Testing Culture &
Helping the Business to Win

Emily Bache
Testing Culture & Helping the Business to Win

Romanian Testing Conference

@emilybache

May 2018
Software is Eating the World

“Six decades into the computer revolution, four decades since the invention of the microprocessor, and two decades into the rise of the modern Internet, all of the technology required to transform industries through software finally works and can be widely delivered at global scale.”
— Marc Andresssen, 2011

https://www.wsj.com/articles/SB10001424053111903480904576512250915629460
Testing Culture & Business Success

My own experiences
Testing Teams

Testers

DevOps Team

Dev Team A

Dev Team B

Operations
(S)-(-)-omeprazole (esomeprazole)

(R)-(+)-omeprazole
Testing Culture & Development Culture
Development Culture & Successful Businesses

Famous examples
Apple’s ‘goto fail’ bug

```c
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
goto fail;
  goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
goto fail;
```

- Skips all subsequent checks after this line
- Enables “man in the middle” attacks on ssl connections

Man-in-the-Middle Attack
The defect in context

- 2000 line file
- Printed in 4 point font in 3 columns, takes 7 A4 pages
- Same algorithm appears 6 times in this file
“The presence of six separate copies of the same algorithm clearly indicates that this bug was not due to a one-time programmer error: This was a pattern. This is evidence of a development culture that tolerates duplicated, untested code.”

http://martinfowler.com/articles/testing-culture.html
Testing Culture and Project success?

https://en.wikipedia.org/wiki/List_of_software_bugs - serious bugs documented in the field

Hard for me to link these to testing culture!
Responding to Change

Loic Le Meur 🌐 @loic
@elonmusk the San Mateo supercharger is always full with idiots who leave their tesla for hours even if already charged.

Elon Musk 🤖 @elonmusk

You're right, this is becoming an issue. Supercharger spots are meant for charging, not parking. Will take action.

5:20 AM - Dec 11, 2016

Updates - in Space


https://www.newscientist.com/article/dn24248-date-glitch-delays-cygnuss-rendezvous-with-iss/
Development Culture & Agility

Embrace Change
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Agility comes from Practices

Development practices  Agility
Agility comes from Development Culture

Development culture → Agility
Agility leads to success?
Development Culture & Helping the Business to Win

Show me the money
Evidence-based Medicine

Statins → lower cholesterol → fewer heart attacks


image attribution: http://www.medicalwp.com/

www.praqma.com
Where's the evidence?

Agile practices

Agility

making money
Measuring what works

“Firms with high-performing IT organisations were twice as likely to exceed their profitability, market share and productivity goals”

IT Performance

• Throughput of code
  • How frequently able to deploy
  • How fast from commit to deploy
• Stability of systems
  • How quickly they recover from downtime
  • How many changes succeed vs fail
Silo Thinking

More frequent releases! We are measured on feature count.

Code Freeze as long as possible! We are measured on test case & bug count.

Less frequent releases! We are measured on uptime & stability.
Tradeoff?

High performers do better at both

Dev <-> Ops

Throughput <-> Stability
Trading off Cost and Quality

Figure 3: U.S. Total Vehicle Sales Market Share by Company, 1961-2015

Source: Ward’s Auto Data

http://econlife.com/2017/05/japanese-american-made-cars/
High Performers

As in previous years, we have found that high performers do significantly better than their lower-performing peers in terms of throughput and stability. In 2017, we found that the high performers have:

- 46 times more frequent code deployments
- 440 times faster lead time from commit to deploy
- 96 times faster mean time to recover from downtime
- 5 times lower change failure rate
  (changes are 1/5 as likely to fail)
Development Culture & DevOps

Becoming a High Performer
Throughput AND Stability

http://web.devopstopologies.com/
State of DevOps Reports

• Annual survey of thousands of IT professionals

• Questions designed to test various hypotheses about DevOps

• Statistical analysis published in free report, and follow-on research in peer-reviewed journals

• Authors: Puppet & DORA
Performance Drivers

https://puppet.com/resources/whitepaper/state-of-devops-report
Continuous Delivery

- Test and deployment automation
- Continuous integration
- Trunk-based development
- Shifting left on security
- Loosely-coupled architecture
- Empowered teams

IT performance
- Organizational performance
- Non-commercial performance

Deployment pain
Significant CD Practices

- Test and deployment automation
- Continuous integration
- Trunk-based development
- Shifting left on security
- Loosely-coupled architecture
- Empowered teams

Frequent commits
No long-lived feature branches

“... it’s easy to modify or replace any individual component or service”

“teams ... don’t depend upon other teams to complete their work.”
<table>
<thead>
<tr>
<th>Survey questions</th>
<th>High IT performers</th>
<th>Medium IT performers</th>
<th>Low IT performers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deployment frequency</strong></td>
<td>On demand (multiple deploys per day)</td>
<td>Between once per week and once per month</td>
<td>Between once per week and once per month*</td>
</tr>
<tr>
<td>For the primary application or service you work on, how often does your organization deploy code?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Lead time for changes</strong></td>
<td>Less than one hour</td>
<td>Between one week and one month</td>
<td>Between one week and one month*</td>
</tr>
<tr>
<td>For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code commit to code successfully running in production)?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mean time to recover (MTTR)</strong></td>
<td>Less than one hour</td>
<td>Less than one day</td>
<td>Between one day and one week</td>
</tr>
<tr>
<td>For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Change failure rate</strong></td>
<td>0-15%</td>
<td>0-15%</td>
<td>31-45%</td>
</tr>
<tr>
<td>For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Note: Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers.
A Lever to move the business

... and I will move the Earth

- Archimedes of Syracuse
A Lever to move the business

- Test and deployment automation
- Continuous integration
- Trunk-based development
- Shifting left on security
- Loosely-coupled architecture
- Empowered teams

Organizational performance
Non-commercial performance
Measure Your Performance

• Throughput of code
  • How frequently able to deploy
  • How fast from commit to deploy
• Stability of systems
  • How quickly you recover from downtime
  • How many changes succeed vs fail
Testing Culture & Helping the Business to Win

Evidence & Statistics

Dev Culture: Duplication

Feedback code: #E568

Testing Culture: in my experience

State of DevOps Report

- Test and deployment automation
- Continuous integration
- Trunk-based development
- Shifting left on security
- Loosely-coupled architecture
- Empowered teams

Continuous Delivery

@emilybache