

Driving Development Using Examples

Sai Venkatakrishnan
Developer in Test, Thoughtworks
<http://developer-in-test.blogspot.com>
Fork me at - <http://github.com/saivenkat>
Follow me at - http://twitter.com/sai_venkat



Introduction

- Problems we are trying to solve.
- How to build the right thing?
- What do we get out of this?
- Gotchas
- Coding time :)



**Are we building the thing right as
well building the right thing?**





TESTING

I FIND YOUR LACK OF TESTS DISTURBING.

Intention and Implementation



How the customer explained it



How the Project Leader understood it



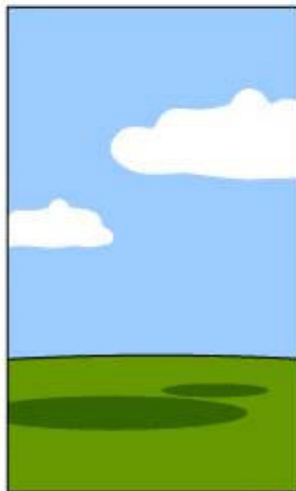
How the Analyst designed it



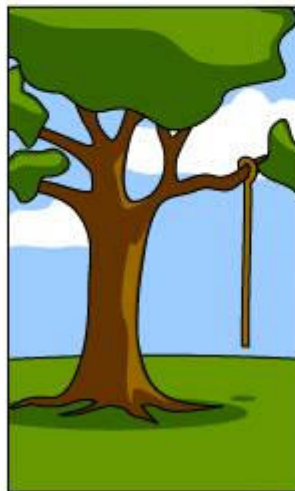
How the Programmer wrote it



How the Business Consultant described it



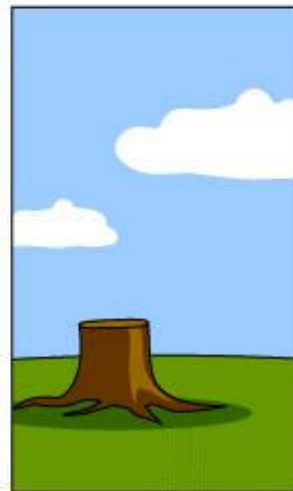
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

Broken Language

靴の間違いに
ご注意ください!!

Please note the mistake
of shoes.

傘は傘立てに
入れてください。

※院内使用禁止

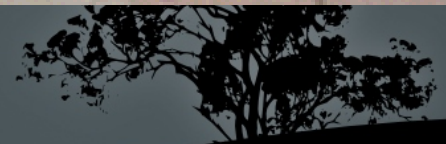


Broken Language

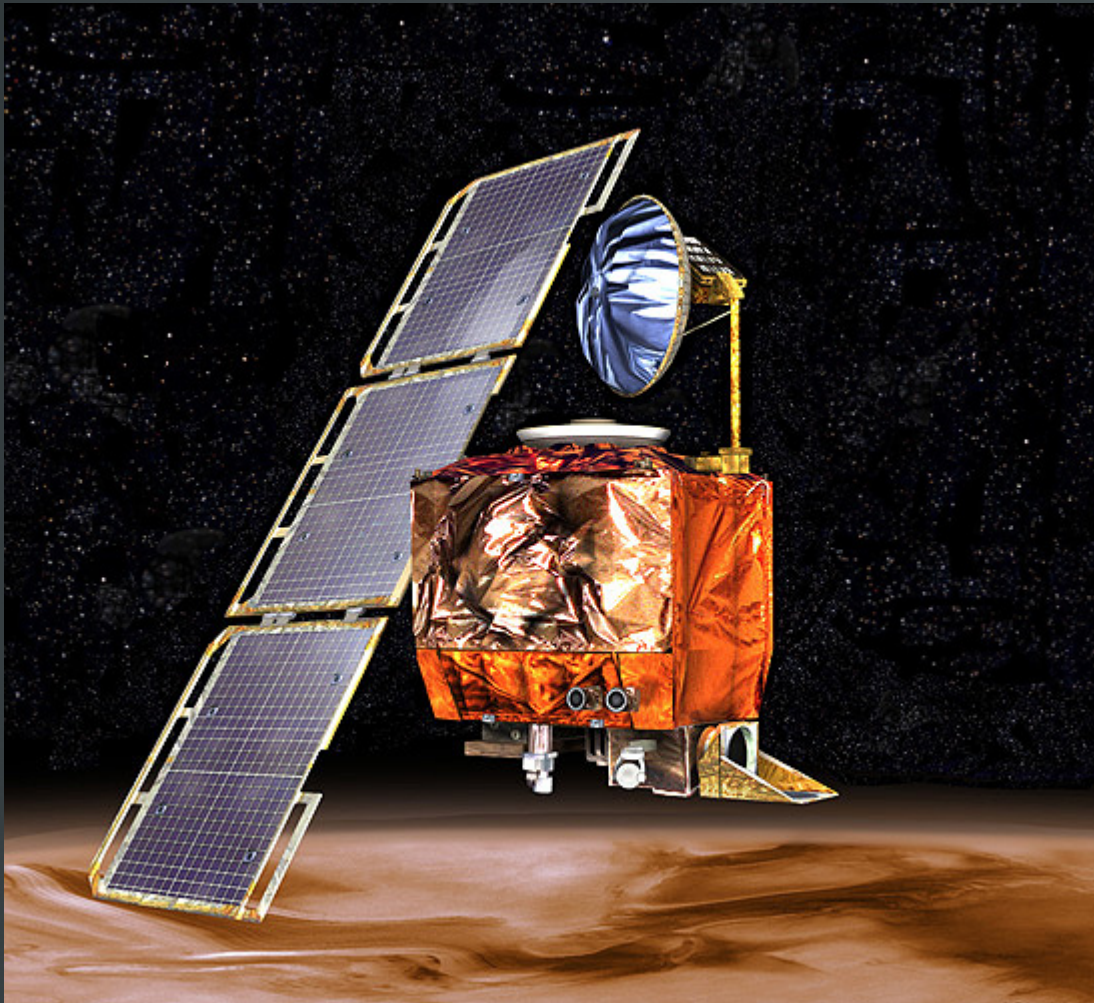
- Team doesn't use a single language (Ubiquitous language).
- Like Telephone or Chinese Whispers game
- People interpret things in their own way
- Things lost in translation



