JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Application Testing in terms of Pega

Karthick E[1, a)], Keerthana P[1, b)], Srimathi M[1, c)], Rajesh K[1, d)], Kavin S[1, e)]
[1]Electrical and Electronics Engineering
V.S.B. Engineering College
Karur, 639 111, India

Abstract- Application testing in Pega is integral to validating the efficiency and reliability of Pega applications through a blend of automated and manual testing techniques. It involves diverse methodologies such as unit testing, integration testing, system testing, and user acceptance testing (UAT), all facilitated by Pega's model-driven architecture. Tools like Pega Unit and the Automated Testing Framework (ATF) are pivotal, enabling early detection and resolution of defects. These testing processes ensure that business processes, rules, and interfaces function as intended, ultimately enhancing application quality, user satisfaction, and reducing time-to-market. This rigorous testing framework is essential for delivering robust Pega solutions that align with organizational goals and user expectations.

IndexTerms - Application testing, unit testing, scenario testing, selenium.

I.INTRODUCTION

Application testing in Pega is a comprehensive process designed to ensure the robustness, functionality, and reliability of Pega applications. This process encompasses various testing methodologies, including unit testing, integration testing, system testing, and user acceptance testing (UAT). Pega's unique model-driven architecture facilitates automated testing, making it easier to validate business processes, rules, and interfaces. Tools like Pega Unit, for unit testing, and Automated Testing Framework (ATF), for end-to-end automation, play crucial roles in identifying and rectifying defects early in the development cycle. By leveraging these tools, along with best practices in test planning and execution, organizations can achieve high-quality Pega applications that meet user requirements and maintain performance standards, ultimately ensuring a seamless user experience and reducing time-to-market.

Production level

Production levels dictate the permissible types of changes and define the environment's purpose. Figure 1 Shows the production level

1 - Sandbox / Experimental
2 - Development
3 - Quality assurance / Testing
4 - Staging / pre-production
5 – Production

High production level

- A limited security setup allowing only a select few users to modify rules.
- Low production level (usually 2) designated for development environments.
- Developers can freely create and modify rules with minimal constraints.

II.SANDBOX

Experiment about required application- i.e., Experiment means we take requirement. The sandbox production level in Pega serves as a safe and isolated space where developers and
designers can work on building, customizing, and testing applications without impacting the stability and integrity of the actual production environment.

Key characteristics and purposes of the sandbox production level in Pega:

Development and Customization:

The sandbox environment is where developers can create, modify, and enhance application features, user interfaces, business logic, and integrations. It allows them to experiment with new functionalities before deploying them to higher production levels.

Figure 1: Production Level

Safe Environment:
Since the sandbox is separate from the actual production environment, any mistakes or errors made during development and testing do not impact real users or critical business processes

Version Control:
Developers can use version control systems to manage the changes made in the sandbox environment. This ensures that changes are properly tracked, and previous versions can be rolled back if necessary.

Prototyping and Proof of Concepts:
The sandbox is an ideal place to create prototypes and proof of concepts to validate ideas and demonstrate new features to stakeholders.

Training and Skill Building:
The sandbox environment is often used for training purposes, where new developers or team members can learn Pega's features and best practices in a risk-free setting.

III. DEVELOPMENT

Environment in which the developers are actually building the application. In development environments, developers often prefer a lower production level, typically level 2, which has fewer restrictions than higher levels. Development in Pega involves a model-driven approach that streamlines the creation of business applications by using visual tools and pre-built components. This development process is tightly integrated with comprehensive application testing to ensure that each component functions as intended from the outset. As developers design workflows, business rules, and interfaces, they employ PegaUnit for unit testing to validate individual pieces of functionality. Integration testing ensures that these components work together seamlessly, while the Automated Testing Framework (ATF) facilitates end-to-end testing by automating complex scenarios and user interactions. Continuous testing during development helps identify and resolve defects early, maintaining code quality and adherence to business requirements. This integrated approach to development and testing in Pega accelerates delivery timelines, enhances application reliability, and ensures that the final product meets user expectations and organizational standards.

IV. TESTING

An application is tested for configuration errors once its developed. Application testing is the process of testing whether an application functions according to various functional and non-functional requirements. There are many methods that you can use to perform application testing, which includes,

- Manual methods
- Automation tools
- A combination of both

Platform
When it comes to testing Pega applications, there are several platforms and tools available that cater to different types of testing needs. Let's explore the different platforms where you can conduct testing of Pega:

Pega Unit Test Framework (PUTF):
As mentioned earlier, Pega Unit Test Framework (PUTF) is specifically designed for unit testing within the Pega Designer Studio. It allows developers to create and execute unit tests for individual rules like activities, data transforms, decision rules, etc. Pega Unit Test Framework provides a convenient and integrated environment for developers to validate the functionality of their rules during the development phase.

