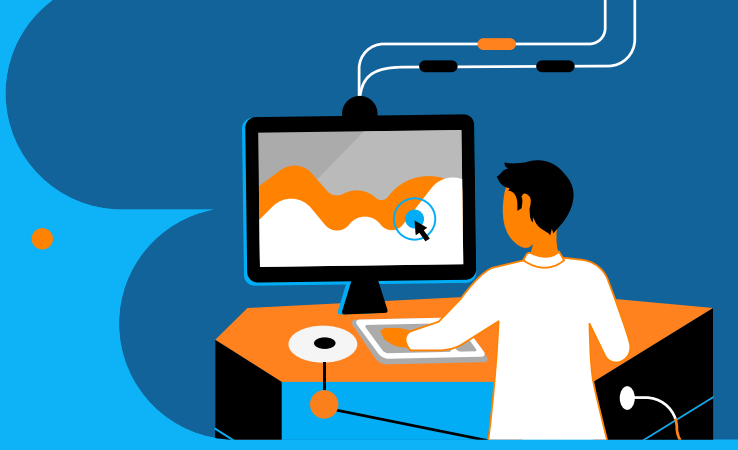
Plutora™



WHITE PAPER

Using Test Environment Management to Reduce Costs and Eliminate Defects

Highlights

- When companies are driven to release production software at an ever-faster rate, software testing can become compromised software releases persist.
- Lack of testing results in defects, angry customers, and higher costs.
- Test Environment Management (TEM) streamlines the testing process, allowing developers to fully and rigorously test their applications.
- Plutora's Test Environment Management solution helps TEM scale to larger enterprises.

Software development does not occur in a vacuum. Aside from the significant cross-team collaboration requirements, the tools and techniques that underly a successful deployment into production pull from a broad range of disciplines and experiences.

Unfortunately, to many enterprises, software testing is one of those aspects of development that is overlooked despite its obvious importance. Why? Because, lacking a thorough familiarity with TEM, most managers assume that benefits gained will be remote while costs incurred will be immediate.

The truth is not so. With a bit of know-how, adherence to the best practices, and the right tools, test environment management can be implemented with little hassle and tremendous savings.

Introduction

Over the past several years, the tools and techniques available to software developers have become more numerous, and more refined. Agile has inspired Scrum, Kanban, Scrumban, LSD, and so on. Flash and JavaScript have been supplanted in their turn by HTML5, Swift, Hack, et al. Yet, in spite of the technological and organizational advantages in the software development industry, a critical arena remains in a relative backwater: software testing.

Software testing is one of those unfortunate disciplines which lie on the cusp between "important to perform" and "easy to neglect." If an application can run on a developer's machine without bugs, the thought process goes, surely it must run equally well on most end-user's devices. While this thought process has obvious fallacies, it is nonetheless a fact that developers can and will sacrifice the testing process to deliver their product faster.

As an example, security testing has gone completely by the wayside. According to a Ponemon Institute report on application security published in March 2016, only 14 percent of the 630 respondents reported that their organizations performed application security testing at every stage of the software development lifecycle.

Neglected software testing might give a company the lead in terms of time-to-market, but that neglect also delivers a long tail of negative externalities in terms of zero-day vulnerabilities, reduced productivity associated with a malfunctioning apps, and negative customer experiences.

The solution, therefore, is not to abandon testing, but rather to test in a way that is fast, comprehensive, and cost-effective—in short, to deploy managed testing environments.

The State of Test Environments in 2016

Test Environments currently exist in a rather disorganized, immature form. As stated, a lot of early-stage test environments get set up and run on local machines. Developers have total control over these machines, and thus the testing process goes rather well.

As the development process inches towards production, however, environments tend to get more complex. As software releases move through various phases such as system integration testing, non-functional testing, user acceptance testing and into staging, the complexity of the environments and associated costs with supporting these environments can grow exponentially.

As software releases move through various phases such as system integration testing, non-functional testing, user acceptance testing and into staging, the complexity of the environments and associated costs with supporting these environments can grow exponentially.

It is therefore important that these finite and expensive resources are managed appropriately to not only reduce errors caused by misconfiguration but also to improve the shared use and utilisation of these environments. Contention becomes a problem when multiple teams have to share one testing environment, and there are constant changes to the environment
and squabbles over who gets to use it, creating inefficiencies.

In yet another example, a dev team creates their testing environment on their company's AWS implementation. Once they've finished, they don't need it again right away, but they might need it soon. They leave it running, driving up costs.

