TESTING WITH PACIFIC

In this white paper, we describe the scope for testing with OpenEdge Reference Architecture, different types of integration testing needed and the approach to testing integrations. It will describe the end-to-end integration test scenarios, testing individual integrations, testing integration between Progress products such as OpenEdge, Rollbase, Modulus, DataDirect Cloud, and Telerik Platform.

PROGRESS

INTRODUCTION

The technology explosion with Cloud deployment of Web and Mobile projects demands that current day software be mobile, agile, responsive to customer feedback, and accessible ubiquitously. More and more software is developed and deployed as SaaS application using cloud platforms (PaaS) such as Progress Rollbase, or the Progress Telerik Platform. Certainly, this software is deployed to multiple devices and on “web” technologies.

In this section, we describe the scope for testing with Pacific Reference Architecture, different types of integration testing needed and the approach to testing integrations. It will describe the end-to-end integration test scenarios, testing individual integrations, testing integration between Progress products such as OpenEdge, Rollbase, Modulus, DataDirect Cloud, and Telerik Platform. The topics covered include:

- Continuous availability
- Performance testing,
- Security testing
- Usability testing,
- Compatibility testing; and
- Automation frameworks

CHALLENGES INTRODUCED BY CLOUD FOR SOFTWARE TESTING

As described in the first section, various characteristics of Cloud Computing introduce several unique challenges for the Software Development Process specifically for software quality assurance. Some important challenges are:

Paradigm shift

With Cloud, web-driven SaaS applications are getting more popular. Traditionally, applications are designed completely upfront, then developed, tested and distributed for installation and use. With Cloud, users can just create an account for a particular software product, provide online payment details and start using the product through the browser in a seamless manner. Everything happens in few mouse clicks rather than purchasing the software online or offline, download or getting it through some electronic form such as CD-ROM, install, set up and start using it. Thus, the emergence of Cloud resulted in a revolution for the software industry and hence for software testing

Seamless upgrades

Traditionally, software upgrade is a lengthy process in which one needs to get the new version of the software in the same way the initial version is obtained, make the system down for a particular period of time (which depends on the complexity of the installation and setup process) and perform the upgrade. But with Cloud, upgrade should happen live with minimal or no down time. Many times, one might see that the software that is being used is upgraded next time when the customer is logged in without any down time. A commonly seen example is that of Google products. So quality assurance team needs to find ways and means to test the upgrade process of software products seamlessly.

Sharing of resources: Multi-tenancy

With Cloud-based applications, the resources used for software development as well as deployment might be in publically shared resources versus against the traditional way of having dedicated resources for individual users. These resources are often being shared among multiple customers. For example, a company might be developing, deploying and running their software in Amazon Web Services (AWS) using AWS CPU machines and the storage services. These resources are often virtualized and are shared among multiple software developers across the globe. This introduces several challenges for the quality assurance team including the need for having specific testing for such systems such as multi-tenant penetration testing.

FUNCTIONALITY TESTING

Devices Testing

Automation:

- Telerik Test Studio
- Telerik Platform Mobile Testing Framework
- Sahi
- Selenium

Cloud based products are primarily driven by web browsers. They are designed to work across any platforms and devices. So compatibility testing needs to be performed for browser compatibility, platform compatibility and device compatibility. In addition many Cloud based solutions support multiple languages and hence internationalization (commonly referred to as i18n) should be performed for all supporting languages. These compatibility circles are illustrated in **Error! Reference source not found.**. An interesting observation is that, with Cloud Operating System compatibility is out of scope. This is because, the applications are designed to be running from the browser itself and OS details are abstracted out. Telerik Test Studio functional testing is a comprehensive yet cost-effective automated testing suite. Users can complete tasks quickly and easily with the product's point-and-click interface, which is augmented by Telerik-exclusive features like a visual storyboard and 3D element selection. Test Studio also offers script-less test automation for Silverlight applications. Telerik Mobile Testing is an automated testing solution for native, hybrid, and web apps. Results can be synced to the Telerik Platform. Write one coded test on the desktop and execute it against any supported devices and clients, including an iOS native app, Android native app, hybrid app(Android/iOS/WP), mobile browser, and desktop browser. These coded tests are written in JavaScript. Descriptors are used on suites, tests, and steps so they can be understood in plain language. You may also use coded element and step repositories for easy reuse.

