# Chapter 7: Validating the Code

Now that you are working with code, tracking changes against work items, and building the code, you need to squeeze our defects. In chapter 4, we discussed how to configure Azure Boards to shed light on every type of work that must be done for a work item to make its way from an idea to the customer. In this way, you are baking defect detection into every part of your process. You can certainly have code that performs perfectly while doing the wrong thing because of poor design or poor analysis. But this chapter is about ensuring that the code is working properly. Since the code is what the software is built from, you want to ensure that your DevOps process and infrastructure is set up to be able to validate it all comprehensively and quickly. You will likely accumulate a volume of code that is impossible to keep in your head. Significant software systems have so many code files that the only way the code can be validated in a manageable way is to automate most of it and create a process for manual review of just the recent changes. This chapter will span steps that will be automated through the continuous integration build, the first deployed environment, and the pull request.

## Strategy for Defect Detection

DRE

Capers jones article

Static analysis

Testing

Inspection

### Strategy and Execution of Defect Detection

Move test design ahead of coding – make it acceptance criteria

Choose static analysis tools

Define an inspection checklist for pull requests

Inspection checklist for other key steps, such as approving a production deployment – document that the checklist was followed and that it passed.

### Code Validation in the DevOps Pipeline

<diagram of pipeline over view with callouts of where each practice goes>

Walk through of the diagram and when it happens.

### Static Analysis

Best place is in build

VS Code Analysis - .Net Framework example – how to configure it in

.Net Core – not yet available via the command line as of writing, but will be

FxCop

Roslyn analyzers

ReSharper code style analysis

Ndepend – screenshots

SonarQube – screenshots

TSLint – typescript screenshots

HTML web accessibility? – in fully deployed environment?

### Testing

Types of tests:

Reference to microsoft’s L0-2

Table with type of testing and where it runs – including manual testing, load testing, etc.

Order of magnitude difference between each

Tying acceptance criteria to ATDD

### Inspections

What an inspection is

We can’t inspect the full system every time – we are inspecting changes

Changes on branches / pull request

Table of inspector roles: submitter/inspector (approver) – one or multiple inspectors? Your choice

Have an actual checklist – inspector is asserting it meets the checklist – contributes to collective code ownership

Checklist structure

* Code works from simple git pull on the branch – w/ private build – build finishes leaving local environment in state where CTRL+F5 works along with new functionality
* Implements architecture of system
* Implements design decisions called out in the work item/feature
* Conforms to existing norms of the code base
* Includes usage of third party packages specified and introduces no others (caper’s research about defect sources from new packages)
* New code is covered by right balance of tests
* All test scenarios in acceptance criteria of work item implemented as full system L2 tests
* Logging is complete
* Performance considerations
* Security considerations
* Code is scannable – factored and named so that it is self-documenting and screams what it does.

## Implementing Defect Detection

Each of the chosen strategies above.

### Static Analysis

Analyzers, link to docs – show it running in .Net Framework

Show it running .Net Core – what packages to include – link to docs.

### Testing

Show unit tests

Show L1/integration test example – tip – clear database before each test – helper class to do this.

Show L2, full system acceptance test – link to selenium documentation

Show how acceptance tests are packaged and deployed.

### Inspections

Walk through a pull request getting inspected and how to run through the checklist – call something out and then finally let it pass inspection.