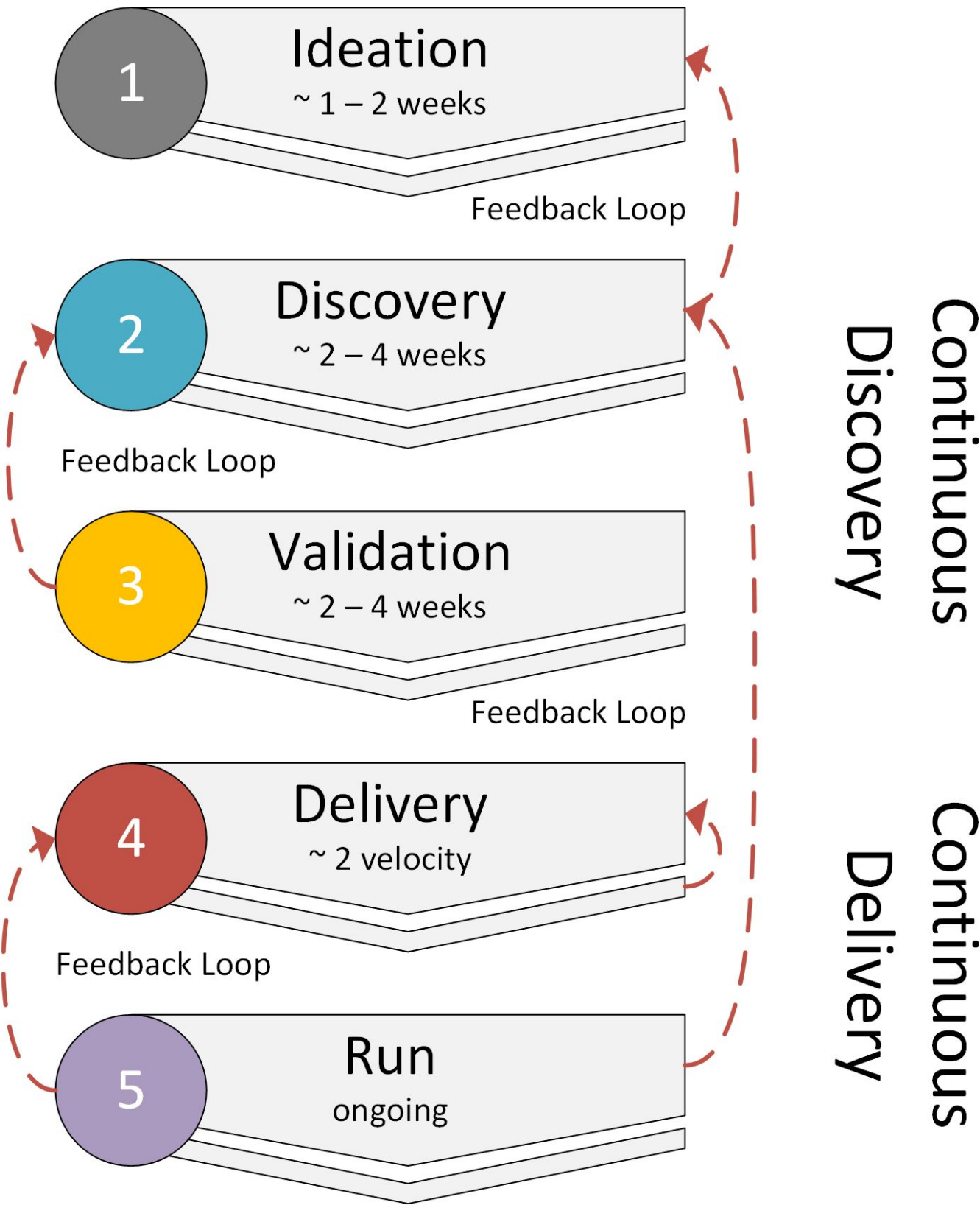


# Delivery Framework

- Outcome Delivery Framework
  - Overview
  - Ideation
  - Discovery
  - Validation
  - Delivery
  - Run
- Perspectives
  - Software Agile Delivery Overview
  - Research & Design: Discovery Plan

# Outcome Delivery Framework

# Overview





# Ideation

~ 1 to 2 weeks

We have lots of insight from multiple sources we think might be worth exploring.

## ➔ Input

- Customer insights
- Market Landscape
- Technology Landscape
- Business Objectives/ Strategy
- Service performance insight
- Idea capture form

## Output ➔

- Identified sponsor
- Prioritised idea candidates to take forward into discovery
- Draft vision
- Value statement
- Idea description
- Strategic fit (relative value impact on existing OKRs)
- Key customer segments and jobs to be done
- Big assumptions to address
- Ideas backlog