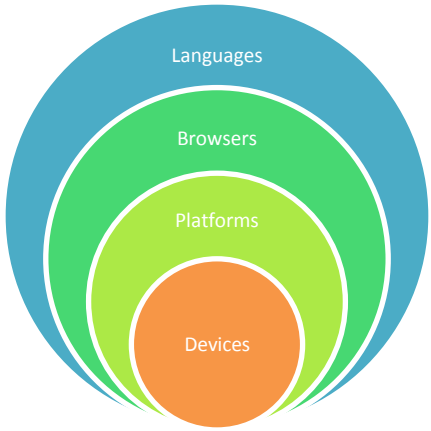


Figure 1: Compatibility circles

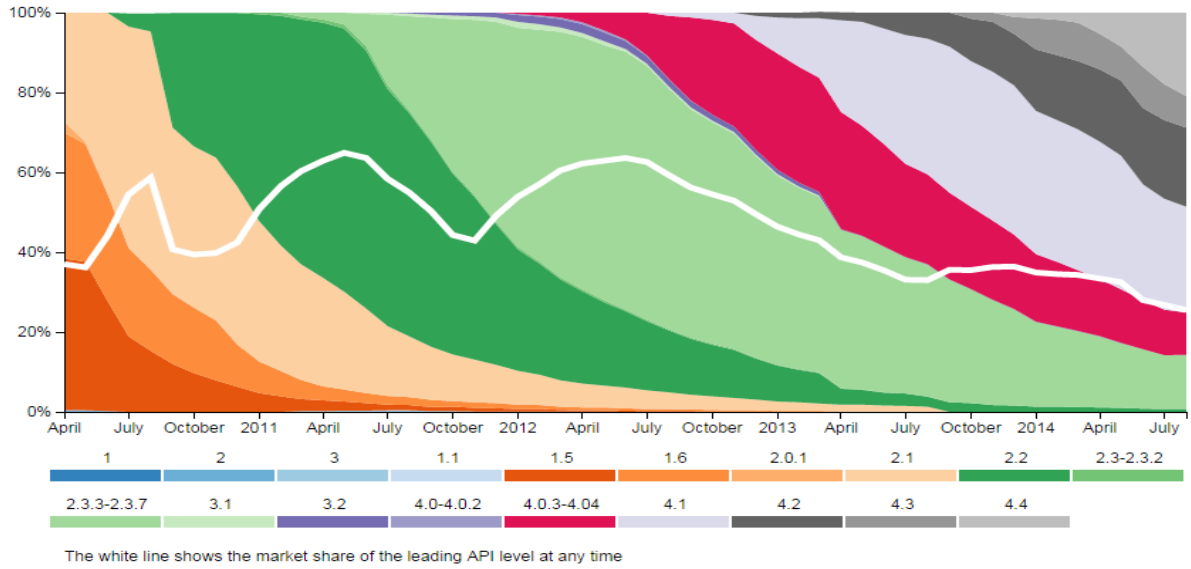


Figure 2: Android OS Fragmentation

Source: *Android Fragmentation Visualized*. OpenSignal (http://opensignal.com/), Article: http://opensignal.com/reports/2014/android-fragmentation/, August 2014

API integration Testing

APIs are direct links between the client code and the infrastructure. APIs run only when requested by Client code. In order to test such systems, one might need to develop a client shell, write sample programs to invoke such APIs and then verify the output generated by such APIs. Further, APIs need to be tested for different types of user loads. APIs are exposed to customers and hence are vulnerable to security attacks. So all necessary security testing should be performed for all the APIs. Under a multi-tenant environment, APIs could be tested to make sure that tenants are isolated from each other through the APIs.

Automation:

- SoapUI
- NeoLoad
- Node.js

Pacific supports several types of APIs including REST APIs, SOAP APIs, AJAX APIs etc. There is a need to test for proper invocation and behavior of these APIs. One approach is to build a Client shell with minimum complexities to test the accurate abilities of the API's. Plug-ins is required to activate the APIs for testing. It allows for the testing framework to be developed for many services using the same language or server. It also allows for division of coding across multiple developers to accelerate testing across a platform.

CLOUD TEST DIMENSIONS

From this section onwards, rest of the chapter focuses on testing software products that are completely or partially developed using some of the Cloud resources. These applications could be pure SaaS applications developed, deployed and running in the Cloud or applications that are built using Cloud platforms or it could be in-house applications that have certain integration with Cloud.

This section describes important characteristics of Cloud Computing and its impact on the software testing. Some of the important Cloud dimensions include (Iyer, Pasimuthu and Loganathan, 2013)

elastic scalability, multi-tenancy, security and performance. Summary of all the Cloud test dimensions and their components are illustrated in Figure 3.

