Welcome!

Secure Agile

Hello

About me...

- 30 years in the industry
- 25 years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!

Agenda

- Learning objectives
- Agile vs. Waterfall
- Planning with security in mind
- Security-focused testing

How We’re Going to Work Together

- Slides and words to highlight key concepts
- Demos to bring those concepts “to life”
- Discussion groups and lab work (which will take place in sandboxes provided via AWS WorkSpaces) for hands-on reinforcement
- NOTE: I welcome being interrupted – if you need more info, or clarification, or anything else, just break in and ask. I am here to help you.

Learning Objectives

Learning Objectives

- Understand key concepts and considerations when leveraging DevSecOps in an Agile environment to help “shift security left” during software development
- Construct and prioritize a threat model for an application being developed and use that threat model as a planning tool to guide each phase of the SDLC in a security-minded manner
- Speak to the value of DevSecOps and its consistent application as a set of standards and best practices

Learning Objectives

- Monitor, patch and scan for vulnerabilities in the Operating System (Windows and Linx) and underlying Infrastructure configuration
- Effectively utilize GitLab for Source Code Management and CI/CD, including:
    - Understanding and navigating practical activities related to source code repositories in GitLab
    - Effectively use Software Composition Analysis (SCA), Static Application Security Testing (SAST), and Dynamic Application Security Testing (DAST)

Agile vs. Waterfall

Waterfall Software Development

A sequential development process that flows like a waterfall through all phases of a project (requirements, design, implementation, testing, and deployment for example), with each phase completely wrapping up before the next phase begins.

Key aspects:

- Majority of research done up front
- More accurate time estimates
- More predictable release date

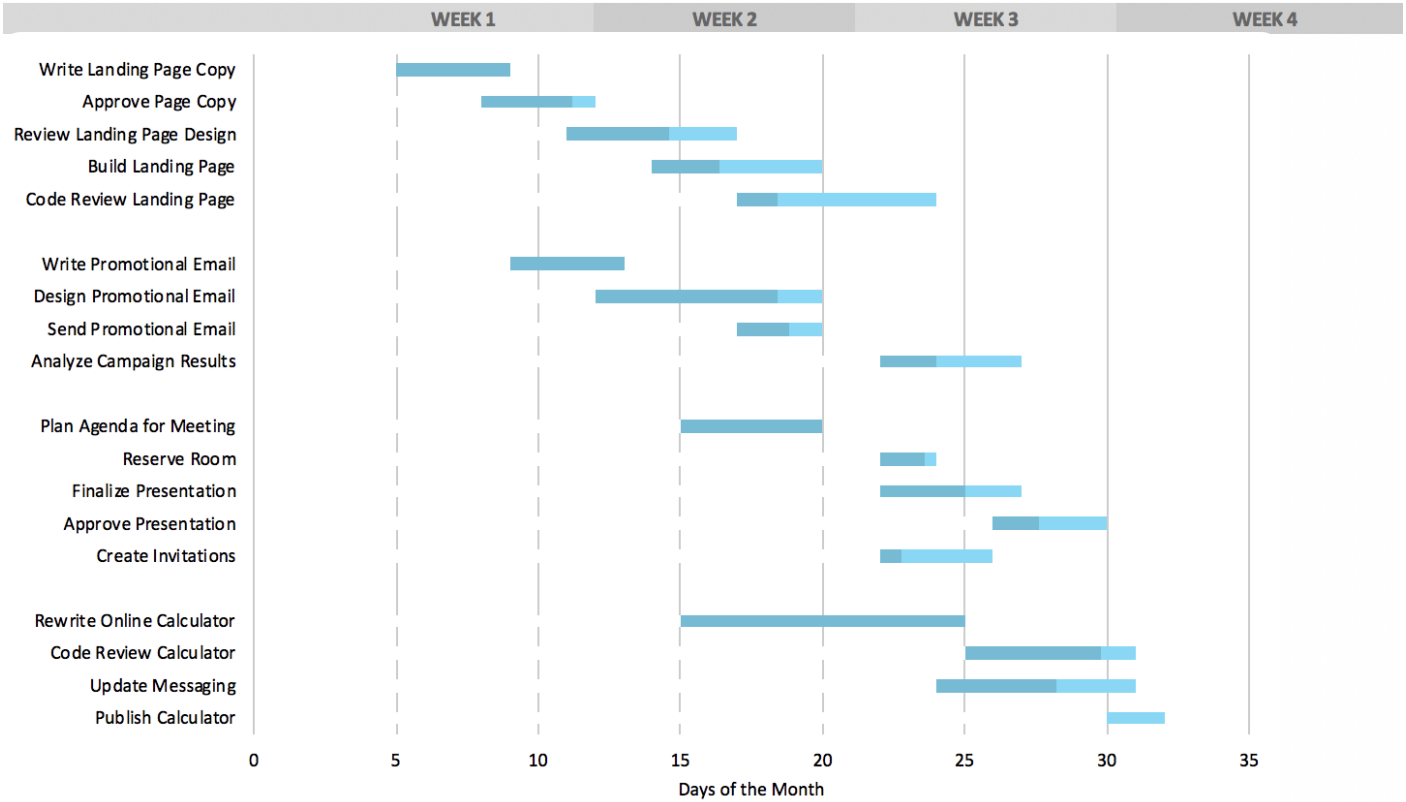
Cons:

- Process is brittle – can’t pivot easily in terms of changing requirements
- Long lead times – difficult to respond to rapid business evolution

Tools:

- Commonly use Gantt charts (or something similar) to track projects, subtasks and dependencies

Typical Gantt Chart



Agile Software Development

A group of software development methodologies based on iterative development.

Requirements and solutions evolve through collaboration between self-organizing cross-functional teams that include direct engagement with business stakeholders.

Key aspects:

- Incremental delivery
- Always ready to ship
- Continuous inspection of work product and process provides feedback for continuous improvement
- Better business and tech alignment

The Agile Manifesto and its Origins

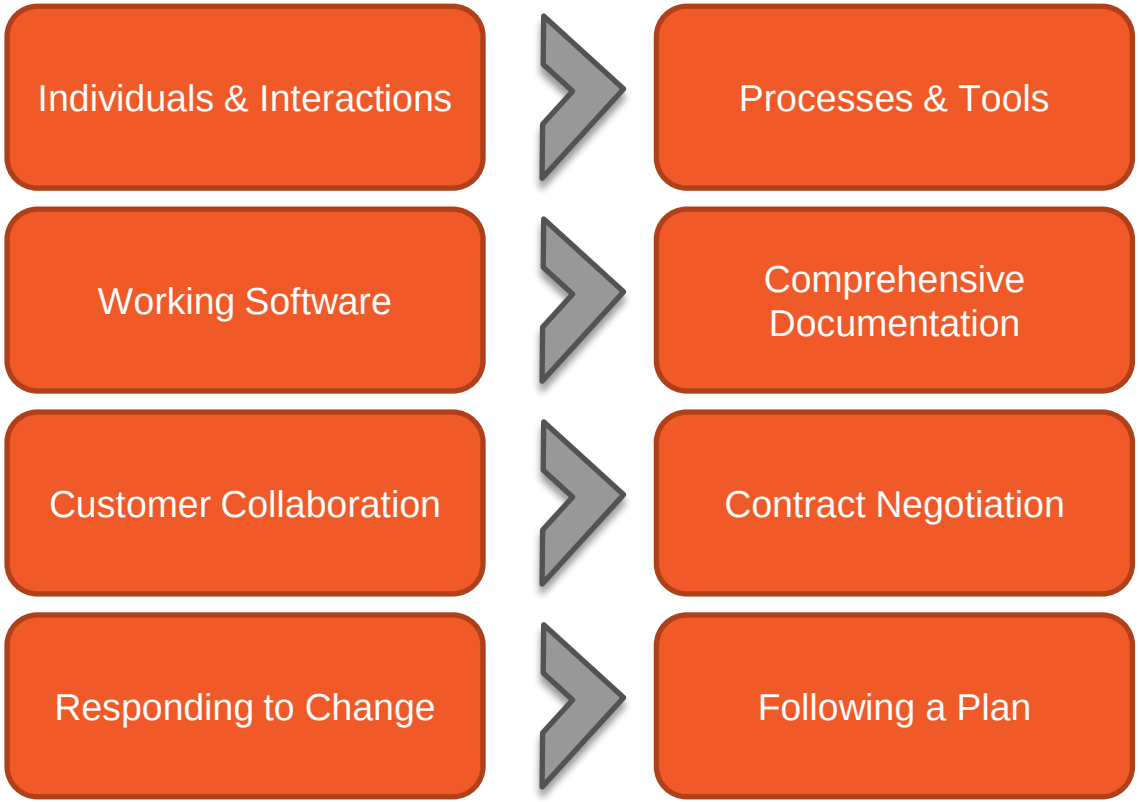
What is it? A statement of values regarding approaches to how teams create and ship software

When was it made? 2001, by 17 top software developers and consultants

Key aspects:

- 12 values were agreed upon
- These values are NOT a process, implementation or standard; they are opinions
- Many popular project management methodologies have arisen that are based on these values
- Many Agile implementations are poorly executed; it takes great cooperation between business leadership and a competent, self-guided dev team to make it work

Agile Values



Key Elements of the Agile Approach

The Agile method is an iterative and incremental tactic to software design that utilizes constant planning, understanding, upgrading, team partnership, development, and delivery.

