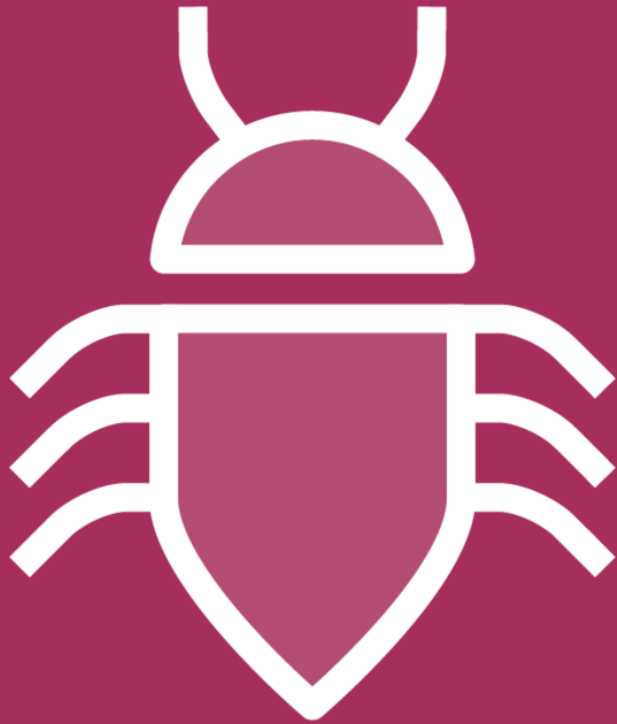




Software Testing



Software Bugs Have a Cost!

They take time to correct, and if they make it to production they have an impact on customers and users.





Infamous Software Defects



The Ariane 5 disaster

- A bug in a function that wasn't even necessary to run on that rocket
- Loss of almost 400 million dollars

The Therac-25 radiation therapy overdoses

- Software errors resulted in the device overdosing some patients
- Contributed to several human fatalities

Software Project Formality

Alistair Cockburn's Project Classification Scale

- Intended to help decide how formal a project's process should be
- Criticality : the worst possible effect of a defect
- Also helps thinking about the importance of testing for a particular endeavor



