Workshop
State of Washington

18F

Mar 10, 2022

3 session agenda

Session #1 - learning session

↪ Overview, methods, and goals

Session #2 - small group work session

↪ Product vision, roadmap review, and decision points

Sessions #3 - small group work session

↪ QASP review + RFQ Drafting!

WA HHS Coalition milestones

Jan - Mar 2022	Apr - Jun 2022	Apr-June 2022	Jul-Sept 2022
• Platform and Product 1 RFPs are written + shared w/ CMS • Identify and prioritize questions that need to be answered to enable best collaboration with vendors • Start prep for M&O recompete	• Hiring and staffing internal positions • RFPs are released • Licences + environments start getting set up as possible • Conduct research + collaboration sessions to answer questions + prep for vendors • Start drafting M&O recomplete RFP	• Vendors evaluated + selected • Contracts approved by CMS + executed • Prep for next DP begins • M&O Recompete RFP is out • Begin business + policy processes to create streamlined application	• Vendors start work • Next DP is submitted • M&O vendor(s) selected

Agile Contract Format

Background & purpose

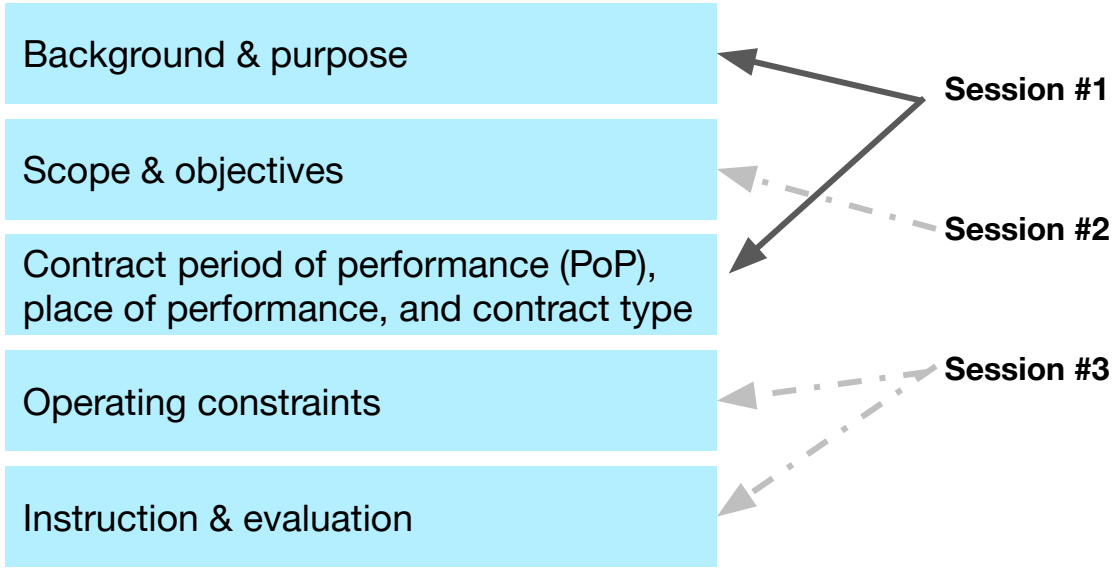
Scope & objectives

Contract period of performance (PoP), place of performance, and contract type

Operating constraints

Instruction & evaluation

Agile Contract Format



Session #1 agenda

1

Intros + norms

2

Intro to agile and product ownership

Break

3

HCD and DevOps 101

Break

4

Agile acquisitions

5

Reflection + next steps

Welcome!

We are honored to work with you.

18F is a digital services consultancy for government, inside the U.S. government.

(We're federal employees.)

We help agencies build, buy, and share the best digital government services available.

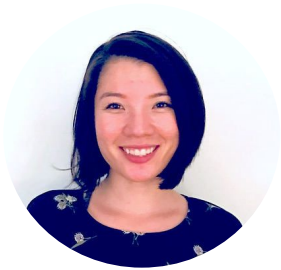
Your 18F team

Nina Mak
Product Management Lead

Greg Walker
Consulting Engineer + Project lead

Randy Hart
Acquisition Lead

Austin Hernandez
Senior Product Designer



Amy Ashida
Principal Consultant

Erica Vosseller
Account Manager

Who is here today?

Write in the chat:

- Your name
- Your division and program
- Your role in this meeting
- One thing that makes you proud to work for your organization

Session norms

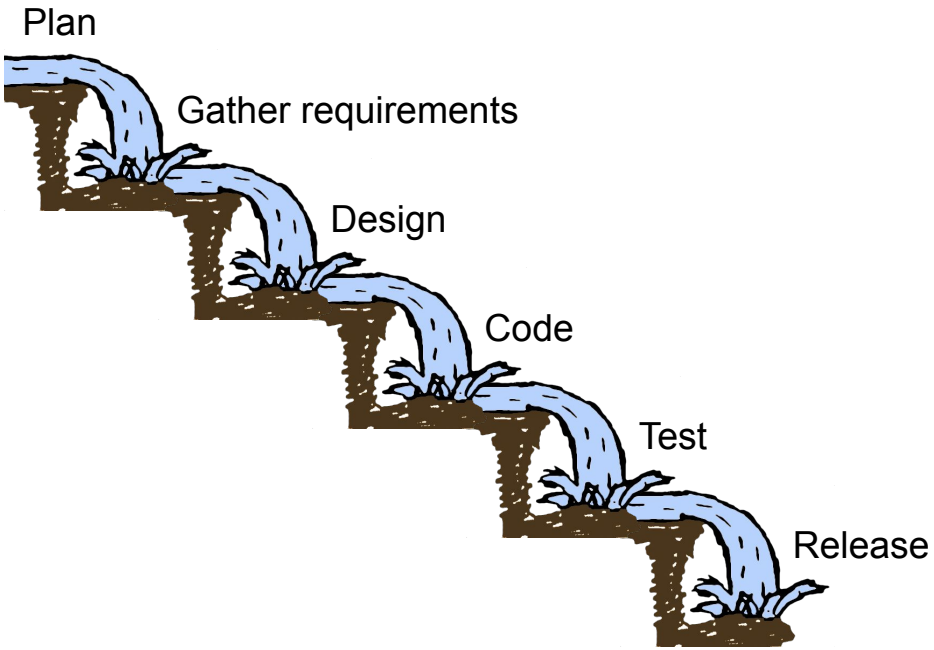
During this time we're aiming to build shared understanding of goals, ways of working, and paths forward. How will we as a group behave to make that possible?

- Take space + make space for others
- Be comfortable with silence
- Put challenges in the middle of the table for consideration by all
- Interrupt with questions
- We'll maintain a parking lot of things to come back to
-
-
-
-

A quick poll to get us started

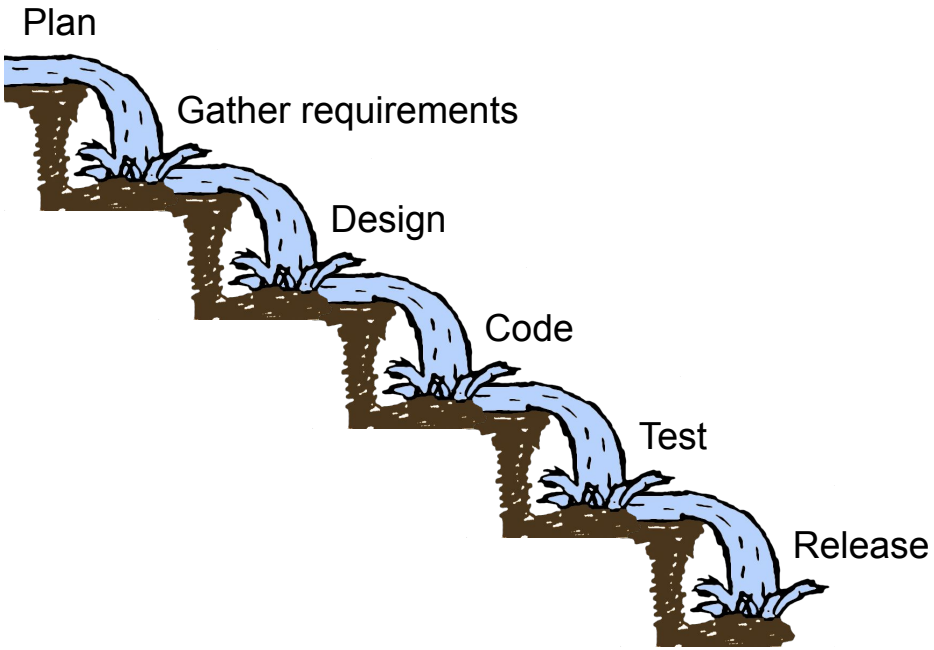
Agile & Agile misconceptions

Waterfall development



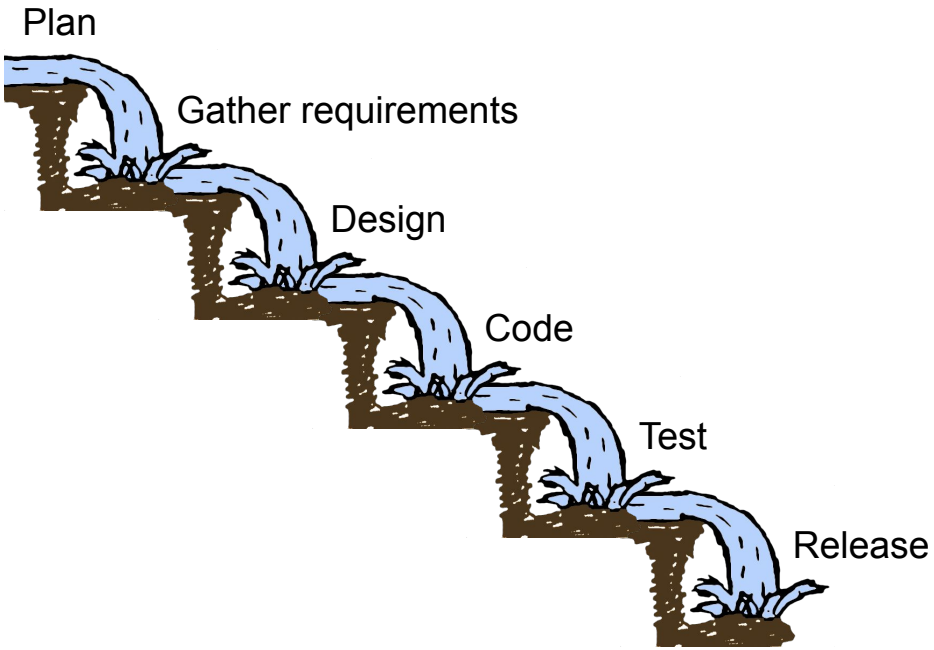
- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*

Waterfall development



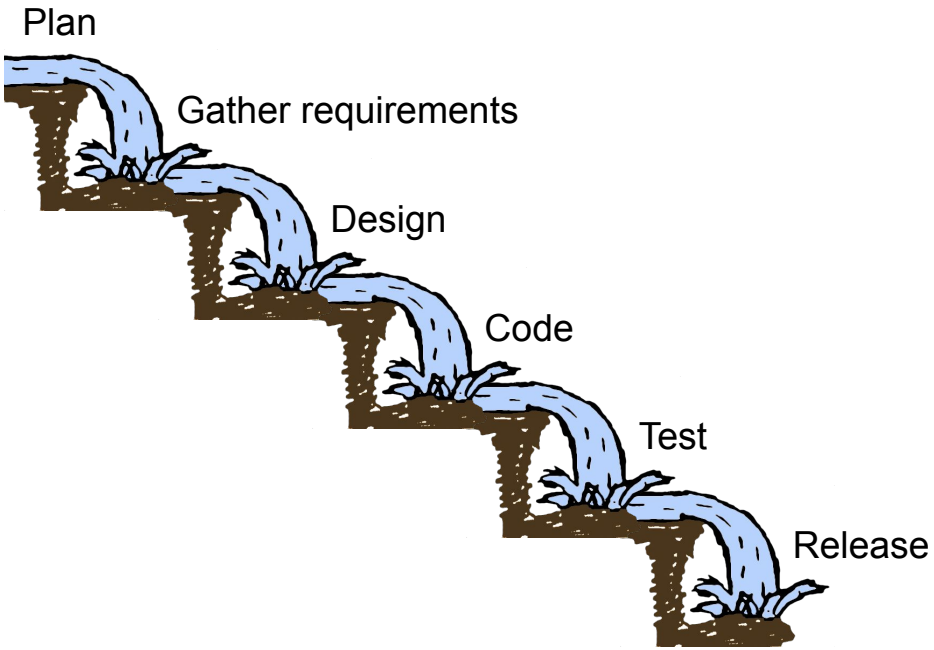
- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*
- He showed a variation of this diagram. It's right after the 4th paragraph of the paper. Just before the 5th paragraph.

Waterfall development



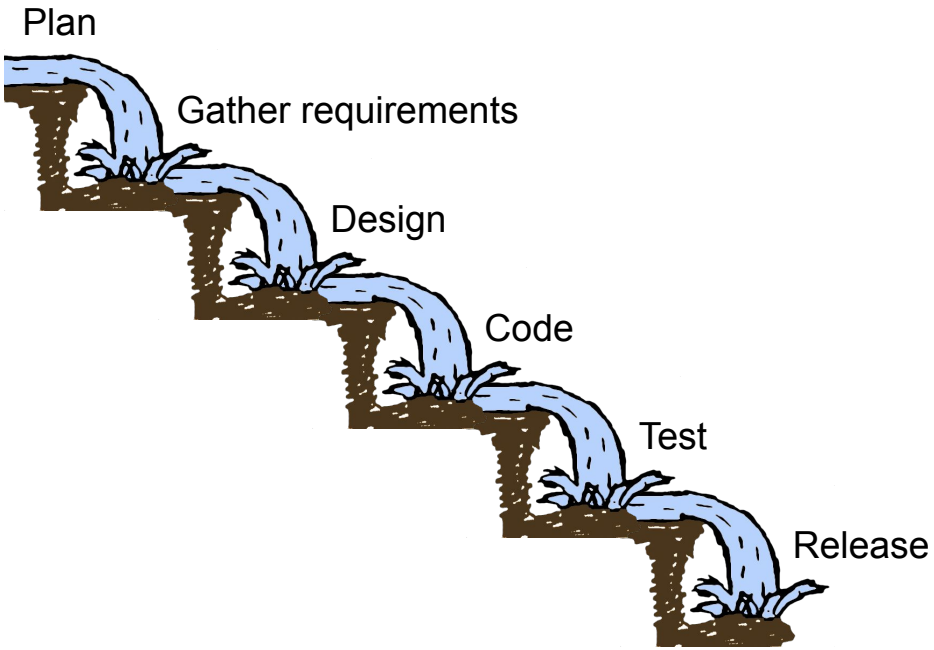
- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*
- He showed a variation of this diagram. It's right after the 4th paragraph of the paper. Just before the 5th paragraph.
- The 5th paragraph begins by saying it won't work.

Waterfall development



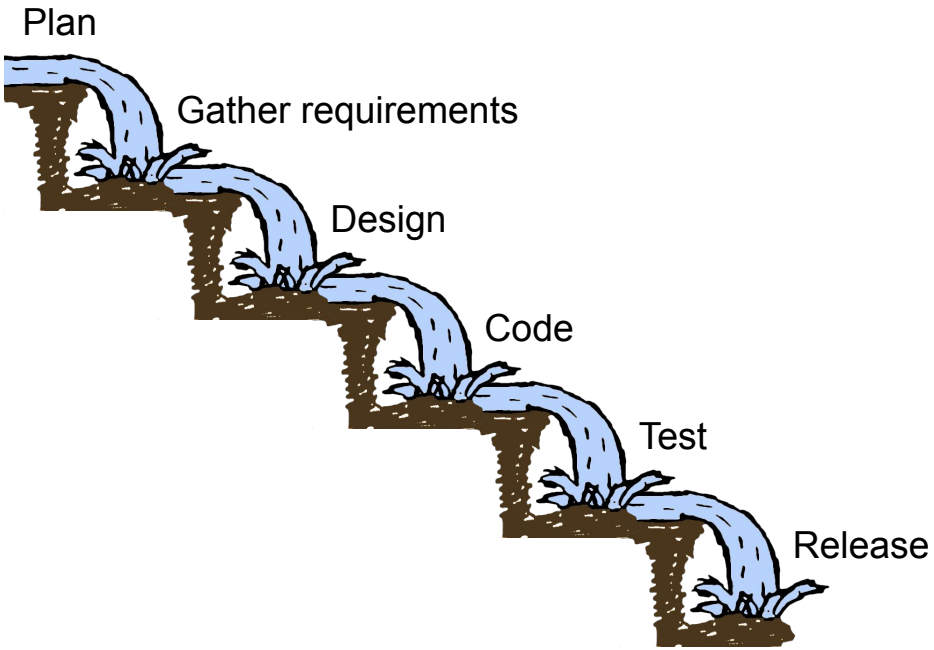
- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*
- He showed a variation of this diagram. It's right after the 4th paragraph of the paper. Just before the 5th paragraph.
- The 5th paragraph begins by saying it won't work.
- We never should have done this.

