



Targeting Change: How to Test Less & Manage the Risk

Twin Cities Quality Assurance Association

February 8, 2018

Andrey Madan – Lead Solution Architect

Agenda

- Testing Challenge in the Agile world
- Testing Pyramid
- Managing Risk of Change
- Continuous Testing Maturity Model
- Discussion and Q & A



THE MOST INNOVATIVE AND COMPLEX TIME IN SOFTWARE DEVELOPMENT HISTORY

50% #1

Agile adoption is top business priority

2011

92% #5

Time-to-market is top business priority

2012

\$26.5BB

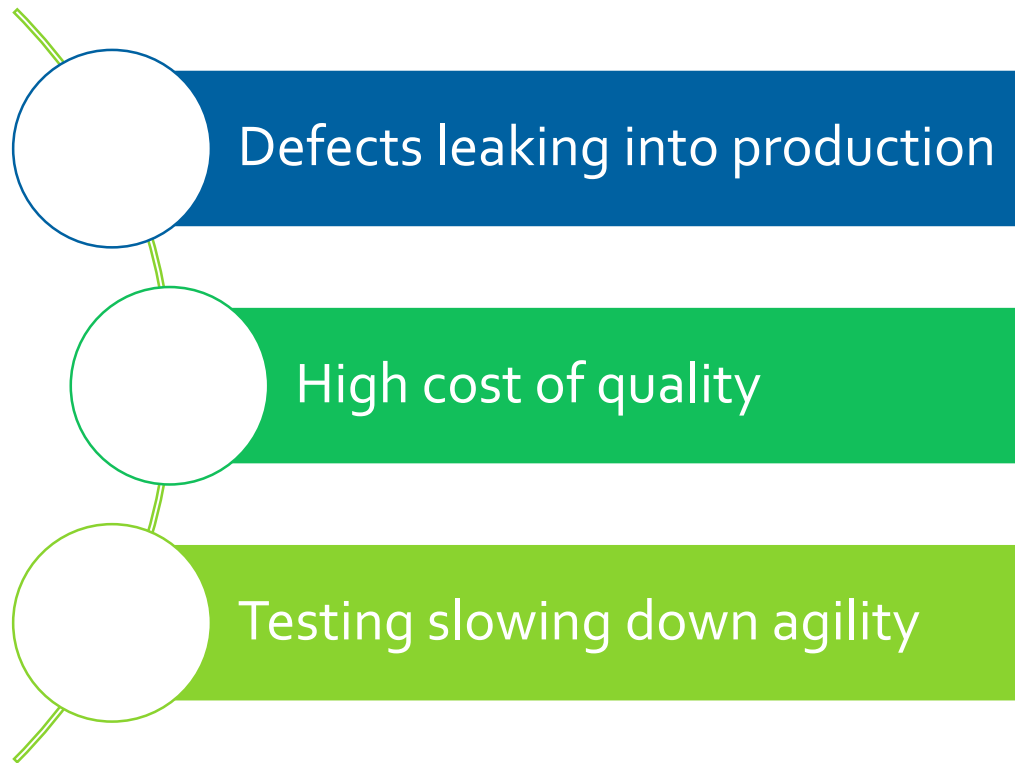
Lost revenues on the Internet of Things

2015



Balancing Quality and Speed

Top challenges faced by software development organizations

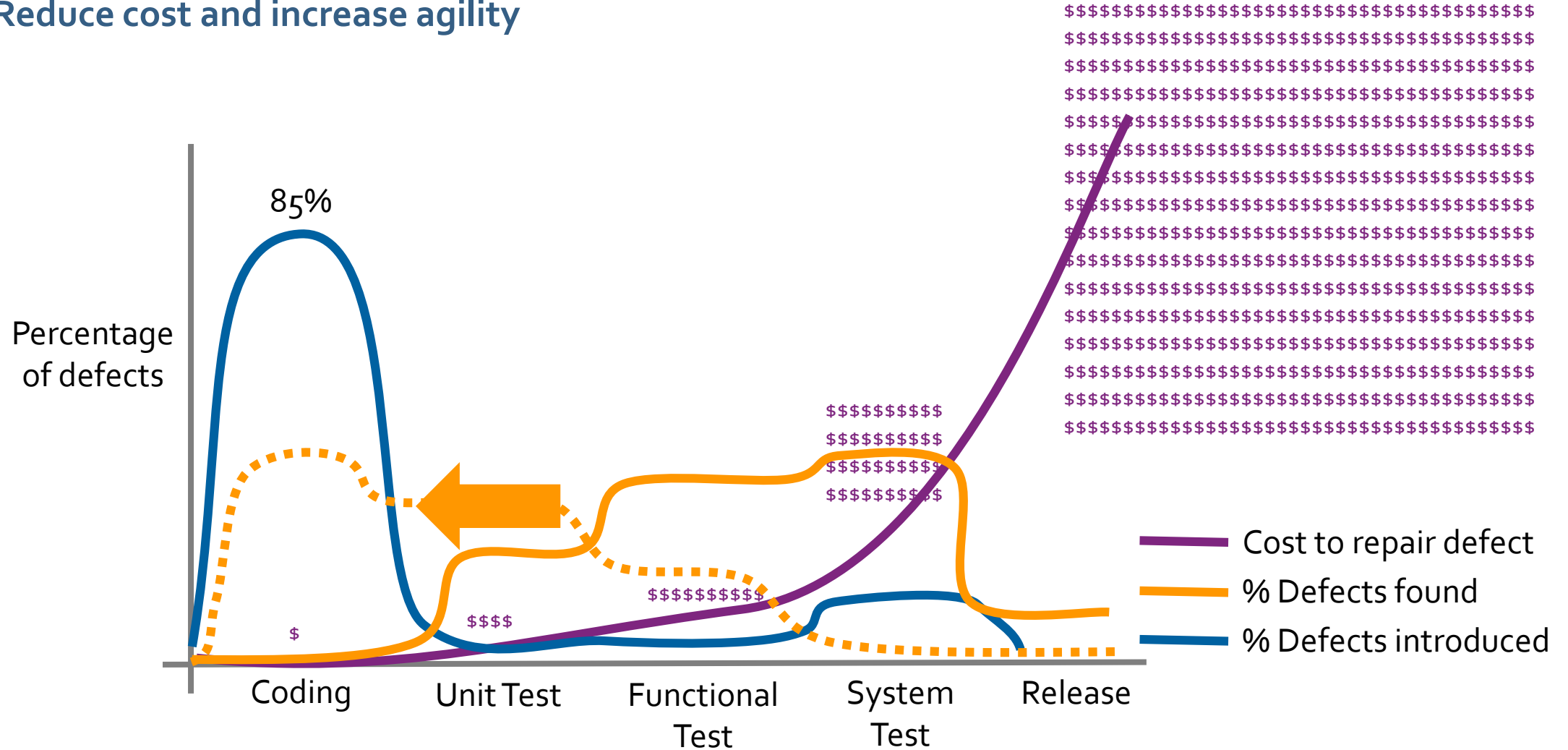


- Waiting to test until everything is ready
- Lack of quality practices early in the SDLC
- Silo-ed Dev / QA ... "QA is a bottleneck"
- Incomplete test coverage
- Late-cycle defect detection
- Last minute changes or fixes
- Performance and security testing "just before release"