Pega Scenario Testing Framework:
The Pega Scenario Testing Framework is used for scenario-based testing of Pega applications. This framework allows testers to create end-to-end test scenarios that simulate real-world user interactions with the application. Testers can define test data, automate test cases, and validate expected outcomes to ensure that the application performs as expected in various scenarios.

Automated Unit Testing with PegaUnit:
PegaUnit is a testing framework that enables developers to write and execute automated unit tests for their Pega applications. Developers can write test cases using Java and JavaScript and use PegaUnit's built-in assertion methods to validate expected outcomes. Automated unit testing with PegaUnit facilitates continuous integration and helps catch issues early in the development process.

Pega Automated Testing with Selenium:

Pega applications can be tested using Selenium, which is a popular open-source tool for automating web browsers. Selenium allows testers to create automated test scripts that interact with Pega applications through the web browser, mimicking user actions. This approach is particularly useful for regression testing and ensuring the compatibility of Pega applications across different web browsers.

Third-Party Testing Tools Integration:

Pega provides integration capabilities with various third-party testing tools, such as JUnit, TestNG, Cucumber, and Jenkins. By integrating with these tools, Pega applications can leverage existing testing infrastructures and benefit from advanced testing features not available in Pega's native testing frameworks.

User Acceptance Testing (UAT) Platforms:

For user acceptance testing, Pega applications can be tested on platforms that cater specifically to UAT. These platforms allow business stakeholders and end-users to validate the application's functionality against business requirements and use cases. UAT platforms help ensure that the application meets business objectives before it goes into production.

Performance Testing Tools:

Performance testing is essential to evaluate the scalability and responsiveness of Pega applications. There are various performance testing tools available, such as Apache JMeter and LoadRunner, which can be used to simulate heavy user loads and measure the application's performance under stress.

Remember that the choice of testing platform depends on the specific testing requirements, team expertise, and the scale of the Pega application. A well-rounded testing strategy should include a combination of unit testing, scenario testing, automated testing, and user acceptance testing to ensure the overall quality and reliability of Pega applications. Figure 2 shows the types of testing

Testing Types:

- Functional testing
- Non-functional testing
- Regression testing

Functional testing

Functional testing in Pega is the process of testing the functionality of a Pega application to ensure that it meets the specified requirements. It involves testing the application's features and functions from an end-user perspective to ensure that they work as expected. Pega provides several approaches for functional testing, including unit testing, UI testing, and scenario testing. Unit testing is performed using Pega Units, while UI testing is performed using Pega's built-in UI testing tool or third-party tools like Selenium and Cucumber. Pega Scenario testing is a built-in tool that enables end-to-end scenario testing.

Functional testing types

- White Box testing
- Black Box testing

White Box testing

White Box testing targets the internal workings of an application, utilizing various techniques to validate the code and ensure that statements, methods, conditions, and other logic function correctly. Conducting White Box testing requires knowledge of the application's internal structure.

White Box testing types

- Unit testing
- Integration testing

Unit testing

Unit testing is a software testing technique mainly focused on examining individual units (separate division) or components of a software application in isolation from the rest part of the application. The goal is to ensure each unit functions as intended. Automated unit testing involves creating test cases for individual rules, grouping these cases into test suites, running the tests, and reviewing the results. During testing, the outcomes are compared to the expected results defined in assertions. Unit testing is crucial in continuous development and continuous integration models of application development. Continuous and thorough testing helps identify and fix issues before releasing the application, thereby enhancing its quality.

Pega 8.8 supports both manual and automated methods of unit testing. Manual unit testing can be performed by creating test cases for individual rules and running them manually. Automated unit testing can be performed using PegaUnit testing, which is a headless testing framework that allows you to create automated test cases that run quickly to ensure the

Figure 2: Quality assurance/Testing

quality of your application at a unit level. To create PegaUnit test cases, you can use the PegaUnit Test Case wizard in Dev Studio. You can also group multiple test cases into test suites and run them together.

Integration testing

Integration testing combines various software components or units to verify their communication and interaction. Stubs and drivers, which are dummy programs, play a crucial role in facilitating this testing process.

Integration testing in Pega is a type of testing that verifies the integration between different systems or components. Pega provides several tools and frameworks to perform integration testing, including the Pega Unit testing framework, the Pega API testing tool, and the Pega Scenario testing tool. Pega also supports third-party testing tools such as Selenium and Cucumber for UI testing and REST-assured, Citrus, POSTMAN, and SoapUI for API or services testing.

Top-down model

In this model, testing begins with the higher-level modules and progressively integrates the lower-level modules. Stubs are generally used in place of modules that are not ready.

Bottom-up model

In this model, testing begins with the lower-level modules and incrementally integrates the higher-level modules. Drivers are typically used to invoke the lower-level modules.

