CEN 5016: Software Engineering

Fall 2024

University of Central Florida

Dr. Kevin Moran

Week 3 - Class 1: Project Planning & Agile Development

Administrivia

- Let me know if you are not on Ed Discussions
- Team-forming this week - *Due Tues, Sept 10th*
  - Teams of 3 students
  - See Ed Discussions Post
- Assignment 1 Graded by Thurs
- Assignment 2 out tomorrow

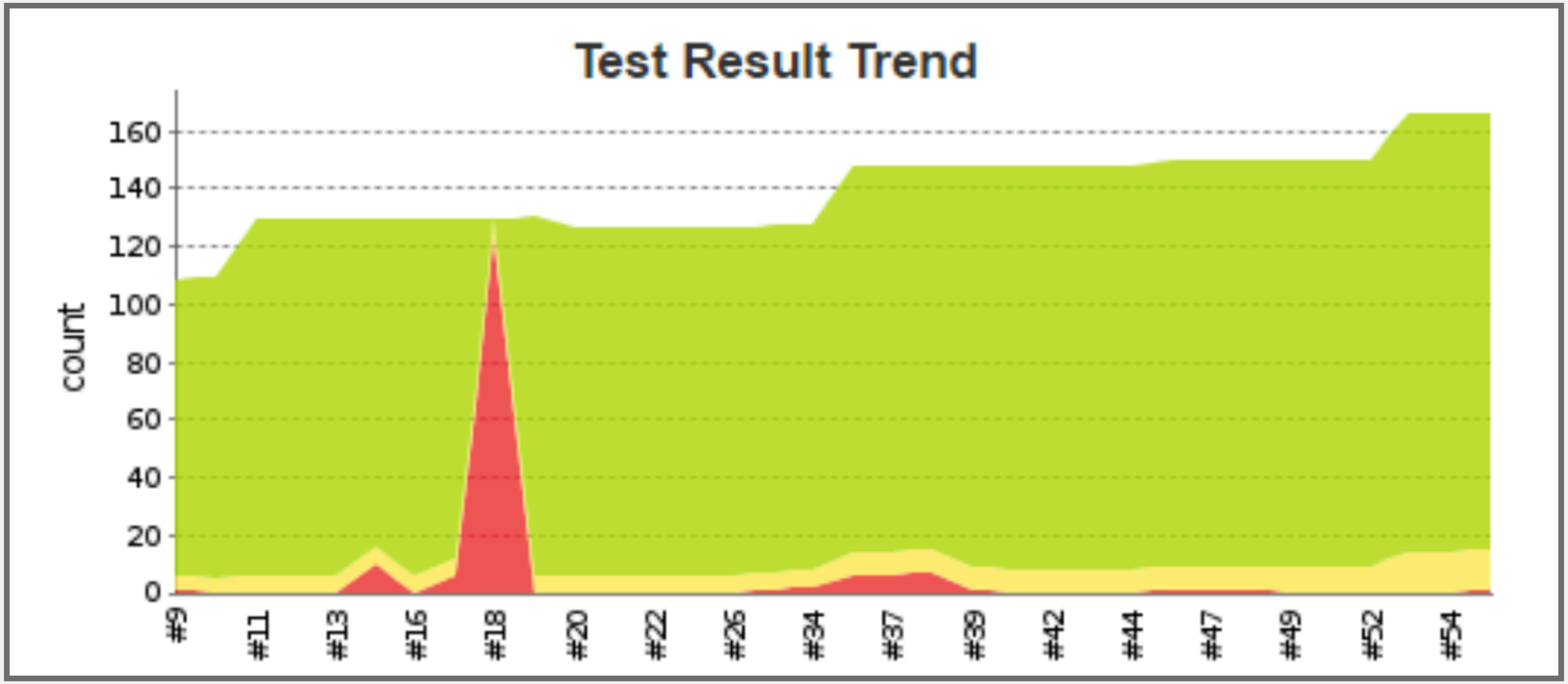
Software Measurement & Metrics

Why Measure?

Measurement for Decision Making

- Fund project?
- More testing?
- Fast enough? Secure enough?
- Code quality sufficient?
- Which feature to focus on?
- Developer bonus?
- Time and cost estimation? Predictions reliable?

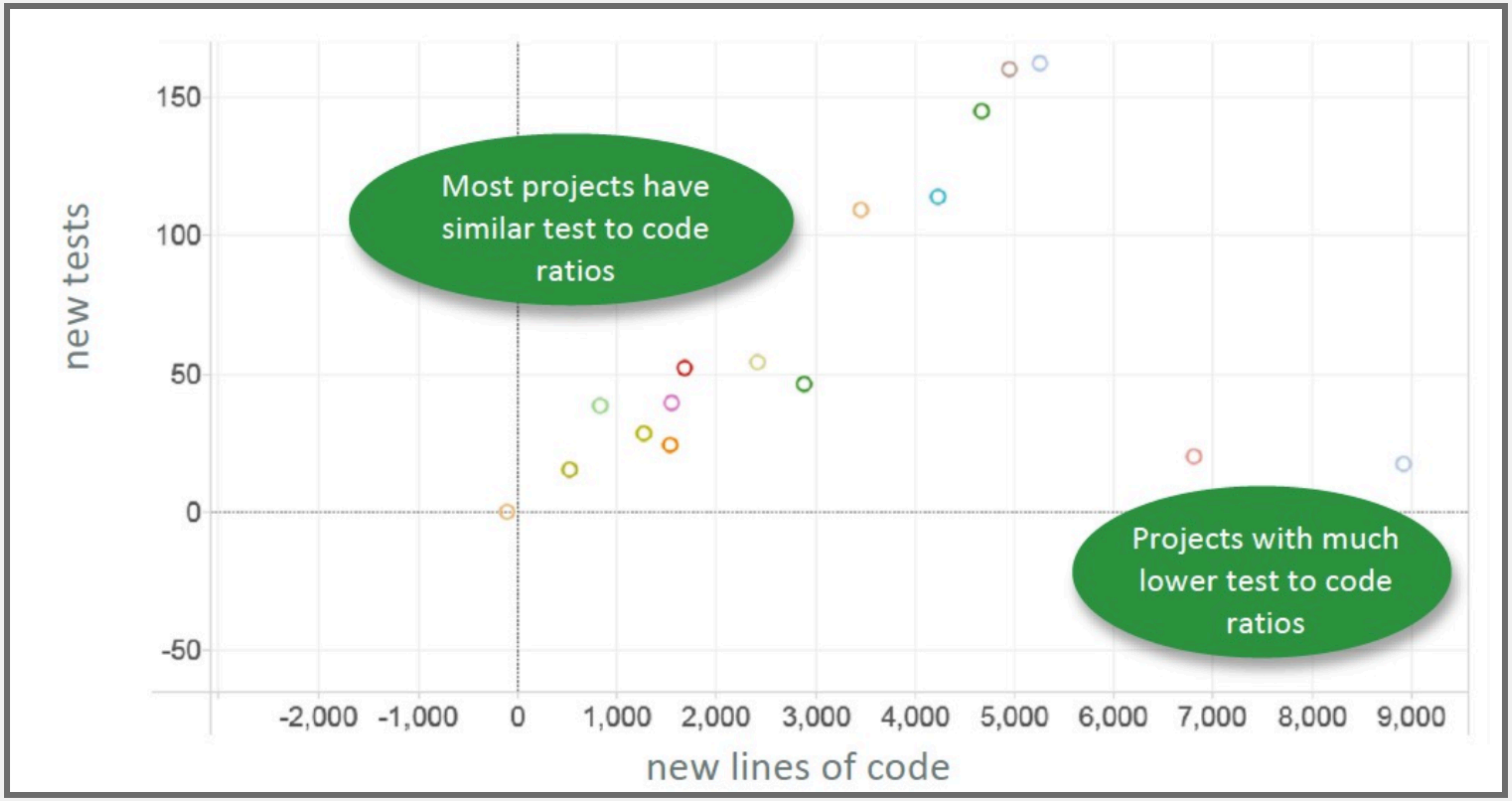
Trend Analyses



Benchmarking Against Standards

- Monitor many projects or many modules, get typical values for metrics

- Report deviations

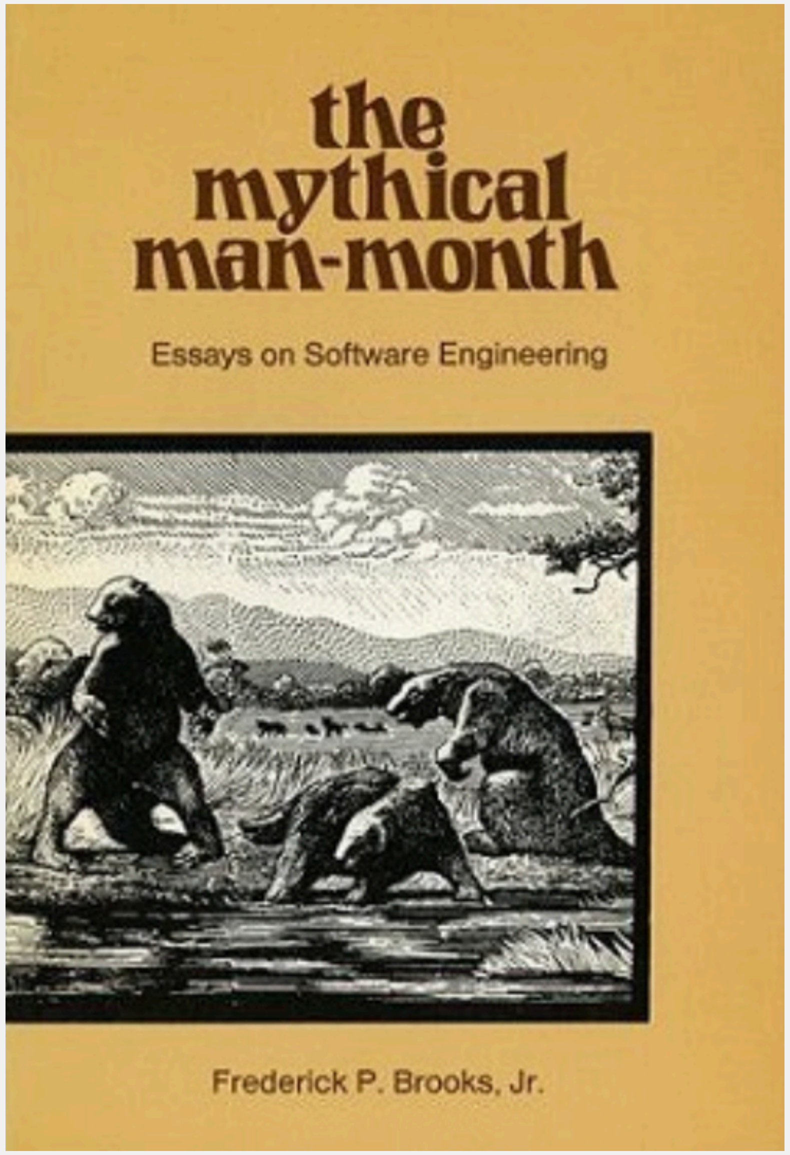


Antipatterns in Effort Estimation

- IBM in the 60s: Would account in “person-months”
  e.g. Team of 2 working 3 months = 6 person-months

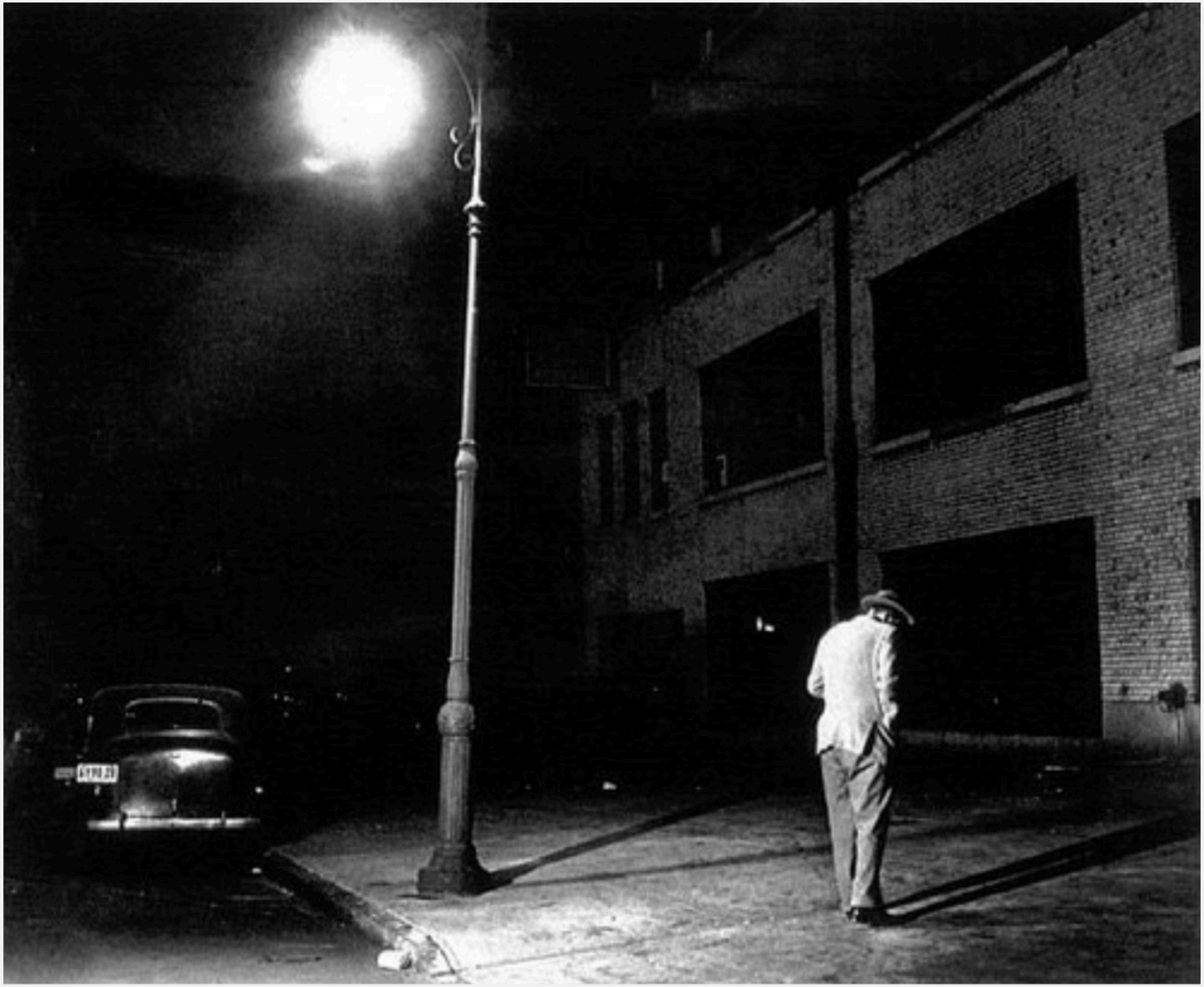
- LoC ~ Person-months ~ $$$

- Brooks: “Adding manpower to a late software project [just] makes it later.”



