Automatically testable apps in distributed environment

or how to test micro services (and more)

Piotr Litwinski

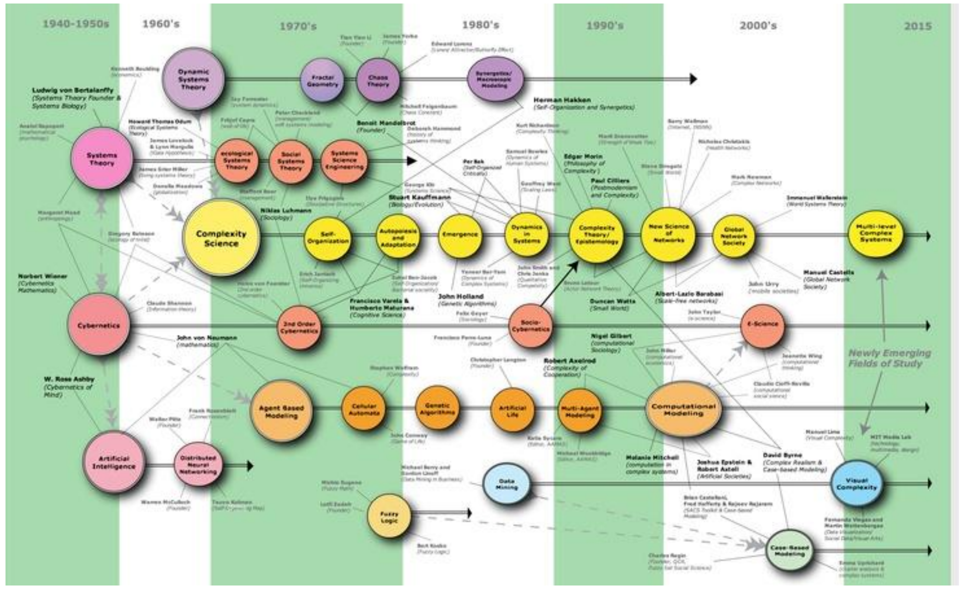
Software Engineer, Zartis

github: https://github.com/plitwinski

Agenda

- What’s the problem?
- Types of tests
- What tools to use?
- How to write automated tests?
- How to test automated tests?
- Summary
- Pros and cons

What is the problem?


1940-1950s
1960's
1970's
1980's
1990's
2000's
2015
Dynamic Systems Theory
Fractal Geometry
Chaos Theory
Synergetics/ Macroscopic Modeling
Ludwig von Bertalanffy (Systems Theory Founder & Systems Biology)
Systems Theory
ecological Systems Theory
Social Systems Theory
Systems Science Engineering
Benoit Mandelbrot (Founder)
Herman Hakken (Self-Organization and Synergetics)
Edgar Morin (Philosophy of Complexity)
Paul Cilliers (Postmodernism and Complexity)
Stuart Kauffmann (Biology/Evolution)
Niklas Luhmann (Sociology)
Complexity Science
Self-Organization
Autopoiesis and Adaptation
Emergence
Dynamics in Systems
Complexity Theory/ Epistemology
New Science of Networks
Global Network Society
Multi-level Complex Systems
Norbert Wiener (Cybernetics Mathematics)
Cybernetics
2nd Order Cybernetics
Francisco Varela & Humberto Maturana (Cognitive Science)
John Holland (Genetic Algorithms)
Socio-Cybernetics
Duncan Watts (Small World)
Albert-Lazlo Barabasi (Scale-free networks)
Manuel Castells (Global Network Society)
E-Science
John von Neumann (mathematics)
Nigel Gilbert (computational Sociology)
Robert Axelrod (Complexity of Cooperation)
Newly Emerging Fields of Study
W. Ross Ashby (Cybernetics of Mind)
Agent Based Modeling
Cellular Automata
Genetic Algorithms
Artificial Life
Multi-Agent Modeling
Computational Modeling
Artificial Intelligence
Distributed Neural Networking
Melanie Mitchell (computation in complex systems)
Joshua Epstein & Robert Axtell (Artificial Societies)
David Byrne (Complex Realism & Case-based Modeling)
Data Mining
Visual Complexity
Fuzzy Logic
Brian Castellani, Fred Hafferty & Rajeev Rajaram (SACS Toolkit & Case-based Modeling)
Case-Based Modeling

Types of tests

Build time tests

- Unit tests
- Contract tests
- Service/Component tests (in-memory integration tests)
- Classic integration tests

