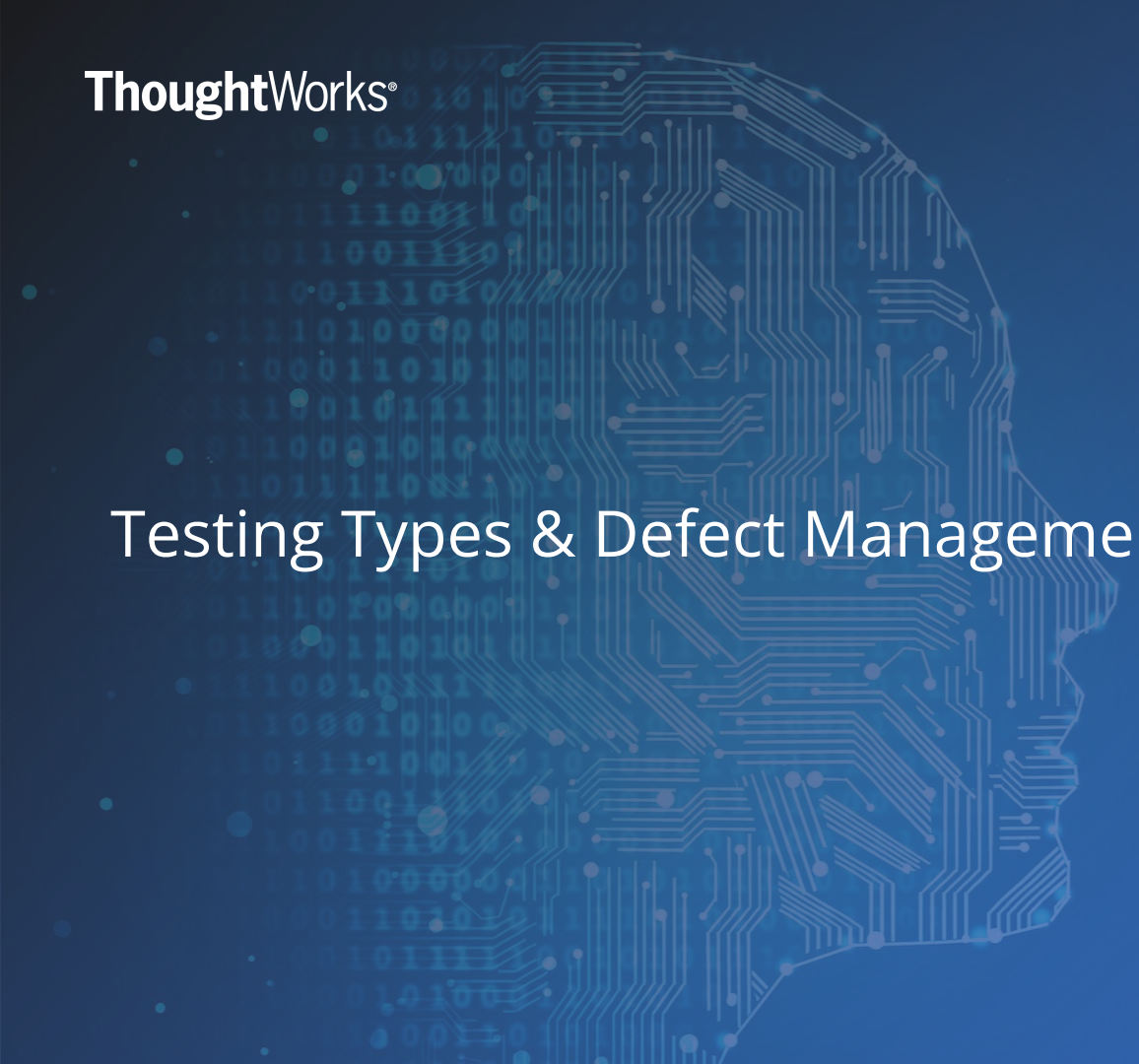
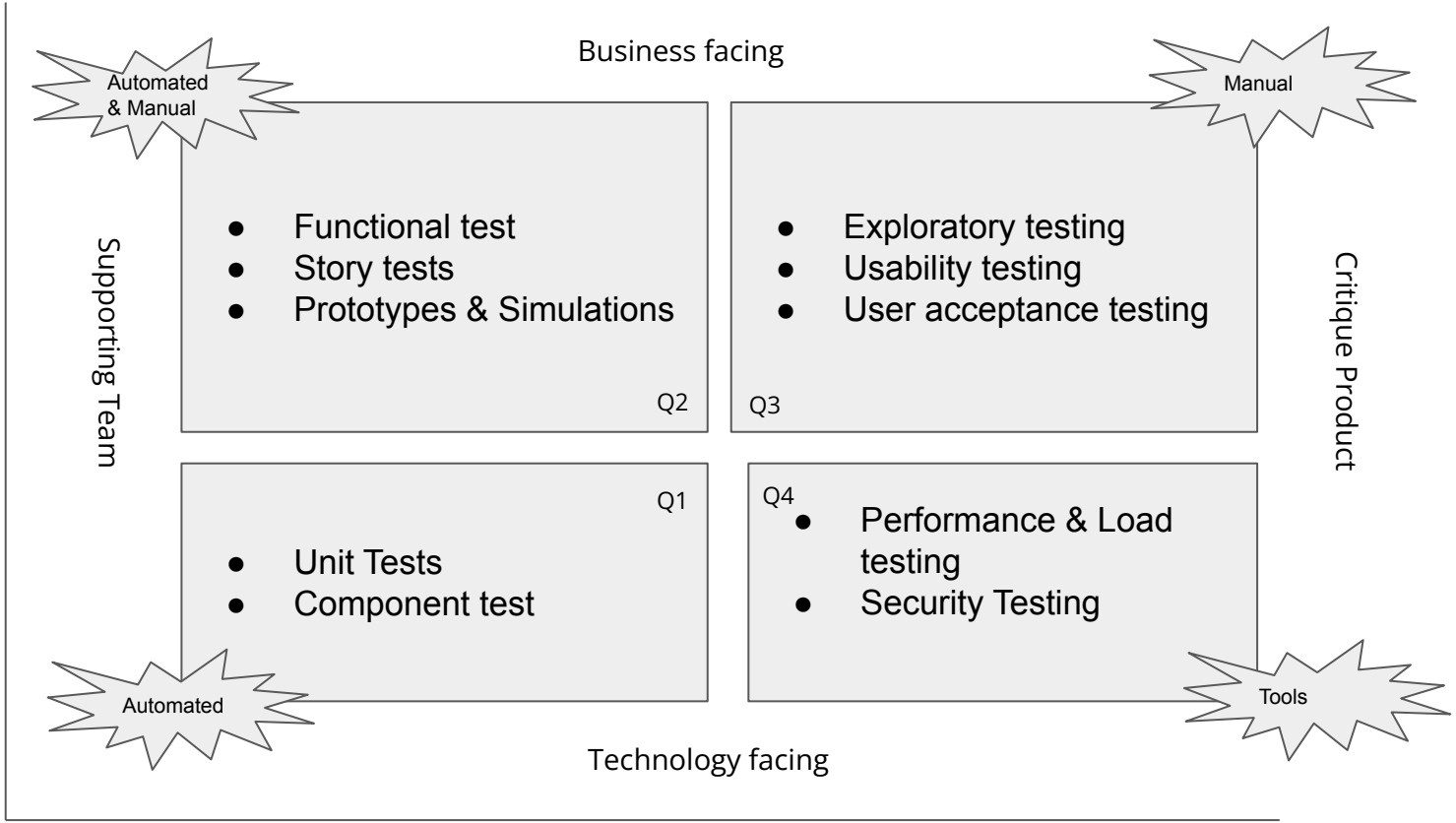
ThoughtWorks®