It is driven by the principles of providing value and collaborating with stakeholders.

It starts with customers defining the end uses of the final product and the kind of problems the final product attempts to address.

Designated teams start to plan and work on a complete process through planning, implementing, and appraising.

Since the development process is iterative, errors are resolved in the intermediate stage of the project.

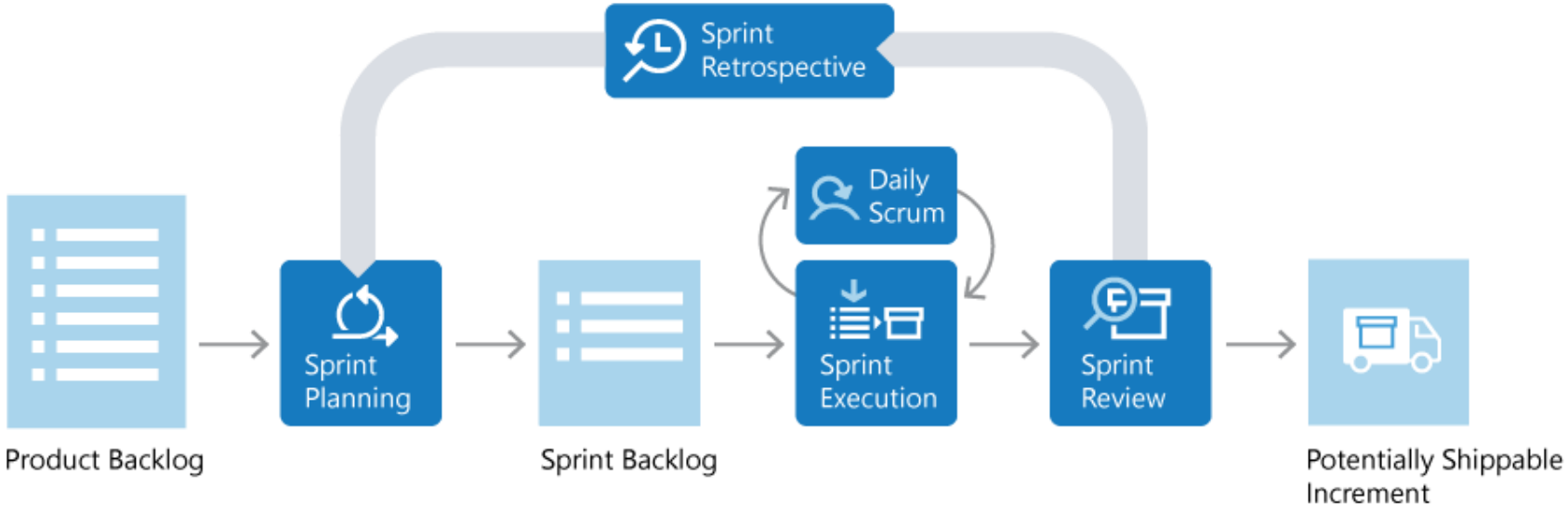
Modern Approaches to Agile Software Development

Several project management approaches implement Agile values in some way. Often, they bring in elements of other approaches.

Popular methodologies:

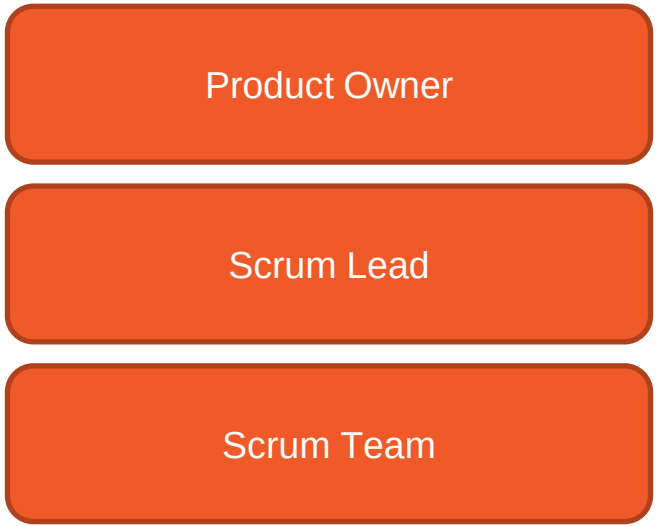
Kanban	Scrum	Scaled Agile Framework (SAFe)
• Uses visual boards to view & organize tasks • Uses elements of “just in time” lean manufacturing strategies • Emphasizes throughput	• Aligns closely with Agile values • Breaks down product development into iterative sprints • Makes use of exclusive roles (“Scrum Lead”, “Product Owner”) • Emphasizes constant communication	• Workflow and organizational patterns to help deploy Agile at scale • Can support large organizations

Diagram of Typical Scrum Sprint

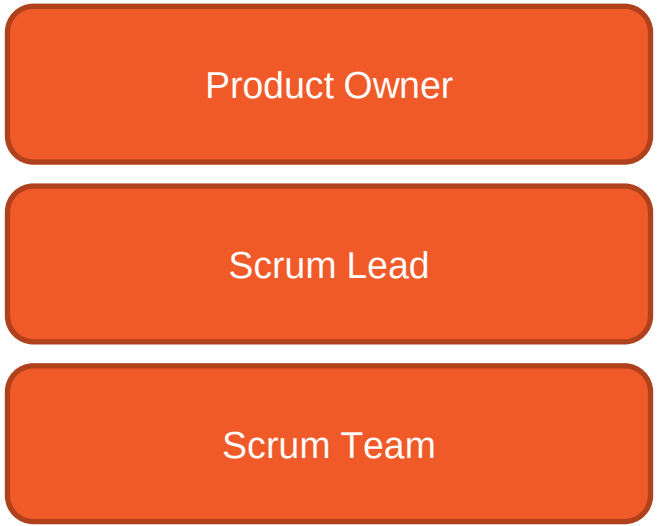


Source: https://docs.microsoft.com/en-us/devops/plan/what-is-scrum

Scrum Roles



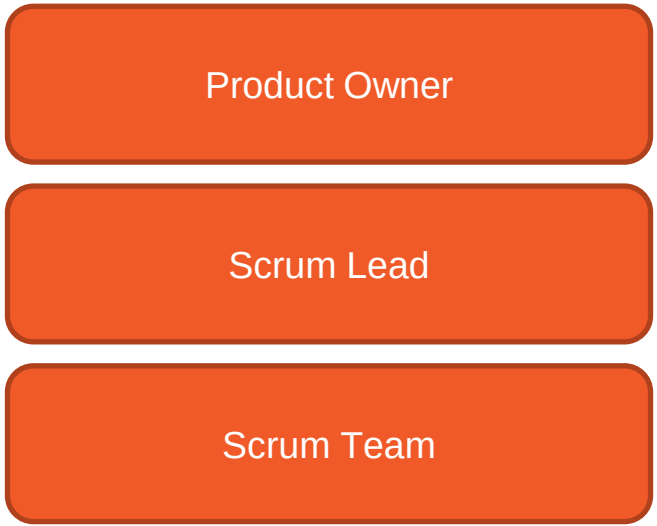
Scrum Roles



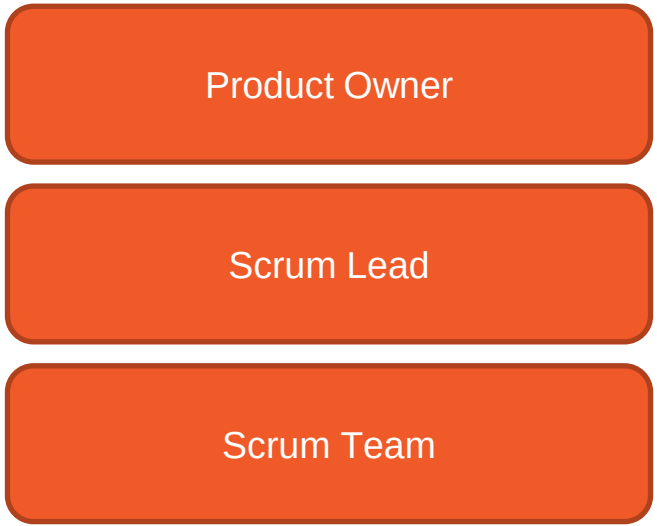
- Responsible for what team builds and why
- Keeps backlog up to date and in correct priority order

Scrum Roles

- Ensures that Scrum process is followed by team
- Responsible for the fidelity of the process
- Part coach, part team member, part cheerleader



Scrum Roles



- People building the product
- Responsible for product build (and associated quality)

Planning with Security in Mind

The Flat Backlog

Observation 1: Do You Know What You‘re Building?