Loss of Comfort



Loss of Essential or Discretionary Money



Loss of Life

Types of QA Engineers



Investigator



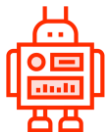
Learner / Analyst



Reporter / Information Radiator

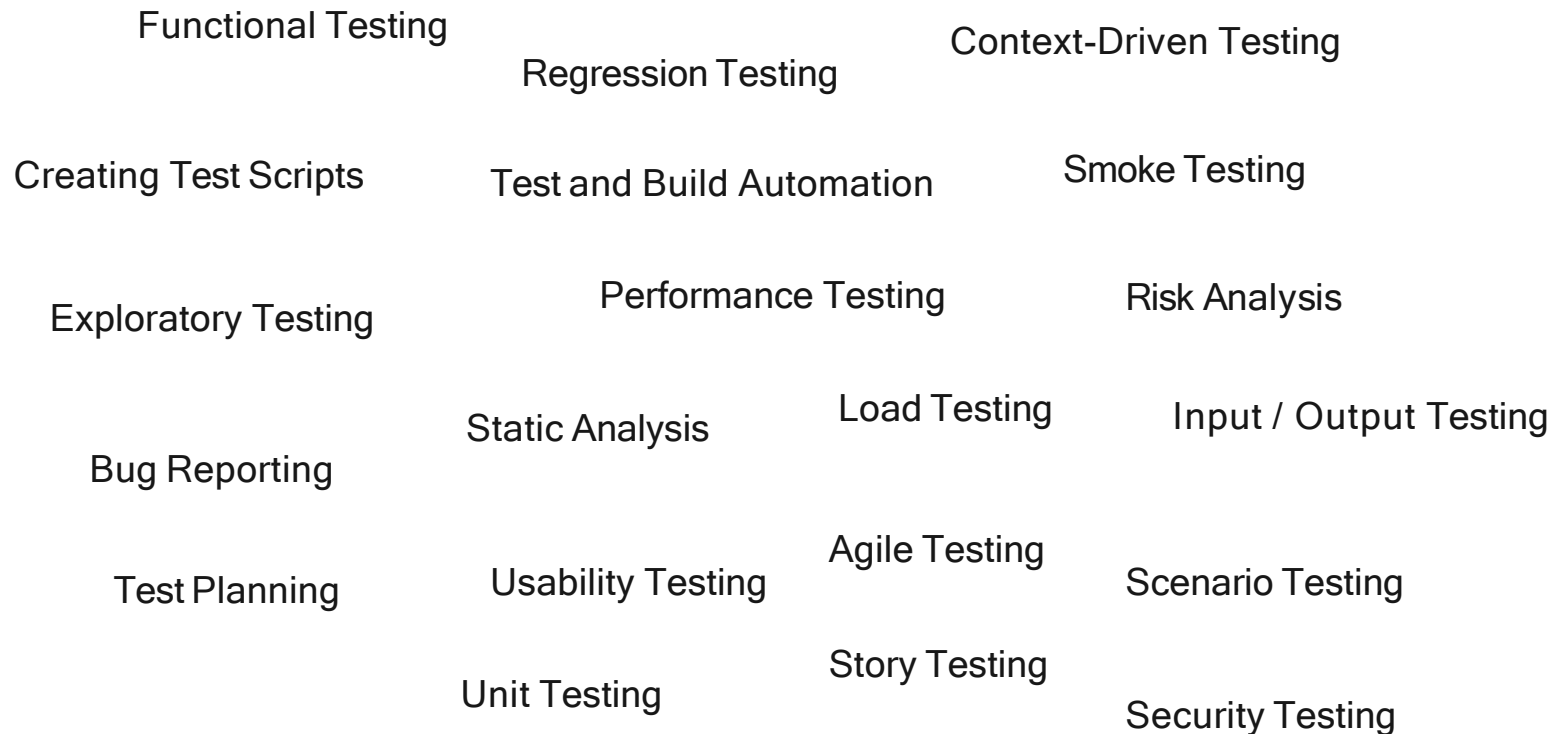


Communicator




Automator

Testing Techniques, Activities and Skills





Common Testing Organizational Models

- 
- Outsourced
 - QA Department with “over the wall” services
 - Testers allocated to project / product teams
 - Testers fully integrated with project / product teams

Tightly Integrated vs. Separated Testers

Integrated

Can support the team during development, dedicated focused resource, some team bias possible

Separate

Post deployment testing focus, "outsider" perspective



The Gatekeeper




QA approves or rejects releases

May be more often found where QA is a separate body from dev teams

Can tend to encourage a more adversarial vs. collaborative relationship between devs and testers

This model is discouraged by the agile testing school of thought



Schools of Testing Thought



Traditional Testing

- Analytical, specifications-driven
- Strong boundaries around testing, gatekeeping possible
- Tendency to emphasize written test scripts and their execution

Context-driven Testing

- Testing techniques and approaches are best driven by context
- Exploratory, investigative testing and learning are emphasized

Agile Testing

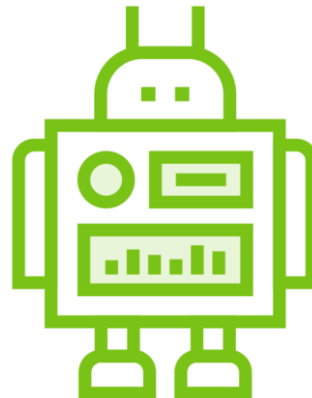
- Emphasizes collaborative, iterative approaches
- Testers support the team in a variety of ways

Vocabulary




“Test” vs. “Check”

Test is a commonly used overloaded word, but some testers prefer to call checking expected results “checking.”



Automated Testing vs. Manual Testing

All testing involves a human using their brain and possibly tools. There are, however, career specializations for testers focused on programming.



