GlobalLogic®

Testing in Production: The Future of Testing

Kapil Saxena, Test Engineering

Noida, India

When it comes to software services testing, there are typically two opposing camps: those who prefer testing in a closed lab environment to make it as close to production as possible, and those who prefer testing in production (TiP). Although this second approach may seem like a riskier process, it actually enables an organization to quickly detect and respond to problems before they impact users. This white paper explores various TiP methodologies and provides real-world examples of how leading software services companies are deploying them.

www.globallogic.com

Table of Contents

Introduction

Although many testers have been burned by the infamous "missed bug," the frustrating truth is that no test lab can ever totally emulate a production environment. From purchasing the same servers and load balancers to executing sanitized dumps of production data into the test lab, testers can only approximate the production environment to a certain degree through a big up-front testing (BUFT) approach.

Even with a thorough understanding of the production environment and the ability to anticipate edge cases, the unpredictability of real-world users can create unexpected issues. Furthermore, data centers are themselves hugely complex systems with myriad interactions between servers, networks, power supplies, and cooling systems.

But accepting the limitations of a test lab does not mean accepting a buggy product. Nor does it mean that a certain amount of up-front testing is not still important – just not "big" up-front testing. While there are plenty of scenarios that test well in a lab, organizations will only experience diminishing returns by trying to perfectly simulate a production environment. By accepting that some bugs will always leak into production, organizations can instead focus their resources on how to mitigate risks once those bugs are out.

What is Testing in Production?

The goal of testing in production (TiP) is to quickly detect bugs in a production environment in order to minimize their impact on users. By making a production environment more test-friendly, testers can ultimately improve the overall quality of a product or software service.

For some TiP methodologies, you can further reduce risks by limiting the exposure of the new code that is being tested. This technique is called "Exposure Control" and limits the user base that is potentially impacted by the new code. We will discuss more TiP methodologies in the next section.

TiP Methodologies

As an emerging trend, TiP nomenclature and taxonomy are far from finalized. That being said, eleven TiP methodologies have already been identified, as demonstrated below.

Ramped Deployment
- Launch of new software by first exposing it to a subset of users, then steadily increasing user exposure
- Purpose is deployment and may include assessment
- Users may be hand-picked or aware they are testing a new system

Controlled Test Flight
- Parallel deployment of new and old code
- Random, unbiased assignment of unaware users to each set of code
- Purpose is to assess quality of new code and possibly deploy it
- May be part of ramped deployment

Experimentation for Design
- Parallel deployment of new and old user experiences
- New user experience is usually well-tested prior to experiment
- Random, unbiased assignment of unaware users to each user experience
- Purpose is to assess business impact of new experience

Dogfood/Beta
- User-aware participation in using new code, often by invitation
- Feedback may include telemetry but is often manual/asynchronous

Synthetic Test in Production
- Execution of functional test cases (using synthetic data and usually at API level) against in-production systems
- "Write once, test anywhere" is preferred (i.e., same test can run in test and production environments)
- Synthetic tests in production environment may use production monitors/diagnostics to assess pass/fail

Load/Capacity Test in Production

- Injection of synthetic load onto production systems, usually on top of existing real-user load
- Purpose is to assess systems capacity
- Requires careful (often automated) monitoring of SUT and back-off mechanisms

Outside/In Load Performance Testing

- Injection of synthetic load at (or close to) same point-of-origin as user load from distributed sources
- Purpose is to measure end-to-end performance of one or more cycles from user to SUT and back to user again

User Scenario Execution

- Execution of end-to-end user scenarios against live production system from (or close to) same point-of-origin as user-originated scenarios
- Results assessed for pass/fail
- May also include manual testing

Data Mining

- Test cases search through real user data to find specific scenarios
- Scenarios that fail their specified oracle are filed as bugs (sometimes in real-time)

Destructive Testing

- Injection of faults into production systems (e.g., services, servers, network) to validate service continuity in the event of a real fault

Product Validation

- Continuous monitoring (or monitoring upon deployment) of production environment for file compatibility, connection health, certificate installation and validity, content freshness, etc.

TiP in Action

Software services such as Gmail, Facebook, Netflix, and Bing especially benefit from TiP because:

- Users do not (or do not have to) install desktop applications to use them.
- The software provider controls when upgrades are deployed and which features are exposed to users.
- The provider has visibility into the data center running the service and can therefore grant access to system data, diagnostics, and even user data subject to privacy policies.

