Testing with Agile Strategies

Michael Gildein

Agenda

- Agile Refresher
- Agile Testing
- Devops

Agile Refresher

Agile Model

Iterate
- Iterate often and in small chunks
- Iterations last 2-4 weeks

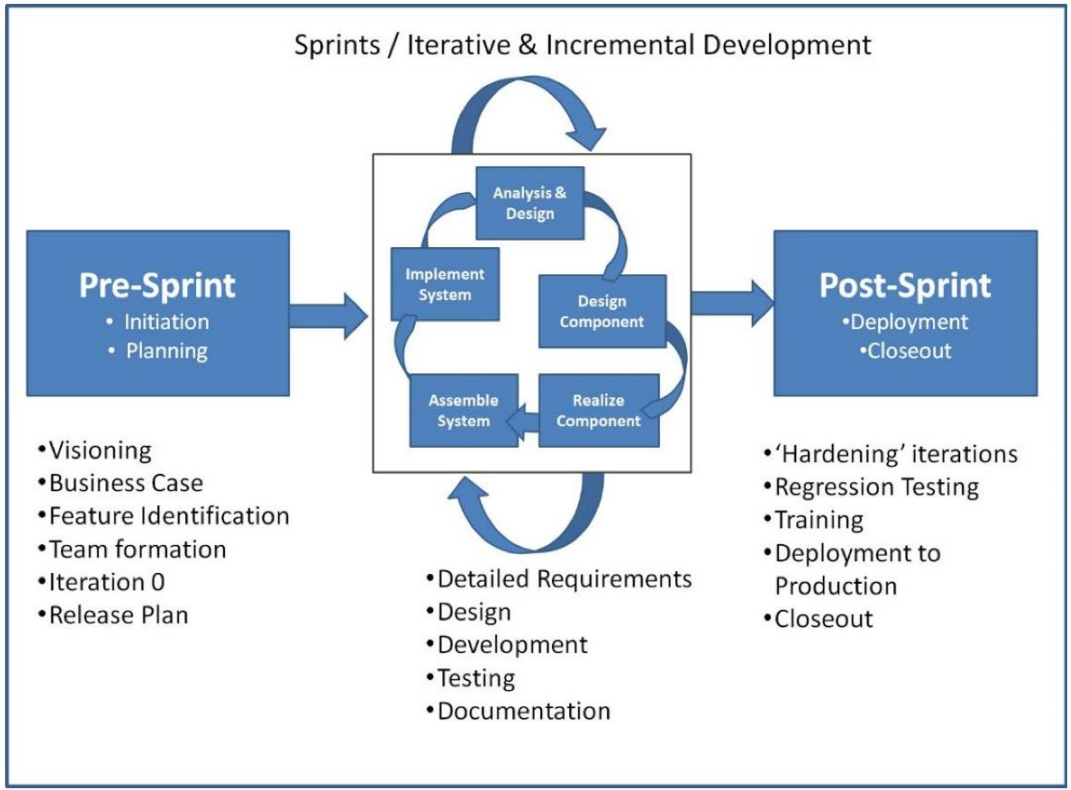
Teams
- Focused cross-functional teams
  - 2 pizza box sized (4-10)
- Dev and test together

Value
- Collaboration w/ customer
- Value working code over doc
- Adaptation over planning
- Automation is key

Pros
- Short time to market
- Constant feedback
- Good for small teams
- Projects with change
- Less doc

Cons
- Increased test efforts
- Need seasoned dev
- Focused teams

Agile Cycle

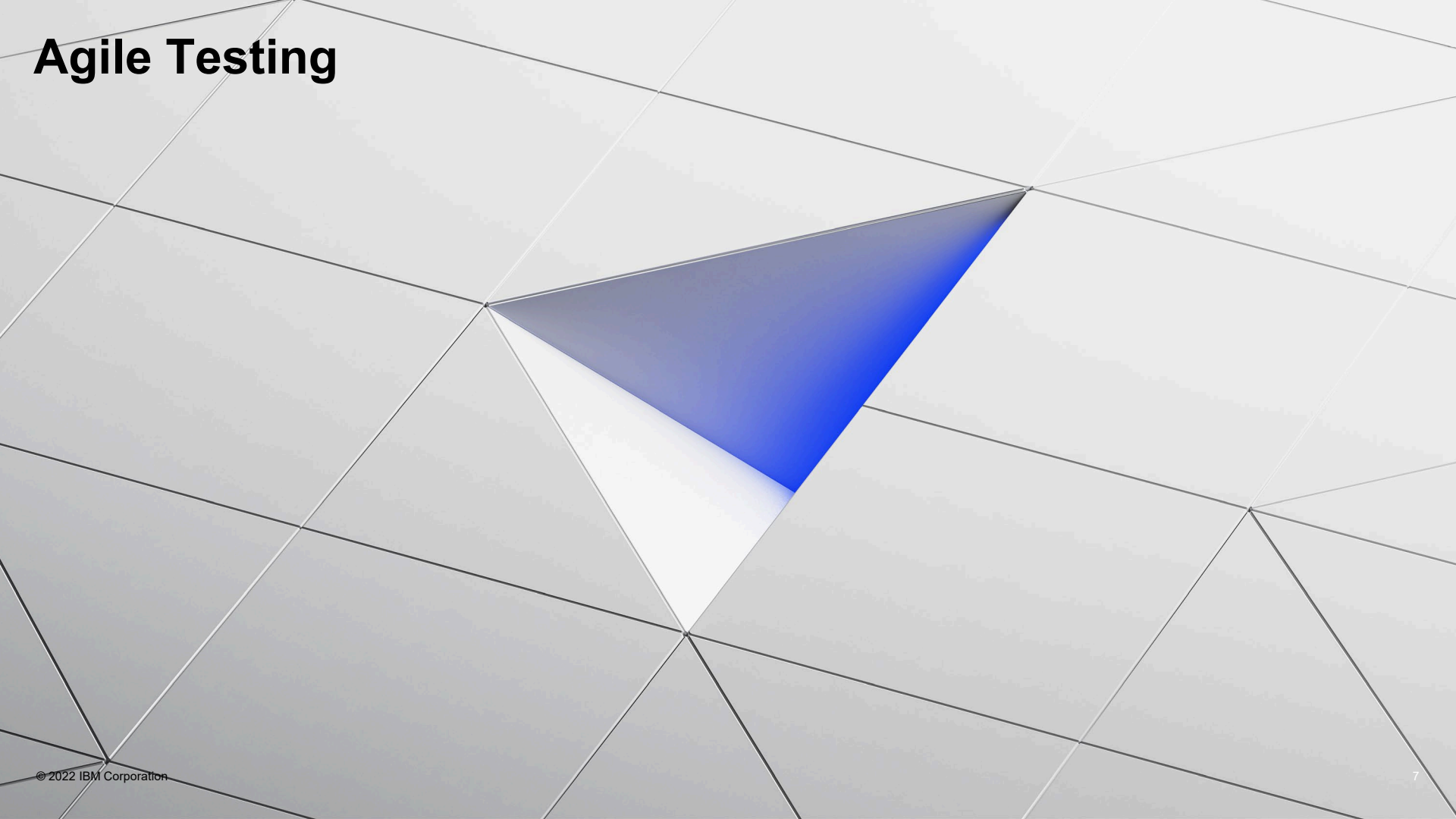

Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
Analysis & Design
Design Component
Realize Component
Assemble System
Implement System
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
Post-Sprint
•Deployment
•Closeout
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Agile Model

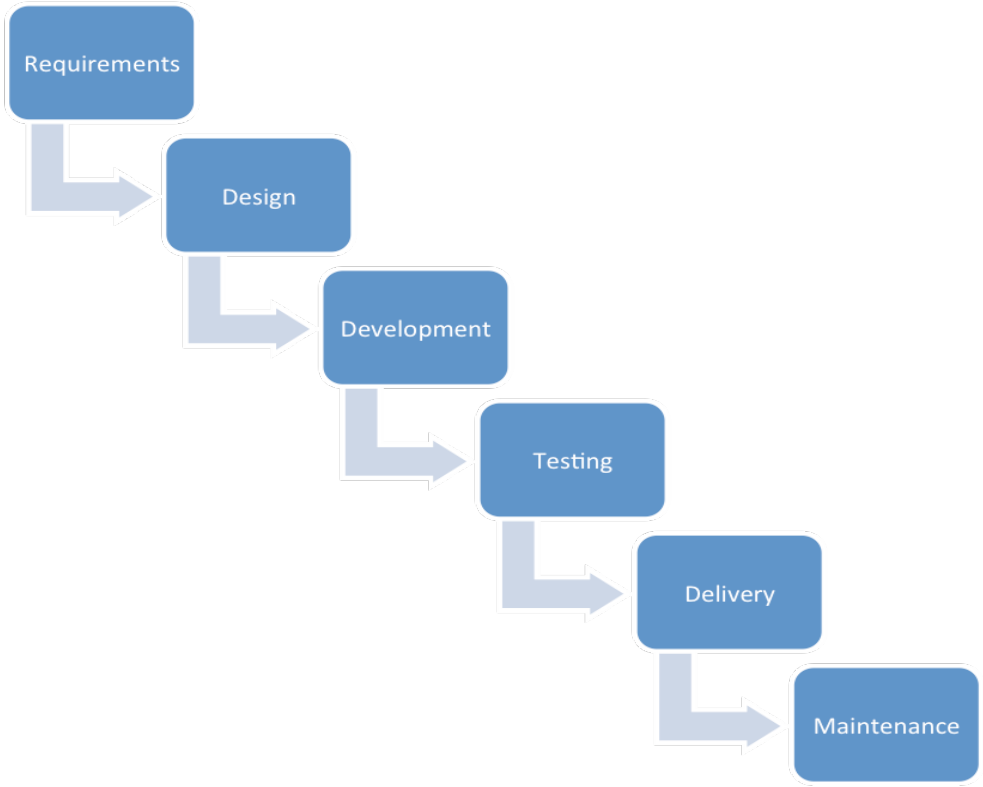
Agile Manifesto Principles:

1. **Customer satisfaction** by early and continuous delivery of valuable software
2. Welcome **changing requirements**, even in late development
3. Working software is **delivered frequently** (weeks rather than months)
4. **Close, daily cooperation** between business people and developers
5. Projects are built around **motivated individuals**, who should be trusted
6. **Face-to-face** conversation is the best form of communication (co-location)
7. **Working software** is the principal measure of progress
8. **Sustainable development**, able to maintain a constant pace
9. Continuous attention to technical excellence and **good design**
10. **Simplicity**—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from **self-organizing teams**
12. Regularly, the team **reflects** on how to become more effective, and adjusts accordingly