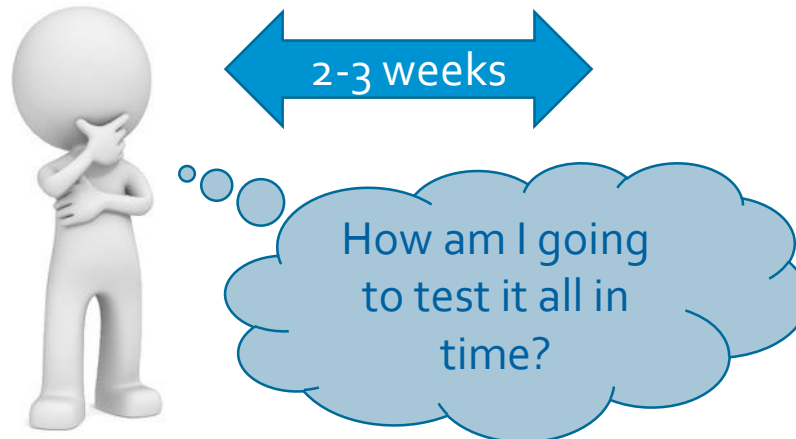
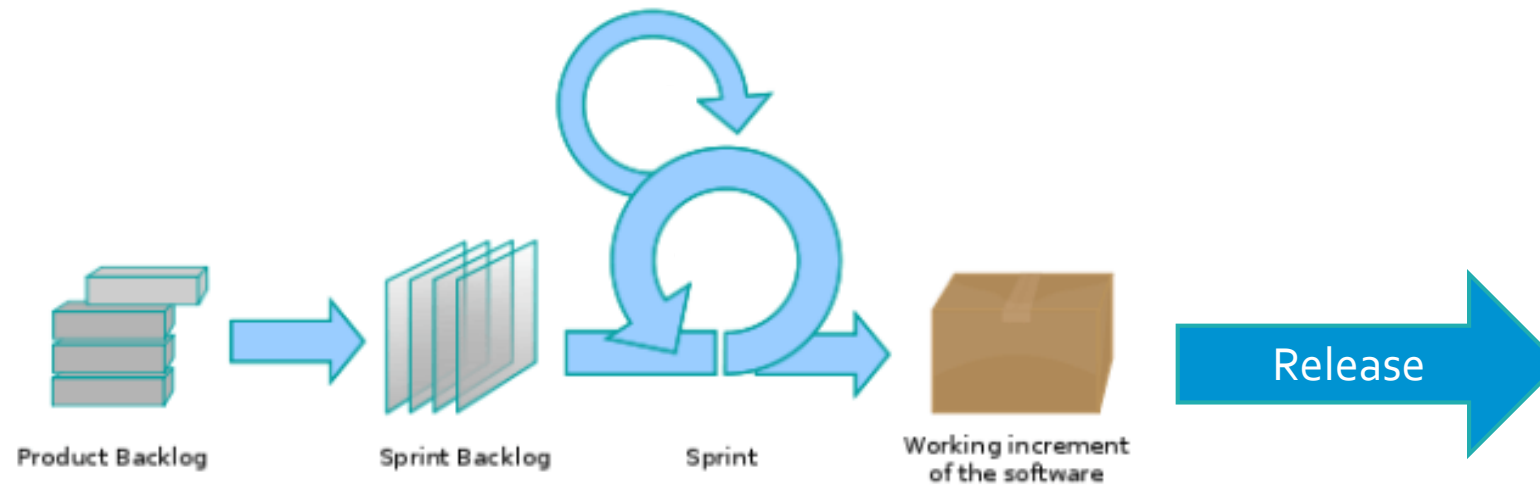


