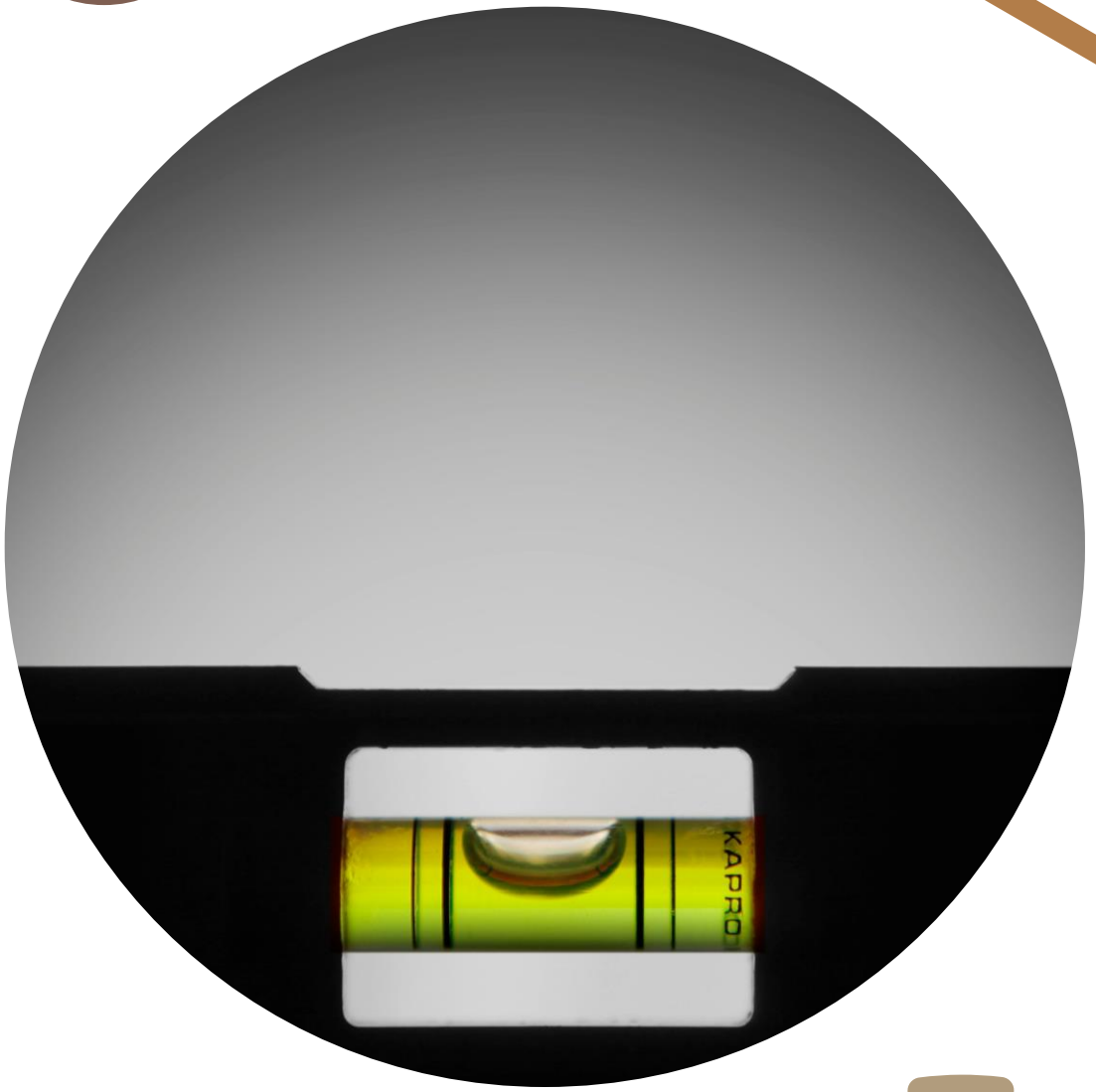


Testing Your Tests

Mutation Testing in C# with Stryker

Daniel Ward

Slides and repo at daninacan.com

https://www.meetup.com/sadnug/

https://www.meetup.com/austin-net-user-group/

https://devsanantonio.com

Independent. Innovative. And always open source.

dotnetfoundation.org

.NET Foundation Virtual User Group

Find virtual .NET events happening around the globe!