Waterfall development



- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*
- He showed a variation of this diagram. It's right after the 4th paragraph of the paper. Just before the 5th paragraph.
- The 5th paragraph begins by saying it won't work.
- We never should have done this.
- The guy who identified it said so.

Waterfall development



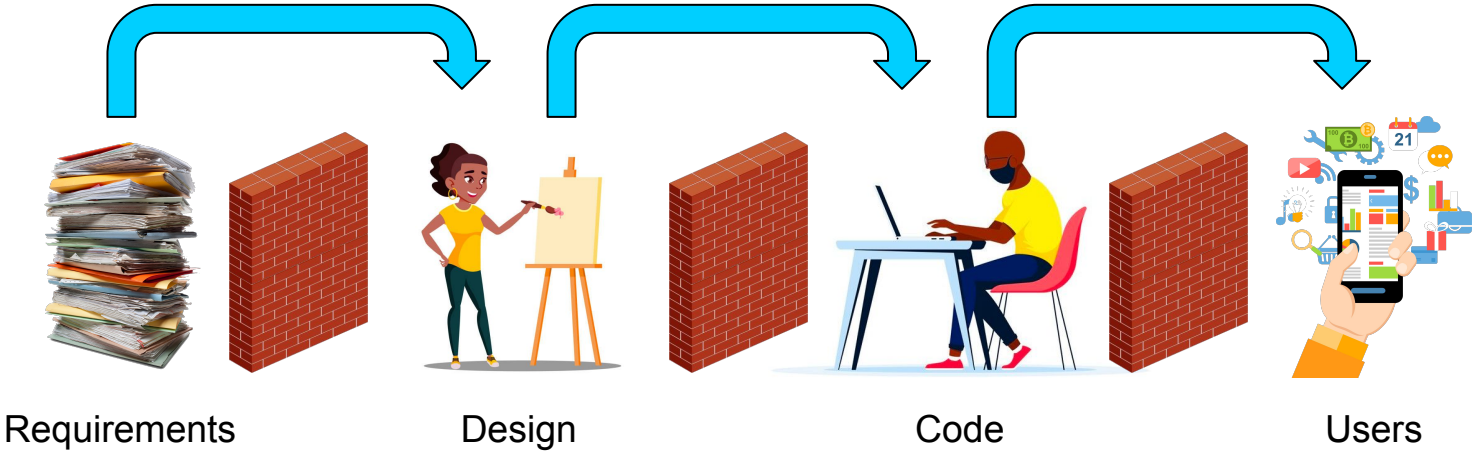
- Introduced by Winston Royce in 1970, *Managing the Development of Large Software Systems*
- He showed a variation of this diagram. It's right after the 4th paragraph of the paper. Just before the 5th paragraph.
- The 5th paragraph begins by saying it won't work.
- We never should have done this.
- The guy who identified it said so.
- He said we should be iterative, but we basically ignored him until 2001.

Agile is not only a way of working, it’s a way of thinking and acting, it’s our aspiration to build products faster, in a collaborative manner and, with the Customer on the center of our thoughts.”

- *João Baptista Leite, CEO, Unicre – Instituição Financeira de Crédito, S.A.*

Agile vs waterfall

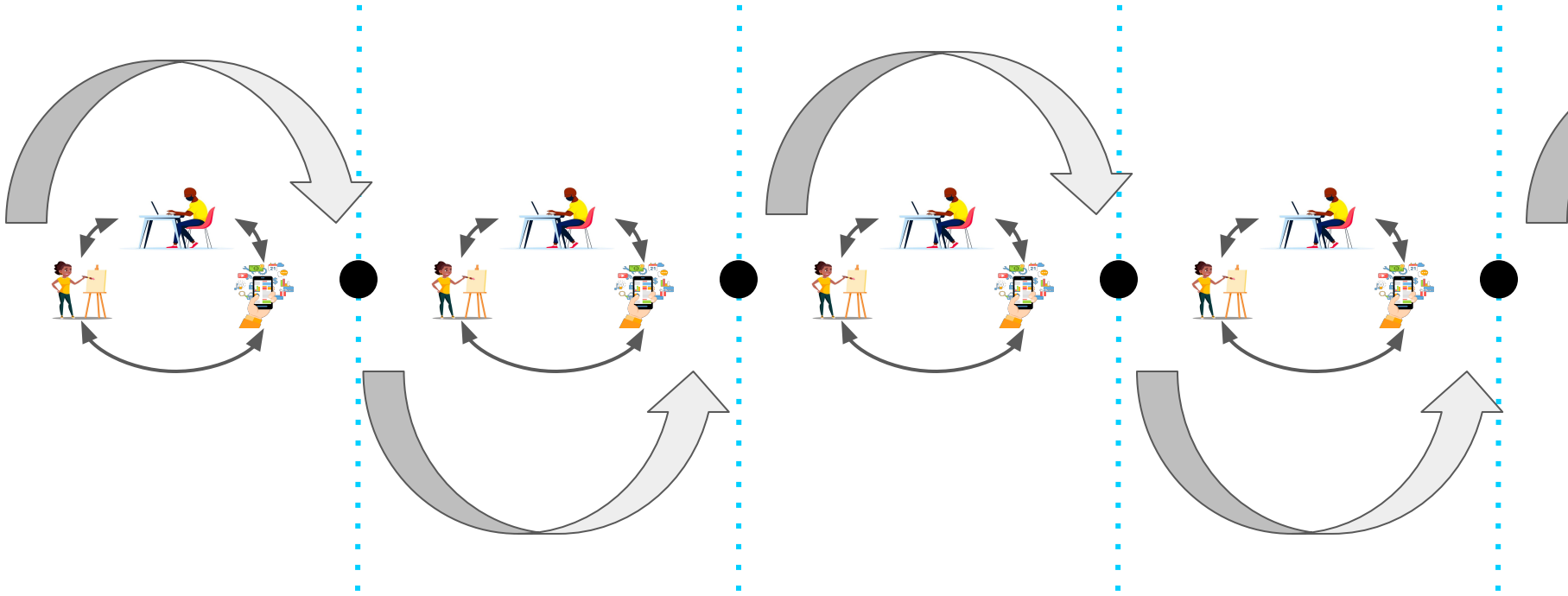
Waterfall and the walls



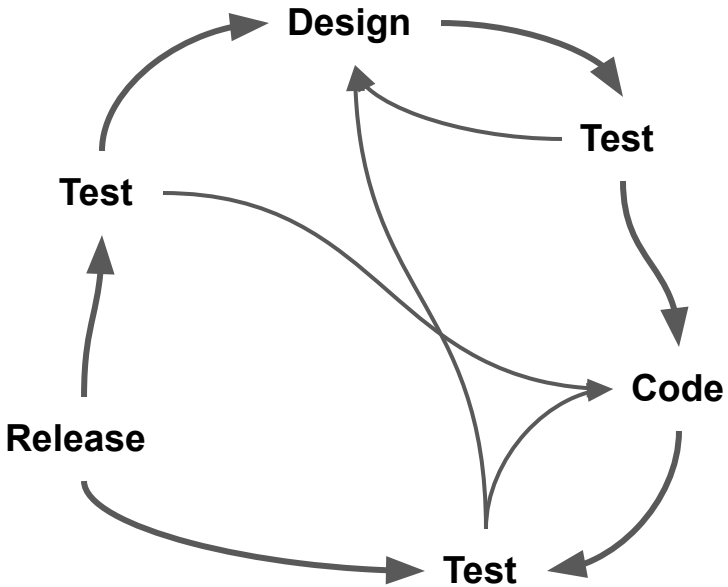
Agile comes in like a wrecking ball

Agile comes in like a wrecking ball

Agile keeps stakeholders in the loop



Agile is full of loops



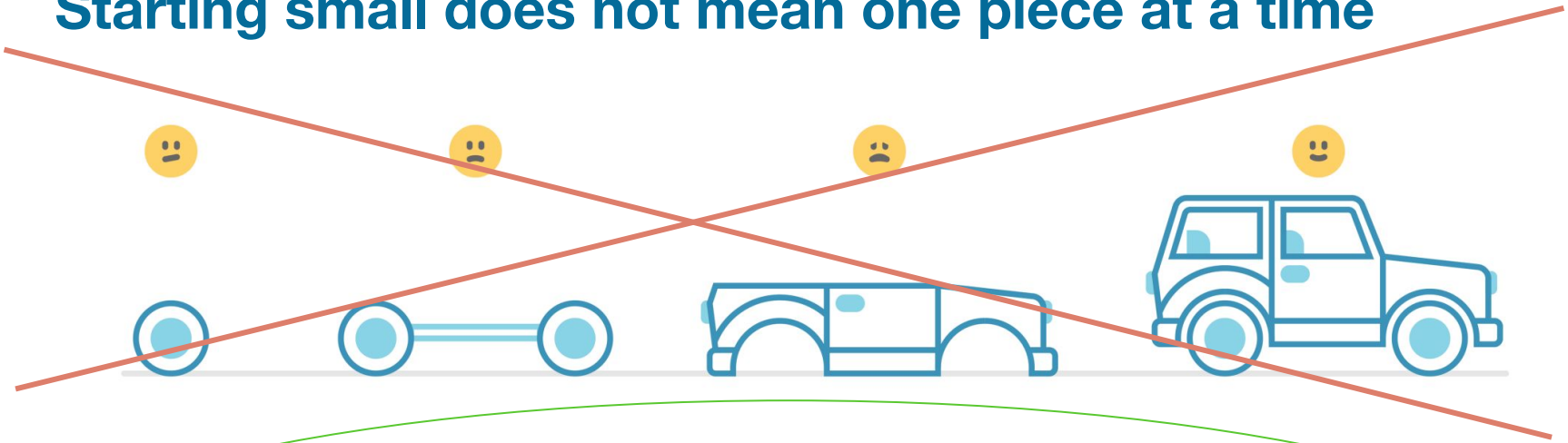
Agile misconceptions

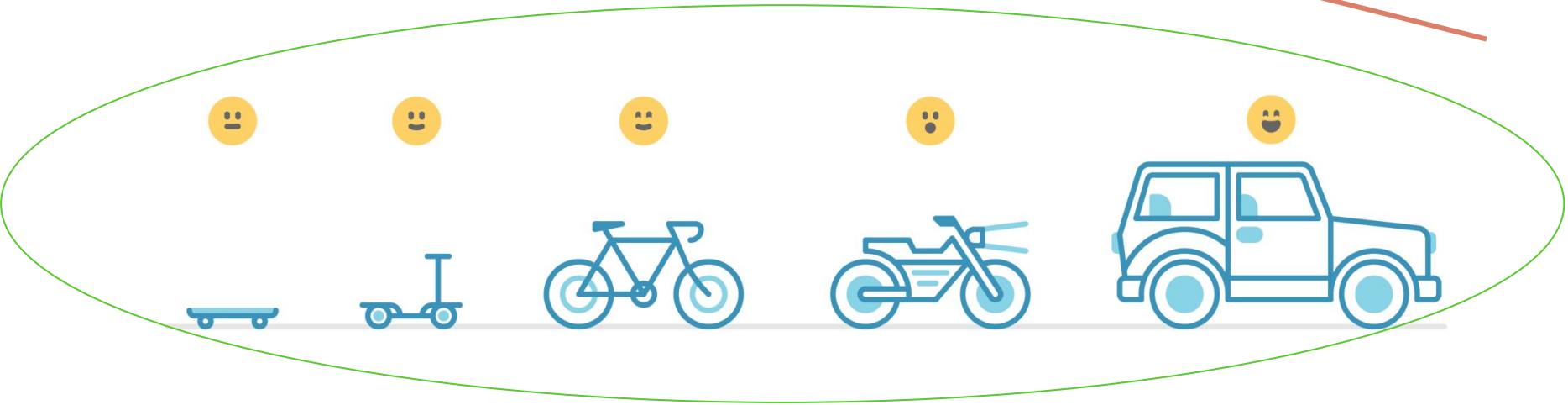
Common assumptions made about agile and what we mean

Common misconceptions:

- Starting small means one piece at a time
- Agile means less planning
- Agile means less/no documentation
- Agile is fast/less work

Starting small does not mean one piece at a time



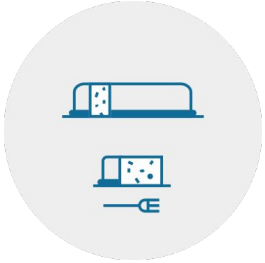


Agile does not mean no/less planning

1. Learn about users

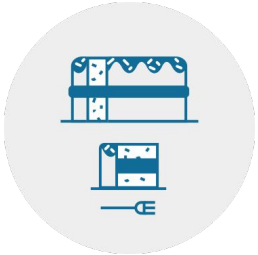
2. Make prototypes and test them

3. Build a small version of the real thing



4. Get feedback from real users

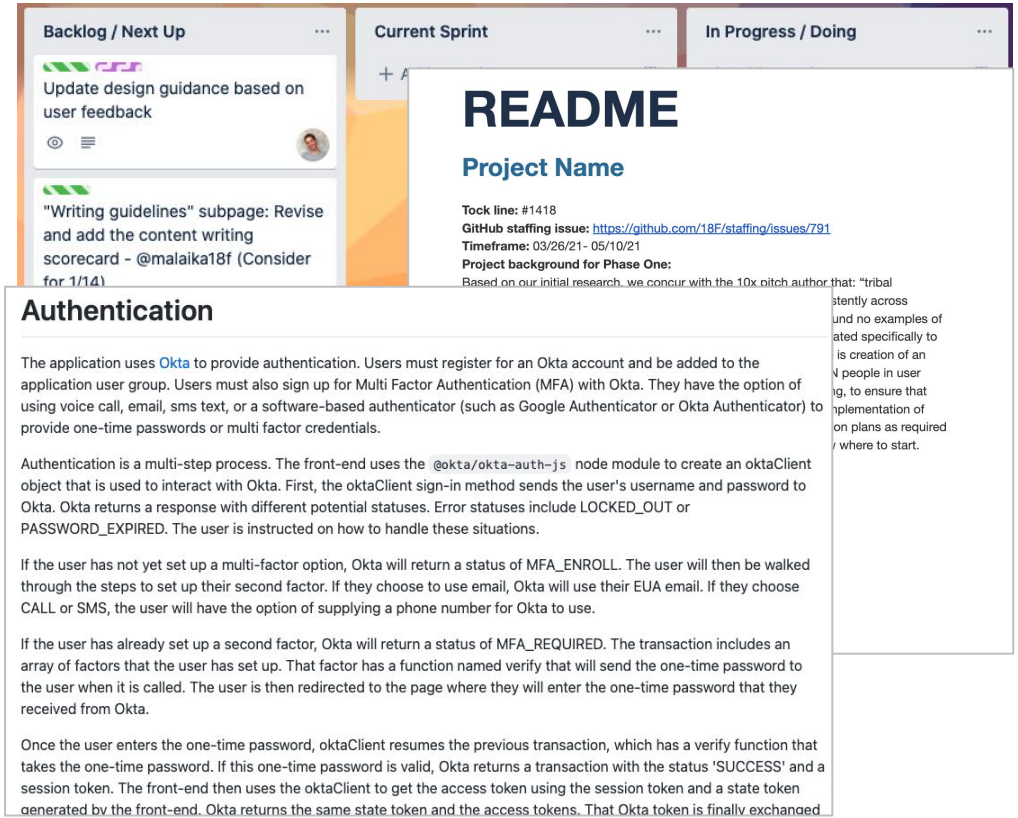
5. Use feedback to adjust and improve



6. Rinse and repeat!

Agile does not mean no/less documentation

- Transparency of work and team capacity
- Shared vision/goals
- Clear idea of what has happened and what needs to be done
- Less rework and duplication of work
- Technical and research findings for future teams