Elasticity Testing	Security Testing	HA & Performance	Devices	API Integration	Multi-Tenancy
Resource acquisition/ release time	Traversal vulnerability	Time to deploy	Accessibility testing	Connectivity and invocation testing	Multi-tenant penetration
Provisioning on-the-go	User access/ roles	Multi-tenancy	Globalization and localization testing	API load testing	Rigid failure containment
Table text load testing for ELB	Identity federation management	Time to genesis	Compatibility under different situations	API security	Availability and business continuity
	Communication latency over SSL	Connectivity and reliability with 3rd parties		Multi-tenancy	Risk of correlated behaviors
	Multi-tenant penetration	Reliability and availability			Service transition activity analysis
		Latency			

Figure 3: Cloud Test Dimensions

Elasticity and Scalability

Auto-elasticity is the key for Cloud based business solutions. i.e. Software solutions can be developed and deployed in Cloud where resource provisioning should happen on-demand within a span of few minutes with ease. So test team should identify various possible resource requirement patterns and perform scalability testing for all such patterns. One has to test for cases where users can elastically up as well as down their resource requirements quickly and easily.

Some elastic load generation patterns are illustrated in Figure 4. Top left graph shows the constant load increase pattern typically the case of applications such as Gmail and Facebook where new users are coming at a steady rate with time. The top right pattern illustrates typically step increases in load with time, which also witnesses sudden dip in the resource usage at times. The bottom left pattern represents typical usage of resources that follow exponential resource usage. Finally, bottom right pattern represents the spike applications where the resources usage is constant in general with spikes in between.

Scenarios involving both vertical and horizontal scalability need to be tested. With vertical scalability, user should get better performance by replacing current resource with a more powerful resource in order to satisfy the increasing demand. In case of horizontal scalability, a customer should get increased performance by adding more resources of same type in the platform.

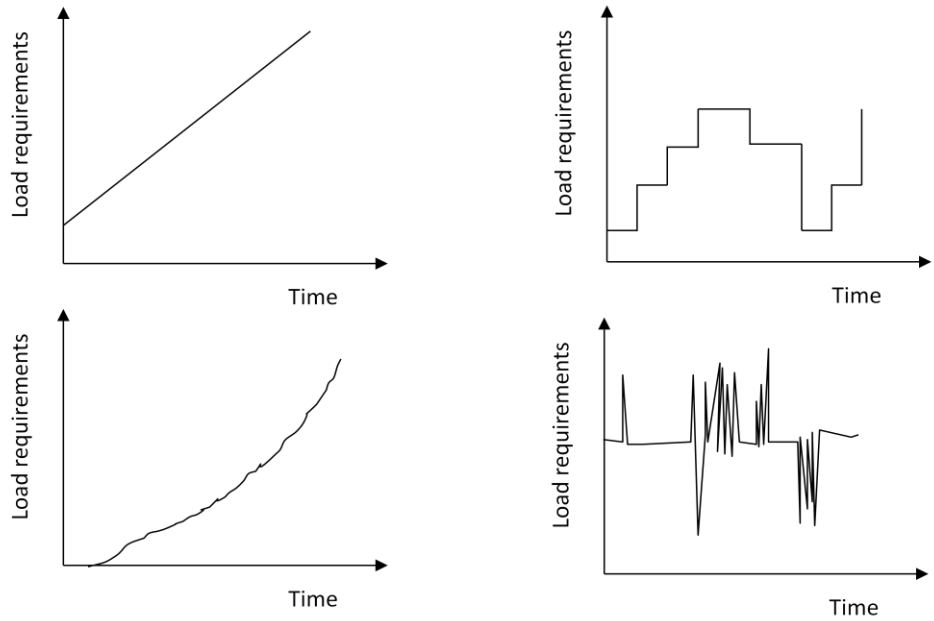


Figure 4: Sample load patterns for testing

One also needs to test the functionality of fully-automated scripts for automatic resource provisioning on-the-go. Further, testing is required for finding out the worst-case elasticity that can be achieved (in terms of scale-up). Often, IaaS providers offer load balancers to balance the load among multiple resources based on the incoming traffic flow (Elastic Load Balancer in case of Amazon Web Services for example). Such providers should test the load balancing capability with different load patterns.

Security Testing

With Cloud, security is considered as one of the great challenges and hence the challenge for the testing as well. In addition to regular security issues, Cloud imposes a set of unique security constraints such as multi-tenant penetration testing, application development user interface (UI) testing identity federation management testing etc. A summary of various security attacks that need special focus by software testing team is illustrated in Figure 5.

