WHITEPAPER

SOLUTIONS

Ensuring Quality through Progressive Approach in **Test Automation**

www.igtsolutions.com

Executive Summary

Emerging technological innovations along with dynamic business needs of the travel industry are shaping the travel IT industry to set new and modern trends. The dynamism in the business needs of the travel industry have led transformation of system architectures from monolithic and slow batches processes to SOA and real time processes. To ensure good quality of IT systems thorough testing of the applications is crucial.

Manually testing an application to find defects (or bugs) is time consuming, expensive, tedious and subject to human error. On the contrary, automated testing, in which Quality Assurance (QA) teams use software tools to run detailed, repetitive and data-intensive tests, helps improve software quality and provide best possible utilization of limited testing resources.

The classical automation approach focuses primarily on automating regressive or unit test cases which have already been manually tested while progressive test automation is an innovative approach that enables the automation of functional test cases during the application development process, which in turn helps to:

- Accelerate the time to market of the end product
- Identify defects early in the development lifecycle, which helps to generate better Return On Investment (ROI)

This document is an attempt by IGT Solutions (IGT) to introduce our approach towards Progressive Automation Testing in Simple Object Access Protocol (SOAP) based web services. This approach has helped IGT achieve better ROI for our clients.

Progressive Automated Testing

Progressive automation testing is an innovative approach which is cross-compatible across all software development lifecycle models. However, the Agile and Iterative methodologies have been observed to provide the most optimum results. Using progressive approach automation can be done in the Nth sprint itself instead of the usual N-1 or N-2 sprint as in case of Regressive automation.

In order to implement automation in the Nth sprint, scripts are written to test new functionality which is not yet developed/ development is in progress. Testers get involved early in the development cycle and work closely with the business users as well as the development team so that all stakeholders (testers, developers and business users) share the same knowledge base progressive automation is adopted to replace manual execution effort in the QA life cycle and allows the functional testers and automation QA to work parallel in. They do the initial analysis and create the optimum number of functional test cases having maximum coverage.

While the functional tester creates the functional test cases, the automation QA analyze the automation approach to be followed and in design framework utilities.

Once the functional test cases are created and reviewed they are converted to automation scripts. One of the challenges here is also frequent updates in the manual test case as the development is still in progress. An initial review is recommended for the manual test cases to avoid un-necessary changes. Reviewed functional test cases are then automated using the approach analyzed earlier by the automation expert.

Once the automation suite is created the dry run executed on a virtualized environment and the results are validated by manual team. Once the designated environment is up and running, the executions are done on actual dedicated environment. The manual tester is involved in the analysis of the automated results after the script executions. Few test cases that are not in automation scope might have to be executed manually. Defects once identified are logged against defect tracking tool.

The diagram below shows how automation is being introduced early in the process and the time gain that we get utilizing the automated executions