Meetup

Part of **.NET Foundation – 380 groups**

.NET Virtual User Group

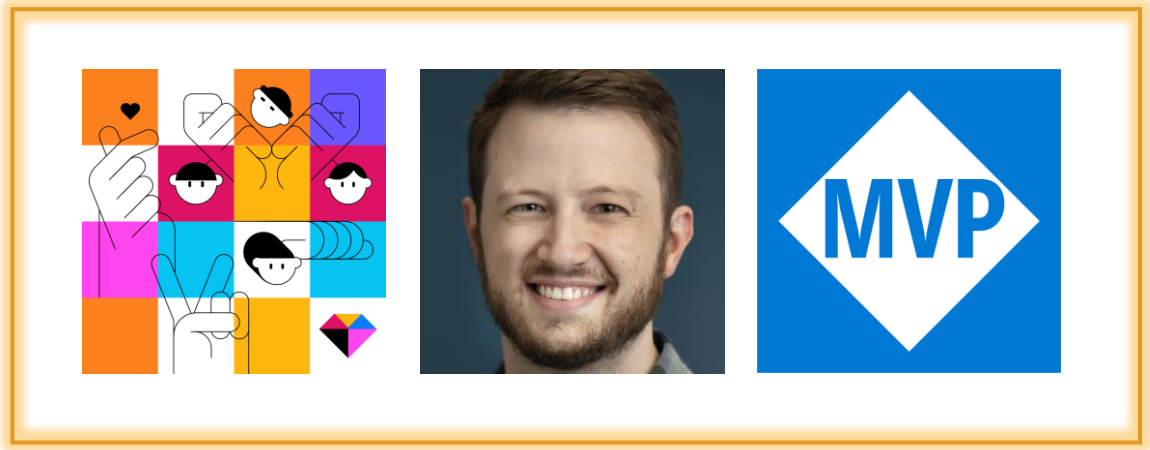
Seattle, WA

2,710 members · Public group

Organized by **James Montemagno** and **14 others**

meetup.com/dotnet-virtual-user-group

Me



- daninacan.com
- daniel-ward-dev
- danielwarddev.bsky.social
- @danielwarddev

LEAN TECHNIQUES

https://leantechniques.com

Slides and repo at daninacan.com

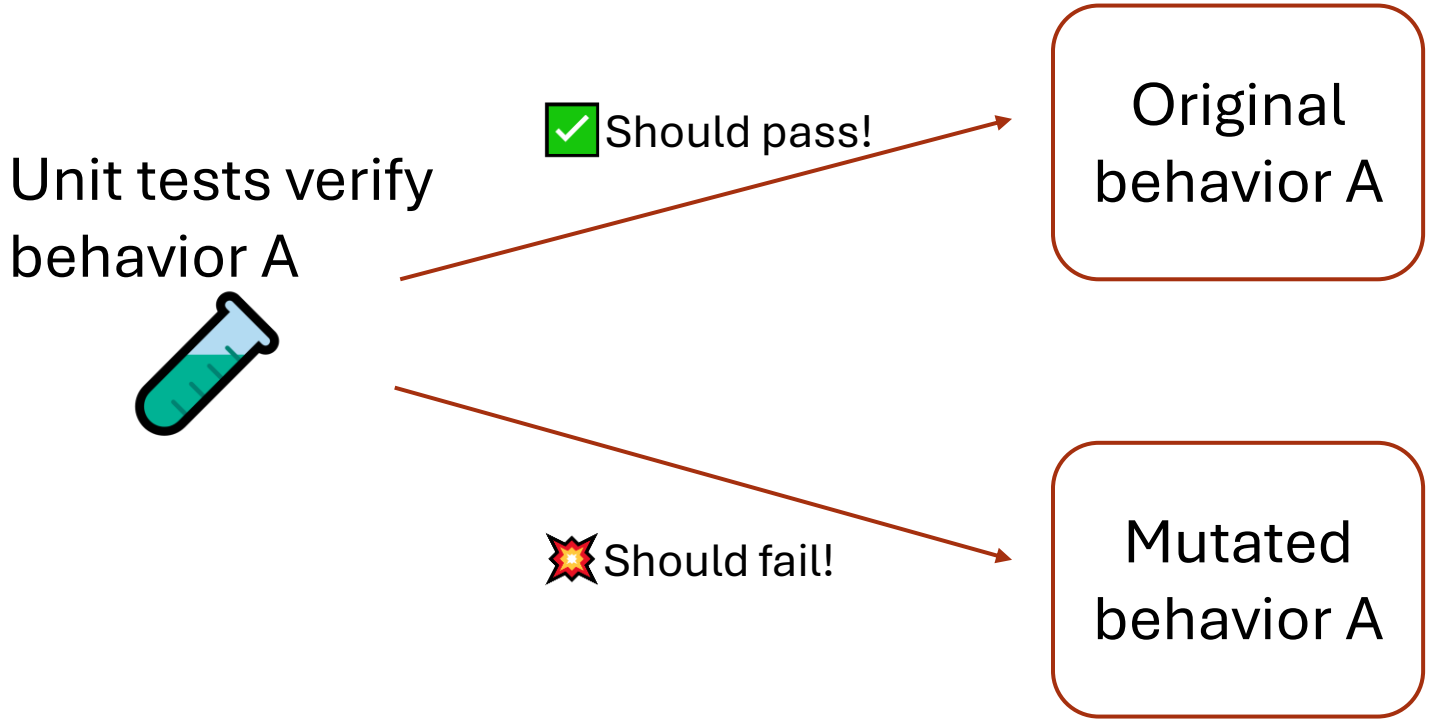
Outline

- Mutation testing
  - What
  - Why
- Installing Stryker
- Coding demo – how Stryker works
- Bonus info!
  - Mutations
  - In a deployment pipeline
  - Baselines
  - More!

Slides and repo at daninacan.com

What is mutation testing?

- A way to test your tests