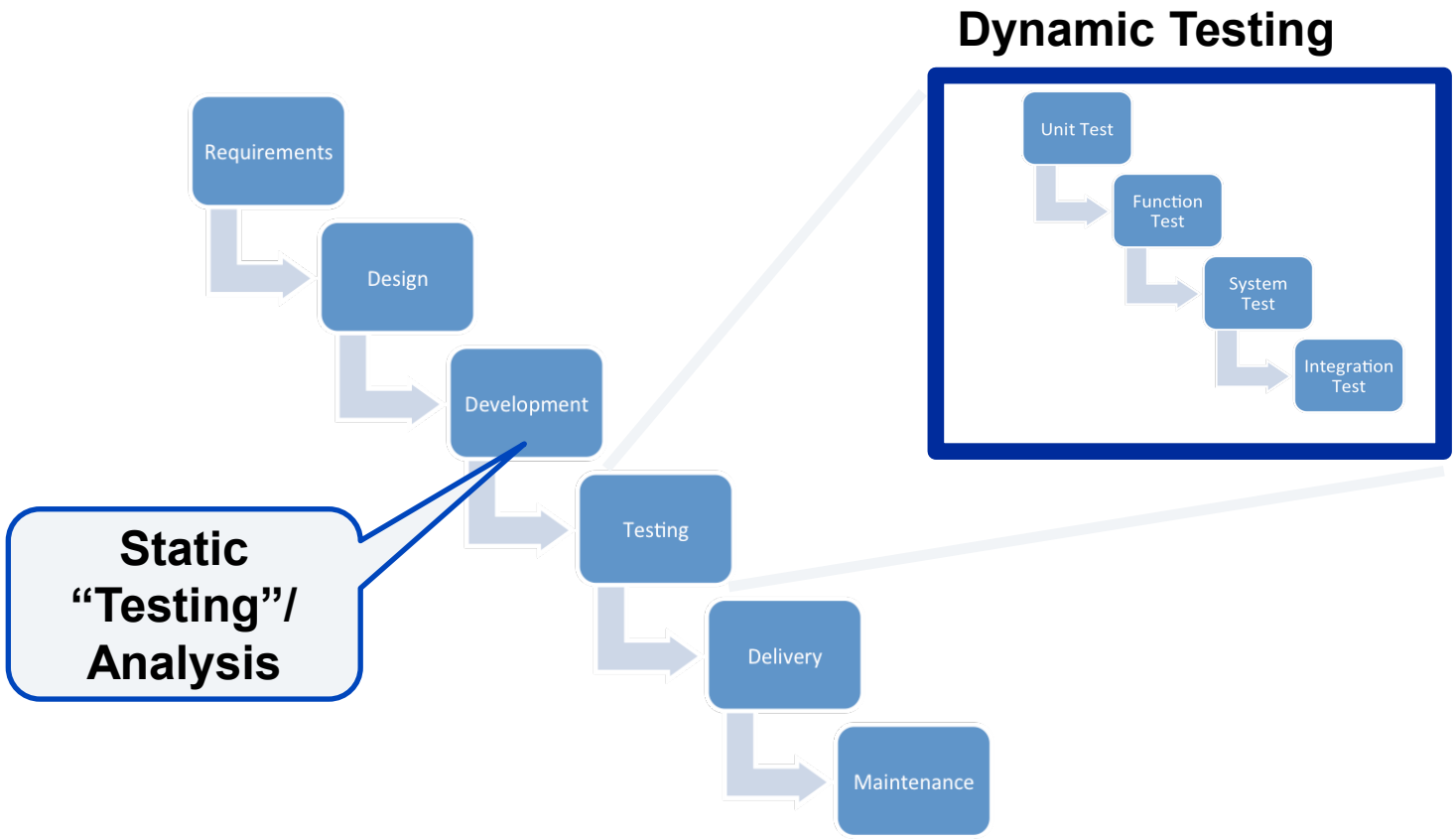
Agile Testing



Traditional Software Development



Traditional Software Development



Static vs Dynamic Technique

Static

- Program/System not executed
- Inexpensive

Examples

- Code inspection/reviews
- Document review
- Intellectual Property (IP) scans
- Complexity analysis
- Security scans
- Coding standards & patterns