Let’s take a deeper look at how Google, Facebook, Netflix, and Microsoft leverage certain TiP methodologies to improve these offerings.

Experimentation for Design is a variation of Controlled Online Experimentation, sometimes known as A/B Testing. In this methodology, changes to the user experience (e.g., different messaging, layout, controls) are launched to a limited number of unaware users. Measurements from both the exposed users and the unexposed (control) users are collected to determine whether or not the new proposed change is a beneficial one. Both Bing and Google make extensive use of this methodology. According to Eric Schmidt, former Google CEO, “We do these 1% launches where we float something out and measure that. We can dice and slice in any way you can possibly fathom.”

Controlled Test Flight is also a variation of Controlled Online Experimentation, although it actually tests the new version of the software service instead of just a new user experience. Often both Controlled Test Flight and Experimentation for Design methodologies are executed at the same time to assess both the quality of the new change and its impact on users.

For example, when rolling out new code, Facebook analyzes both user behavior (i.e., the percentage of users who engage with a Facebook feature) and error logs, load, and memory. Facebook also launches its new code in stages, from an internal release to a small external release to a full external release. Testing a release internally like this can also be considered part of the Dogfood TiP methodology.

Controlled Test Flight can also be enabled via a TiP technique called “Shadowing,” in which new code is exposed to users, but users are not exposed to the code. When Google first tested Google Talk, the presence status indicator presented a challenge for testing because the expected scale was billions of packets per day. To overcome this challenge, Google created a scenario within Orkut (a similar Google product) where users unknowingly triggered presence status changes in the backend servers, thus allowing Google’s engineers to assess the health and quality of the system.

This approach also utilized the previously discussed Exposure Control technique because at first, only 1% of Orkut page views triggered the presence status changes before slowly ramping up.

Destructive Testing, in which you deliberately "kill" the services and servers running your production software, might sound like a recipe for disaster. But you will certainly experience random and unexpected faults in any real-world software service of scale. In one year alone, Google expects to see twenty rack failures, three router failures, and thousands of server failures. If these failures are certain to occur, it is a tester's duty to ensure that the software service can handle them.

A good example of this methodology can be seen with Netflix's "Simian Army," a set of destructive scripts that the company deploys to simulate various failures. Its Chaos Monkey script randomly kills instances and services within Netflix's production architecture; Latency Monkey induces artificial delays; Conformity Monkey finds instances that do not adhere to best practices and shuts them down; and Janitor Monkey searches for and disposes of unused resources.

Synthetic Tests in Production may be more familiar to testers who are new to TiP because it requires them to essentially run the same tests they have always run, albeit against production systems. However, testers must be infinitely more aware of their impact on real-time users since they no longer have the safety net of an isolated test environment. Proper test data management is essential in TiP. Real user data should not be modified, and synthetic data must be identified and handled in such a way as to avoid contaminating production data. Testers also cannot depend on "clean" starting points for the systems under test and their environments.

The Microsoft Exchange team faced this challenge when it had to copy a very large and complex enterprise product to the cloud to run as a service – all while continuing to support the company's enterprise "shrink-wrap" product. The team ran 70,000 automated test cases on a 5,000-machine test lab and took the following steps to manage the process:

- Re-engineered its test automation by adding another level of abstraction to separate the tests from the machine and environment on which they run
- Created a TiP framework on Microsoft Azure to run the same test in the lab for the enterprise edition and in the cloud for the hosted service version
- Leveraged the elasticity of the cloud to (a) run tests continuously, not just upon deployment, (b) use parallelization to run thousands of tests per run, and (c) collect and display data in scorecards to provide continuous evaluation of quality and service availability

Conclusion

Production environments have traditionally been off-limits to testers because organizations feared the impact of testing on real-time users. However, bugs will inevitably leak into any production environment, regardless of the precautions taken in the test lab. Instead of investing in the near impossible dream of a perfectly mirrored test environment, it is much more effective to combine up-front testing with testing in production. By using sound TiP methodologies, testers can quickly identify and eliminate bugs in real-time, thus improving the overall quality of a product or software service.

References

1. How Google Fuels Its Idea Factory, BusinessWeek http://www.businessweek.com/magazine/content/08_19/b4083054277984.htm

2. How Facebook Ships Code, FrameThink http://framethink.wordpress.com/2011/01/17/how-facebook-ships-code/

About GlobalLogic Inc.

GlobalLogic®

Contact

Emily Younger
+1.512.394.7745
emily.younger@globallogic.com