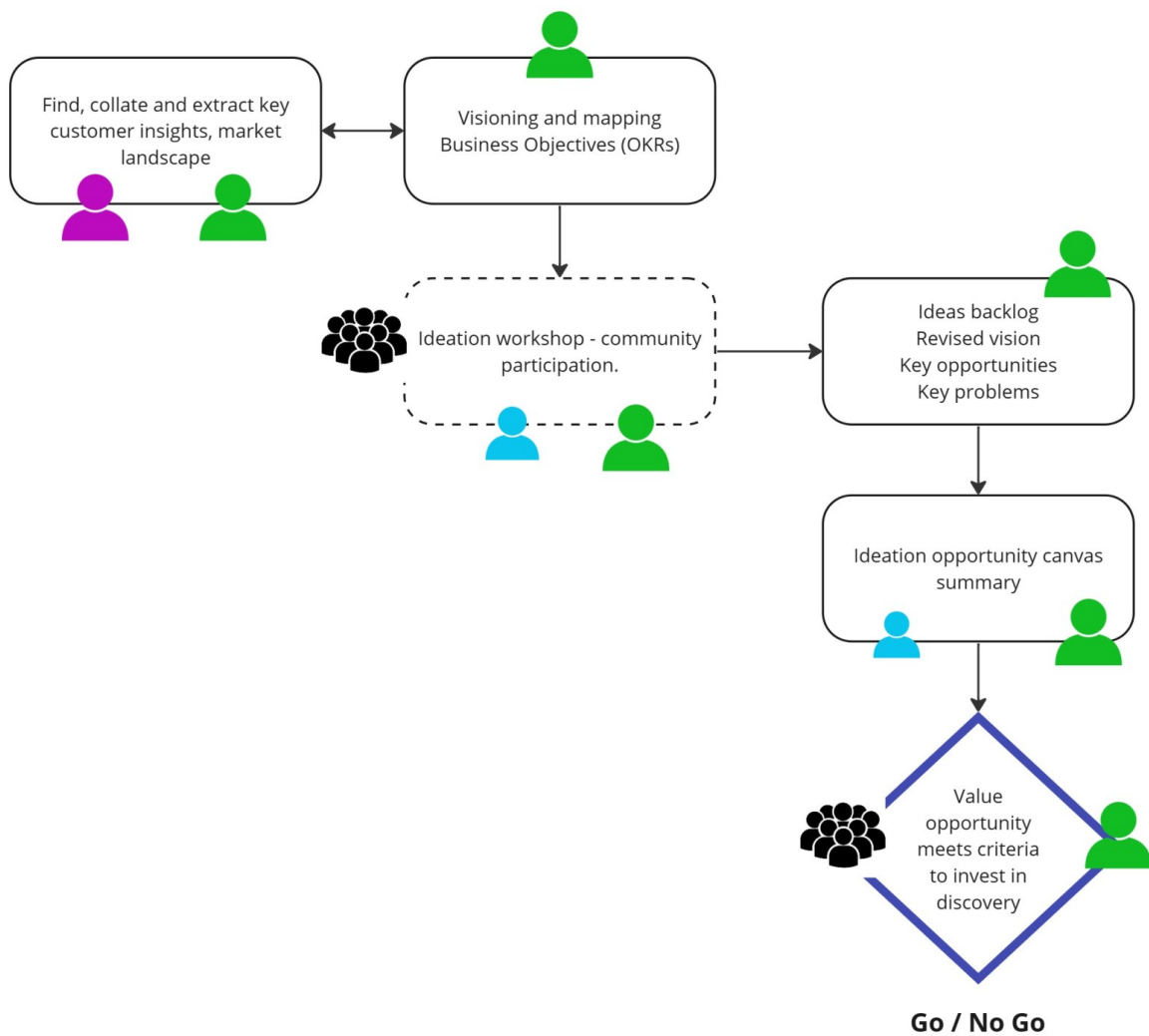
The above outputs are typically summarised within an opportunity canvas

## ☐☐ Golden Thread Practices


- [Brainstorming](#)
- [Visioning](#)
- [Value definition](#)


## ☐☐ Golden Thread Techniques

- [Amazon PR release](#)
- [Ideation workshop](#)
- [Value proposition canvas](#)
- [6 thinking hats](#)
- [OKR mapping](#)
- [Idea capture form](#)



# Key capabilities


Core Team 

User Research 


Service Design 

UX Design 


Tech Architecture 


Product 

Development 

Operations 

Delivery 

Data / Analytics 

Forum 

# Discovery

~2 - 4 weeks

We have prioritised ideas we want to explore in more detail so we can confirm it solves a real problem for the customer

## ➔ Input

Problems to Solve

or

Ideation output

- Ideas backlog
- Draft vision statement
- Agreed sponsor
- Capacity / Plan
- Opportunity Canvas
- Value statement

## Output ➔

- Defined Problem Statement
- User Segmentation
- Hypotheses
- Research
- Data Insight
- Key unknowns / assumptions
- Indicative size of opportunity
- Baselined vision statement
- Refined Value Proposition
- High level outcome roadmap
- High level current technology baseline

The above outputs are typically summarised within a discovery paper



# ☐☐Golden Thread Practices

- [Visioning](#)
- [Root cause analysis](#)
- [Generative research](#)
- [Human centred design](#)
- [Hypothesis thinking](#)
- [Value definition](#)
- [Value Prioritisation](#)

# ☐☐Golden Thread Techniques

[5 Whys](#)

[Value Proposition Canvas](#)

[Empathy Mapping](#)

[User Journey Mapping](#)

[OKR Mapping](#)

[Problem Definition](#)