Arranging user stories in the order you build them doesn‘t help when you want to answer the question

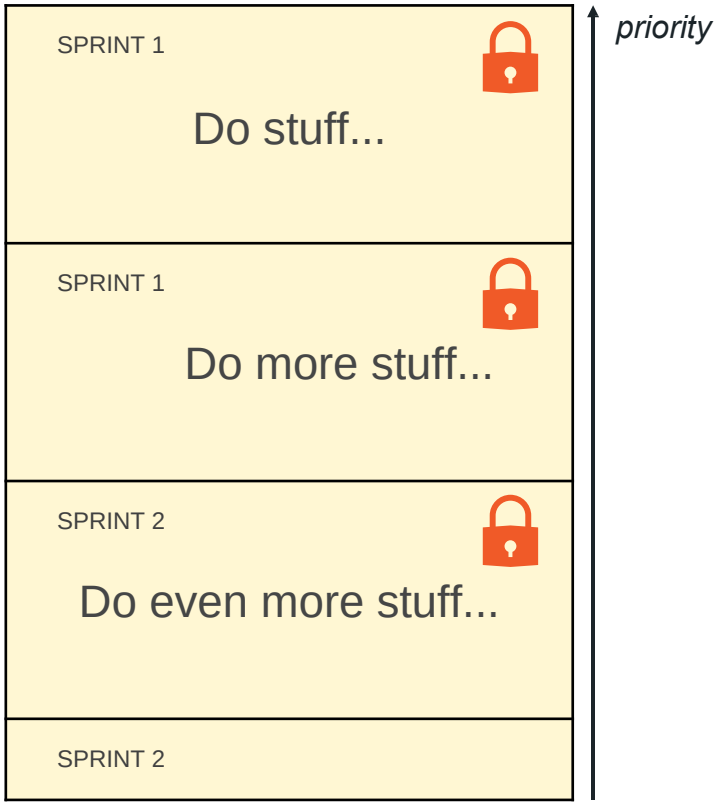
"What does the system you‘re building do?" to others.

Observation 2: Know How Things Relate?

You can‘t see how everything fits together.
Making decisions on what to build next is difficult.
Sure you haven‘t forgotten about important features?

Observation 3: Know You Build The Right Things?

You can‘t see how your users experience the product.
Planning coherent, value-driven releases is difficult.



The Backlog

priority

SPRINT 1

Do stuff...

SPRINT 1

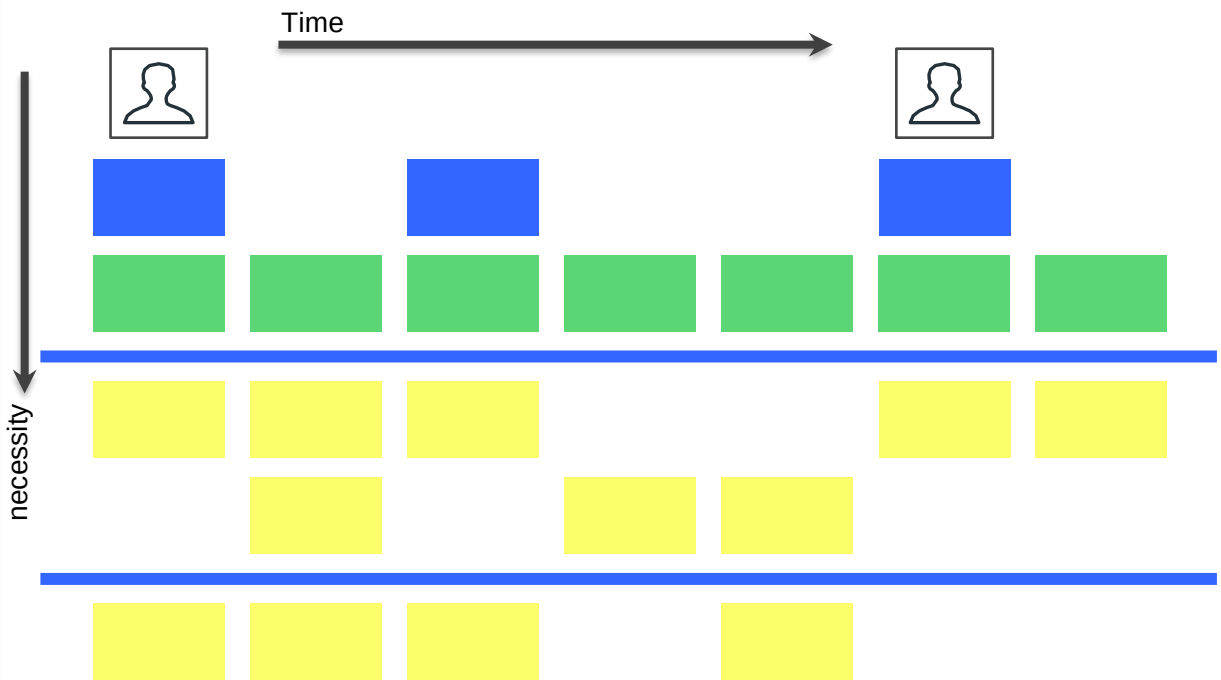
Do more stuff...

SPRINT 2

Do even more stuff...