Is obvious really obvious?



Small misunderstanding can cost a lot of money



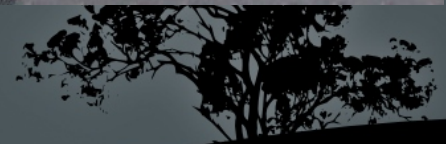
Mars Climate Orbiter Crash

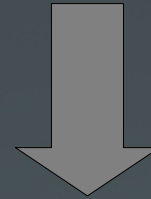
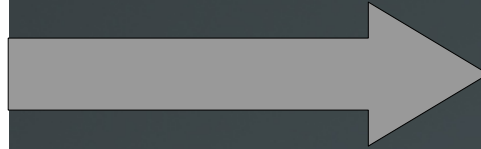
- Total project cost was \$327.6 million
- The metric/imperial mix-up that destroyed the craft was caused by a software error back on Earth.
- The software was working in pounds force, while the spacecraft expected figures in newtons; 1 pound force equals approximately 4.45 newtons.



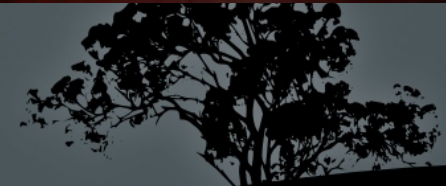
Are we there yet?

Is there a common definition of done?





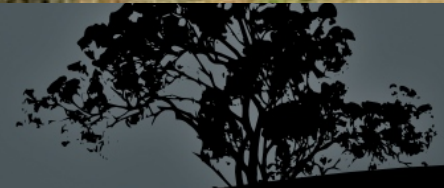
Evolution of
codebase



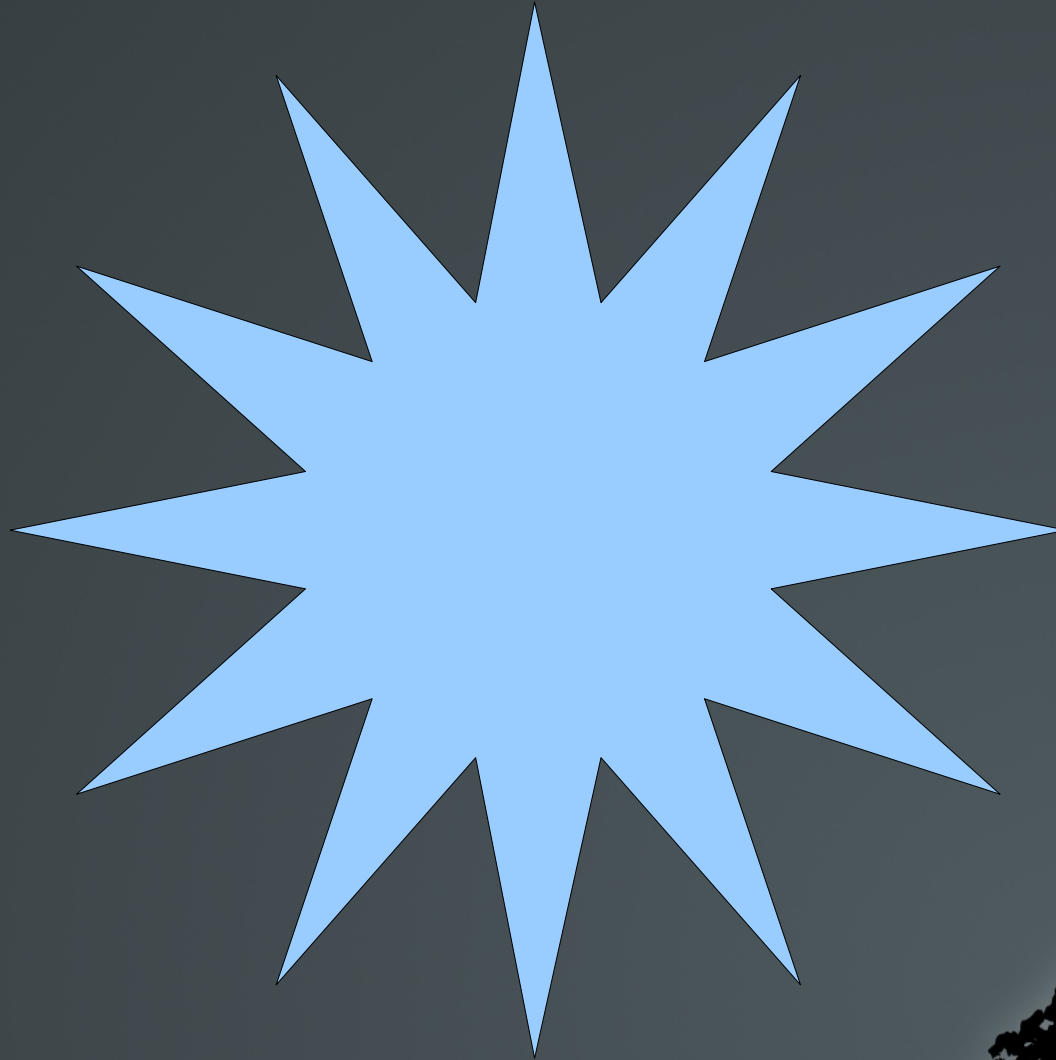
Production Release after manual testing (Scripted Testing)