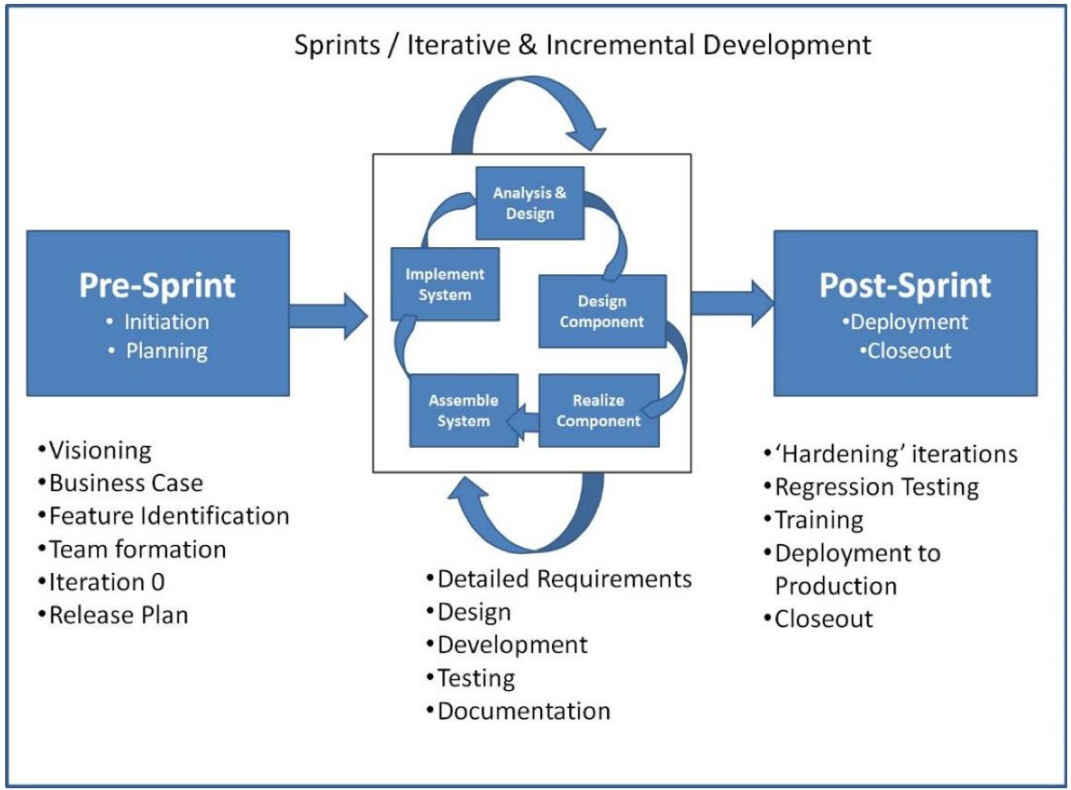
Dynamic

- Program/system is executed
- Expensive

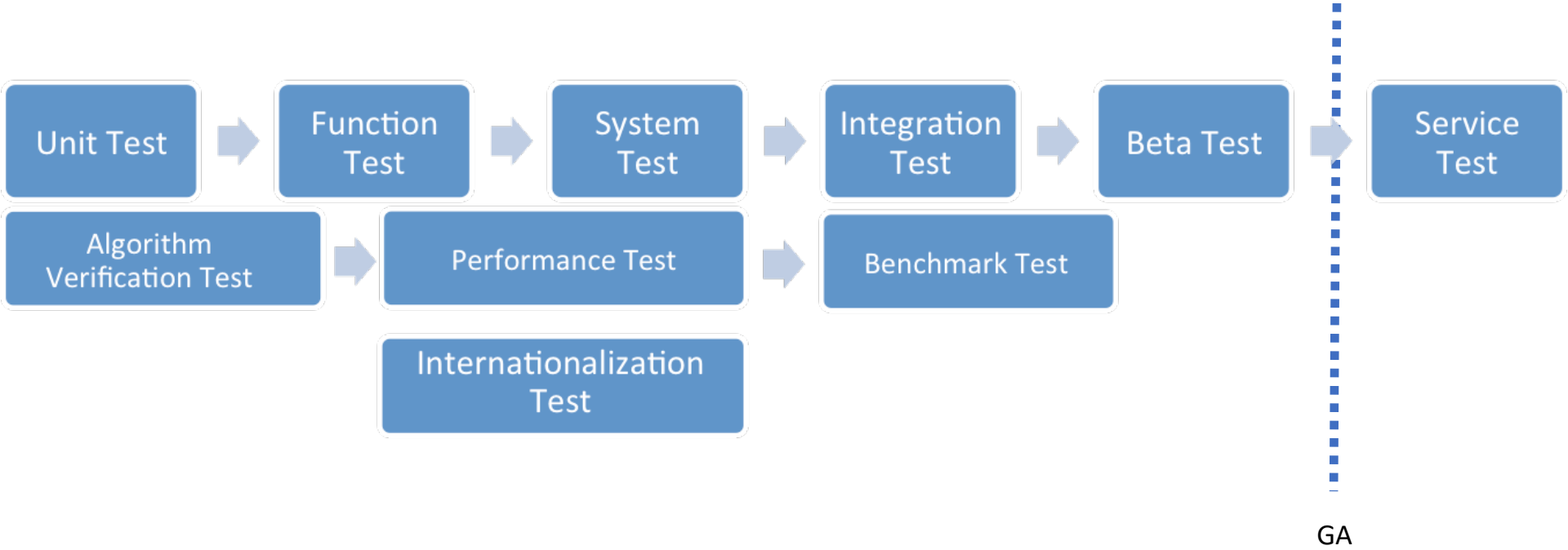
Examples

- Classic testing
  - Regression
  - Load/Stress
  - Phases

Agile Cycle

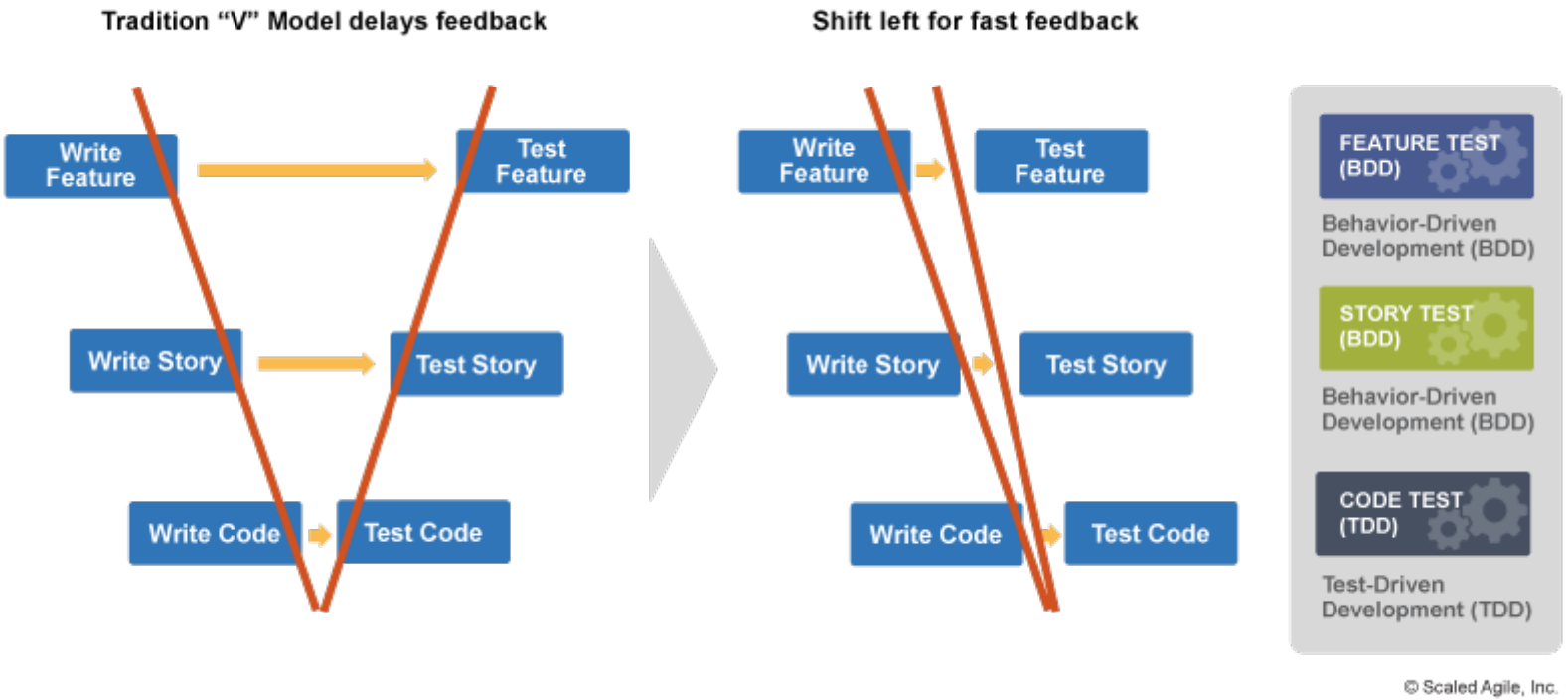

Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
Analysis & Design
Design Component
Realize Component
Assemble System
Implement System
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
Post-Sprint
•Deployment
•Closeout
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Traditional Enterprise Software Test Phases



EVERYONE is responsible for quality!

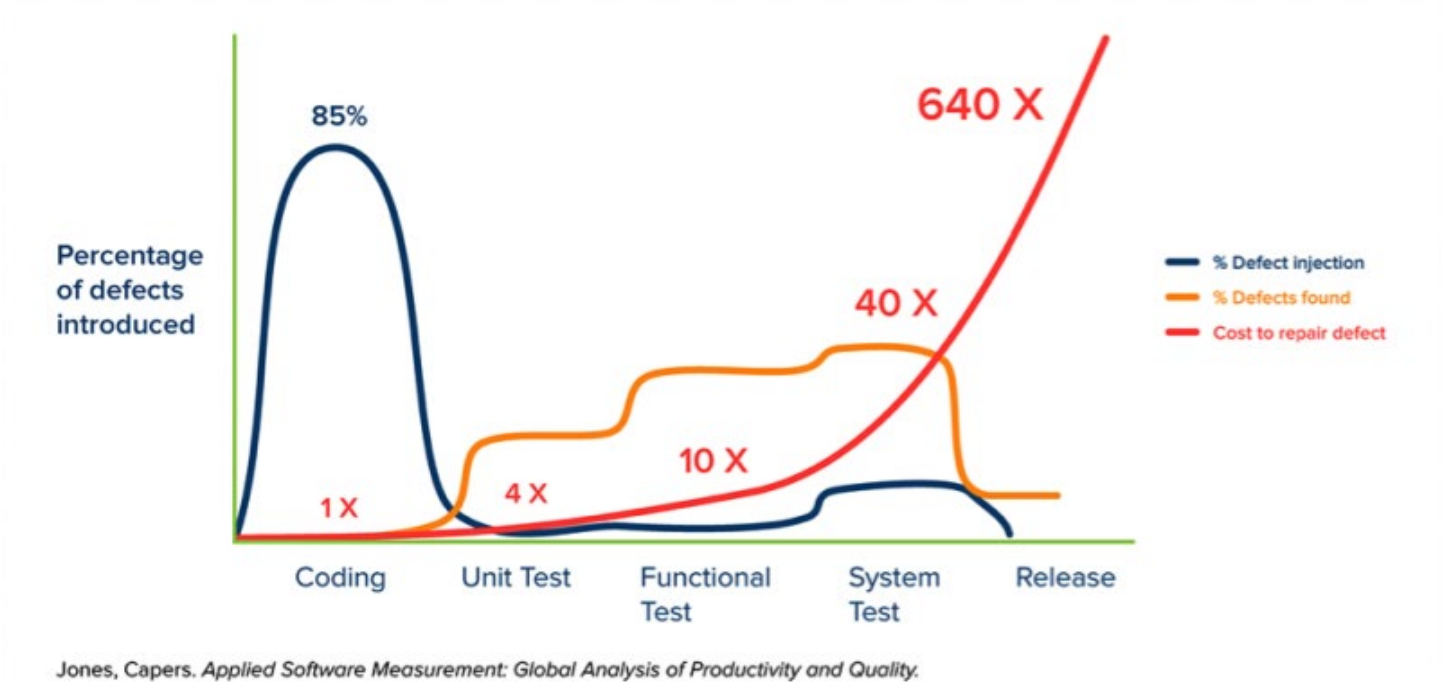
Always try to improve turnaround time



https://www.scaledagileframework.com/built-In-quality/

Shifting Left

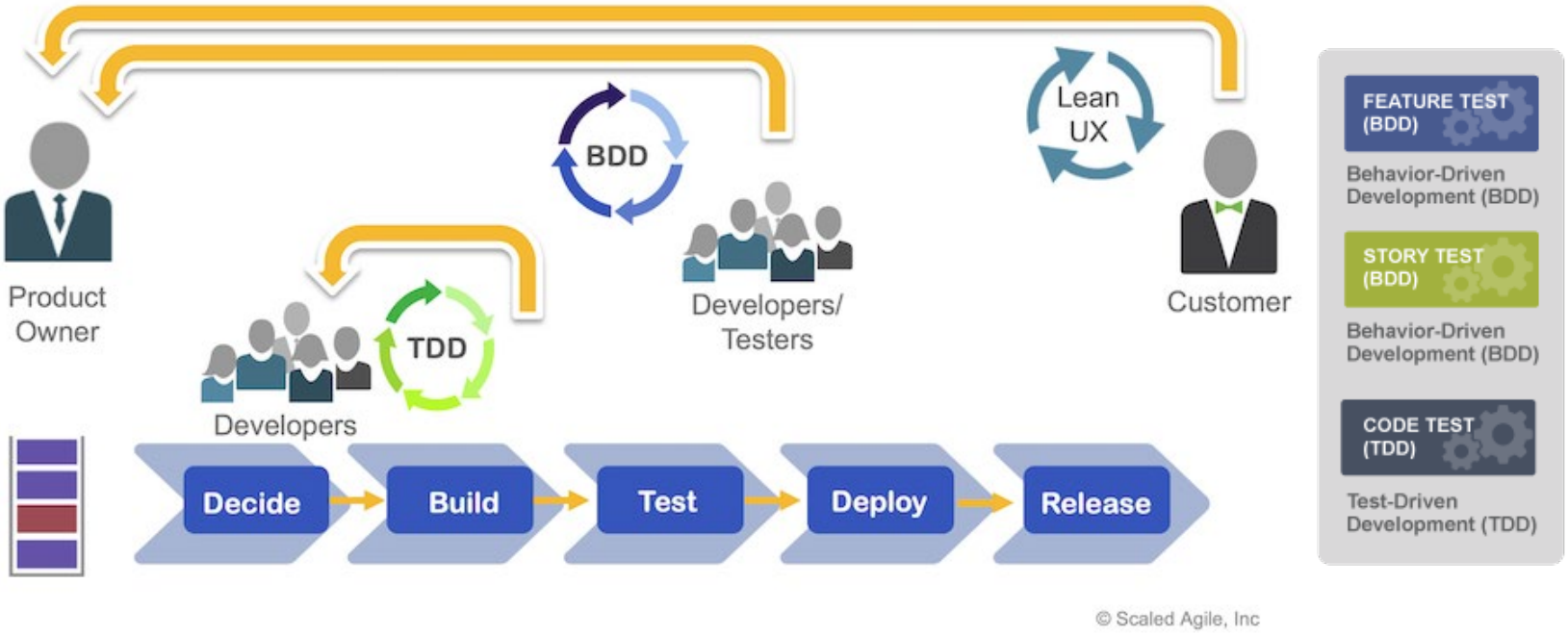
Defects have impact in terms of cost, reputation/trust, and legal issues.



Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality.*

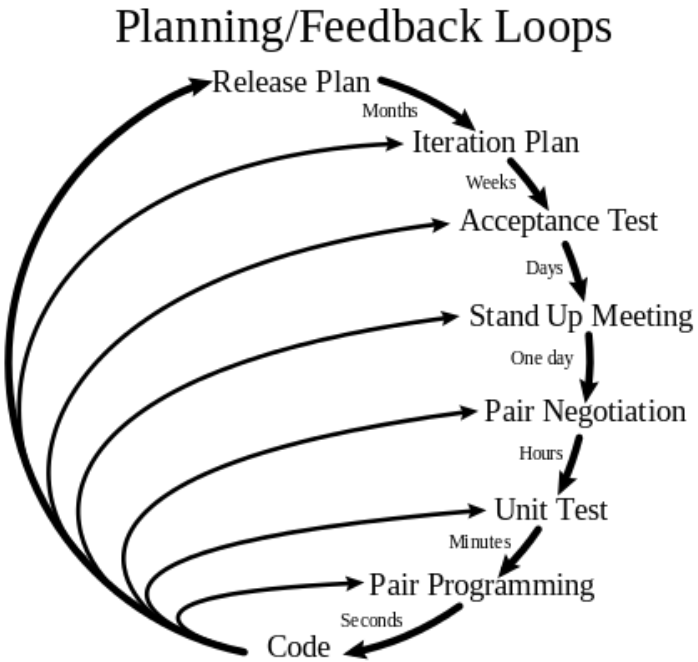
https://www.stickyminds.com/sites/default/files/shared/2018-12-10%20ArthurHicken%20The%20Shift-Left%20Approach%20to%20Software%20Testing%20image3.png

Establish Feedback Loops



https://www.scaledagileframework.com/built-In-quality/

Extreme Programming (XP) Feedback Loops

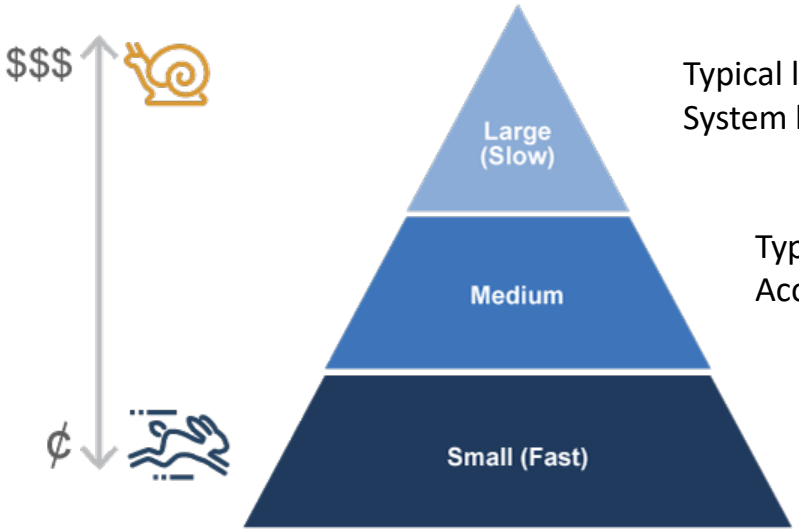


https://upload.wikimedia.org/wikipedia/commons/thumb/8/84/Extreme_Programming.svg/480px-Extreme_Programming.svg.png

Where to start?