'Shift-left' defect detection and remediation

Reduce cost and increase agility

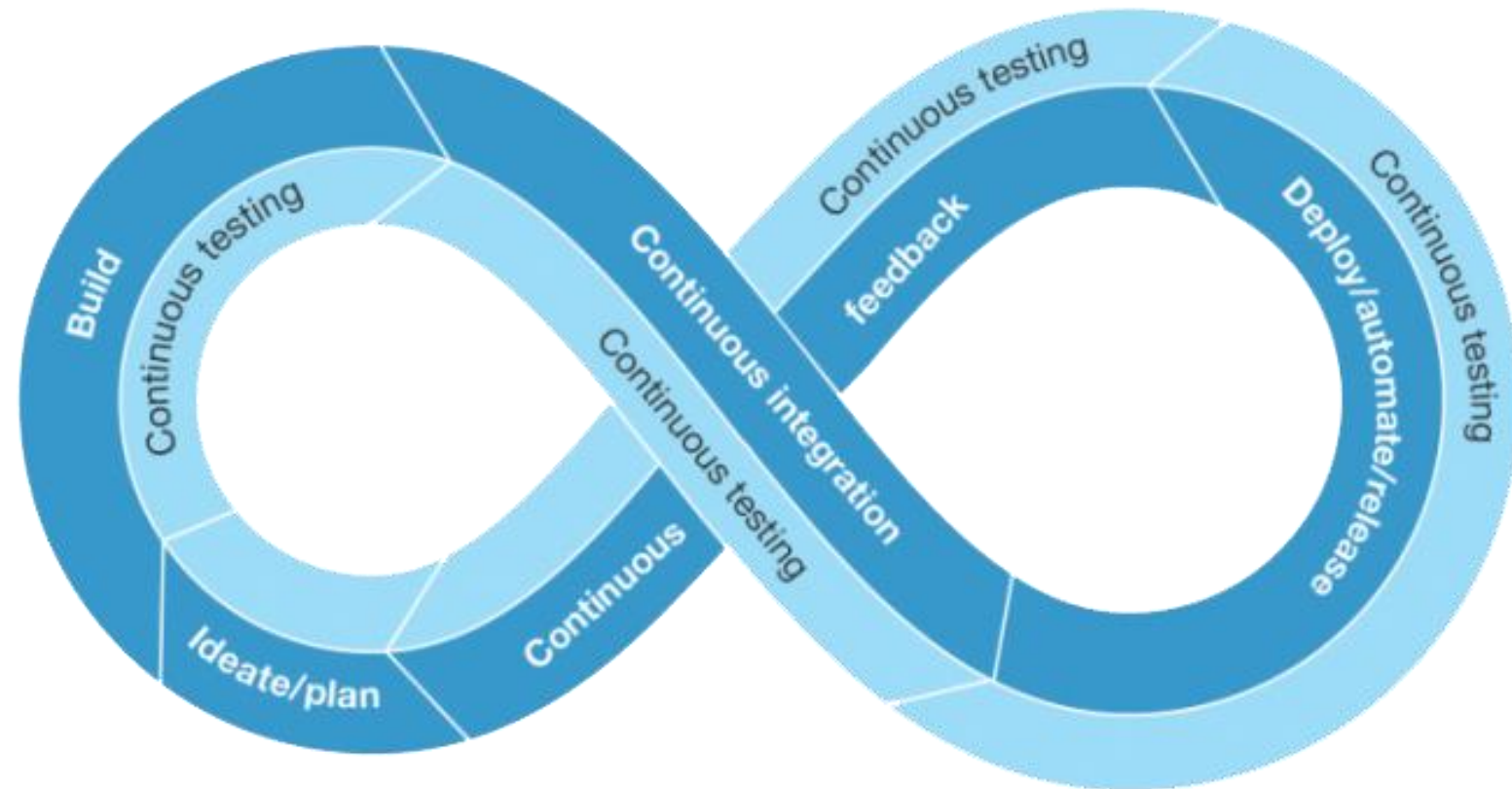


Agile = "I don't have time to test EVERYTHING"



The key to unlocking the benefits of Agile

Continuous execution of 'all' tests during each stage of the SDLC



The Approach

Beck's Directive

- Make it Work
- Make it Right
- Make it Fast

- **Make It Work** – Setup Your Testing Ecosystem (harness, tools, frameworks)
- **Make It Right** – Create a solid Testing Pyramid
- **Make It Fast** – Focus on “Testing the Change”