Knowledge Lost in Time



How many points are there?



What do we need to develop?



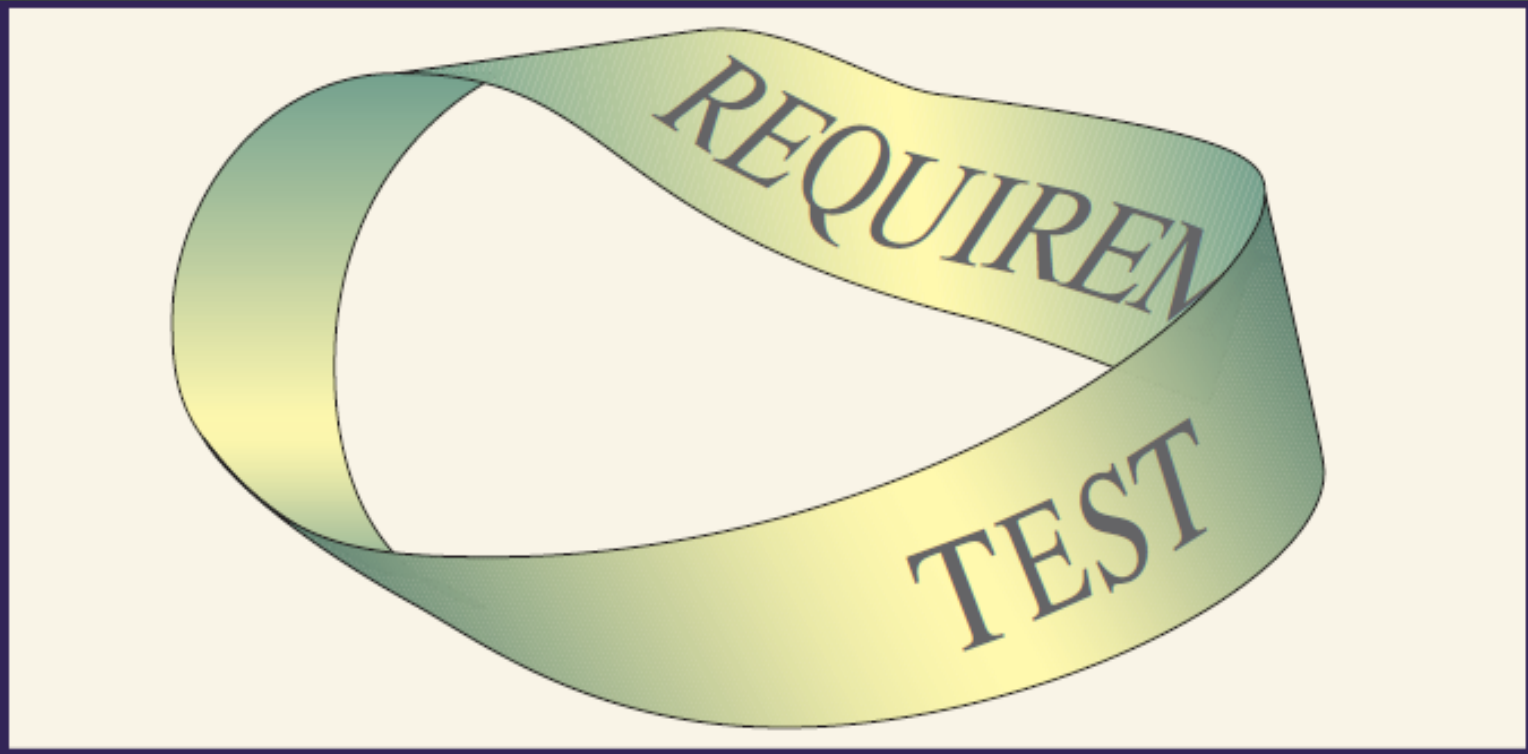
- Client name - Nile book store
- A famous web bookstore. Offer during 2nd year celebration.
 - Any customer who has been buying for past 2 years, living in US and bought books for worth 500 \$ will be awarded 300\$ gift coupon.
 - Any customer who has been buying for past 2 year and a non US customer and bought books worth 300 \$ will get 250 \$ worth gift coupons.
 - Any customer who has been buying for past 1 year, and living in US and bought books worth 1000 \$ will get 200 \$ worth gift coupons.