Agile is not (necessarily) fast/less work

More discovery work upfront

To avoid rework and costly oversights

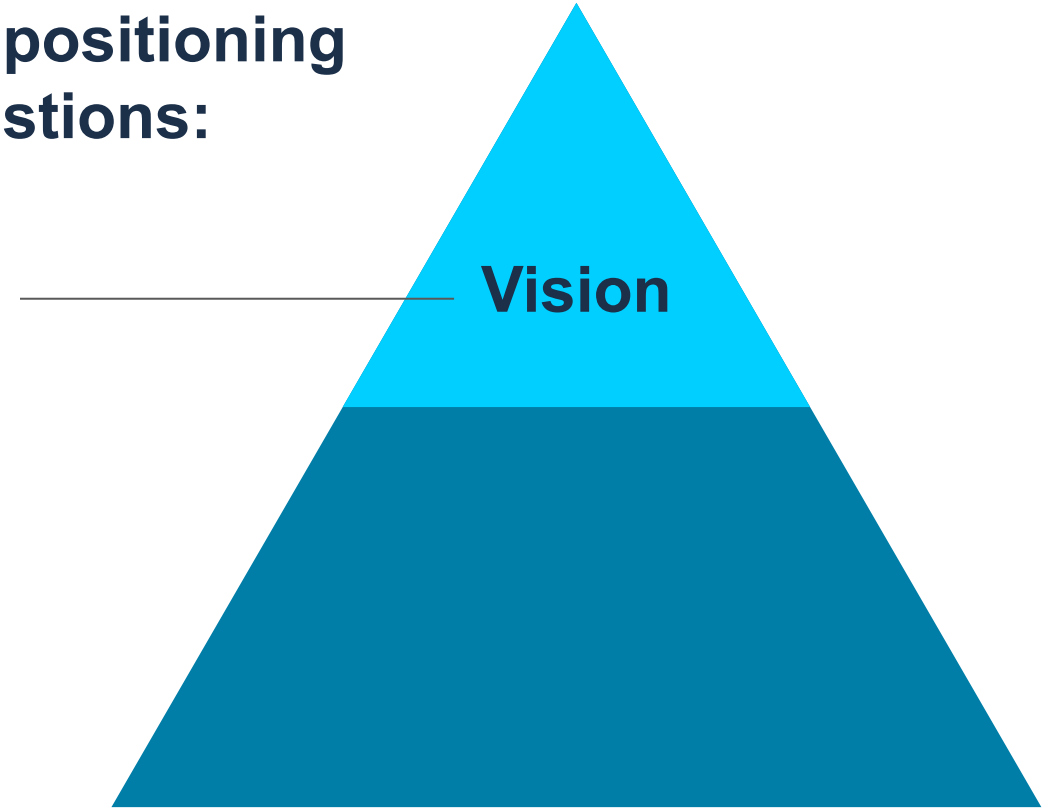
Product ownership

Moving toward successful outcomes (not output)

Who owns the vision? The strategy?
The State of Washington

Product uses that positioning to answer two questions:

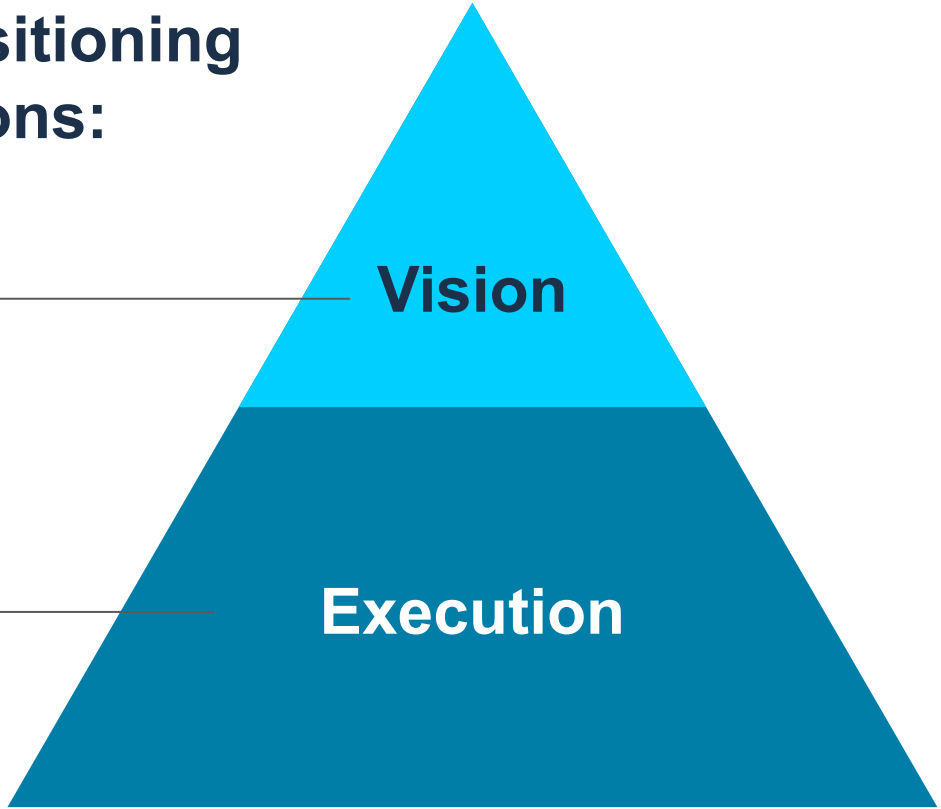
1. **What should we do? Why?**



Product uses that positioning to answer two questions:

1. **What should we do? Why?**

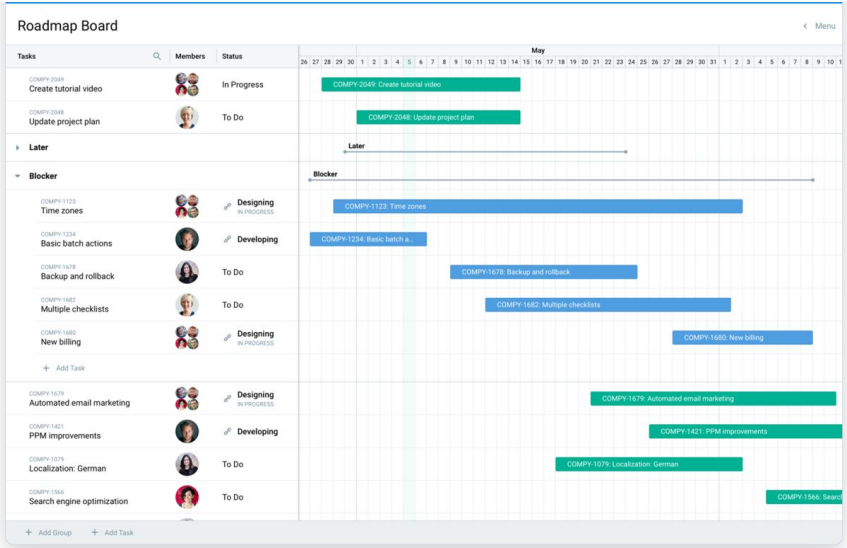
2. **How do we get there?**

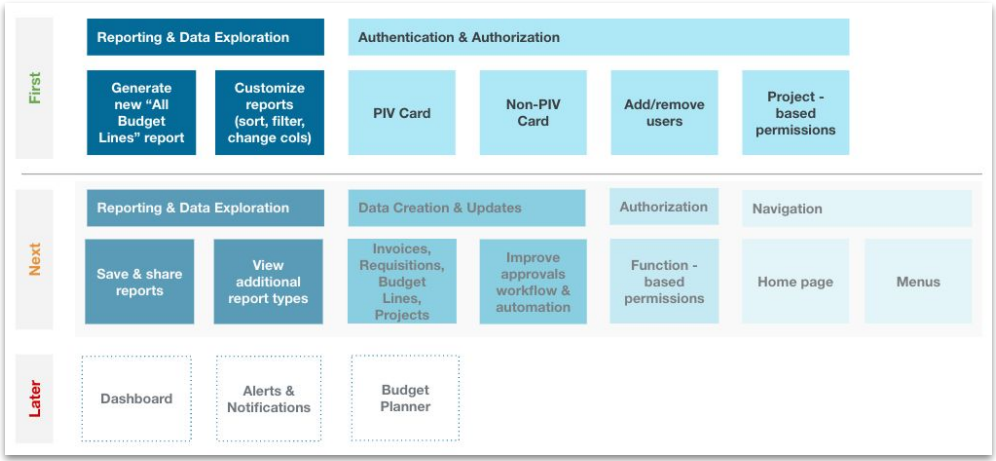


Product owners collaborate with team members and stakeholders to create a shared vision, then communicate it internally and externally.

From the vision, product owners create strategy, roadmap, and feature definition of a product.

Product roadmaps are a powerful tool for product teams charting out a strategic plan





Product roadmaps communicate priorities and sequence

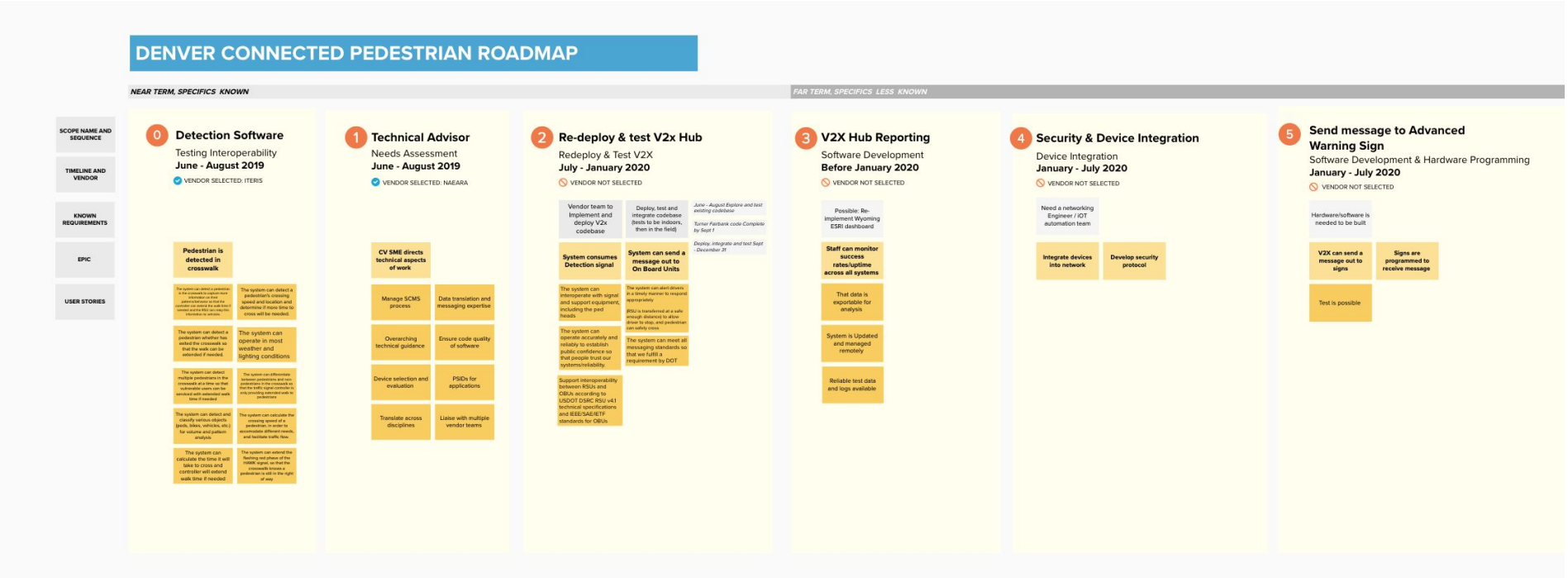
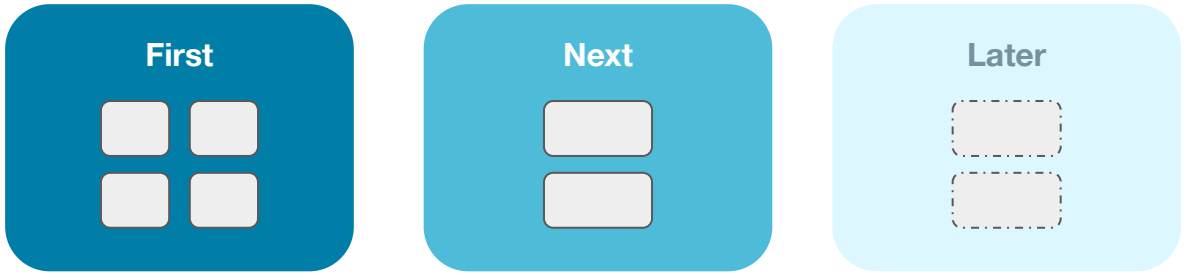


Image 9: IE&E Modernization Roadmap

Status Tracker Roadmap



The roadmap is a bridge between your strategic vision and your backlog of work

Product owners protect the project team, scope, and what gets built.

Product owners protect the project team, scope, and what gets built.

They should say “no” often, and leadership should give them the space to do so.

Design and execution matter

Product owners own four questions:

1. **What’s the problem we’re solving? (and why does it matter?)**

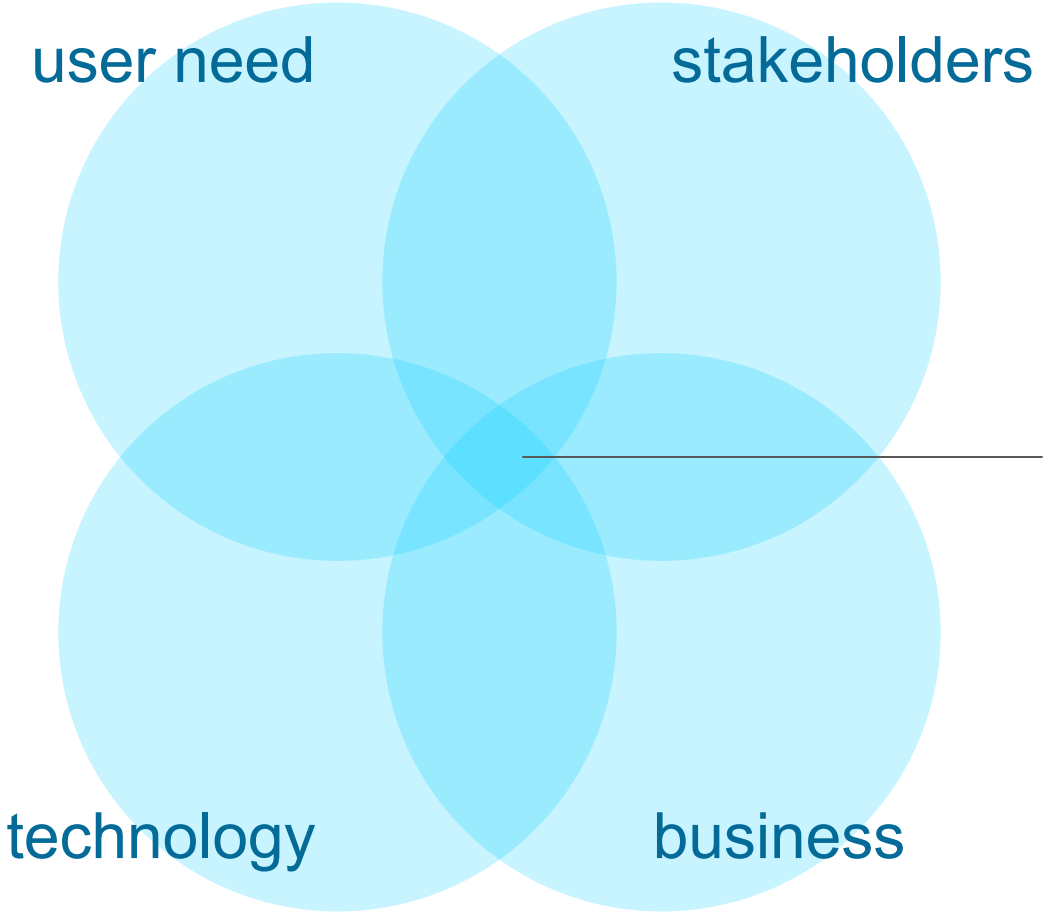
1. What's the problem we're solving? (and why does it matter?)
2. **What will our impact be?**

1. What's the problem we're solving? (and why does it matter?)
2. What will our impact be?
3. **How are we doing it?**

1. What's the problem we're solving? (and why does it matter?)
2. What will our impact be?
3. How are we doing it?
4. **Is our solution good?**

Product teams have to be able to provide good answers to all of these questions.