- Think of the test pyramid idea!

- We suffer a lot from turn around times
- We need fast feedback
- We need confidence
- We need to run tests after EVERY single code change

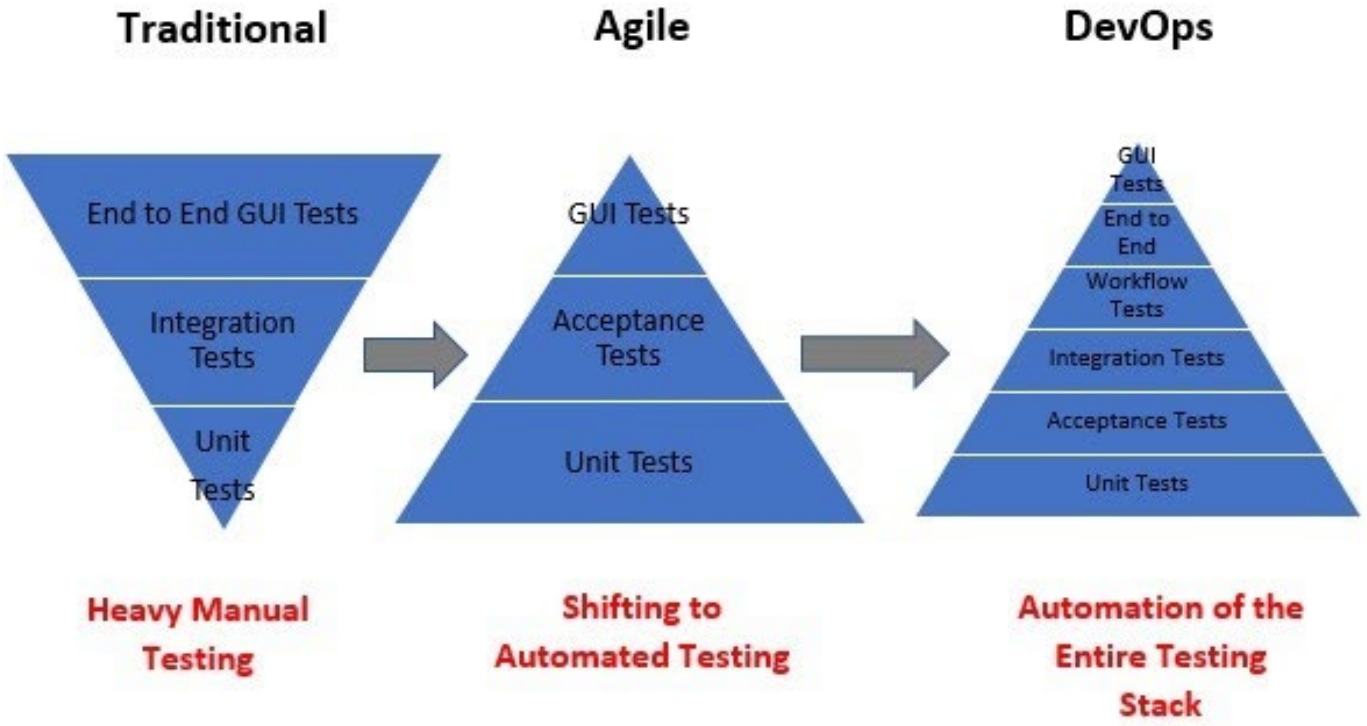


Typical large
System level scenarios

Typical medium
Acceptance tests

Typical small
Running at build time
Unit & integration tests

Inverting the Pyramid



http://www.adapttransformation.com/wp-content/uploads/flip.jpg

Test Driven Development (TDD)

- Requirements specified as tests
- Ensure test fails first
- Small dev units

Phases

1. Add a test
2. Run all and see if one fails
3. Write code
4. Run tests
5. Refactor code
6. Repeat

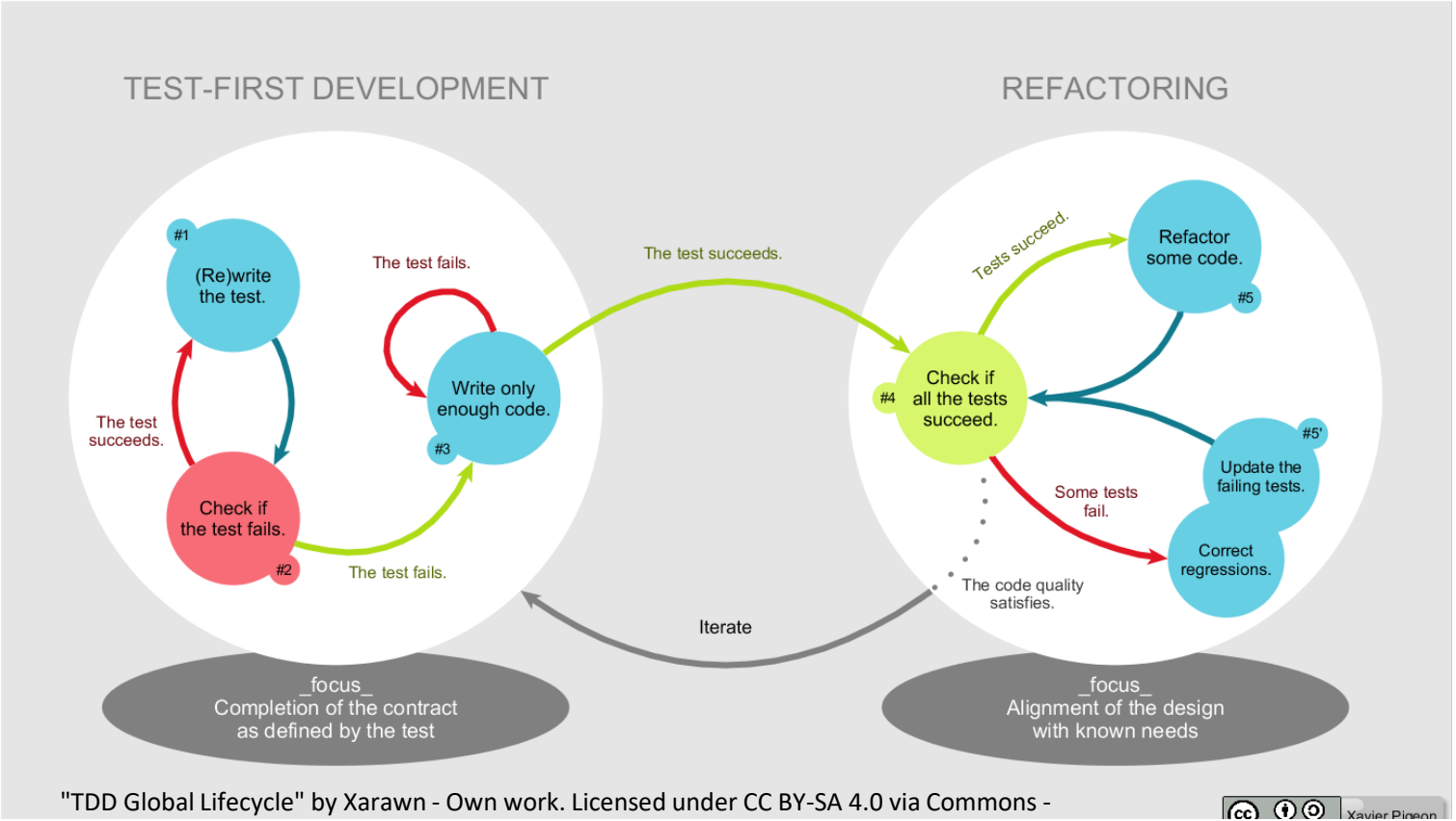
Pros

- Increased quality?
- Increased testability
- Everything gets tested
- Reduced debugging effort
- Revert vs debug

Cons

- Still need non-unit test phases

Test Driven Development (TDD)



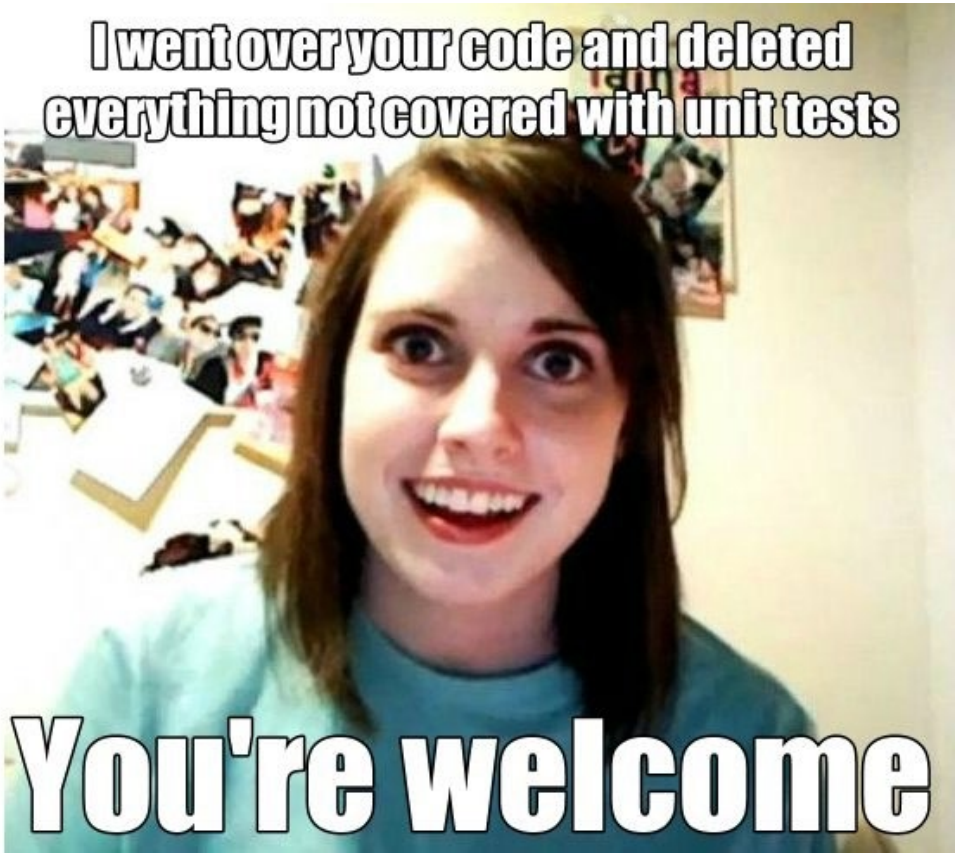
"TDD Global Lifecycle" by Xarawn - Own work. Licensed under CC BY-SA 4.0 via Commons - https://commons.wikimedia.org/wiki/File:TDD_Global_Lifecycle.png#/media/File:TDD_Global_Lifecycle.png