Measurement is Difficult

The Streetlight Effect



The Streetlight Effect

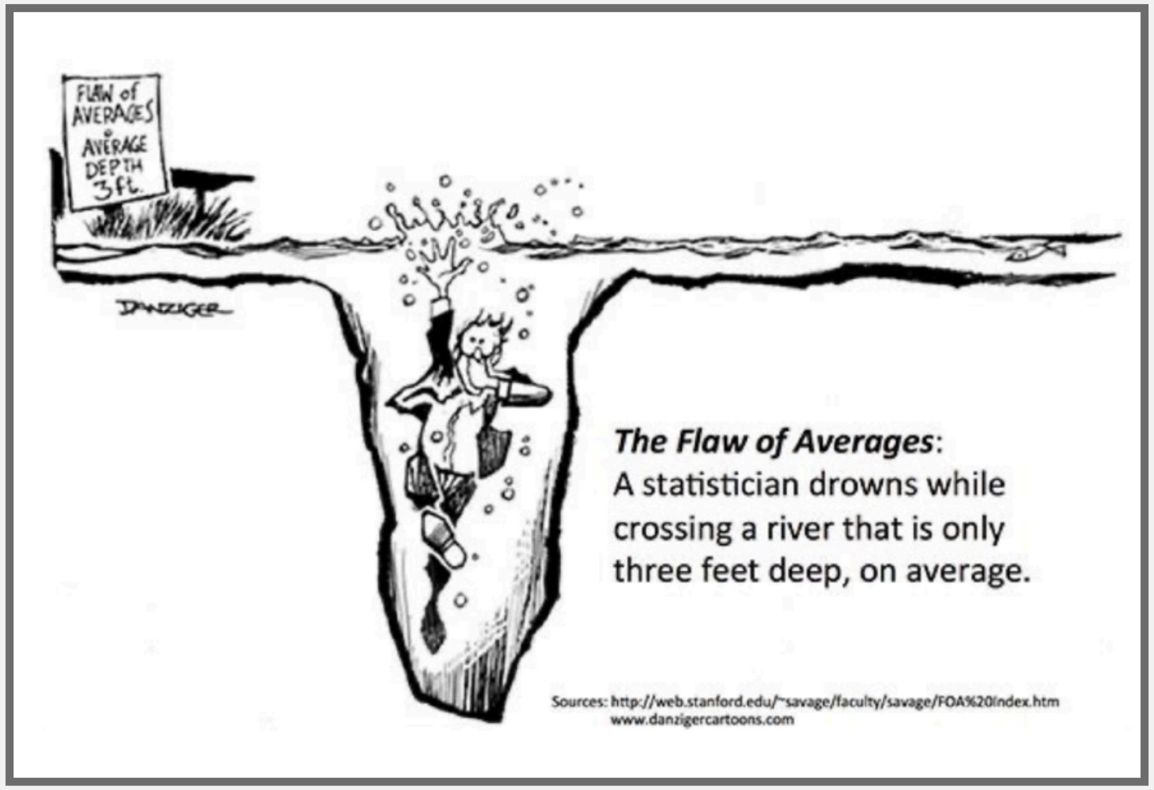
- A known observational bias.
- People tend to look for something only where it's easiest to do so.
- If you drop your keys at night, you'll tend to look for it under streetlights.

What could Possibly go Wrong?

- Bad statistics: A basic misunderstanding of measurement theory and what is being measured.

- Bad decisions: The incorrect use of measurement data, leading to unintended side effects.

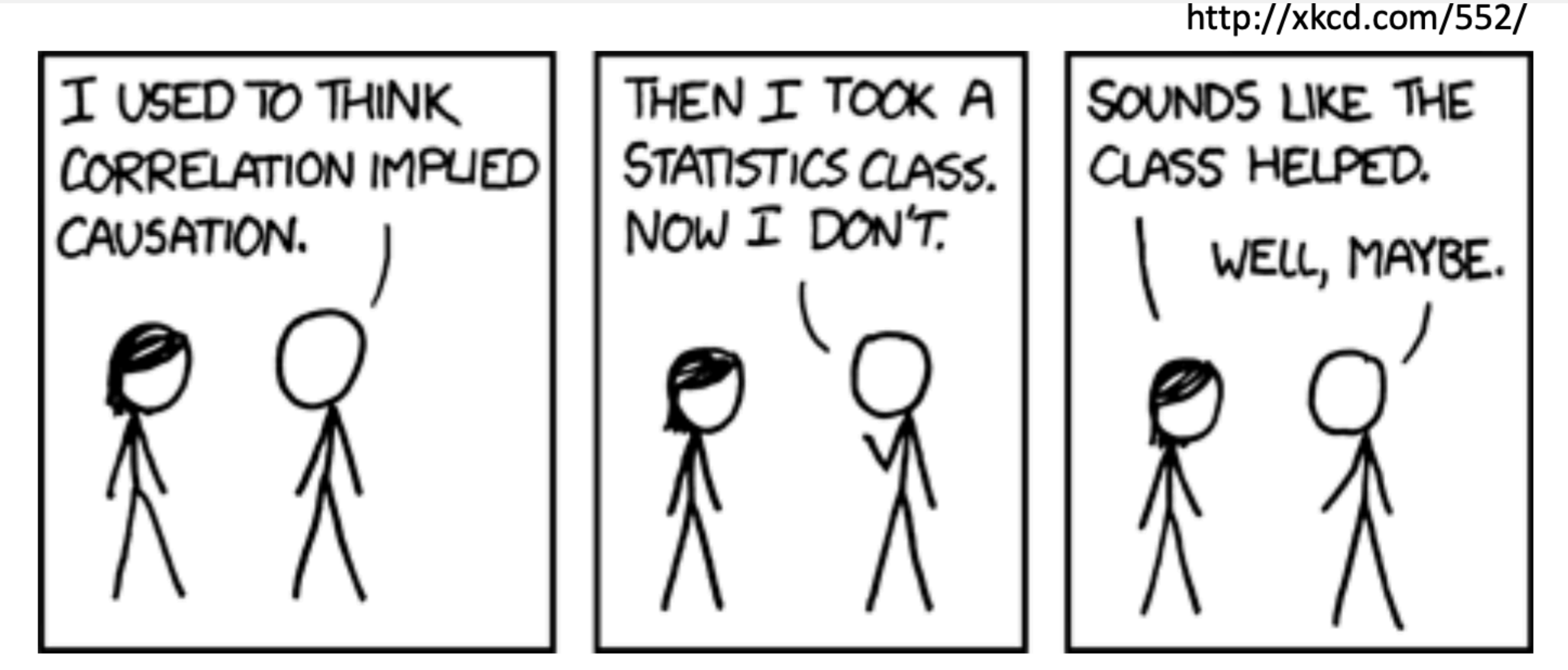
- Bad incentives: Disregard for the human factors, or how the cultural change of taking measurements will affect people.



The Flaw of Averages: A statistician drowns while crossing a river that is only three feet deep, on average.

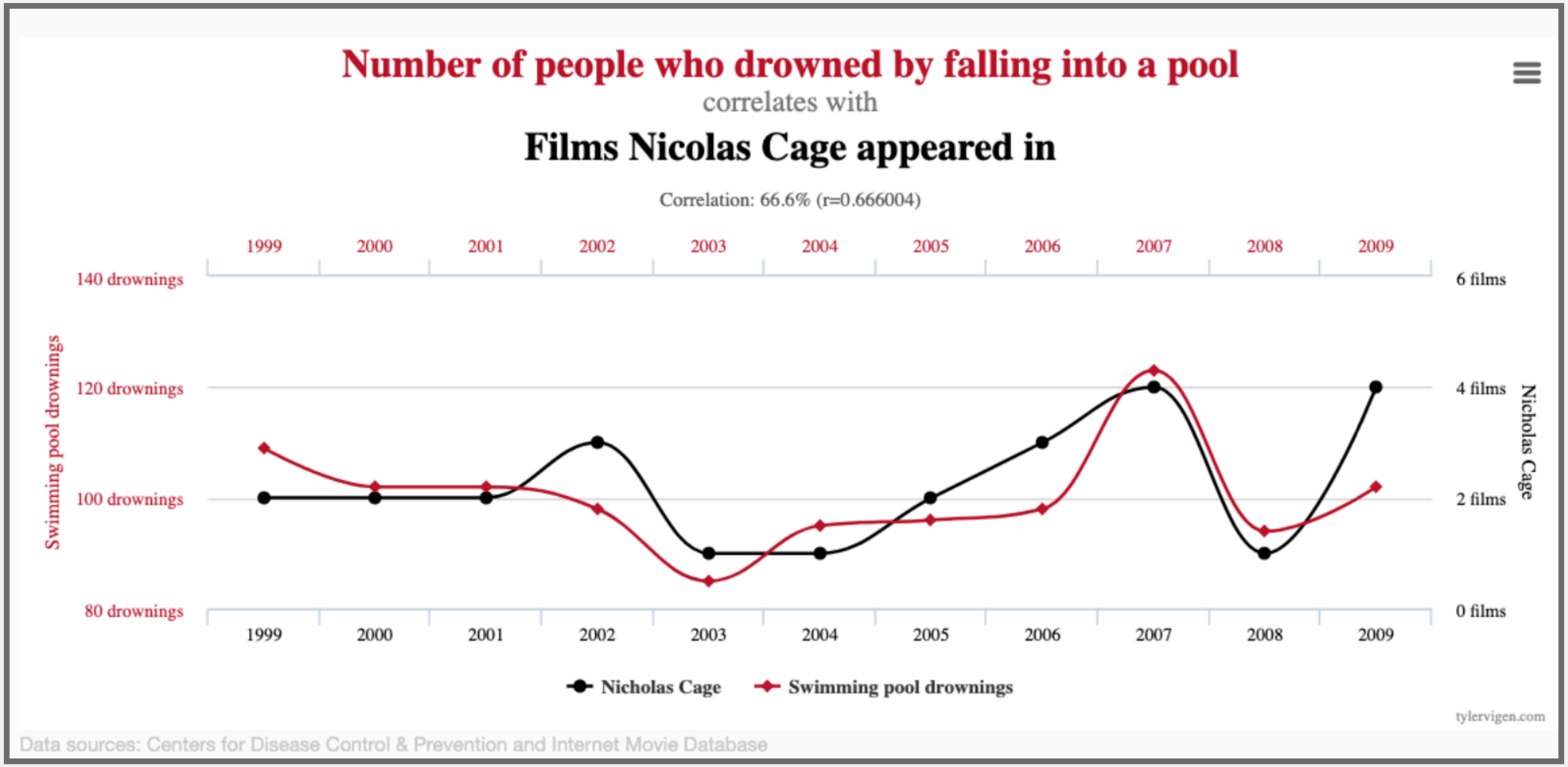
Sources: http://web.stanford.edu/~savage/faculty/savage/FOA%20Index.htm
www.danzigercartoons.com

Making Inferences


http://xkcd.com/552/
I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

- To infer causation:
  - Provide a theory (from domain knowledge, independent of data)
  - Show correlation
  - Demonstrate ability to predict new cases (replicate/validate)

Spurious Correlations



Number of people who drowned by falling into a pool

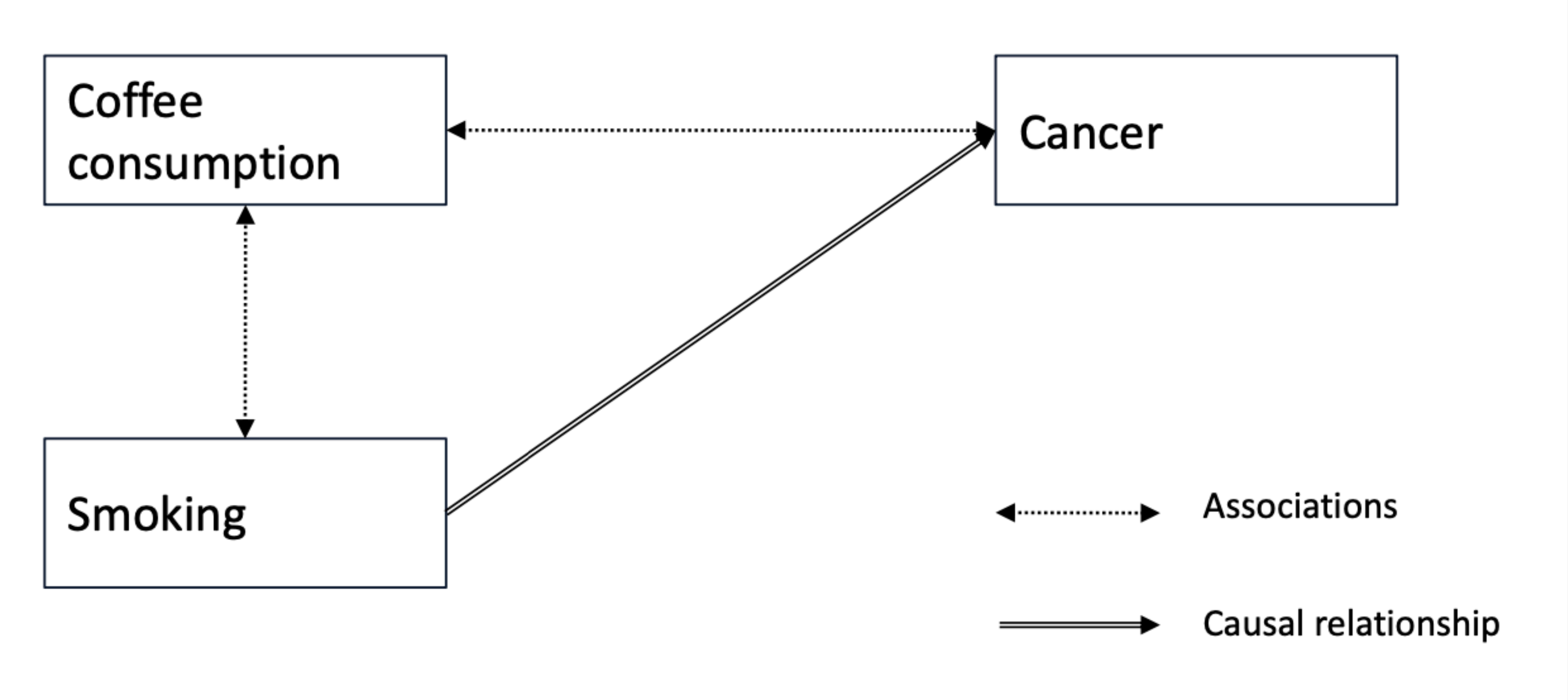
correlates with

Films Nicolas Cage appeared in

Correlation: 66.6% (r=0.666004)

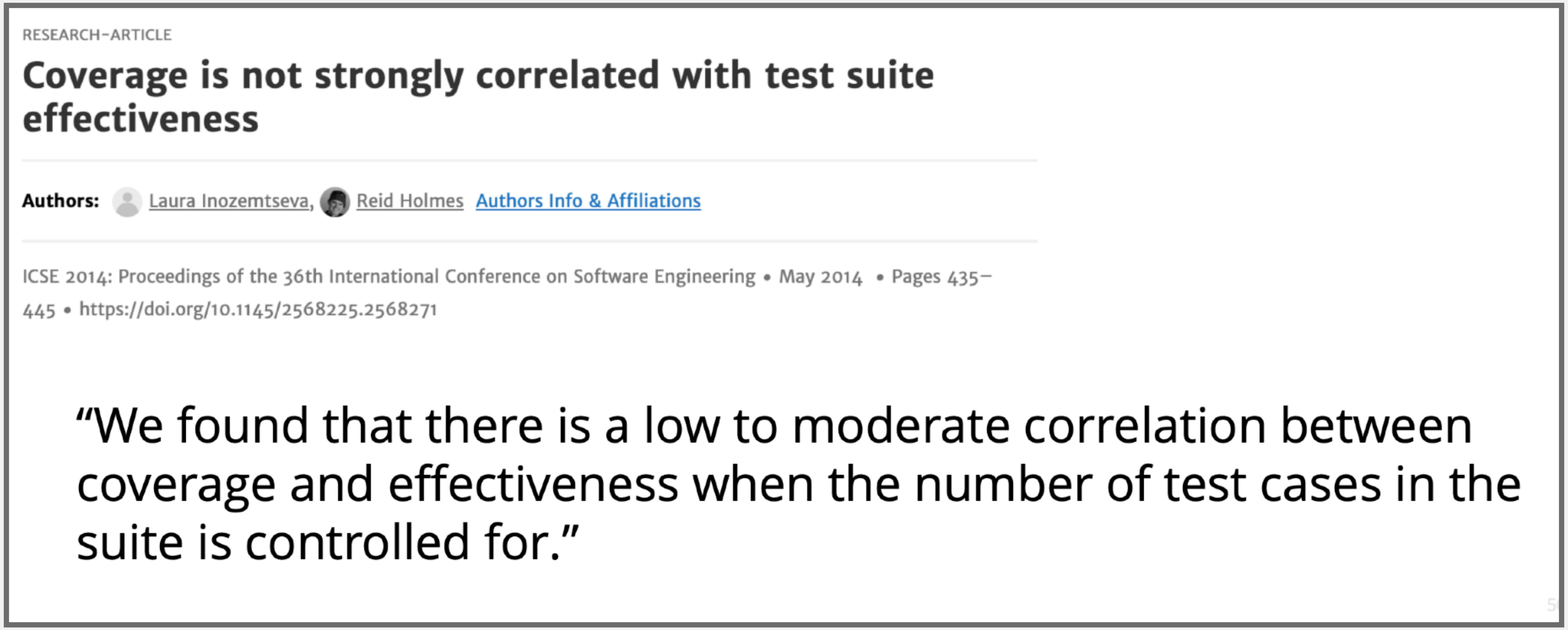
Data sources: Centers for Disease Control & Prevention and Internet Movie Database

Confounding Variables



- If you look only at the coffee consumption → cancer relationship, you can get very misleading results
- Smoking is a confounder

SWE Research



RESEARCH-ARTICLE

Coverage is not strongly correlated with test suite effectiveness

Authors: Laura Inozemtseva, Reid Holmes Authors Info & Affiliations

ICSE 2014: Proceedings of the 36th International Conference on Software Engineering • May 2014 • Pages 435–445 • https://doi.org/10.1145/2568225.2568271

"We found that there is a low to moderate correlation between coverage and effectiveness when the number of test cases in the suite is controlled for."