Product teams have to be able to provide good answers to all of these questions. **When the answer is “we don’t know,” they need to have a concrete plan to get to an answer.**



Product owners sit at the intersection of design and research, development, business, policy, and politics.

A word of warning: product ownership is a major commitment.

Treating product owners as “other duties as assigned” is not setting the product owner, or the effort, up for success.

Product owners' days are filled with

Planning and strategizing

- Defining outcomes
- Breaking down the work
- Resolving ambiguity
- Driving decisions & alignment
- Prioritizing
- Balancing

Executing and delivering

- Refining the backlog
- Writing user stories
- Clearing blockers for the team
- Reviewing work

Maintaining and retrospecting

- Monitoring usage
- Soliciting feedback
- Prioritizing fixes

Meeting with users **Communicating with stakeholders** **Attending usability sessions**

Human-centered design

Putting people at the center of all problem solving

Human-centered design helps us design for actual humans who will use what we're making.

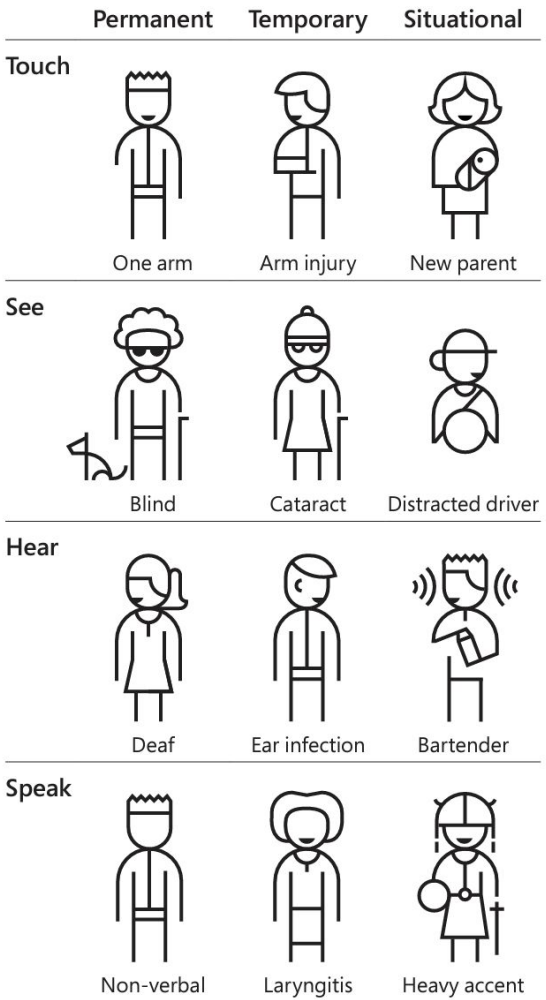
Who are the people you are designing for?

____ who ____

Ex.
A person who receives benefits.

Microsoft's inclusive design principles:

- Solve for one, extend to many.
- All humans grow and adapt to the world around them.
- Exclusion happens when we solve problems using our own biases.
- Include and learn from people with a range of perspectives.

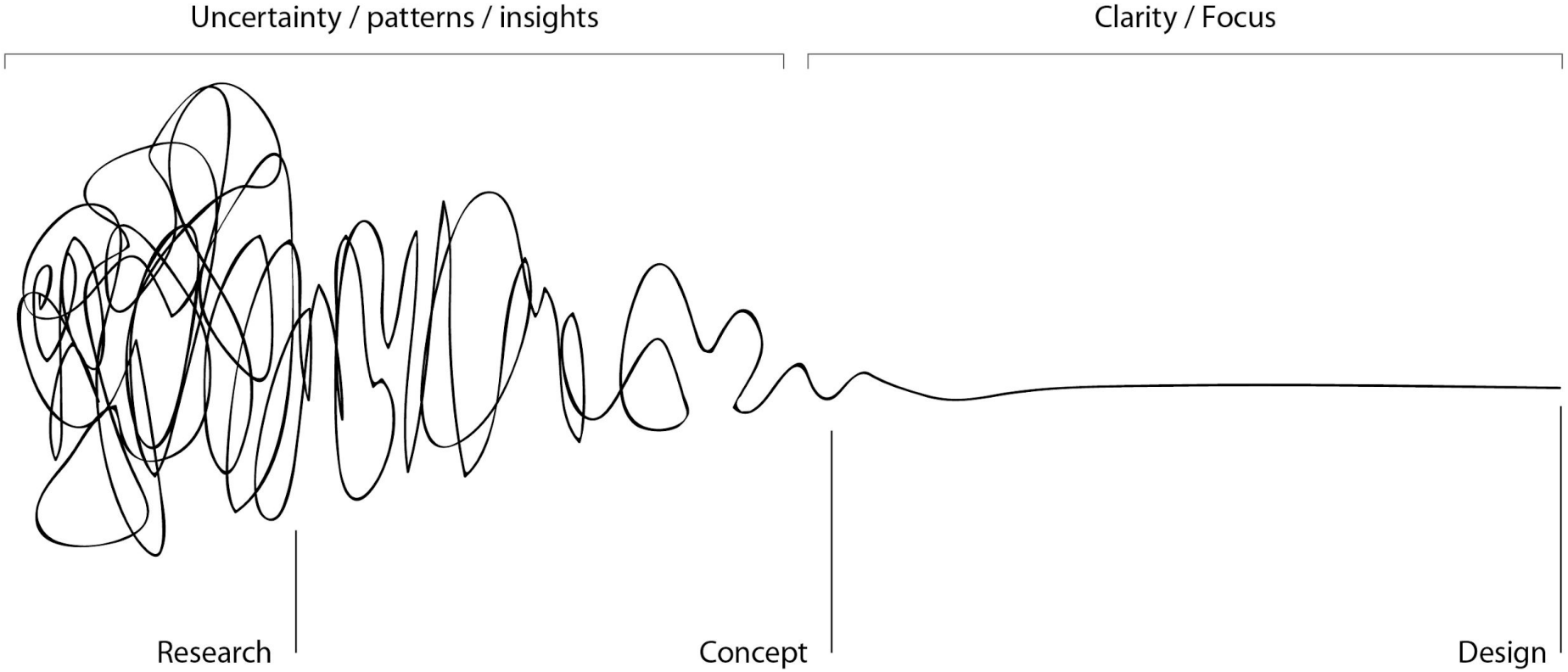


The Persona Spectrum

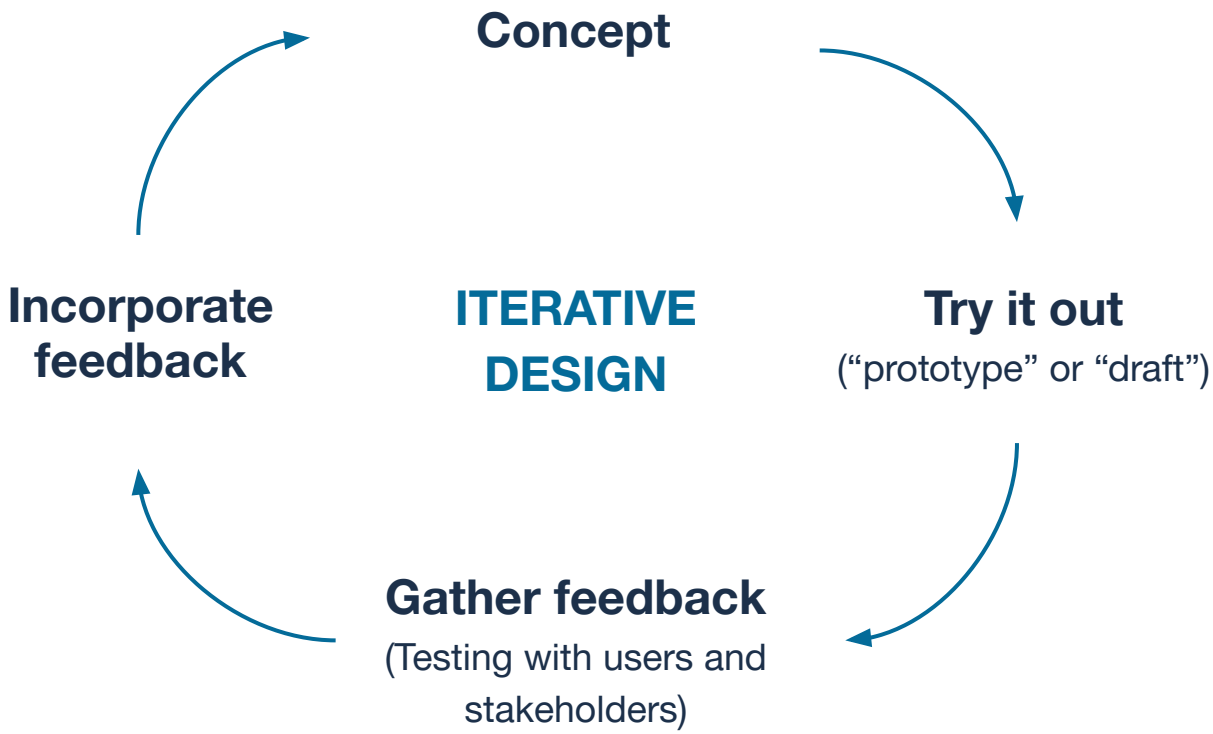
We use the Persona Spectrum to understand related mismatches and motivations across a spectrum of permanent, temporary, and situational scenarios. It's a quick tool to help foster empathy and to show how a solution scales to a broader audience.

Courtesy of Microsoft

There’s no one design process
but there are consistent patterns.

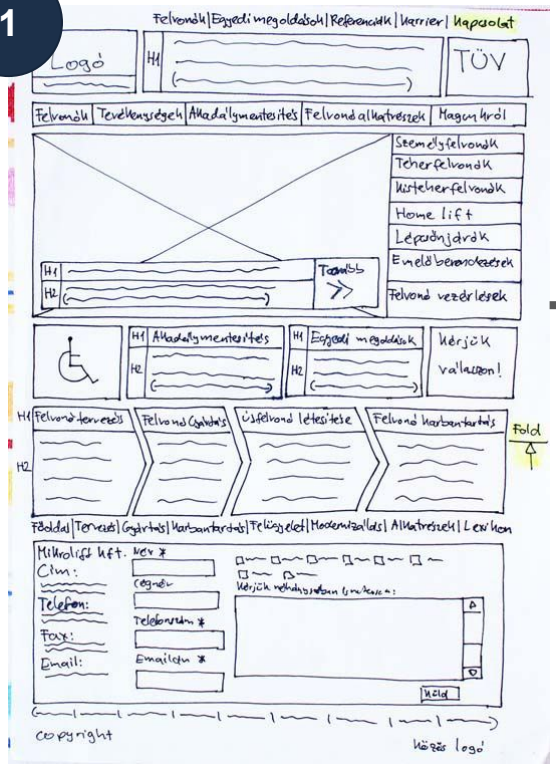


Courtesy of ReVision Lab

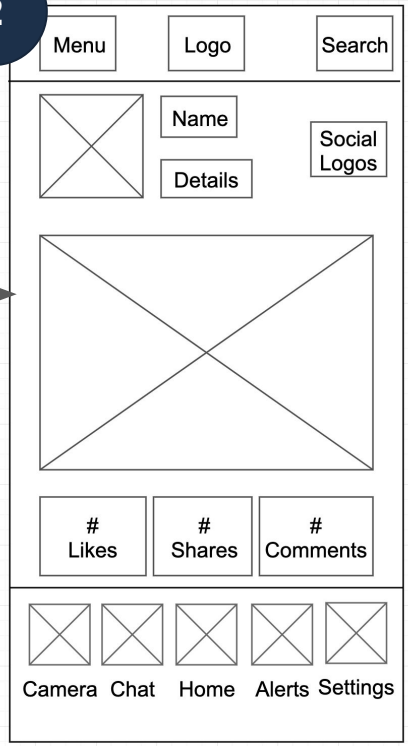

Concept
Try it out
(“prototype” or “draft”)
Gather feedback
(Testing with users and stakeholders)
Incorporate feedback
ITERATIVE DESIGN

Trying things out doesn't need to be costly

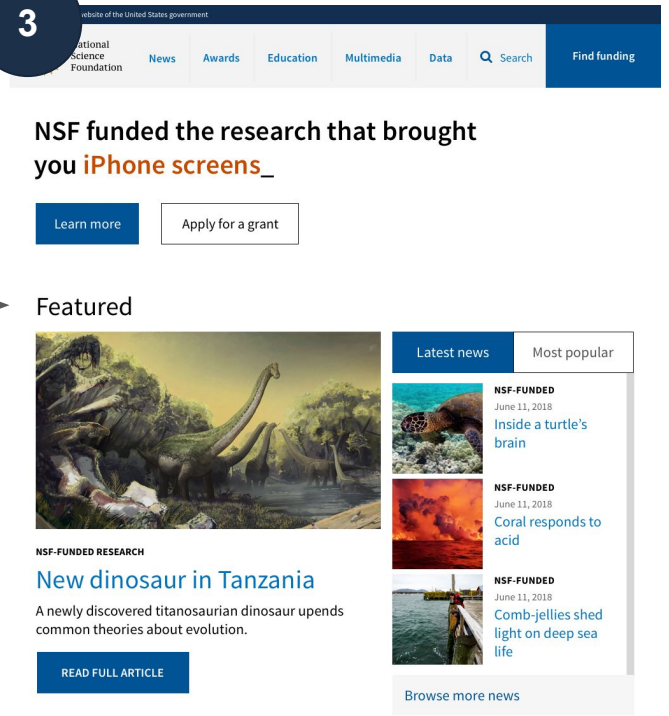
1



2

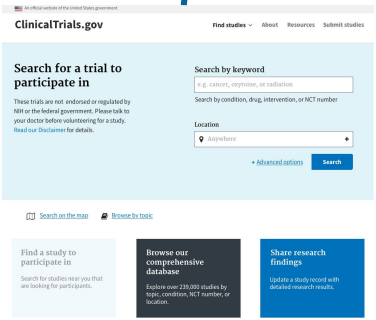


3

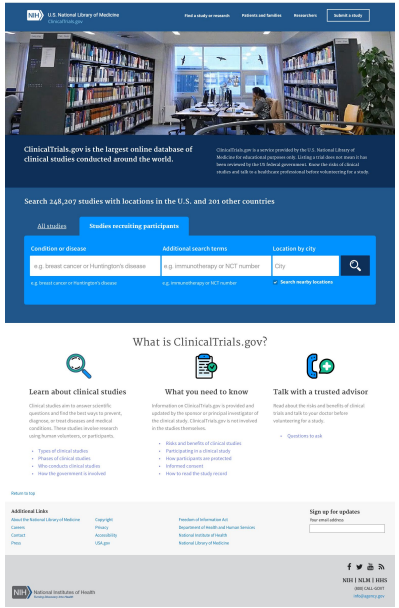


Learn from people and incorporate feedback

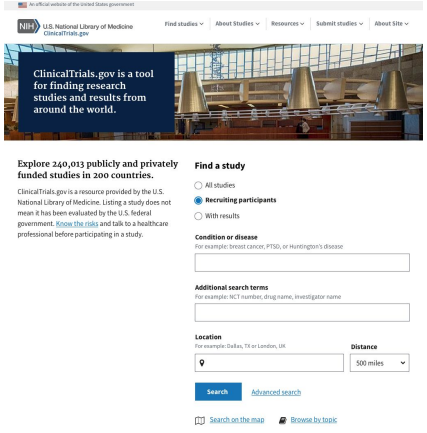
Sprint 1

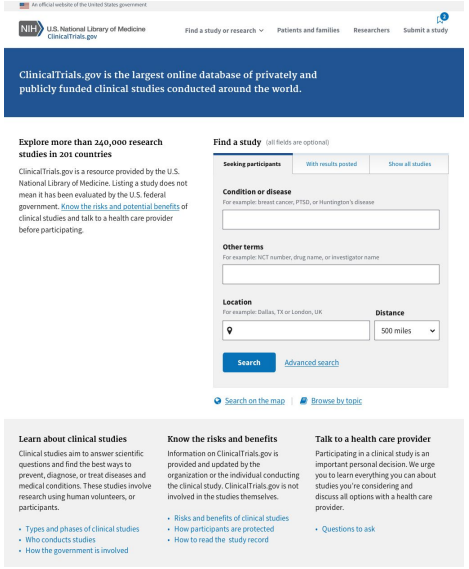


Sprint 2



Beta launch design





Product Owner

Tech Lead

Project Manager

Business Analyst

Procurement Specialist

Everyone has a role to play

Research Lead

Note taker / Documenter

Interviewer/Moderator

Coordinator

Observer

Everyone has a role to play

Good design habits to start practicing now

Do user research early and often

Rely on assumptions about your end-users

Explore multiple approaches

Double down on a single solution early

Define the outcomes you want

Stick to a predefined solution

DevSecOps

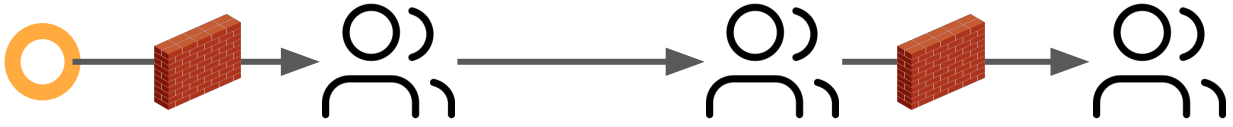
Helping us build robust solutions that we can constantly test, integrate, and deploy

18F

Letting people and computers do the things they're each good at

DevSecOps combines software development, security, and operations into one team, and automates as many of their processes as possible. People focus on the hard problems and let the computers handle the repetitive stuff.

