Testing the Manifesto

June 2020

Chicago Quality Assurance Association

Speaker: Paul Herzog

Agenda

01

Setting Agile Expectations

Let's see what everyone has done before…myself included….

02

Reviewing the Manifesto

Where does it come from and what are we trying to accomplish through it?

03

Living the Principles

Focus on the philosophy, not on the day-to-day

04

Q & A Discussion

What else do we want to talk about?

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Individuals and Interactions over Software and Tools

- Are you fully integrated with your team? Sit with them (if that's a possibility), participate in all the ceremonies and meetings?
- Do time and complexity estimates include testing, or just development effort?
- Do you take notes or otherwise document face-to-face communication, so that understanding is confirmed, and conclusions shared with other team members?

Working Software over Comprehensive Documentation

- Do you have access and knowledge of software architecture and repository? Or are you waiting for the team to throw you a new build?
- How aware are you of what is contained in each new release?
- Does the team write up other design or other documents beyond what's in the User Story? How are those documents used?

Customer Collaboration over Contract Negotiation

- Do you KNOW your customer? Who has the greatest stake in the delivery of your application?
- How often do you see them? Know their names?
- What business problem your application is solving? How will your software add value to the company?

Responding to Change over Following a Plan

- How often does your team pick up new stories in the middle of a Sprint?
- Do you have a good process for deciding what gets dropped if something gets added?

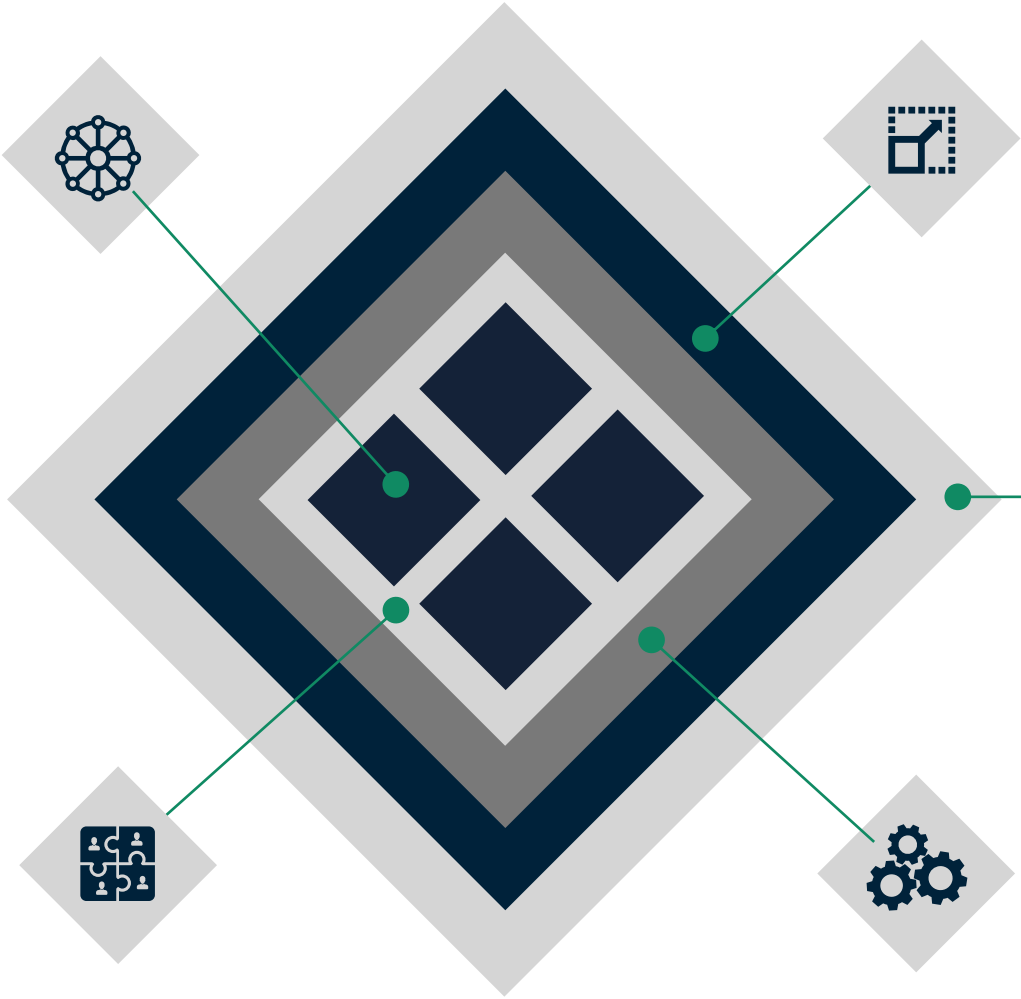
Testing Based on the 12 Principles

Principle #1: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

UNIT TESTING
Verifying positive & negative conditions for each atomic unit of code

- Tests must be run during each stage of a CI/CD pipeline without having a dramatic impact on time to delivery, starting in Sprint 1
- Automate everything that can be automated…but no more

SERVICE TESTING
The message formats and behavior between units at a service or data level



REGRESSION TESTING
Ensuring new units, services and functions did not break what worked before