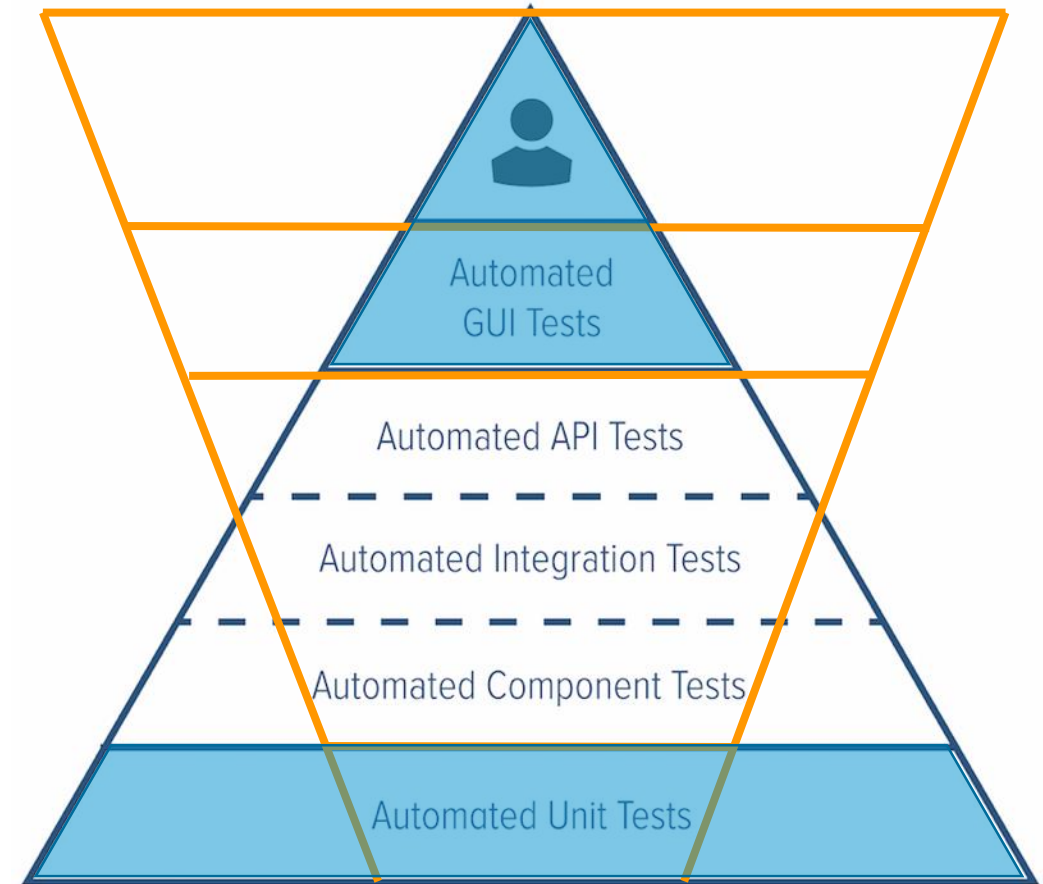


Building a solid Testing Pyramid

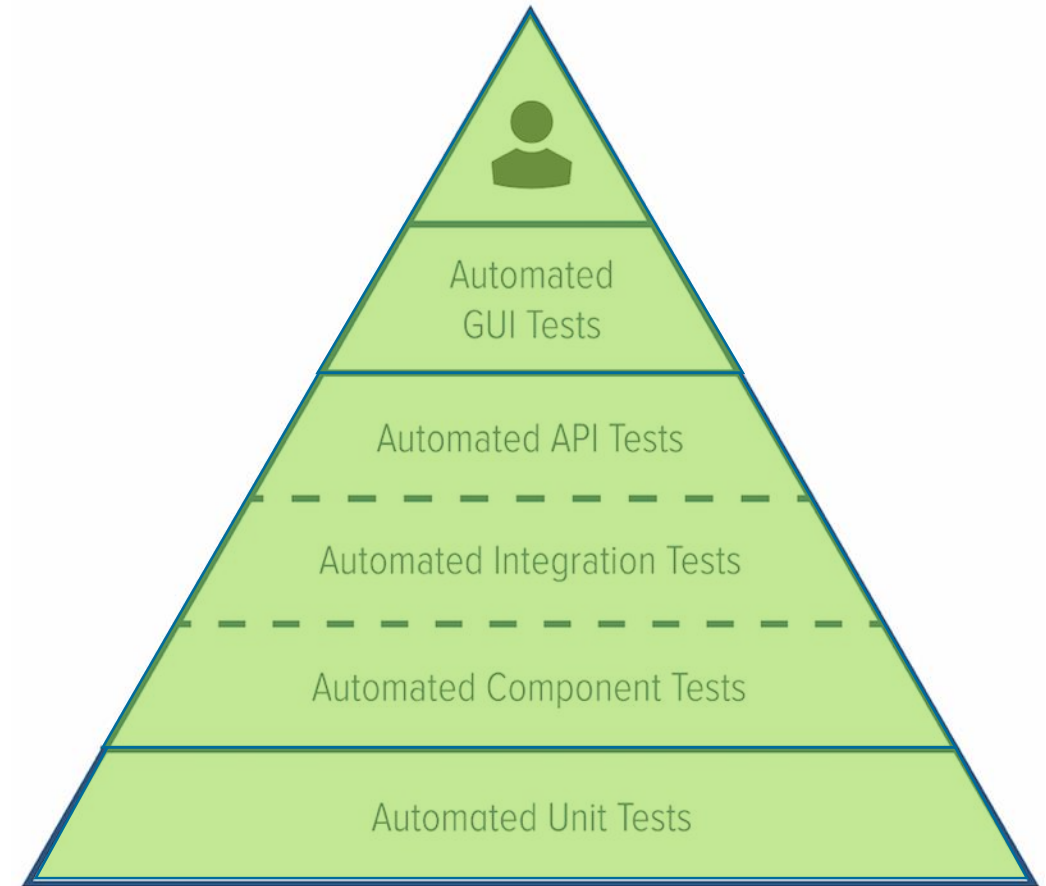
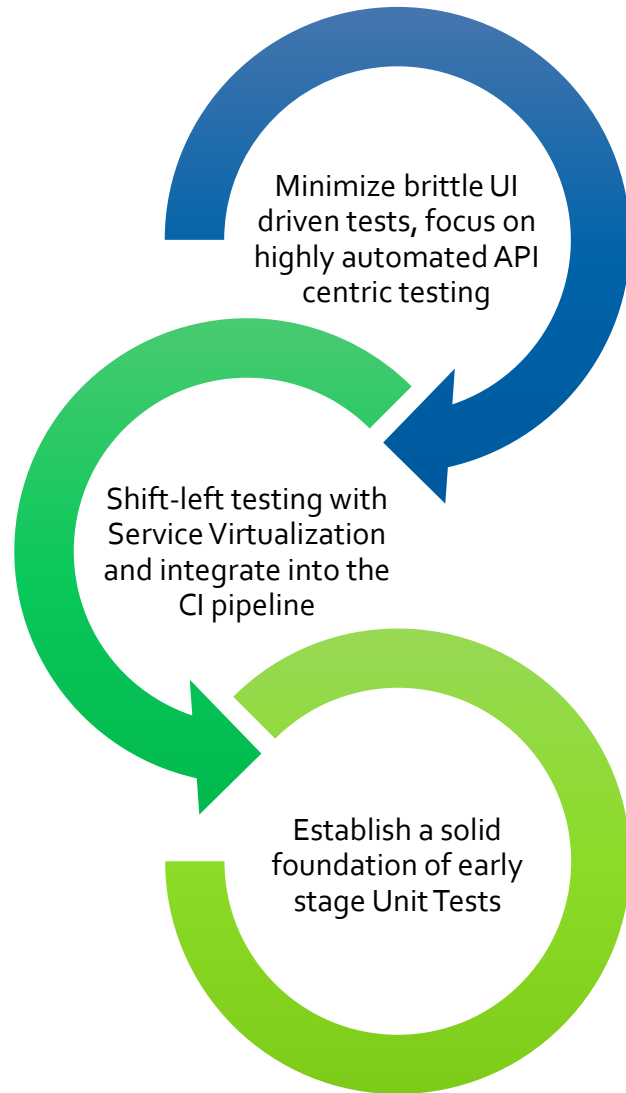
Quick to define, Time consuming to execute, complex test environment, **hard to automate**



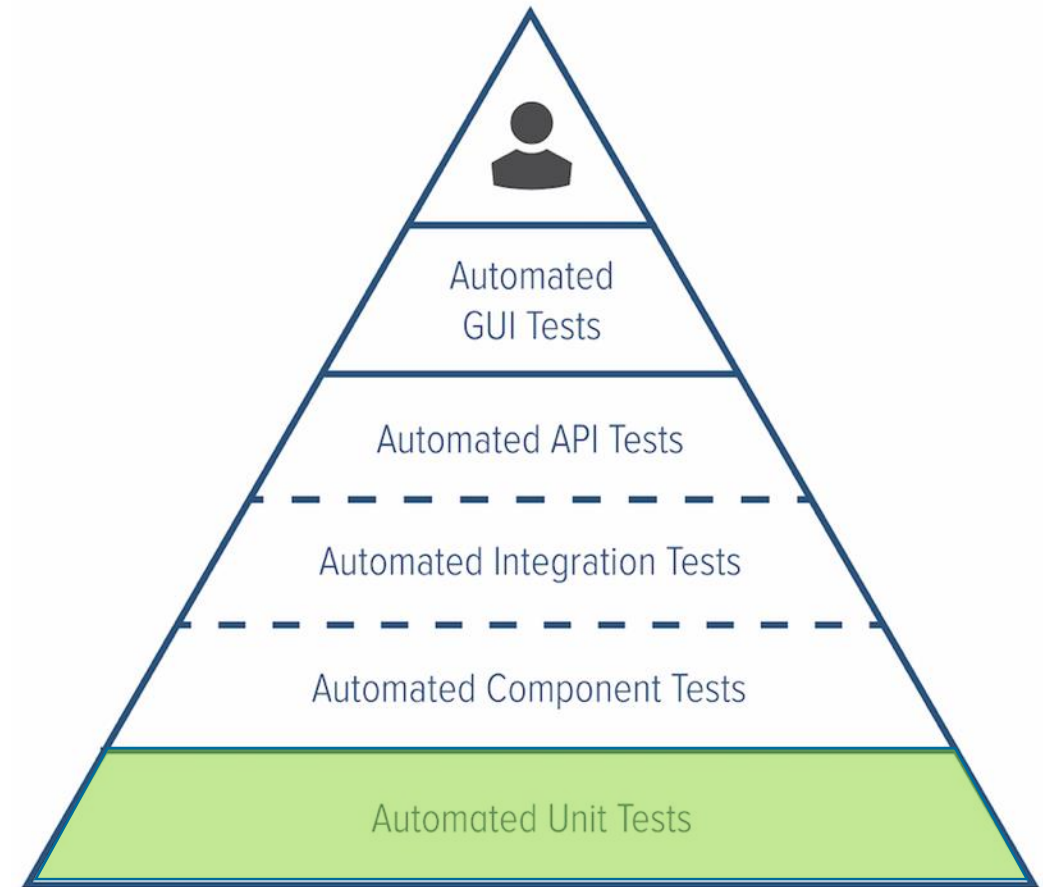
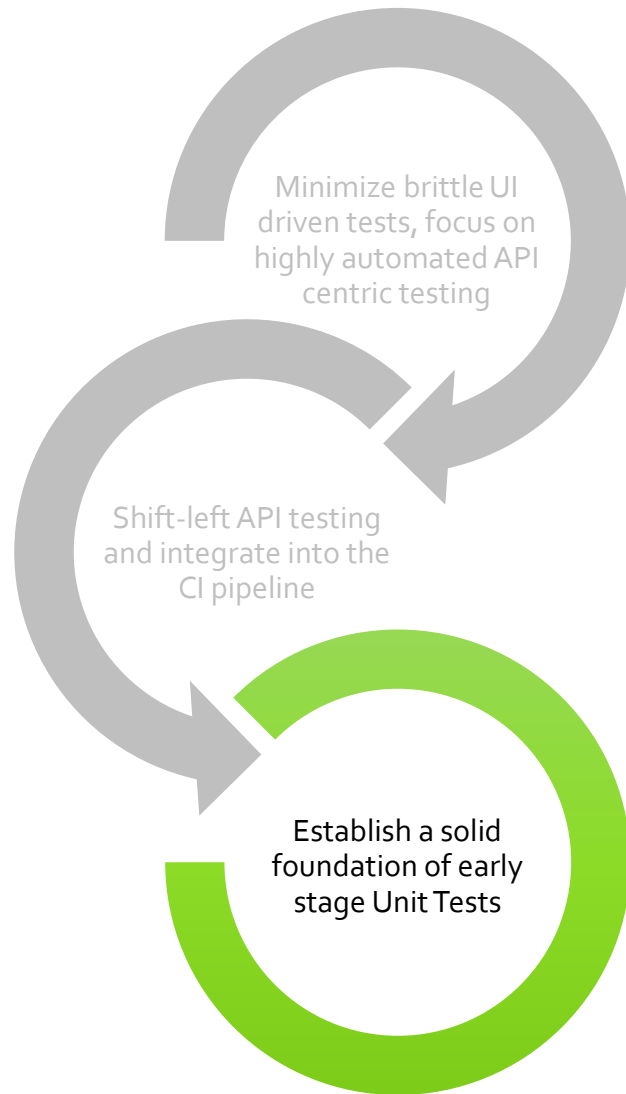
Quick to execute, Simple test environment, easy to automate but **hard to create**



