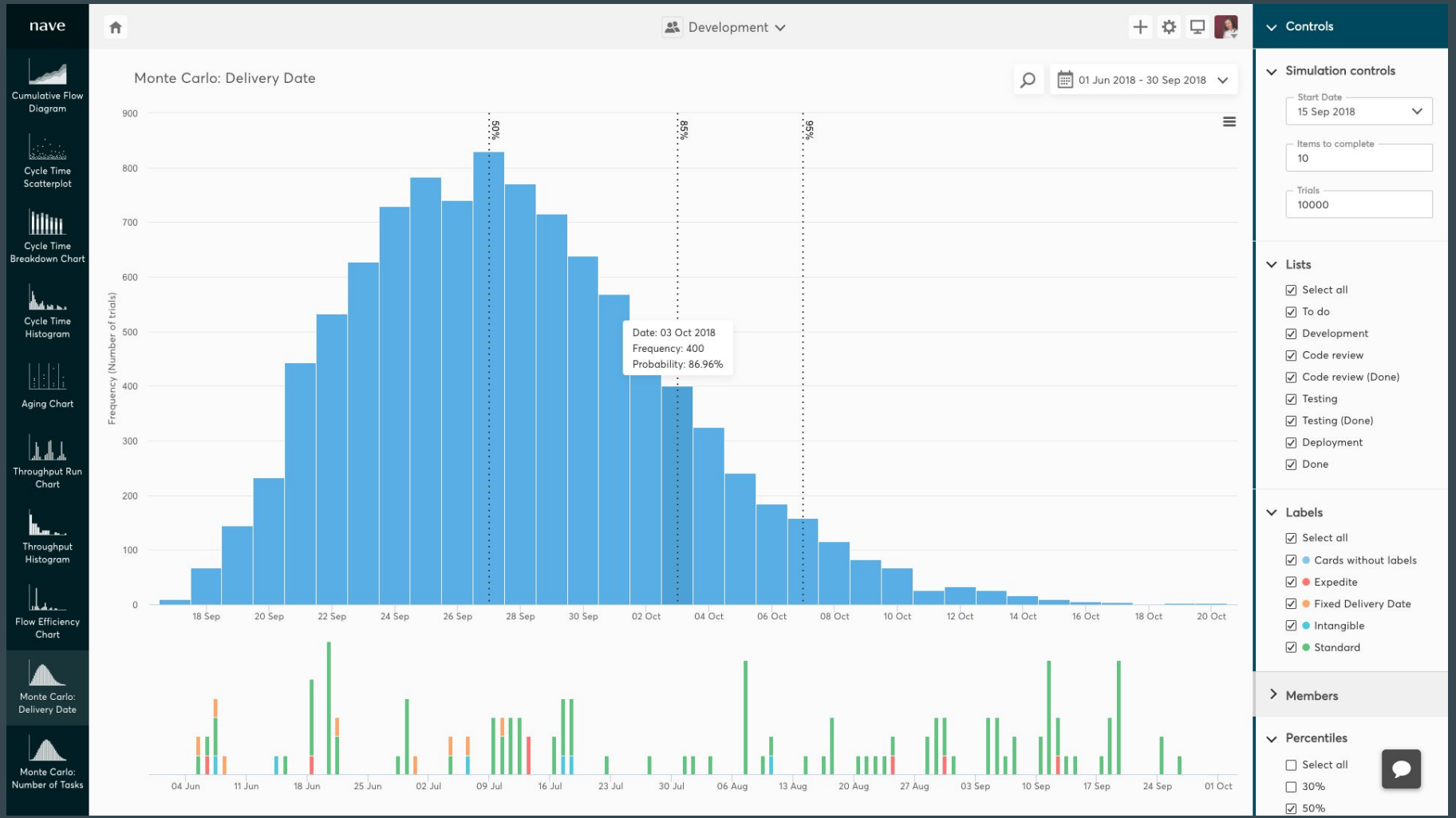
0 7 2 6 6 3 7 2 9 1 13 0 0 2

Sample next 5 days:

2 0 2 7 0 = 11

Next week, we'll finish 11 stories


nave
Cumulative Flow Diagram
Cycle Time Scatterplot
Cycle Time Breakdown Chart
Cycle Time Histogram
Aging Chart
Throughput Run Chart
Throughput Histogram
Flow Efficiency Chart
Monte Carlo: Delivery Date
Monte Carlo: Number of Tasks
Development
Monte Carlo: Number of Tasks
01 Jun 2018 - 30 Sep 2018
95%
85%
50%
Number of Tasks: 22
Frequency: 359
Probability: 82.89%
Frequency (Number of trials)
Number of Tasks
04 Jun
11 Jun
18 Jun
25 Jun
02 Jul
09 Jul
16 Jul
23 Jul
30 Jul
06 Aug
13 Aug
20 Aug
27 Aug
03 Sep
10 Sep
17 Sep
24 Sep
01 Oct
Controls
Simulation controls
Start Date
15 Sep 2018
End Date
15 Oct 2018
Trials
10000
Lists
Select all
To do
Development
Code review
Code review (Done)
Testing
Testing (Done)
Deployment
Done
Labels
Select all
Cards without labels
Expedite
Fixed Delivery Date
Intangible
Standard
Members
Percentiles
Select all
30%
50%


nave
Development
Controls
Cumulative Flow Diagram
Cycle Time Scatterplot
Cycle Time Breakdown Chart
Cycle Time Histogram
Aging Chart
Throughput Run Chart
Throughput Histogram
Flow Efficiency Chart
Monte Carlo: Delivery Date
Monte Carlo: Number of Tasks
Monte Carlo: Delivery Date
01 Jun 2018 - 30 Sep 2018
50%
85%
95%
Frequency (Number of trials)
Date: 03 Oct 2018
Frequency: 400
Probability: 86.96%
18 Sep
20 Sep
22 Sep
24 Sep
26 Sep
28 Sep
30 Sep
02 Oct
04 Oct
06 Oct
08 Oct
10 Oct
12 Oct
14 Oct
16 Oct
18 Oct
20 Oct
04 Jun
11 Jun
18 Jun
25 Jun
02 Jul
09 Jul
16 Jul
23 Jul
30 Jul
06 Aug
13 Aug
20 Aug
27 Aug
03 Sep
10 Sep
17 Sep
24 Sep
01 Oct
Simulation controls
Start Date
15 Sep 2018
Items to complete
10
Trials
10000
Lists
Select all
To do
Development
Code review
Code review (Done)
Testing
Testing (Done)
Deployment
Done
Labels
Select all
Cards without labels
Expedite
Fixed Delivery Date
Intangible
Standard
Members
Percentiles
Select all
30%
50%

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Have a strong Definition of Done

Absolute

Automated

Agreed with PO

Have a strong Definition of Done

Absolute

Automated

Agreed with PO

Never lie about Done

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Ineffective retrospectives


THIS IS FINE.