UAT
Beta set of end-user community representatives give their evaluation

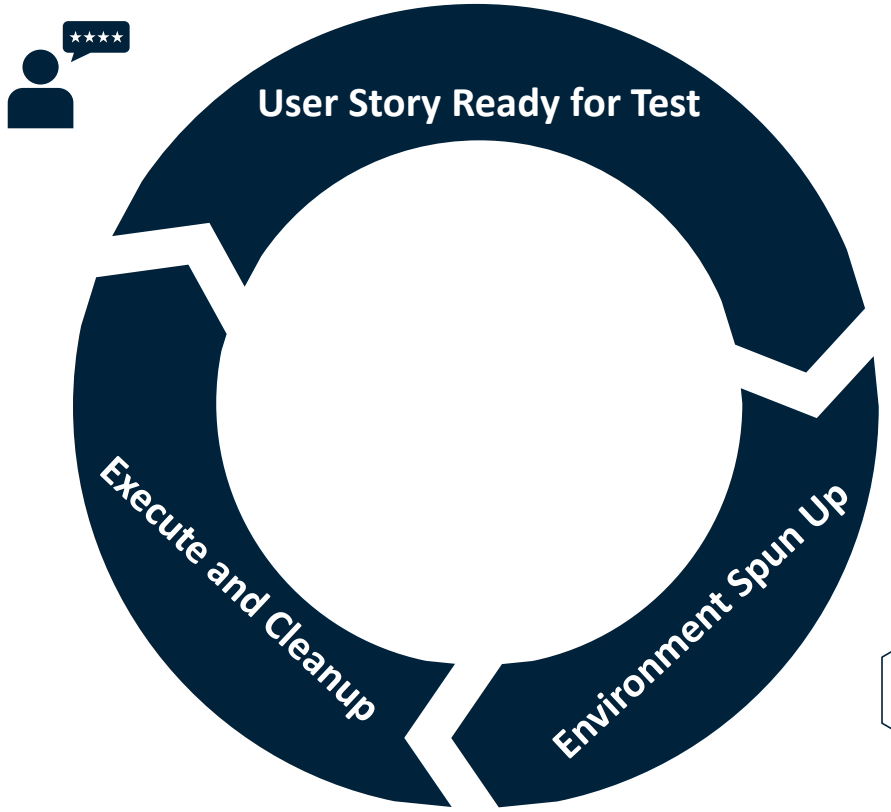
FUNCTIONAL TESTING
Validating the user stories and acceptance criteria for accurate and robust behavior

Principle #1: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

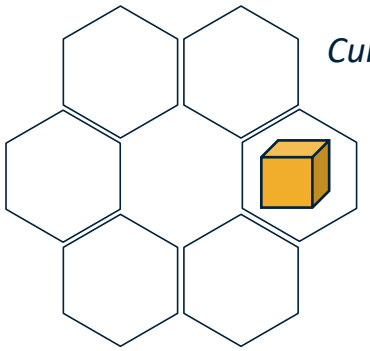
- Value is only provided by verification and validation of software against architecture specifications, customer expectations and business requirements, in environments which reasonably reflect those to be used by the customer.
- Things which provide no value:
  - “It works on my machine”
  - Environments containing an insufficient or incomplete amount of data
  - Stale data, obsolete schema, old configurations, application remnants blocking an efficient test cycle
  - Processes blocking equivalent roles or authentication from taking place in test environments
  - Testers who do not go out of their way to understand the customer’s business model and what role this application plays within it

Principle #1 solution: DevOps workflows support on-demand testing environments for common usage

"New story is ready to test!"



"I've created a CD pipeline. Here's a VM from our cluster for testing."



Current feature or release branch
Environment & app config
Default data image

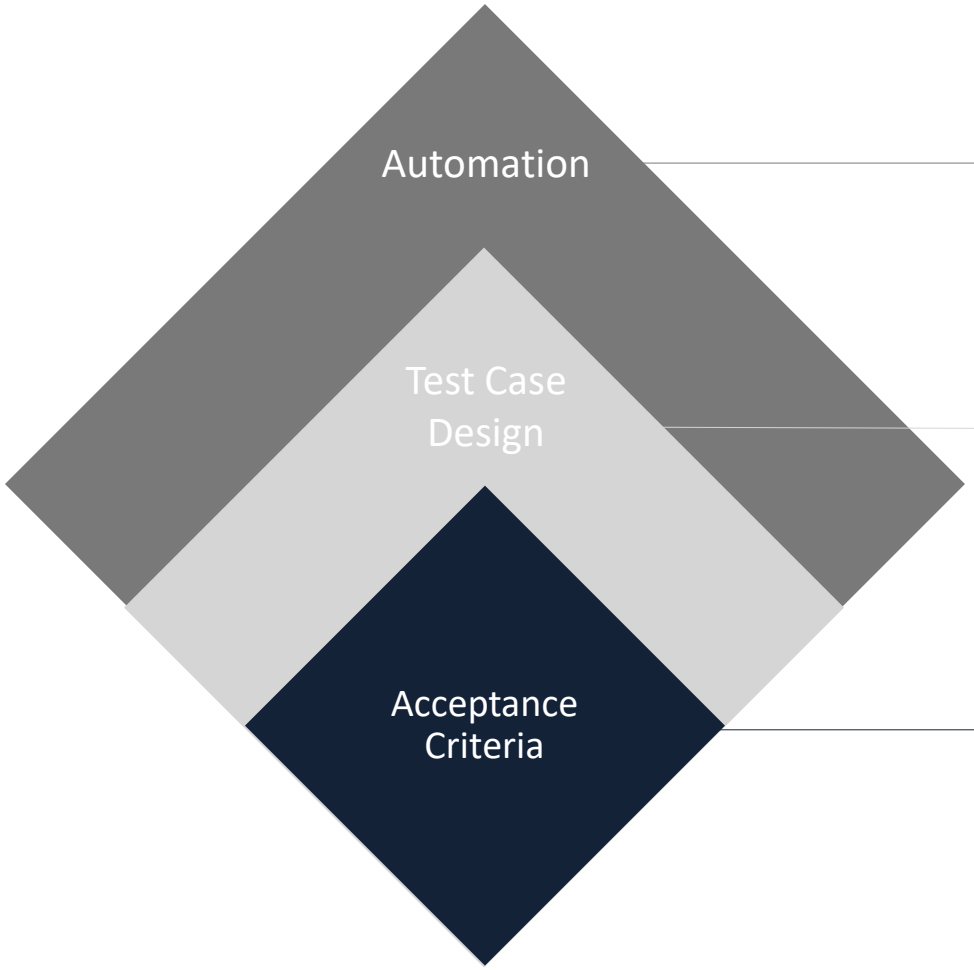
"Test, test, test...oh, look, a bug! Show the developer, write it up. And…we're done!