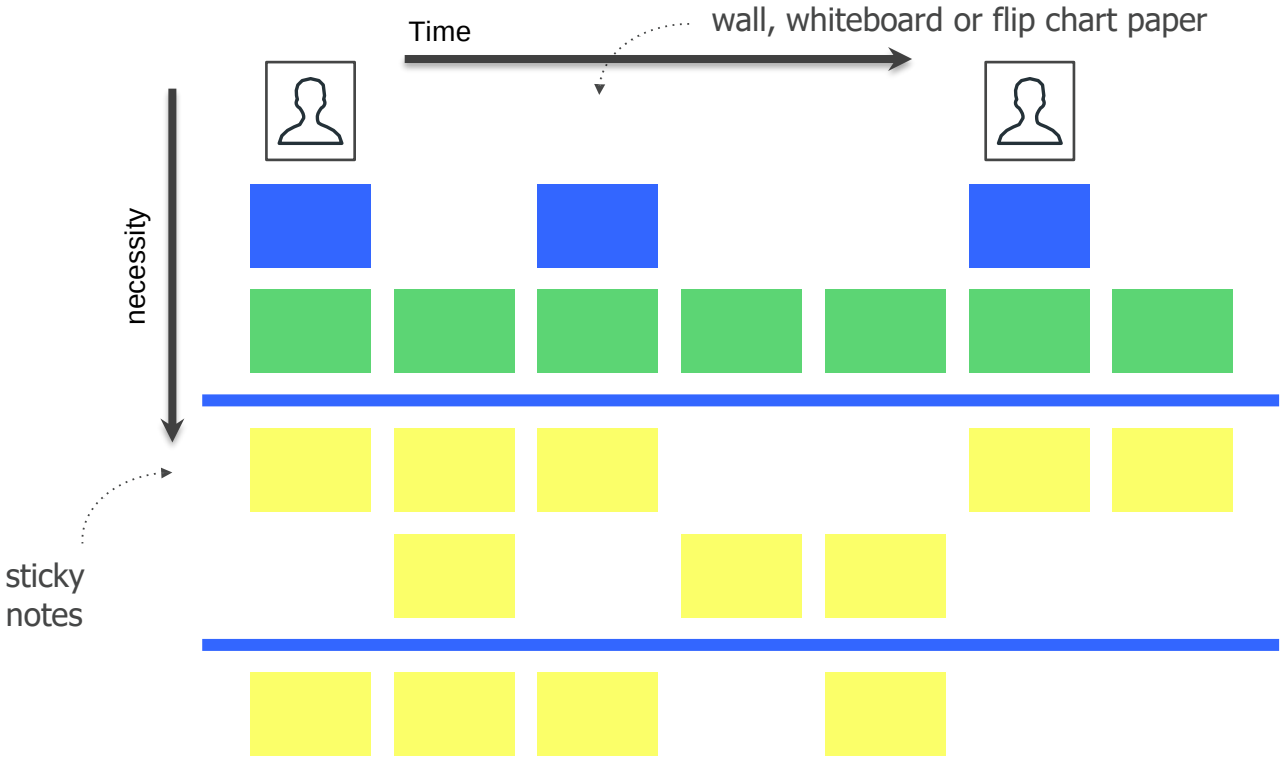
SPRINT 2

The User Story Map



User Story Map – Prerequisites

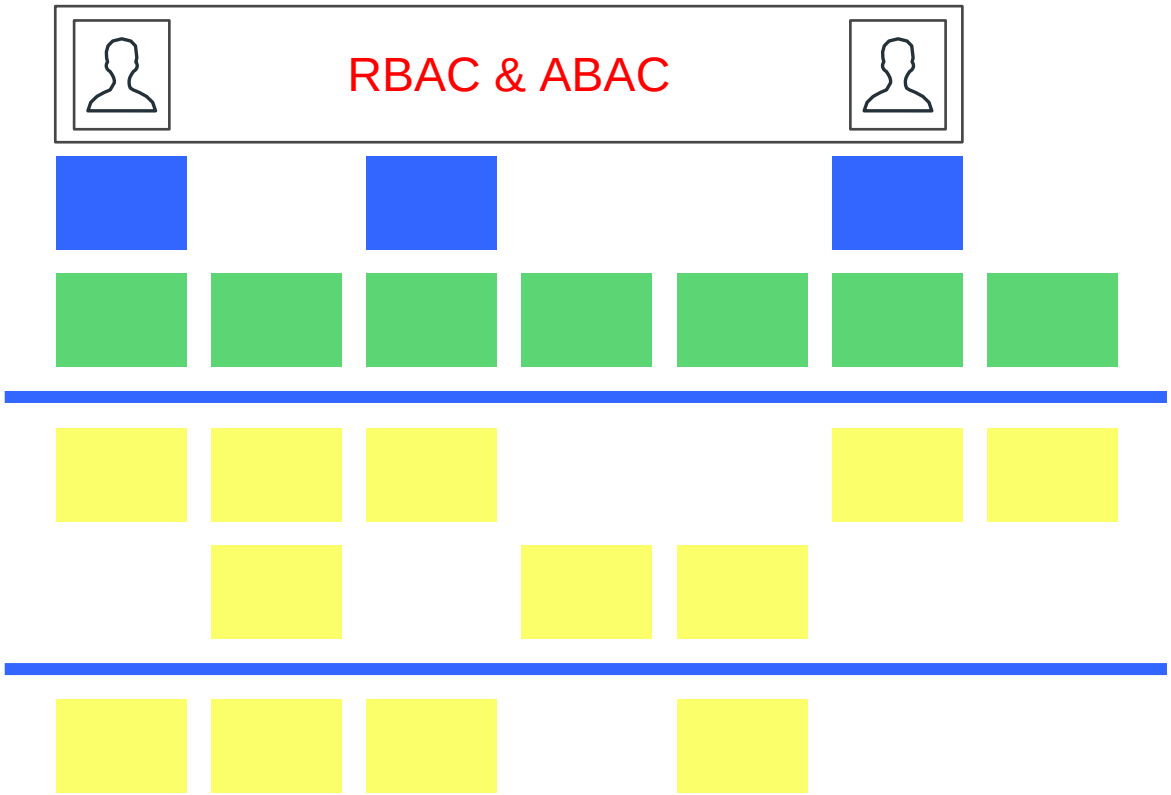
- Customer/Stakeholders
- Product Owner
- SMEs
- Architects
- Agile Team
- Scrum Master
- Big Wall, whiteboard, flip chart papers
- Sticky Notes
- Sharpies
- Time & Patience
- Security Resources
- Threat Model



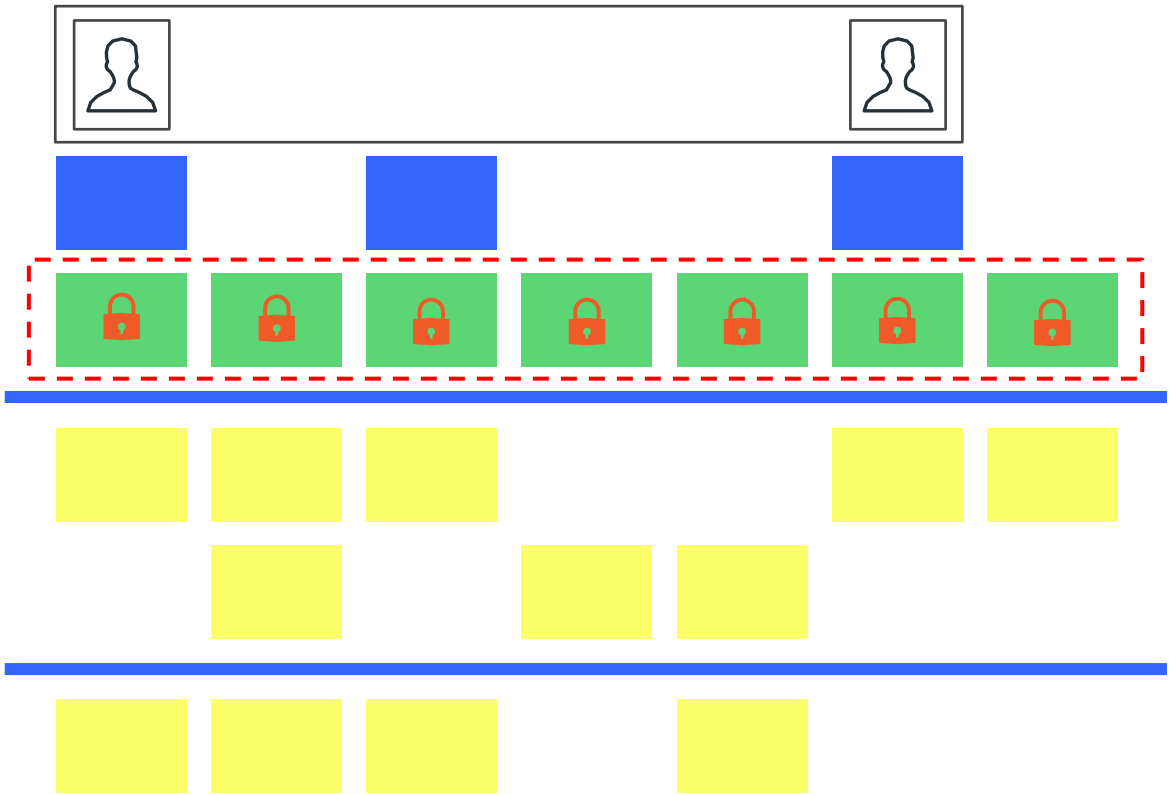
Anatomy Of A Map

Users / Personas

Who are your users and what are their goals?



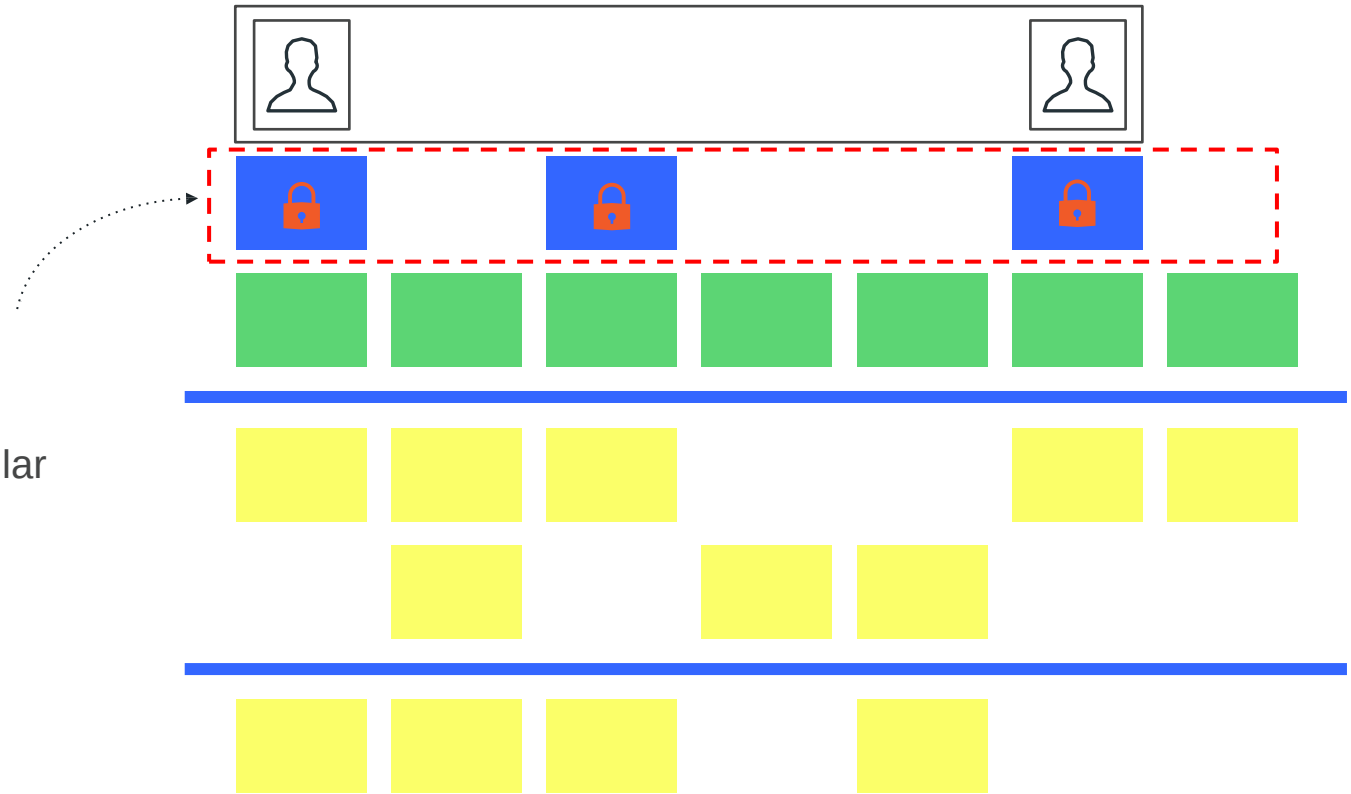
Anatomy Of A Map



Go this way First, i.e., breadth

User Tasks ("Walking Skeleton“) Things a user does to achieve a goal. Starts with a verb, e.g., *"Send Email“*.

Anatomy Of A Map



User Activities ("Backbone“)

Groupings of similar tasks.