Software testing encompasses much more than just defining expected results and writing and automating scripts to check them.

Five-fold Testing System

Kaner, Bach, and Pettichord - 2002

The dimensions of a testing technique:

- *Testers*: Who is testing
- *Coverage*: What is tested
- *Potential Problems*: Why you're testing
- *Activities*: How you're testing
- *Evaluation*: How results are determined

Being aware of these dimensions can help you choose complimentary techniques and find gaps in a testing strategy

Black Box vs. White Box

Black Box Testing

Knowledge of how the system is built does not inform the testing

White Box Testing

Knowledge of how the system is built informs the testing or is even the object of the tests

Test Planning

Define what you're going to test and how

Make choices based on what you've learned about the system and mission goals

More specific, finer-grained objectives than a testing strategy

- a release
- an iteration's work
- A single new feature

Too much test planning documentation can be wasteful – keep it lean unless context requires



Exhaustive Testing is Impossible.

You can't test everything, in every way, in all possible combinations.

So choices must be made, and that means planning.



Example: What to Test?



Written coverage notes



Test what's new



Test for the biggest risks



Test the most important or most commonly used features



Test what's most likely to have been broken

What is a Test Script?

Step by step description of a test with input data, interactions, and expected results

A common form of regression testing

Often written by one person and performed by others

Often recorded in tools, which may also keep execution history

Pros and Cons of Test Scripts

Pros

They help insure an important test can be replicated exactly

They are repeatable by others – you can hand them off

They are a measurable test artifact – how many test scripts do we have, and what do they cover?

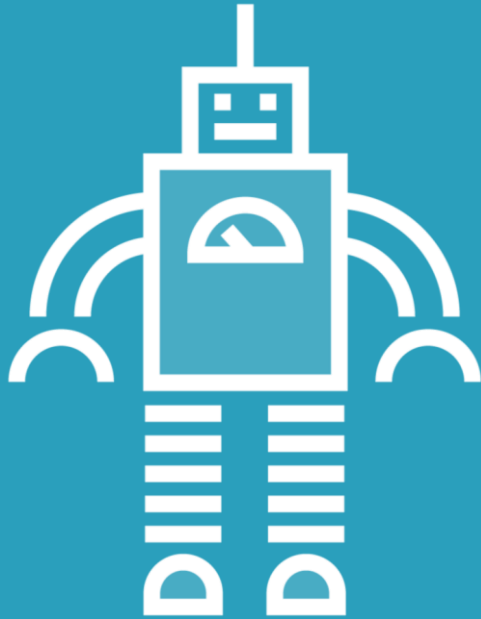
Cons

They accumulate and take increasing time to execute

They are testing the same things in the same ways repeatedly

They can dampen creativity


Changes to the system may invalidate steps of scripts due to their high level of detail



Computers are good at
executing detailed steps
repeatedly.

Humans are good at
creativity.





Improving Script Effectiveness



Allow test executor to be creative by reducing specificity of data inputs where possible

Reduce brittleness by wording tester actions more abstractly

Try higher granularities for test scripts - less use of specific steps

Categorize scripts for traceability and to aid targeting which scripts to execute




What is Exploratory Testing?

An approach to testing that allows testers to evolve and vary what they test based on current objectives


A response to scripted testing

Allows the tester to intuit and explore tangents during the test
– to use their brain

Encompasses multiple activities and techniques



Not equivalent
to ad-hoc testing



There is a purpose for each exploratory test
**Preparation such as test data, interviews,
research may be made before exploratory
testing**





Professional Bug Reporting



Establish what should be recorded

- Information to help developers understand the nature of the problem and recreate it is paramount

Try to avoid an accusatory tone

Explain any potential impacts of the bug if you have insight into them

- You likely don't decide what gets fixed, but you provide the information that helps that decision get made

What's a Bug?