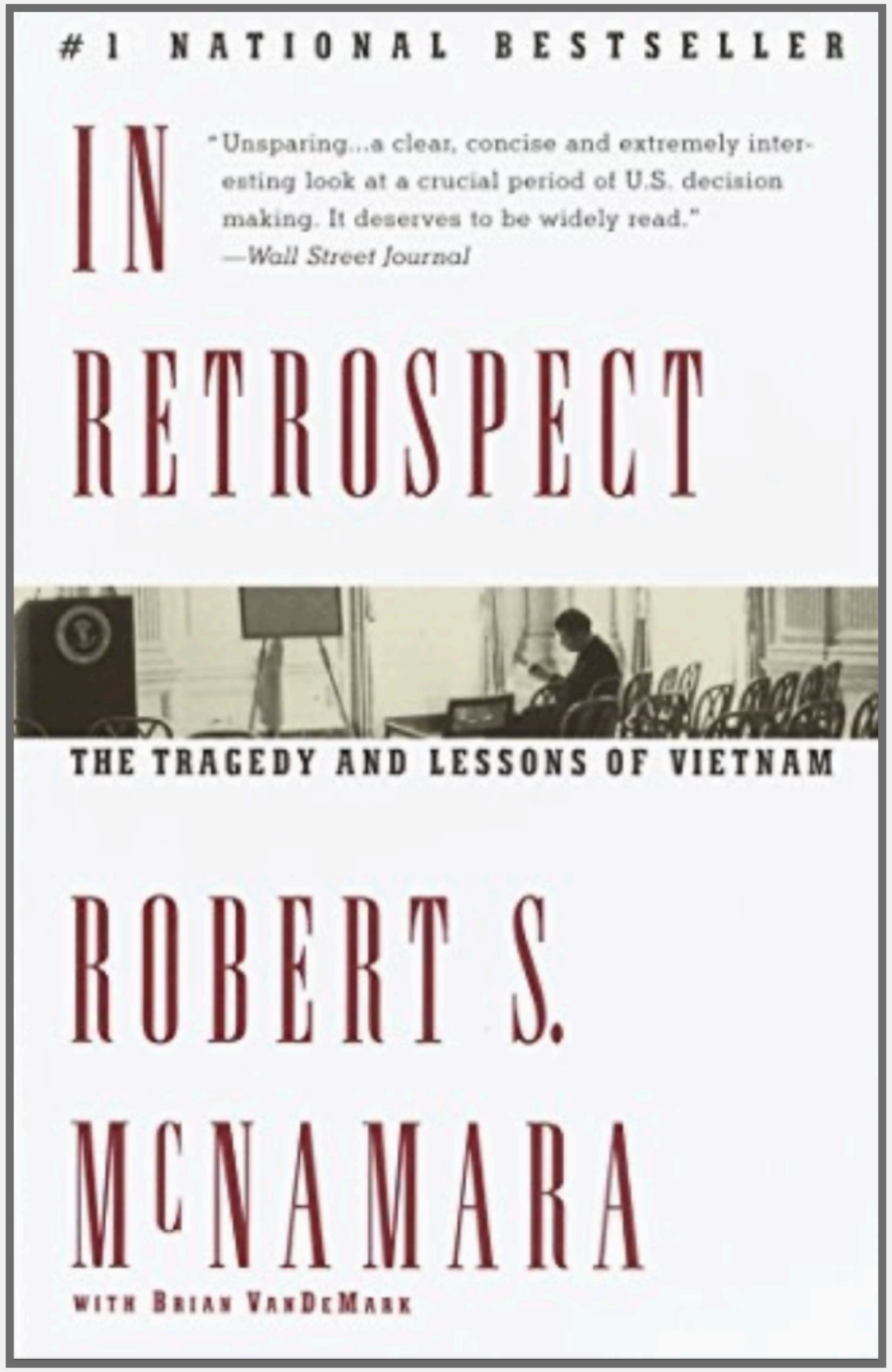
Measurements Validity

- **Construct validity** – Are we measuring what we intended to measure?
- **Internal validity** – The extent to which the measurement can be used to explain some other characteristic of the entity being measured
- **External validity** – Concerns the generalization of the findings to contexts and environments, other than the one studied

Measurements Reliability

- Extent to which a measurement yields similar results when applied multiple times
- Goal is to reduce uncertainty, increase consistency
- Example: Performance
  - Time, memory usage
  - Cache misses, I/O operations, instruction execution count, etc.
- Law of large numbers
  - Taking multiple measurements to reduce error
- Trade-off with cost

McNamara Fallacy



#1 NATIONAL BESTSELLER

IN RETROSPECT

"Unsparing...a clear, concise and extremely interesting look at a crucial period of U.S. decision making. It deserves to be widely read."
—*Wall Street Journal*

THE TRAGEDY AND LESSONS OF VIETNAM

ROBERT S. McNAMARA

WITH BRIAN VANDEMARK

McNamara Fallacy

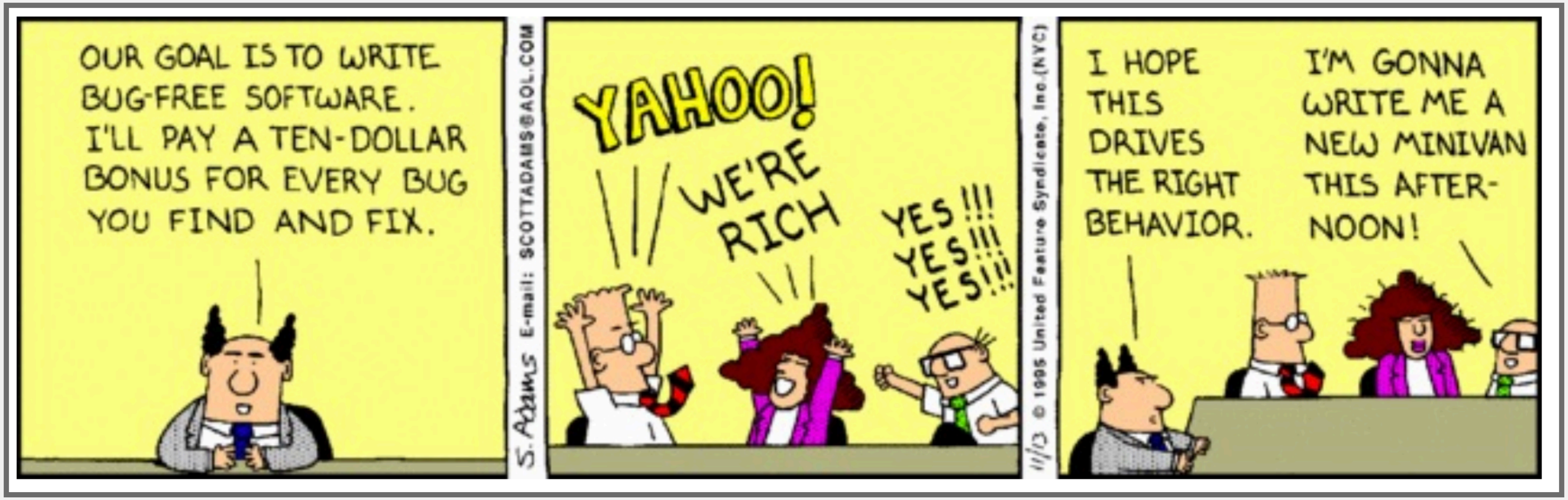
- Measure whatever can be easily measured.
- Disregard that which cannot be measured easily.
- Presume that which cannot be measured easily is not important.
- Presume that which cannot be measured easily does not exist.

McNamara Fallacy

- There seems to be a general misunderstanding to the effect that a mathematical model cannot be undertaken until every constant and functional relationship is known to high accuracy. This often leads to the omission of admittedly highly significant factors (most of the “intangibles” influences on decisions) because these are unmeasured or unmeasurable. To omit such variables is equivalent to saying that they have zero effect... Probably the only value known to be wrong...
  - J. W. Forrester, Industrial Dynamics, The MIT Press, 1961

Metrics & Incentives

- Goodhart’s law: “When a measure becomes a target, it ceases to be a good measure.”


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
11/13 © 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

Simplistic Productivity Measures

- Lines of code per day?
  - Industry average 10-50 lines/day
  - Debugging + rework ca. 50% of time

- • Function/object/application points per month • Bugs fixed?
  - Milestones reached?

Incentivizing Productivity

- What happens when developer bonuses are based on
  - Lines of code per day?
  - Amount of documentation written?
  - Low number of reported bugs in their code?
  - Low number of open bugs in their code?
  - High number of fixed bugs?
  - Accuracy of time estimates?

Developer Productivity Myths

- Productivity is all about developer activity
- Productivity is only about individual performance
- One productivity metric can tell us everything
- Productivity measures are useful only for managers
- Productivity is only about engineering systems and developer tools

WARNING!!

- Most software metrics are controversial
  - Usually only plausibility arguments, rarely rigorously validated
  - Cyclomatic complexity was repeatedly refuted, yet is still used
  - “Similar to the attempt of measuring the intelligence of a person in terms of the weight or circumference of the brain”
- Use carefully!
- Code size dominates many metrics
- Avoid claims about human factors (e.g., readability) and quality, unless validated
- Calibrate metrics in project history and other projects
- Metrics can be gamed; you get what you measure

Summary

- Measurement is difficult but important for decision making
- Software metrics are easy to measure but hard to interpret,
  validity often not established
- Many metrics exist, often composed; pick or design suitable metrics if needed
- Careful in use: monitoring vs incentives
- Strategies beyond metrics

Questions to Consider for Your Projects

- What properties do we care about and how do we measure them?
- What is being measured? Does it (to what degree) capture the thing you care about? What are its limitations?
- How should it be incorporated into process?
- What are potentially negative side effects or incentives?