Principle #2: Welcome changing requirements, even late in development

- Additional, modified, or reprioritized requirements should meet the team's standards for completeness
- Changes must come with acceptance criteria definitions with enough detail to break down the positive & negative scenarios

- To quickly incorporate changes, information must be presented as consistently and completely as possible
  - Teams following common definition syntax for user stories, acceptance criteria, etc., can dramatically accelerate understanding and development of work

Principle #2 solution: Behavior-Driven Development usage across the project backlog



BDD AUTOMATION PLATFORMS

Cucumber: open-source for Web automation, based in Ruby
SpecFlow: Cucumber port for C#/.NET for nUnit & xUnit
jBehave: Java-based alternative for jUnit integration

GHERKIN IS NATURAL FOR TESTS

Given: Pre-condition state required for test
When: Action steps for execution
Then: The expected results

GIVEN-WHEN-THEN

Common syntax for all team members
Consistently appropriate level of detail
Granular separation of criteria into positive/negative scenarios

Gherkin example

"As a mobile phone retailer, I want my back-end inventory system to automatically update with each transaction, so that I always have an accurate product count."

Acceptance Scenario #1: Customer buys iPhone

Given a customer is buying a new iPhone

When the purchase transaction is completed

Then the inventory count of iPhones is decremented by 1

Acceptance Scenario #2: Customer returns iPhone

Given a customer is returning a previously purchased iPhone

When the return transaction is completed

Then the inventory count of iPhones is incremented by 1

Acceptance Scenario #3: Customer exchanges iPhone

Given a customer is exchanging a previously purchased iPhone for a different one

When the exchange transaction is completed

Then the inventory count of iPhones is incremented by 1

And the inventory count of iPhones is decremented by 1

Acceptance Scenario #4: iPhone purchase cancelled

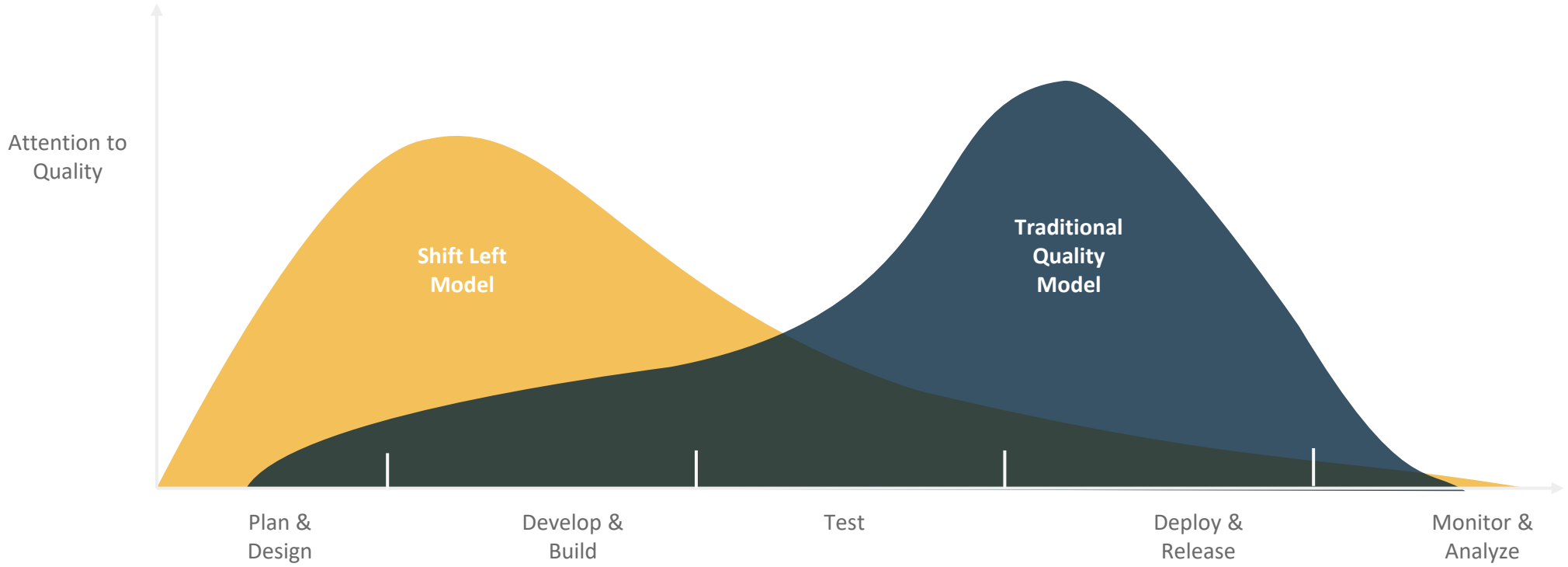
Given a customer is buying a new iPhone

When I cancel before the purchase transaction is completed

Then the inventory count of iPhones is unchanged

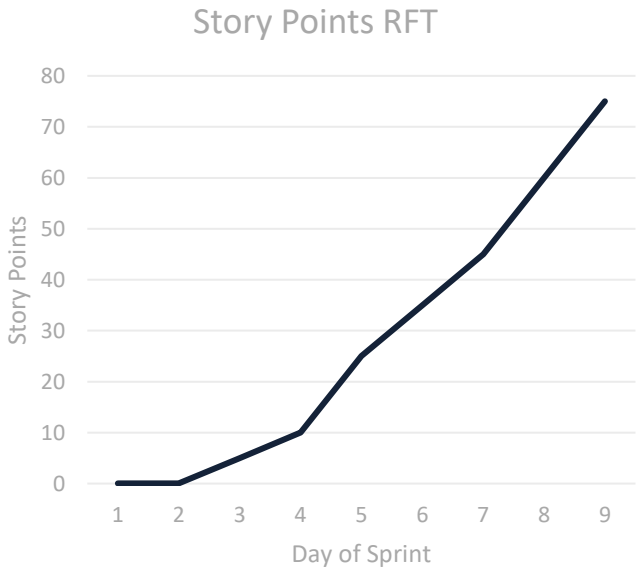
Principle #3: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

- Testers embedded within the team are planning, writing and executing tests as soon as possible
- Shift Left!