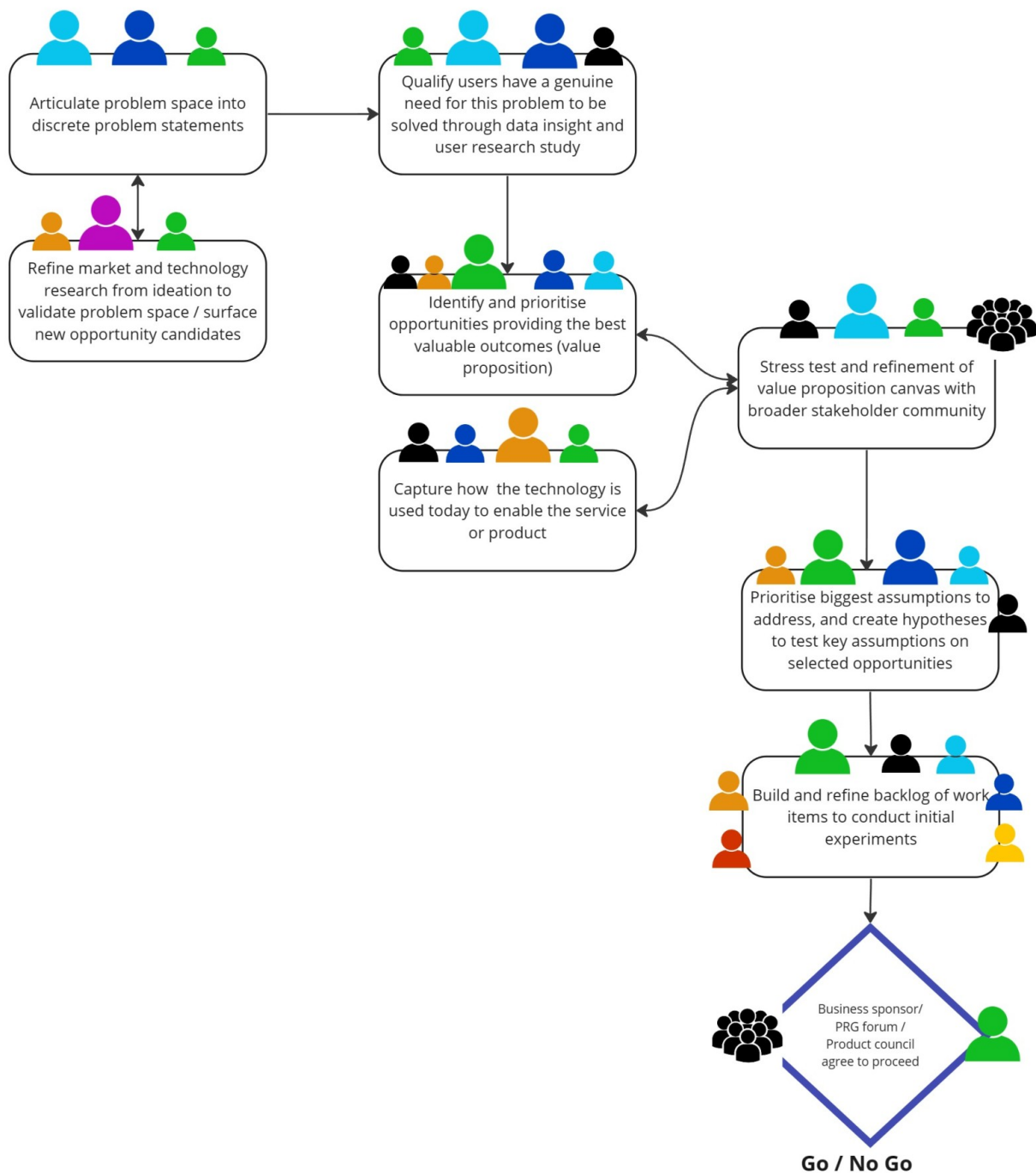
[Opportunity Solution Tree](#)

[User interviews](#)

[Observational study](#)

[Surveys](#)

[Assumption impact quad...](#)



## Key capabilities





# Validation

~ 2 to 4 weeks

We want to find a solution that demonstrably solves a valuable customer problem

## → Input

- Prioritised experiments (test and learn cards)
- Defined Problem Statement
- User Segmentation
- Hypothesis
- Supporting research
- Key unknowns / assumptions

## Output →

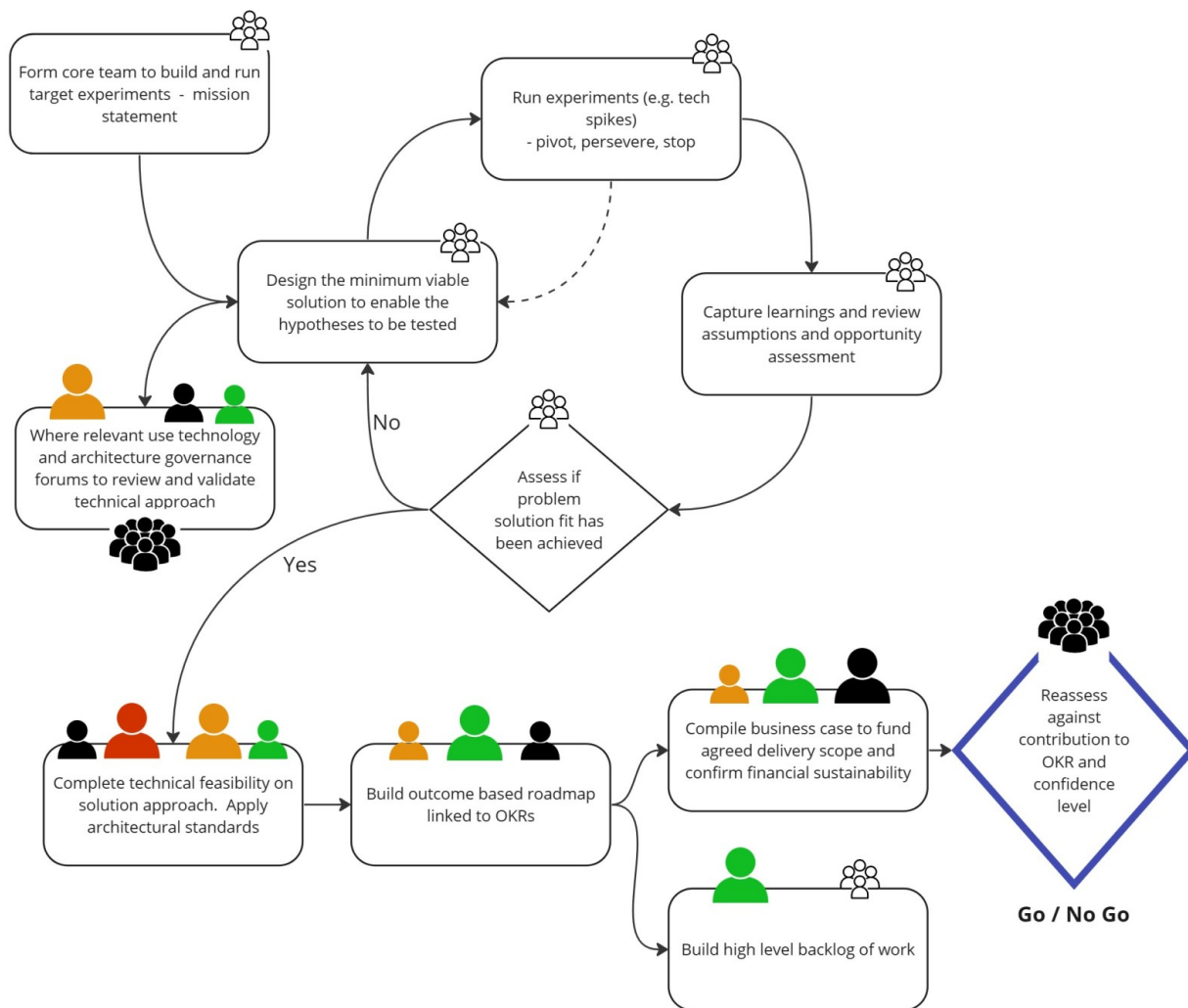
- Business case
- Emergent Hypothesis
- Solution Approach + Options
- Solution architecture design
- Service Design
- Backlog (DEEP)
- Refined outcome roadmap
- Ruthless MVP candidate