Black Box testing

Black box testing focuses on the software application's behavior and output, rather than its internal structure. This approach involves testing methods that treat the application as a complete system and includes an end-to-end testing process. To perform Black box testing, knowledge of the application's requirement specifications is essential.

Black Box testing types

- System testing
- Acceptance testing

System testing

System testing involves evaluating a fully integrated software product as a whole. The purpose of system testing is to simulate production-like scenarios to assess end-to-end system requirements, including established dependencies, data integrity, and communication with other systems.

System testing types

- Scenario testing
- Performance testing
- Security testing

Scenario testing

Scenario testing is a type of testing that verifies that an application functions correctly from end to end. It involves recording a set of interactions for a case type or portal in scenario tests. You can group related scenario tests into test suites to run multiple scenario test cases in a specified order. You can also run scenario tests that are specific to an access group that your operator includes. Upon initiating a scenario test, you have the option to specify if it is role-specific and assign a corresponding access group. Scenario testing is a feature available in Pega Platform version 8.5 and later, provided as a standalone component. It offers a user-friendly graphical tool for creating tests, enabling enhanced test coverage without the need for intricate coding. Users can review and execute scenario test cases and access reports to identify case types and portals that failed the scenario testing process.

Performance testing

Performance testing in Pega involves testing the application to ensure that it meets the performance requirements and can handle the expected load. It is done to identify and eliminate performance bottlenecks, ensure that the application can handle the expected user load, and provide a good user experience. Pega provides several tools and guidelines for performance testing, including PegaUnit test suites, load testing with Apache JMeter, and monitoring with Pega Predictive Diagnostic Cloud (PDC).

Security testing

Security testing in Pega refers to the process of identifying vulnerabilities and weaknesses in Pega applications and infrastructure. It includes both Static Analysis Security Testing (SAST) and Dynamic Analysis Security Testing (DAST). Pega provides several tools for security testing, including the Pega Rule Security Analyzer (RSA) for SAST and other available tools such as IBM AppScan, Burp Suite, and OWASP ZAP tools for DAST. Pega also has a Vulnerability Testing Process for applications on Pega Cloud, which outlines the terms and conditions for conducting security assessments. Pega takes security very seriously and is subject to thorough penetration tests performed by independent third-party evaluators.

Acceptance testing

Acceptance testing evaluates if a system meets predefined acceptance criteria and aids stakeholders in deciding whether to approve the system. Typically, acceptance testing is conducted by customers. Some examples of end-to-end testing tools are:

- Test Complete
- Selenium
- IBM Rational Functional Tester
- Eggplant
- Telerik

Non-functional testing

Non-functional testing in Pega 8.8 refers to the testing of aspects of a Pega application that are not directly related to its functionality, such as performance, security, and accessibility. Pega provides various tools for non-functional testing, such as Pega Accessibility Inspector for accessibility testing, Pega Rule Security Analyzer (Pega RSA) for security testing, and Apache JMeter, Blaze Meter, and HP LoadRunner for performance testing. Additionally, Pega Predictive Diagnostic Cloud (PDC) can be used for monitoring the performance of a production system. Other third-party tools such as IBM AppScan, Burp Suite, and OWASP ZAP can also be used for security vulnerability assessment.

Regression testing

Regression testing in Pega 8.8 refers to the process of testing an application after an update to Pega Platform version 8.5.1 or later. The purpose of regression testing is to identify any unexpected changes to application behaviour after the update and verify that all features work as expected. Regression testing involves running all automated tests, manual test cases and checks, performance and other non-functional tests and checks, Smoke tests are utilized to confirm the continued functionality of essential features and to assess whether any alterations following updates have impacted these features. Additionally, sanity testing is conducted to identify the effects of changes, such as new features that we done or bug fixes, does not introduce any issues after the update. After verifying the change, other functionality that the change might have affected is tested. To reduce testing cycle times, an automated testing pipeline can be implemented.

Roles and Responsibilities

- Team Lead
- Developer
- Tester
- Business Analyst
- Quality Assurance Lead

V.STAGING

Act as intermediaries between development and production. After testing send to staging, means which server we need, full set of configurations in server and IP etc. Some load testing, Finding defects & report to Developer. Staging in Pega application testing is a pivotal phase where applications are deployed in an environment that closely mirrors the production setting. This stage serves as a final checkpoint before production release, allowing for thorough testing of the entire application in conditions that simulate real-world usage. During staging, various testing methodologies, including performance testing, load testing, and user acceptance testing (UAT), are conducted to ensure that all components of the application function correctly under expected load and usage scenarios. Staging also enables the identification and resolution of any last-minute issues, ensuring that the application is stable,

reliable, and ready for deployment. By utilizing Pega's Automated Testing Framework (ATF) and other tools, teams can automate repetitive tests, enhance test coverage, and streamline the validation process, ultimately leading to a higher quality product and smoother transition to production.