User access/roles

- Authentication and authorization
- Identity federation management - SSO
- Access from different clients to the platform
- VPN, firewall settings, anti-virus
- User privileges

App Development UI Security

- SQL Injection
- URL Manipulation
- Cross-site scripting
- Password cracking
- Hidden-field manipulation

Vulnerabilities and Attacks

- Multi-tenancy penetration testing
- Traversal vulnerability
- DDoS attacks

Data Storage

- Data management at DB level(Encryption security)
- Data retention and destruction for DB: Erase and sanitize when space is reallocated
- Intruder detection capability under migration

Other Security Concerns

- Fault-injection-based testing for platform services (Including verification for all input fields, network interface, environment variables etc)
- Fuzzy testing for platform services (Injecting random data into application to determine whether it can run normally under the jumbled input)
- Data privacy: Custom SLA capabilities

Figure 5: Various security vulnerabilities to be considered

User access and various user roles need to be tested to make sure that those roles work properly as expected. Identity federation management is another important aspect to be tested. This is called as Single-Sign-On (SSO) mechanism with which a user once logged in successfully can access other

components of the systems without being prompted for the credentials again. For example, Cloud software may have the default application areas as well as community support for help and support. In such environment, the user once logged into the application area, the user should be able to access the forums without being prompted for his/her credentials again.

Automation:

- Websecurify
- ZAP tool
- Scripts

The SSO concept can be understood using the below example. Consider that you have provided your credentials to log into Gmail service by Google. Then, if you open Google drive in another tab, it will open your Google drive without being asked you to log in again. Then assume, you have opened Blogger (another Google application) in next tab. Blogger as well will be opened automatically detecting the credentials provided for Gmail. Now assume that you have logged out from Google Drive. You will observe that entire session is terminated including those for Gmail and Blogger. This is a typical SSO implementation mechanism.

Another important security vulnerability to be tested is the top issues identified by Open Web Application Security Project (OWASP) (OWASP, 2013) such as SQL injection, URL manipulation, Cross-site scripting (XSS) etc. A penetration test is a method of evaluating the security by simulating an attack by a malicious user. The intent of a penetration test is to determine feasibility of an attack and the amount of business impact of a successful exploit. Attacks like SQL Injection, Cross Site Scripting are possible to a greater extent in a multi -tenant application. Test cases should be written to attack the system from the point of view of a rogue user who possesses the ID as one of the tenant of the application and that user's ability to penetrate the system and view the information of other tenants.

Some important security test considerations are multi-tenant database security testing and validating the storage and retrieval of confidential information such as credit card information and passwords.

Cloud Performance testing

With the unique characteristics of Cloud environments, a quality team needs to perform the accuracy of various data present in the Cloud, latency and throughput. Further, elastic load testing and multi-tenant performance testing are other key items to be considered. Moreover, high availability and failover testing are required to test the behavior of the platform and applications under resilience scenarios.

Automation:

- Telerik Test Studio
- NeoLoad
- JMeter

Resilience is particualy very important in the Cloud context. It is the ability for the Cloud to work normal under extreme and adverse conditions. While testing possible security threats such as DoS attacks is one way to measure reiliency, measuring system's behaviour when hardware failure happens is another way to measure resiliency. A Cloud should be able to quickly adapt to such such failures through various ways such as spawning new instance, load replication etc. While some activity is going on in a particular Cloud instance, test team should kill that VM and measure the time taken to revcover from such failures and the impact to the undergoing activity.

Another possible resiliency issue could be resulted due to the presence of multiple tenants sharing same resources. In order to test such scenarios, as one tenant, create and run a process which can continuously allocate some memory or disk space. Then measure the performance of other tenants in the system and findout any issues to them caused by such activities. Refer (Cartlidge and Sriram, 2011) for a simulation model to simulate and evaluate resiliency for Cloud-scale data centers.

Performance test metrics differ for different types of Clouds. For example the main performance metrics for a SaaS application could be the time it takes to customize the application based on the business requirements, the time it takes to load the application etc. For a PaaS solution, it could be the time it takes for developing a solution from scratch or using given templates, the response time for executing database related queries etc. Finally for an IaaS solution such as CPU or memory, it would be throughput, provisioning time, memory and CPU utilization etc.

With an integrated Cloud-based product, quality teams have to test other aspects such as latency, reliability and connectivity with other third party components. Quality teams have to also perform special performance testing such as high availability testing, longevity testing, longevity along with scalability testing etc.

In terms of load, we have to identify and execute various user load scenarios and verify that those scenarios can be catered by our system. We should include normal load in the system, spikes of demand and time-based load to system etc in the load testing scenarios. Few scenarios are data scaling alone, UI behavior with large data and does right data record get fetches with large and complex data etc.

There are several automation frameworks available for performance and laod testing. For example, Telerik Test Studio has a load test framework with which you can use your functional test scripts to run in different test scenarios out of the box for perofrmanc testing. Test Studio enables customers to gather the performance and stability information of the server, such as throughput, CPU usage and web methods' response time when there are thousands of concurrent users. Modern web applications employ diverse web services and many pages have AJAX calls to restful web services. Web Services with high response time and poor performance can lead to unsatisfied web application users. Whether you use in-house or third party services such as stock market tickers, weather reports or news feeds, Test Studio enables you to load test your web services to help ensure their reliable performance.

