Fixing Your Automation Challenges in the Era of CI/CD

Moshe Milman
Co-Founder @ Applitools

@MosheMilman
WHO AM I?

- Co-Founder of Applitools
- Spent the last years working with multiple companies on software development and testing/automation projects
Agenda

• Automation challenges
• Market trends
• How can we fix it?
• Q&A
AUTOMATION
CHALLENGES
The modern software delivery cycle
Automation is key!
Your Brand Is Digital

OS Types

Browser Versions

Devices & Screen Types

Microservices
Test automation workflow

Product change

Author new tests

Execute tests

Analyze failures

Report bugs

Find cause

Fix bugs

Fix broken tests

Tester

Developer
Challenge #1: Low test coverage

Product change

Author new tests

Execute tests

Analyze failures

Report bugs

Fix broken tests

Find cause

Fix bugs

40-70% OF TESTER TIME IS SPENT HERE!
UI Testing

• The most widespread type of **functional** testing (using Selenium, Appium, Cypress, ...).

• Tests the end-to-end functionality of a system through its UI

• A test is a sequence of steps in which you
  - Simulate user inputs: clicks, swipes, keystrokes, etc.
  - Validate the system output on the page using assertions:

  ```javascript
  assertEquals(browser.getAttribute("//div[@id='container']/div[3]/div[2]/a[1]@class"), "Active");
  ```

• Problems:
  - A line of code for each UI element attribute assertion
  - What happens when the UI changes?
Every UI page consists of hundreds of UI elements
Analyzing test results
Challenge #1: Low test coverage

• Every assertion increases the maintenance overhead
• Is it really worth automating?
• Checks system functionality via the UI – Not the UI!
• Can only catch expected bugs
• Manual testing is still needed!
Challenge #2: Slow feedback

FULLY MANUAL

Product change

Execute tests

Analyze failures

Report bugs

Fix broken tests

Find cause

Fix bugs

Author new tests

Tester

Developer
Challenge #3: Root cause analysis

Product change -> Execute tests -> Analyze failures

Report bugs -> Find cause

Fix broken tests -> Fix bugs

Author new tests

REPRODUCE / COMPARE WITH WORKING CODE

Tester | Developer
Challenge #4: Ineffective and slow cross env tests

- Test on the same devices and browsers customers are using
- Prolongs test execution and test results analysis
- Requires an expensive test lab with devices and browsers
- What bugs are we trying to find?
  - Server bugs are 99% environment agnostic
    - For example: a broken DB query
  - App bugs are 99% environment agnostic (except for legacy browsers)
    - For example: not validating an input field, reversed sorting order, etc.
Challenge #5: Lack of skills / experience + Access to Talent
CI-CD and Digital Transformation are top R&D trends

Q1A - Which of the following initiatives were part of your company's plans in 2018?

- CI-CD: 60%
- Digital Transformation: 51%
- AI: 39%
- Shift Left: 39%
- Shift Right: 18%
- Other: 3%
- None of the above: 5%
Most companies are in various stages of partial automation

<table>
<thead>
<tr>
<th>Test Automation Coverage</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 75%</td>
<td>13%</td>
</tr>
<tr>
<td>51% to 75%</td>
<td>29%</td>
</tr>
<tr>
<td>26% to 50%</td>
<td>25%</td>
</tr>
<tr>
<td>10% to 25%</td>
<td>20%</td>
</tr>
<tr>
<td>Less than 10%</td>
<td>8%</td>
</tr>
<tr>
<td>No test automation at all</td>
<td>6%</td>
</tr>
</tbody>
</table>
50% still do manual visual testing or none at all

- Banking/finance somewhat more manual than others

- 100% Manual visual UI testing
- Mostly manual visual UI testing with some automated UI visual testing
- About 50% manual / 50% automated visual UI testing
- Mostly automated visual UI testing with some manual visual testing
- 100% automated visual UI testing
- We do not do any visual UI testing

Q3L - How do you manage visual UI testing today?
The pace of Digital Transformation is accelerating rapidly.

- This is only increasing the workload for already stretched teams

<table>
<thead>
<tr>
<th></th>
<th>Number Currently in Production</th>
<th>Number Planning to Add, Next 12 Months</th>
<th>Percentage Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apps</td>
<td>40.4</td>
<td>14.9</td>
<td>+63%</td>
</tr>
<tr>
<td>Pages or Screens</td>
<td>203.2</td>
<td>163.3</td>
<td>+20%</td>
</tr>
<tr>
<td>Viewports</td>
<td>5.3</td>
<td>3.6</td>
<td>+33%</td>
</tr>
<tr>
<td>Languages</td>
<td>6.5</td>
<td>3.5</td>
<td>+46%</td>
</tr>
<tr>
<td>Releases</td>
<td>16.0</td>
<td>11.9</td>
<td>+26%</td>
</tr>
</tbody>
</table>
AI assisted testing