How to build software that matter?



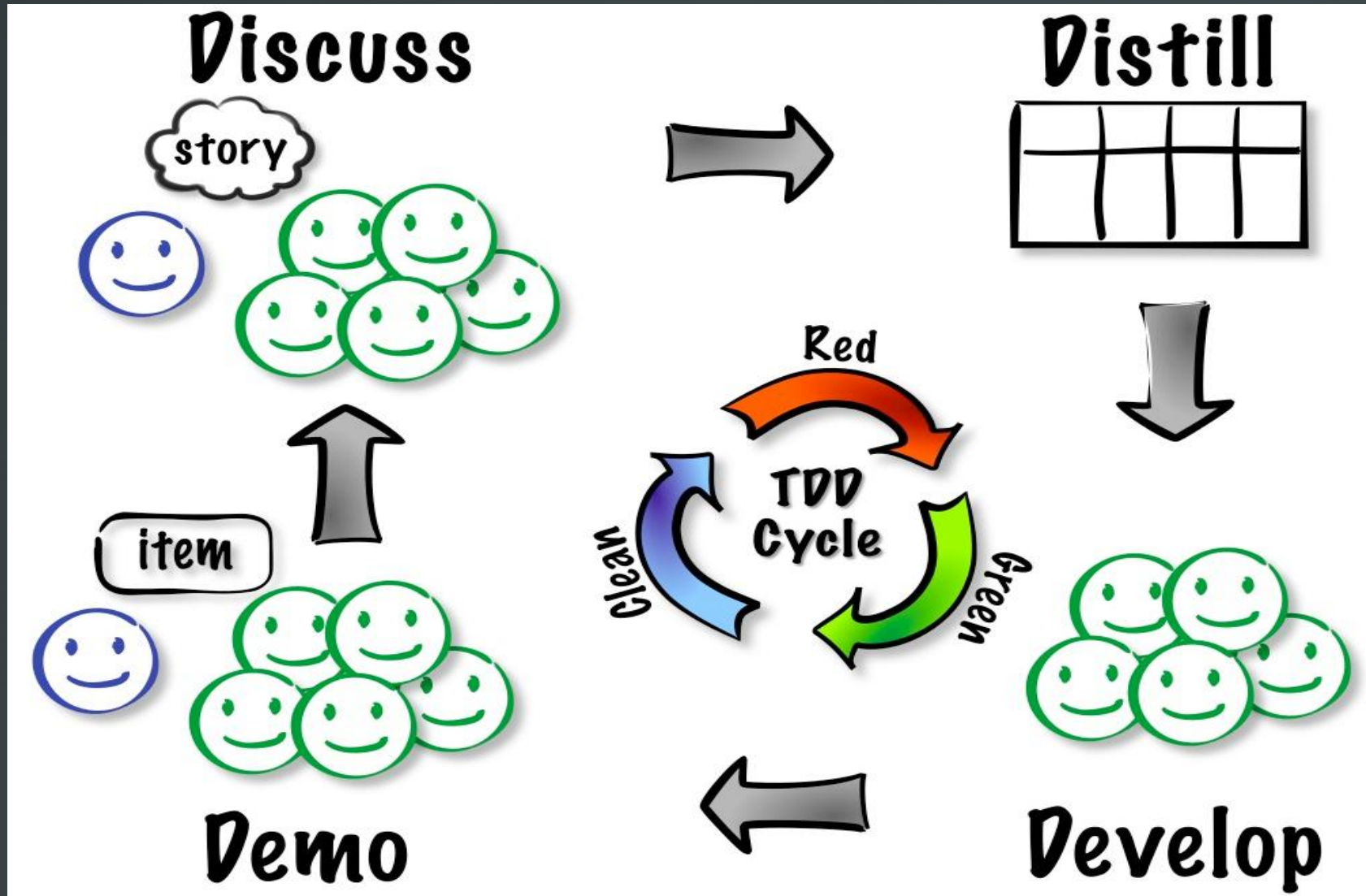
Requirement – Test Paradox



As formality increases, tests and requirements become indistinguishable. At the limits tests and requirements are equivalent – Uncle Bob



Tests as Requirements



Specification workshop

- Discuss and distill requirements, identify gaps, expand examples and get common definition of done
- Should include product owner, domain expert, tester, developer.. or the whole team :)
- Product owner describes his requirement and system behavior in form of concrete examples
- Team questions and discusses the requirements, find alternative ways, and edge cases
- Ends when team reaches an agreement on sufficient number of examples of happy path and edge cases



Specification workshop

- <Picture of table with team discussing>



Specification Development

- The tester with developer creates fixtures to turn the examples into executable acceptance tests.
- The tests initially fail, since the feature is not yet implemented.
- The tests may target the scriptable interface (API) or user interface (or both).
- Only the necessary functionality to pass the tests is implemented (and no more).
- This doesn't in anyway change the normal TDD cycle.