Testing Types & Defect Management

Agile Testing Quadrants



Technology facing tests + Supporting Team(Q1)

Foundation of Agile testing & development.

- **Ownership**: Primary developers
- **Run**: Automated
- **Purpose**
  - Saves Qas time in finding low level bugs, inturn creating bandwidth for other testing
  - Testability baked into the code
  - Quick feedback to Developer
- **Examples**
  - Unit test
  - Component test etc

Business Facing + Supporting Team(Q2)

Addresses business requirement.

- **Ownership**: Primary QAs
- **Run**: Automated + Manual
- **Purpose**
  - Common language to understand requirements, tells what to code.
  - Helps customer to think of all desired and undesired functionality
  - Help mitigate risk of missing functionality
- **Examples**
  - BDD test/story acceptance test
  - Design/Mock up review
  - Automation: Selenium, Cypress, Appium etc
  - Behaviour Driven development

Business Facing + Critique Product(Q3)

Recreate actual experience and different business scenarios to make experience more realistic.

- **Ownership**: QAs + BAs/PO
- **Run**: Manual
- **Purpose**
  - Are we building the Right product.
  - Explore System under test
  - Uncovering areas of Product where we can have more automation or new requirements
- **Examples**
  - Exploratory testing
  - User acceptance test
  - Beta testing

Technology Facing + Critique Product(Q4)

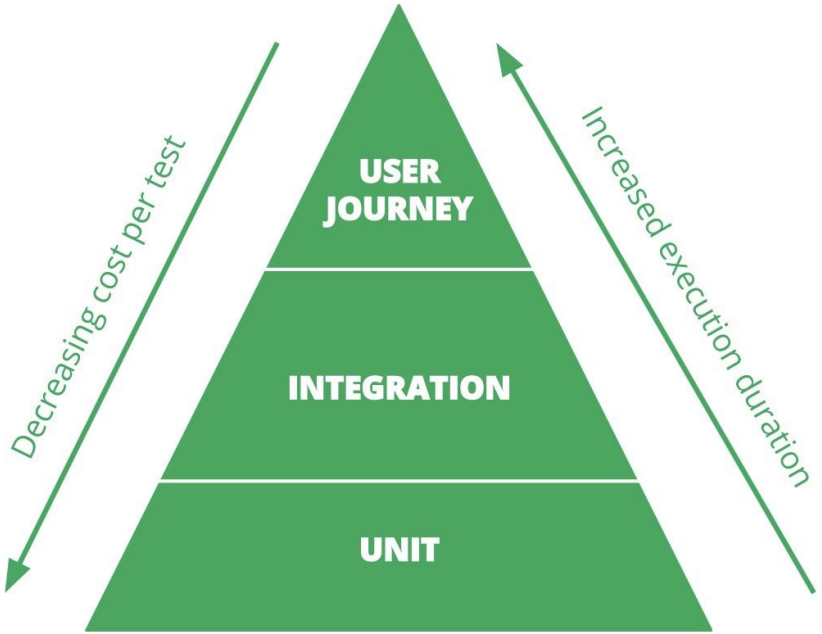
Non functional/Cross functional requirements of the product

- **Ownership**: Specialist group
- **Run**: Automation with Tools
- **Purpose**
  - Are we delivering right business values.
  - Limits of the application and checklist driven
  - Uncovering technical issues.
- **Examples**
  - Performance, Load, Stress, scalability testing
  - Security testing
  - Interoperability, compatibility, recovery testing
  - “Ility” testing based on Domain