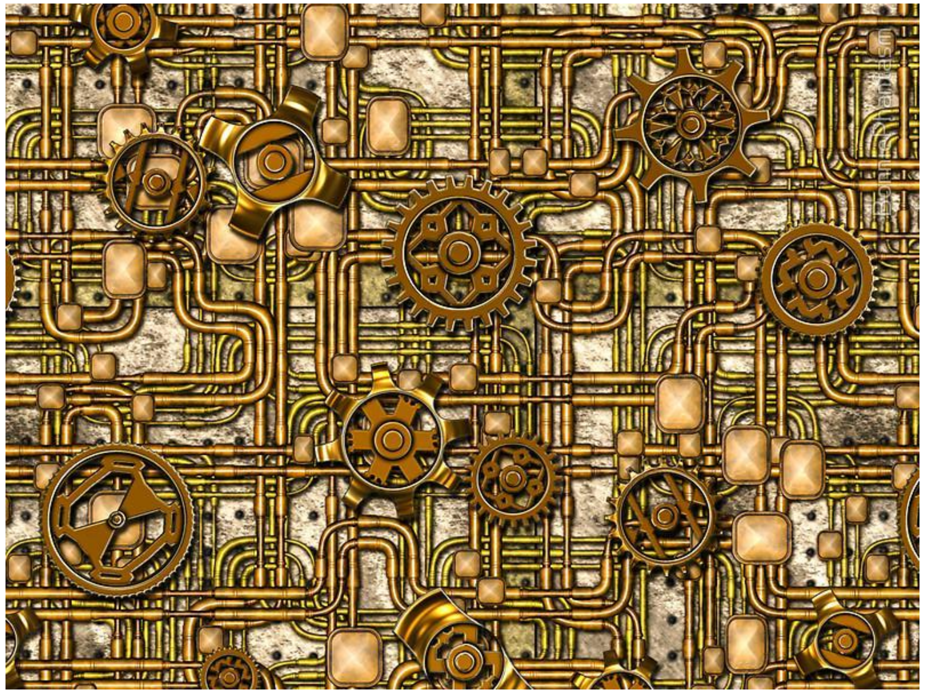
Environmental tests

- Deployed
- E2E
- Exploratory

System tests

- Performance
- Resilience / Availability

What tools to use?



Developer tools

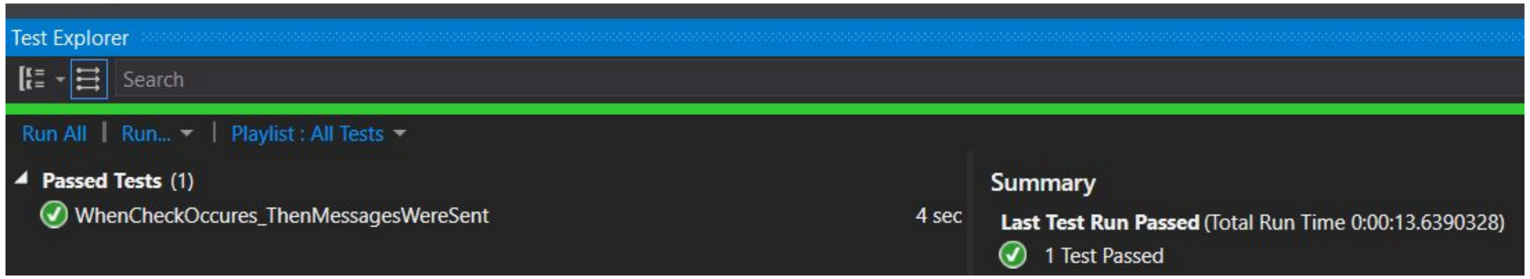
- In-process tools
  - Stubs & mocks
  - Coarse grained „unit” tests
  - Well defined contracts
  - In-memory testing frameworks (np.: in-memory databases, test server)
- Machine level tools
  - Contract verification
  - Webdriver + (Headless) browsers
  - Local counterparts (np.: local sql, json server, etc.)
  - Easily distributable counterparts (e.g. MySQL hosted on docker)

Devops tools

- Tools to use after successful deployment to given environment
  - Deployed agent runs deployed tests in the environment
  - App runs it’s own diagnostics at startup (fails to start in case of errors)