Some other performance scenarios are:

- Server connection changes from wifi/mobile data/LAN connection, response time
- Overuse of memory, cleaning of temp files, garbage collection
- Limitations on max objects/applications at a time
- Number of applications that can be developed per Pacific instance
- 100's of administrators accessing the management console
- Scaling capability for Pacific Cluster Editions across multiple physical or virtual servers
- Elastic Load testing is required with different types of load scenarios.

Continuous Availability

24/7 availability of the services and round the clock services should be accessible. Identify a subsetet of key test cases and execute the cases round the clock to ensure continuous availability.

Live Upgrade and Disaster Recovery testing

This is closely related to understanding the performance of the system when an upgrade of the software/platform happens and to understand its capability to continue its business services to users even when the upgrade is going on. i.e. product company needs to make sure that business continuity is there even when software/hardware maintenance/upgrade is performed.

It is a mandate often to perform, live testing as well for disaster recovery and related issues (Vijayanathan N, 2012). This includes testing robustness of the platform against the disaster, measuring the recovery time in case of disaster and self-healing ability in case of disaster. It also includes live upgrade testing wherein the management ability of customer applications for maintenance/upgrade.

Further, it is required to test the cases where migration of an application developed in one PaaS to another PaaS (Cunha, Neves and Sousa, 2012). For example, Progress Pacific (Progress Software, 2013) is a rapid application development platform which allows users to quickly develop and deploy high-end business solutions. It also offers users to migrate their application built using Force.com platform in few mouse clicks. Test engineer need to identify such scenarios and test all such cases. In addition, Progress Pacific also offers both Cloud-based hosting services as well as on-premise self hosting services. It then allows users to develop application in one of the two environments and import that in another environment. Quality analysis team has to identify such cases and test all of them.

Multi-tenancy testing

Automation:

- UI frameworks
- Hybrid frameworks

It is required to test for rigid failure containment between tenants (Eric Bauer, 2011). This means that, failure of one tenant instance does not cascade to other tenant instances, and that service transition activities properly apply to individual application instances rather than inadvertently impacting multiple tenant application instances. A quality team also needs to test for availability and business continuity in a multi-tenant environment. Further, multi-tenancy introduces the risk of correlated or synchronized behaviors (Eric Bauer, 2011) that can stress the underlying virtualized platform, such as when multiple application instances execute the same recovery action or periodic maintenance actions simultaneously.

Moreover, if the application supports multi-tenancy, then the service transition activity analysis should also verify that no service transition activity impacts active application instances that are not the explicit target of the activity. In addition to traditional service transition activities, the multi-tenancy analysis should also verify that there is no service impact on other tenant instances when each and every tenant-specific configuration parameter is changed.

Further, a unique issue to be tested in the virtual environment is the “Traversal Vulnerability” (Owens, 2010). One is able to traverse from one VM (virtual machine) client environment to other client environments being managed by the same hypervisor. Case in point: imagine that many customers are being managed by a single hypervisor within a cloud provider. The traversal vulnerability might allow a customer to access the virtual instances of other customers’ applications. Consider the impact if your bank or particularly sensitive federal government or national defense information happen to be managed in this sort of environment, and the cloud provider does not immediately deal with, or even know about, a vulnerability of this nature. With this bit of background, it is clear that providing adequate administrative separation between virtual customer environments will be a significant security challenge with elasticity.

Another security challenge (Owens, 2010) that develops out of this scenario and in the same vein is for enforcing proper configuration and change management in this elastic, dynamic and scalable model. Even where a portal is capable of granular-access controls that control which actions a given user is able to perform, it also needs to enforce when and under what circumstances a user is allowed to perform certain actions. Without this ability, untested code or system changes could result in business-impacting (or even devastating) results. Even something as “slight” as rolling a new system into production without ensuring that proper server and application patches have been applied could result in significant damage to an organization. Therefore, a mechanism within self-service portals for enforcing an organization’s change policies becomes a worthy and necessary capability. One Cloud test framework which takes into account many of the Cloud induced test aspects can be found in (Iyer, Pasimuthu and Loganathan, 2013).

OTHER TEST DIMENSIONS IN PACIFIC

Internationalization / Localization Testing

Progress Pacific supports working with several languages. Hence we need to perform entire functionality testing for all supported languages for all the platform development functionality as well as the application run time functionality.