• Reduce the test maintenance overhead
  • Self healing tests
  • Timing anomalies
  • Smart crawlers
Codeless testing

• Codeless testing is re-gaining traction
• Adopted for various use cases
• Selenium IDE is back!
More developers write tests
There’s a need for speed
There is no release

Realize the truth

There is no release
“throwing things over the wall” is fading away
The QA Gateway is fading away
Developers MUST Test
Tests must be part of the development cycle
Tests must be fast

• Developers shouldn’t wait
• They have to commit and push now
• They need to know now that the code runs
So they’ve started writing tests!
There are two main schools of testing
The “Pyramid” School

- Unit
- Integration
- E2E
The “Diamond” School

E2E

Integration

Unit
Which is better?
It’s a Religion!
Commonalities

- The need for speed
- Few E2E tests
- Currently, functionality only
Developers finally have a (somewhat) common methodology
Write tests. Not too many. Mostly integration.
Kent C. Dodds and The Trophy of Testing
Cross-browser testing is declining
There are only 3 main browsers today

- Chrome
- Firefox
- Safari
What about Mobile?

- Chrome
- Safari
What about IE/Edge?

- IE: mostly dead
- Edge: will move to Chrom(ium) in 2019
For most applications, browsers are functionally the same.
Finding browser-specific bugs is getting rarer

But the time and money needed to do so is not getting cheaper!
Except for Visual Bugs!

- Still differences between browsers
- Add to that responsive design, for cross-width testing
  - iPhone size
  - iPad size
  - Desktop size
  - Retina size
HOW CAN WE FIX IT?
Visual Testing???

- **Visual Testing** is the process of validating all the visual aspects of an application's UI on all platforms.

- Goes beyond functional testing tools like Selenium to ensure things like colors, fonts, buttons appear correctly.
Challenges #1, #2: Low test coverage and slow feedback
Full page (functional) validation
99.9999% accuracy … and improving!
Last 30 batch runs

No filtering

Failed: 0/100 | Resolved diffs: 0% | 76 unresolved steps

- Layout1
  - 09/08/2016 at 5:07 PM
  - Failed
- Layout2
  - 09/08/2016 at 5:00 PM
  - Failed
- Homepage
  - 09/19/2016 at 7:44 PM
  - Failed
- Github
  - 03/29/2016 at 2:13 PM
  - Failed
- Wikimedia
  - 03/17/2016 at 4:30 PM
  - Failed
Low test coverage and slow feedback

SOLVED!

- Visual and functional coverage
- Every team member can easily analyze test results and fix tests
- Automated maintenance facilitates scale
- Easy to author new tests
- Easy to review test results
Challenge #3 - Root cause analysis
Challenge #3 - Root cause analysis – SOLVED!
Challenge #4: Ineffective and slow cross-env tests
This is how test code looks like

Navigate

Click, click, click

checkScreenshot

Click, click, click

checkScreenshot

Click, click, click

checkScreenshot

Click, click, click

checkScreenshot
Simple, Right?
Because you need to visually test for...

- Different browsers
  - Chrome, Firefox, Safari

- Different responsive widths and pixel densities
  - iPhone width and pixel density
  - Tablet width
  - Regular desktop width
  - Retina display width and pixel density
So...

Navigate
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot

Navigate
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot

Navigate
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Click, click, click checkScreenshot
Run each test ten times?
Run each test ten times?

(not even with Cypress…)
How do we solve this problem?

We parallelize
Solution: Parallelizing the tests

Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
Click, click, click
checkScreenshot
There’s a better solution
Parallelize the screenshots!

Navigate

Click, click, click
checkScreenshot #1

Click, click, click
checkScreenshot #2

Click, click, click
checkScreenshot #3

Click, click, click
checkScreenshot #4
How? DOM Snapshots!

Navigate
Click, click, click checkScreenshot #1
Click, click, click checkScreenshot #2
Click, click, click checkScreenshot #3
Click, click, click checkScreenshot #4
And what do we do with the screenshots?
Challenge #4: Ineffective and slow cross-env tests - solved!

- Cover all environments at the speed of running a single test
- Find visual bugs which are likely to be env specific
- Fraction of the cost&speed penalty of a full featured physical browser / device farm
OK, OK. But how?

How do I write tests?

Show me some code!
One more thing...

Actually two 😊
Applitools Test Automation University

Non profit initiative to help more engineers get into automation

Already trained over 11K engineers in 3 months

Making Open Source Visually Perfect

Open source projects get free access to Applitools*

https://testautomationu.applitools.com/
Thank You
Stay in touch!
@MosheMilman