Developer tools – examples 1, 2

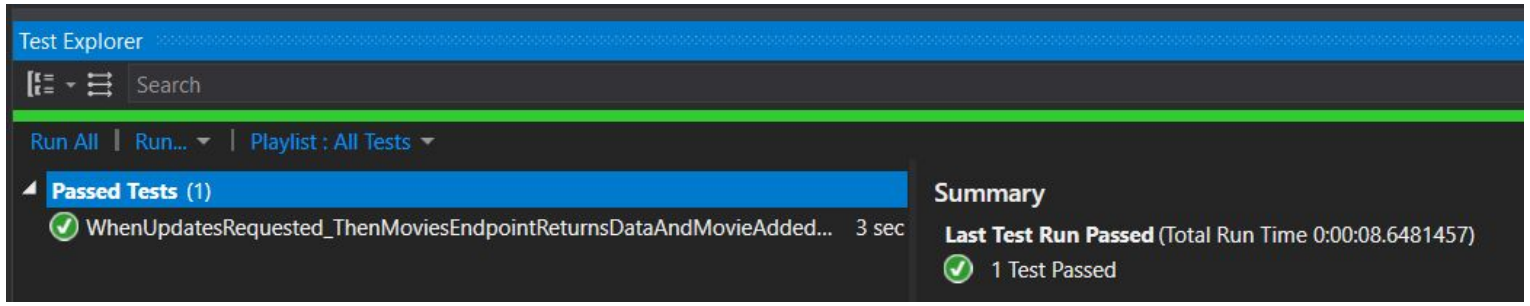
- Coarse grained unit tests



Passed Tests (1)
WhenCheckOccures_ThenMessagesWereSent 4 sec

Summary
Last Test Run Passed (Total Run Time 0:00:13.6390328)
1 Test Passed

- InMemory database and test server

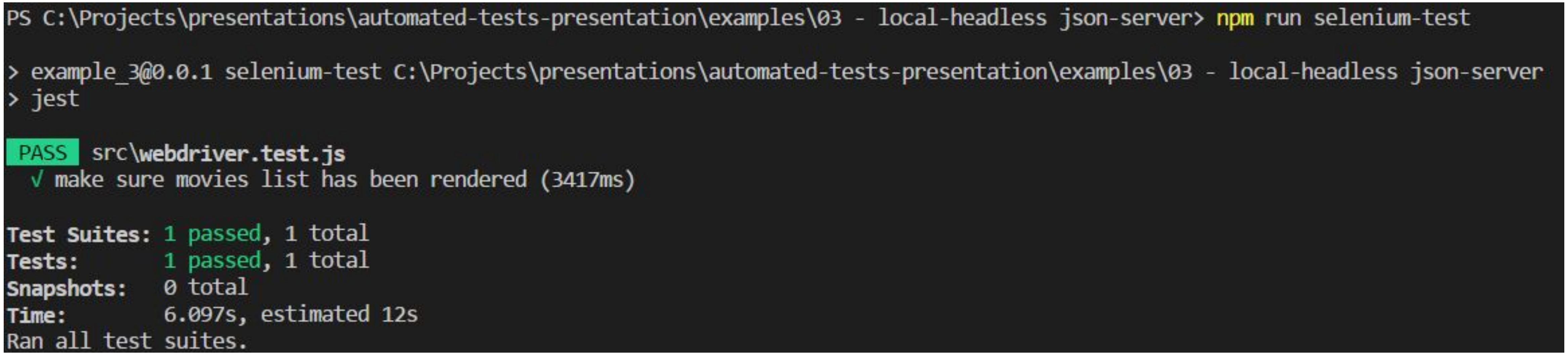


Passed Tests (1)
WhenUpdatesRequested_ThenMoviesEndpointReturnsDataAndMovieAdded... 3 sec

Summary
Last Test Run Passed (Total Run Time 0:00:08.6481457)
1 Test Passed

Developer tools – example 3

- Selenium WebDriver + Http Server + Json Server + headless Firefox
  - Steps:
    - npm run startWebServer (in a separate process)
    - npm run startApi (in a separate process
    - npm run selenium-test (in a separate process)



```
PS C:\Projects\presentations\automated-tests-presentation\examples\03 - local-headless json-server> npm run selenium-test

> example_3@0.0.1 selenium-test C:\Projects\presentations\automated-tests-presentation\examples\03 - local-headless json-server
> jest

 PASS  src\webdriver.test.js
  √ make sure movies list has been rendered (3417ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        6.097s, estimated 12s
Ran all test suites.
```