Product to Product Integration Testing

In this type of test scenarios, interactions between two products of Progress will be in focus. An example is when Rollbase is used where an OpenEdge database is used for storing the Rollbase objects. In these scenarios, the products not only interacts each other, but also interact with other back-office and 3rd party systems.

A/B Testing

Many times, with short release cycles, we might need to perform testing of a subset of components. For example, if the current release has more changes on triggers, we might want to stress on testing trigger functionality more compared to other features. There are many ways to choose what to test and how much to test. One mechanism is to identify what features customers use more often and then stress more on testing those features. Another approach could be to identify weak product areas (which customer use more often) and then stress more testing for those features.

SUMMARY OF TEST AUTOMATION APPROACHES

Functional Testing – (Web/Mobile UI)	Telerik Test Studio Telerik Platform Mobile Testing Framework Sahi / Selinium scripts
Performance Testing	Telerik Test Studio for Load / Performance Testing Neoload, JMeter
API Integration Testing	SOAP UI / Node.js
Integration Testing	Telerik Test Studio + C# Approach Test-NG + Java + Sahi combined approach
Security Testing	WebSecurify ZAP Tool

Figure 6: Overview of automation frameworks

In this section we summarize the automation tools that we can use for such testing. Primarily we can use Telerik Test Studio (or Telerik Platfrom Mobile Testing) to cover most of the cases such as function testing (both web and Mobile), devices testing, performance testing, load testing and Integration testing.

TEST CHALLENGES AND APPROACHES FOR CLOUD INTEGRATION TESTING

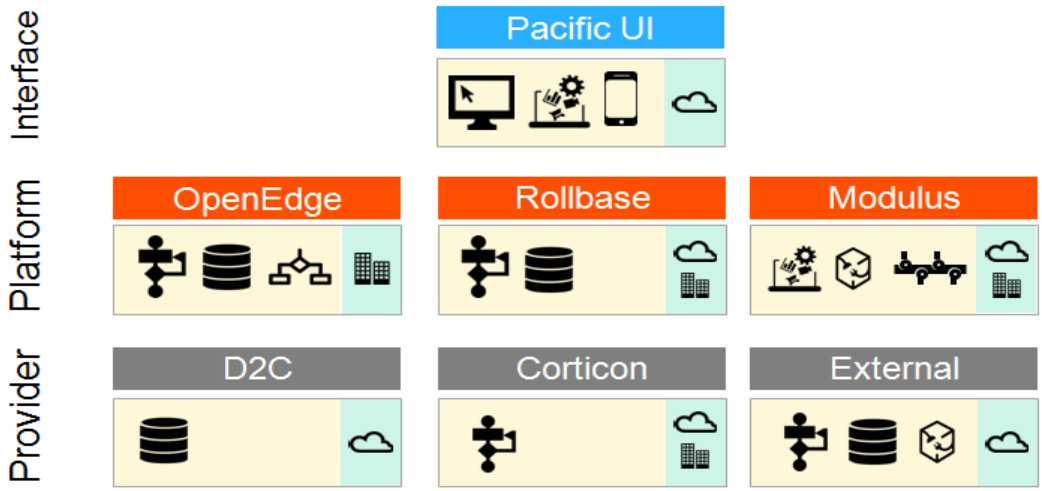
Often times, Cloud software will not be a standalone system. It will have the core product components integrated with many back office systems as well as 3rd party components. For example, accepting credit card details and processing invoice needs special compliance certification based on security guidelines. In such cases, a software product may integrate itself with 3rd party payment gateways such as Bill Desk or PayPal. Hence, test team needs to get sandbox environments (that are identitical to its production environment) for such payment gateways in order to obtain same behavior in both test as well as live environments.

There are several issues which need to be resolved for automating an integrated Cloud-based product testing. They are:

- In an integrated Cloud-based product, multiple systems behave differently. Further, verification process for different systems differ each other.
- Some of the systems do not allow automated deletion of data created for testing. So it imposes a unique requirement to have unique users created every time an engineer perform such test automation
- Unpredictable delays in updating various systems.
- Different types of environments for testing. For example, requirement to perform web UI testing and runtime testing in one test scenario poses its own unique challenges

Scope for Integration Testing of PRA

Pacific primarily consists of UI as the interface, OpenEdge, Rollbase and modules as the Platforms servers as providers for ISVs, SMEs and D2C, Corticon and external system serves as the Data, business logic, and Business rules providers.



Necessity of Integration Testing

Integration testing is essential to identify systems interoperability when multiple system(s) interaction with each other. Further, it also helps in the following aspects.