A manual deployment process



developers make a set of changes

staff do tests, often on their own computer

IT staff run security scans and approve or deny

product technical staff deploy the product

The DevSecOps culture

- Open communications and **knowledge sharing** enhance trust within the team
- **Continuous feedback** results in tighter processes that get better and better
- Emphasizes the entire **system**, not just one piece, department, or interest
- **Experimentation** lets us learn from successes and failures, constantly

DevSecOps covers three broad areas

1

Testing - Making sure things work as expected

2

Continuous Integration - Regularly making sure everything still comes together correctly

3

Continuous Delivery - Regularly publishing the latest version of the product for users to use

1/ Testing

Does it do what it’s supposed to? Do the things that worked yesterday still work today?

Testing is about making sure the application does what it’s supposed to and catching bugs as early as possible.

Testing the product

1

Assert what the features of the application should do:

```
hello-app
assert screen-output =
“Hello!”
```

Testing the product

1. **Assert** what the features of the application should do:

```
hello-app
assert screen-output =
“Hello!”
```

2. **Write** some code to implement those features:

```
puts “Hello!”
```

Testing the product

1. **Assert** what the features of the application should do:

```
hello-app
assert screen-output =
“Hello!”
```

2. **Write** some code to implement those features:

```
puts “Hello!”
```

3. **Execute** the tests to make sure the application does what we expect:

```
Finished in 0.3s, 1 assertion
0 failures, 0 errors
```

2/ Continuous Integration

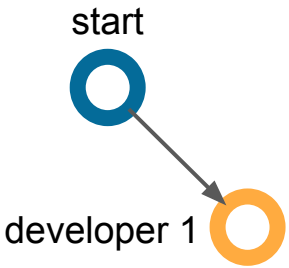
18F

Making sure the different parts go together, each step of the way

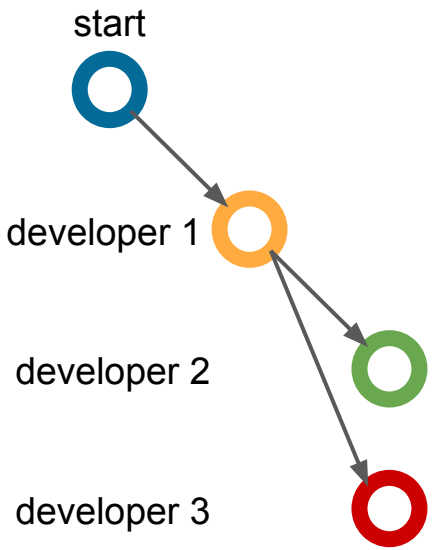
Continuous integration is taking each change, as it happens, and checking whether it breaks the product. This way the team knows quickly, as soon as a change is made, rather than days or weeks later.

From many, one

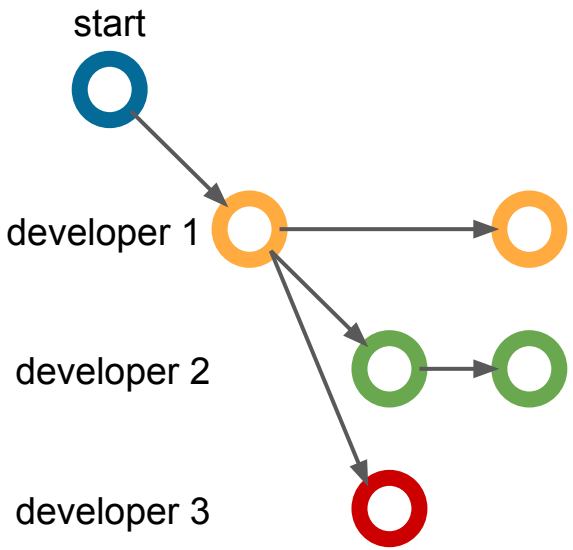
From many, one



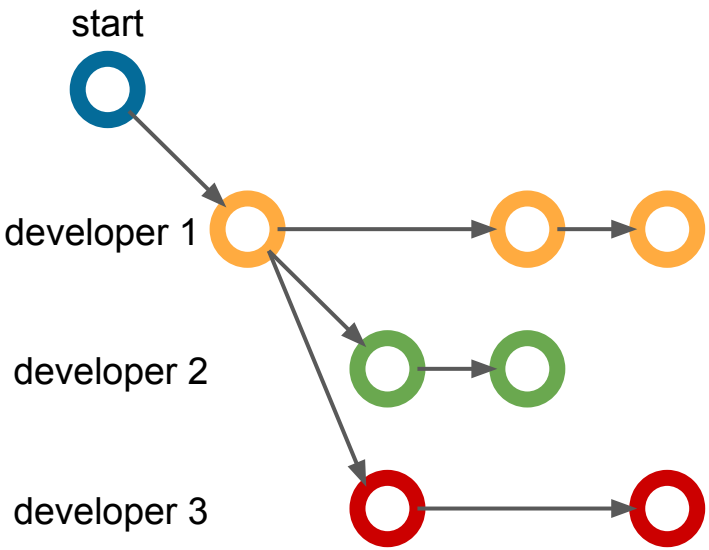
From many, one



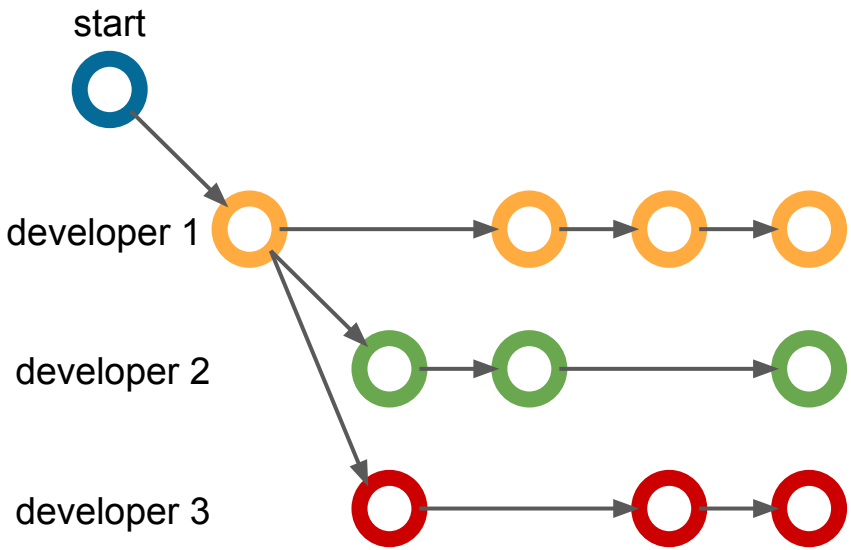
From many, one



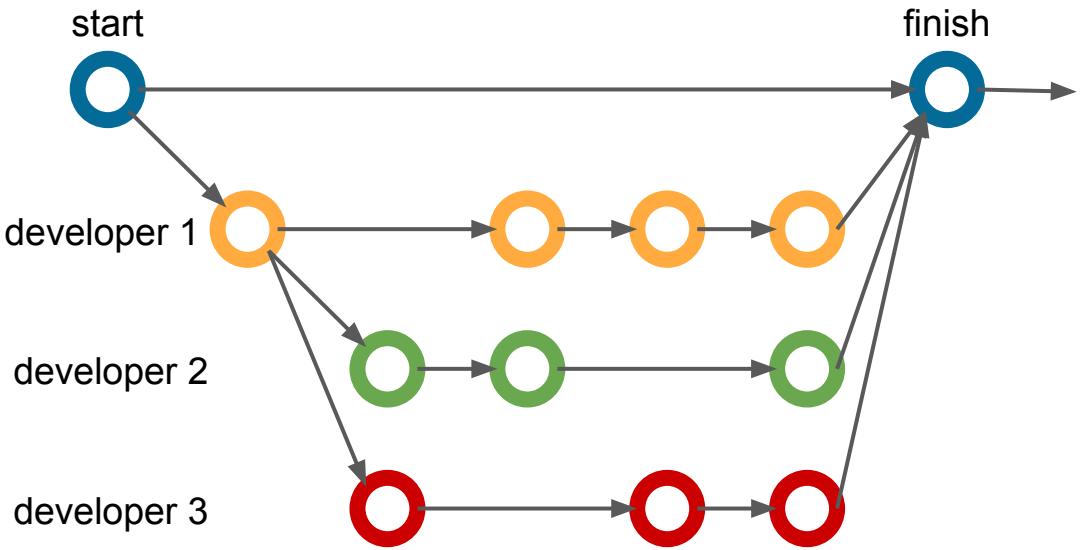
From many, one



From many, one

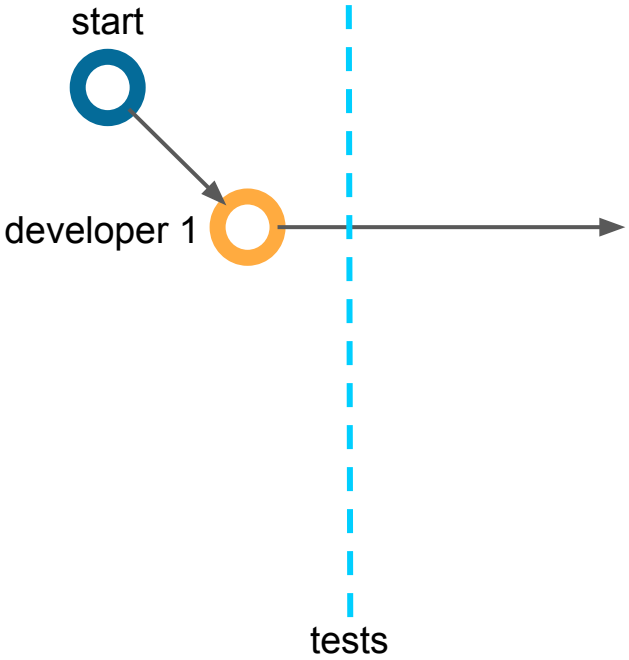


From many, one

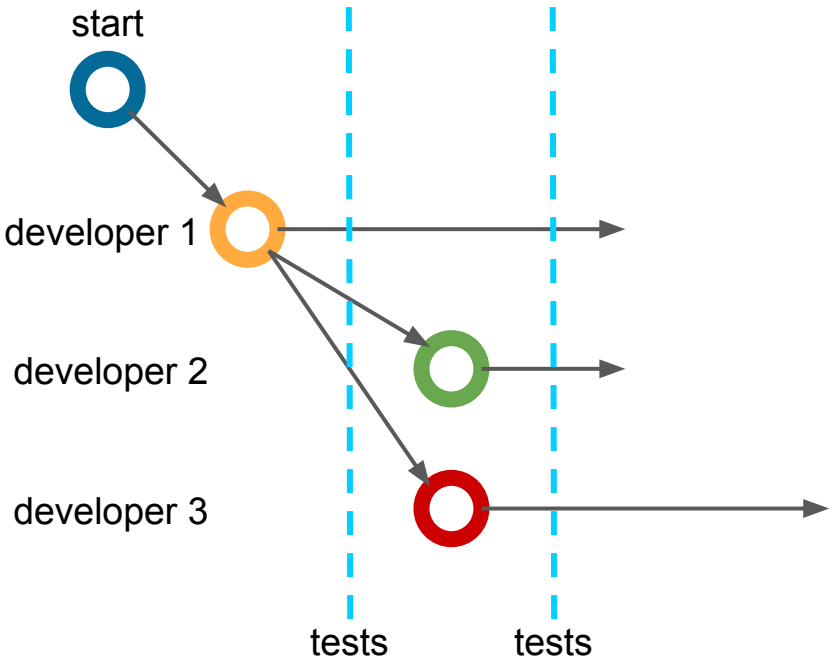


That looks like a disaster about to happen...

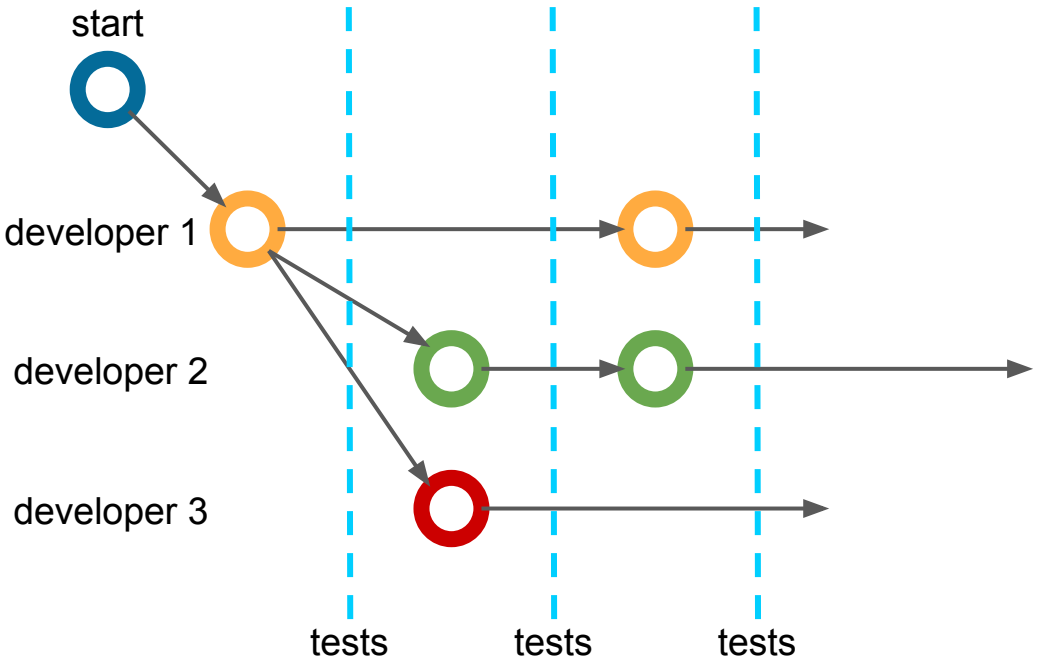
Everything is okay...