## ☐☐Golden Thread Practices

- [Generative research](#)
- [Backlog refinement](#)
- [Mobilisation](#)
- [Lean experimentation](#)
- [Prototyping](#)
- [Data driven decision m...](#)
- [Value based prioritisa...](#)

## ☐☐Golden Thread Techniques

- [Team Charter](#)
- [Test and Learn Cards](#)
- [Hypothesis Creation](#)
- [Cost of Delay Prioriti...](#)
- [Effort/Value Matrix \(2...](#)
- [A/B Testing](#)
- [Card Sorting](#)
- [Data Analysis](#)
- [NPS/ CSAT metric](#)
- [Lo-Fi Prototyping](#)
- [Service Design Bluepri...](#)
- [Usability testing](#)
- [User Story Mapping](#)
-



## Key capabilities



# Delivery

Optimal velocity target is ~2 weeks

We prioritise our backlog of committed work to maximise delivery of value to our customers as early as possible

## → Input

- Prioritised Backlog
- Acceptance Criteria
- Test Scenarios
- Release Plan / Road Map
- Reference Solution Architecture
- Wireframe / Assets
- Service Guide / Maps
- Measures for success

## Output →

- Deliverable Increment
- Test Cases
- Documentation (Min Viable Doc)
- Solution Architecture
- Service Guides
- Released solution
- End user documentation
- Internal and External Comms

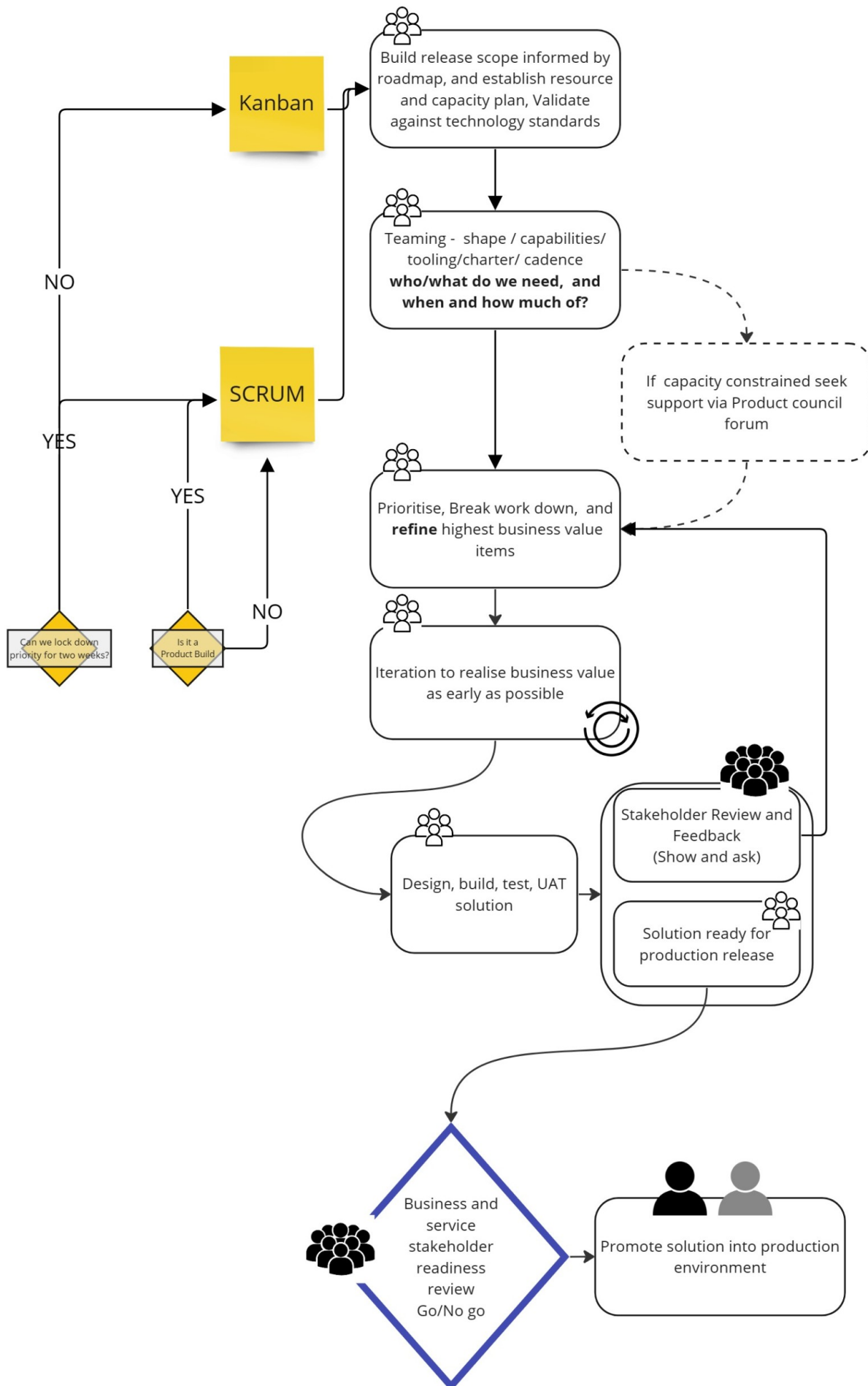
# ☐☐Golden Thread Practices

- Visualisation
- Value Prioritisation
- Backlog refinement
- Estimation
- Incremental Delivery
- CI/CD Continuous Integ...
- Source Control
- Automated Testing
- Release Management

# ☐☐Golden Thread Techniques

- User Story Mapping
- 3Cs (Backlog Refinement)
- User Story Writing
- Cost of Delay Prioriti...
- Effort/Value Matrix (2...
- BDD - Behaviour Driven...
- Vertical Story Slicing
- User Story Splitting
- Estimation - Planning ...
- Sprint Planning
- Sprint Retrospectives
- Sprint Review
- Infrastructure as Code...