Demonstrate

- A feature which passes the acceptance test has just crossed half the bridge.
- More tests in form on Exploratory testing, performance, usability and more are needed
- A feature is not done until it is deployed and used in production (Continuous Deployment)



What do we get out of this?

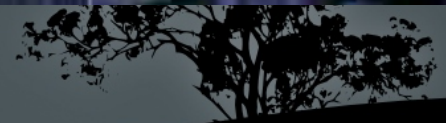
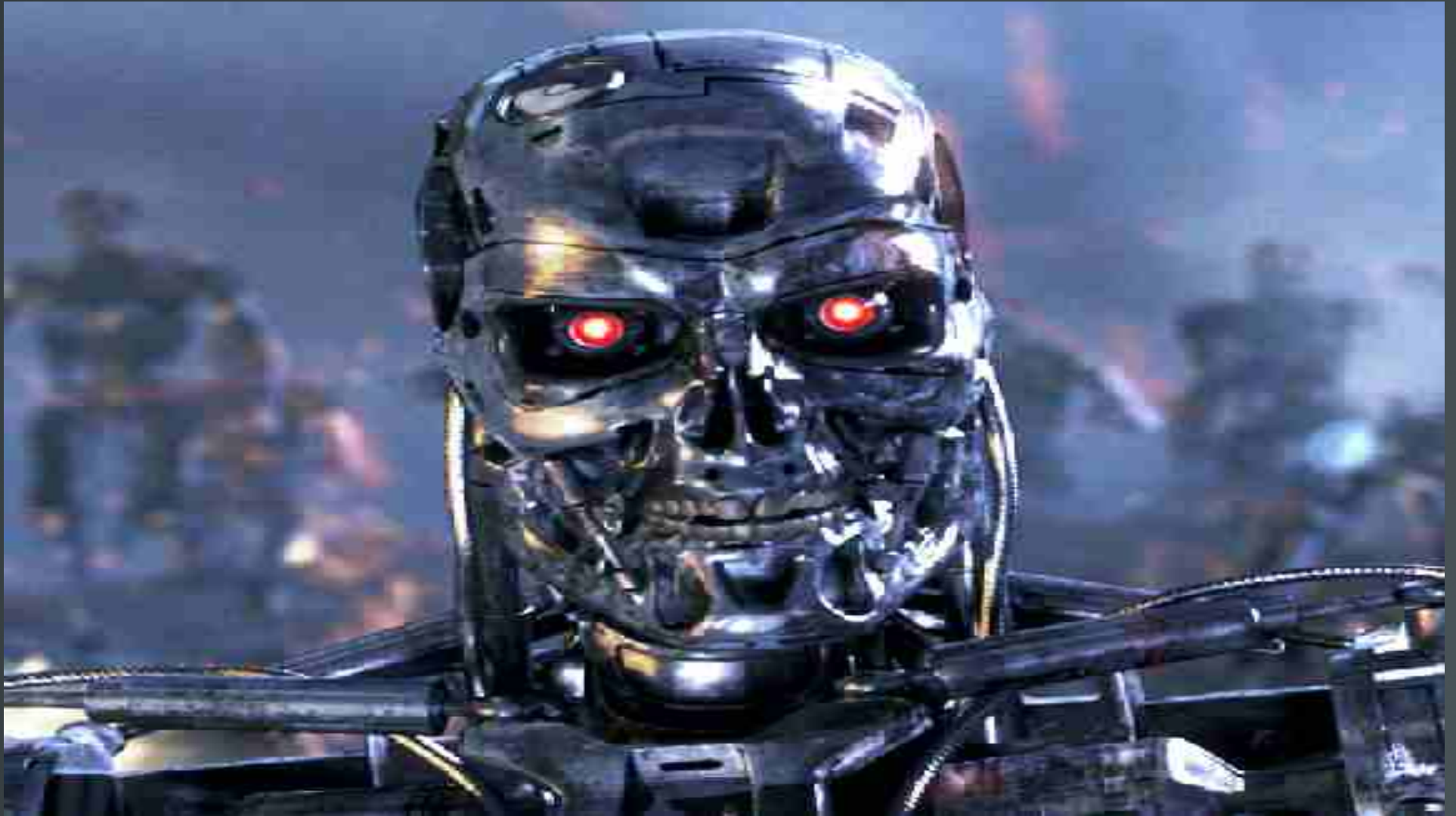
- Executable and living Specifications
 - Better test scripts run as part of build (Side effect)
 - Shared understanding of domain and ubiquitous language
 - Better understanding of intricacies of domain (Helps in Domain Driven Design)
 - Better clarity on what we are building and are we doing it right?
 - Better consensus on when we are done
- 

