Testing In Production

the many flavors

Keith Stobie
Doyenz
(while at Microsoft)
Pre-Requisites (risk reduction)

• **Redundancy** – state quickly recovered if catastrophic failure,
• Data corruption or state-related **failures** extremely **unlikely** *(functional testing must first be passing)*,
• Errors must be detected and the engineering team (rather than operations) must be **monitoring system health** of the code in **test**
• Can quickly **roll back** all changes **Rollback tested** before going into production
Isolate Test Data

- Don’t let test data affect Production metrics. Can’t bill for test data!
- Number of users requests must be distinct from test requests.
- Is test data visible? Fake products?
Exposure Control

Growing visibility (exposure / risk)

- Opt-in: Preview, Beta, . . .
- Google: Internal, Friends & Family, Beta, . . .
- % of Machines/DCs: 1%, 5%, 10%, 25%, 50%

- Markets: US, Europe, Asia, America
Asserts == Testing

• Leave asserts that don’t significantly impact performance.
  Assert(inputRef != null)
  *(if not in frequent path)*

• Self-test, like hardware, some software has low-pri thread for idle times to walk and verify data structures, etc.
Monitoring == Testing

- Example: In Testbed, is the response time < threshold(s)?
- Monitor same thresholds in production:
  - External “synthetic” transaction
  - Real time data collection
- Transactional availability
- Feature availability
- “pure” monitor is a test Oracle

<table>
<thead>
<tr>
<th>% of requests</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>100ms</td>
</tr>
<tr>
<td>90%</td>
<td>200ms</td>
</tr>
<tr>
<td>99%</td>
<td>300ms</td>
</tr>
</tbody>
</table>
Delta Assertion (Meszaros)

Image from: http://xunitpatterns.com/Delta%20Assertion.html
Model of a Service

User Requests → Frontend service → Service → Backend service → Data → Data Feed

Testing in Production -- Keith Stobie

Copies may not be made or distributed for commercial use
Test in Production Environment

(Offline) Testing in Production Environment

User Requests → Frontend service → Service → Backend service → Data Feed

Production Environment
(Identical Hardware, operations, etc.)
Fork(copy) Traffic (drop test return)
Experiment with Graphics/Color

• Which one converts (to search) better?

**National Alert Registry**

**Does a sexual offender live in your neighborhood?**

Free search for sex offenders in your area:

Registered Sex Offender Search
Enter Zip Code: 
Email Address: 

Perform Sex Offender Search
(average search takes less than 5 seconds)

Source: Marketing Experiments
http://www.marketingexperiments.com
A/B Test

Visitors Randomly Distributed

50%

CTR

Version B is better than Version A

Version A
(Control)

1.2% of users with a Page View clicked on Signup

User interactions instrumented, analyzed and compared

Version B
(Treatment)

2.8% of users with a Page View clicked on Signup

Is the observed difference statistically significant?

YES

Version B is better than Version A

Click Thru Rate

User interactions instrumented, analyzed and compared

1.2% of users with a Page View clicked on Signup

2.8% of users with a Page View clicked on Signup

Figure courtesy of Seth Elliot from his SASQAG (Seattle Area Software Quality Assurance Group) April 2011 Talk Testing in Production - Your Key to Engaging Customers
Experiments (alternate return)

Separate experimental service providing the 2nd alternate output
Experiments (alternate code)

Service architected internally to have code paths for different alternate output

2011-10-10 PNSQC

Copies may not be made or distributed for commercial use
Experiments (alternate data)

Sometimes it is just the data, e.g. different ranking values that are being tried.
Shadowing (compare returns)

User Requests → Frontend service → Service → Backend service → Data Feed

Production Environment

Equally good? → Service

Future Environment

Data

Test

Next Environ

Production
Fake Traffic (capacity test)

Capacity test a single service, or entire offering (via the frontend)
Modern Cap Generation Toolset

1. Controller drives each of the clients across any/all DCs
2. Capacity is generated within each datacenter to avoid cross-DC network utilization
3. Controller pulls data from monitoring system auto-throttling based on SLA
4. 2-3 engineers required for a typical full-scale test
Tracers (data flow)

Observe test data as it flows across components and out of the system.
1. Product search page displays product when users queries for it?
2. Review search page displays reviews when users queries for it?
3. Product details page for a given product contains matched offer and review information?
Fault Injection

Data corruption
Network latency

User Requests
Frontend service
Service
Data
Backend service
Data Feed

Image from: http://behance.vo.llnwd.net/profiles/78972/projects/114549/789721221731305.jpg
# Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>Risk of Causing issues</th>
<th>Exposure Control</th>
<th>Data Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asserts</td>
<td>Medium</td>
<td>Optional</td>
<td>--</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Low</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Offline</td>
<td>Low</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Controlled Experiment</td>
<td>Medium</td>
<td>Required</td>
<td>Useful</td>
</tr>
<tr>
<td>Live Capacity</td>
<td>High</td>
<td>Useful</td>
<td>Required</td>
</tr>
<tr>
<td>Tracers</td>
<td>Medium</td>
<td>Useful</td>
<td>Useful</td>
</tr>
<tr>
<td>Fault Injection</td>
<td>High</td>
<td>Optional</td>
<td>Useful</td>
</tr>
</tbody>
</table>
Testing in Production