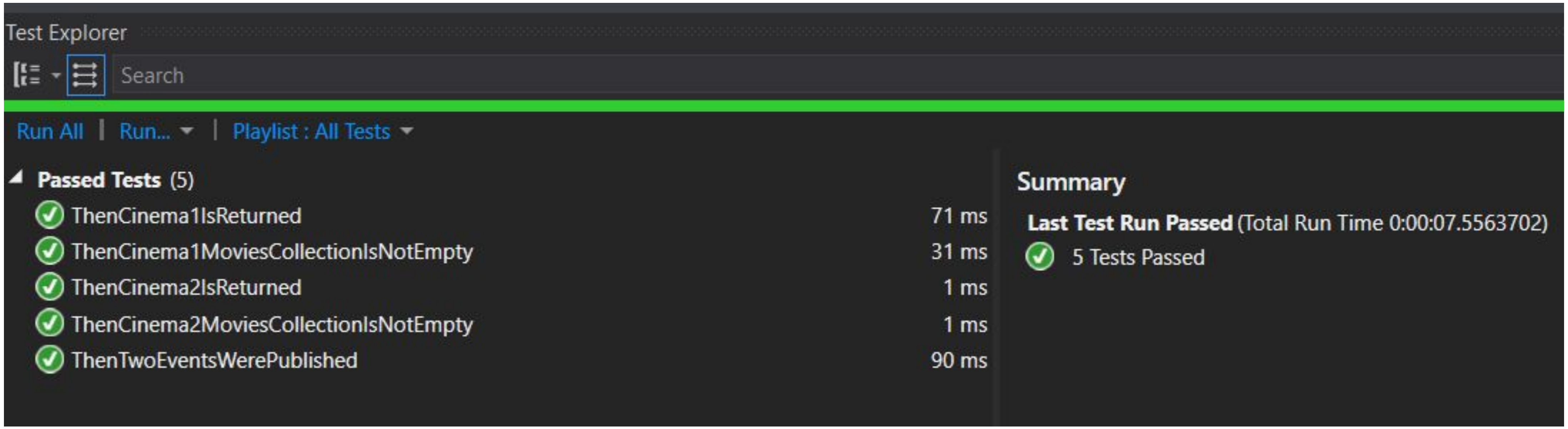
Ops tools

- Pipelines
- Monitoring
- Dashboards

How to write automated tests?

- Choose strategy – what is the purpose of the test(s)?
- Choose methodology – test function/class/component vs scenario testing
- Plan – before writing any code think about what tests are needed upfront
- Modularity - use building blocks

Modularity – example 4



Test Explorer

Search

Run All | Run... | Playlist : All Tests

Passed Tests (5)

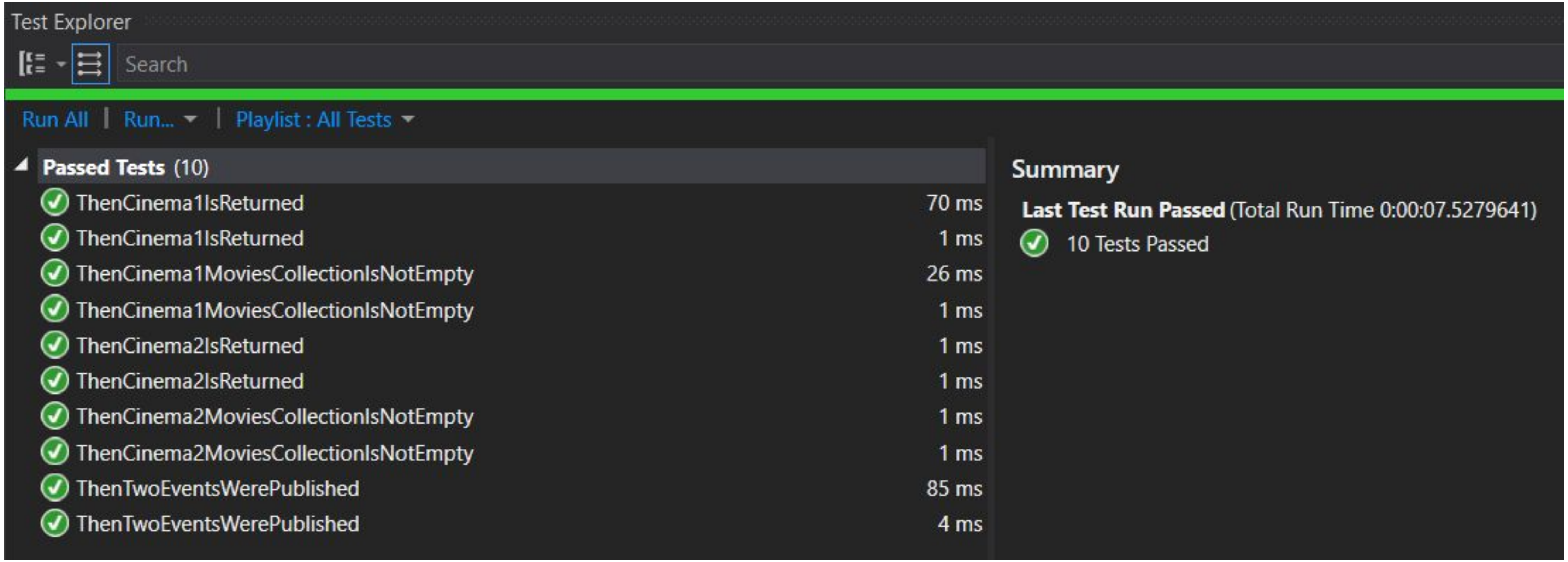
```
ThenCinema1IsReturned                     71 ms
ThenCinema1MoviesCollectionIsNotEmpty     31 ms
ThenCinema2IsReturned                      1 ms
ThenCinema2MoviesCollectionIsNotEmpty      1 ms
ThenTwoEventsWerePublished                90 ms
```