Automated Testing pyramid

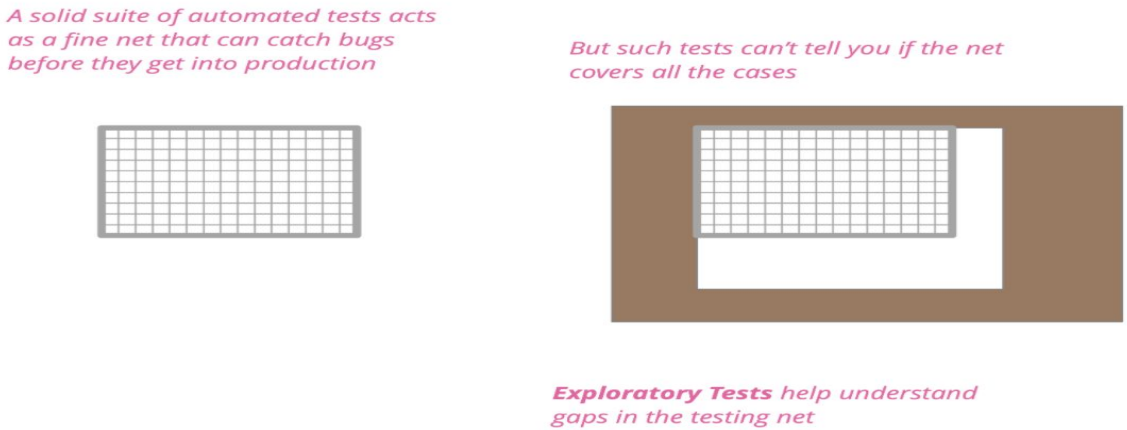
The testing pyramid serves as a guide to where the focus on test automation should be, specifically.

- Focus on the layers from the bottom up first
- More lower level testing reduces the need for more expensive higher level testing.



Exploratory Testing(Q3)

“*Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution”. Reference:Wikipedia*



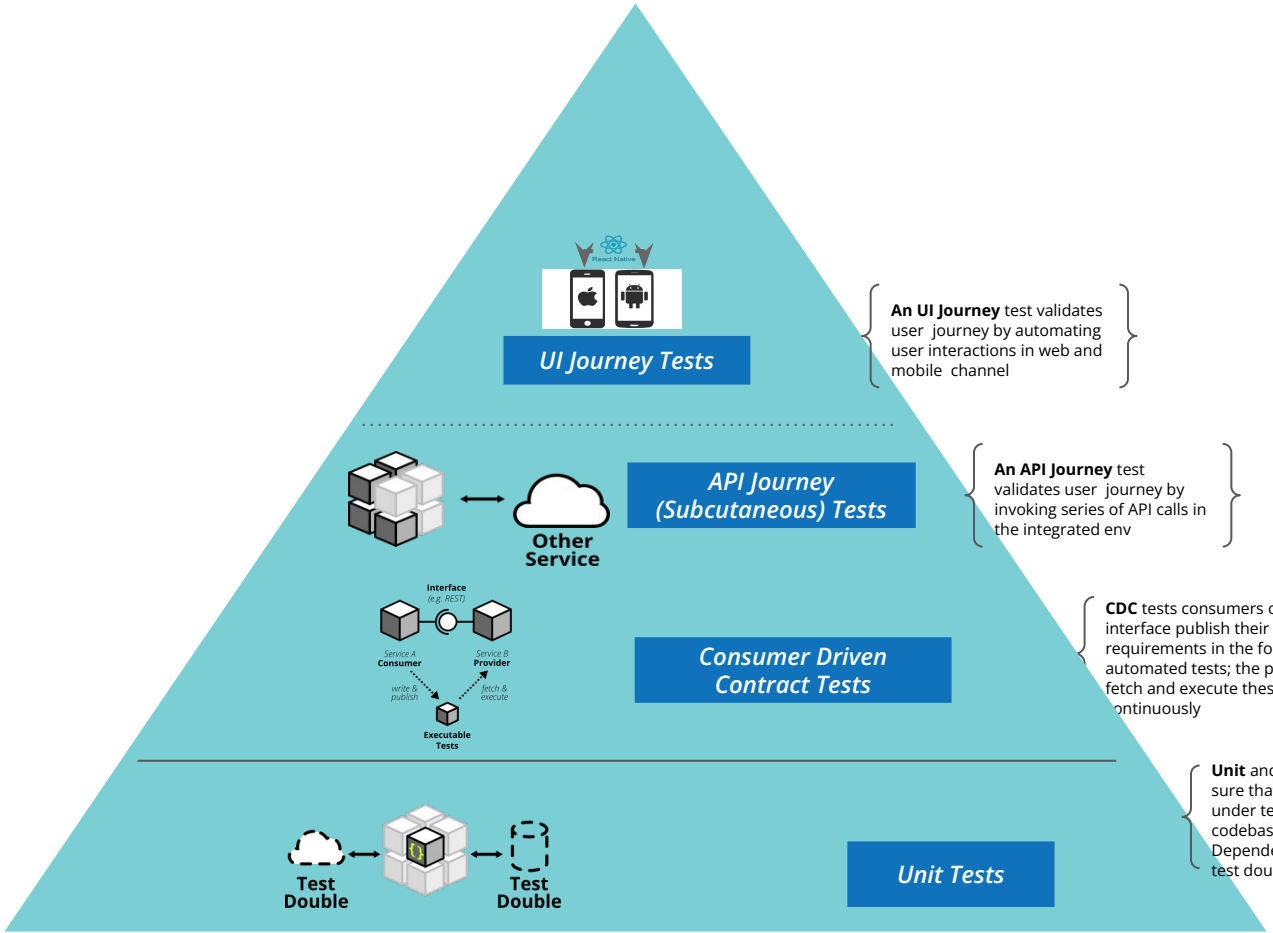
Exploratory Testing Approach

- Make use of Heuristics to guide testing
- Critical thinking
- Careful observation & Competitive analysis
- Session based testing: Choose mission for focused testing
- Add API Testing & Web Service testing.
- Use Automation can be used to setup different test data, perform repetitive process etc

Test Strategy

Slower

$$



An UI Journey test validates user journey by automating user interactions in web and mobile channel

An API Journey test validates user journey by invoking series of API calls in the integrated env

CDC tests consumers of an interface publish their requirements in the form of automated tests; the providers fetch and execute these tests continuously

Unit and **component** tests make sure that a certain unit (your subject under test) or component of your codebase works as intended. Dependencies are stubbed out with test doubles.