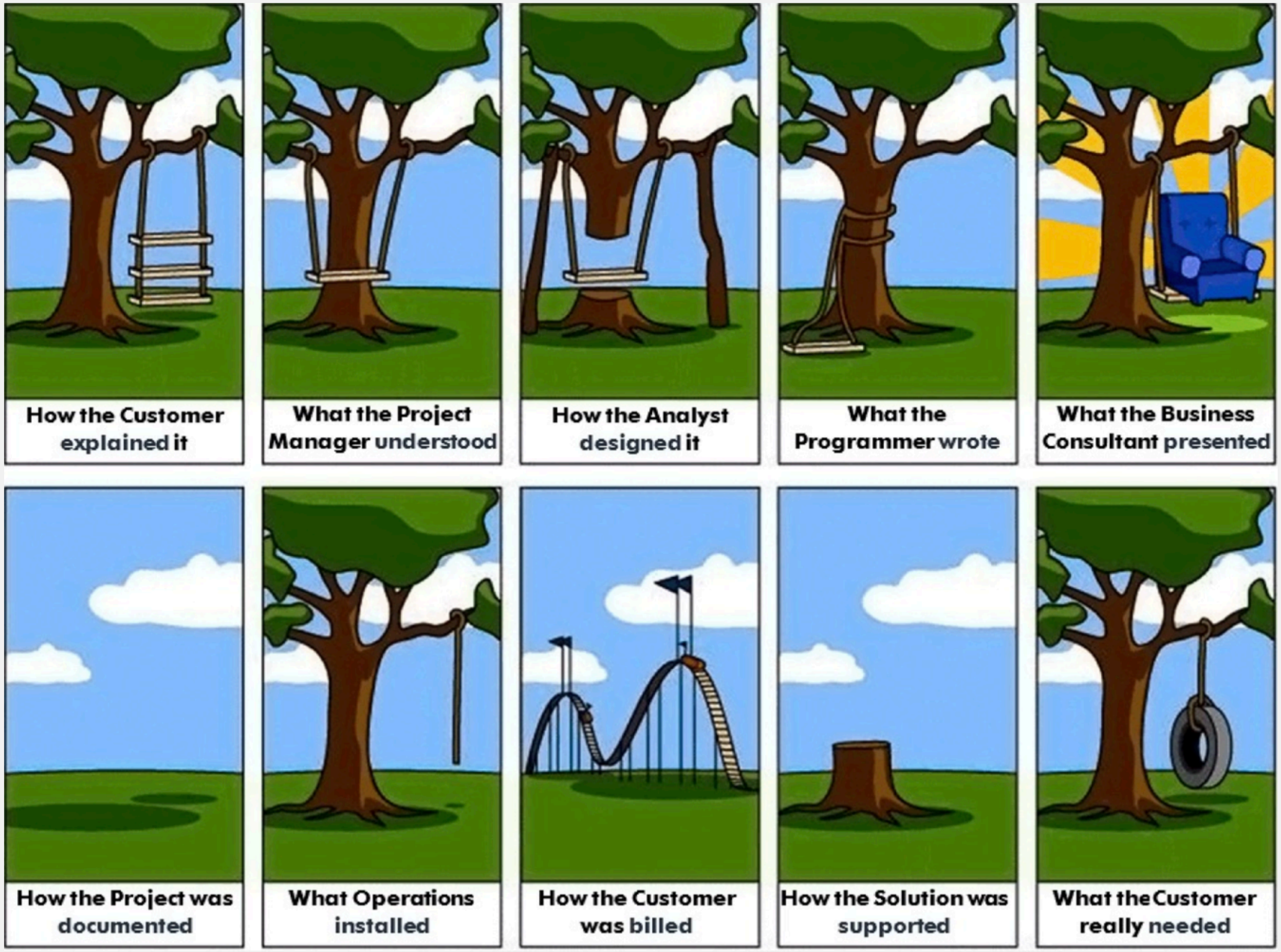
Project Planning & Agile Development

Learning Goals

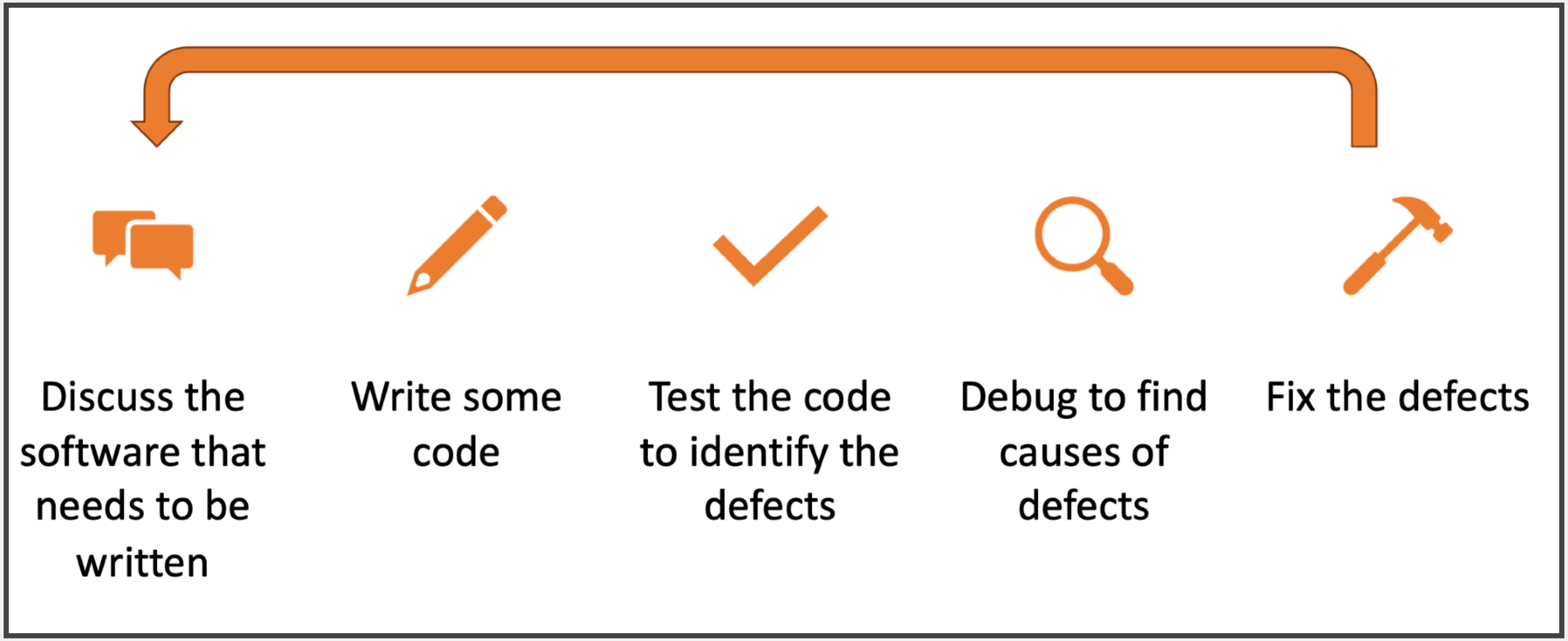
- Recognize the importance of project planning
- Understand the difficulty of measuring progress
- Identify why software development has project characteristics
- Use milestones for planning and progress measurement
- Understand backlogs and user stories
- Get to know your team!

Software Process

- “The set of activities and associated results that produce a software product.”


How the Customer explained it
What the Project Manager understood
How the Analyst designed it
What the Programmer wrote
What the Business Consultant presented
How the Project was documented
What Operations installed
How the Customer was billed
How the Solution was supported
What the Customer really needed