VI. PRODUCTION

In production environment deploy actual application. Once your application is operating smoothly within your Pega Cloud Staging environment, employ Pega Deployment Manager to transition the application to the production environment. In Pega, Production Environment is the final and live deployment stage where the application is accessible to end-users. It is the most critical environment, and any changes or updates made here can directly impact business processes and users. Production Environment should be highly stable & secure to handle real-world workloads. In this environment, no defects should be there. Each transition involves rigorous testing and validation to ensure that the application is ready for the next level and that changes do not introduce issues or disruptions in the live environment. Proper version control and deployment tools are used to manage and track changes made to the application as it progresses through the various environments. This ensures that the application's integrity and stability are maintained at each stage of the development life cycle. If any defect is missed in testing & product comes into production environment, will be lots of consequences like then the product will lose its value., client will be unhappy and so on...

VII. CONCLUSION

Application testing in Pega is a critical component that ensures the delivery of high-quality, reliable, and efficient Pega applications. By employing a range of testing methodologies, from unit testing to user acceptance testing (UAT), and leveraging robust tools like PegaUnit and the Automated Testing Framework (ATF), organizations can detect and address errors early in the development process. This comprehensive testing approach not only validates the functionality and performance of business processes, rules, and interfaces but also enhances user satisfaction and accelerates time-to-market.

REFERENCES

[1] P. Pareek, R. Chaturvedi and H. Bhargava, "A Comparative Study of Mobile Application Testing Frameworks" in BICON 2015, pp. 4-5, 2015.

[2] R. Rattan and Shallu, "Performance Evaluation and Comparison of Software Testing Tool", *Int. J. Inf. Comput. Technol.*, vol. 3, no. 7, pp. 711-716, 2013.

[3] A. M and S. KN, "Comparative Study on Different Mobile Application Frameworks", *Int. Res. J. Eng. Technol.*, pp. 1299-1300, 2017.

[4] P. N. Tran and N. Boukhatem, "The Distance To the Ideal Alternative (DiA) Algorithm for Interface Selection in Heterogeneous Wireless Networks", *Proc. 6th ACM Int. Symp. Mobil. Manag. Wirel. access-MobiWac '08*, pp. 61, 2008.

[5] Linares-Vásquez, Mario, Kevin Moran and Denys Poshyvanyk, "Continuous evolutionary and large-scale: A new perspective for automated mobile app testing", *Software Maintenance and Evolution (ICSME) 2017 IEEE International Conference on*, 2017.

[6] Haneen Anjum et al., "A Comparative Analysis of Quality Assurance of Mobile Applications using Automated Testing Tools", *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 8, no. 7, pp. 249-255, 2017.

[7] Priyanka Rathi and Vipul Mehra, "Analysis of Automation and Manual Testing Using Software Testing Tool", 2015.

[8] V. Bargavi, "Survey on automation testing tools for mobile applications", *International journal of Advanced Engineering Research and Science (IJIAERS)*, vol. 2, no. 11, 2015.

[9] A. Jain et al., "A Comparison of RANOREX and QTP Automated TestingTools and their impact on Software Testing", *IJEMS*, vol. 1, pp. 8-12, 2014.

[10] Sebastian Huhn, Rolf Drechsler, "Next Generation Design For Testability, Debug and Reliability Using Formal Techniques", *2022 IEEE International Test Conference (ITC)*, pp.609-618, 2022.

[11] ebastian Huhn, Daniel Tille, Rolf Drechsler, "A Hybrid Embedded Multichannel Test Compression Architecture for Low-Pin Count Test Environments in Safety-Critical Systems", *2019 IEEE International Test Conference in Asia (ITC-Asia)*, pp.115-120, 2019.

[12] Kunwer Mrityunjay, Santosh Biswas, Jatindra Kumar Deka, "ATPG for Incomplete Testing of SoC and Power Aware TAM Architecture", *2018 15th IEEE India Council International Conference (INDICON)*, pp.1-6, 2018.

[13] Sungyoul Seo, Yong Lee, Sungho Kang, "Tri-State Coding Using Reconfiguration of Twisted Ring Counter for Test Data Compression", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.35, no.2, pp.274-284, 2016.

[14] eewon Cho, Woosung Lee, Jooyoung Kim, Sungho Kang, "Failure bitmap compression method for 3D-IC redundancy analysis", *2015 International SoC Design Conference (ISOCC)*, pp.335-336, 2015.

[15] Panagiotis Sismanoglou, Dimitris Nikolos, "Input Test Data Compression Based on the Reuse of Parts of Dictionary Entries: Static and Dynamic Approaches", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.32, no.11, pp.1762-1775, 2013.