Not all inconsistencies are bugs

- Potential problem with requirements
- Something potentially confusing to a user
- Any other kind of possible quality gap

Your tracking system will still likely require you to report these as “bugs” or “defects”

- Stay professional and factual

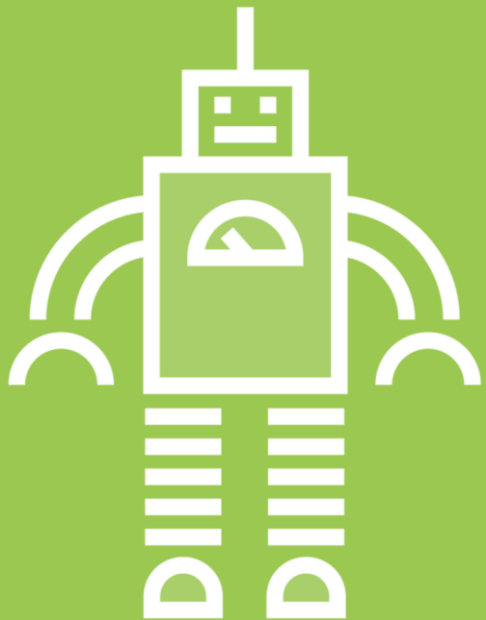
Bugs that can't be recreated are probably still a problem

Report them as informatively as possible

Defect Tracking Systems

Bug tracking systems are common in the industry

- You'll likely use a number of them over the course of your career
- They are often the same tool in which features are tracked



Automated Test

The use of software or a technology tool to check some aspect of a system in a repeatable fashion.




Automated Testing as a Career

Essentially a Developer

**“Software Developer in Test”
is currently the most common
job title for this specialization**

Important for Testers to Understand

**How your testing can be
complimented by tools and
automation is valuable to
explore and learn**



Common Types of Automated Tests



Unit Tests

- Developers author, but can be useful for testers to be aware of

Executable Functional Specifications

- Customers and testers author, but developers wire to code under test

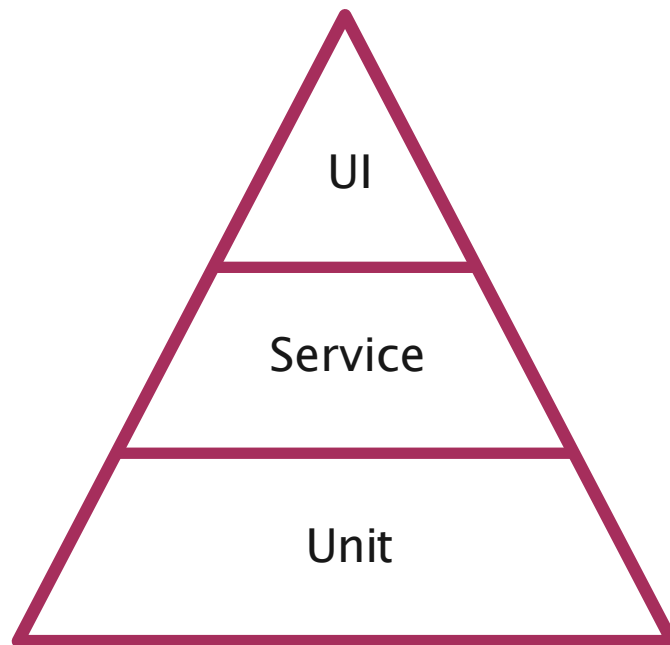
End-End or “UI” Tests


- Executable scripts that exercise the UI of the application in a deployed environment and check results
- Manual regression test scripts are often targeted for this type of automation

The Test Pyramid Strategy

**End-end tests through the UI
can be expensive, slow, and
brittle**

**Service tests check the
business services provided by
the application without going
through the UI**





Automated Test Strategies


Not practical, or desirable, to automate everything

Choose your automation - prioritize

- Things that are important to regression check with every release
- Things that are easy to automate - easy to provide inputs and check the results

Devs can help

- Testers skilled in programming and automation are in demand
- Your developers are also skilled in programming and automation
- The more time a tester is programming, the less they are exploring and thinking