All Software Development Processes



Discuss the software that needs to be written

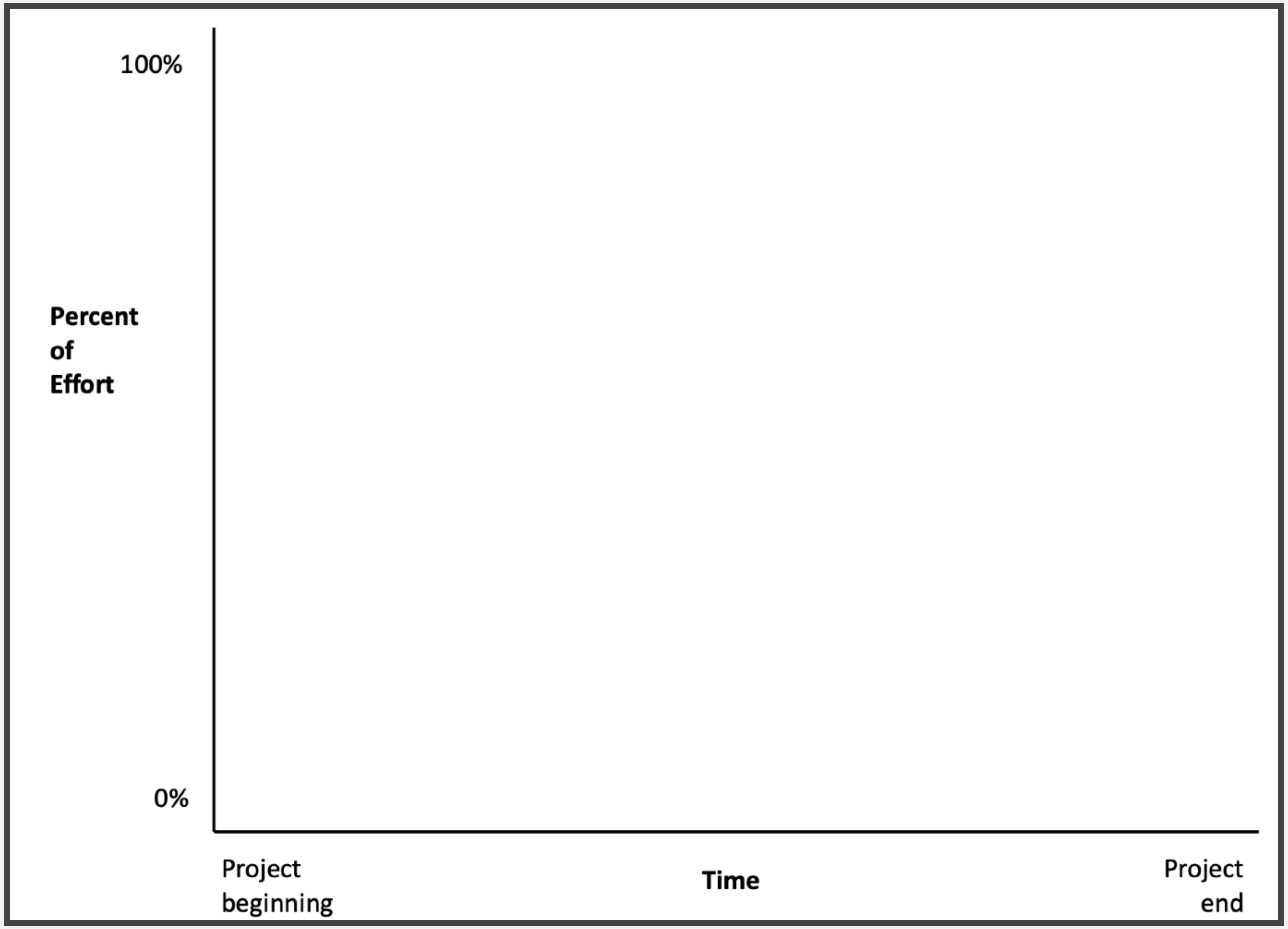
Write some code

Test the code to identify the defects

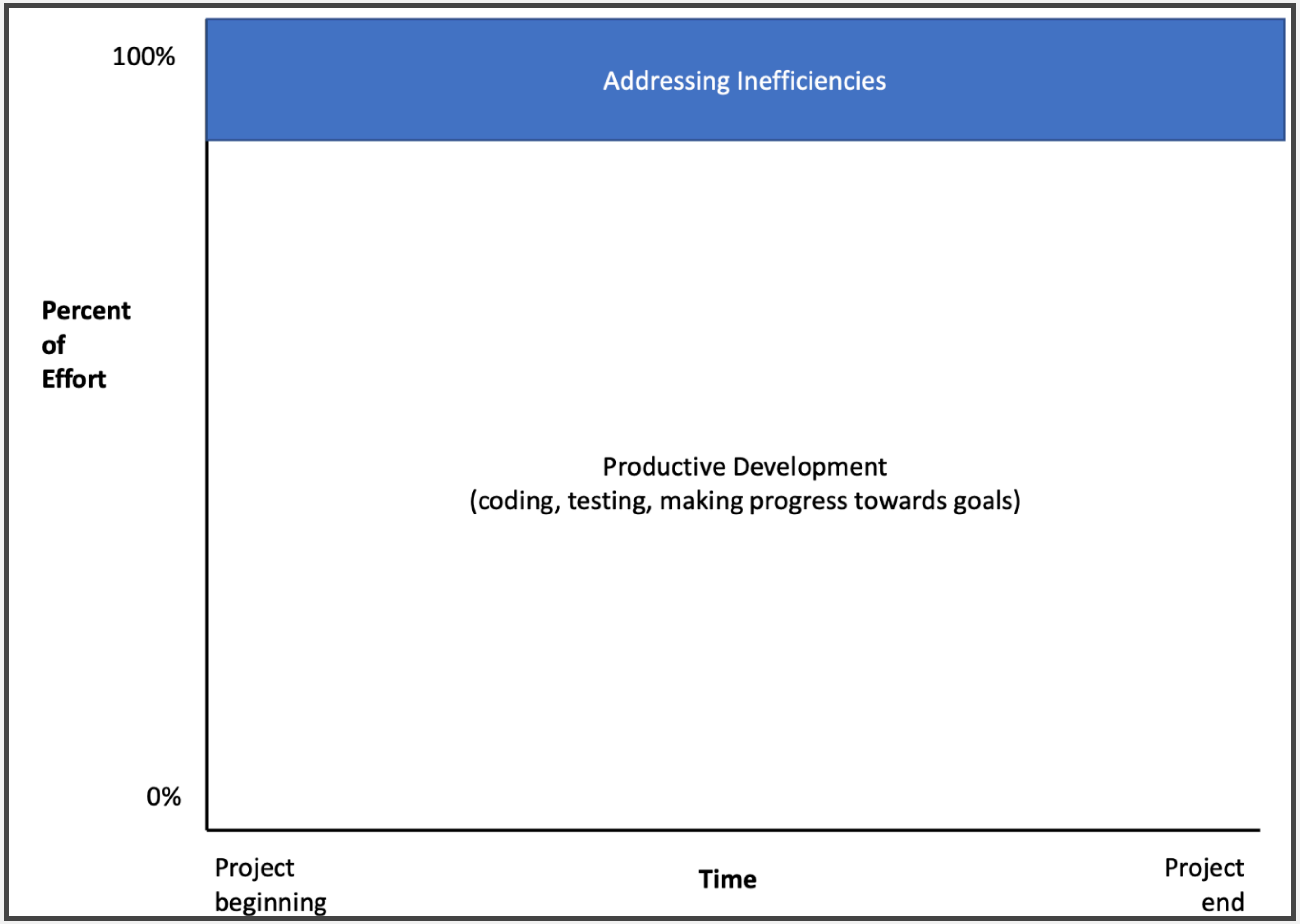
Debug to find causes of defects

Fix the defects

Effort Spent During the Process



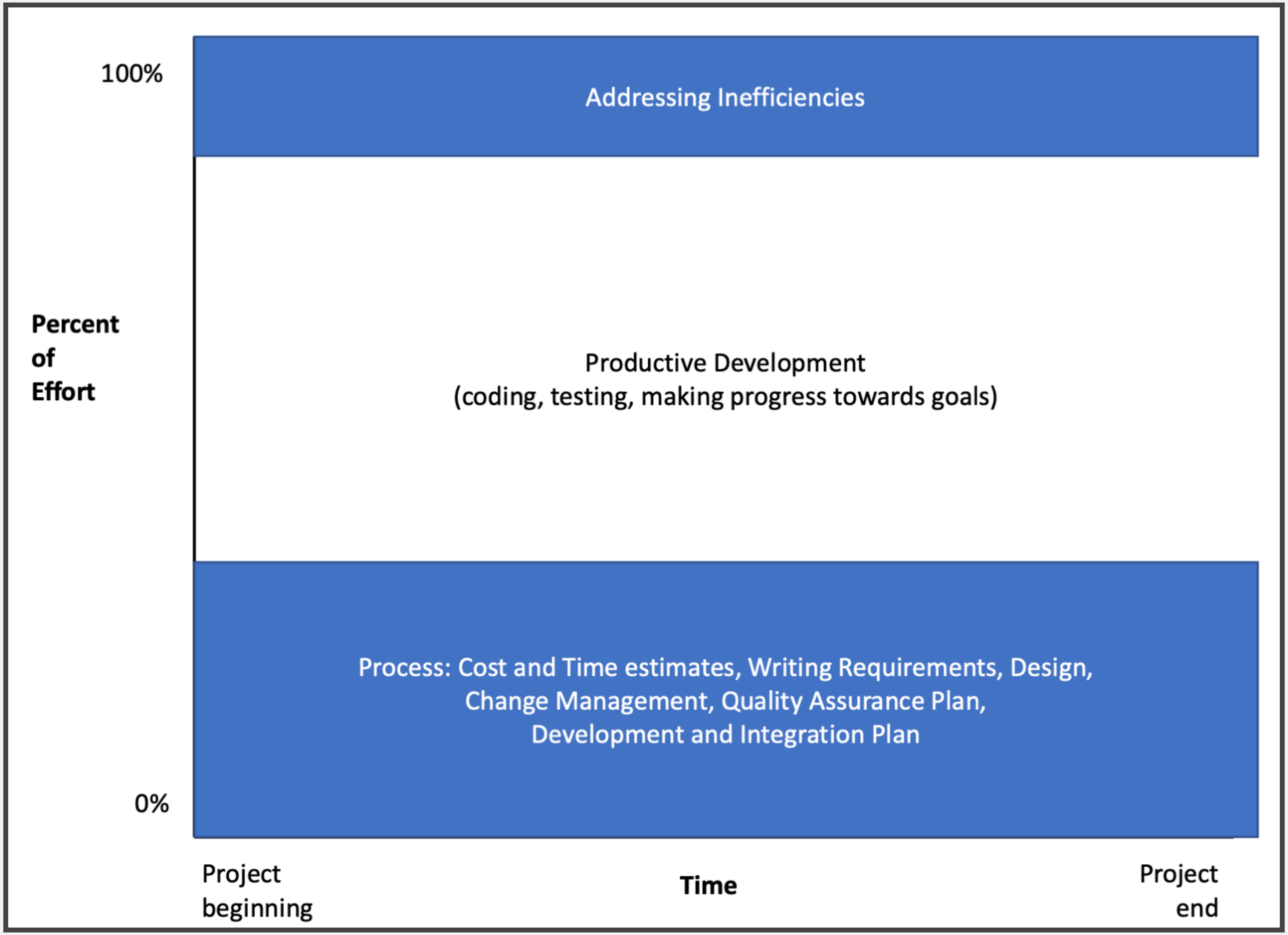
Effort Spent During the Process



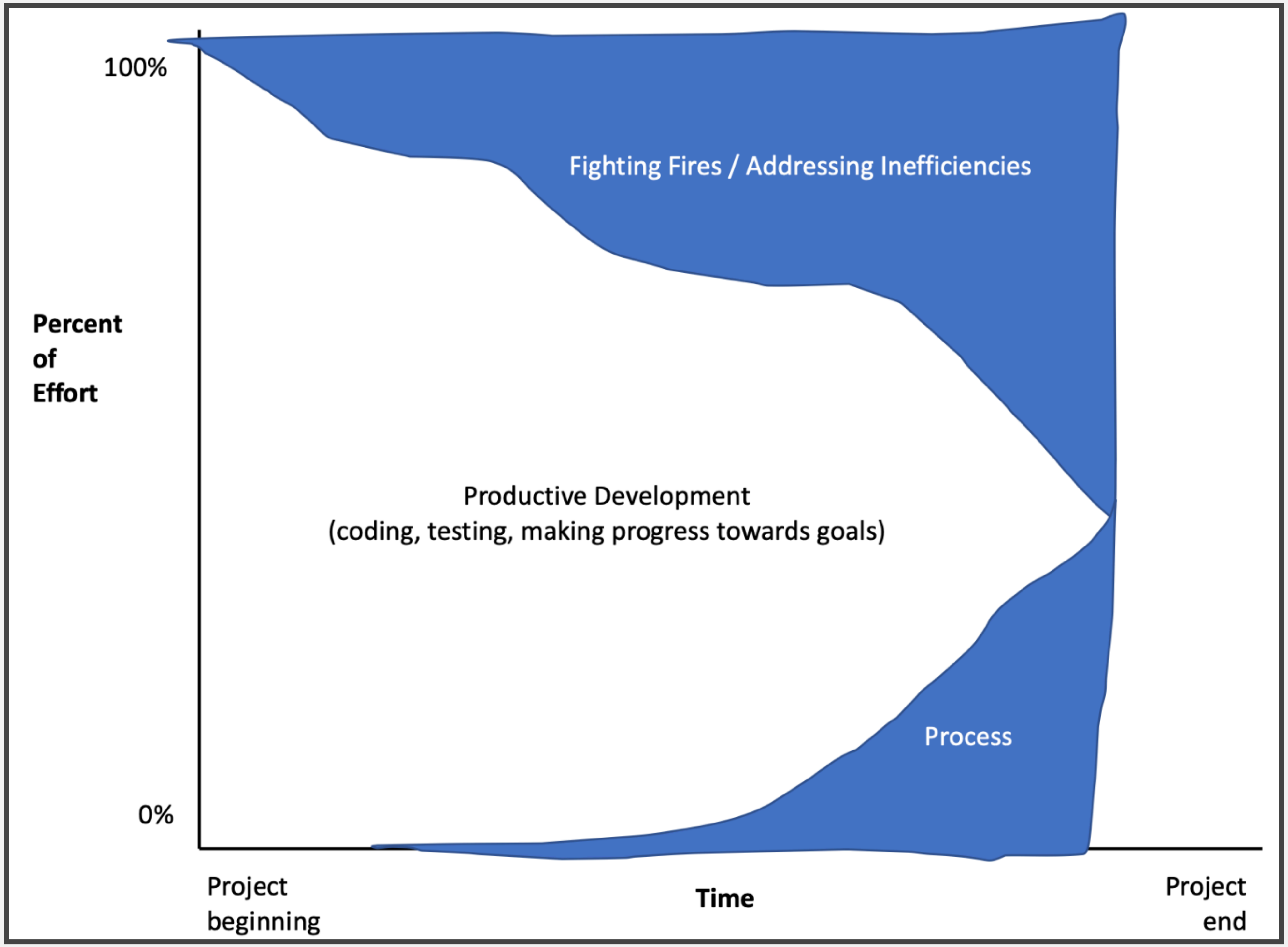
Let’s Improve the Reliability of this Process

- Write down all requirements
  - Review requirements
  - Require approval for all changes to requirements
- Use version control for all changes
  - Review code
- Track all work items
  - Break down feature development into small tasks
  - Write down and monitor all reported bugs
- Hold regular, frequent status meetings
  - Plan and conduct quality assurance
  - Employ a DevOps framework to push code between developers and operations

Effort Spent During the Process



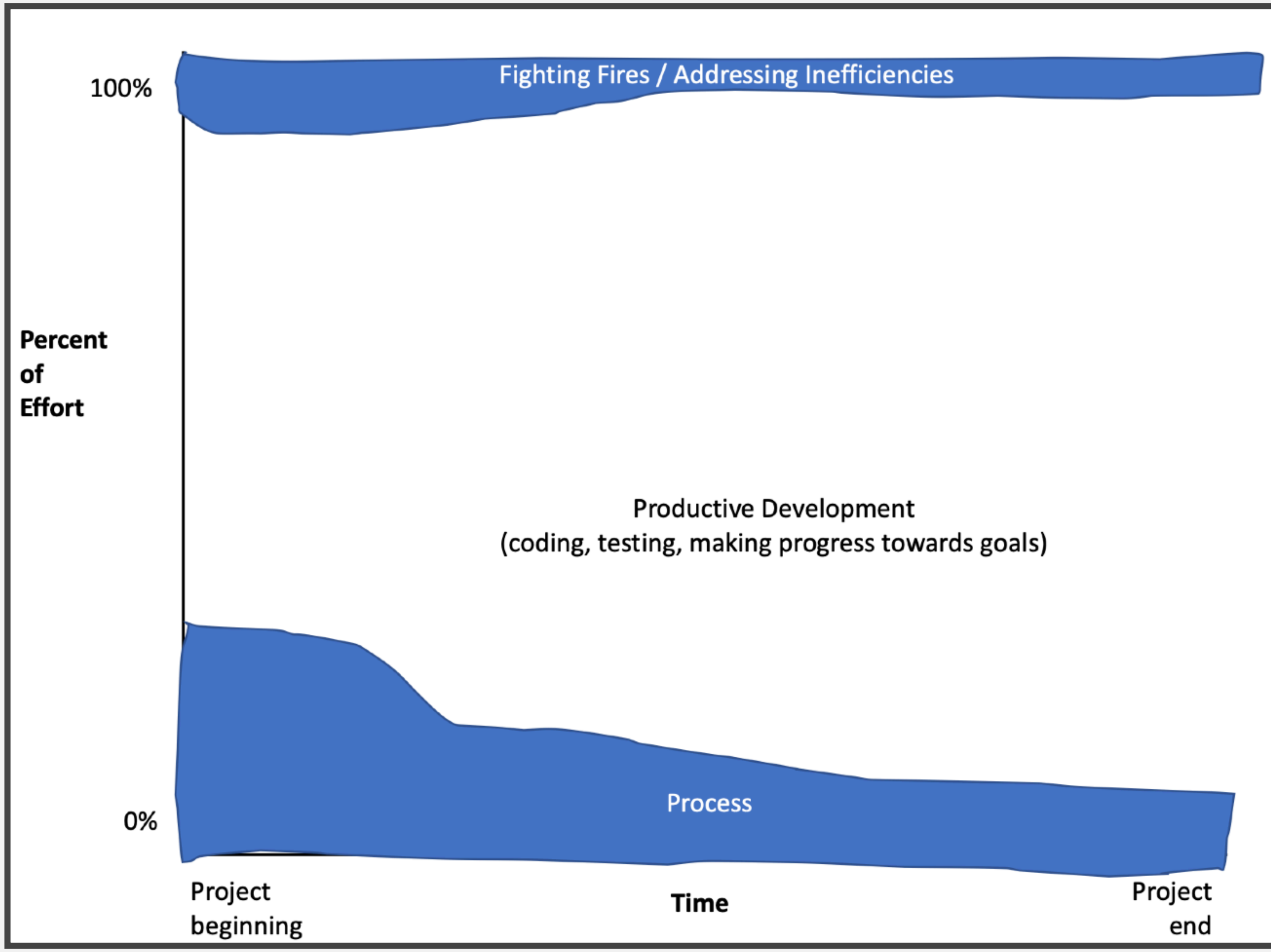
Effort Spent During the Process



Example Process Issues

- **Change Control:** Mid-project informal agreement to changes suggested by customer. Project scope expands 25-50%
- **Quality Assurance:** Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- **Defect Tracking:** Bug reports collected informally. Bugs are overlooked.
- **System Integration:** Integration of independently developed components at the very end of the project. Interfaces out of sync.
- **Source Code Control:** Accidentally overwrote changes. Lost work.
- **Scheduling:** Late project. Developers asked to re-estimate work effort weekly.

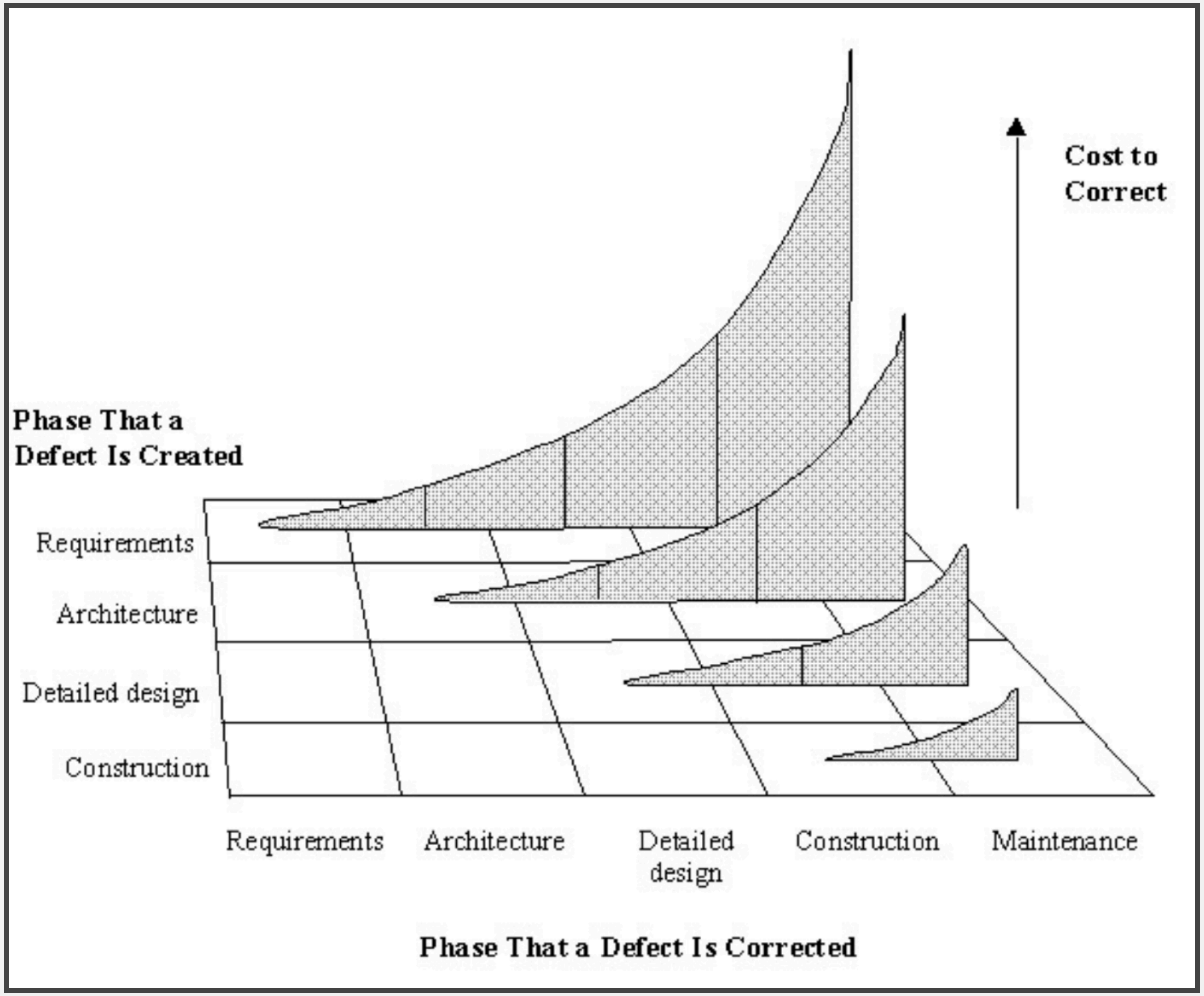
Effort Spent During the Process



Hypothesis: Process increases flexibility and efficiency

Ideal Curve: Upfront investment for later greater returns

Defect Correction Effort



Planning

Time Estimation

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

https://xkcd.com/612/

Activity: Estimate Time

- **Task A:** Web version of the Monopoly board game with Orlando street names
  - **Team:** just you

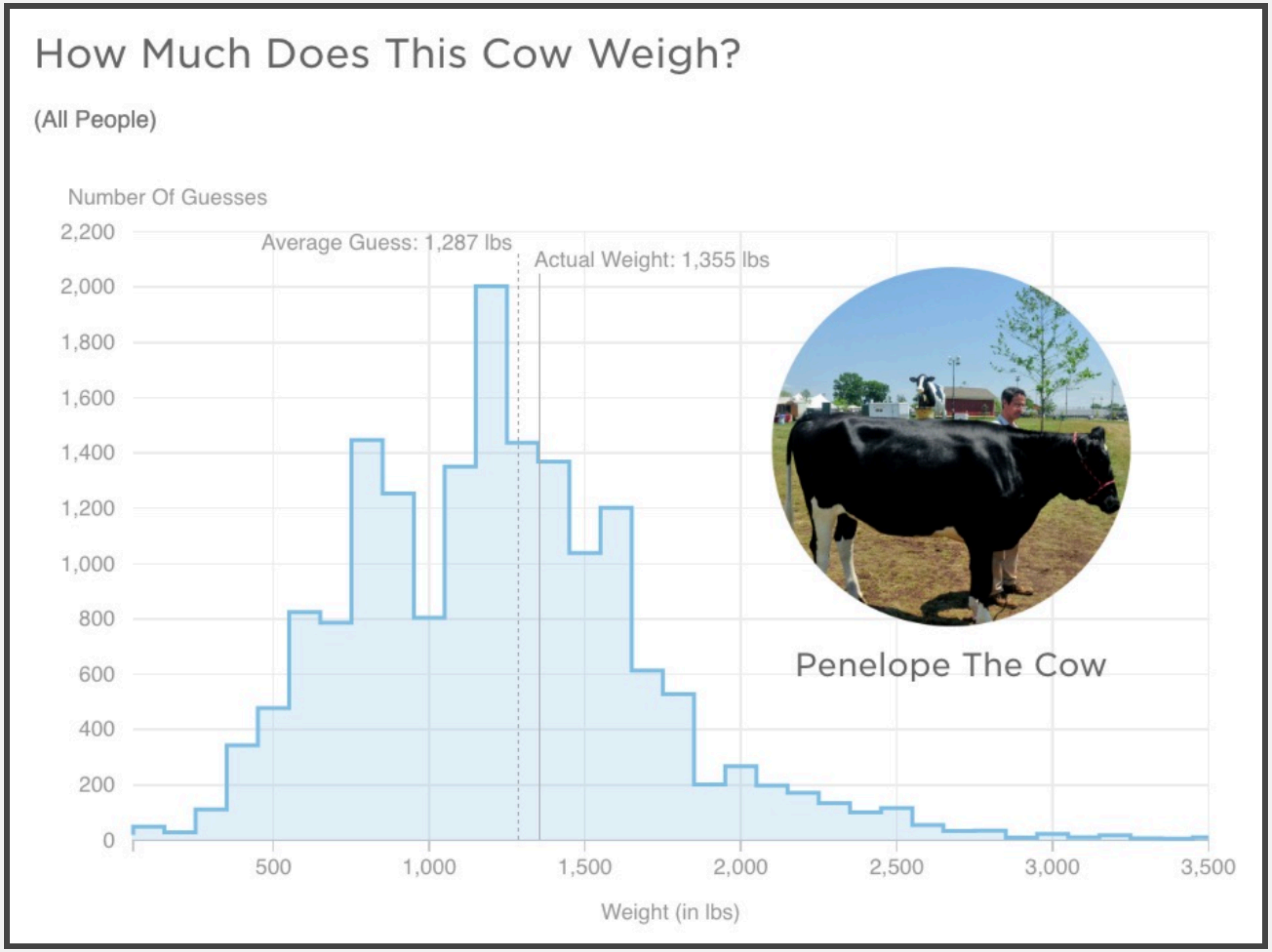
- **Task B:** Bank smartphone app
  - **Team:** you with team of 4 developers, one experienced with iPhone apps, one with background in security