- Slightly changes your code (“mutants”)
- Then runs your tests – they should now fail! (“killing the mutant”)



Mutation examples

- + ➡ -
- != ➡ ==
- && ➡ ||
- Max() ➡ Min()
- “my string value” ➡ “”

Under the covers

- You don’t see this part

```
if(Environment.GetEnvironmentVariable("ActiveMutation") == "1") {
  i--; // mutated code
} else {
  i++; // original code
}
```

- Not all mutations can be compiled!

```
if (Environment.GetEnvironmentVariable("ActiveMutation") == "1") {
  return "hello " - "world"; // mutated code
} else {
  return "hello " + "world"; // original code
}
```

Goal

Kill all the mutants!

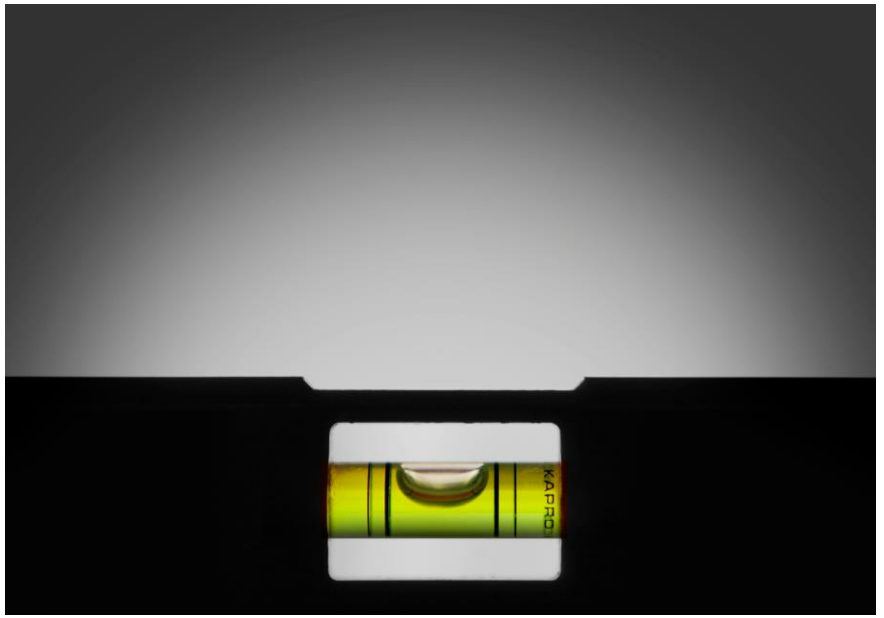
Why do I need to test my tests?