Summary

Last Test Run Passed (Total Run Time 0:00:07.5563702)

✓ 5 Tests Passed

Modularity - example 5



Test Explorer

Search

Run All | Run... | Playlist : All Tests

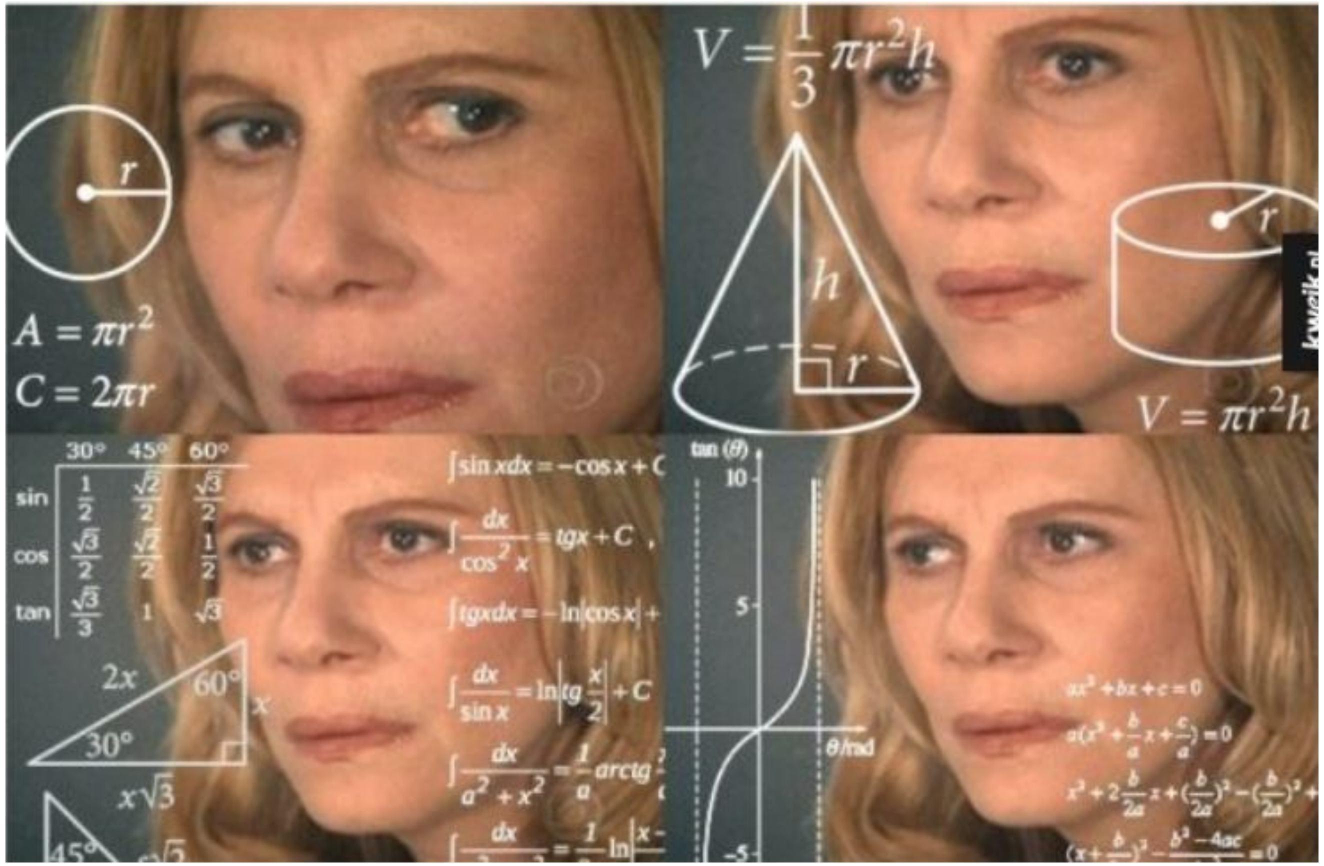
```
Passed Tests (10)
ThenCinema1IsReturned                      70 ms
ThenCinema1IsReturned                       1 ms
ThenCinema1MoviesCollectionIsNotEmpty      26 ms
ThenCinema1MoviesCollectionIsNotEmpty       1 ms
ThenCinema2IsReturned                       1 ms
ThenCinema2IsReturned                       1 ms
ThenCinema2MoviesCollectionIsNotEmpty       1 ms
ThenCinema2MoviesCollectionIsNotEmpty       1 ms
ThenTwoEventsWerePublished                 85 ms
ThenTwoEventsWerePublished                  4 ms
```