## Key capabilities





# Run

and improve through measuring and learning

ongoing

We measure metrics that matter, continually capture customer feedback and proactively identify new opportunities to improve the product experience

## ➔ Input

- Service analytics
- Customer feedback analytics
- Operational analytics
- Customer service insight

## Output ➔

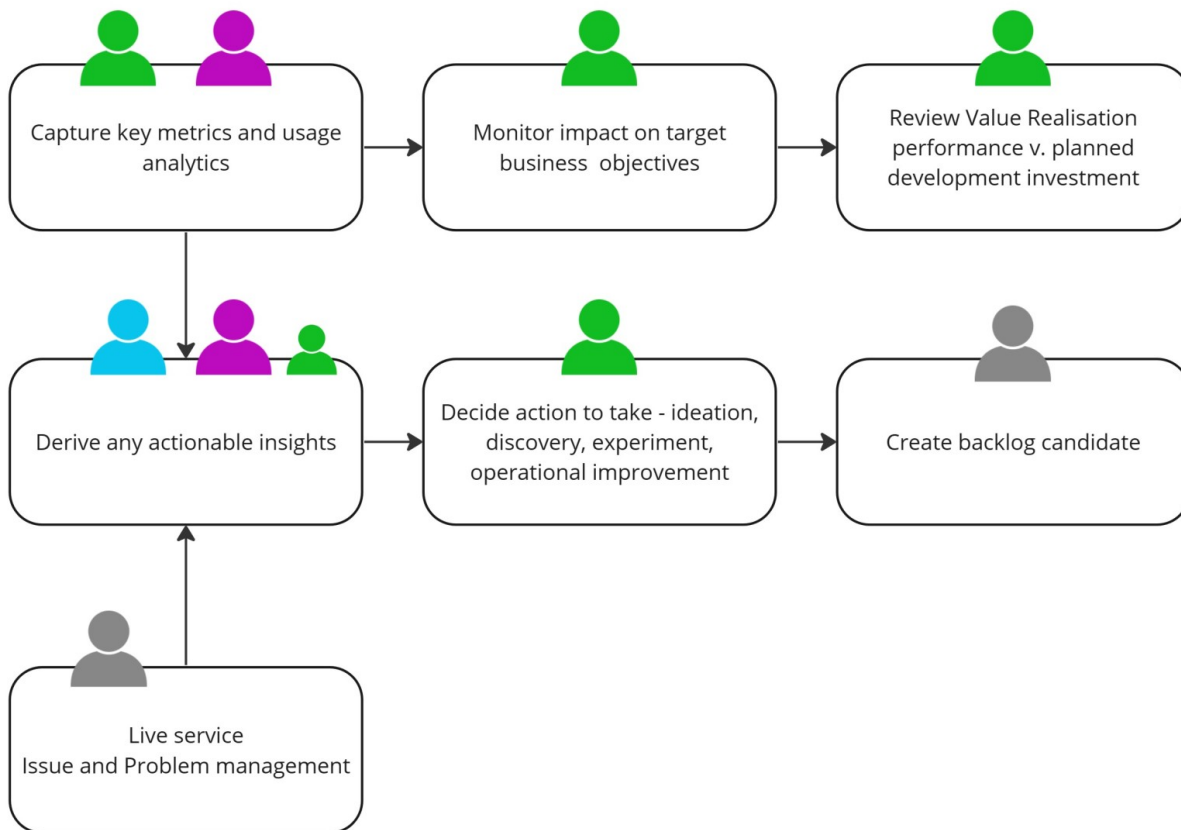
- Idea candidates for discovery
- Defects backlog
- Operational improvements
- Candidates for experiments

## □□ Golden Thread Practices

- [Proactive monitoring](#)
- [Service Operations - I...](#)
- [Continuous Discovery](#)

## □□ Golden Thread Techniques

- [Cohort Analysis](#)
- [Logging and Monitoring](#)