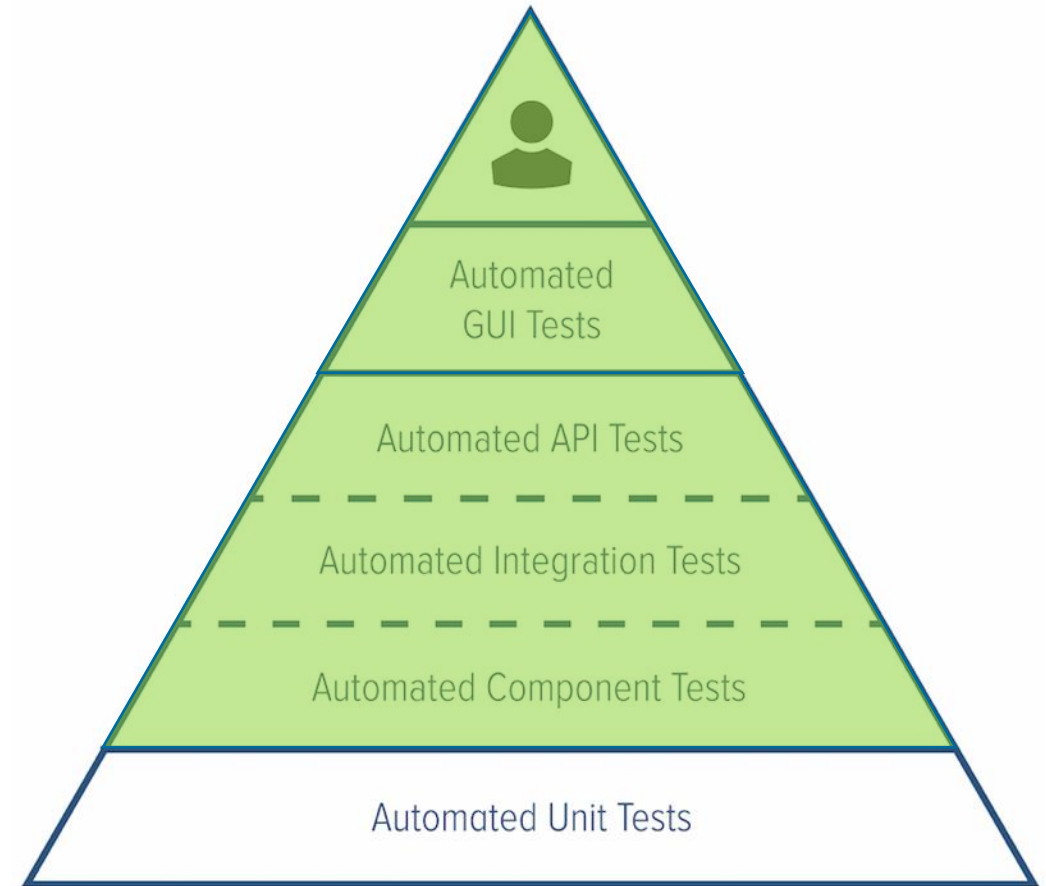
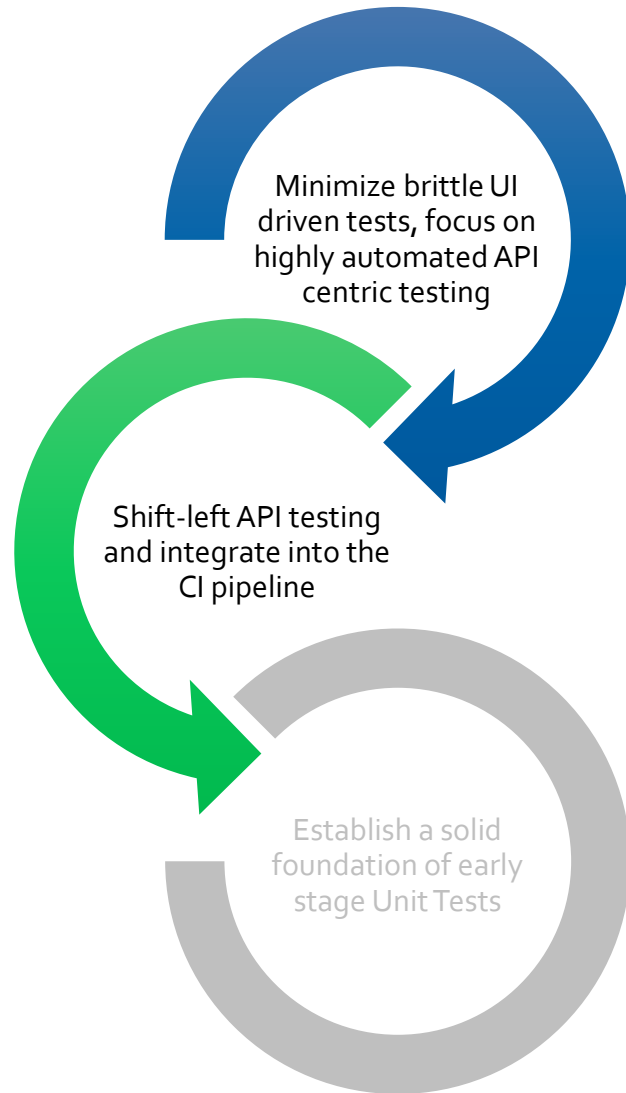
Building a solid Testing Pyramid