Summary

Last Test Run Passed (Total Run Time 0:00:07.5279641)

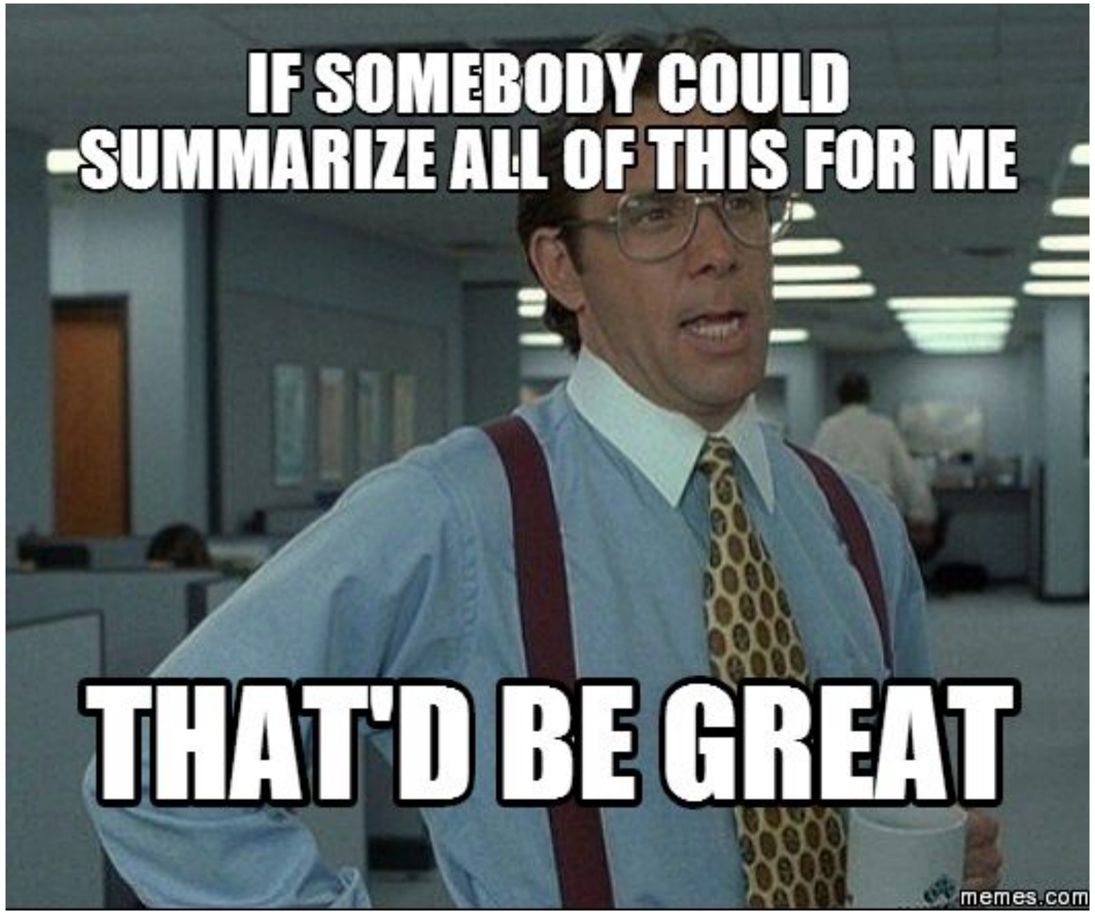
✓ 10 Tests Passed

How to test tests?



- Planning
- Code reviews
- Metrics - keep track of bugs caused by incorrect implementation in production
- Test coverage
- Mutation tests

Summary - pros & cons



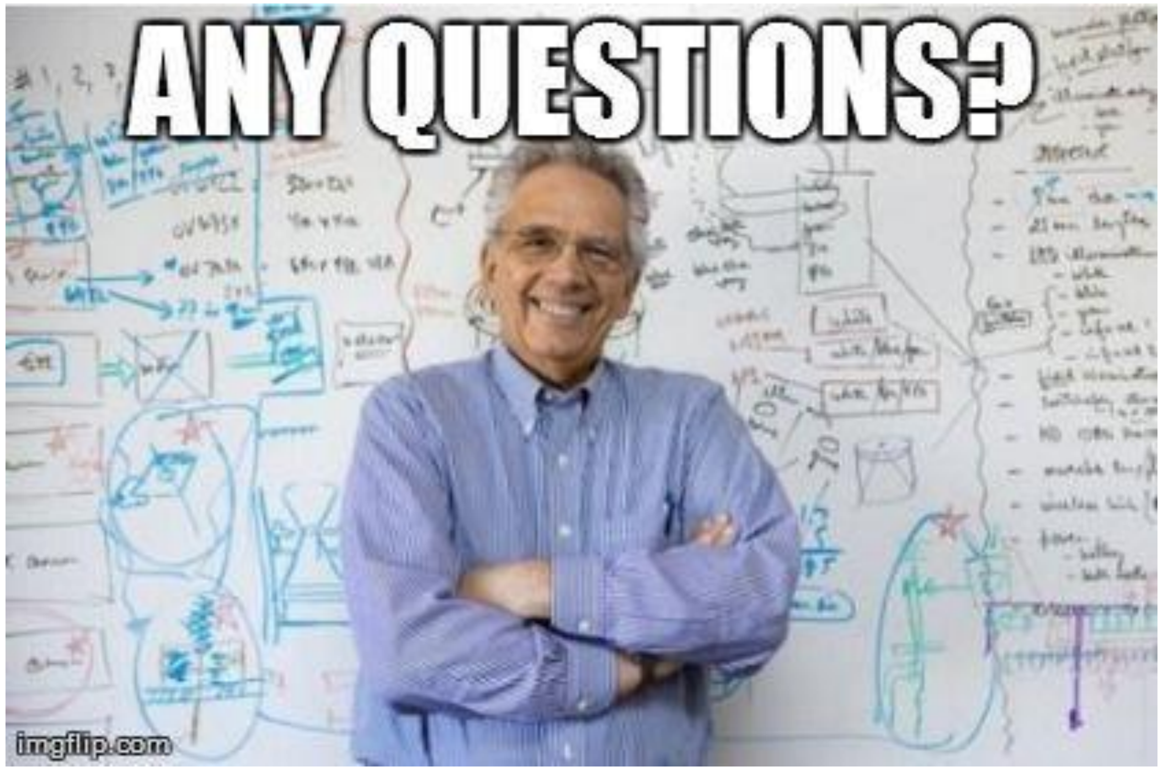
Difficulties

- Standardization and automation on a company level
- More complex changes can be …. more complex
- Cost (in short term) - time, tools, learning curve, training, etc.

Pros

- Huge savings on regression
- Catching up bugs much earlier
- More confidence while refactoring or expanding a system
- Standardization across the company

Questions?



Useful links

https://github.com/plitwinski/automated-tests-presentation

https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116

https://en.wikipedia.org/wiki/Mutation_testing

https://martinfowler.com/articles/microservice-testing