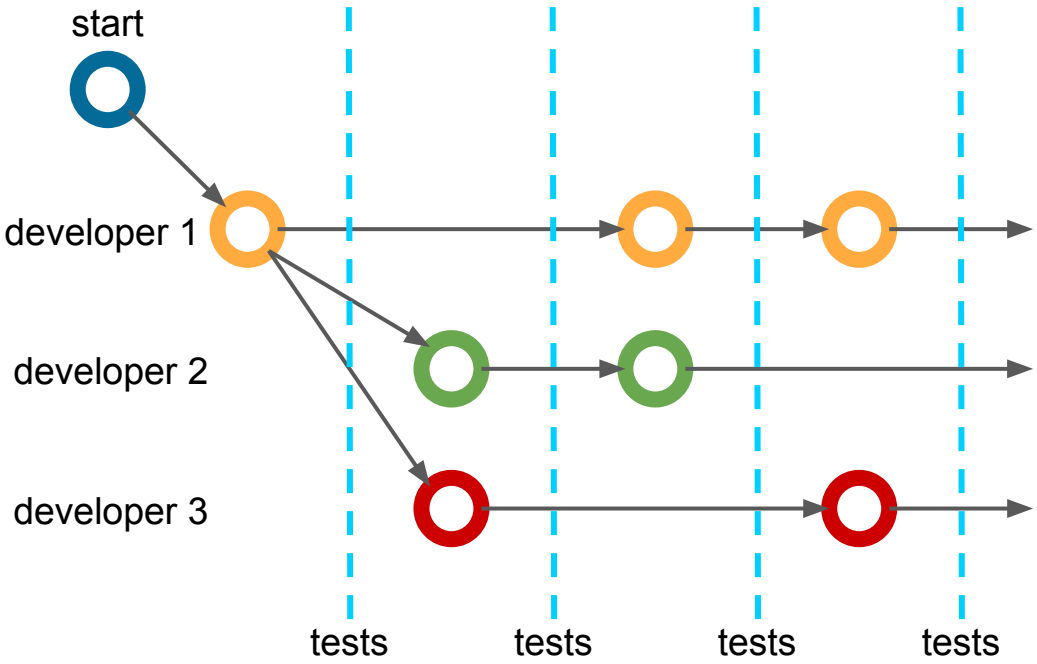
...and it’s still okay...



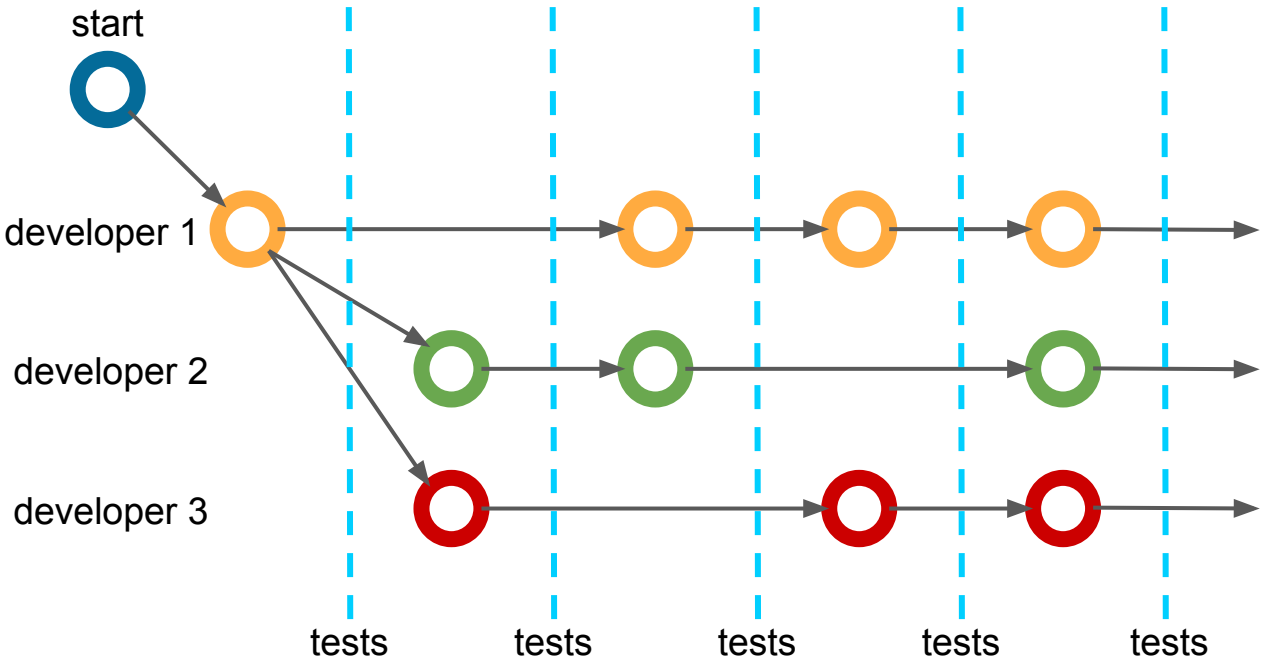
...and it's still okay...



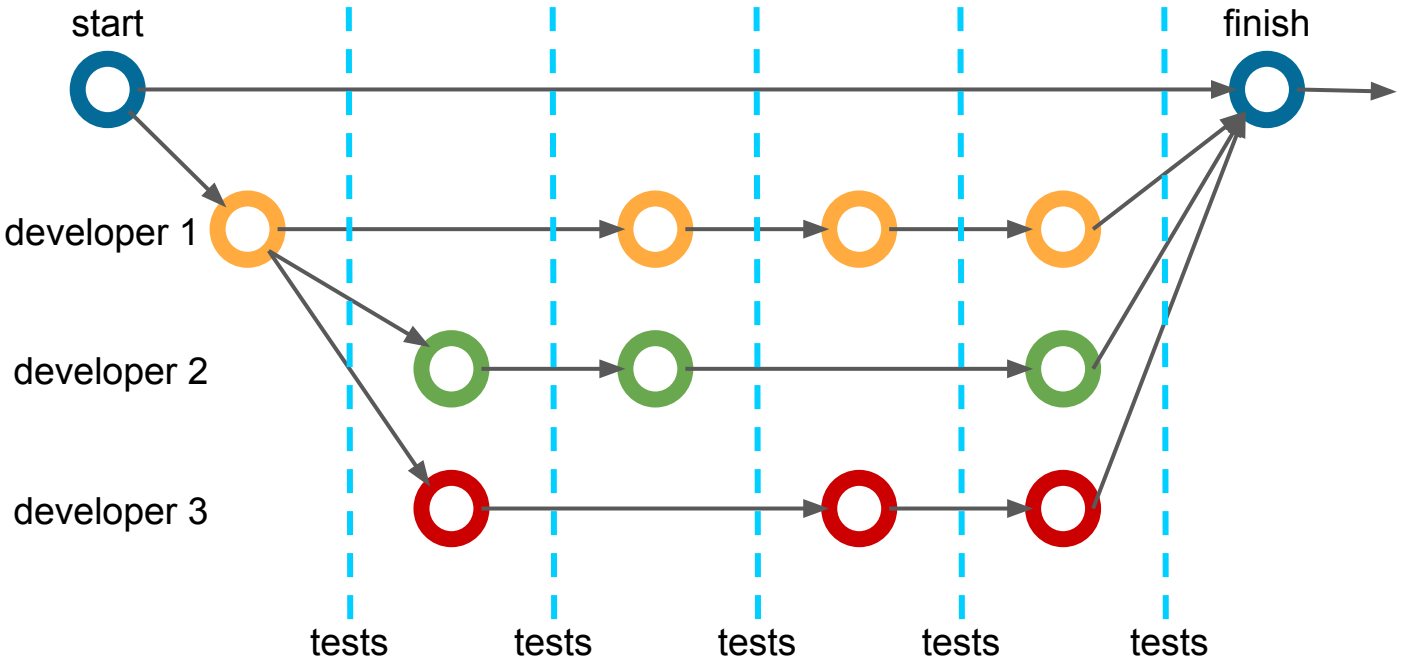
...and it's still okay...



...and it’s still okay...



...and everything is good!

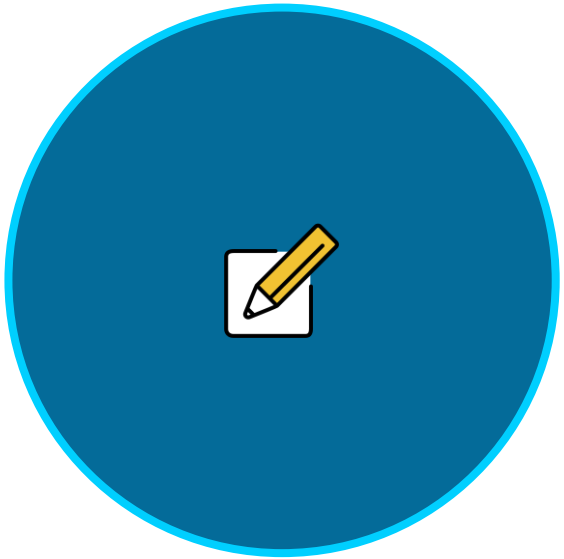


3/ Continuous Delivery

Making the new, updated product available as often as possible

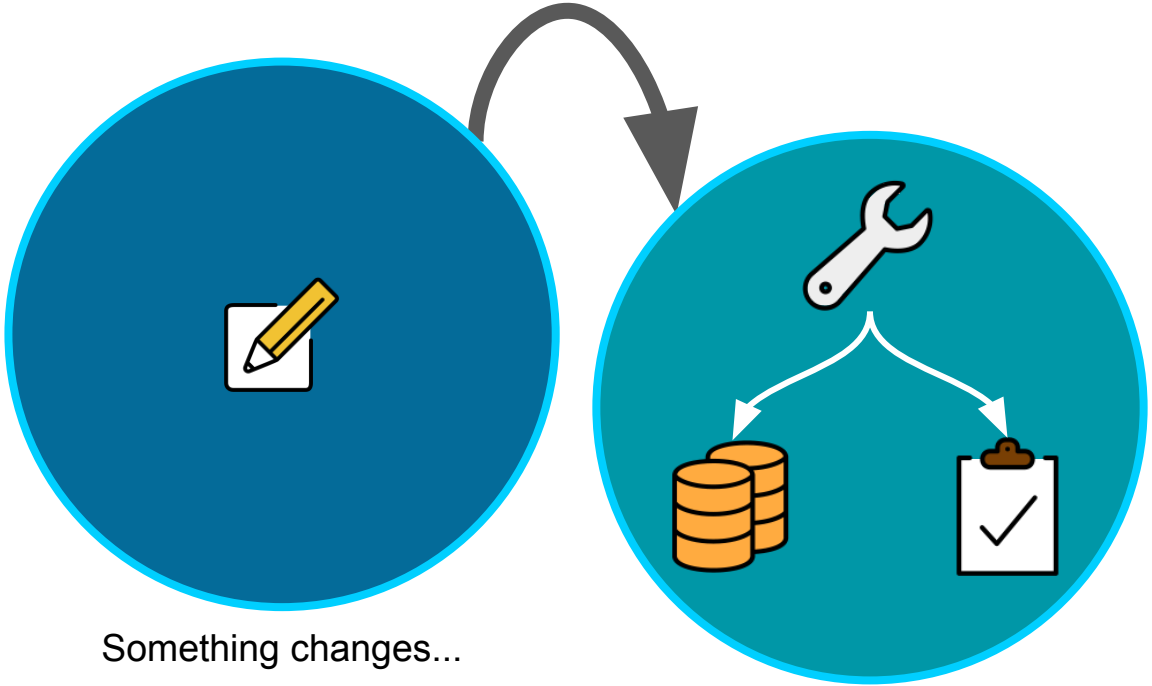
Continuous delivery is about deploying the most recent version of your product. That means every time some changes gets pulled into the app, the app gets deployed again.

The delivery



Something changes...

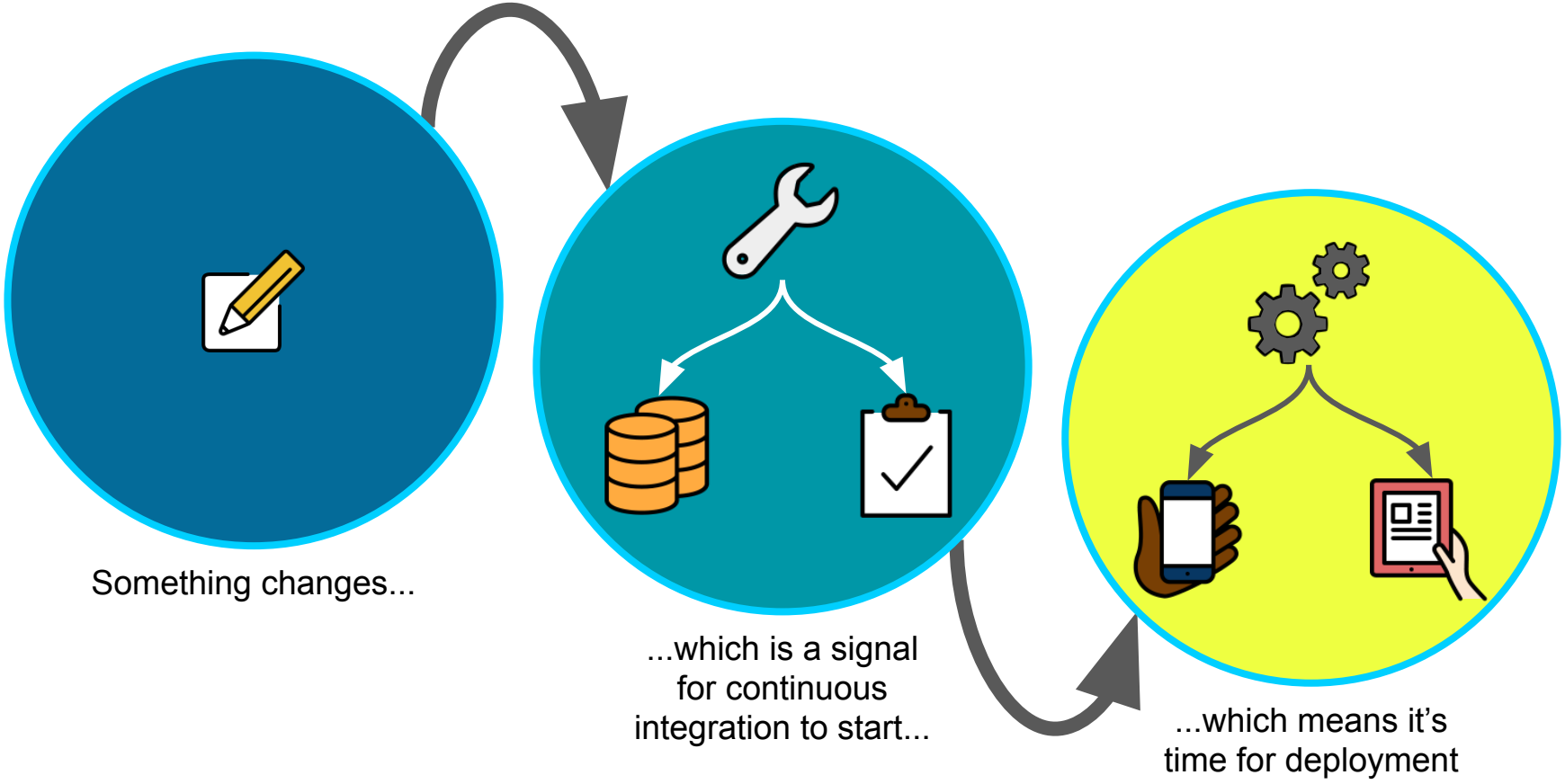
The delivery



Something changes...

...which is a signal
for continuous
integration to start...

The delivery



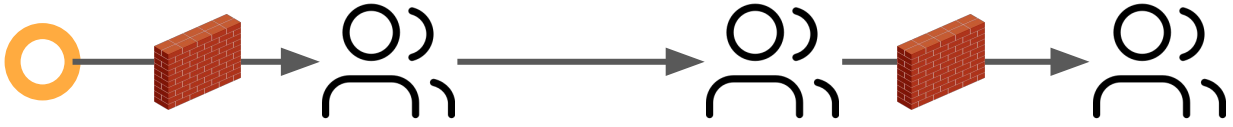
Something changes...

...which is a signal for continuous integration to start...