TDD

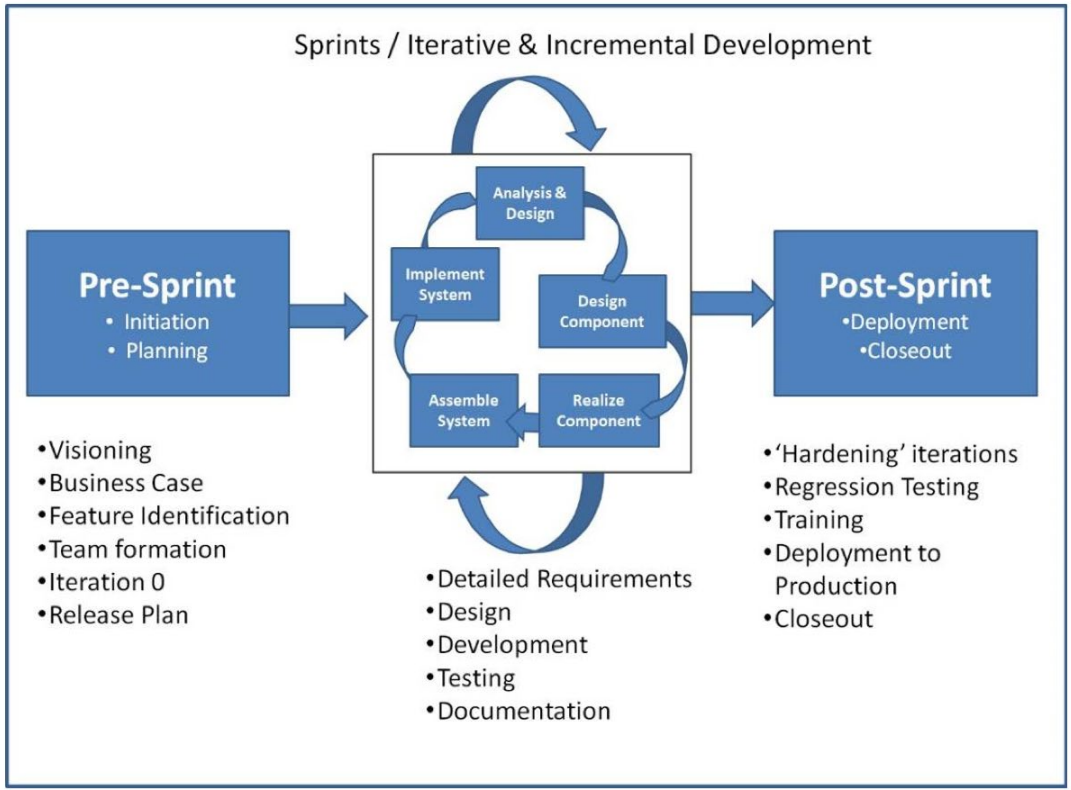
- Incremental dev aid
- Write the small tests first
- Provides testability
- Steep learning curve

Code Coverage

- Code coverage is the percentage of code which is covered by automated tests
- Use it, it really helps!
- Increase in coverage yields confidence not reliability/quality
- Small Advice:
  Never measure developers or testers by code coverage!
- Goal should be 100% coverage of at least new or changed code



Agile Cycle


Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
Analysis & Design
Implement System
Design Component
Assemble System
Realize Component
Post-Sprint
•Deployment
•Closeout
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Pre-Sprint

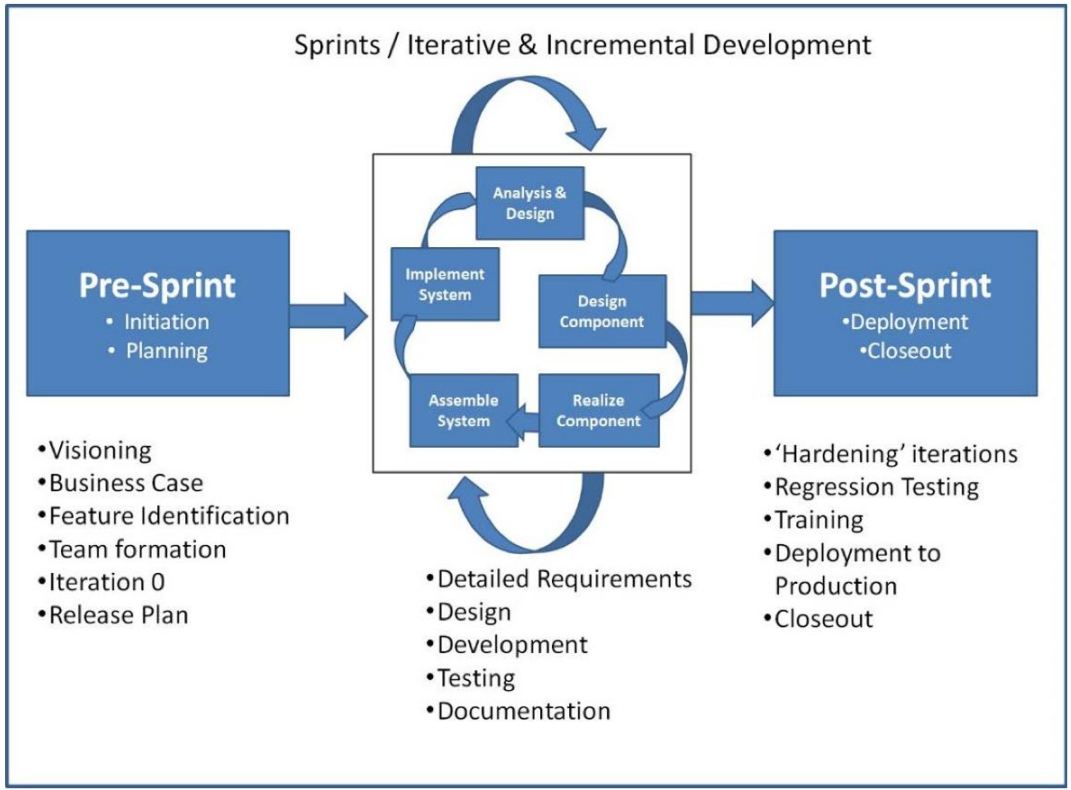
Before starting any project, it is good practice to complete an "iteration 0" to document design decisions and test considerations.

A **test brief** is a living, breathing document that serves as a place to document test strategies, acceptance and exit criteria, stakeholders and requirements to consider testing "complete" for a project.

Also a good time to acquire and arrange **dependencies**, develop **workloads**, and work on **automation**.

Identify a solution level test architect or **focal to oversee the end-to-end** solution.

Agile Cycle


Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
Analysis & Design
Implement System
Design Component
Assemble System
Realize Component
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
Post-Sprint
•Deployment
•Closeout
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Test Tasks within an Iteration

- Iteration Planning
- Attend Design Reviews
- Put in tool requirements for test
- Attend Code Reviews
- Write Variations / Scenarios
- Review Variations / Scenarios
- Write/Update test infrastructure
- Write Test Programs (test cases)
- Develop Workloads
- Review Test Programs

Test Tasks within an Iteration

- Iteration Planning
- Attend Design Reviews
- Put in tool requirements for test
- Attend Code Reviews
- Write Variations / Scenarios
- Review Variations / Scenarios
- Write/Update test infrastructure
- Write Test Programs (test cases)
- Develop Workloads
- Review Test Programs

- Execute Test programs / Workloads
- Debug Test programs or problems found
- Document problems found
- Re-run to validate fixes
- Execute Regression Tests
- Review Publications/Service Transfer Education
- Hold Iteration Reflections/Retrospectives
- Misc Meetings: standup/status/education
- Reference/Update Test Brief
- COMMUNICATE w/ Project/Release Managers and TestOps

Why do testers attend Code Reviews?

- Find errors in code
- Education on work item
- Ask questions to clarify code
- Get ideas for test strategy / variations

What are we looking for in Code reviews?

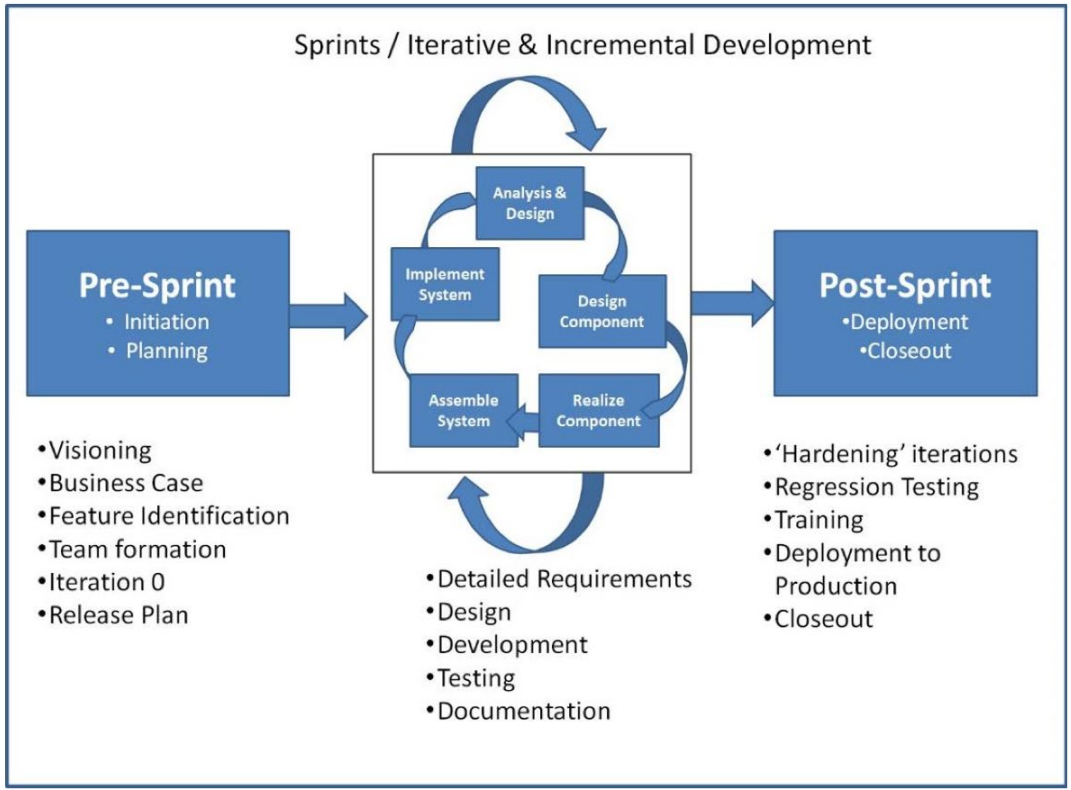
- **Complicated parts of the code** – more chance of mistakes
- **Resources used** – make sure that they are released
- **Serialization** – are resources being used properly
- **Authorization** – are we manipulating user storage and system storage correctly
- **Recovery/Retry/percolation**
- **Boundary conditions** (external and internal)

What are we looking for in Code reviews? (cont)

- **Interactions** between different functions/code modules/components
- **Environmental** requirements/restrictions
- Validate that code **implements the design**

  (return/reason codes, abend codes, trace records)
- **Break points** – testability (traces)
- **Serviceability** – comments, explanations, readability, variable names, etc.
- Gather variation ideas

Agile Cycle


Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
Analysis & Design
Design Component
Realize Component
Assemble System
Implement System
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
Post-Sprint
•Deployment
•Closeout
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Iteration Planning

- Why do iteration planning?
- What questions need to be answered to help with the planning?
- How do we answer these iteration planning questions?