- **Estimate:** 8h days, 20 workdays in a month, 220 workdays per year

Revise Time Estimate

- Do you have comparable experience to base an estimate on?
- How much design do you need for each task?
- How much testing time do you need for each task?
- Let's break down the task into ~5 smaller tasks and estimate
  their lengths.
- Revise our overall estimate, if necessary

Wisdom of the Crowd



Measuring Progress

- “I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”

Measuring Progress

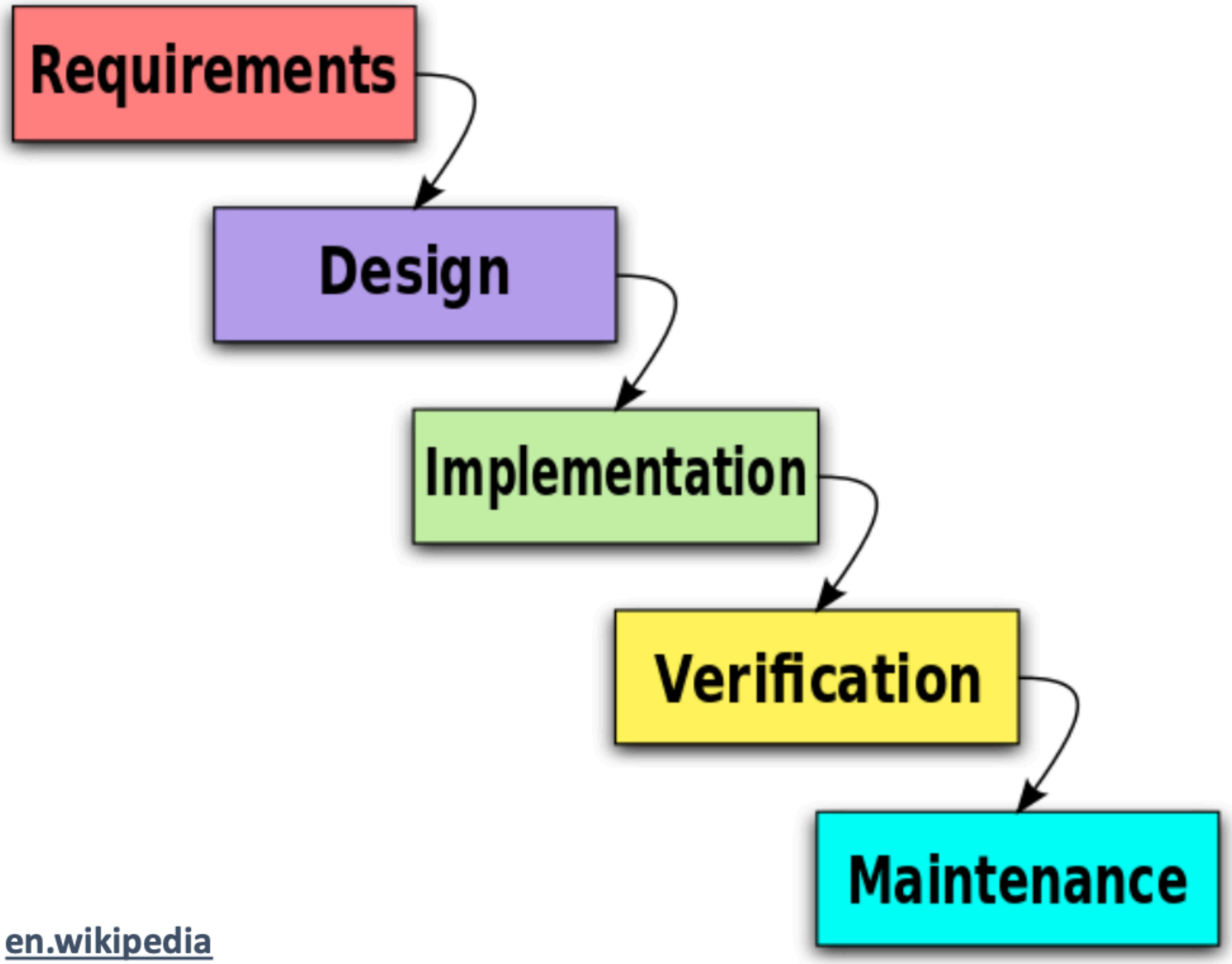
- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?

Milestones and Deliverables Make Progress Observable

- **Milestone:** clear end point of a (sub)tasks
  - For project manager
  - Reports, prototypes, completed subprojects
  - “80% done“ is not a suitable mile stone
- **Deliverable:** Result for customer
  - Similar to a milestone, but for customers
  - Reports, prototypes, completed subsystems

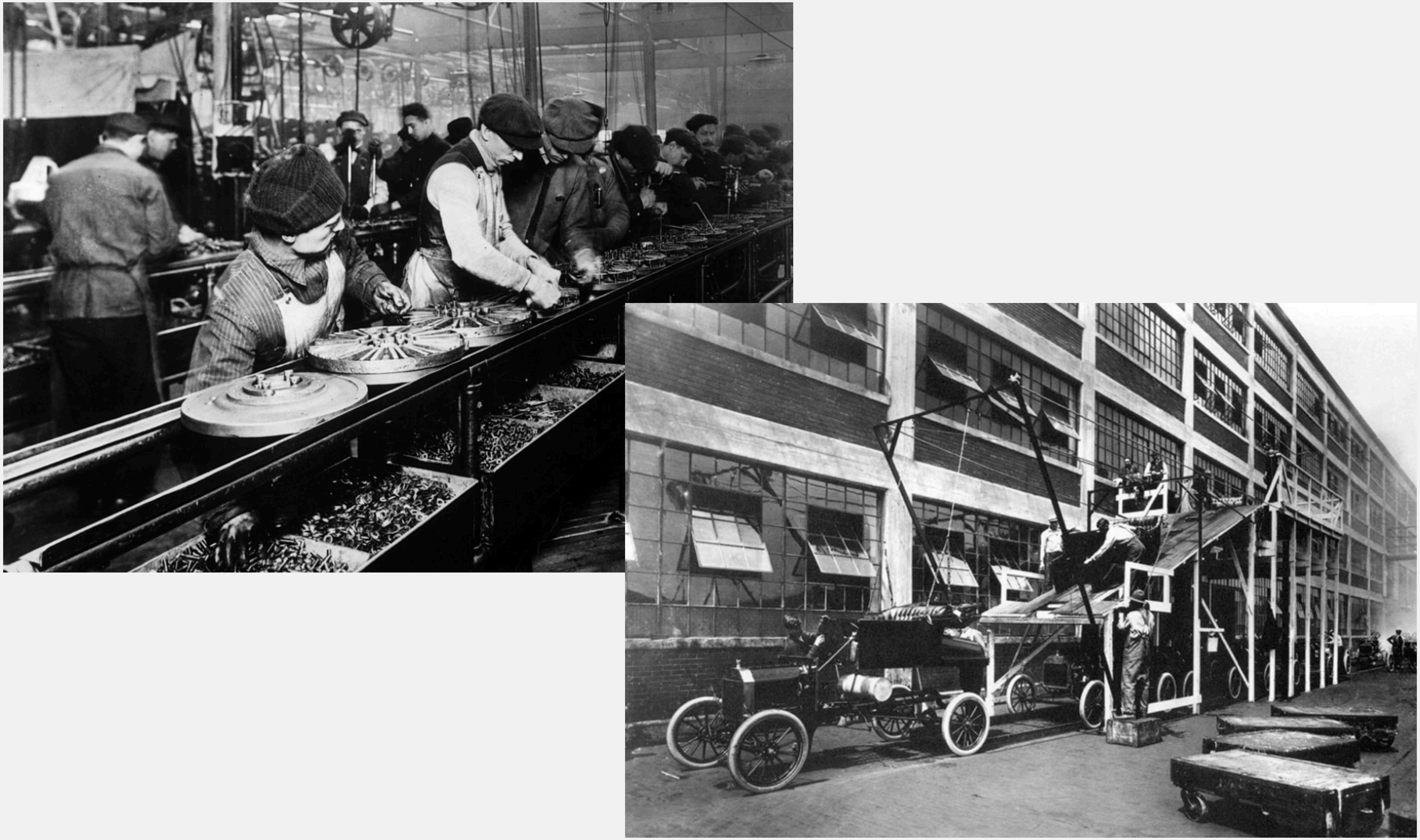
Processes

Waterfall was the OG Software Process



Waterfall diagram CC-BY 3.0 Paulsmith99 at en.wikipedia

Akin to Processes Pioneered in Auto Manufacturing by Ford

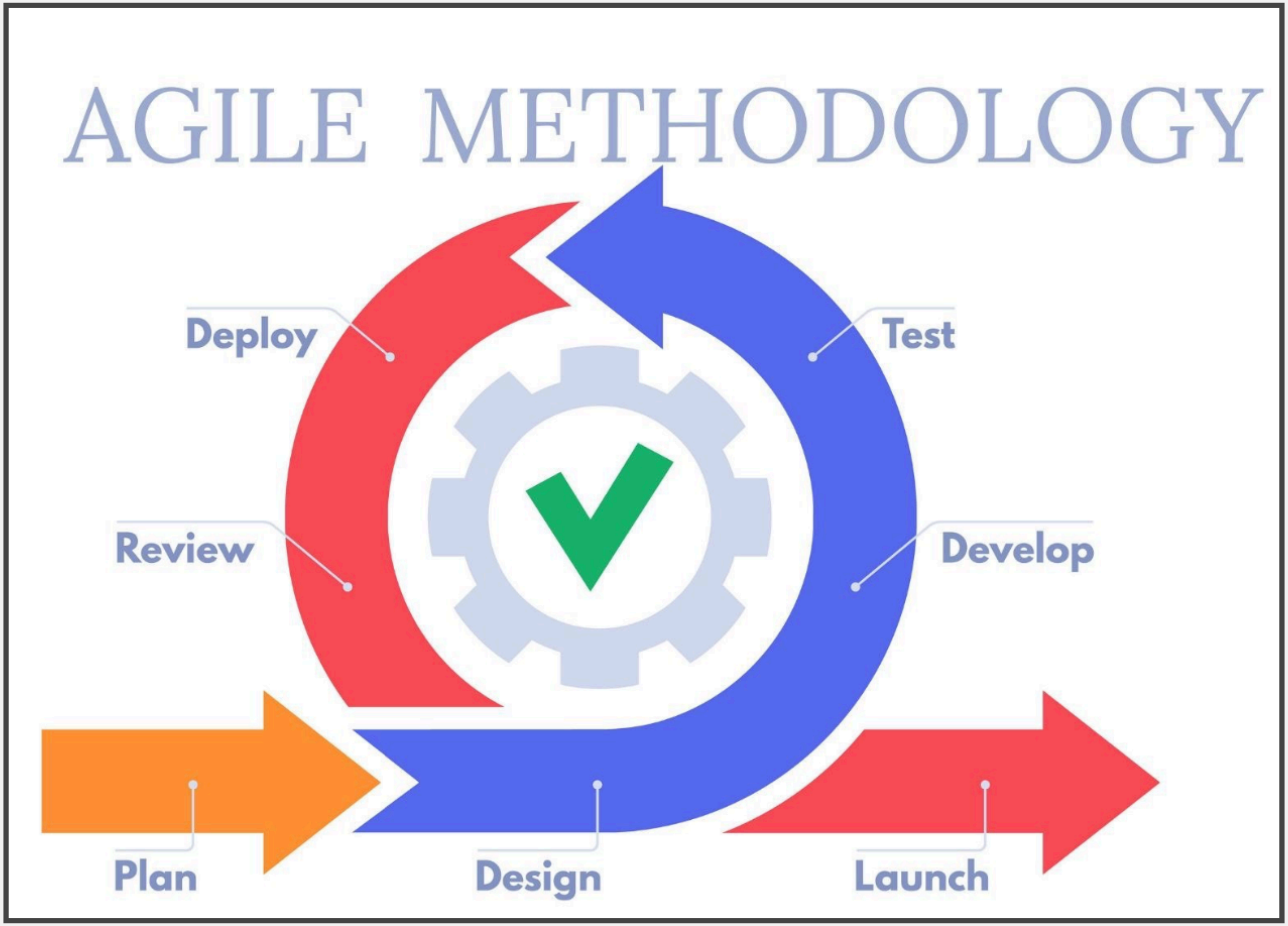


LEAN Production Adapts to Variable Demand

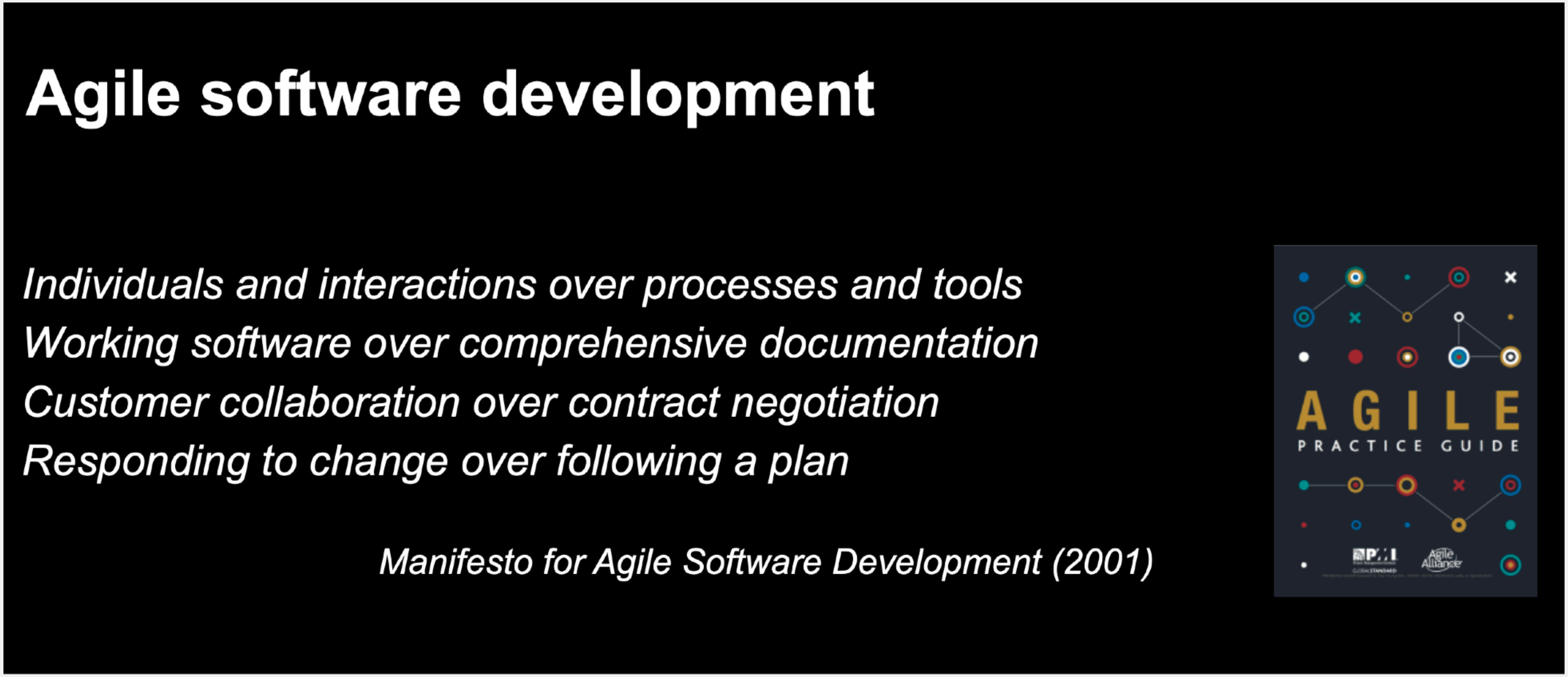
Taiichi Ohno

- Toyota Production System (TPS)
  - Build only what is needed, only when it is needed.
  - Use the “pull” system to avoid overproduction (Kanban)
  - Stop to fix problems, to get quality right from the start (Jidoka)
  - Workers are multi-skilled and understand the whole process; take ownership
- Lots of recent software buzzwords build on these ideas
  - Just-in-time, DevOps, Shift-Left