...which means it's time for deployment

4/ Automate ALL THE THINGS

A manual deployment process



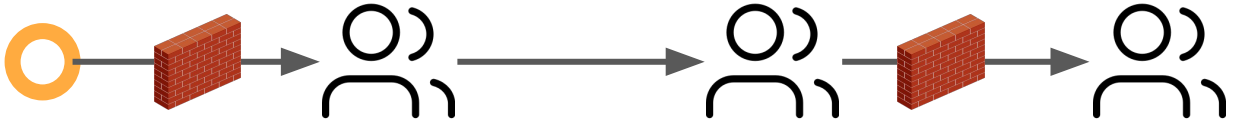
developers make a set of changes

staff do tests, often on their own computer

IT staff run security scans and approve or deny

product technical staff deploy the product

A manual deployment process



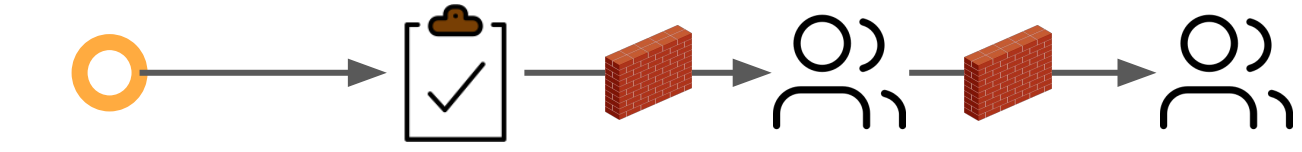
a developer makes a change

staff do tests, often on their own computer

IT staff run security scans and approve or deny

product technical staff deploy the product

A continuous deployment process



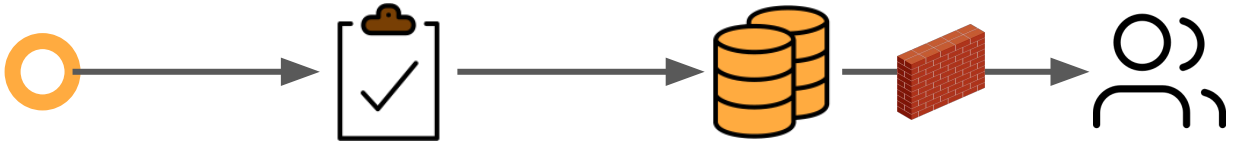
a developer makes a change

CI process detects change and runs tests

IT staff run security scans and approve or deny

product technical staff deploy the product

A continuous deployment process



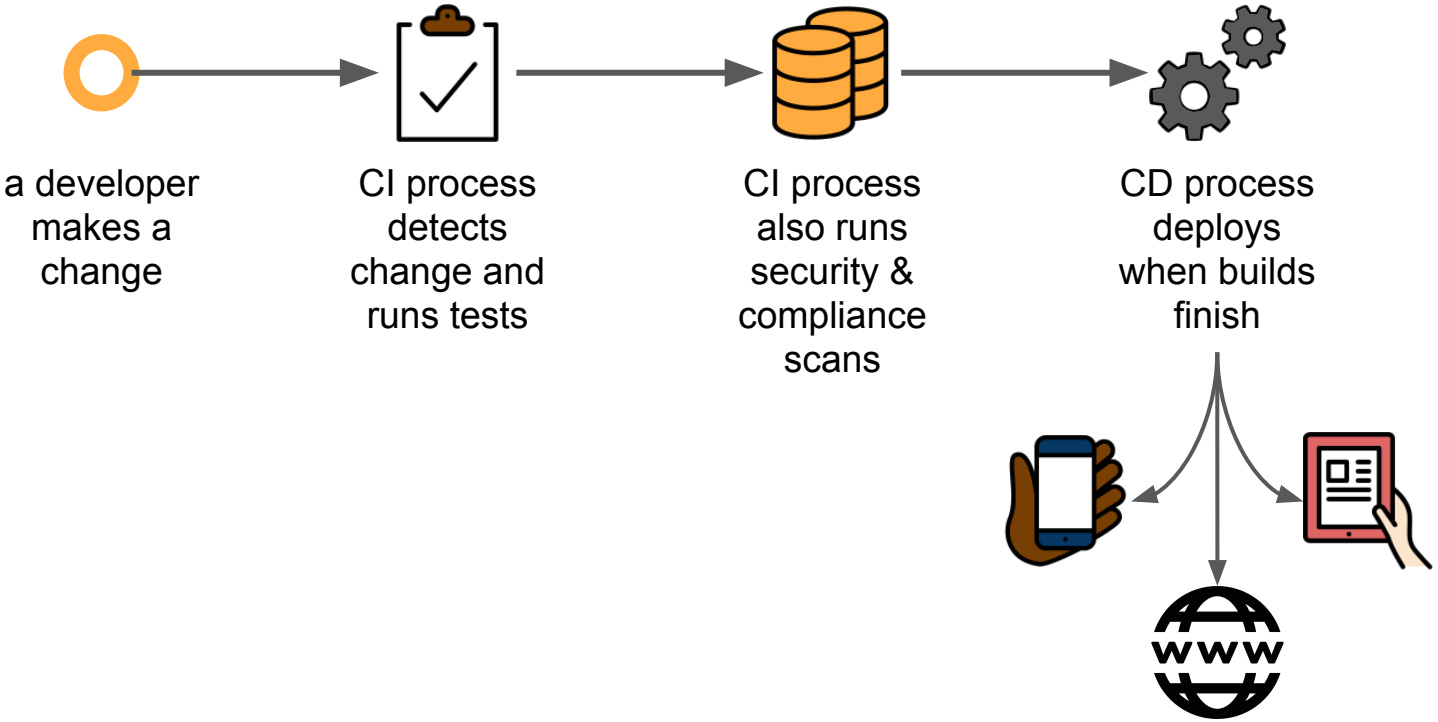
a developer makes a change

CI process detects change and runs tests

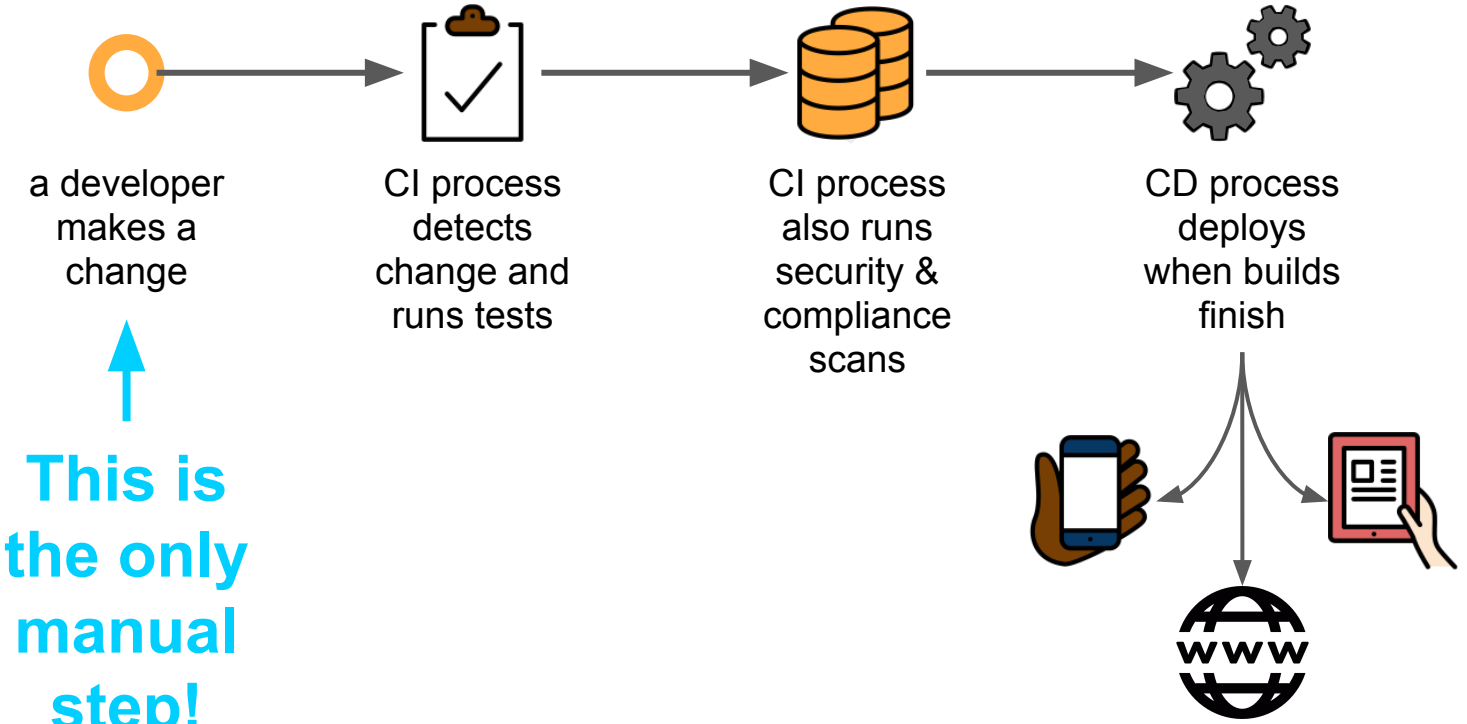
CI process also runs security & compliance scans

product technical staff deploy the product

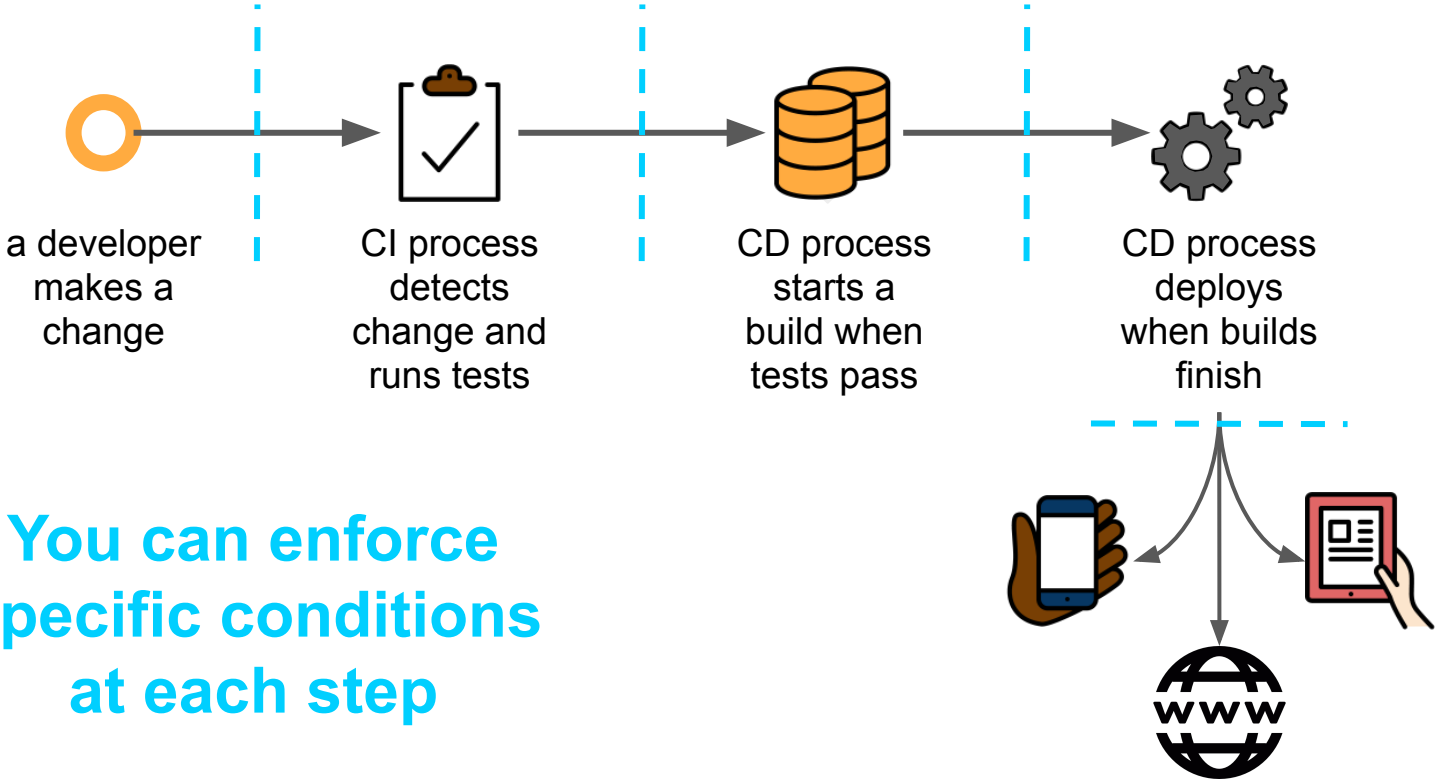
A continuous deployment process



A continuous, *automated* deployment process



A continuous, automated deployment process



You can enforce specific conditions at each step

The key takeaway:

DevSecOps is an enabler for **continuous, iterative improvement**

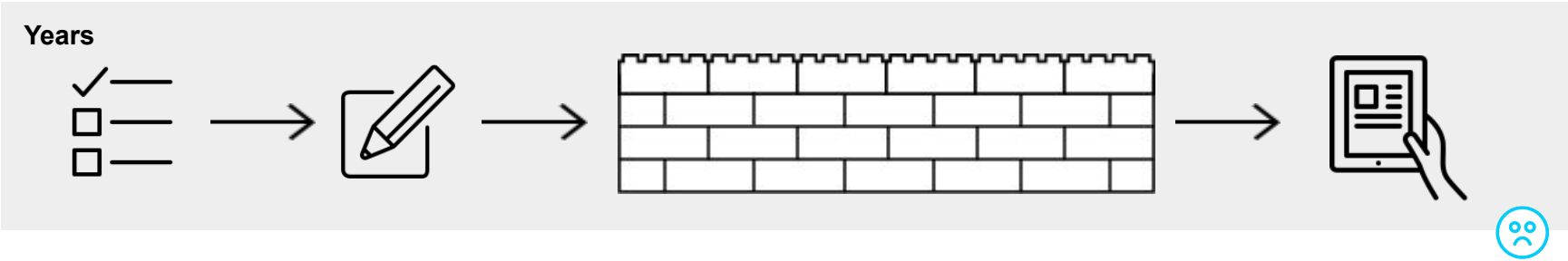
Agile acquisitions

Structuring RFPs to support modern development practices

18F

The traditional way of approaching ‘big bang’ procurements

FROM



It’s risky and makes juicy headlines

California junks $179 million Medicaid IT modernization project with Xerox

New Hampshire's Medicaid Billing Computer System Still Having Glitches

The system has cost more than $117 million since the first contract was approved in 2005 for about $60 million, with the most recent amendment adding $6.8 million to the price tag.

Oregon Health Authority director Lynne Saxton, whose agency was in charge of the $166.7 million Medicaid enrollment system, defends the finished product.

Maryland fires firm upgrading Medicaid technology, may seek money back

That’s how it was supposed to work, anyway. What happened instead was a case of epic mismanagement that threatens to leave Texas taxpayers on the hook for more than $130 million.