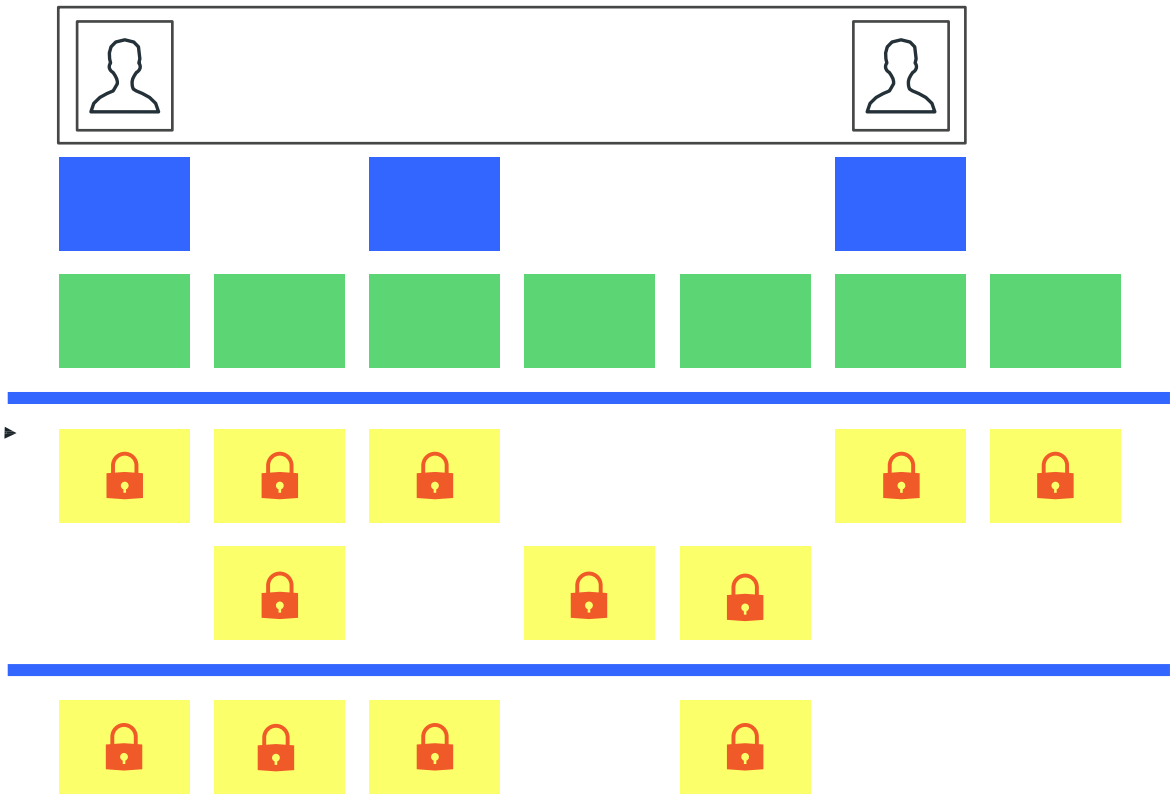
Anatomy Of A Map

User Stories

Flesh out the user journey in detail:

Sub-tasks, alternatives, exceptions, etc.

Then this way, i.e., depth



Release(s) from Story Map

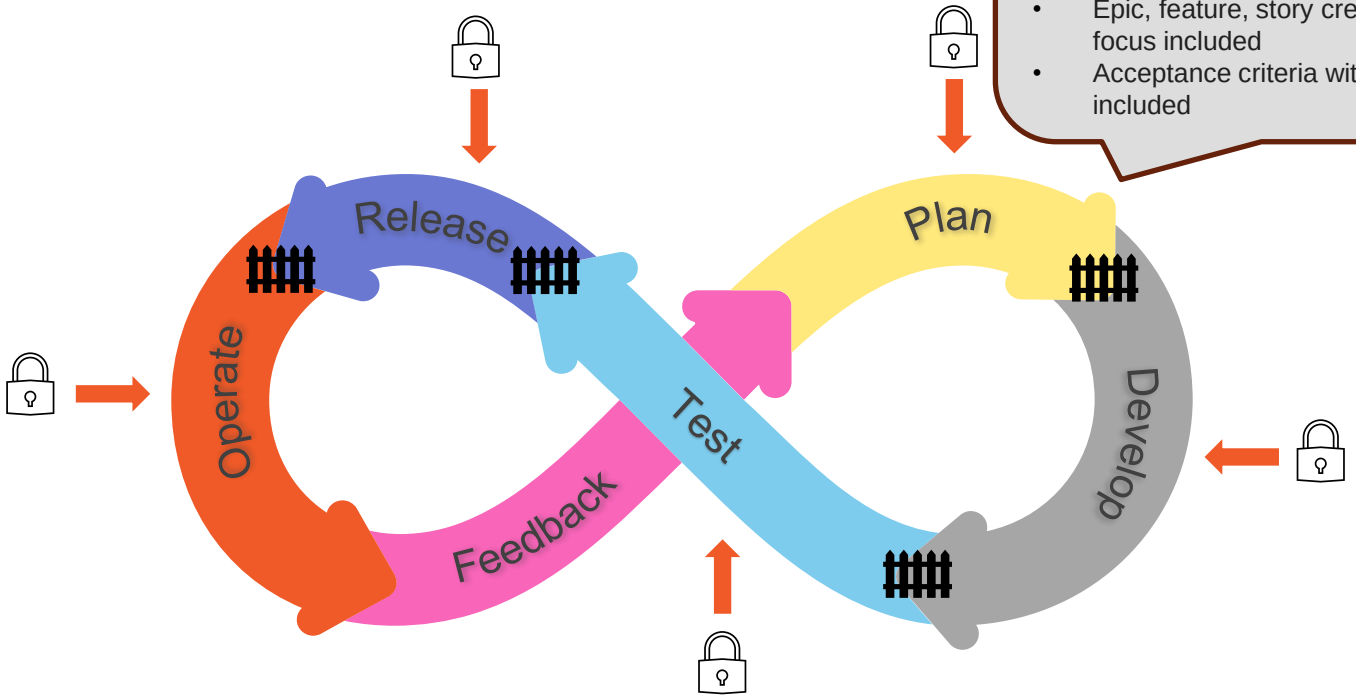
Release Slice

Identifies the smallest number of user stories for each task.

Achieves your user‘s goals.

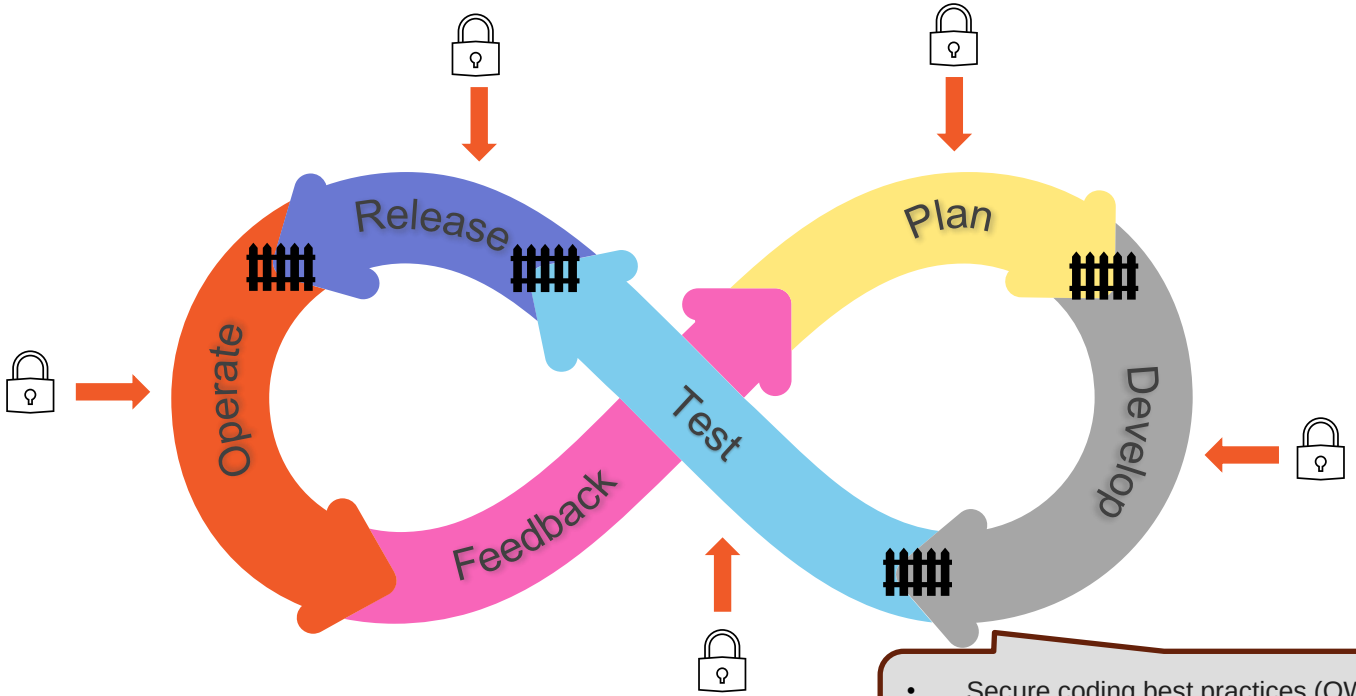
Sprint Planning with Security in Mind

- Threat modeling
- RBAC & ABAC requirements
- Secure design principles (OWASP Top 10)
- Data classification & risk assessment
- Agile design reviews with security focus included
- Epic, feature, story creation with security focus included
- Acceptance criteria with security focus included