Why do Iteration Planning?

- To map out the work for the iteration.
- To understand how long it will take to do the work
- To understand what is needed to do the work within the iteration

What questions need to be answered to help with the planning?

- Are there any Hardware/Software dependencies?
- What is the experience level of the team? (in test; component; test tools)
- How large is the work item? (lines of code)
- How complex is the work item?
- What tools are required to test this work item?
- What problem areas exist from previous releases?

What questions need to be answered to help with the planning? (cont)

- Does the team need any education on work item?
- How many testers will be needed?
- Does the regression bucket need to be updated/created?
- How many variations will it take to test the work item?
- What are the high level test objectives for this epic?
- What is being implemented by this story?
- How is this story being validated?

How do we answer these iteration planning questions?

- Look at design documents for dependencies
- Understand experience level of team
- Talk with developers to get a sizing of the work item
- Talk with developers to get understand the complexity of the work item
- Talk with developers and other testers to understand what tools are needed
- Talk with service team to understand problem areas in component

FVT considerations for Iteration Planning

- Talk with the team about the work item to determine what education is needed
- Understand the regression bucket for your component/area
- Use Gross estimate for number of function test variations
  - General Rule: 1 variation per 10-20 lines of code
  - More variations if testing is external; Less if testing is internal
  - Execution time:
    - Complex – 1-3vars/day
    - Medium – 4-7vars/day
    - Trivial – 8-10vars/day

SVT considerations for Iteration Planning

Planning involves defining what will be done and then how it will be done

Is a system test required?

- Explicit – Specific targeted testing or regression
- Implicit – Function is verified simply by having an active system
- Implement Only – Function/feature must be turned on explicitly

How many machine shots are required?

- Do I need consecutive machine shifts/shots?
- Do I need specific hardware or a specific configuration?
- Do I need to perform non-disruptive vs disruptive testing?

Are additional testing teams required?

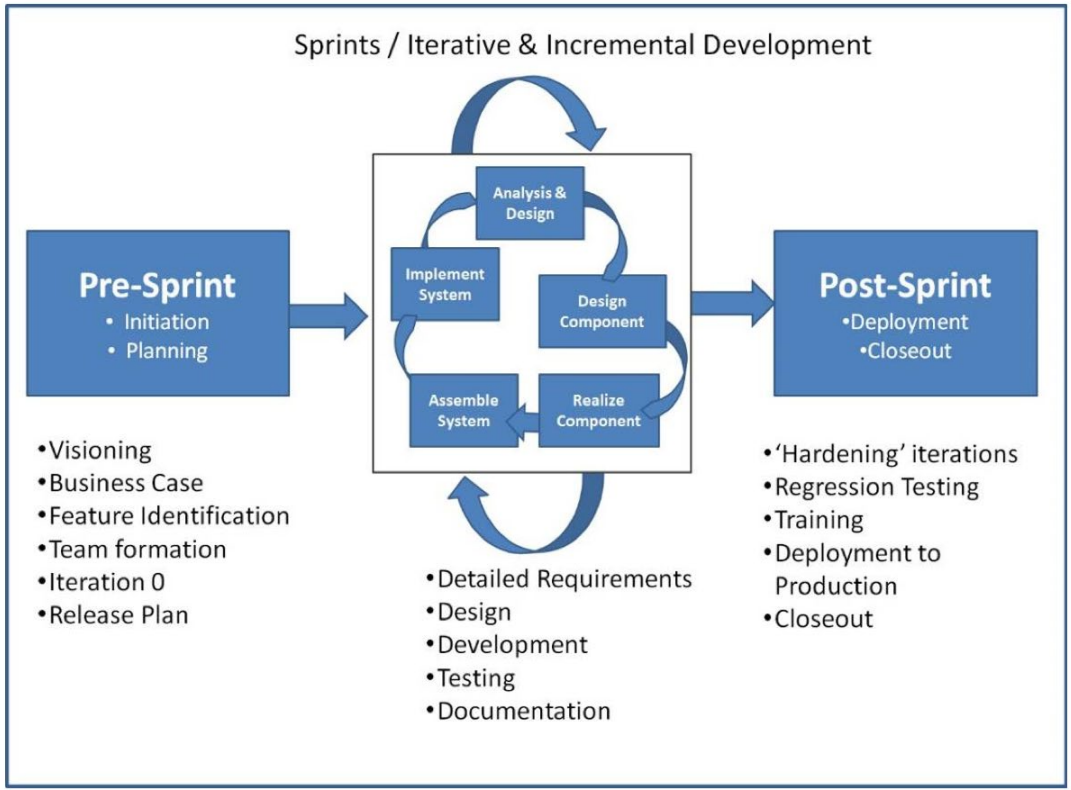
SVT considerations for Iteration Planning (cont)

Contingency plans

- Finding defects may gate or delay progress while waiting for fix
  e.g. ~1 week?...What if the developer is on vacation?
- Time needed to verify fixes
- Machine availability – usually only 1-2 usable machine shots available per week
- Experience of testers
- Unknown unknowns
  Machine failures, unable to boot, firmware updates, environment issues

Total time required to test may be double what was anticipated

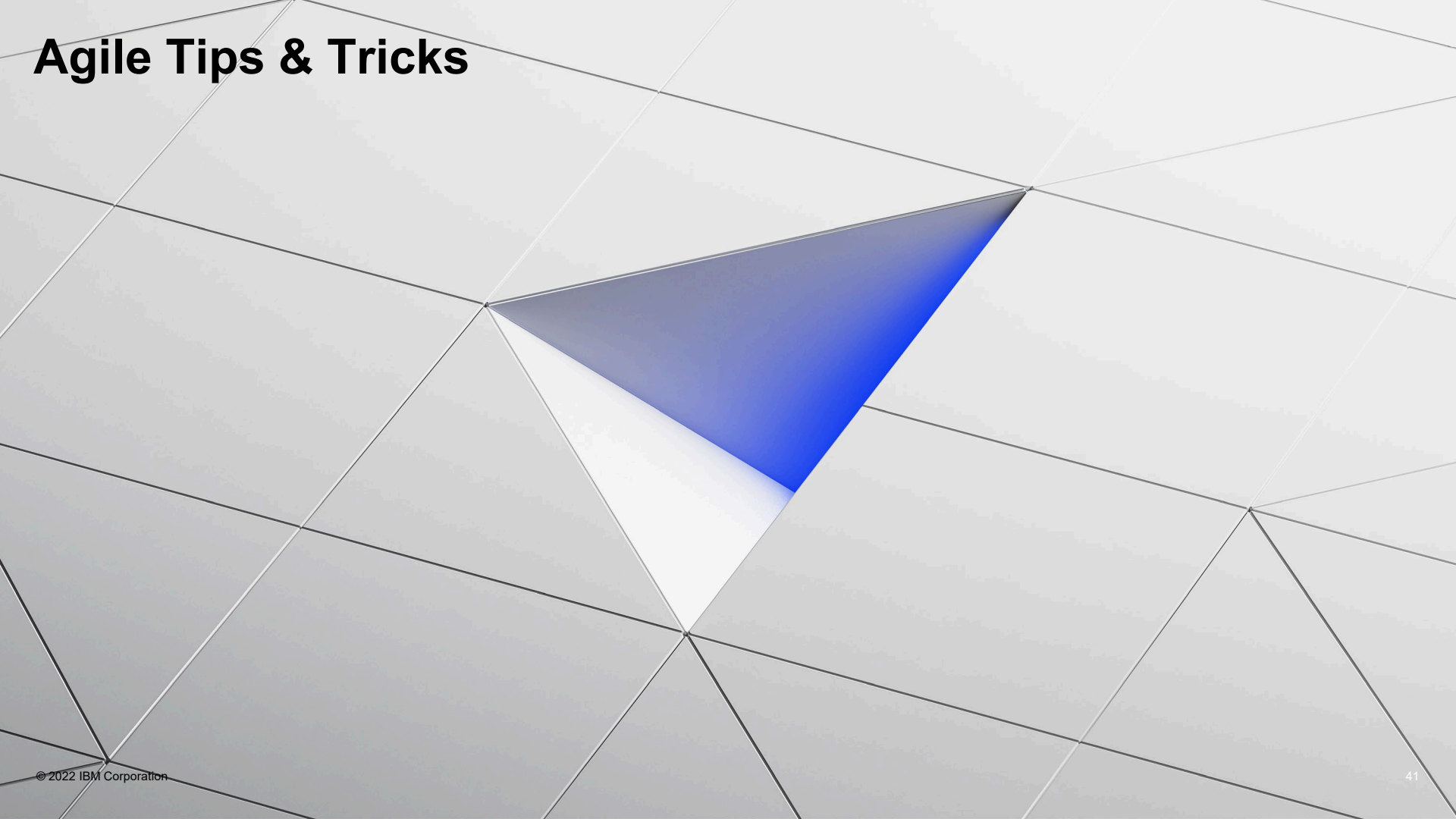
Agile Cycle


Sprints / Iterative & Incremental Development
Pre-Sprint
• Initiation
• Planning
Analysis & Design
Implement System
Design Component
Assemble System
Realize Component
Post-Sprint
•Deployment
•Closeout
•Visioning
•Business Case
•Feature Identification
•Team formation
•Iteration 0
•Release Plan
•Detailed Requirements
•Design
•Development
•Testing
•Documentation
•'Hardening' iterations
•Regression Testing
•Training
•Deployment to Production
•Closeout