## Key capabilities



# Perspectives

# Software Agile Delivery Overview

## Backlog Refinement and Sprint Planning

### Adding to the backlog

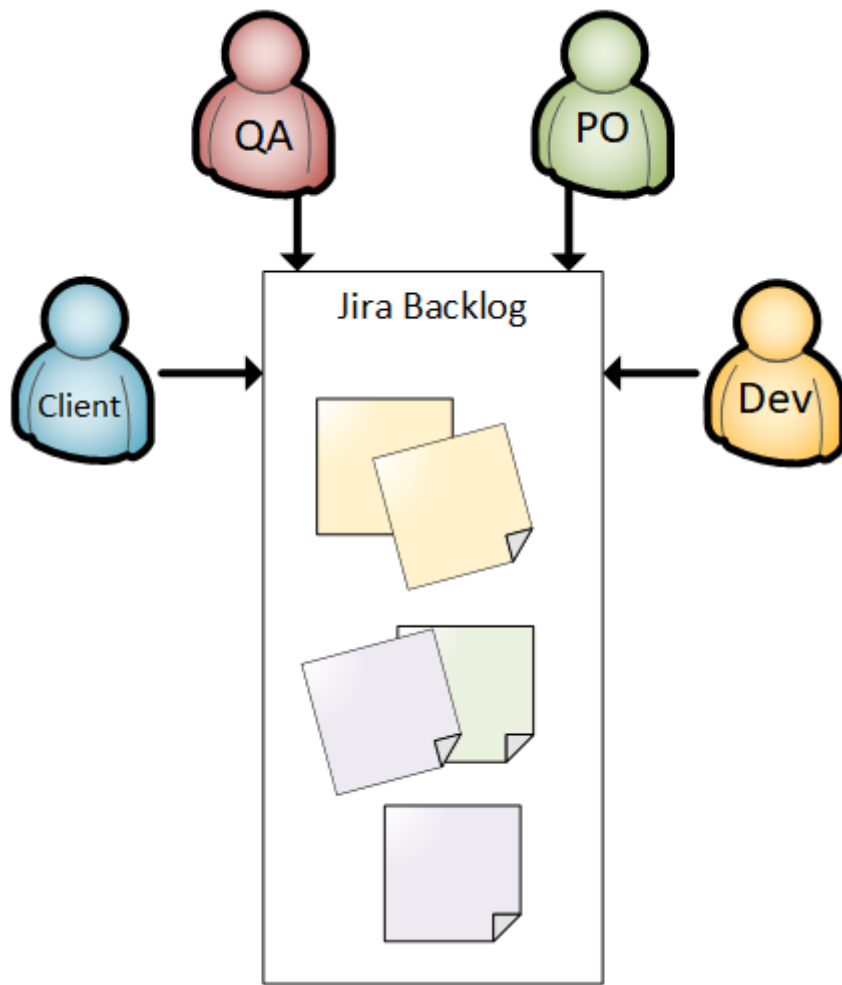
We run a flexible system where anyone in the project can add an issue to the backlog in Jira.

The backlog will contain a mix of technical tasks, bugs, user stories and spikes .

\*see below

Larger pieces of work, that might span several sprints, are entered as Epics. Other issues can be assigned to an epic to show that they will contribute to it.

Just because an issue is in the backlog, does not necessarily mean that it will be worked on.



## Refining the backlog

The development team must have a complete understanding of each ticket before they can agree to work on it. Each ticket must be understood at a business level and at a technical level.

Where the definition of done is not well understood, or there are edge cases that aren't specified, the ticket will be referred to stakeholders for clarification.

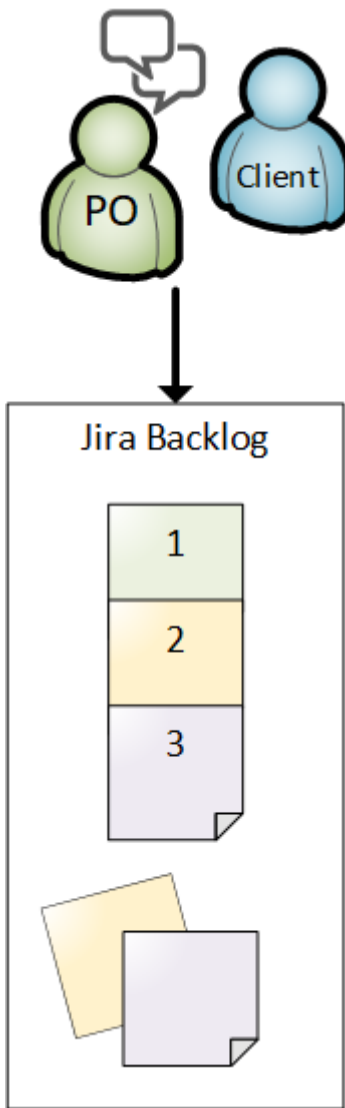
Where the ticket is not understood technically, we will raise a time-boxed spike to investigate.

\*

For these reasons we prefer to plan our sprints well in advance, so that all preliminary work can be undertaken first.

Once a ticket is well understood the development team will collaboratively 'score' the ticket, reaching a multi-faceted consensus on the amount of work, the complexity of the task and their experience with tasks of that nature.

Our Product Owner and the Stakeholders will collaborate to prioritise the backlog, with stories that are 'Ready to Play' at the top.



## Sprint Planning

The top tickets in the backlog are moved into the current sprint until it contains tickets with a combined score equal to the team's velocity (based on past experience).

It is worth noting that two, equally performant teams might allocate different scores to the same ticket, if they were both presented with it, and will have different sprint velocities accordingly. Our scoring system is deliberately unitless.

## During a Sprint

The functional test team will create a test plan from the User Stories in the backlog.



# Development

Developers will write code to fulfil the selected user stories, technical tasks and so on. Our code is supported by the automated tests we write, to prove our code works first in isolation and then when integrated with the rest of the solution.

In parallel to this, our automated testers will produce the scripts to validate the work produced at a UI level.