BUG BASH

DEPLOY & TEST

Testing Blogs

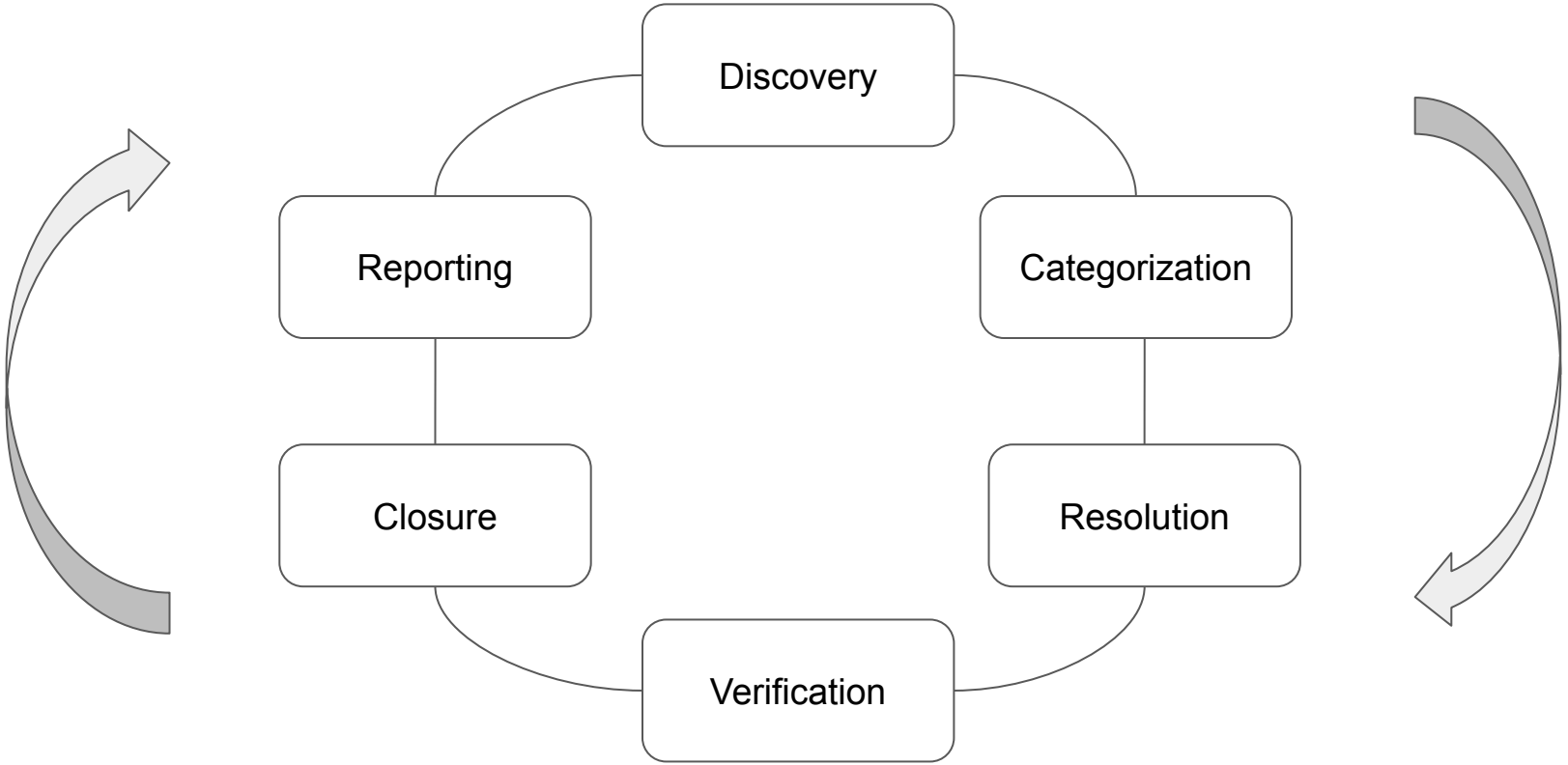
- https://www.ministryoftesting.com/feeds/blogs
- https://blog.testproject.io/