Performance Testing

Performance

- Typically how long it takes to complete a user scenario and how responsive the application is to a user

Load and Stress

- Simulate a heavy load on a system
- Determine how it performs under an expected high load (load), and the upper limits before it becomes unusable (stress)

Scalability

- Examine the ability of the system to scale up or out

Highly specialized tools that can simulate traffic and measure responsiveness are used



Security Testing



Penetration Testing

- Simulated attack on a system to discover exploitable vulnerabilities

Component Vulnerability Analysis

- Scan for known vulnerabilities in open source components used by the application

API Security Testing

- Test the security of the business API for an application

Static Analysis

- Inspect code of the application for insecure practices



Testing in an Agile Context



Early agile works had little to say about professional testing.

The focus of the movement was on other concerns.

Ideas about how professional testers can contribute in this context came later.



Testers as Part of an Agile Team

Opportunities

You can test and provide feedback earlier in the lifecycle

You can influence testability of features as well as the entire system

You gain white-box knowledge of the system to inform testing

You don't have to test an entire system in one "big bang" - you test incrementally as the system evolves

Challenges

Short cycle times for releases

More meetings, conversations, and responsibilities

Loss of end-user, fresh perspective

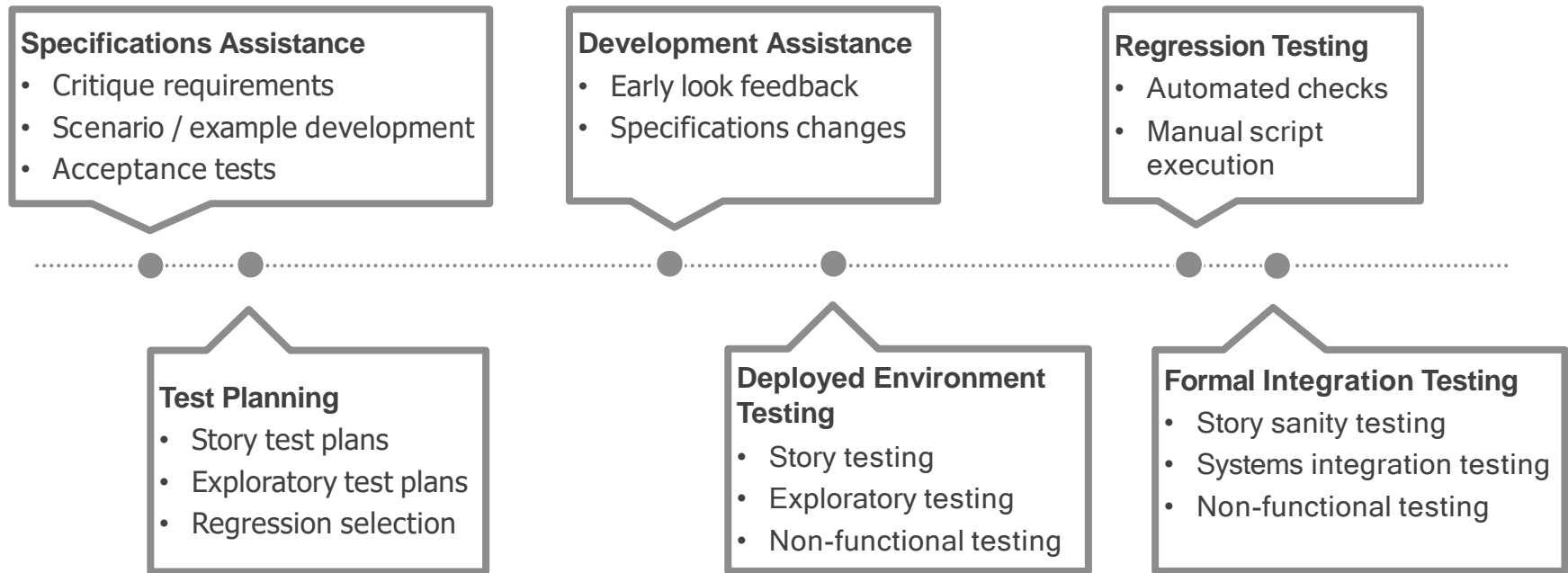
Short increments of time to test means planning and prioritizing.