There are many further examples, but here's the gist: The use of test environments in the enterprise does not currently take efficiency, collaboration, or cost-effectiveness into account.

The Basics of Test Environment Management

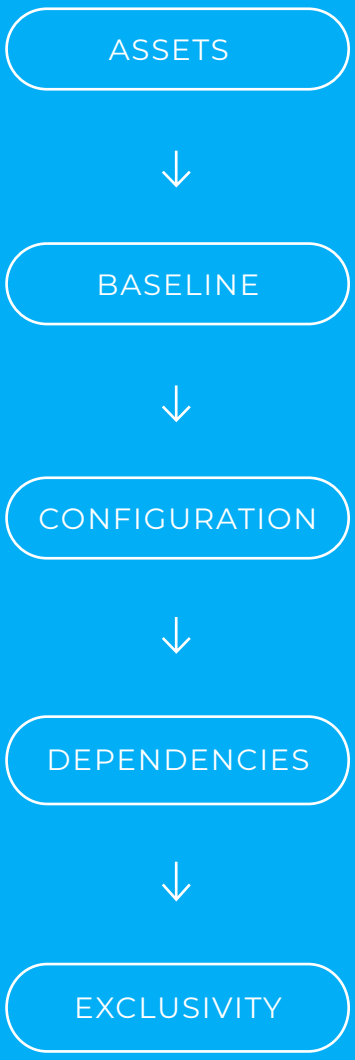
In order to successfully compete in the software development arena, one must reduce obvious inefficiencies in time and cost. This isn't a developer's job, and when developers manage testing environments ad hoc, inefficiencies creep into the process. Here's a better approach: let's take a look at how to utilize Testing Environment Management (TEM) in order to streamline the software testing process.

As a caveat, this is basics for beginners, and should not constitute a roadmap for implementation at larger enterprises.

If an organization is starting to implement TEM midstream, then they most likely have several environments already in place. The first step is then to audit their pre-existing assets. These assets are comprised of specific hardware and software. These components are the baseline, and the way they are set up is your configuration. Some environments will be subsets of other environments, or will be running on the same hardware or software platform. These are dependencies. Lastly, the TEM organization will want to consider exclusivity – whether or not the environment can be shared.

Together, these factors make up an ABCDE approach to auditing test environment infrastructure: Assets, Baseline, Configuration, Dependencies, and Exclusivity.

Together, these factors make up an ABCDE approach to auditing test environment infrastructure: Assets, Baseline, Configuration, Dependencies, and Exclusivity.



As an aside, an enterprise with thousands of people may include dozens or hundreds of test environments. The auditing process will, of necessity, be slow and methodical. Changes may be made in testing environments while the audit is in process– these must be incorporated into your baseline as they occur.

Once you have a thorough understanding of the testing environments that are under your control, the next step is to follow through and actually manage them. There will be several teams, several projects underway, and a limited number of testing environments: who needs what, when?

Start with a calendar. Again, the management process will most likely begin while several projects are already in-flight. The first iteration of the timetable will show which teams have control over what assets, and how long they're going to keep them. At this point, one of the examples from the section above might be encountered: two teams squabbling for control of an asset that they both need.

Once these conflicts are taken care of—and unfortunately, there's no definitive roadmap as to how—the TEM team will be able start anticipating future demand. Once it comes time to meet this demand, the team will begin to actually exercise their power.

As time goes on, the team might discover that there simply aren't enough environments to allow developers to get thoroughly-tested software out the door in a reasonable amount of time. Thus, it may become necessary create a new environment to meet demand. On the other hand, the team may find that a testing environment is underused, and then delete it.

A dev team will enter the application testing phase, and request one or more testing environments for their use. They might also find that there isn't currently an available testing environment that fits their needs. They may request that an existing environment be changed or updated in order to suit their purposes.

The responsibilities of a test environment manager at their broadest definition include: creating, deleting, assigning, and updating.

The goal is to create a system that ensures developers are never under-resourced, and that testing assets are never under-utilized.

Creating, deleting, assigning, and updating – these are the responsibilities of a test environment manager at their broadest definition.

Benefits of a Mature Testing Environment Management Model

As the ability to manage testing environments matures, the TEM team will produce increasing benefits in terms of lowered costs, heightened productivity, and fewer bugs in production apps.

In the past, if a development team couldn't get access to testing environments on time, that would mean that they had a smaller amount of time to test an application. It wouldn't mean that an application would be delayed – because enterprises will no longer flex release dates to accommodate for a lack of testing. The team would simply have to make do, and then fix any defects that arose after the app went into production.