ThoughtWorks®

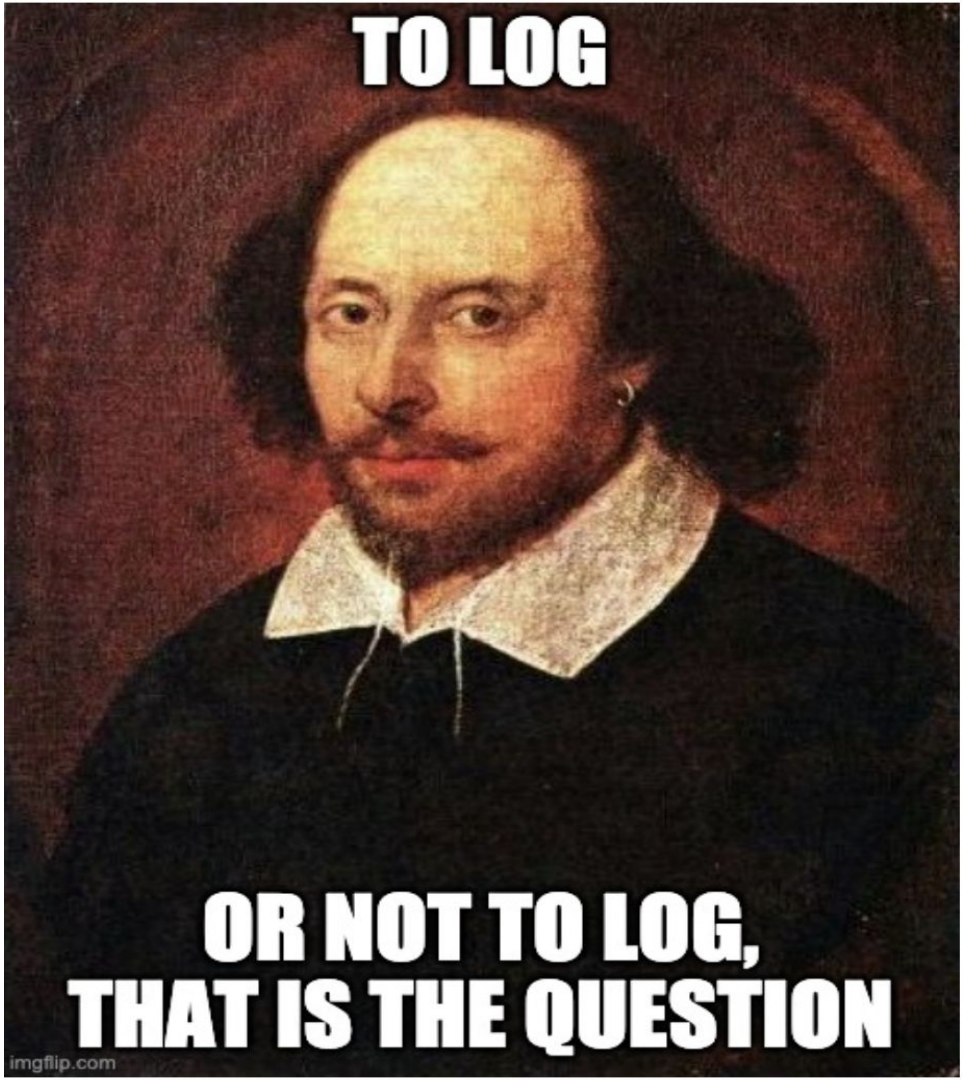
Defect Management

Reporting a defect is an art

What is Defect Management ?



Defects to Log



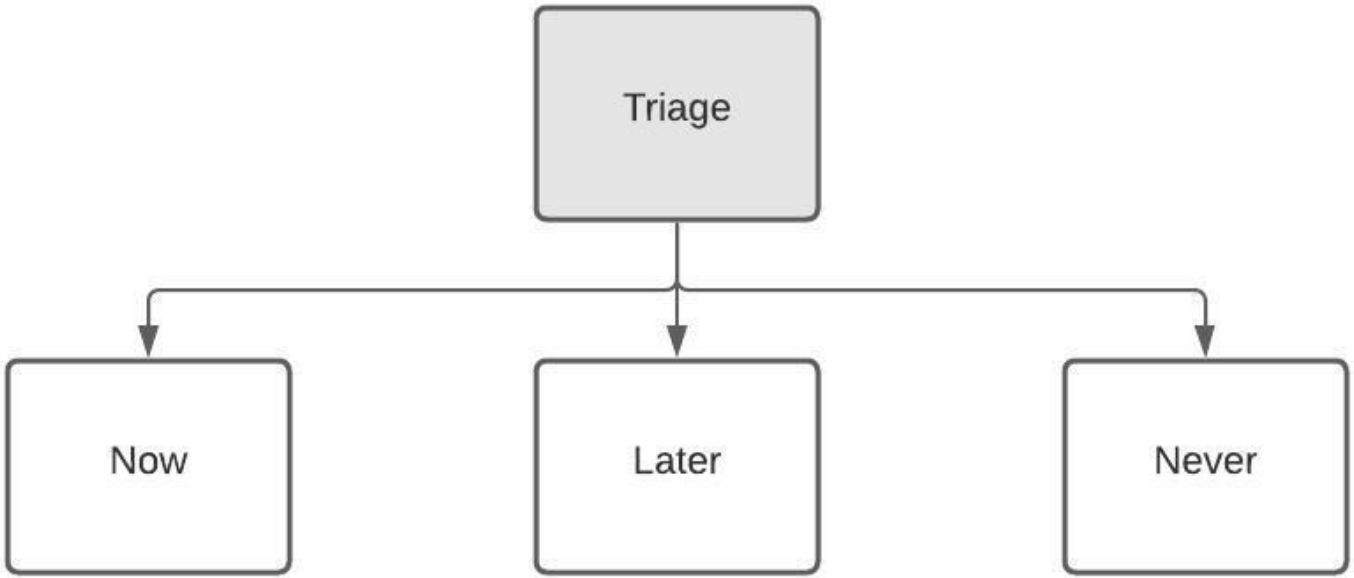
Log

- Post iteration defects
- Production defects
- Any defect which cannot be fixed immediately