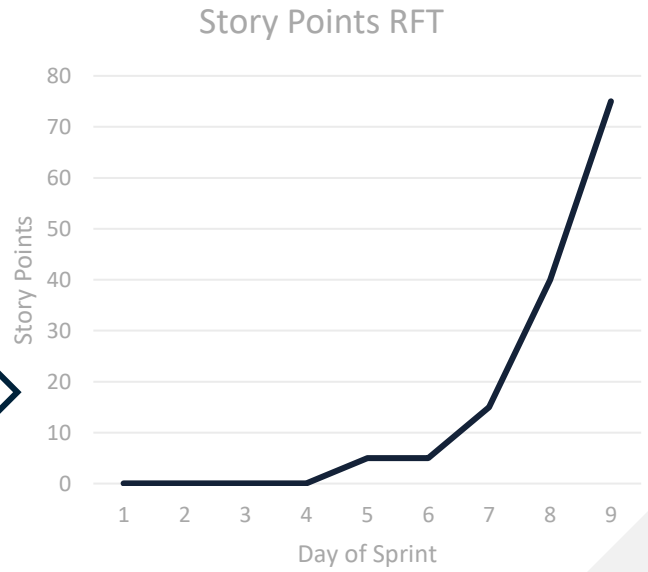
Principle #3 solution: Do not treat sprints as “mini-waterfalls” in delivering Ready For Test builds

- Each Sprint contains 9 days for developers to work on multiple stories
  - We lose 1 day, on average for Demo/Retro/Planning
- Developers should tackle stories and other work items one at a time, based on priority and dev/test workload balance
  - Multiple developers should team up on the highest-priority stories whenever possible
  - 2+ stories should never be “In Progress” for the same developer unless there are blocking issues
  - Stories with short dev and large test estimates should be tackled early in the Sprint



This!...

...not That!



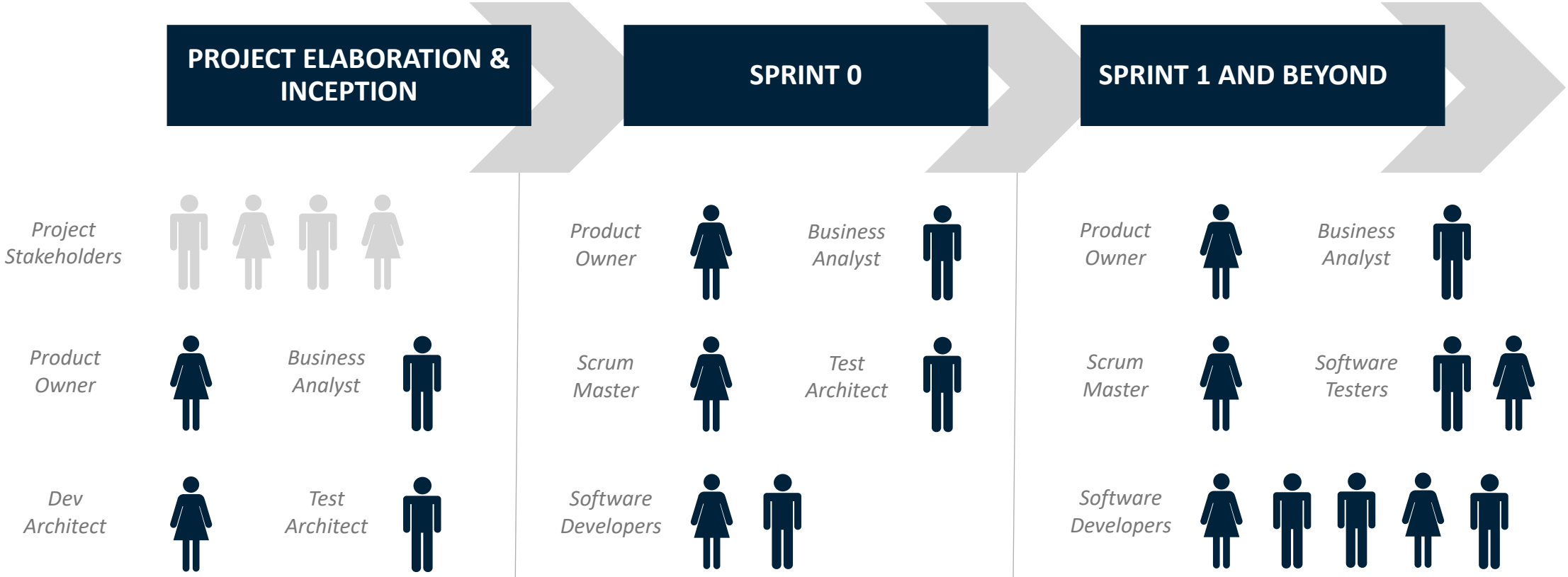
Principle #4: Business people & developers must work together daily throughout the project

- This includes testing…the tightest working relationship on a Scrum Delivery team should be between Business Analysts and Test Engineers
  - Both need strong end-to-end knowledge of how the application under development supports users
  - Both need the greatest clarity around the Acceptance Criteria, and what “good enough” means for a story or larger feature
  - Both should be speaking the Given / When / Then syntax for Acceptance Criteria and Test Scenarios

- The two roles can pair for a “Tiger Team”, taking the very first independent look at the completion of a story or task
  - Does the UI design match the wireframe and style sheet?
  - Do any simple operations fail?
  - Is there anything else which can be fixed quickly prior to check-in?