Building a solid Testing Pyramid



Building a solid Testing Pyramid



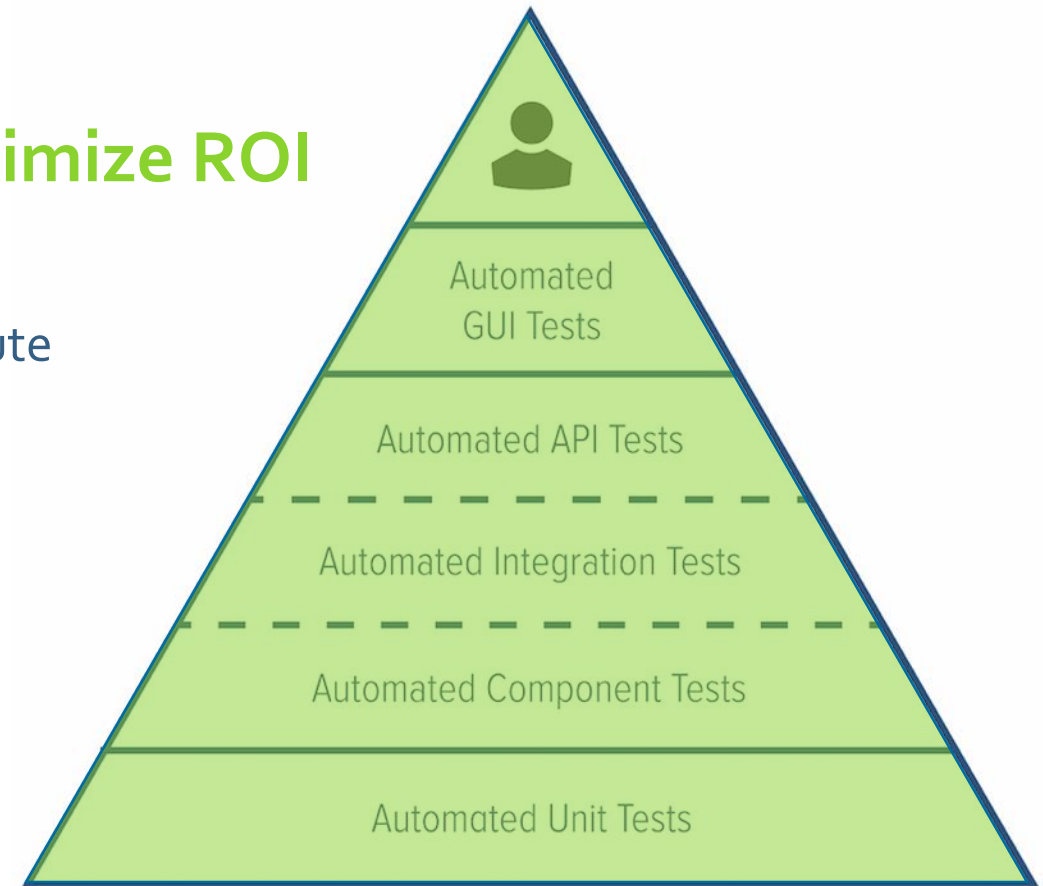
Building a solid Testing Pyramid

1. Blend testing techniques to **maximize ROI**

- R = Quality/Risk Mitigation
- I = Developer/Tester's time to create and execute

2. Build a foundation of **stable and continuous automated tests.**

3. Leverage advance analytics to focus on change and **accelerate testing schedules**



... but there's still a problem

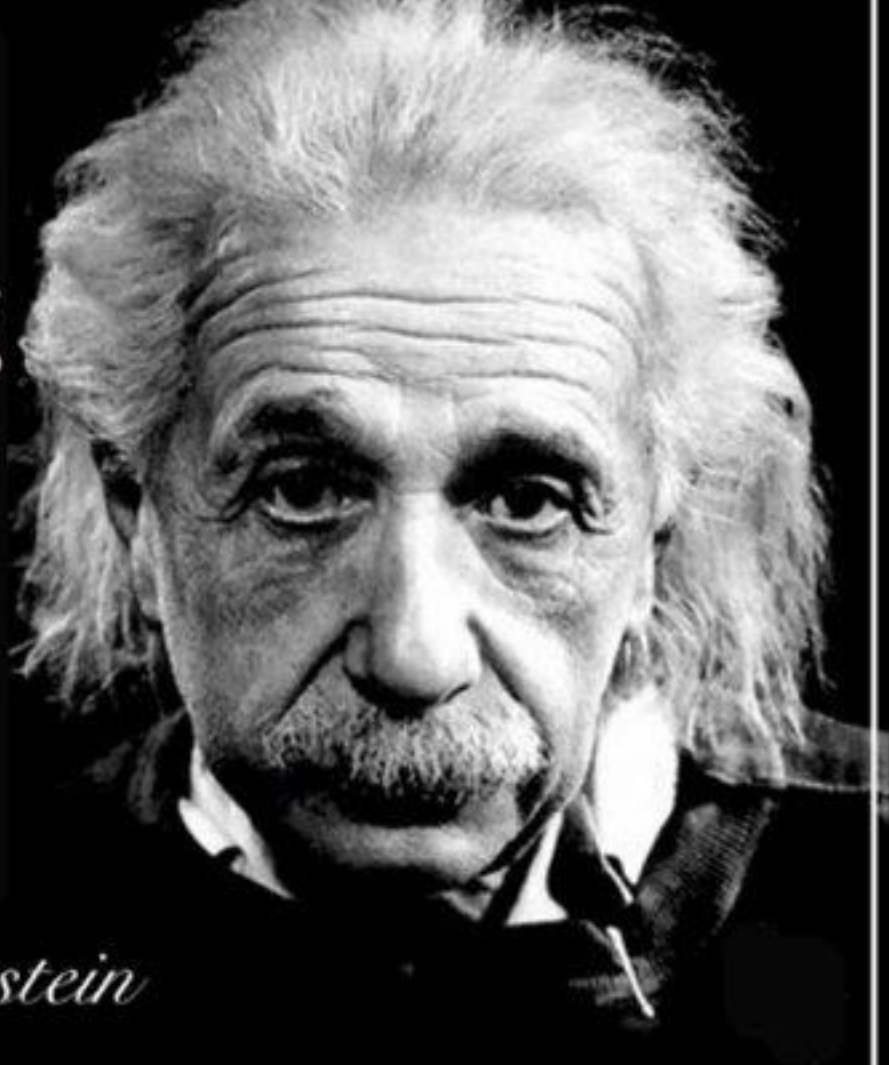


“Why is the release delayed AGAIN?”



Insanity:
doing the same thing
over and over again
and expecting
different results.