Post-Sprint/Iteration

- Full regression testing
- Promote new tests into regression buckets/suites
- Verify documentation and examples
- Verify any outstanding fixes
- Final packaging test
- Higher level retrospectives/reflections
- Defect trend analysis
- Finalize intellectual property (e.g. patents)
- Hand off to external test teams (e.g. TestOps, Service Test)

Agile Tips & Tricks



Guilds

- Community of like minded people with a common interest
- Common education
- Un-conferences (Conferences but less rigorous)
- Hack days
- Mentoring

Guilds

- Community of like minded people with a common interest
- Common education
- Un-conferences (Conferences but less rigorous)
- Hack days
- Mentoring

Test Guild

- Larger community of testers
- Learning and trying new ideas together
- All phases/kinds of testers
- May have chapters (smaller groups) in guild with more specific common goals
    - By test discipline or automation framework/tool

Agile Testing Priorities

- Create a 1 to N list of prioritized tests across test phases and goals
- Needs to be approved by whole squad, brand, test lead, and PO
- Utilize various defect tagging
- During iteration
  - Mainline function
  - Security
  - Regression
  - Main client environments
  - Stress test
- Post-sprint or hardening iteration
  - Recovery
  - Edge cases
  - Full regression
  - Narrow environments
  - Limits testing

Agile Testing Tips/Tricks

- Use Test Driven Development (TDD), it helps improve testability and reduce “gold plating”
- Constantly reevaluate testing inclusion/coverage

  always be automating and "shifting left" (aka speedup iteration feedback)
- Test Code Quality must be equal to Production Code Quality
- Don’t get hung up on formalism or sharp definitions, primarily try to focus on constantly improve confidence, quality and turnaround times
- All testers included from project initiation (e.g. TestOps, multiple component areas, …)
- Test and acceptance or complete criteria need to be defined at multiple levels
  - Stories – Test and regression based on new/changed feature
  - Epics – Full end to end test w/ full regression runs

DevOps

Software Development Lifecycle (SDLC)

Development

- Design
- Code
- Build
- Test
- Release

Operations

- Staging
- Deploy
- Operate & Monitor

Done, throw it over the wall!

What is DevOps?

”DevOps speeds delivery of higher quality software by combining and automating the work of software development and IT operations teams.”

https://www.ibm.com/cloud/learn/devops-a-complete-guide

What is DevOps?

”DevOps **speeds delivery** of higher quality software by combining and automating the work of software development and IT operations teams.”

https://www.ibm.com/cloud/learn/devops-a-complete-guide

What is DevOps?

”DevOps speeds delivery of **higher quality** software by combining and automating the work of software development and IT operations teams.”

https://www.ibm.com/cloud/learn/devops-a-complete-guide

What is DevOps?

”DevOps speeds delivery of higher quality software by **combining and automating** the work of software development and IT operations teams.”

https://www.ibm.com/cloud/learn/devops-a-complete-guide

What is DevOps?

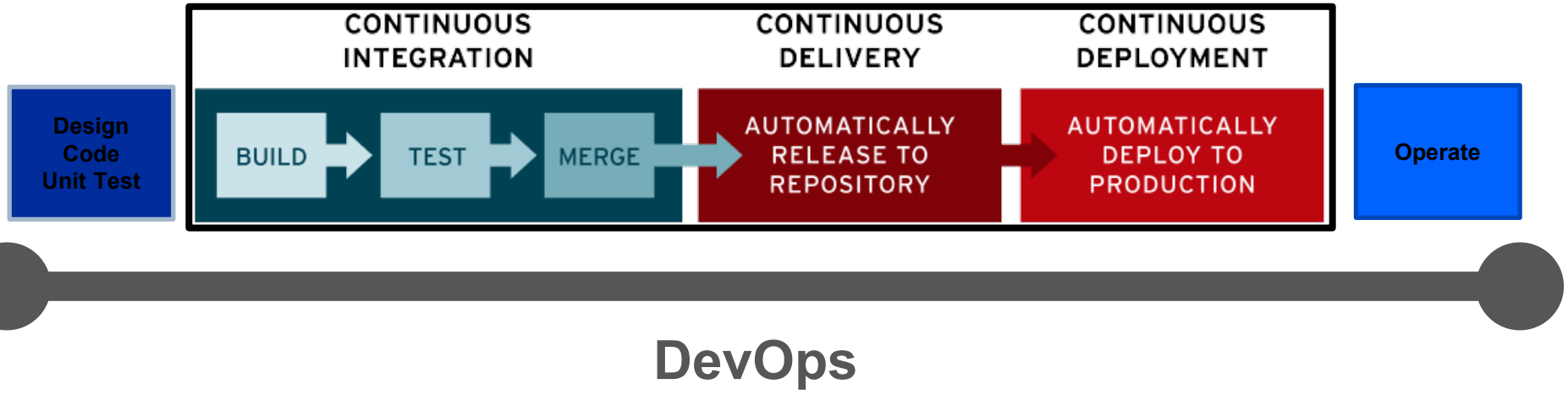
”DevOps speeds delivery of higher quality software by combining and automating the work of **software development and IT operations teams**.”

https://www.ibm.com/cloud/learn/devops-a-complete-guide

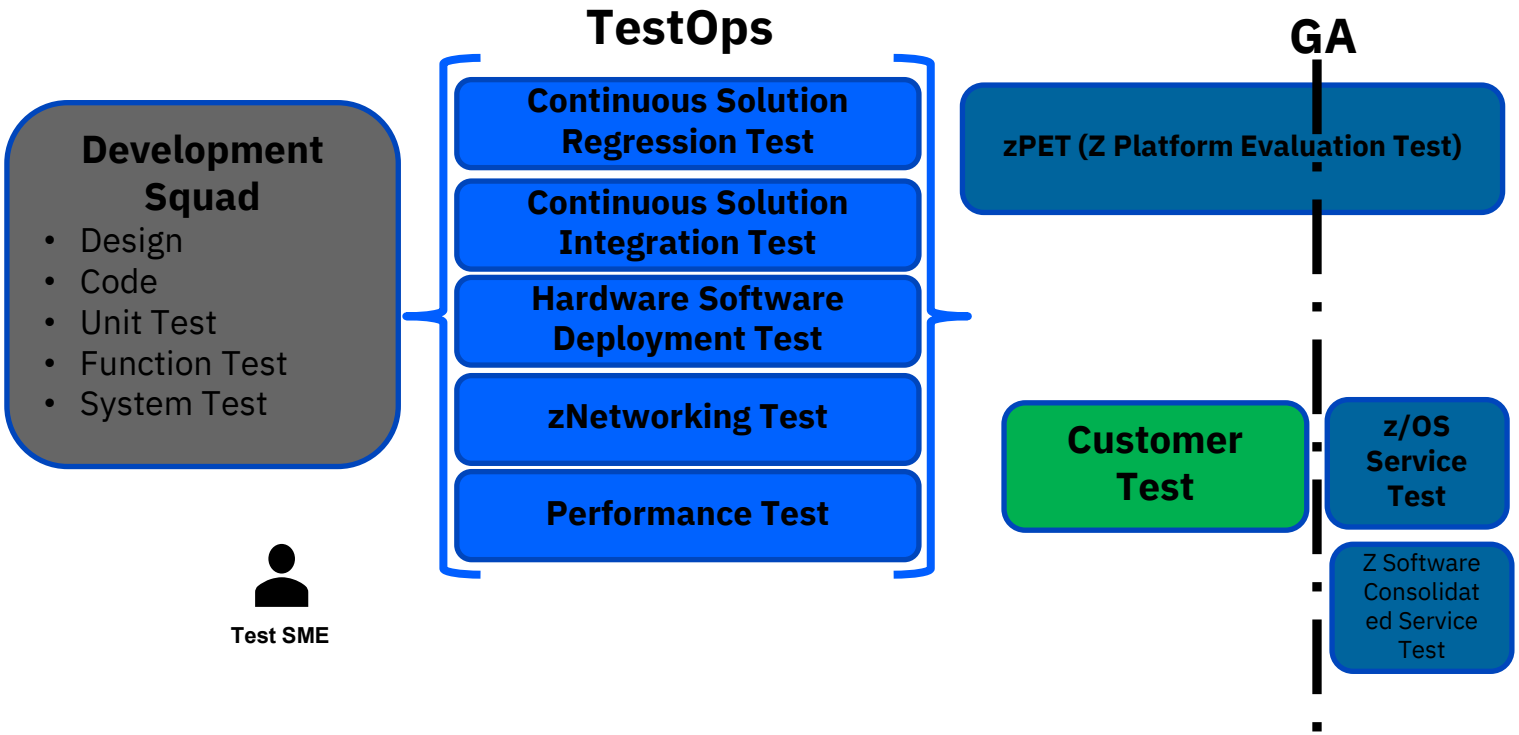
CI/CD

https://www.redhat.com/en/topics/devops/what-is-ci-cd

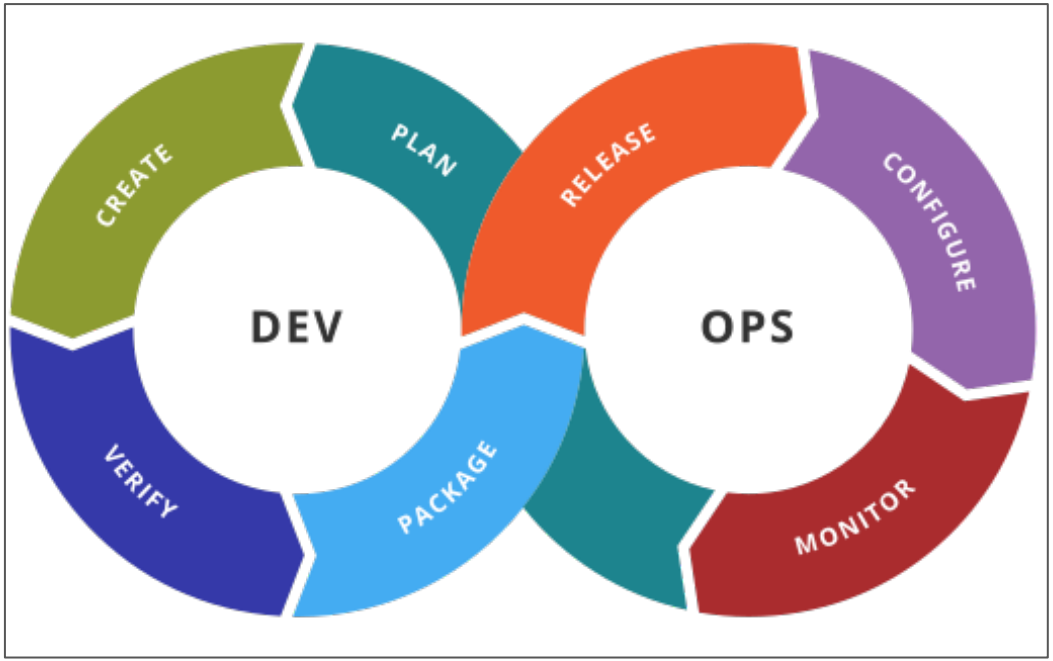
DevOps


CONTINUOUS INTEGRATION
CONTINUOUS DELIVERY
CONTINUOUS DEPLOYMENT
Design Code Unit Test
BUILD
TEST
MERGE
AUTOMATICALLY RELEASE TO REPOSITORY
AUTOMATICALLY DEPLOY TO PRODUCTION
Operate
DevOps