Quality gates guard against moving security defects forward

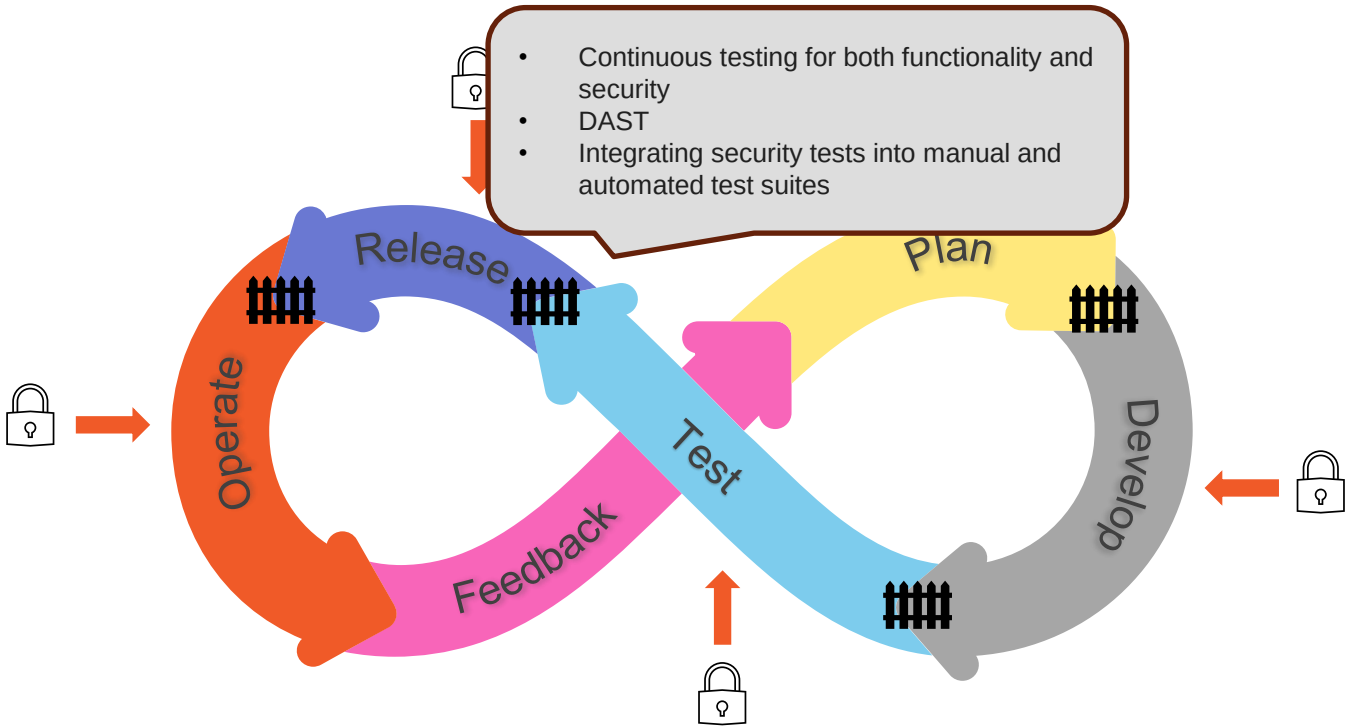
Sprint Planning with Security in Mind



- Secure coding best practices (OWASP Top 10)
- Static code analysis (SCA & SAST)
- Security-focused unit tests

Quality gates guard against moving security defects forward

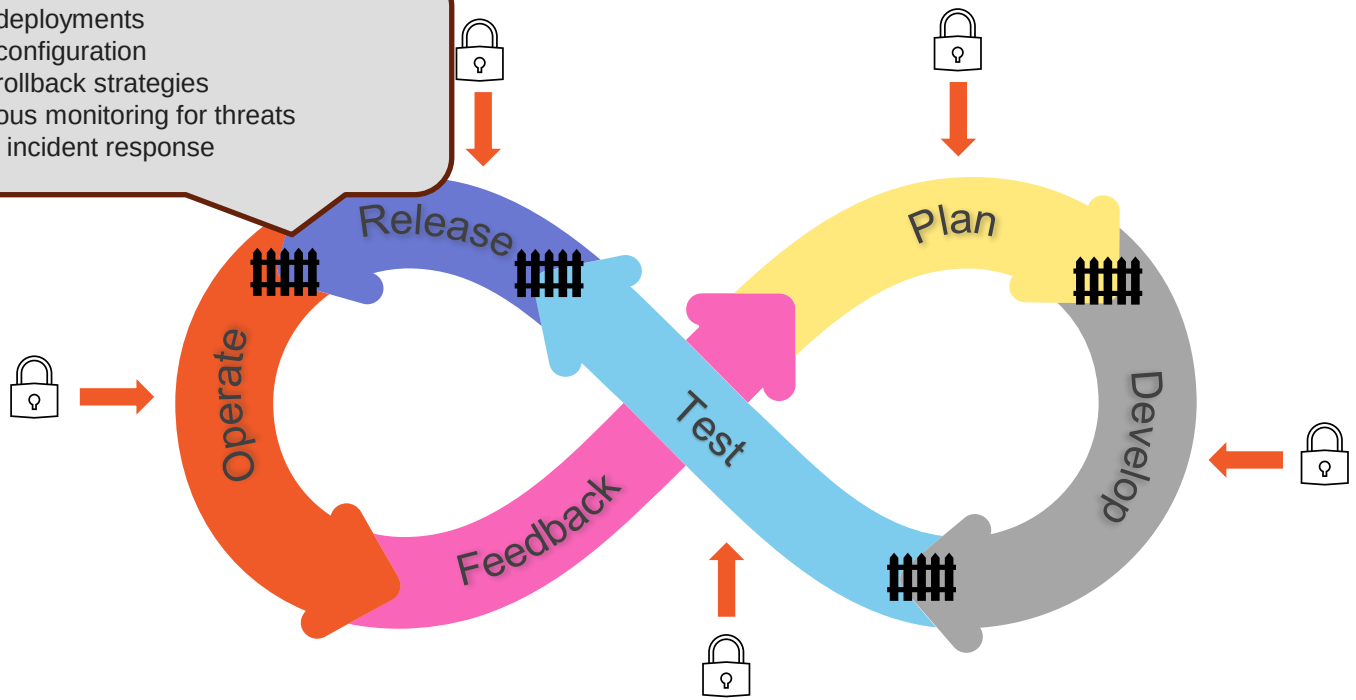
Sprint Planning with Security in Mind



Quality gates guard against moving security defects forward

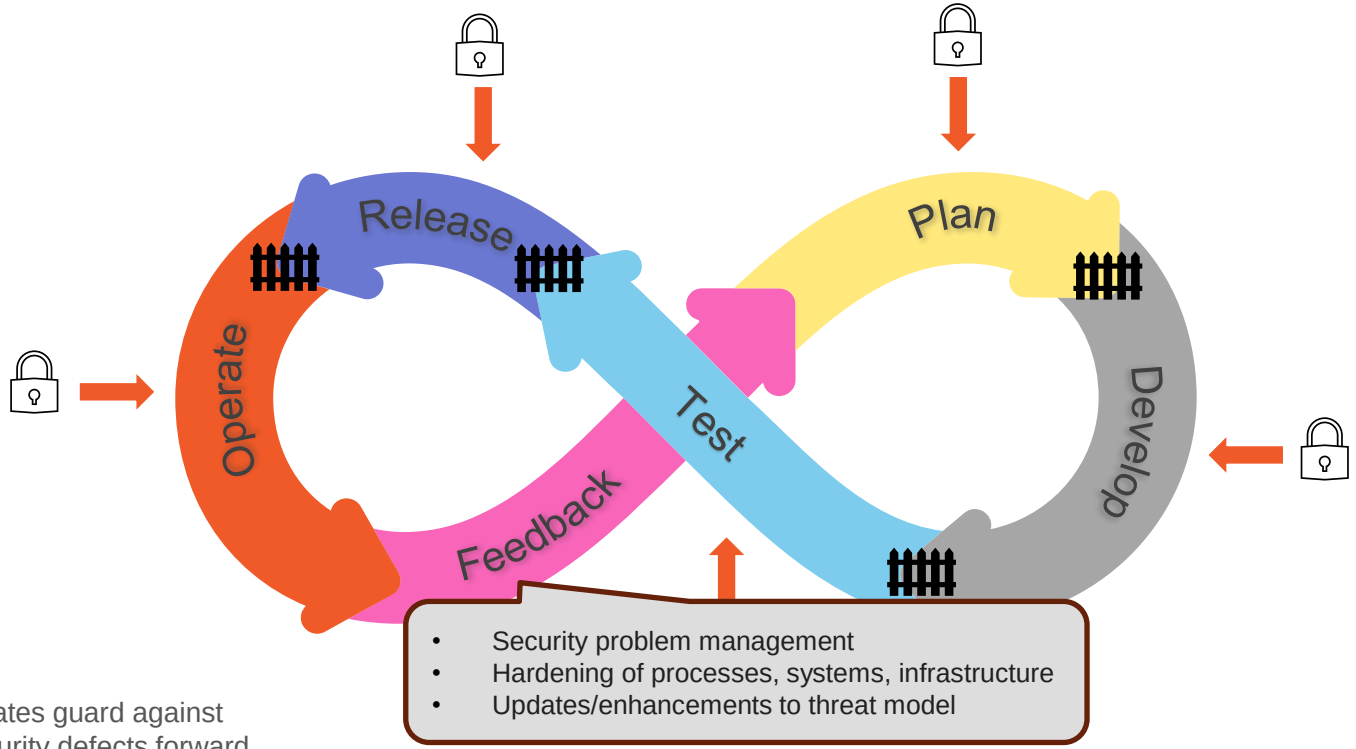
Sprint Planning with Security in Mind

- Secure deployments
- Secure configuration
- Secure rollback strategies
- Continuous monitoring for threats
- Security incident response