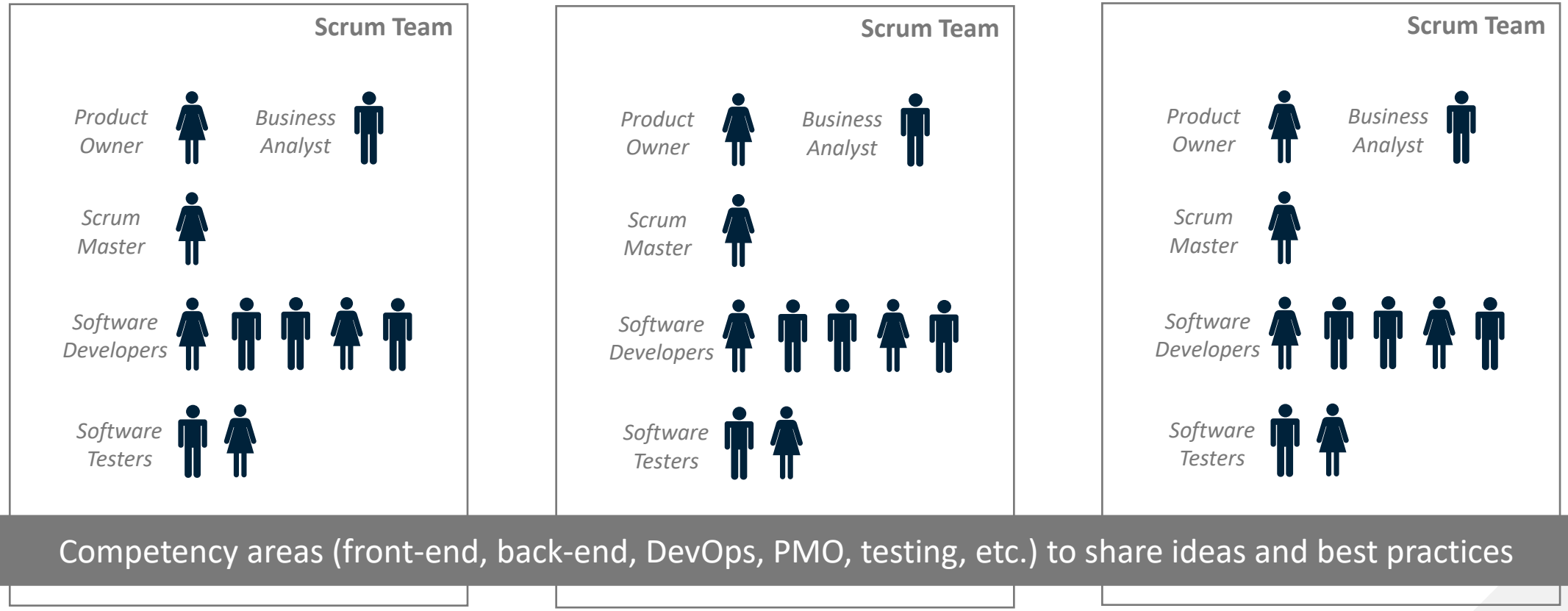
Principle #4 solution: the Tiger Team

Principle #5: Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.



Principle #5 solution: Avoid the centralized organization structure…testing or otherwise

The concept of staffing from a "Center of Excellence" is passé and works against the concept of a self-reliant project team



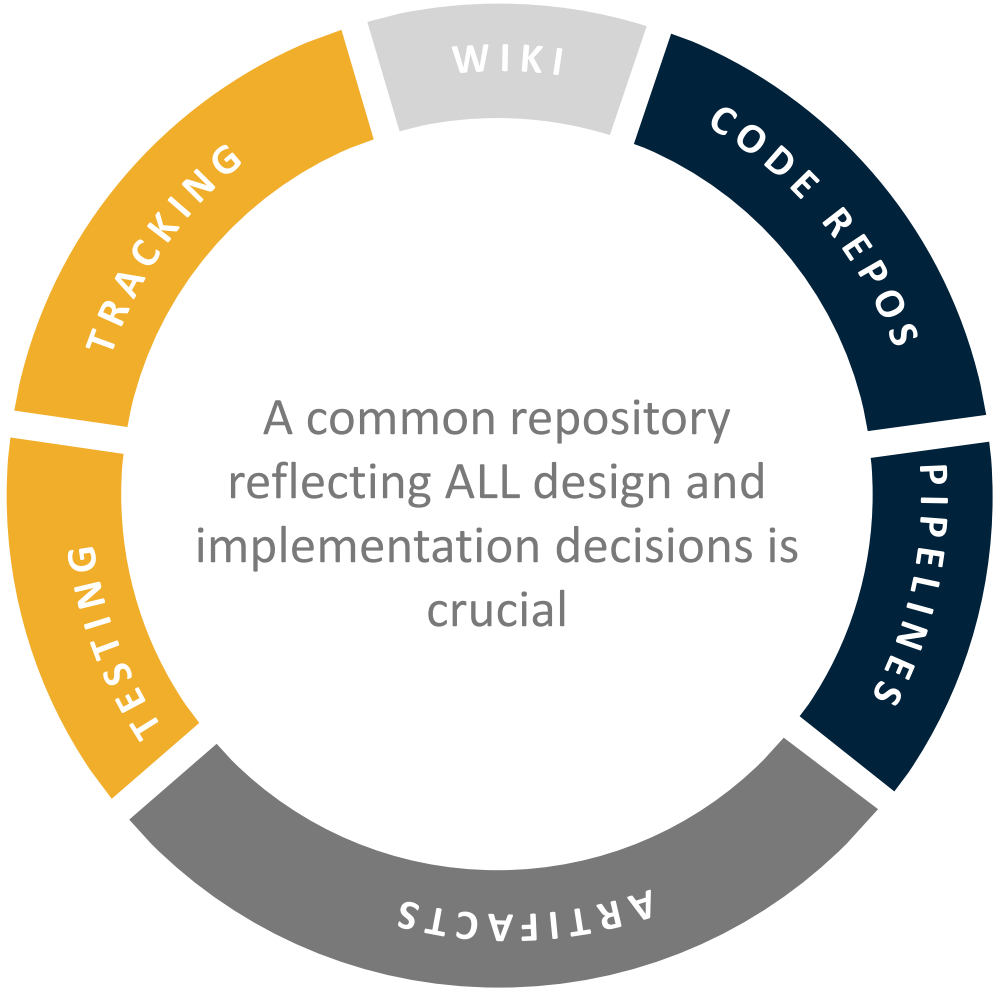
Competency areas (front-end, back-end, DevOps, PMO, testing, etc.) to share ideas and best practices

