Our Road to Continuous Delivery @ Tango
Amit Mathur
Tango Overview

- Founded in 2009 and headquartered in Mountain View, Tango has over 300 employees.
- Tango is a messaging platform that combines communications, social, and content.
- We combine chat, video calling, games, social discovery, and other entertaining content.
- 300M+ people use Tango to keep in touch with friends and family.
- Tango has raised $370M to date with the most recent Series D funding of $280M led by Alibaba.
Continuous Delivery

- Commit Code
- Unit & Feature Test
- Deploy to Testing Environment
- Acceptance Test
- Deploy to Production Environment

Continuous Integration

Continuous Delivery

Continuous Deployment
Continuous Delivery Goals

- Monthly Release Cycle -- Enable 2 week production client bake time for new releases by certifying server components in 2 weeks
- Minimize regressions due to new servers deployed in production
Certification Flow

1. Testing in Feature Branch
2. Landing Feature Branch to Trunk
3. Release Branch Creation and Stabilization
4. Certify Server Components
5. Server Components Deployed to Production
6. Internal Beta Testing of Clients Pointing to Production
7. Staged Android Rollout
8. iOS Rollout

2 weeks 2 weeks
Continuous Delivery Automation

Diagram:
- Deployment Automation
- Continuous Testing
- Test Intelligence

Processes:
- Virtual Machine
- Mosh Cluster Manager
- Mosh client
- Zydeco Web Service
- Zydeco Continuous Test
- Zydeco Lib
- Salsa Real Device Continuous Test
- TEA Lib
- Tango Engineering Analytics System
- Limbo Test Quality System
- TEA DB
Environment Utilization

DEV1 | INT1
---|---
DEV2 | INT2
DEV3 | INT3

UAT | STA | PROD

Feature branch
Components changes + other components at production level

Release branch
Deployment Sequence and MIX mode testing

Trunk
All features branches combined + other components at production level

Feature branch
Components changes + latest server components
Deployment Automation (mosh)
Tango Test Automation System

UI
- Cross Platform
- Client

Server
- Cross Platform Unit Tests
- Server Unit Tests
- Android/iOS UIAutomation Test
- FeatureTest
- Continuous Delivery Test

Tango

Test Analytics Web Service
Test Analytics UI Dashboard
Analytics DB
Deployed Environment

Jenkins

Jenkins
Test Automation Investments

- Unit
- Feature
- Continuous Delivery
- UI

Shared Environment

Sandbox

Complexity
Continuous Delivery Test

- **RESTful** test that adheres to the following requirements:
  - **Environment Mobility** -- Should be able to run in any environment including localhost
  - **Non-intrusive** -- Should clean up all garbage created by the test and should not interfere with other test instances (including itself)
  - **Security** -- Must not expose environment secrets (test could be run in production environment)
# Test Intelligence

## NEW SCENARIO STATUS

<table>
<thead>
<tr>
<th>Scenario Description</th>
<th>STA-1</th>
<th>STA-2</th>
<th>INT-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels can be subscribed.</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Contact filtering after adding a Tango User</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Read Receipt functionality</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SDK Authentication process</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Photo upload to TC</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Feed functionality.</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Making a video call to the Android robot</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Registration Flow on Android.</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Discovery service</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

* TEST IN PROGRESS

Availability in last 60 minutes

As of Thu Oct 08 2015 - 16:33:51

## COMPONENT STATUS

<table>
<thead>
<tr>
<th>Component</th>
<th>STA-1</th>
<th>STA-2</th>
<th>INT-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>marketplace</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>sdkStorage</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>chatroominator</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>tc2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>facilitator</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>adinfo</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>abregistrar</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Policy</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>bigerator</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Scream</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>acme</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>locatorater</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>preferenator</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>channelsfascade</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>order</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>discovery</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

* TEST IN PROGRESS

Availability in last 60 minutes
Test Quality Problem

- # of automated tests running continuously has dramatically increased
- Flaky tests have become a serious problem
- How do we deal with them?
## Gamification Experiment

<table>
<thead>
<tr>
<th></th>
<th>Test Coverage</th>
<th>Test Execution</th>
<th>Test Quality</th>
<th>Product Quality</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual vs. Desired Automated Tests</td>
<td>Outcome of Tests Run In Last 7 Days</td>
<td>Defects in Automated Tests</td>
<td>Defects in Product</td>
<td>Weighted Average of All Scores</td>
</tr>
<tr>
<td>MAD</td>
<td>95.24</td>
<td>61.9</td>
<td>95.24</td>
<td>95.24</td>
<td>86.91</td>
</tr>
<tr>
<td>GEM</td>
<td>44.0 ▲</td>
<td>70.0 ▲</td>
<td>96.67 ▲</td>
<td>96.67 ▲</td>
<td>76.84 ▲</td>
</tr>
<tr>
<td>EVIL</td>
<td>42.0 ▲</td>
<td>81.82 ▼</td>
<td>86.36</td>
<td>90.91</td>
<td>75.27 ▼</td>
</tr>
<tr>
<td>T1</td>
<td>87.5</td>
<td>57.4 ▼</td>
<td>54.25</td>
<td>85.71</td>
<td>71.22 ▼</td>
</tr>
<tr>
<td>Dragon</td>
<td>40.0 ▲</td>
<td>69.51 ▼</td>
<td>85.06 ▲</td>
<td>90.24 ▲</td>
<td>71.2 ▲</td>
</tr>
<tr>
<td>ROCK</td>
<td>34.44</td>
<td>60.23 ▼</td>
<td>22.35</td>
<td>92.42</td>
<td>52.36 ▼</td>
</tr>
</tbody>
</table>
Test in Quarantine if:
Analysis Rate < 90% or False Positive Rate < 90% in 24 hour period
Key Takeaways

- Deployment automation is key as it helps reduce drift in test environments
- Having continuous delivery tests (UI & server) are key but invest in the right amount of each
- Quarantine those flaky tests!