Now, Most Teams use some form of Agile Methods



Now, Most Teams use some form of Agile Methods



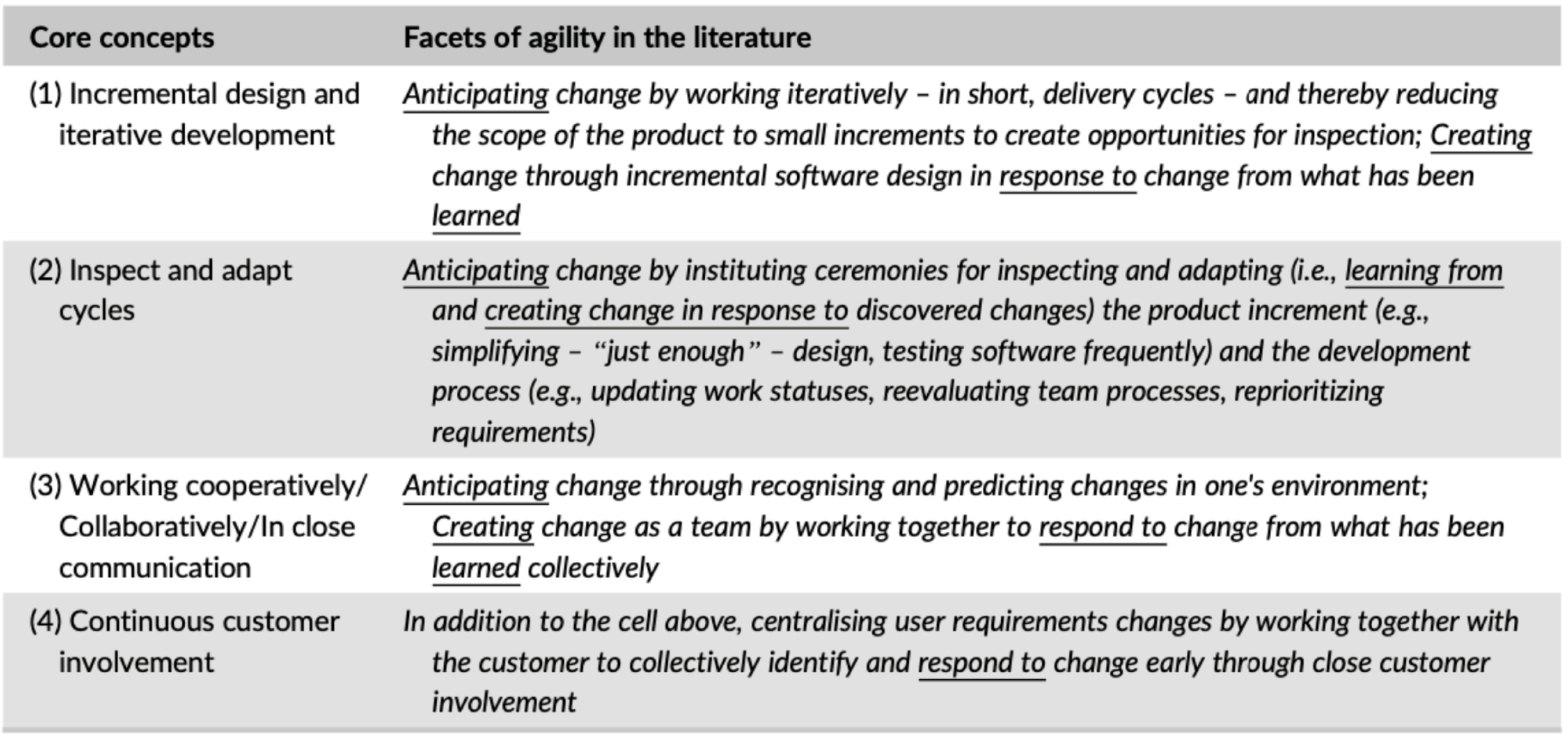
Agile software development

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

Manifesto for Agile Software Development (2001)

AGILE
PRACTICE GUIDE
PMI
Agile Alliance

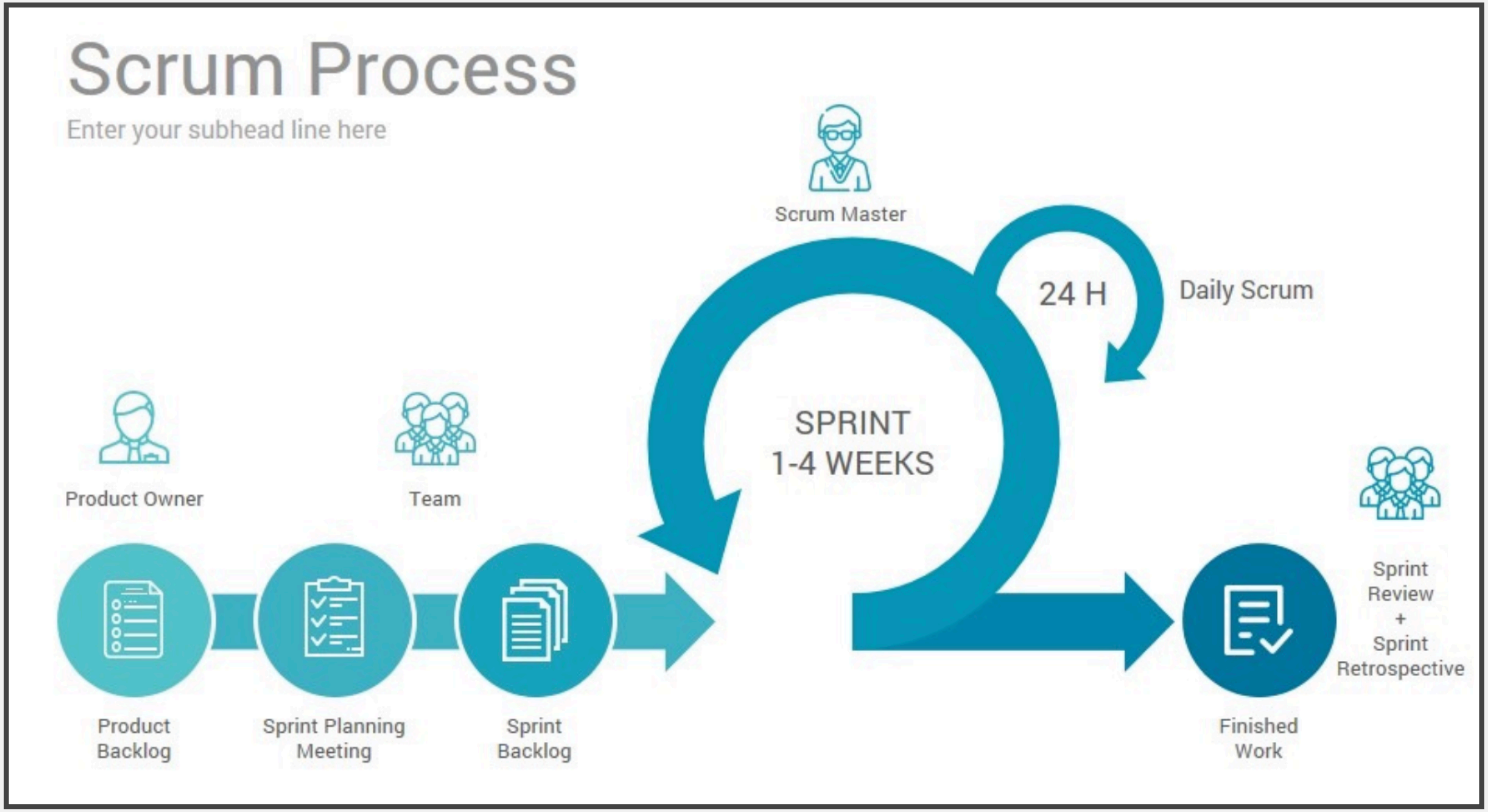
Core Concepts in Agile



Core concepts	Facets of agility in the literature
(1) Incremental design and iterative development	*Anticipating change by working iteratively – in short, delivery cycles – and thereby reducing the scope of the product to small increments to create opportunities for inspection; Creating change through incremental software design in response to change from what has been learned*
(2) Inspect and adapt cycles	*Anticipating change by instituting ceremonies for inspecting and adapting (i.e., learning from and creating change in response to discovered changes) the product increment (e.g., simplifying – "just enough" – design, testing software frequently) and the development process (e.g., updating work statuses, reevaluating team processes, reprioritizing requirements)*
(3) Working cooperatively/ Collaboratively/In close communication	*Anticipating change through recognising and predicting changes in one's environment; Creating change as a team by working together to respond to change from what has been learned collectively*
(4) Continuous customer involvement	*In addition to the cell above, centralising user requirements changes by working together with the customer to collectively identify and respond to change early through close customer involvement*

Scrum

Elements of Scrum



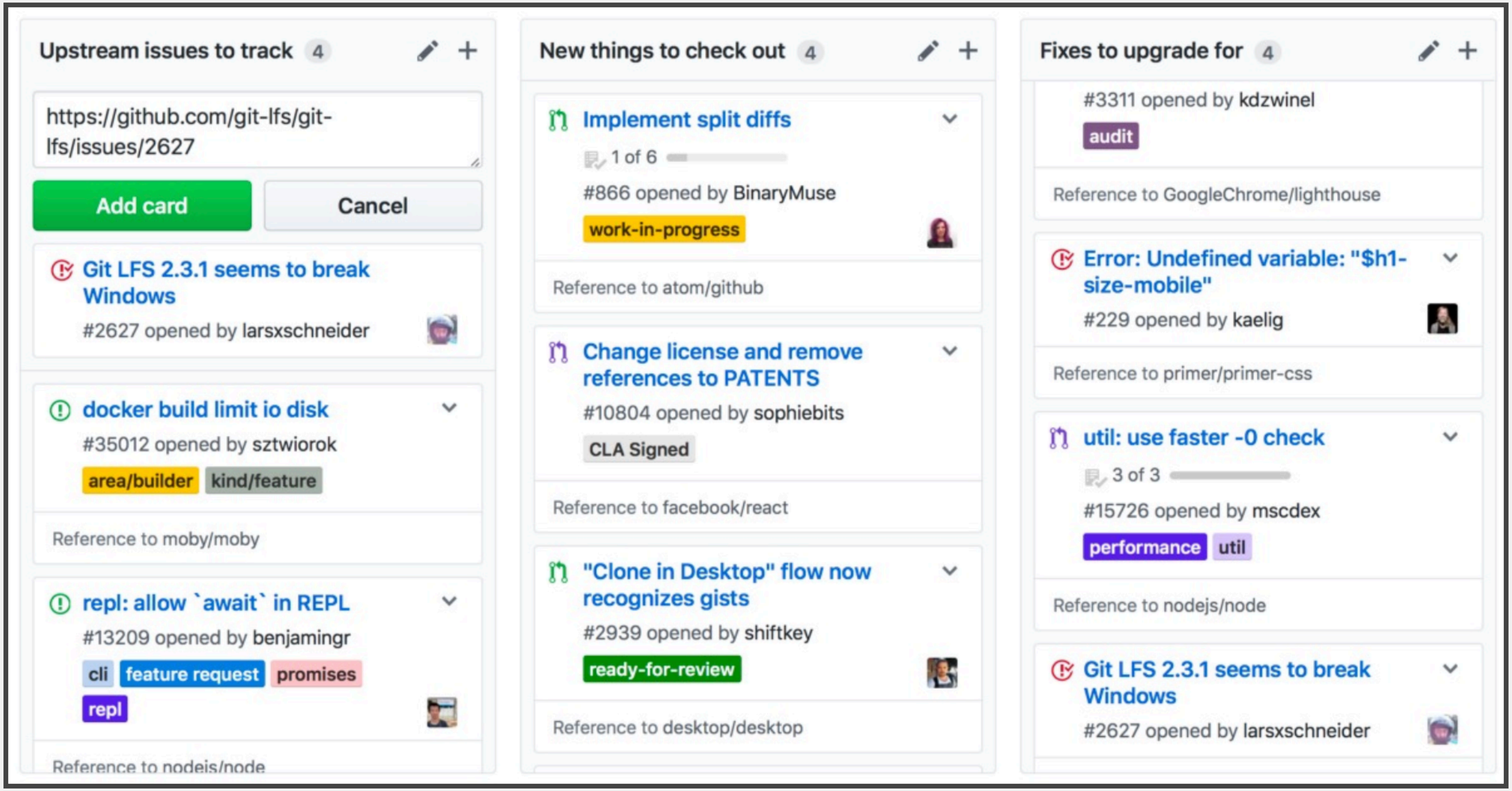
Scrum Process

Enter your subhead line here

Backlogs

- The product backlog is all the features for the product
- The sprint backlog is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:
    - Fine-grained
    - Estimated
    - Assigned to individual team members
    - Acceptance criteria should be defined
- User Stories are often used

Kanban Boards



Upstream issues to track 4

https://github.com/git-lfs/git-lfs/issues/2627

Git LFS 2.3.1 seems to break Windows
#2627 opened by larsxschneider

docker build limit io disk
#35012 opened by sztwiorok
area/builder kind/feature
Reference to moby/moby

repl: allow `await` in REPL
#13209 opened by benjamingr
cli feature request promises
repl
Reference to nodejs/node

New things to check out 4

Implement split diffs
1 of 6
#866 opened by BinaryMuse
work-in-progress
Reference to atom/github

Change license and remove references to PATENTS
#10804 opened by sophiebits
CLA Signed
Reference to facebook/react

"Clone in Desktop" flow now recognizes gists
#2939 opened by shiftkey
ready-for-review
Reference to desktop/desktop

Fixes to upgrade for 4

#3311 opened by kdzwinel
audit
Reference to GoogleChrome/lighthouse

Error: Undefined variable: "$h1-size-mobile"
#229 opened by kaelig
Reference to primer/primer-css

util: use faster -0 check
3 of 3
#15726 opened by mscdex
performance util
Reference to nodejs/node