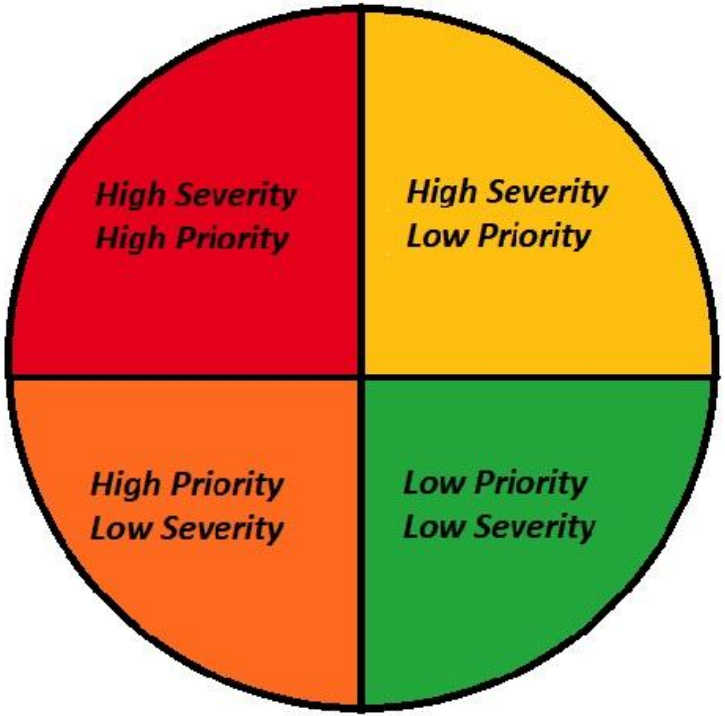
Not Log

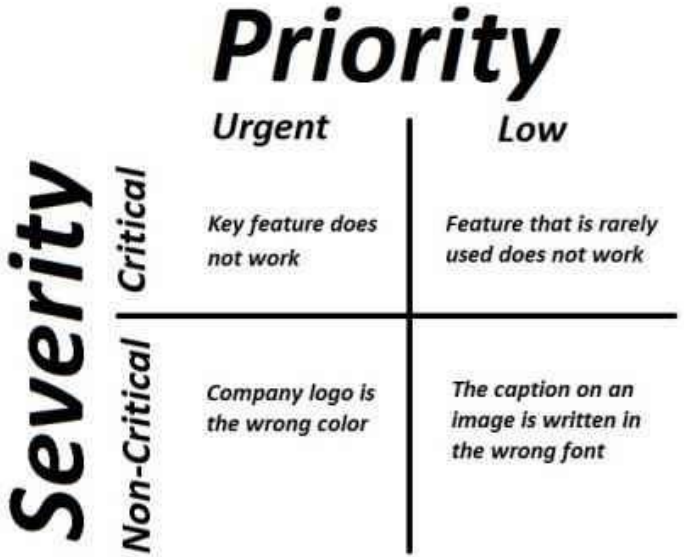
- No need to log all the defects
- Prefer communication with developers over logging defects
- Automated Unit test case failure ,instead act as blocking build
- Flaky tests

When to Fix Defect



Defect Triaging





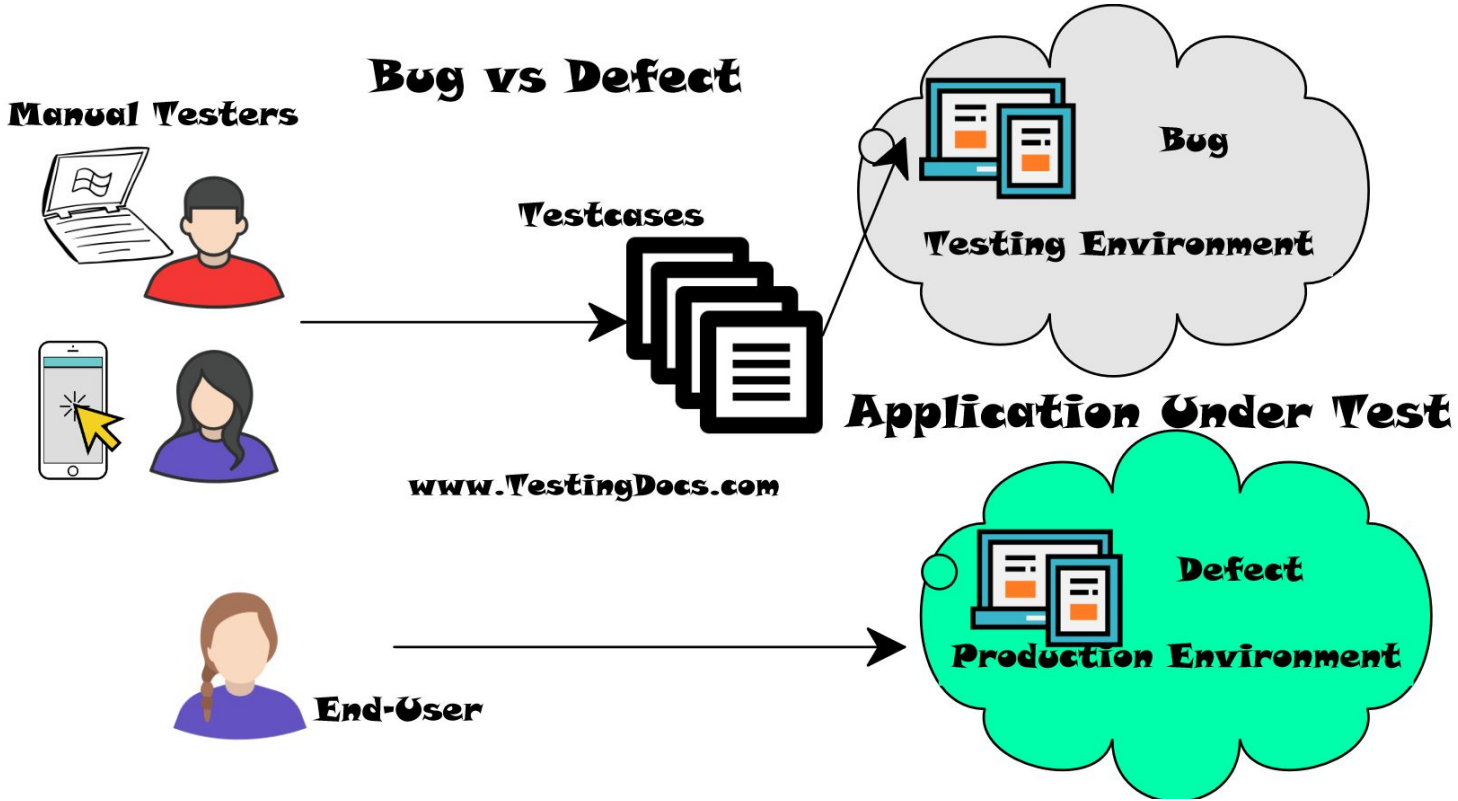
Where to log Defect

- Can chose stickies write observation
- Use color coding
- Set rules for number of open cards