Make retrospectives effective

1-2 high priority improvements, implemented next sprint

Make retrospectives effective

1-2 high priority improvements, implemented next sprint

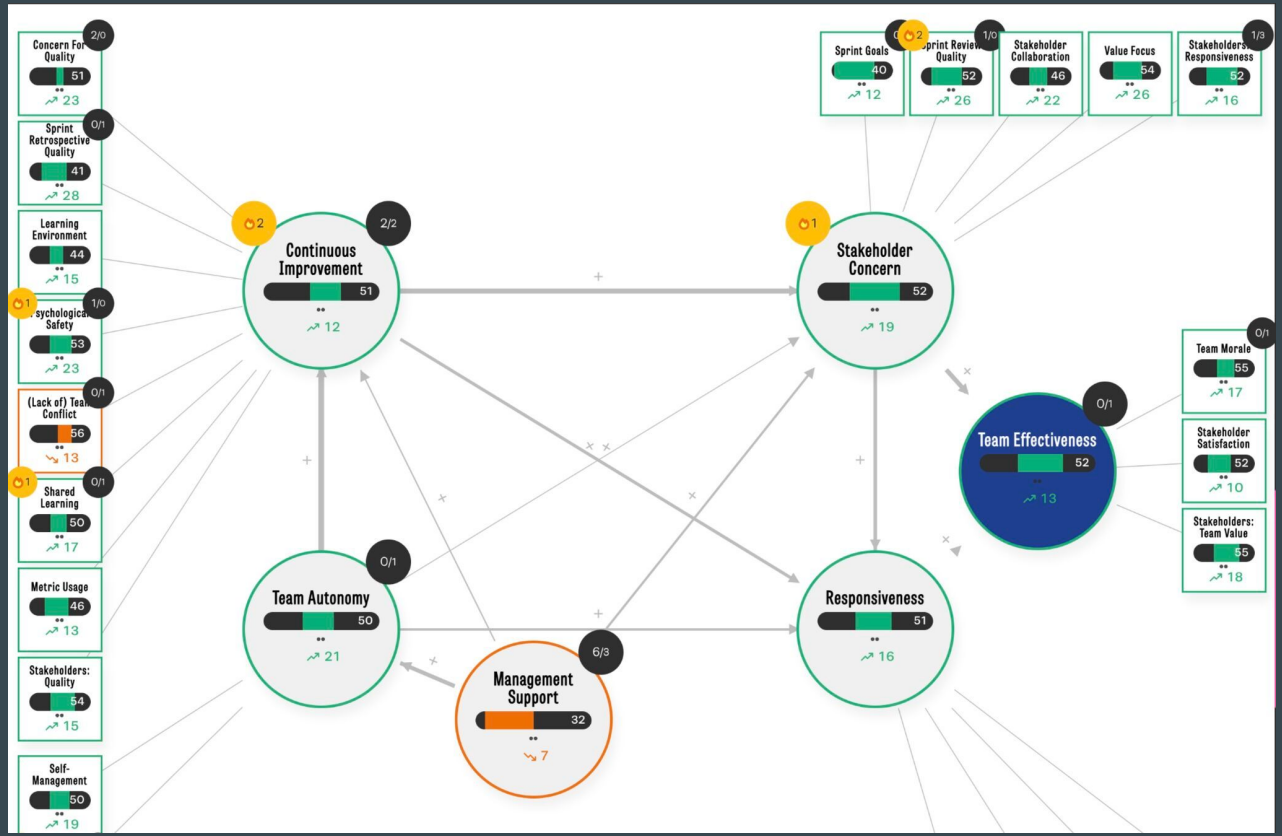
Escalate what you cannot solve

Make retrospectives effective

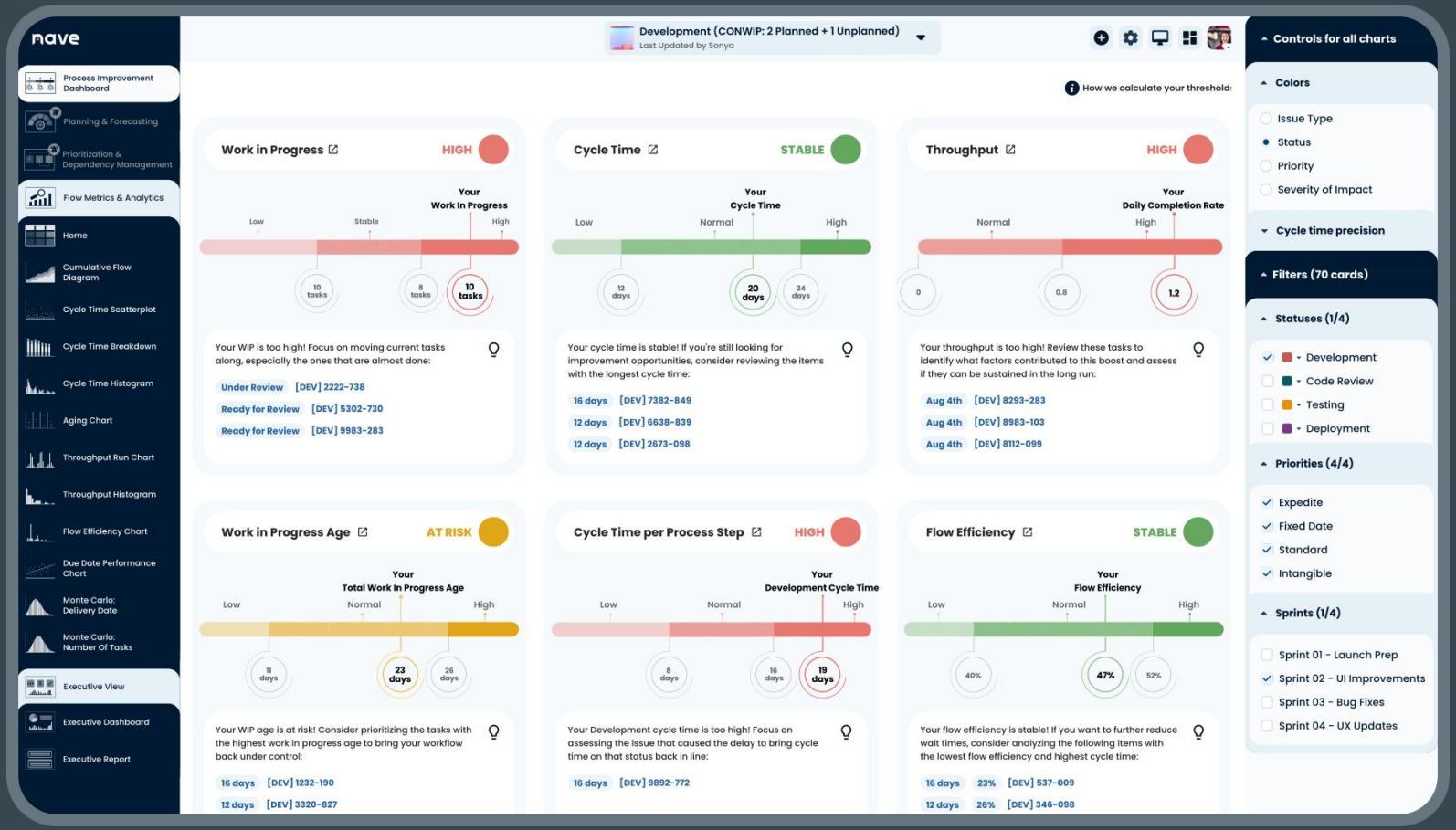
1-2 high priority improvements, implemented next sprint