Git LFS 2.3.1 seems to break Windows
#2627 opened by larsxschneider

Scrum Meetings

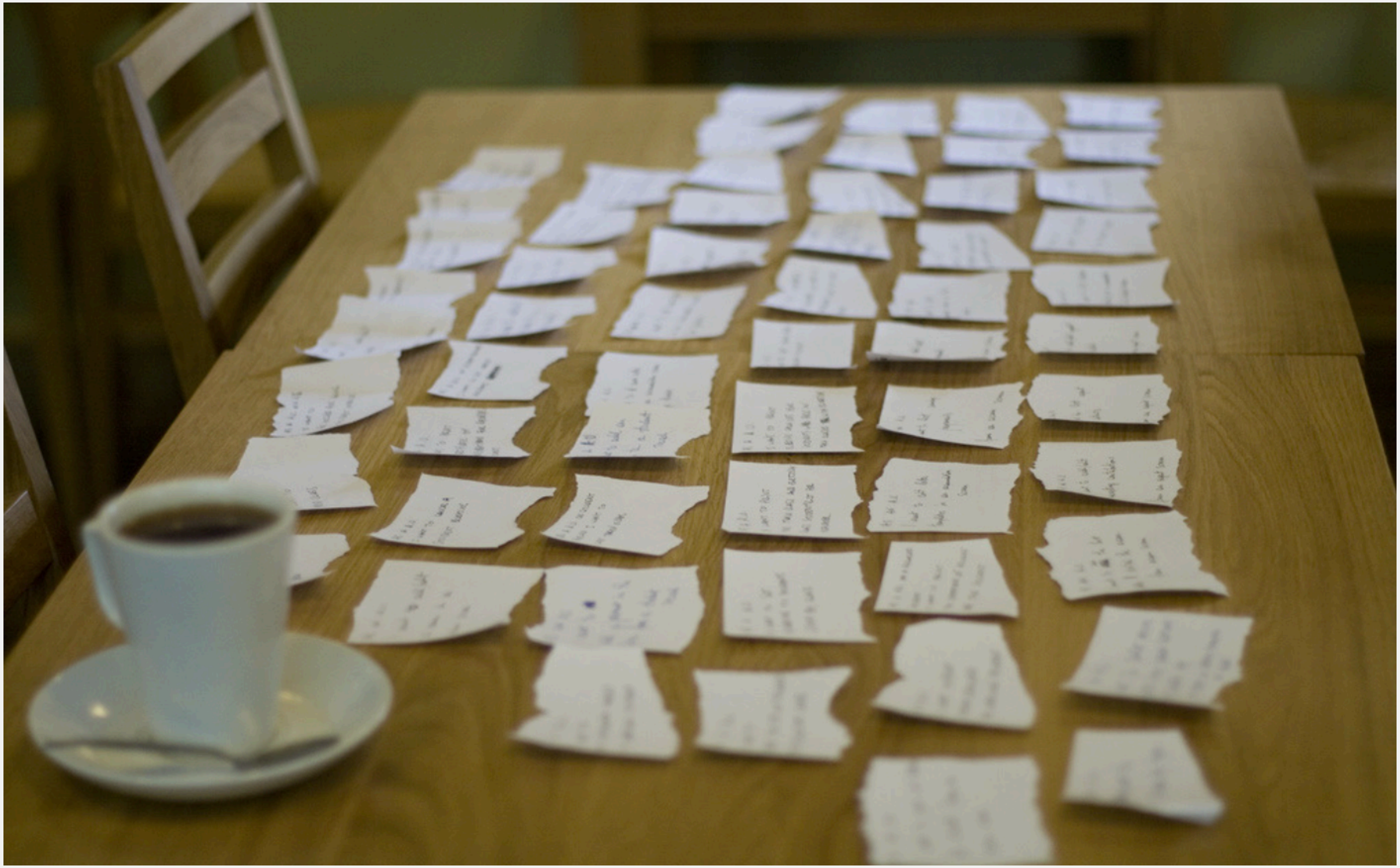
- Sprint Planning Meeting
  - Entire Team decides together what to tackle for that sprint

- Daily Scrum Meeting
  - Quick Meeting to touch base on :
    - What have I done?
    - What am I doing next?
    - What am I stuck on/need help?

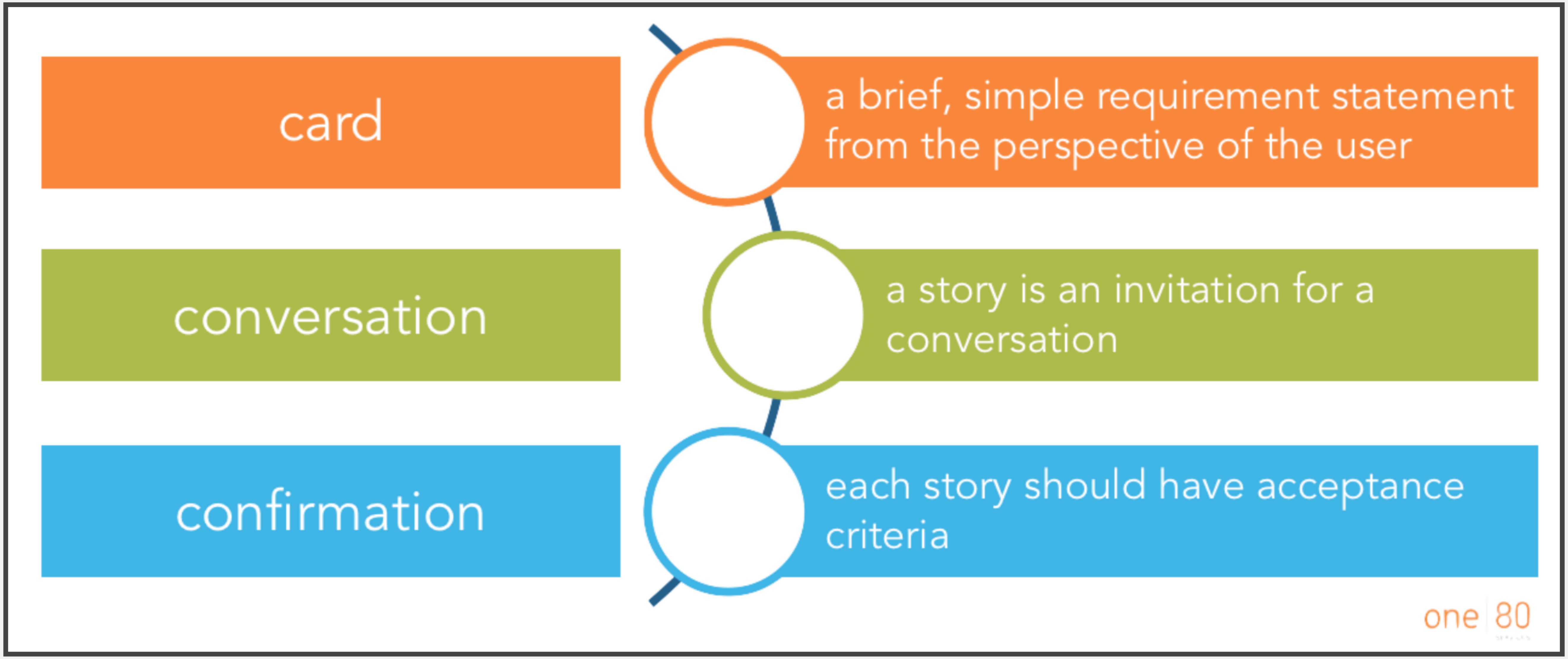
- Sprint Retrospective
  - Review sprint process

- Sprint Review Meeting
  - Review Product

User Stories



User Stories



card

a brief, simple requirement statement from the perspective of the user

conversation

a story is an invitation for a conversation

confirmation

each story should have acceptance criteria

Card

- “As a [role], I want [function], so that [value]”

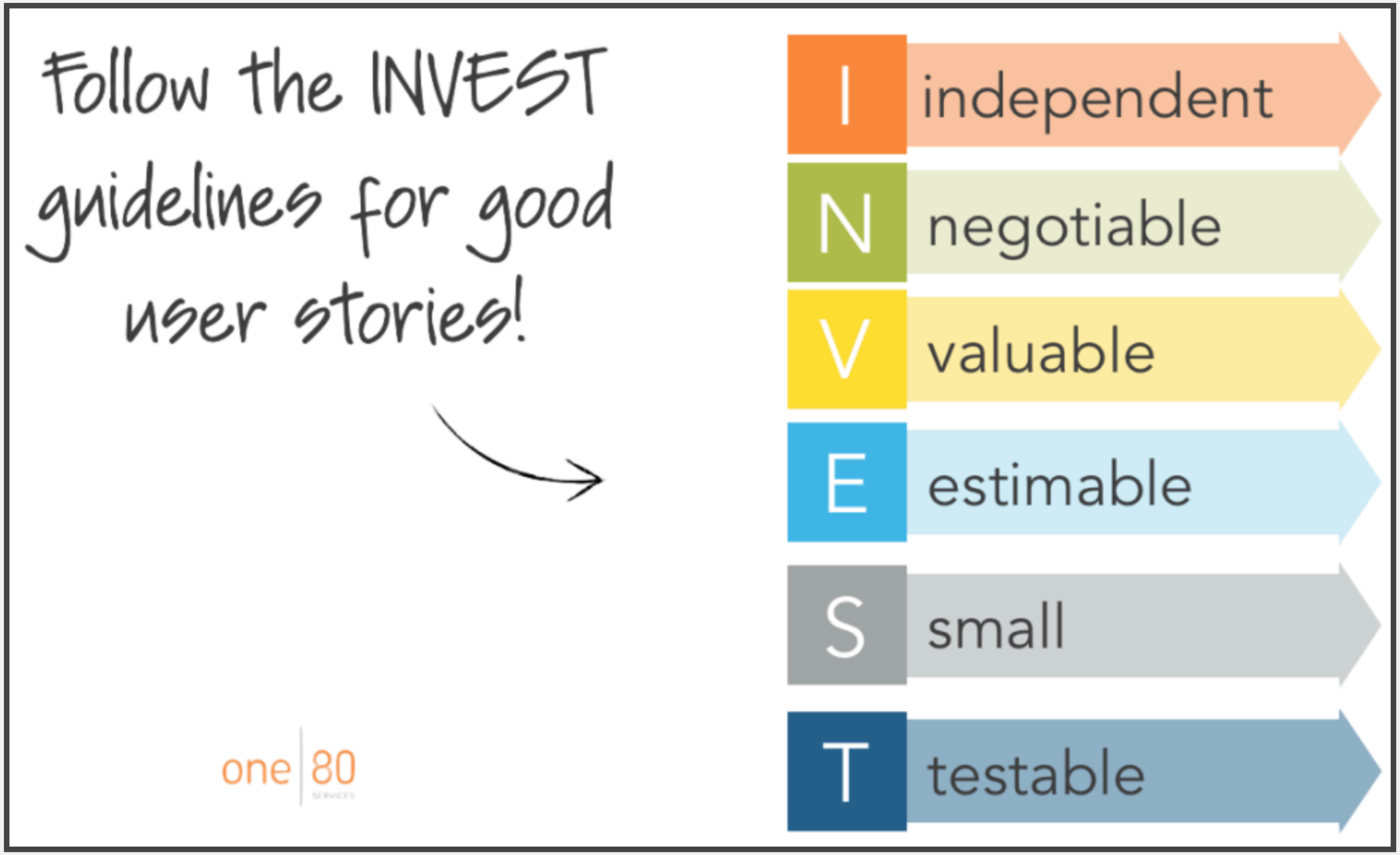
Card

- What must a developer do to implement this user story?

Confirmation

- How can we tell that the user story has been achieved
- It’s easy to tell when the developer finished the code.
- But, how do you tell that the customer is happy?

How to Evaluate a User Story

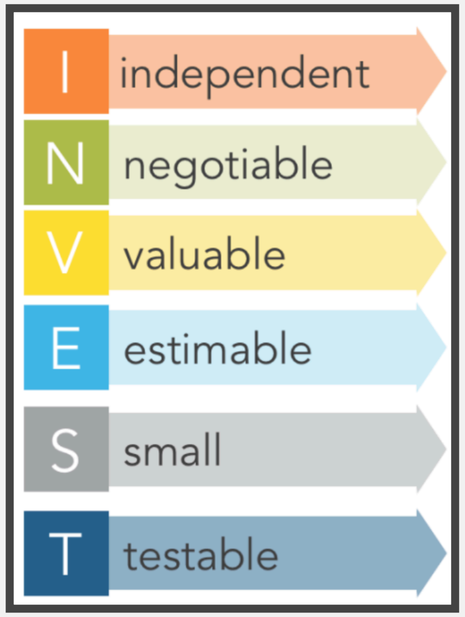


Follow the INVEST guidelines for good user stories!

- I independent
- N negotiable
- V valuable
- E estimable
- S small
- T testable

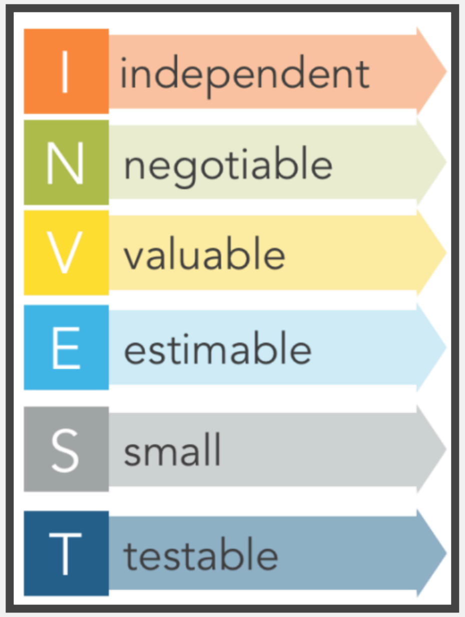
Independent

- Schedule in any order.
- Not overlapping in concept.
- Not always possible.



Negotiable

- Details to be negotiated during development.
- A good story captures the essence, not the details.



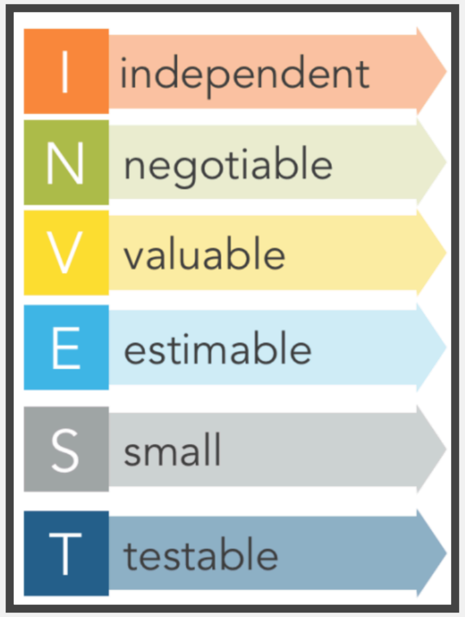
Valuable

- This story needs to have value to someone (hopefully the customer).
- Especially relevant to splitting up issues.

I independent
N negotiable
V valuable
E estimable
S small
T testable

Estimable

- Helps keep the size small.
- Ensure we negotiated correctly.
- “Plans are nothing, planning is everything” - Dwight D. Eisenhower



I independent

N negotiable

V valuable

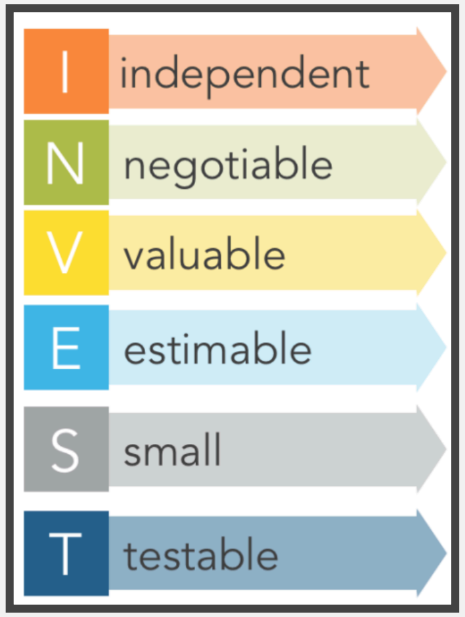
E estimable

S small

T testable

Small

- Can be written on a 3x5 card.
- At most two person-weeks of work.
- Too big === unable to estimate



I independent

N negotiable

V valuable

E estimable

S small

T testable

Testable

- Ensures understanding of task
- We know when we can mark task "Done"
- Unable to test === I do not understand it

I independent
N negotiable
V valuable
E estimable
S small
T testable