Principle #6: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

- Face-to-face conversations:
  - Tend to be more honest and open
  - Allow for body language and other non-verbal communication
  - Are spontaneous and facilitated by the team working in a common area

- People need to recognize other team members not in those conversations may also find value in that information
  - A tester will be affected by the outcome of a clarification discussion between a BA and a developer
  - A Scrum Master needs to be aware of a potential blocking issue holding up a developer and the lead
  - Every role needs understanding of a backlog priority change from the Product Owner
  - etc. etc. etc.

- The team requires a Single Source of Truth to function efficiently!



1 **WIKI**
Meeting notes, collaboration outcomes, design, etc.

2 **CODE REPOSITORY**
Source and version control of code libraries

3 **PIPELINES**
Build and release services, support for continuous integration and deployment

4 **ARTIFACTS**
Public and private release packages to be shared within the Pipelines

5 **TESTING**
Managing test cases, execution cycles and metrics

6 **WORK TRACKING**
Boards and other tools for managing user stories, related tasks, defects, technical debt, etc.

Principle #7: Working software is the primary measure of progress

Verification and validation are the very definition of the term "working"

QUESTION

Verification: Are we building the product right?

Validation: Are we building the right product?

OBJECTIVE

Verification: Prove the letter of the law for requirements and design

Validation: Prove suitability of the product from a user and business perspective

ACTIVITIES

Verification: What people think of as testing functional requirements

Validation: What people think of as testing non-functional requirements

Principle #7: Working software is the primary measure of progress

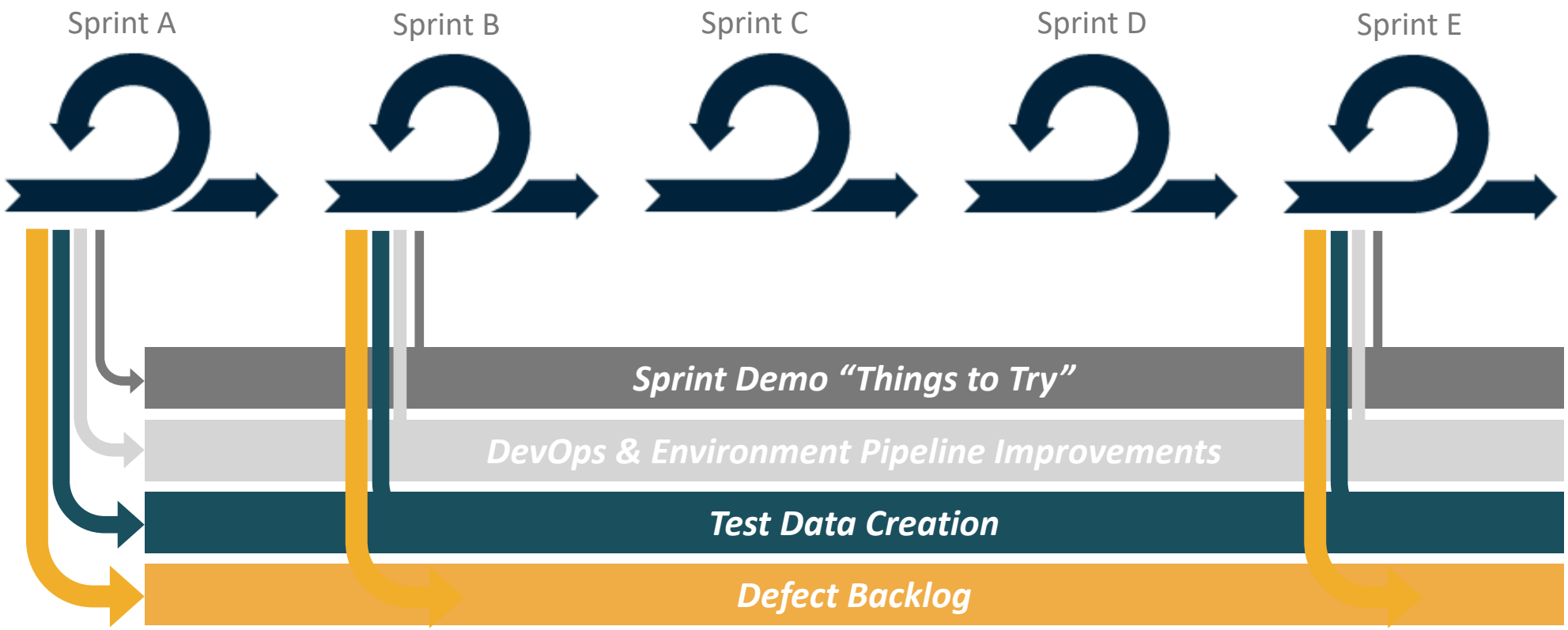
The Definition of Done is the measure for delivery progress, and must include multiple levels of testing

- Functional
  - ***All test cases for story acceptance criteria pass.***
  - All non-functional requirements have been met.
  - UX has approved the implementation against their mockups (if applicable).
  - Browser/Device Support requirements pass testing
- Quality
  - ***Unit tests have been written for new code (min. coverage 80%).***
  - ***100% of unit tests pass for new code.***
  - ***Test cases and defects have been written and documented in the applicable ALM tool.***
  - ***Automation test story has been created for [CurrentSprint + 1].***
  - ***Automation has been completed for all stories in [CurrentSprint – 1].***
  - Peer code review has been completed, unless pair programmed (Reviewer: FirstName LastName).
  - Acceptable defect backlog exists:
    - No open Critical (Severity 1) or High (Severity 2) defects.
    - Fewer than 5 open Medium (Severity 3) defects.
- Adminstrative
  - Build has run successfully.
  - Code artifacts are promoted through a CD pipeline to a separate testing environment.
  - Logging & Monitoring are enabled.
  - The Product Owner has accepted the user story.