Escalate what you cannot solve

Data-driven decision making



columinity.com


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 threshold
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

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Ineffective retrospectives

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

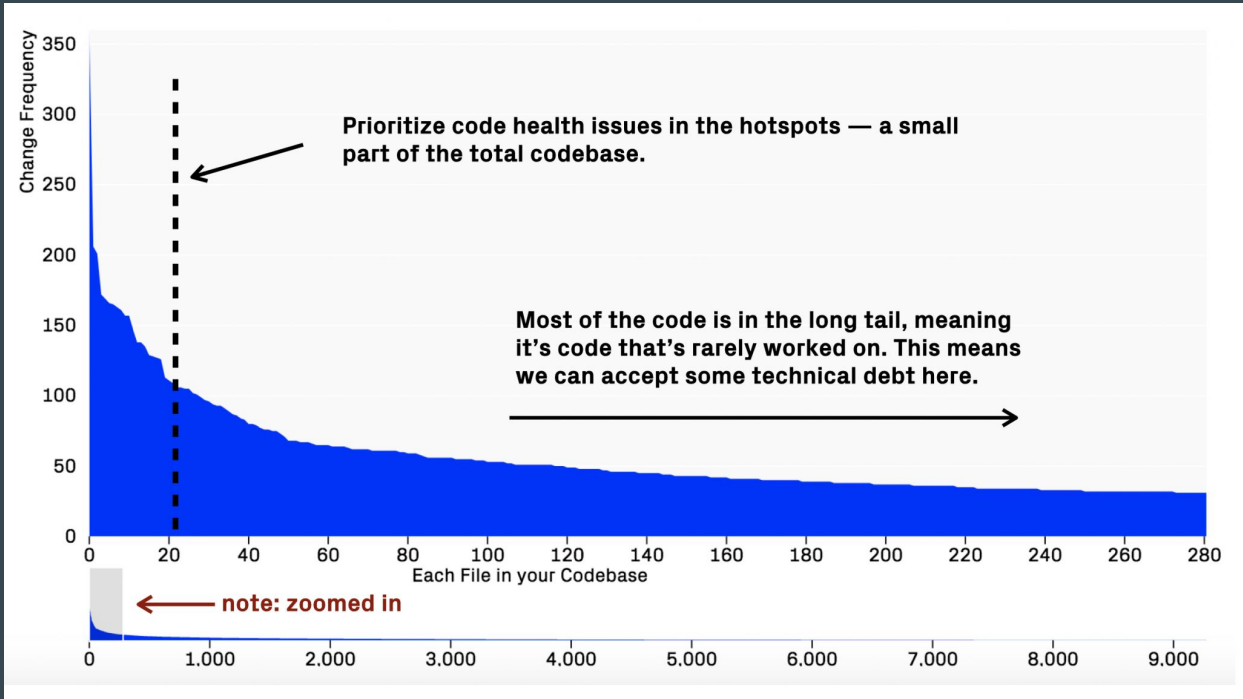
Ineffective retrospectives

Deal with technical debt


QUALITY

CONSISTENCY

Prioritize changing files



https://www.getunleash.io/blog/manage-technical-debt-measure-the-impact-and-prioritize-improvements-guided-by-development-data