Test Environment Management, when done correctly, remediates this issue by providing one or several appropriately-configured testing environments exactly when they are needed. Even by providing this single benefit, TEM allows development teams to deliver higher-quality production software without lengthening the software development lifecycle. It also saves time and money post-release, as teams will no longer have to devote as much time or as many resources to chasing down bugs and vulnerabilities.

As time goes on, a TEM team will begin to develop expertise in the configuration and deployment of testing environments. They will be able to provision test environments more rapidly, extending the benefit shown above. It also means that they will be able to configure test environments with higher accuracy and increased uptime.

Accuracy is of extreme concern in software testing. It is necessary to replicate as much of the production environment as possible, even down to details such as live server traffic. If these details can't be replicated accurately, then developers simply have to wait until their app is in production and see if errors crop up. If testing environments are being assembled by developers on an ad-hoc basis, it's likely that the environments that they create won't be as accurate as possible.

The same holds true for uptime. An ad hoc team of developers won't be able to devote as much time to monitoring the health of a test environment, as opposed to a dedicated TEM team. If the testing environment crashes, they can't test, and the testing process will run longer due to the delay. Having a team that can monitor the testing environment and prevent crashes in advance means that dev teams can perform more exhaustive, and time-consuming tests.

As the ability to manage testing environments matures, the TEM team will produce increasing benefits in terms of lowered costs, heightened productivity, and fewer bugs in production apps.

Having a team that can monitor the testing environment and prevent crashes in advance means that dev teams can perform more exhaustive, and time-consuming tests.

Lastly, a mature TEM team will produce cost savings. Allowing development teams to create their own testing environments results in waste. Developers may not understand the cost of environments, and may create active environments that remain unused for long periods of time. This racks up costs, especially when these environments are hosted in the cloud. A centralized TEM team will have a much better understanding of these costs, and will collect metrics in order to determine which environments cost more than they're worth.

To summarize, testing environment management provides long-term positive benefits. TEM reduces defects in production applications by allowing development teams to lengthen their testing cycle, use more accurate test environments, and perform more resource-intensive tests. TEM also reduces costs, not just by allowing visibility into which environments are wasting money, but also by quantitatively reducing the number of defects in productions apps, thus allowing teams to spend more time working on new products, and less time remediating defects.

TEM reduces defects in production applications by allowing development teams to lengthen their testing cycle, use more accurate test environments, and perform more resource-intensive tests.

Test Environment Management by Plutora

Imagine the scale of a larger enterprise – a bank, telco or insurance company, for example. Imagine not just dozens or hundreds of testing environments, but thousands, hosted in virtual environments and on hardware platforms, belonging to teams that are scattered to the four corners of the Earth.

In this scenario, no unassisted human intelligence will be able to efficiently provision, update, create, or delete testing environments. In order to prevent chaos, automation software is required.

Plutora provides an enterprise-scale Test Environment Manager that allows large organizations to achieve the same positive benefits in terms of producing high-quality software, eliminating redundancies, lowering costs, and reducing time-to-market.

Plutora provides an enterprise-scale Test Environment Manager that allows large organizations to achieve the same positive benefits in terms of producing high-quality software, eliminating redundancies, lowering costs, and reducing time-to-market.

This SaaS-based solution allows test environment managers to rapidly audit pre-existing test environments, and provides a centralized platform for managers to view requests to change or patch testing environments. Managers will be able to track their assets while they're in use, record changes, and estimate their impact in real time.

Lastly, Plutora Test Environment Management can replace an ad-hoc collection of spreadsheets and calendars by controlling the assignment of test environments to dev teams. Developers will have a highly visible and understandable portal with which to view the apportionment of test environments, as well as any potential conflict for resources that may arise.

As the discipline of software development has advanced at an extraordinarily rapid pace, software testing has been left behind. Close the gap with Plutora Test Environment Management, and enjoy the benefits that come with lower costs, better products, and happier customers.

About Plutora

Plutora, the market leader of continuous delivery management solutions for enterprise IT, ensures organizational alignment of software development with business strategy. Plutora improves the speed and quality of application delivery by correlating data from existing tool-chains, coordinating delivery across diverse ecosystem of development methodologies and hybrid test environments, and incorporates test metrics gathered at every step of the delivery pipeline. The platform provides visibility and a system of insights into the entire value stream, guiding continuous improvement efforts through the measured outcomes of each effort.

PLUTORA®

Learn more: www.plutora.com
Email: contact@plutora.com