Principle #8: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

What point value should the Scrum Master assign to this User Story?

Principle #8 solution: Proper balancing of new development & technical debt

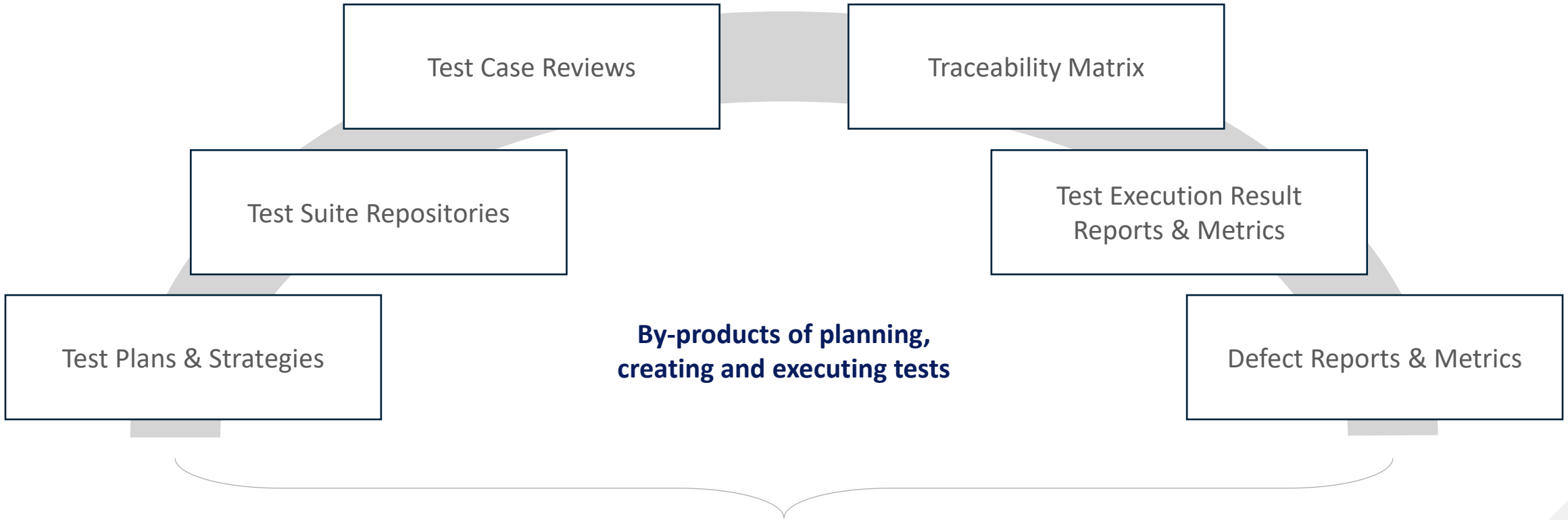


We need to make sure these outputs of each Sprint are given enough priority against "More Stories, More Points!"

Principle #9: Continuous attention to technical excellence and good design enhances agility