z/OS Test Ops Squads

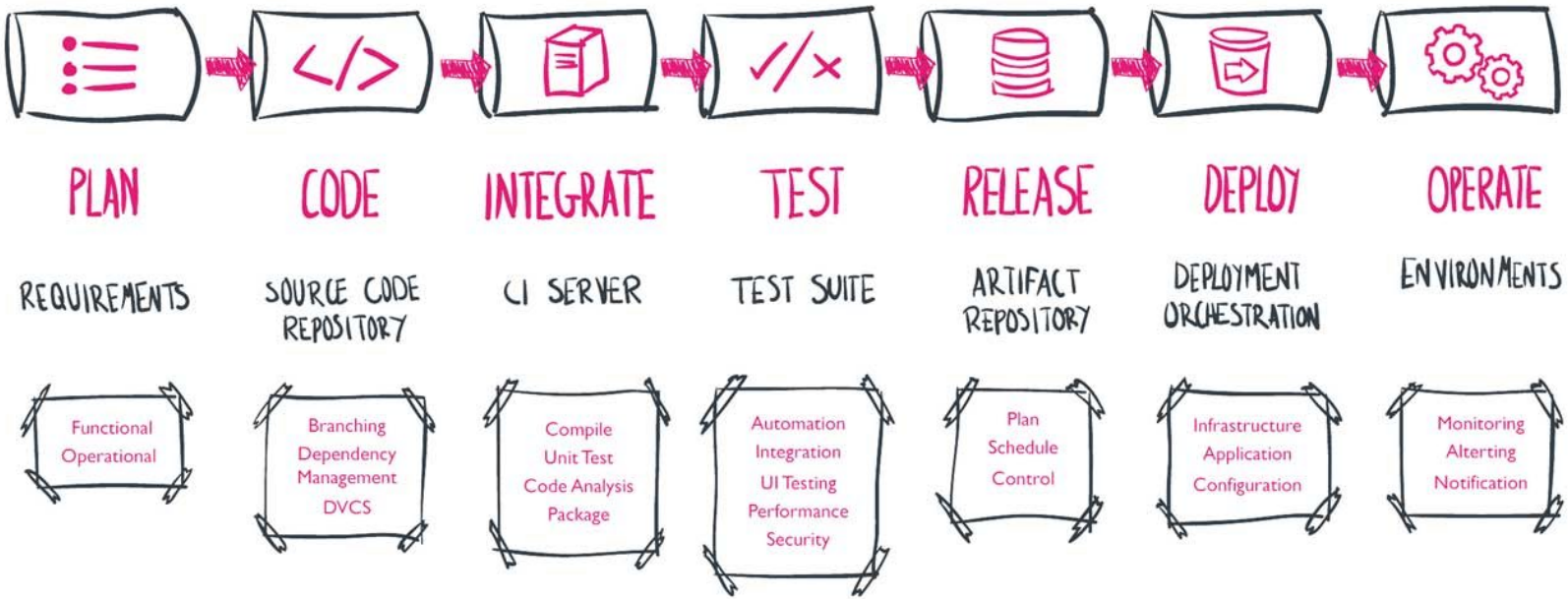


DevOps



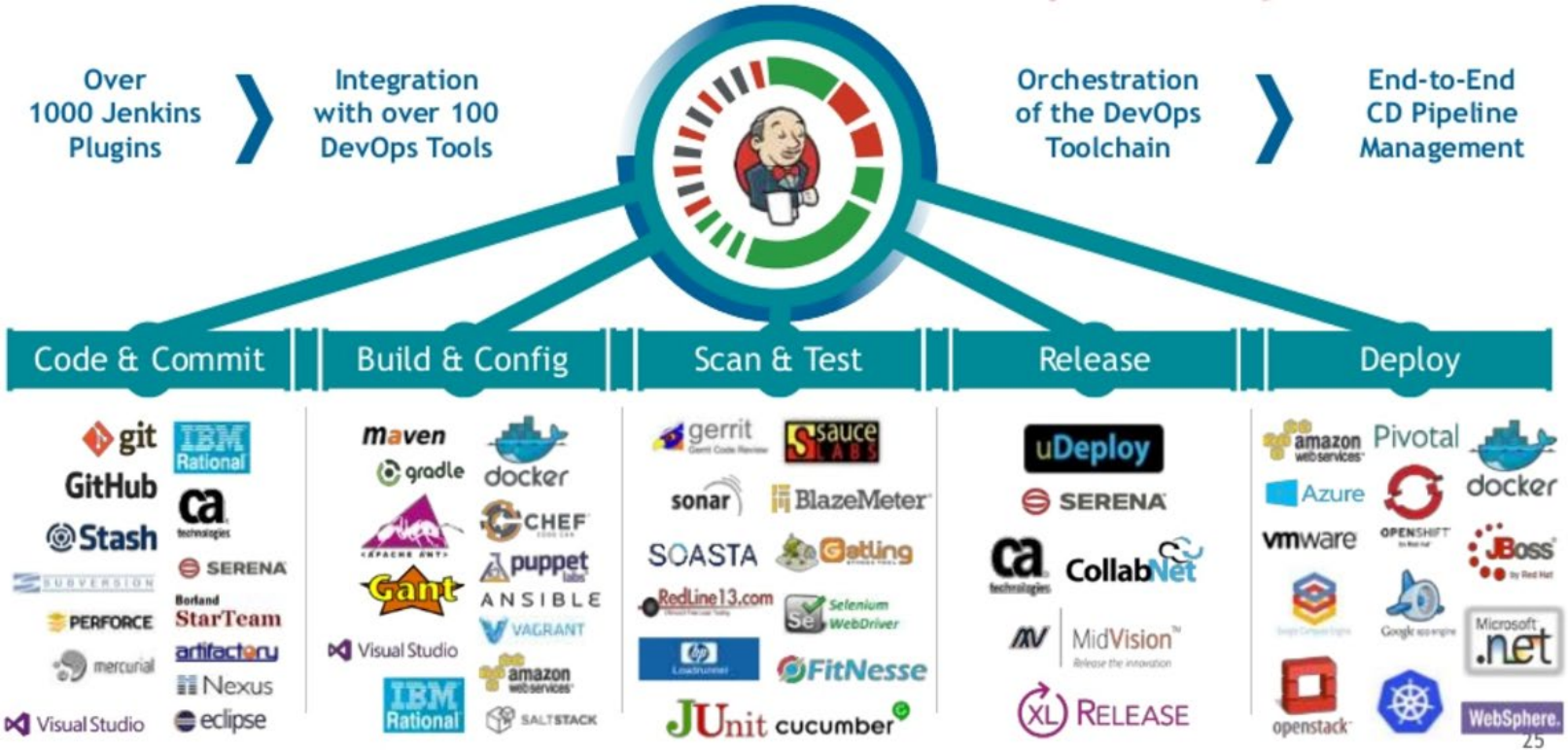
https://en.wikipedia.org/wiki/DevOps#/media/File:Devops-toolchain.svg

DevOps



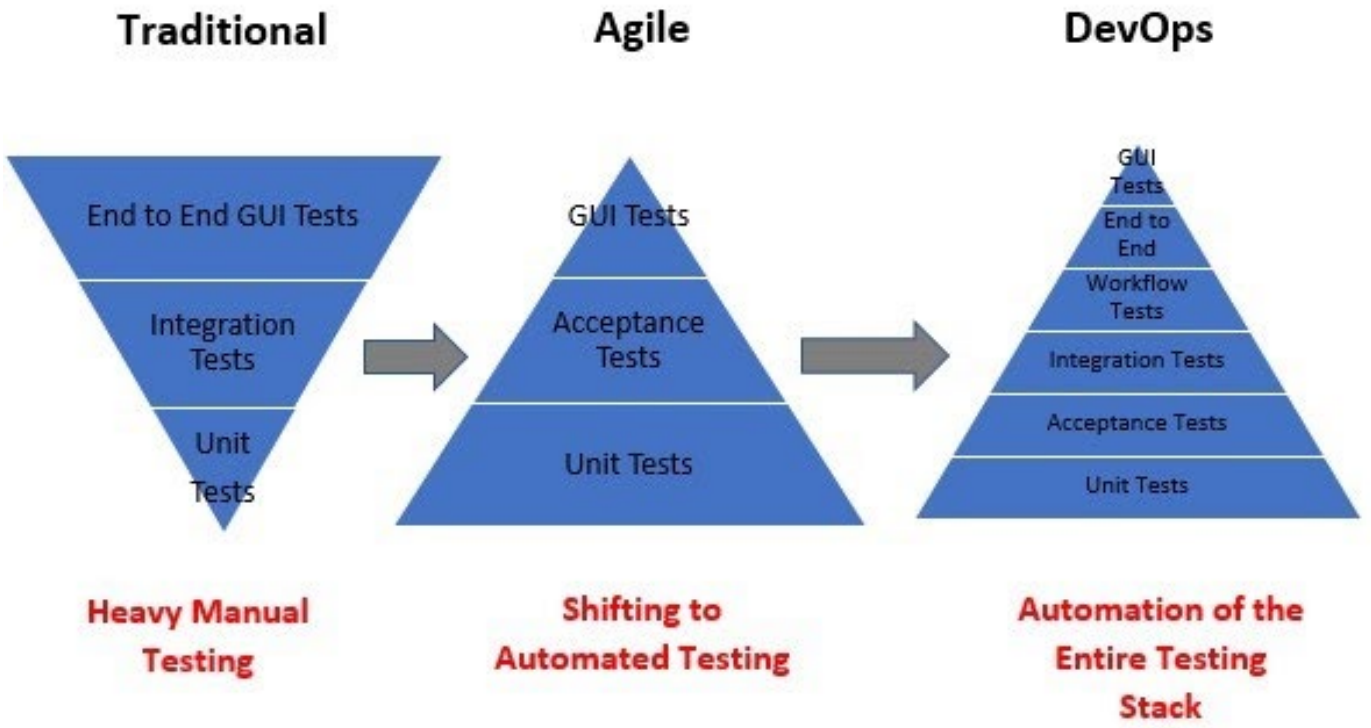
https://www.devopsgroup.com/wp-content/uploads/2019/05/devopsgroup_blog_pipeline_assessment.jpg

DevOps Pipeline



https://hostadvice.com/wp-content/uploads/2018/03/devopsjenkins.png

Inverting the Pyramid



http://www.adapttransformation.com/wp-content/uploads/flip.jpg

Questions?