Gotchas

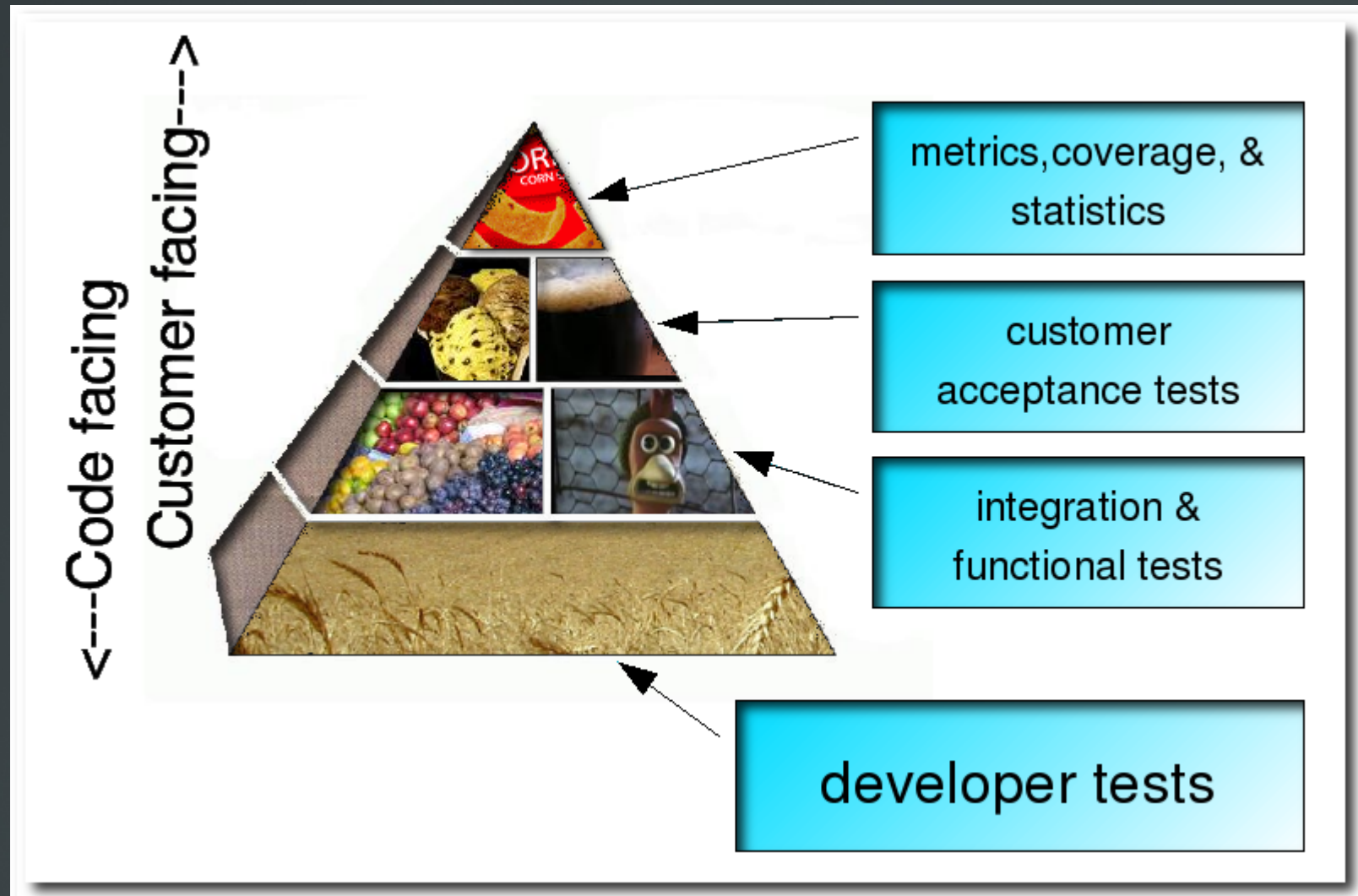
- Existing specifications are often changed
 - You're tied to an implementation
- Lots of "execute" or imperative commands
 - You're writing a script
- Complicated instrumentation
 - You're testing too much in one go
- Complicated fixture code
 - Your fixture code is verbose and hard to follow
- Examples all have the same structure
 - Your examples are too generic