- Albert Einstein



The Approach

Beck's Directive

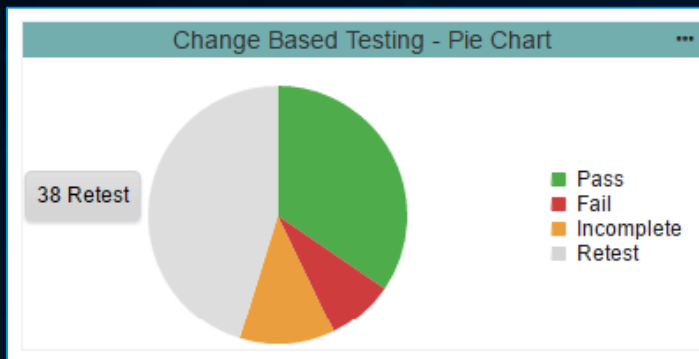
- Make it Work
- Make it Right
- Make it Fast

- **Make It Work** – Setup Your Testing Ecosystem (harness, tools, frameworks)
- **Make It Right** – Create a solid Testing Pyramid
- **Make It Fast** – Focus on “Testing the Change”



What is Change Based Testing?

1. Understand **what each test covers**
2. Understand **which code changed**
3. Focus on the tests that **validate the changes**



Change Based Testing - Files

Filter: Parabank-v3 Baseline Build: PARABANK3-20160810-38 Target Build: PARABANK3-20160919-11delta

Totals -- Pass: 29 Fail: 7 Incomplete: 10 Retest: 38

File	Pass	Fail	Incomplete	Retest
BookStoreDB.java	2	0	0	15
CartService.java	3	0	0	21
BookStoreDB.java	2	0	0	15
Transaction.java	19	6	10	2
LocalPaymentProvider.java	2	1	0	0



What is Code Coverage?

- Some Types:
 - Line
 - Function
 - Statement
 - Branch
 - Condition
 - Path
 - Modified Condition / Decision Coverage (MCDC)

253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

```
private void division_button_Click(object sender, System.EventArgs e)
{
    _stackList.divide();
}

private void fifo_rbutton_CheckedChanged(object sender, System.EventArgs e)
{
    lifo_rbutton.Checked = false;
    _isLifoStackChosen = true;
}

private void lifo_rbutton_CheckedChanged(object sender, System.EventArgs e)
{
    fifo_rbutton.Checked = false;
    _isLifoStackChosen = false;
}
```



Benefit of code coverage

- It helps in finding areas of a program not exercised by a set of test cases
- It highlights the need for additional test cases to increase coverage
- It helps in determining a quantitative measure of code coverage, which indirectly measure the quality of the application or product.



Disadvantage of code coverage

- There is danger in using a coverage measure. 100% coverage does *not* mean 100% tested
- It measures coverage of what has been written in the code; it cannot say anything about the code that has *not* been written



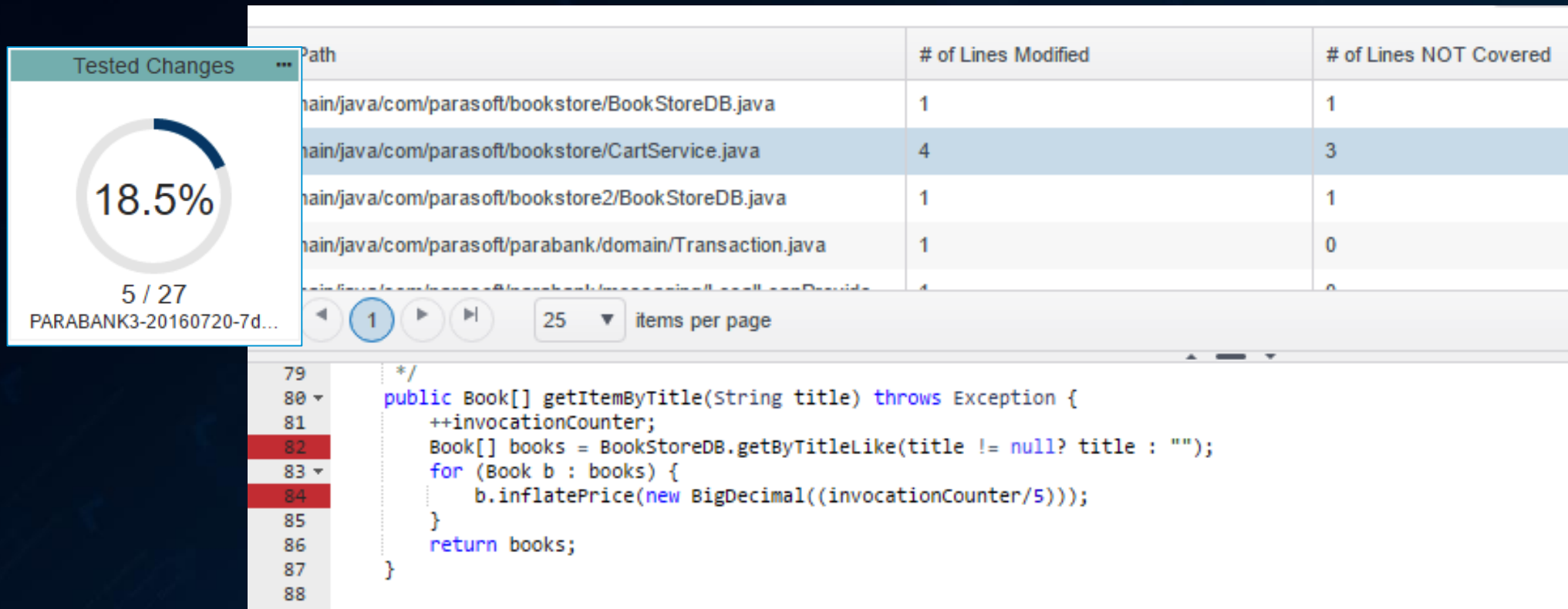
... but did you fully test the change?

- The existing tests executed ... do you need create more?
 - “My coverage is stable”
 - “Code coverage went up to 80% – so everything is good!”
- ... but what is the coverage on the modified code?
 - Many people changing many files, how do you know if you actually tested the change?
 - Did you test the right “80%”?
- With large code bases, and limited resources organizations **don't have time to create tests for “everything”** ...