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Ineffective retrospectives

Deal with technical debt

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Solve business problems

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Ineffective retrospectives

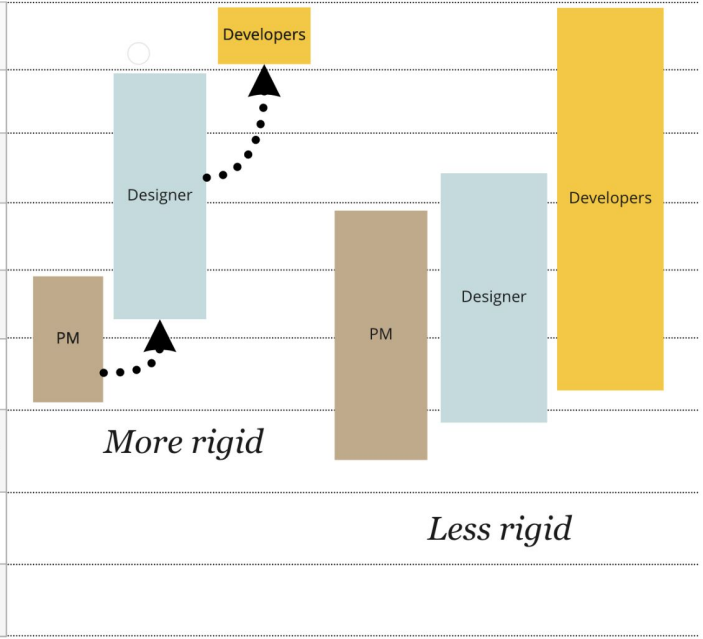
Deal with technical debt

MANDATE LEVELS

@johncutlefish

Effort is happening at all of these levels concurrently. It is all connected (explicitly, and often implicitly).

A	**Build exactly this [to a predetermined specification]**
B	**Build something that does [specific behavior, input-output, interaction]**
C	**Build something that lets a segment of customers complete [some task, activity, goal]**
D	**Solve this [more open-ended customer problem]**
E	**Explore the challenges of, and Improve the experience for, [segment of users/customers]**
F	**Increase/decrease [metric] known to influence a specific business outcome**
G	**Explore various potential leverage points and run experiments to influence [specific business outcome]**
H	**Directly generate [short-term business outcome]**
I	**Generate [long-term business outcome]**