Quality gates guard against moving security defects forward

Sprint Planning with Security in Mind



Quality gates guard against moving security defects forward

LAB:

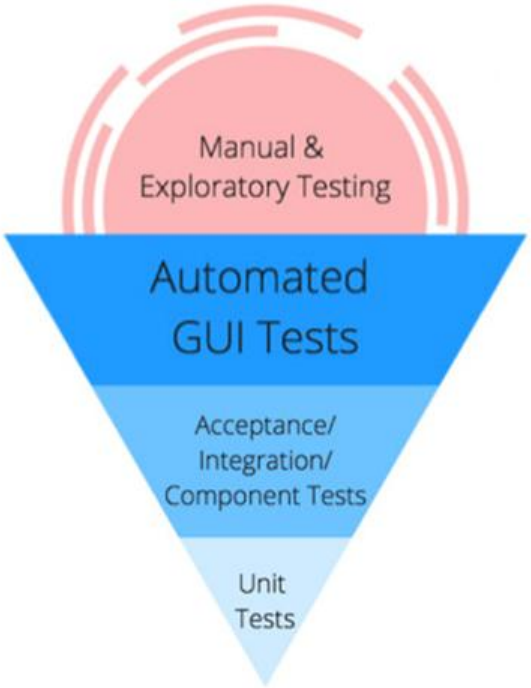
Sprint Planning with Security in Mind

Using the threat model, a list of users/personas, and the high-level diagram you created for your scenario, identify a feature you would like to build in a single sprint. For that sprint, create a set of high-level user stories with acceptance criteria and story points. For each story, discuss the following:

- Have you accounted for security in the definition and acceptance criteria?
- How best to describe each to keep the story detail accessible & understandable to multiple roles (engineers, business, QA)?
- Are the assigned points sufficient to account for security concerns?