N.J. ends $118M contract designed to ease enrollment in Medicaid and other welfare programs

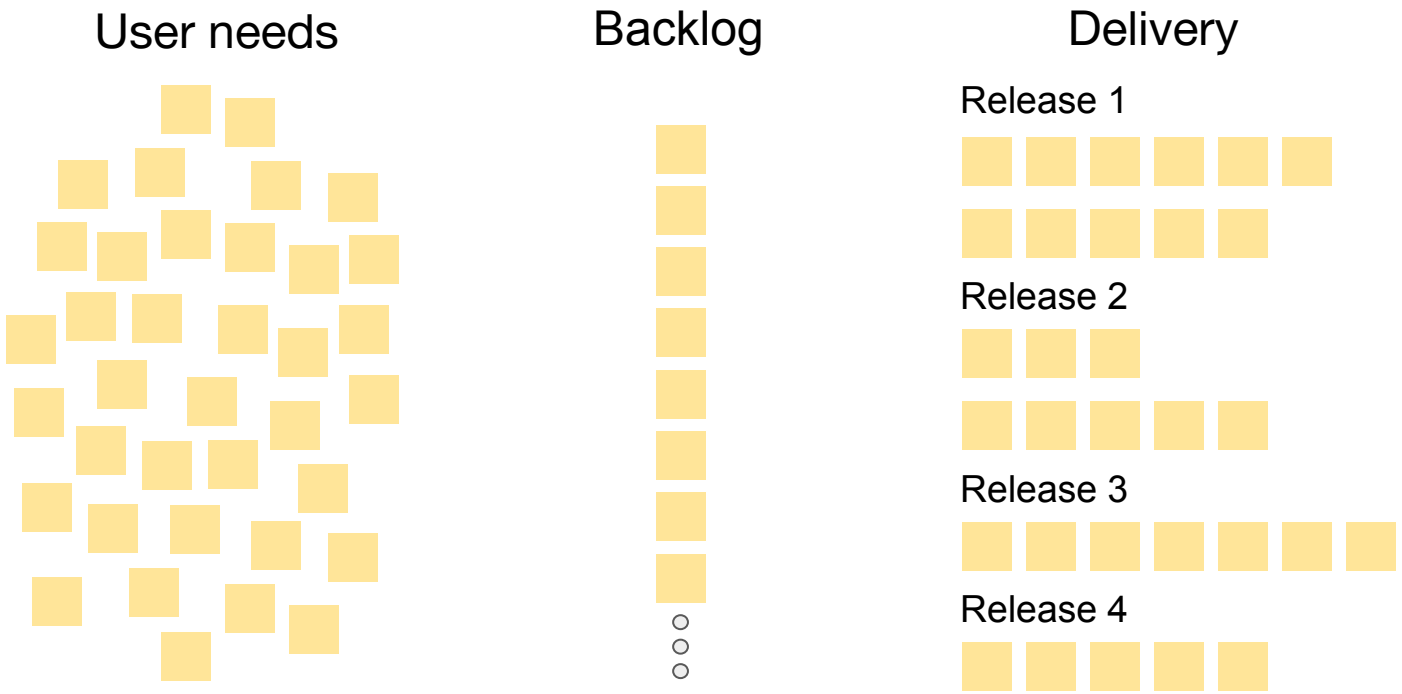
NEWS

Maine's Medicaid Mistakes

R.I. Gov. Raimondo wants payment-for-performance contract with computer contractor

Audit: Xerox's Montana Medicaid project could be 6 years late

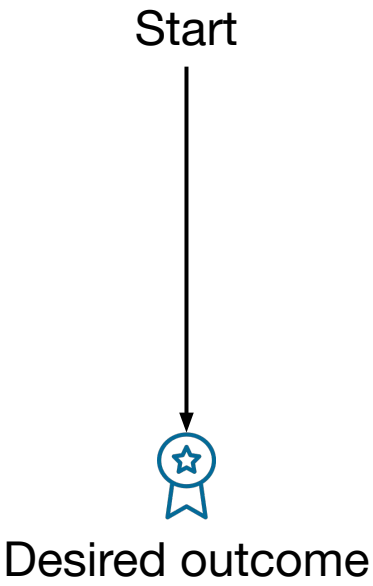
Thinking differently about requirements



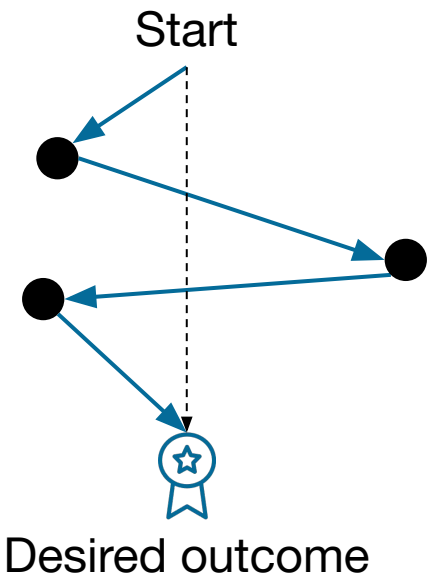
We know these elements need to be part of an agile software process.

Achieving a vision

Plan-driven

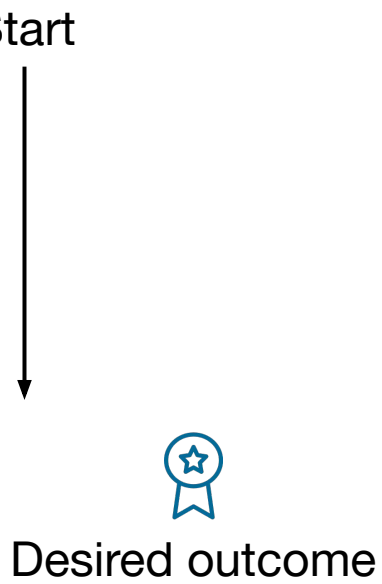


Iterative

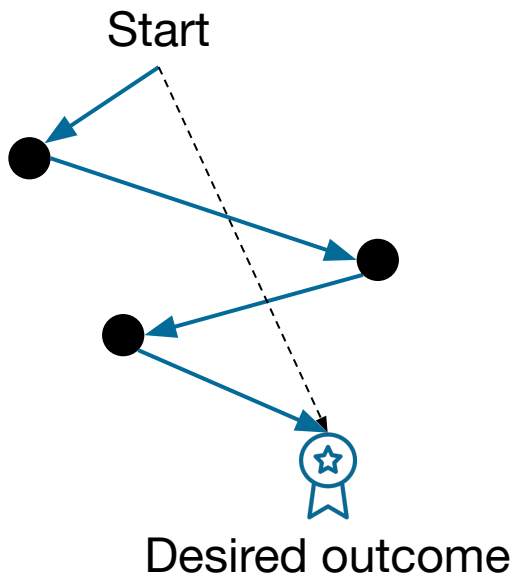


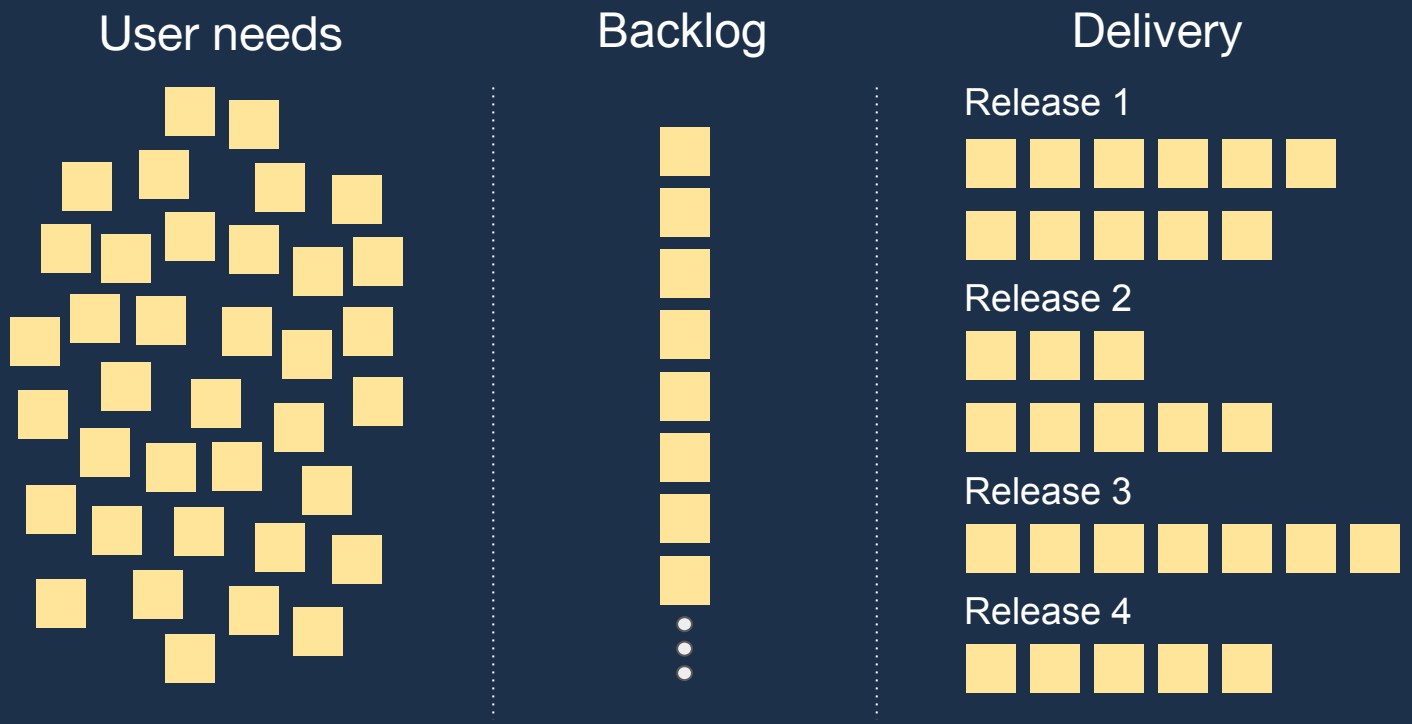
Achieving a vision

Plan-driven

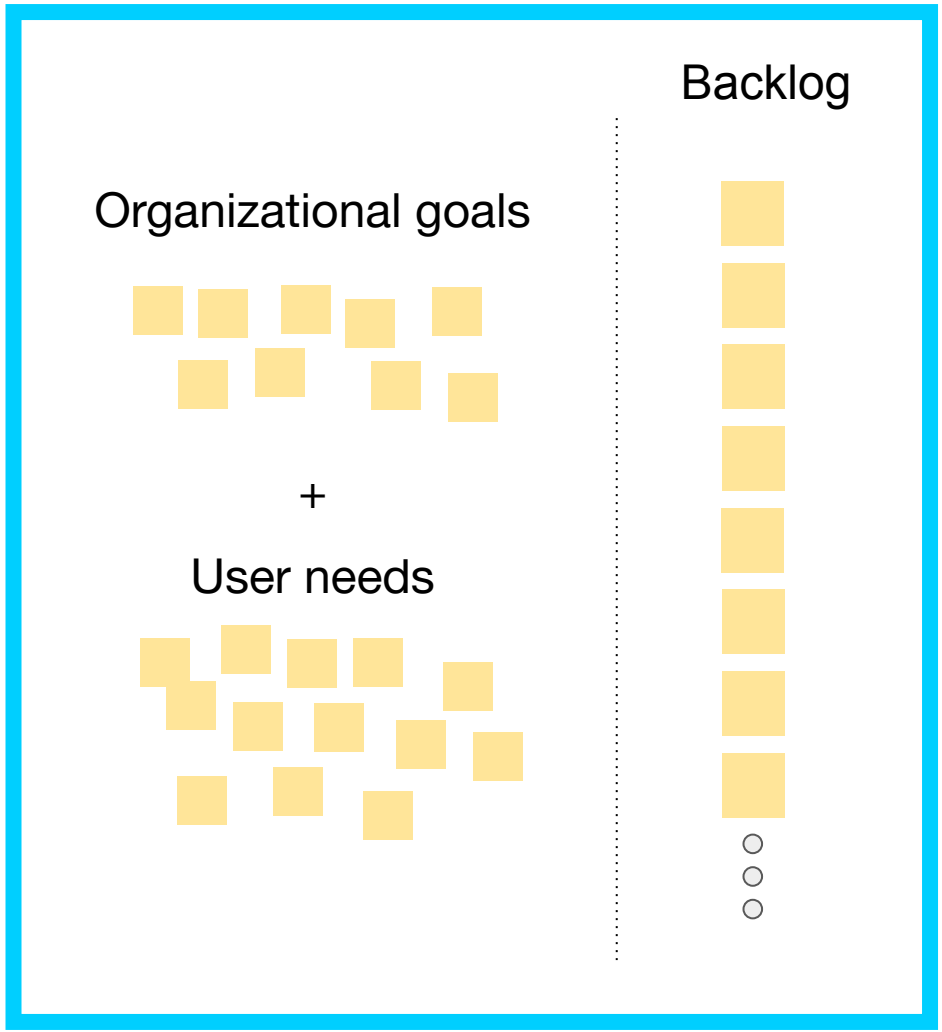


Iterative





Don't confuse the **destination** for the **journey**.



This is the **journey.**
You pursue a vision with a **dedicated** product team.

If you try to skip to the end, **you've destroyed iterative modularity.**

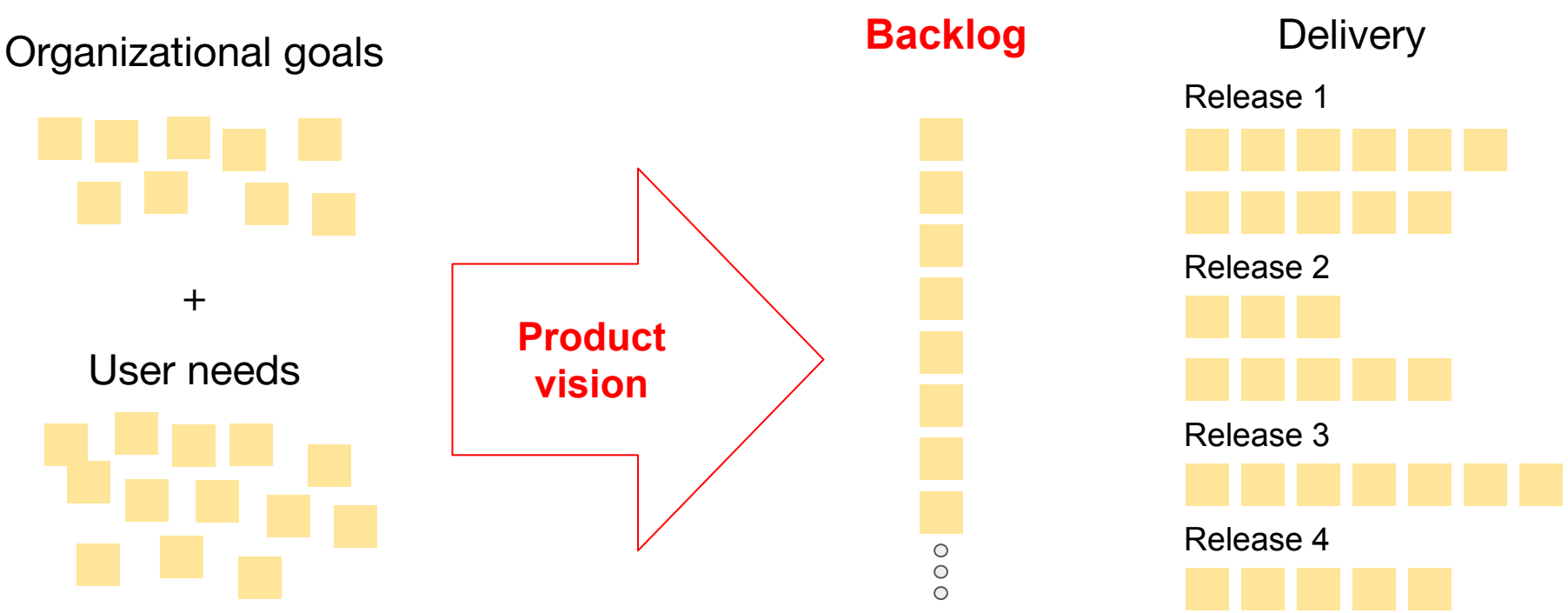
Defining this up front is waterfall.

The traditional approach to solving the incompleteness problem is trying to do a “better” up-front inventory:

- More time spent
- More people involved
- More “requirements,” either actual or potential, captured in writing.

It’s a fantasy *that doesn’t work*.

You cannot achieve a different output unless you change the input.

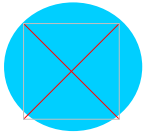


Two key artifacts bridge contracting and agile software development: the **product vision** and a **prioritized backlog** of work to do.

Contracts must also change to support this new, Agile approach

Adhering to Agile principles means that government as a **buyer must be descriptive, not *prescriptive***.

Procurement Principles



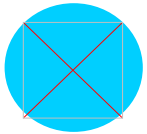
Structure contracts to focus on **working software**, not planning documentation

Structure contracts to **lower costs and avoid vendor lock-in**. Reduce switching costs.

Find vendors that use **human-centered design** to focus on software that meets people's needs

Ensure **full transparency on the vendor team** - working code to actual users seen early and often

Leverage **open-source software**, and select modern tech stacks to ensure long-term sustainability



Collaboration with vendor teams vs. **constant negotiation**.

Remember what you're buying

- With agile contracting, you'll be buying teams of developers and other experts from vendors.

- Remember, this isn't a product — it's the skills and expertise required to make and maintain a product. That's what lets you be flexible.

What does quality look like?

Communicate with vendors about what you expect in working software.
For example, a Quality Assurance Plan should include measures:

- Tested Code
- Properly Styled Code
- Accessible
- Deployed
- Documented
- Available
- Secure
- User Research and Design Artifacts

How does this work in practice?

- **Outreach**: Find quality companies that you'd like to bid. Get them excited about what you're trying to do.
- **Transparency**: Get vendors' feedback on a draft RFP.
- **Brevity**: Don't ask for 100's of pages of proposals. Consider using verbal interviews to get a sense of team dynamic.
- **Post-award management:** Gov't owns the product decisions, but works collaboratively with the vendor. Use a Quality Assurance Surveillance Plan so vendors know what's expected from their research and coding activities.

Questions
& discussion