## Review

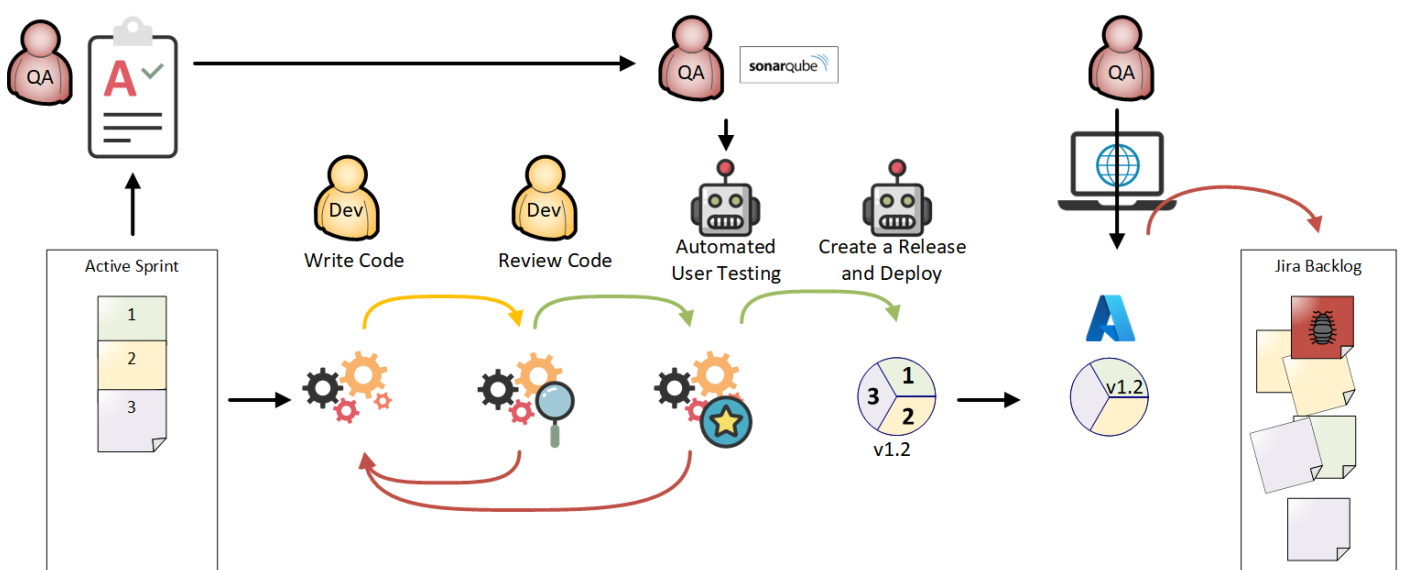
Once a ticket has been completed, we automatically run every unit and integration test for entire project, to make sure the changes we've made in one area of the project don't adversely affect another.

The code is then subject to the review of at least two other development team members. We verify that the work completed is of good quality and that it will fulfil the definition of ready, as specified by the ticket in Jira.

## Internal Signoff

Our code is then deployed to a working environment where we can then run every UI test for the entire project.

If QA are happy with the result they will sign off the ticket. If bugs are found after this point, they must be added as new tickets on the backlog and go through the usual triaging process with the product owners and stakeholders.



# Release management

As the development process continues, we accrue new completed work in our development environment, for internal sign off, that is not in our staging area, for external sign off.

## External Signoff

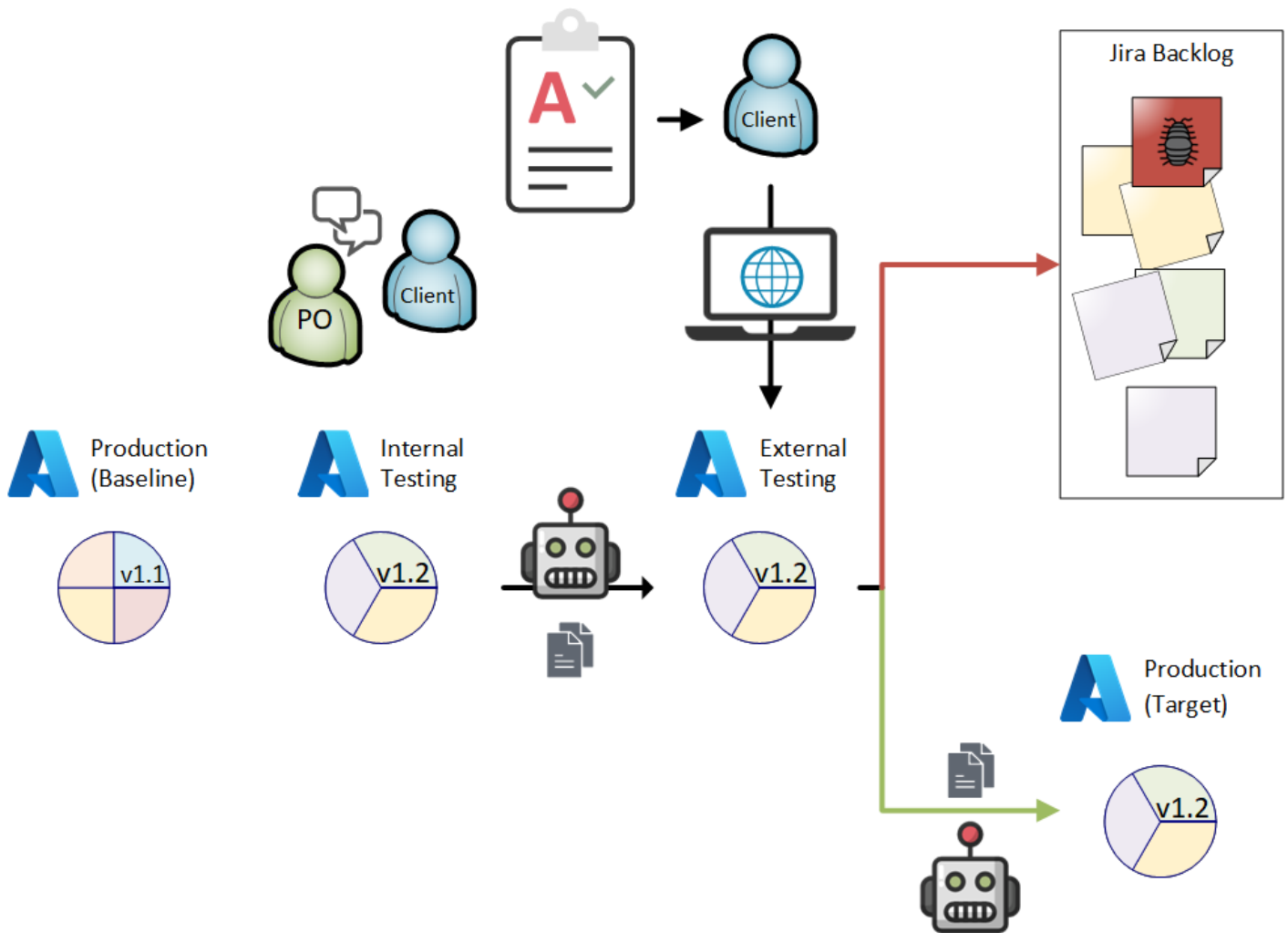
Stakeholders can communicate that they would like to see a new version of the product at any time.