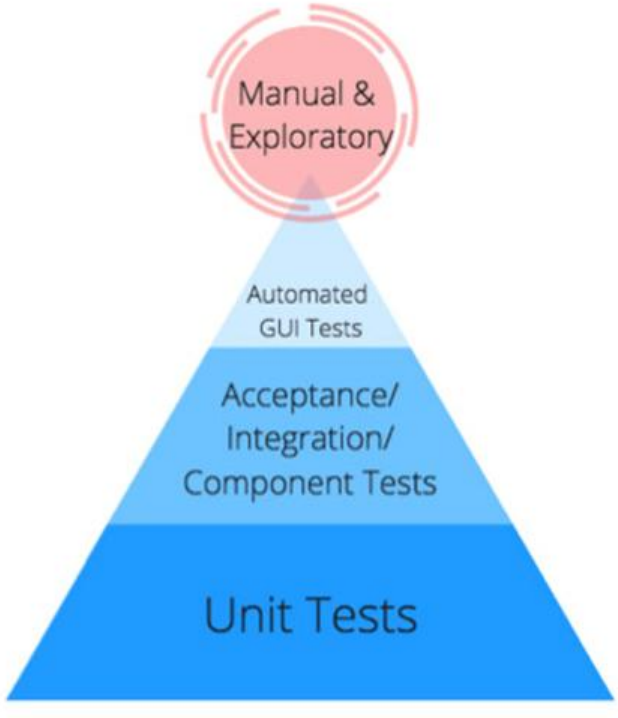
Security Focused Testing

Testing Pyramids

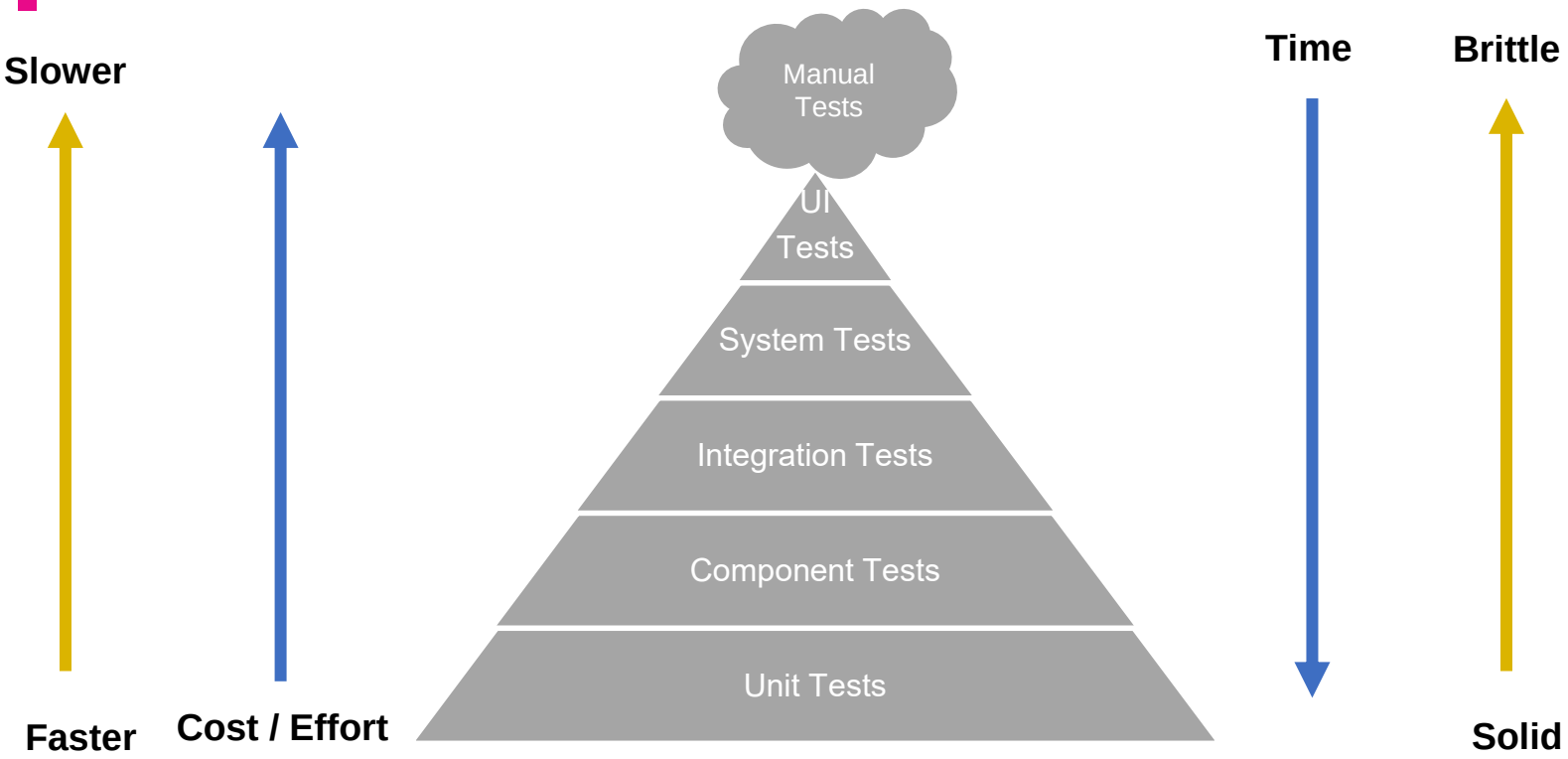


More Time & Effort

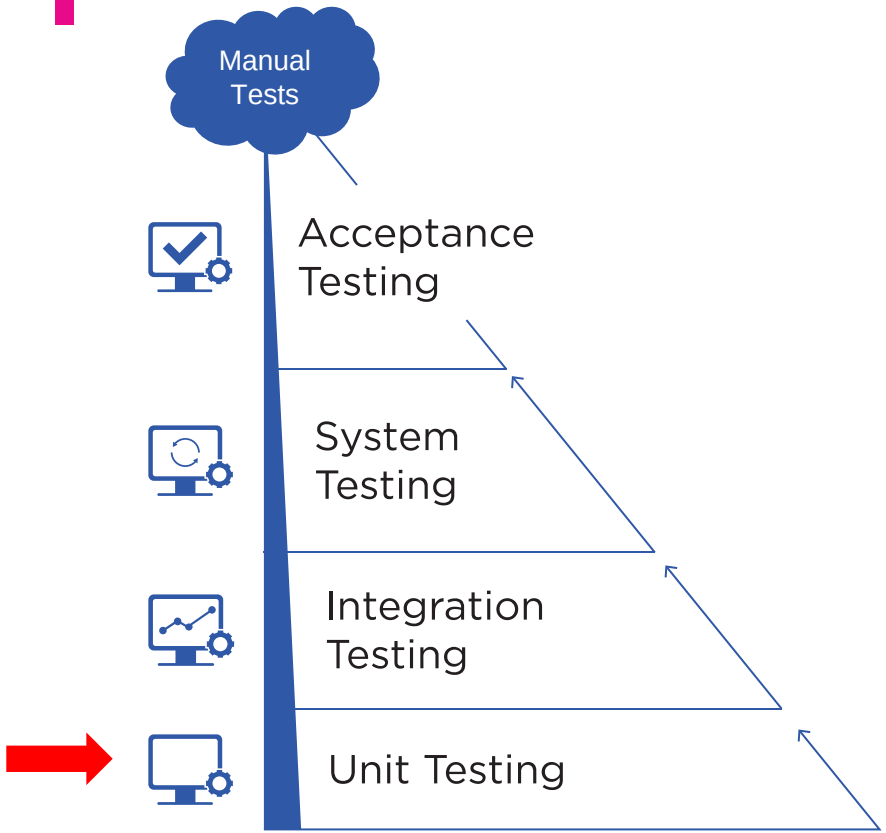
Higher ROI



Agile Testing Pyramid

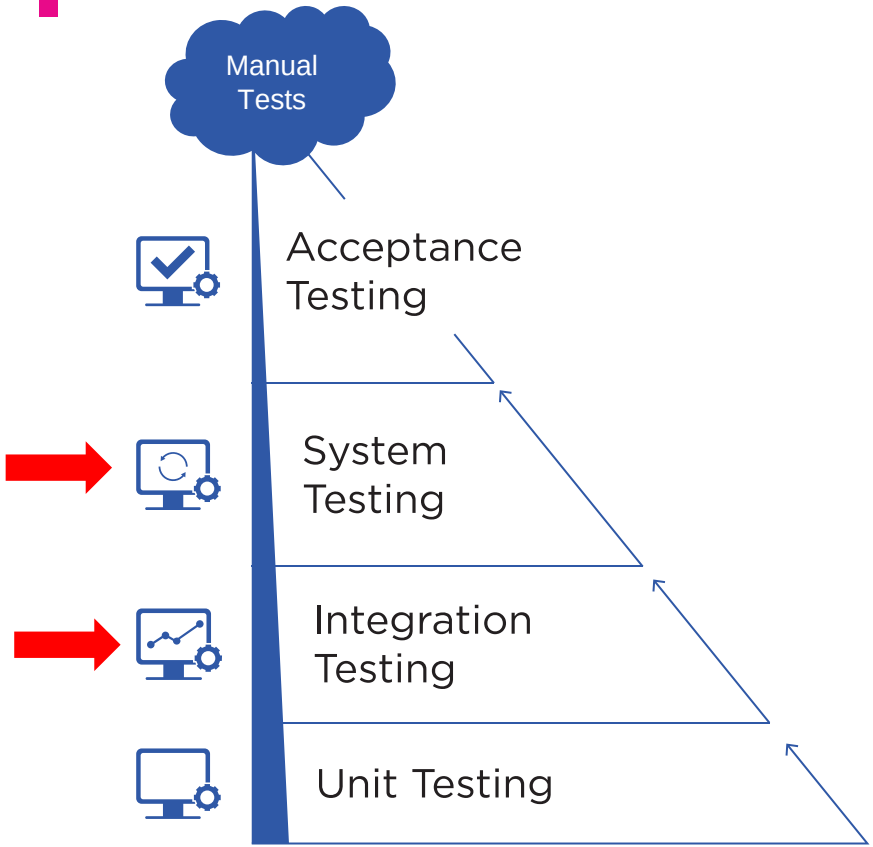


Unit Testing



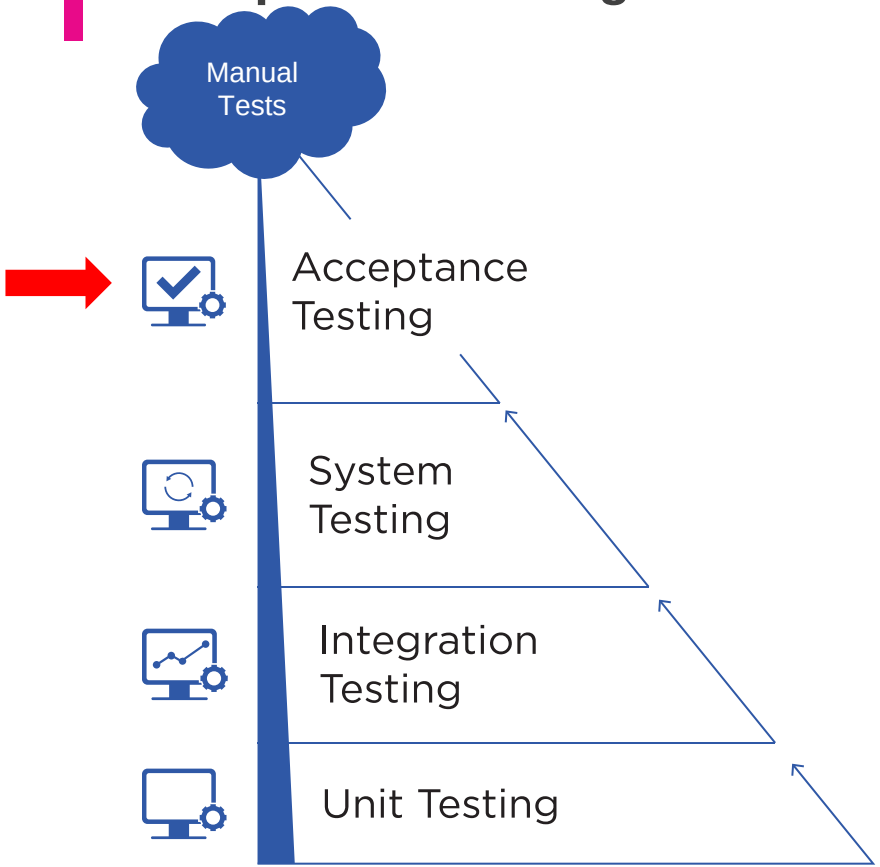
- Usually written by developers
- Uses Assertions
- Testing smallest part of the system in isolation
- Best Performance – execute early, often and within few seconds
- Smallest scope - easier to locate and understand errors
- Frequency – Every code check-in or pull request
- Use of test doubles to replace dependencies – dummies, stubs, spikes, mocks
- Increases confidence in changing/maintaining code

Middle Layer Testing



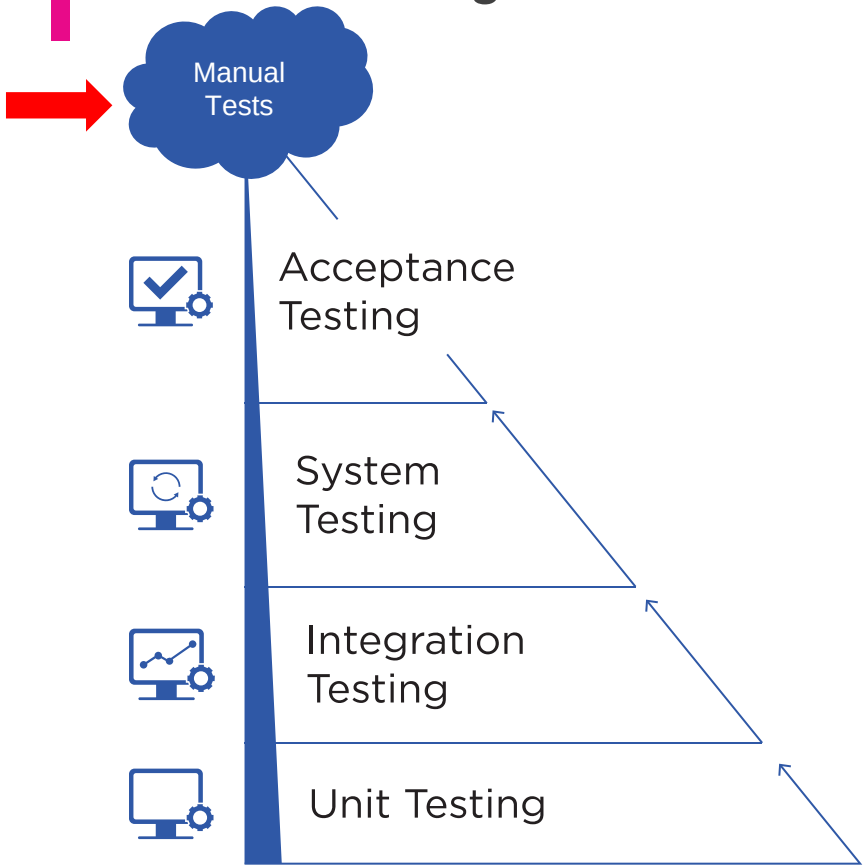
- Together called service/API layer tests, also termed as “middleware layer”.
- Component Tests: Look at individual components. Validates the functionality is working as expected with other components.
- Integration Tests: Targets modules/features that integrate directly with other dependencies outside the application. Can be used as gating/staging from preprod to prod, etc.
- System Integration Tests: Large scale integration suite testing end-to-end workflows. Often coordinated across teams and unique to your application or system.

Acceptance Testing



- Similar to integration tests in the sense that they test different parts of the application
- End-to-end testing
- Ensure high-level functionality works as expected or described and delivers business value
- Maximum scope such as - logics, UI workflows, navigation, transitions, calculations, buttons, layouts etc.
- Can be brittle or flaky and a lot of work to maintain
- Test critical workflows

Manual Testing



- Not regression testing
- Experience + creativity
- Learn about the system, discover defects & improve automated testing
- Can be based on missions/test charter/persona
- Covers scenarios which can’t be automated or are too complex to automate
- Edge Scenarios for mission critical applications to prevent failure

Thank you!

If you have additional questions,
please reach out to me at:
asanders@gamuttechnologysvcs.com

PLURALSIGHT