- Tests are good, but how do you know they're working?
- Easy to write bad test cases
  - Or good test cases but miss some
- Code coverage doesn't mean effective tests
- Avoid false confidence
- Write better tests over time

Keep your tests on the level

Mutation testing is based on 2 things:

1. **Competent programmer hypothesis** – Competent programmers write programs that are close to being correct
2. **Coupling effect** – Simple faults can cascade (couple) to create other faults



Stryker

- For C#, Javascript/Typescript, Scala
- .NET 8+ required
  - Just installed, app doesn't need to target it
  - Core 1.1+, Framework 4.5+, Standard 1.3+
- Free and open source

For someone who hates mutants... you certainly keep some strange company.
- *Professor X*
Oh, they serve their purpose... as long as they can be controlled.
- *William Stryker*

Installing Stryker

- Stryker is a .NET tool (NuGet package + console app)
- Global install
  - dotnet tool install -g dotnet-stryker
- ⭐ Local Install
  - dotnet new tool-manifest
  - dotnet tool install dotnet-stryker

Coding demo

Slides and repo at daninacan.com

In a deployment pipeline

- Use either the project-level install or --tool-path

```yaml
- task: DotNetCoreCLI@2
    displayName: 'Install dotnet-stryker'
    inputs:
      command: custom
      custom: tool
      arguments: install dotnet-stryker --tool-path $(Agent.BuildDirectory)/tools
```

```yaml
- task: Powershell@2
    displayName: 'Run dotnet-stryker'
    inputs:
      workingDirectory: <test-project-folder-here>
      targetType: 'inline'
      pwsh: true
      script: $(Agent.BuildDirectory)/tools/dotnet-stryker
```

- Minimum mutation score requirement with --break-at <#>

Reporters

- HTML, progress, cleartext, cleartext tree, dots, JSON, markdown
- Dashboard
  - Requires API key
  - https://dashboard.stryker-mutator.io/

Using with-baseline and since

- Reduces mutation testing time
- Can use --since:<committish> to only mutate only code changes since that point
- Can use --with-baseline:<committish> to additionally save the report to a storage location
  - Same as since in that it only runs changed mutants, but includes a full report

Stryker mutations

- https://stryker-mutator.io/docs/stryker-net/mutations/

Ignoring mutations

- Ignore specific kinds of mutations
- Ignore files
- Ignore method calls

```
"stryker-config": {
    "ignore-mutations": [
        "string"
    ]
}
```

```
"stryker-config": {
    "mutate": ["!**/*.Generated.cs"]
}
```

```
"stryker-config": {
    "ignore-methods": [
        "*Log", // Ignores all methods ending with Log
        "Console.Write*", // Ignores all methods starting with Write in the class Console
        "*Exception.ctor" // Ignores all exception constructors
    ]
}
```

Slides and repo at daninacan.com

SonarQube

- Integrates with SonarQube
- jq filter to convert a JSON mutation testing report to SonarQube

Suggested practices

- Slow feedback – probably best done in pipeline
- Exclude files, classes, third-party code, etc. as needed
  - Doesn’t need to be mutated (e.g. logging statements, Exceptions)
  - Large project that is adding Stryker
- Take time to learn foundational testing knowledge first
- Can help you learn to write better tests

Recap

- Mutation testing – what it is, why to do it
- How to install + use Stryker – coding demo
- Using Stryker in a pipeline
- Other Stryker features
  - Reporters
  - Different mutations + ignoring
  - Reduce mutation testing time
  - SonarQube integration

Slides and repo at daninacan.com

Thank you!

- daninacan.com
- daniel-ward-dev
- danielwarddev.bsky.social
- @danielwarddev

Slides and repo at daninacan.com