Too much automation



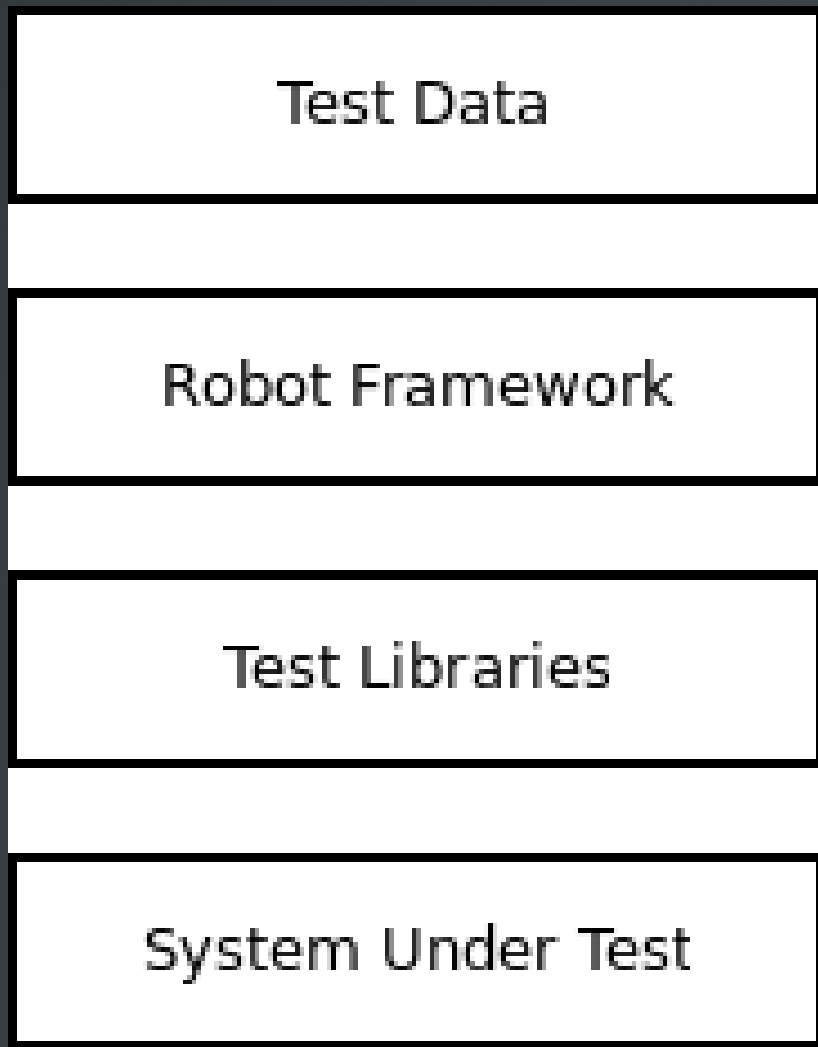
Test Automation Pyramid



Robot Framework Introduction



Test Automation Framework



- Implemented using Python language
 - Python is the recommended language to create test libraries and otherwise extend the framework
- Runs on JVM on top of Jython
 - Possible to implement test libraries using Java



Keyword Driven Approach

Test Case	Action	Argument	Argument
User can create an account and log in	Create Valid User	fred	P4ssw0rd
	Attempt to Login with Credentials	fred	P4ssw0rd
	Status Should Be	Logged In	
User cannot log in with bad password	Create Valid User	betty	P4ssw0rd
	Attempt to Login with Credentials	betty	wrong
	Status Should Be	Access Denied	



Data Driven Approach

Test Case	Action	Password	Expected error message
Too short password	Creating user with invalid password should fail	abCD5	#{PWD INVALID LENGTH}
Too long password	Creating user with invalid password should fail	abCD567890123	#{PWD INVALID LENGTH}
Password without lowercase letters	Creating user with invalid password should fail	123DEFG	#{PWD INVALID CONTENT}
Password without capital letters	Creating user with invalid password should fail	abcd56789	#{PWD INVALID CONTENT}
Password without numbers	Creating user with invalid password should fail	AbCdEfGh	#{PWD INVALID CONTENT}
Password with special characters	Creating user with invalid password should fail	abCD56+	#{PWD INVALID CONTENT}



Fixture Implementation

```
import os
import sys

class LoginLibrary:

    def __init__(self):
        sut_path = os.path.join(os.path.dirname(os.path.abspath(__file__)),
                                '..', 'sut', 'login.py')
        self._command_prefix = "%s" %s " % (sys.executable, sut_path)
        self._status = ''

    def create_user(self, username, password):
        self._run_command('create', username, password)

    def change_password(self, username, old_pwd, new_pwd):
        self._run_command('change-password', username, old_pwd, new_pwd)

    def attempt_to_login_with_credentials(self, username, password):
        self._run_command('login', username, password)

    def status_should_be(self, expected_status):
        if expected_status != self._status:
            raise AssertionError("Expected status to be '%s' but was '%s'"
                                % (expected_status, self._status))

    def _run_command(self, command, *args):
        command = '%s %s %s' % (self._command_prefix, command, ' '.join(args))
        process = os.popen(command)
        self._status = process.read().strip()
        process.close()
```

Higher Level Keyword

Keyword	Action	Argument	Argument
Clear login database	Remove file	\${DATABASE FILE}	
Create valid user	[Arguments]	\${username}	\${password}
	Create user	\${username}	\${password}
	Status should be	SUCCESS	
Creating user with invalid password should fail	[Arguments]	\${password}	\${error}
	Create user	example	\${password}
	Status should be	Creating user failed: \${error}	
Login	[Arguments]	\${username}	\${password}
	Attempt to login with credentials	\${username}	\${password}
	Status should be	Logged In	