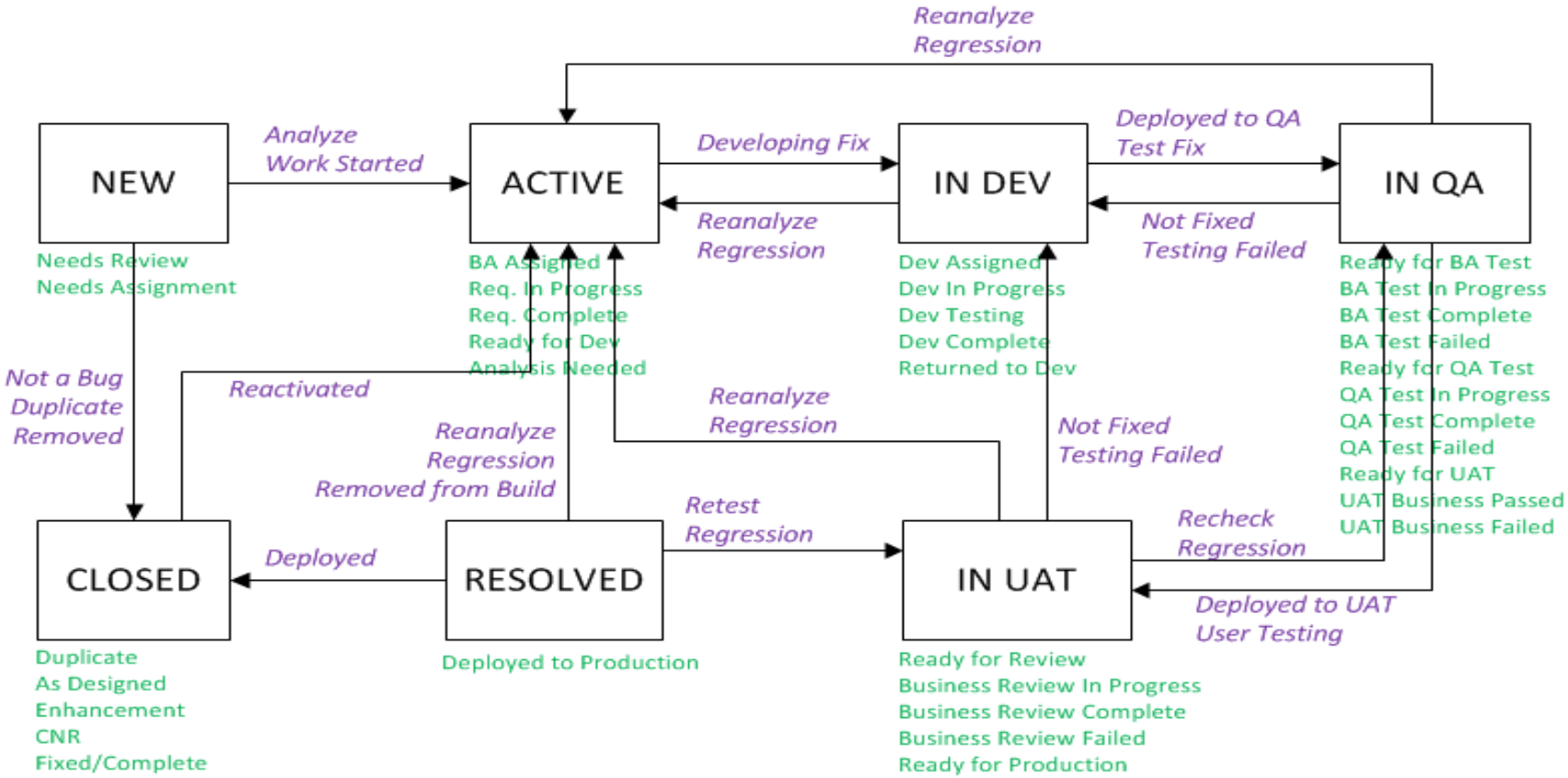
- RPA and AI/ML could be the final nail in the coffin of the manually operating UI-centric tester
- What unique technical value do you add to a project team?
- Stay on top of the latest trends in model-based (scriptless) automation and framework accelerators
  - You don't have to be a Java/C# engineer to contribute to test automation
  - It is vital to be a “full-stack” tester, including data & API services throughout the various cycles

Principle #10: Simplicity – the art of maximizing the amount of work not done – is essential



If your team is not actively using any of these artifacts to improve delivery quality…DON'T DO THEM

Principle #10 solution: Streamline your models and processes



Each one of these states and reasons is a piece of information that can be used to improve…but how can anyone use this?

Principle #11: The best architectures, requirements, and designs emerge from self-organizing teams

- As a software testing professional on an Agile team, what does this mean?
  - You shouldn't have anyone telling you what to do or how to do it
  - You don't have a "boss"
  - You are the proxy "voice of the customer"

- You contribute to the delivery team in every way your current skill set allows and are always working to increase those skills

- Your value must go far beyond "finding bugs"

- Waiting for software to be delivered, then testing it, does not proactively contribute to the success of your team

Principle #11 solution: Scripted v. Exploratory Testing provides additional independent direction and self-guidance within the team

Directed from requirements	Directed from observation and experience
Test cases determined in advance	Test cases determined during execution
Requirement confirmation	Application domain investigation
Predictability and decision making	Adaptability and learning
The script is in control	The tester’s mind is in control
Reading a prepared speech	Having an improvised conversation

Principle #12: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

The feedback in these Sprint Retrospective categories is typically qualitative, not quantitative, even though the latter is far easier to use to identify courses of action and measure improvement

Principle #12 solution: Testers come to Sprint Retro ready with both information to celebrate and actionable items for improvement, backed by test execution and defect metrics.

Being prepared with facts, deliverables, and potential solutions helps drive the team toward quality.

Paul Herzog

Principal Consultant, Technology Practice

pherzog@westmonroepartners.com

Paul's core experience and skills include:

- Manual, Automated and Performance testing technologies
- Test Strategy and Solution Design
- Agile Processes for Quality Improvement
- Software Lifecycle and Testing Assessments and Coaching
- Testing and Project Management process adoption and implementation

INDUSTRY EXPERTISE

- Audit
- Medical
- Banking
- Medical Devices
- E-Commerce
- Non-Profit
- Telecommunication

Paul Herzog is a Principal Consultant in the Technology practice at West Monroe Partners. Paul has over 25 years of experience, specializing in software testing technologies and solutions through successful implementation of the right skills, processes and tools to meet the needs of each project. Paul brings a background in systems engineering, business analysis and Agile Scrum management, providing a holistic approach to software quality throughout the development lifecycle.

Paul's blend of consulting and corporate experience across a wide variety of industries, from health care to defense to banking to gaming, provides him with the flexibility to create unique, focused testing solutions for West Monroe's clients. His contributions cover the entire duration of a project, from test strategy, cycle planning and repository design at the beginning to execution coverage, metrics and reporting dashboards at the end of a project to help validate its overall health.

Prior to joining West Monroe in 2019, Paul was a Test Architect and Practice Manager at SPR Consulting, where he provided software testing guidance to both client projects and test analysts in the Practice. His services included project management, delivery oversight, and "best practices" training to direct SPR reports and client personnel alike. His efforts led to improved tool adoption, requirements verification and traceability, project reporting and process improvement at clients in the accounting, medical, e-commerce and financial sectors, amongst others.

Paul is an active speaker at testing conferences across the country and in the Chicago Quality Assurance Association, where he has spoken on topics related to Agile testing, test data management, user acceptance and process improvement.

RELEVANT EXPERIENCE

- SDLC process assessment and adoption strategies for an international bank
- Technical project management and digital workflow recommendations for a shipping logistics provider
- Test strategy and governance for SharePoint-based audit platform
- Test assessment, coaching and transformation at a leading vendor of computing equipment
- Service virtualization and automation analysis for a medical equipment manufacturer