... focus on the Modified Coverage

- Target 100% 'modified coverage' vs 80% of overall coverage



The screenshot displays a code coverage tool interface. On the left, a summary widget titled "Tested Changes" shows a donut chart with 18.5% coverage, representing 5 out of 27 lines. Below the chart, the text "PARABANK3-20160720-7d..." is visible. The main area features a table with the following data:

Path	# of Lines Modified	# of Lines NOT Covered
main/java/com/parasoft/bookstore/BookStoreDB.java	1	1
main/java/com/parasoft/bookstore/CartService.java	4	3
main/java/com/parasoft/bookstore2/BookStoreDB.java	1	1
main/java/com/parasoft/parabank/domain/Transaction.java	1	0
main/java/com/parasoft/parabank/messaging/LocalAppProvider.java	1	0

Below the table, a code editor shows the following Java code snippet:

```
79  */
80  public Book[] getItemByTitle(String title) throws Exception {
81      ++invocationCounter;
82      Book[] books = BookStoreDB.getByTitleLike(title != null? title : "");
83      for (Book b : books) {
84          b.inflatePrice(new BigDecimal((invocationCounter/5)));
85      }
86      return books;
87  }
88  }
```



Continuous Testing Maturity Model

Ad-hock / Reactive

- **Focus: Overcoming short-term testing 'point problems'**
- Limited automated execution of test scenarios
- Reacting to 'immediate' problem with the test environment
- Focused on eliminating specific unstable/unavailable environment constraints
- Short time-life/disposable artifacts
- Static/fix responses and example payloads

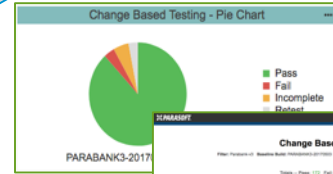
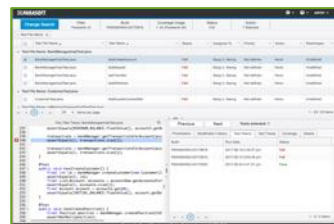
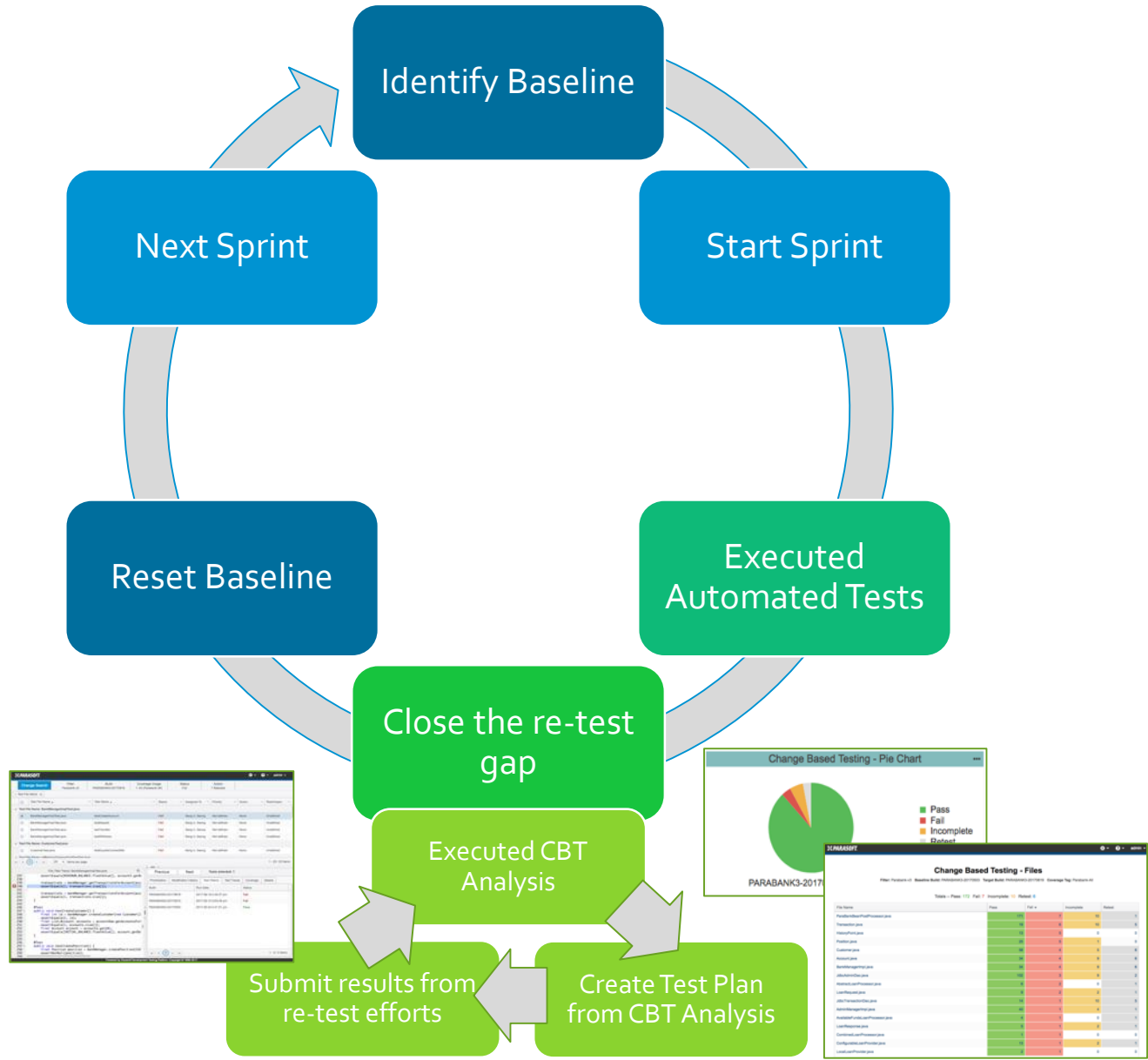
Automated / Managed

- **Focus: Embedding automation into CI pipelines**
- Automated tests decoupled from constrained test environments
- Reusable virtual assets supporting different functional and performance scenarios
- Long-lasting and regularly maintained artifacts
- Leveraging data for parameterization of test scenarios and virtual assets

Optimized / Proactive

- **Focus: Continuous Testing to enable Continuous Delivery**
- Validating 'what-if' scenarios and 'corner-case'
- Dynamic deploy-and-destroy of test environments (via containers)
- Larger Portfolio of virtual assets with different levels of complexity; CRUD, data learning
- Module design of test suites
- Reuse of API test scenarios to enable shift-left performance and security testing
- Data modeling and abstraction for expanded test coverage





File Name	Pass	Fail	Incomplete	Blocked
TestFile1.txt	10	0	0	0
TestFile2.txt	5	2	1	0
TestFile3.txt	3	1	0	0
TestFile4.txt	2	0	0	0
TestFile5.txt	1	0	0	0
TestFile6.txt	0	0	0	0
TestFile7.txt	0	0	0	0
TestFile8.txt	0	0	0	0
TestFile9.txt	0	0	0	0
TestFile10.txt	0	0	0	0



The Approach

Beck's Directive

- Make it Work
- Make it Right
- Make it Fast

- ✓ **Make It Work** – Setup Your Testing Ecosystem (harness, tools, frameworks)
- ✓ **Make It Right** – Create a solid Testing Pyramid
- ✓ **Make It Fast** – Focus on “Testing the Change”



Time for a Quick Demo



Summary

- ✓ Focus on test strategy, not tools: Build a solid Testing Pyramid
- ✓ Prioritize creation of new tests on the changed lines of code not covered by existing regressions
- ✓ Leverage intelligent analytics, including Change-Based Testing, Modified Code Coverage, and Risky Code Change, to prioritize your agile testing activities





Thank you

Contact information:
andrey.madan@parasoft.com