Variables

Variable	Value
<code>\${USERNAME}</code>	janedoe
<code>\${PASSWORD}</code>	J4n3D0e
<code>\${NEW PASSWORD}</code>	e0D3n4J
<code>\${DATABASE FILE}</code>	<code>\${TMPDIR}\${/}robotframework-quickstart-db.txt</code>
<code>\${PWD INVALID LENGTH}</code>	Password must be 7-12 characters long
<code>\${PWD INVALID CONTENT}</code>	Password must be a combination of lowercase and uppercase letters and numbers



Tagging

- Free metadata to categorize test cases
- Statistics by tags collected automatically
- Select test cases to be executed
 - --include and --exclude options
- Specify which test cases are considered critical



Clear Reports

Login Tests Test Report

Generated
20080613 14:13:20 GMT +02:00
1 minute 0 seconds ago

Summary Information

Status: **6 critical tests failed**
 Start Time: 20080613 14:12:08.445
 End Time: 20080613 14:12:39.666
 Elapsed Time: 00:00:31.221

Test Statistics

Total Statistics	Total	Pass	Fail	Graph
Critical Tests	10	4	6	
All Tests	10	4	6	

Statistics by Tag	Total	Pass	Fail	Graph
regression	10	4	6	
smoke	4	4	0	

Statistics by Suite	Total	Pass	Fail	Graph
Login Tests	10	4	6	
I.Higher Level Login	3	3	0	
I.Invalid Login	6	0	6	
I.Simple Login	1	1	0	

Test Details by Suite

Name	Documentation	Metadata / Tags	Crit.	Status	Message
Login Tests			N/A	FAIL	10 critical tests, 4 passed, 6 tests total, 4 passed
I.Higher Level Login			N/A	PASS	3 critical tests, 3 passed, 3 tests total, 3 passed
I.h.Higher Level Valid Login		regression, smoke	yes	PASS	
I.h.Even Higher Level Valid Login		regression, smoke	yes	PASS	
I.h.Highest Level Login		regression, smoke	yes	PASS	
I.Invalid Login			N/A	FAIL	6 critical tests, 0 passed, 6 tests total, 0 passed
I.i.Invalid Username		regression	yes	FAIL	Location should have been 'http://localhost/error.html'
I.i.Invalid Password		regression	yes	FAIL	Location should have been 'http://localhost/error.html'
I.i.Invalid Username			yes	FAIL	Location should have been 'http://localhost/error.html'

Login Tests Test Report

Generated
20080613 13:39:08 GMT +02:00
1 minute 5 seconds ago

Summary Information

Status: **All tests passed**
 Documentation: Demo test cases for Robot Framework using Selenium test library
 Start Time: 20080613 13:38:36.191
 End Time: 20080613 13:39:08.068
 Elapsed Time: 00:00:31.877

Test Statistics

Total Statistics	Total	Pass	Fail	Graph
Critical Tests	10	10	0	
All Tests	10	10	0	

Statistics by Tag	Total	Pass	Fail	Graph
regression	10	10	0	
smoke	4	4	0	

Statistics by Suite	Total	Pass	Fail	Graph
Login Tests	10	10	0	
I.Higher Level Login	3	3	0	
I.Invalid Login	6	6	0	
I.Simple Login	1	1	0	

Test Details by Suite

Name	Documentation	Metadata / Tags	Crit.	Status	Message	Start / Elapsed
Login Tests	Demo test cases for Robot Framework using Selenium test library		N/A	PASS	10 critical tests, 10 passed, 0 failed 10 tests total, 10 passed, 0 failed	20080613 13:38:36 00:00:32
I.Higher Level Login			N/A	PASS	3 critical tests, 3 passed, 0 failed 3 tests total, 3 passed, 0 failed	20080613 13:38:36 00:00:17
I.h.Higher Level Valid Login		regression, smoke	yes	PASS		20080613 13:38:36 00:00:06
I.h.Even Higher Level Valid Login		regression, smoke	yes	PASS		20080613 13:38:41 00:00:05
I.h.Highest Level Login		regression, smoke	yes	PASS		20080613 13:38:47 00:00:06
I.Invalid Login			N/A	PASS	6 critical tests, 6 passed, 0 failed 6 tests total, 6 passed, 0 failed	20080613 13:38:52 00:00:10
I.i.Invalid Username		regression	yes	PASS		20080613 13:38:57 00:00:01
I.i.Invalid Password		regression	yes	PASS		20080613 13:38:58 00:00:01
I.i.Invalid Username		regression	yes	PASS		20080613 13:38:59

Similar Tools

- Fit and FitNesse
- Concordion
- BDD Frameworks
 - should_dsl - <http://github.com/hugobr/should-dsl>
 - Behavior - <http://pypi.python.org/pypi/Behaviour/0.1a4>