The work that is signed off internally is bundled into a release build and migrated the external sign off environment. We will provide our own functional test plan to the stakeholders, as a good starting point, but the stakeholders are then free to test the system as they see fit.

## Going live

If the stakeholders are happy with the product then the version that is currently in our external testing environment is promoted to our live environment.

This process can happen at any time, although it is advisable to do this earlier in the week in case the need to rollback our work arises.



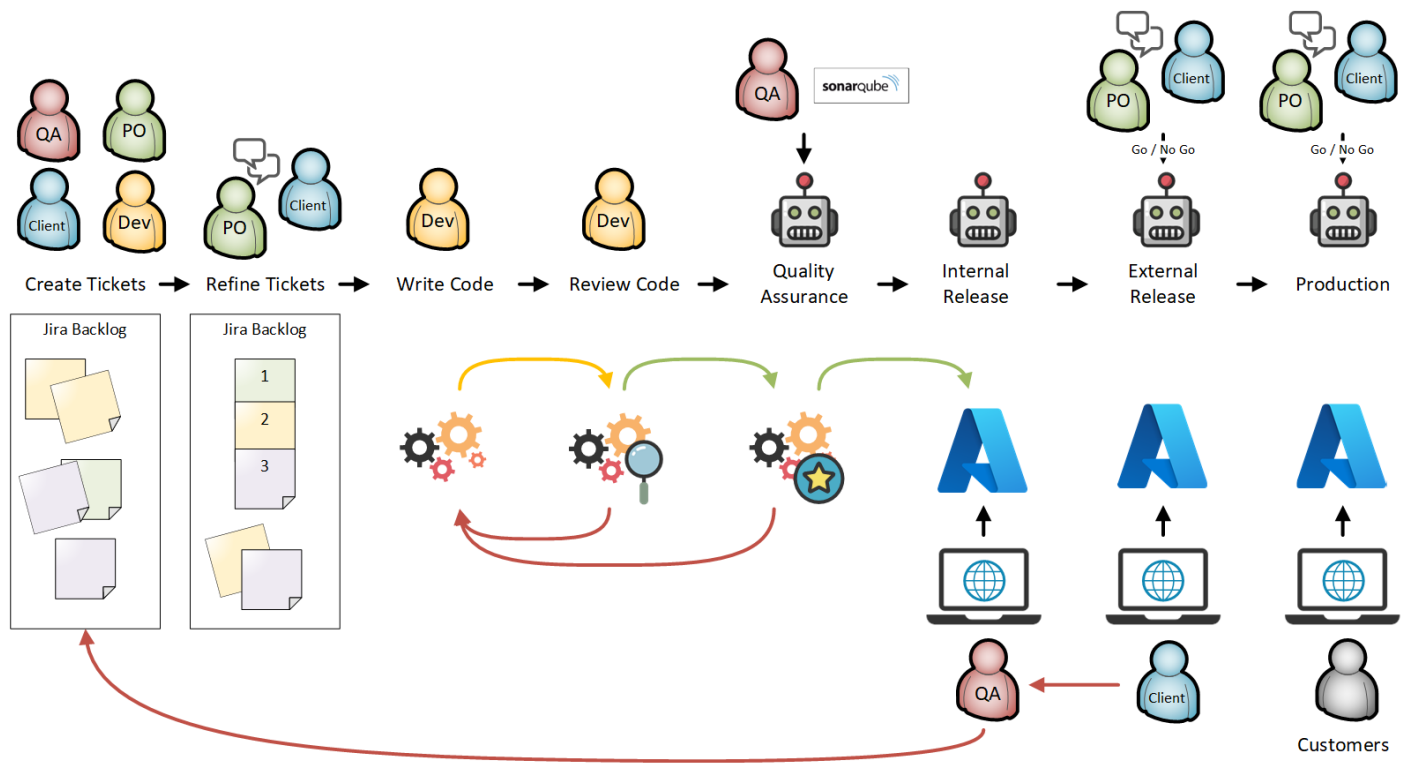
## Q: Where are the Gant charts?

Our development, testing and infrastructure team practice continuous deployment.

We can deliver code to our external testing and production environments rapidly, and on demand. We can also roll back changes just as quickly.

Our formal meetings to review each sprint's work may be a good time for stakeholders to request a new version of the product, although this could also happen at any other time.

The backlog in Jira delivers transparency over our work and allows for dynamic re-prioritisation of that work by the stakeholders.



# Research & Design: Discovery Plan

Scoping, Resourcing & Planning Discovery

## Pre-Discovery Planning

**Everything that needs to be in done by R&D before Discovery Starts**

- Identify key Stakeholders
- Identify Key Services Involved.
- Plan & Schedule Primary Research
- Kick-off brief session with Core Discovery team
- Plan Stakeholder / Service Workshops
- Define Discovery brief & Manage expectations

## Discovery Team

### Core Discovery

- Product / Project Manager
- User Researcher
- Service Designer

### Supportive Discovery

- Technical Architect
- Data Analyst
- UX Designer

## Discovery Phase

**We can't make an informed decision**

- Primary Research
- Current State Blueprint
- Expert Review
- Competitor Analysis

- Existing Systems & Processes
- Existing Analytics
- Identify best practice guidelines & principles for the service
- Workshops with the Services, Contact Centre
- Technical Discovery

# Discovery Conclusion

## We can make an informed decision

### Key Owner: **Architect, UX, UR & SD**

- Current State Blueprints
- Primary Research
- Existing Architecture
- Technical Discovery
- Expert Review
- Competitor analysis
- Existing Analytics

### Key Owner: **Product Manager**

- Defined problem statement
- Hypothesis
- Key unknowns & Assumptions
- Refined Value Proposition
- High level outcome roadmap
- Experiment backlog
- Indicative size of opportunity