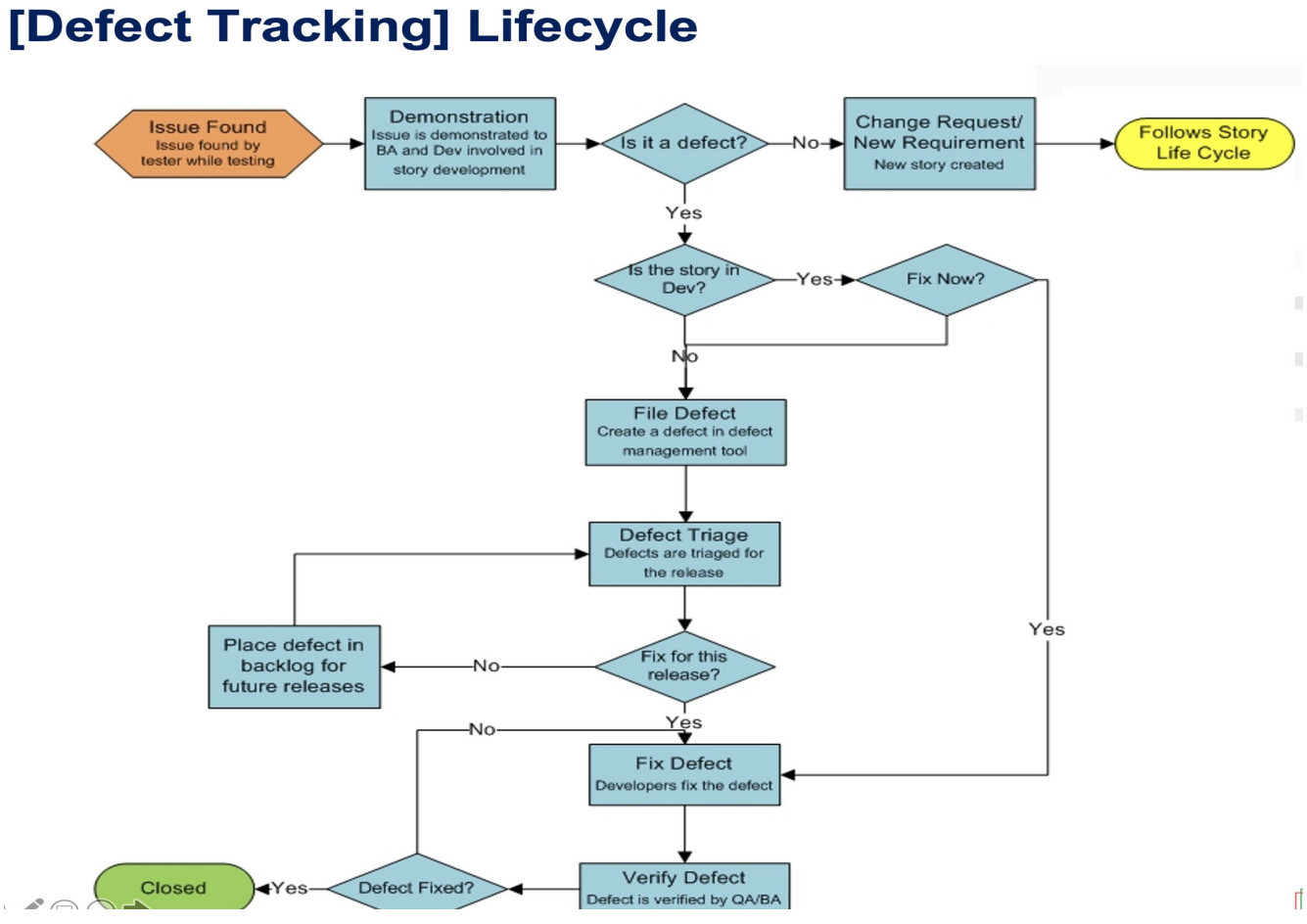
Else

- Defect tracking system like Jira etc

Defect vs Bug

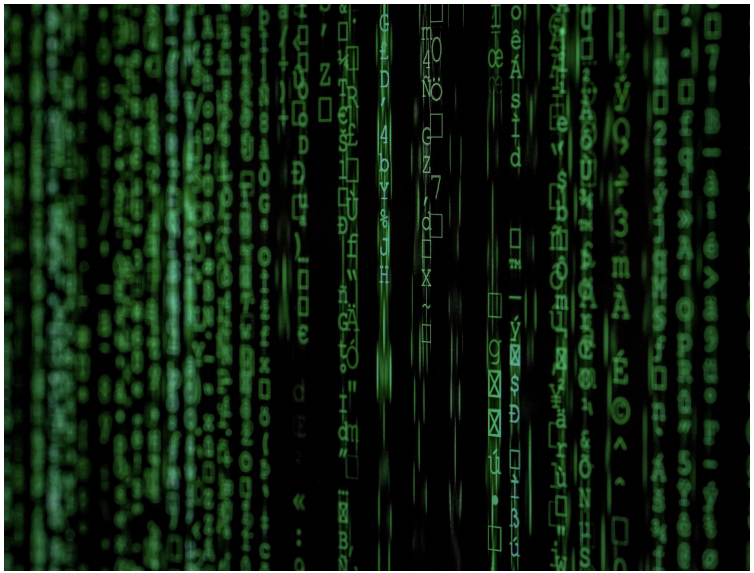


Defect Tracking Life Cycle

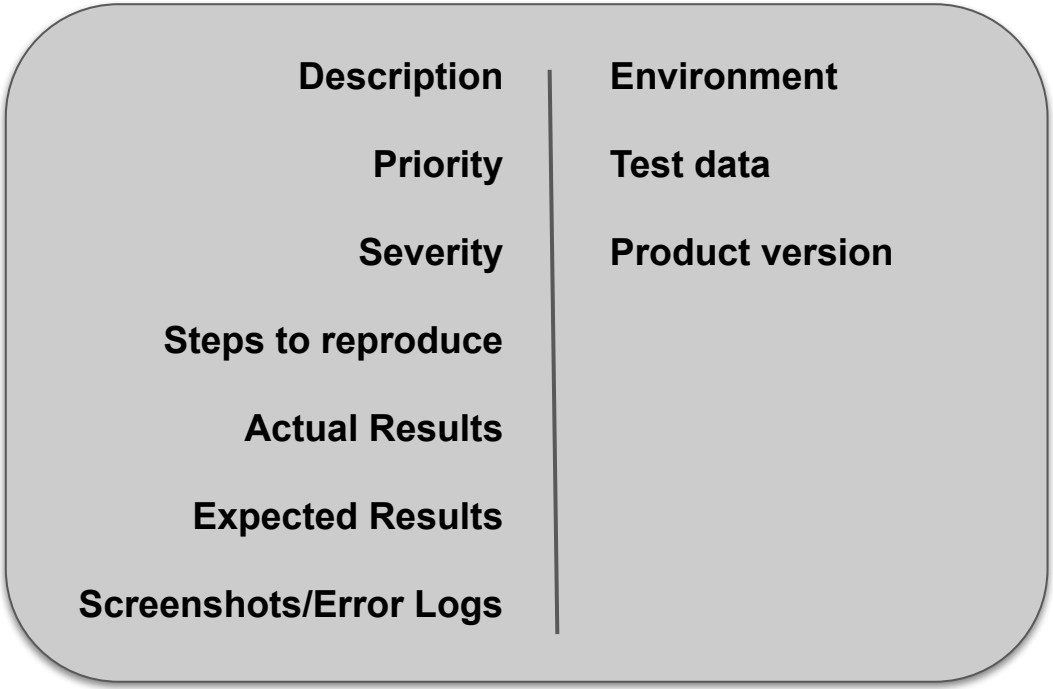


Defect Matrix, Do we need one?

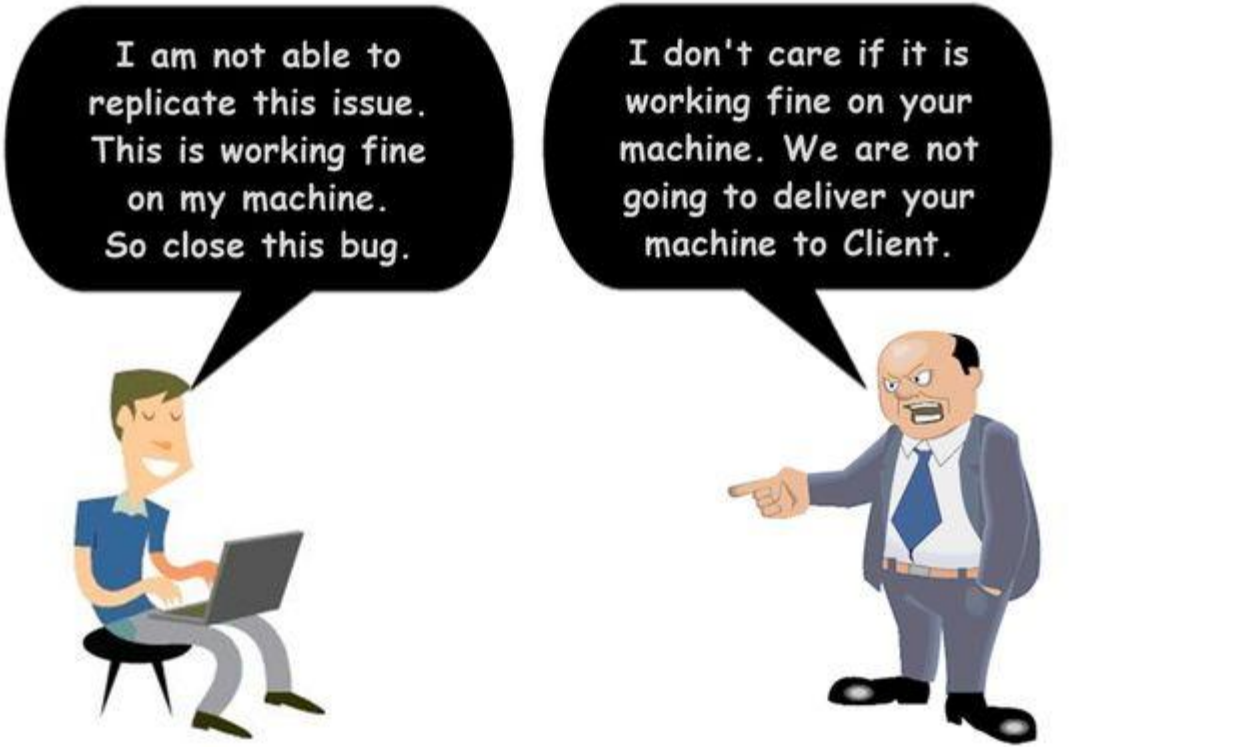
- Stage where the defect is found
  - Design, Development, Testing, UAT, Production
- Defect trend and categorization of the same like UI, Performance etc
- Failing builds
  - Functional defects Vs Automation script failures



A defect


Description
Priority
Severity
Steps to reproduce
Actual Results
Expected Results
Screenshots/Error Logs
Environment
Test data
Product version

Developer vs Tester



DBWebSolutions.com