- Dataflow between sub-systems (Presentation layer, Platform and Data providers in cloud as well as on-promise)
- User interface representation on multiple devices like mobile, Tablet and web (Desktop) and workflows.
- Systems behavior on abnormal conditions

    - Systems failure
    - Timeouts
    - Resiliency (including multiprotocol systems in place)
    - Data Logging
    - Usability
    - Vulnerability
- Systems behavior under normal conditions
    - System behavior under large volumes of data
    - Vulnerability
    - Data Logging
    - Usability
    - Synchronous Vs Asynchronous data flows
- Scalability on different SLAs
- Capture inputs from different groups constitute of whole system and identify issues early stage.

Several integration points exist in PRA platform. We can classify the integration points into three:

- Integration between Progress Products/Technology/External systems
- Back-office Integration
- Integration with 3rd Party components

Integrated Test Automation with Cloud

Software test automation in Cloud imposes its own unique set of challenges. Such software systems consist of both run time components as well as UI components. In order to automate such products, one might need to either develop a test framework that can efficiently test both run time and UI components or use a combination of two frameworks one for run time component testing and the other one for UI component testing. In the later case, some mechanisms are required to initiate the test written in one framework from the other one as well as approaches to generate a combined test result so that the test engineer do not need to analysis two set of results.

Typical test automation framework architecture for such cases is illustrated in Figure 7. In this architecture the outer harness is the run time automation framework and the inner harness is the UI automation framework. In such cases, the runtime component starts executing the test cases with runtime calls such as API calls and wherever required it will trigger the test cases in UI framework for the UI testing. After UI testing has been completed, the runtime framework will continue its execution. Then the runtime component will generate the combined test results and will be stored in the result repository.

For example, consider a Pacific-based SaaS application CRM. User is required to authenticate using a 3rd party system and then gets landed in the application. Similarly, payment is made through Paypal system. One automated test case could be to first call an API of the authentication system using runtime framework with appropriate credentials as parameters and receive a token. Then use the UI test framework to test the CRM application dashboard for all user interface testing. Finally when it comes to payment, runtime test framework will make appropriate API calls or runtime actions to check if payment is correct and then complete the test execution by compiling the test results from both frameworks.

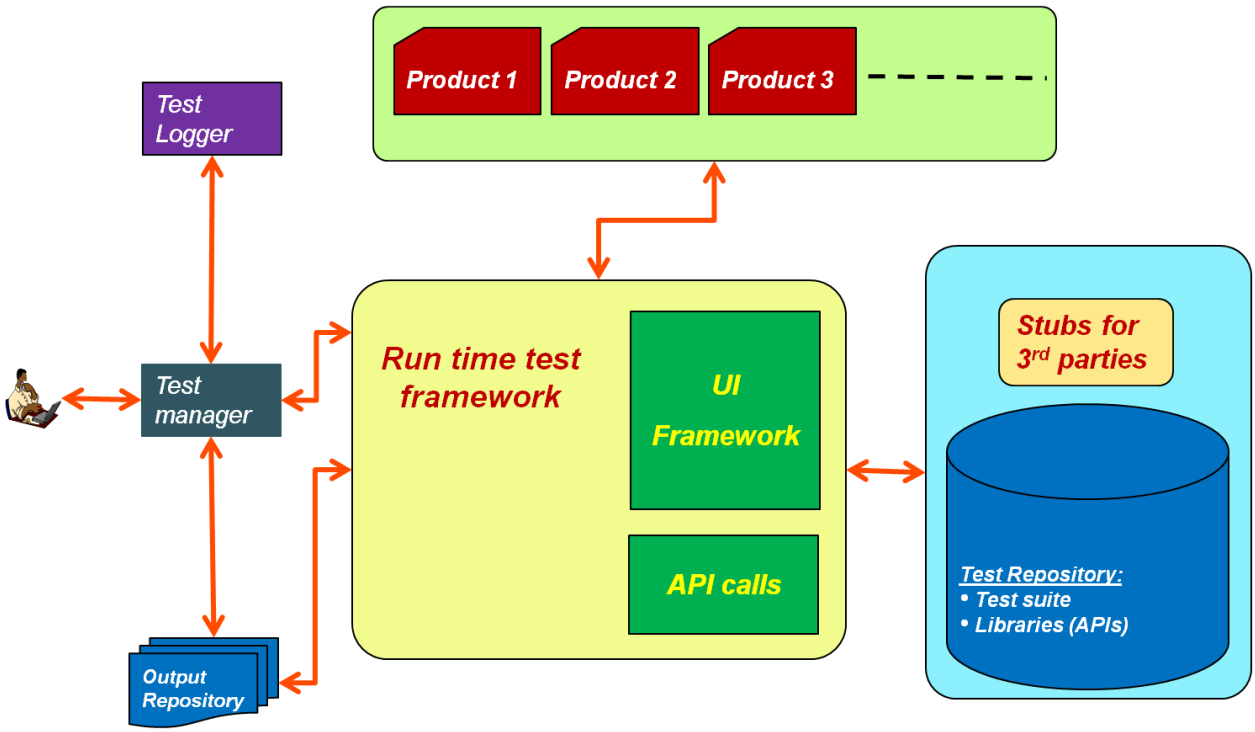


Figure 7: Test Automation Architecture

Two examples are given below for achieving the above architecture.