- How will this story be tested?
- What regression is most important to run before this release?
- Which changes need the most testing?



Testing Activities Over Time





Effective Unit Testing

Prerequisites of Effective Unit Tests

Clean code

Testable design

Context-aware

Understanding of unit testing

Qualities of Effective Unit Tests



Clear and simple

High value

Flexible



Unit of Work

Everything that happens from invoking a public method to it returning the results after it's finished; it's the work done along the path you see the debugger take through your code.



Unit Test

Code that invokes a unit of work within the confines of a project layer while faking external dependencies and validates an assumption about one specific scenario.



Integration Test

Code that invokes a unit of work that crosses project boundaries, uses actual external dependencies, and/or validates many different aspects about the code under test.



Test Doubles

Fakes

Mocks

Stubs



Fake

Object that has working implementations. A fake object implements the same interface as a real object but takes shortcuts to improve performance. Fakes are generally used when we need to test something that depends on an external service or API, and we don't want to make actual calls to that service.



Fake example

An in-memory database is a fake because it implements the same interface as a real database but doesn't use disk storage. It makes it much faster than a real database, but it also means that the data is only persisted in memory and will be lost when the application restarts.



Mock

Object that has predefined behavior. These objects register calls they receive, allowing us to assert how we use them in the code. Unlike fakes, mocks don't have working implementations. Instead, they have pre-programmed expectations about how they will be used in the code.



Mock example

A mock object might be programmed to return a specific value when it is called with certain arguments. Mocks are generally used to test the behavior of our code rather than its output. We can use mocks to verify that our code is calling the dependencies in an expected way.



Stub

Objects that return predefined values. Like mocks, they don't have working implementations. However, unlike mocks, they are not programmed to expect specific calls. Instead, they return values when they are called.



Stub example

A stub might be programmed to always return the same value when called with any arguments. Stubs are generally used to provide data that our code needs to run. This data can be hard-coded or generated dynamically.

Three Part Naming



Unit of work

Initial condition

Expected result




UnitofWork_InitialCondition_ExpectedResult
UserLogIn_WithValidCredentials_RedirectsToHome
UserLogIn_WithBadPassword_ReturnsError
UserLogIn_FailsThreeTimes_LocksOutAccount

Scan test names quickly
Same unit of work sorts together
Read like business rules



Naming tests consistently Creating a
unit test template



```
Public void ReviewSaveTest()
{
    ..snip..

    Assert.IsNotNull(result);

    Assert.AreEqual(3, result.Count);

    Assert.IsNotNull(input.DateUpdated);

    Assert.AreNotEqual(input.DateUpdated,
        input.DateCreated);

    Mock.Assert(() =>
        _reviewDal.Submit(), Occurs.Once());
}
```

◀ Asserting too many expectations makes it harder to find why the test actually failed

```
SavingReview_WhenValid_ReturnsReviews {  
    Assert.IsNotNull(result);  
  
    Assert.AreEqual(3, result.Count);  
}
```

```
SavingReview_WhenValid_SetsDateUpdate {  
    Assert.IsNotNull(input.DateUpdated);  
  
    Assert.AreNotEqual(input.DateUpdated,  
        input.DateCreated);  
}
```

```
SavingReview_WhenValid_CallsReviewSubmit {  
  
    Mock.Assert(() => _reviewDal.Submit(),  
        Occurs.Once());  
}
```

◀ Three separate tests improve precision

High Precision

Test one
expectation per
test

Multiple asserts on
same object can be
OK

Test should point to
precise location of
problem



Literature

Paul C. Jorgensen:

Software Testing - A Craftsman's Approach

John Dooley:

Software Development and Professional Practice (Chapter 14)



Thank You!