https://cutlefish.substack.com/p/tbm-2752-mandate-levels

Solve business problems

Don't start here

Working software in the customers hands

Build a prototype

Get users in the room while designing

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Solve business problems

Avoid this

Experiment before complete methodology

You're not smarter than the customer

Estimation

Ineffective retrospectives

Deal with technical debt

To do

Do this

Working tested software, every sprint

Know how the product is used

Do all work on the board

Have a strong Definition of Done

Solve business problems

Avoid this

Experiment before complete methodology

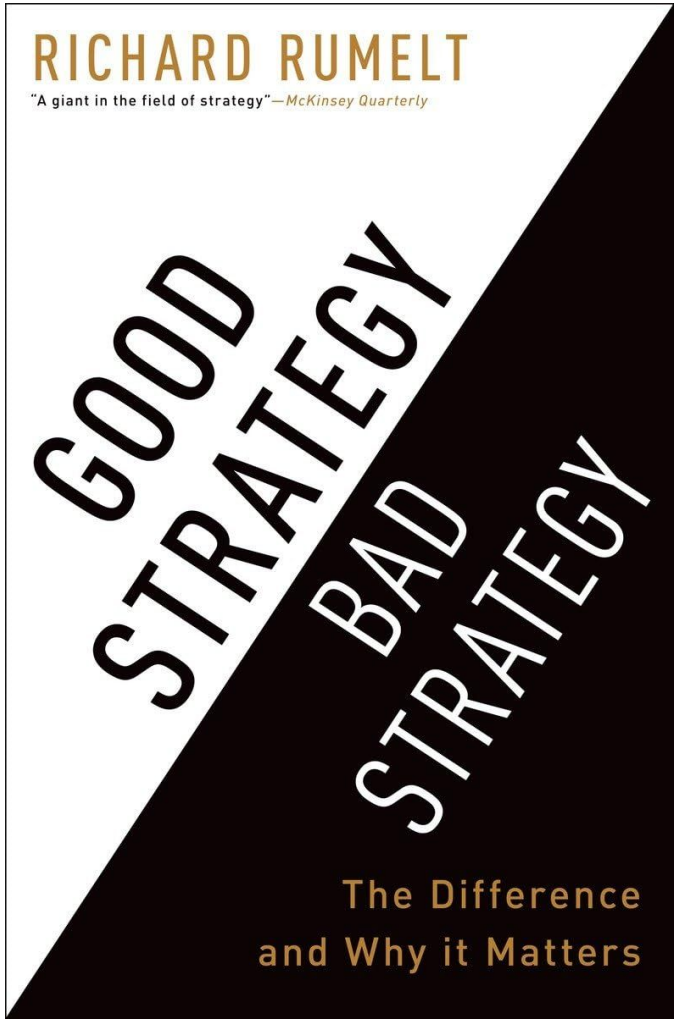
You're not smarter than the customer

Estimation

Ineffective retrospectives

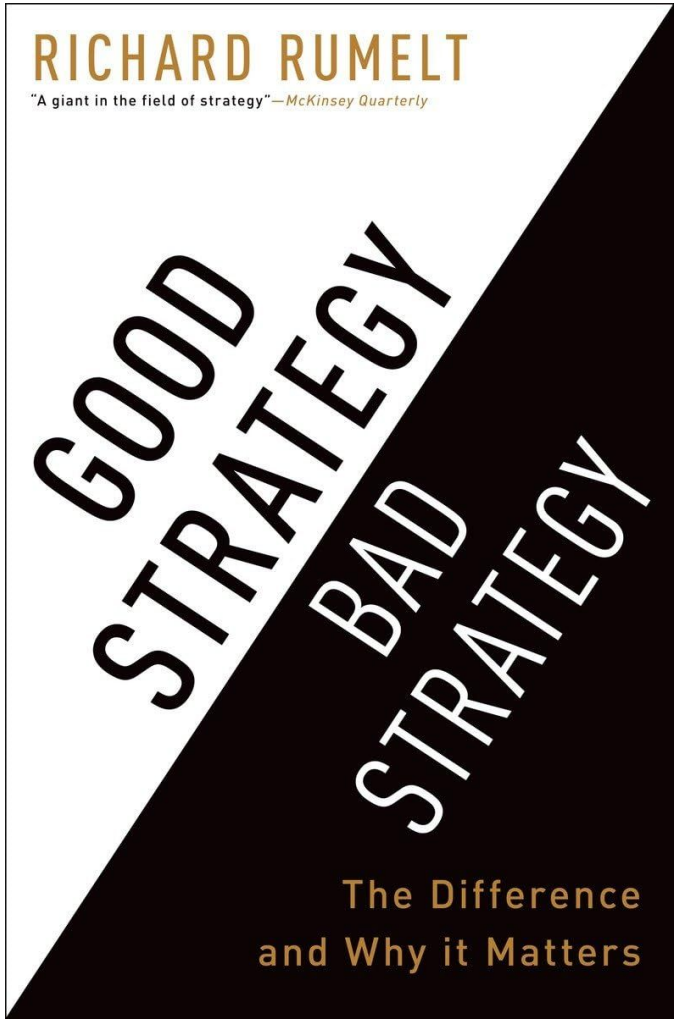
Consistent architecture

How to get started


RICHARD RUMELT
"A giant in the field of strategy"—McKinsey Quarterly
GOOD STRATEGY
BAD STRATEGY
The Difference
and Why it Matters

The kernel of a strategy contains three elements:

a diagnosis,
a guiding policy,
and coherent action.



That's all!

•••

Contact, blog & slides @
www.jakobbuis.nl