Telerik Test Studio as the UI framework C# as the Runtime framework
Sahi as the UI framework Java + TestNG as the Runtime framework

In addition, one might have to use the exposed APIs provided by other third party systems integrated as part of the product. Sometimes, it is also required to develop stubs to simulate certain functionalities of other systems in order to achieve an end to end flow in the automation.

CONCLUSIONS AND FUTURE REMARKS

In this section, authors first described several key challenges introduced by Cloud Computing paradigms for software testing. These include the paradigm shift in software development from legacy on-premise software to web-driven dynamic Cloud-based applications. The chapter then described several benefits of software testing in the Cloud. The key benefits achieved by testing in the Cloud are reduced cost for software testing, faster test execution, ability to test in environments that are close to production environments and better collaboration.

Then various Cloud test dimensions such as elasticity, scalability, security testing, performance testing, compatibility testing, API integration testing, live upgrade testing, disaster recovery testing and multi-tenancy testing are described. Finally, author introduced the test challenges and automation approaches for Cloud integration testing.

There are no known frameworks which can test all the issues described in this section. However combination of two or more frameworks can test all the issues described. An interesting research in future could be to identify such frameworks and develop strategies to combine them so that efficient, reliable and easy software testing can be done for the current day applications.

Bibliography

Cartlidge, J., & Sriram, I. (2011). Modelling Resilience in Cloud-Scale Data Centres. *Proceedings of the 23rd European Modeling & Simulation Symposium (Simulation in Industry).*
Cunha, D., Neves, P., & Sousa, P. (2012). Interoperability and Portability of Cloud Service Enablers in a PaaS Environment. *In Proc. of The 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012)*, (pp. 432-437). Porto, Portugal.
Eric Bauer, R. A. (2011). *Reliability and Availability of Cloud Computing.* USA: IEEE Press, Wiley.
Iyer, G. N., Pasimuthu, J., & Loganathan, R. (2013). PCTF: An Integrated, Extensible Cloud Test Framework for Testing Cloud Platforms and Applications. *Quality Software (QSIC), 2013 13th International Conference on*, (pp. 135-138). Nanjing, China.
L.M. Riungu, O. T. (2010). Software Testing as an Online Service: Observations from Practice. *Proc. 3rd Int'l Software Testing, Verifi cation, and Validation Workshop (ICSTW 10)* (pp. 418-423). IEEE CS.
OWASP. (2013). *OWASP Top Ten security attacks*. Retrieved from https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
Owens, D. (2010). Securing Elasticity in the Cloud. *ACM Queue* .
Progress Software, D. (2013, October). *Progress Pacific*. Retrieved from http://www.progress.com/products/pacific
Riungu-Kalliosaari L, O. T. (2012). Testing in the Cloud: Exploring the Practice. *IEEE Software* , 46-51.
Scott Tilley, T. P. (2012). *Software Testing in the Cloud: Migration and Execution.* Springer.
Shull, F. (2012). A Brave New World of Testing? An Interview with Google's James Whittaker. *IEEE Software* , 4-7.
Vijayanathan N, S. S. (2012). Overcoming challenges associated with SaaS testing. *Whitepaper, Infosys* .
Zilberstein, Z. G. (2009). Cloud-based Performance Testing of Network Management Systems. *IEEE 14th Int'l Workshop Computer Aided Modeling and Design of Communications Links and Networks (CA-MAD 09)* (pp. 1-6). IEEE CS.

Progress Software
Progress Software Corporation (NASDAQ: PRGS) is a global software company that simplifies the development, deployment and management of business applications on-premise or in the cloud, on any platform or device, to any data source, with enhanced performance, minimal IT complexity and low total cost of ownership.

Worldwide Headquarters
Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA
Tel: +1 781 280-4000 Fax: +1 781 280-4095 On the Web at: www.progress.com
Find us on facebook.com/progresssw twitter.com/progresssw youtube.com/progresssw

For regional international office locations and contact information, please refer to the Web page below: www.progress.com/worldwide

Progress, [ALL PRODUCT NAMES LISTED IN PUBLICATION] are trademarks or registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Any other marks contained herein may be trademarks of their respective owners. Specifications subject to change without notice.

© 2013 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.
Rev. MO/YR