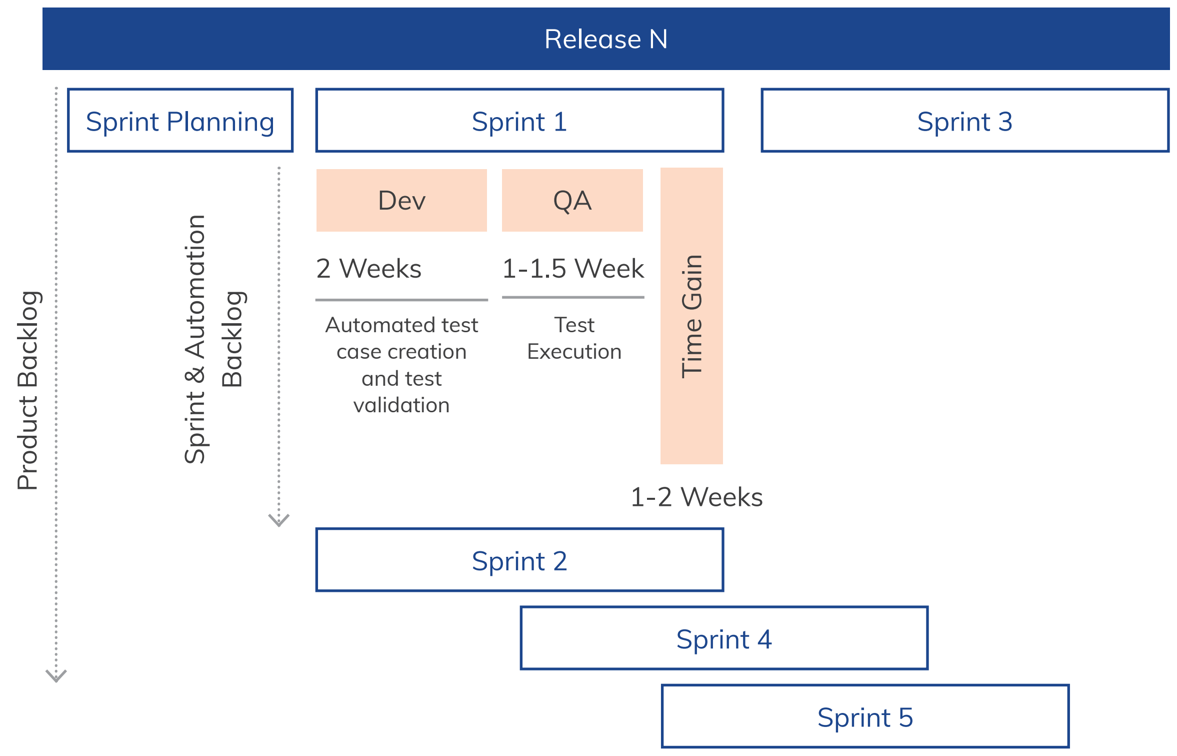

Fig.1: Introducing Progressive Automation in Nth Sprint

Another advantage of using an automated test suite is that regression test cases can be identified from amongst the functional test cases of the Nth sprint and thus automated regression test suite is created. This can be used for automated executions for N+1 sprints easily to ensure that the flow in existing iteration doesn't break due to update(s) in the current or further planned iterations.

How Progressive Automation is different from Classical Automation?

Progressive automation testing starts in parallel with the development phase whereas in classical automation manual test cases are already executed once, before the automation starts. Following points illustrate further how progressive automation differs from the regressive automation

New Functionality Validation: In Progressive automation automated test cases are written to validate new functionality where in classical regressive automation, scripts validate that existing functionality does not break with changes.

Time Saving: In Progressive automation, defects identified are fixed in less time frame and re-testing is done within same iteration while kick off of for next iteration takes place and scenarios to be tested are identified while retesting of previous iteration continues.

Higher ROI: Progressive automation also allows gaining high levels of ROI within a project whereas classical regressive automation suites are built at the end of the project with the hopes of using them again in the next phase/release. For one of the projects a 50% reduction in execution time was observed, using automated scripts developed following a progressive approach.

IGT's Initiative

Service Oriented Architectures (SOA) lifecycles are agile and iterative, with emphasis on web services, increasing the importance of automation testing. These web services are application components that communicate using open protocols and are used by other applications. Generally, web services use XML to code and decode and SOAP to transport the information (using open protocols).

How to achieve continuous testing when manual testing is time taking and companies are at loggerheads to meet aggressive deadlines and delivering quality product?

In order to meet this challenge, IGT has successfully implemented the progressive approach with the testing of web services in one of the SOA based projects by utilizing the best automation tool as per the requirements and marking almost fifty percent reduction in the QA cycle time, without compromising on the test coverage and the ultimate quality of the product

However, creating scripts for services without User Interfaces (UI) or non-functional services posed a challenge, so service virtualization was implemented to utilize automated scripts at lesser cost and risk by using the service mocking feature in the automation tool chosen for creating the scripts.

Service Mocking
Web service mocking or virtualization allows creation of a simulation or approximation for a dummy execution before the actual web service is made live to remove dependency constraints on the development and testing teams. So while the software is yet under development, using service mocking, a mock response can be created according to the functional requirements and removing the constraint of waiting for actual services to be tested.

Maintainability

An important factor in progressive testing is to measure how quickly test scripts can be provisioned to accept new changes without much additional effort. To meet this target, code snippets in the form of reusable functions are created so that the same function can be utilized in any number of test cases. The test data is maintained in individual data sheets, allowing testing with multiple datasets (test data parameterization) to be performed effectively, saving time and making the testing process more effective.

Environment Handling

As virtualized environments are being used to implement the automated test scripts and for their dry run executions a provision should be there to easily replace it with the actual environment once the designated environment is ready.IGT implements provision of handling multiple environments during automation testing from the beginning of the development lifecycle

Executions and Results Sharing

When the automation test scripts are ready, they are executed against mock services. Later, these scripts are executed against the main service with minimum modifications, if required. The results are then shared with the stakeholders (development and business users) to ensure that the validations and verifications implemented in the test scripts are as per specifications. Once the developed code is ready to be tested, initially, a smoke run is executed, and if successful, the already automated test suite is executed.

The defects logged against the test cases are fixed in quicker releases where the executions can be performed multiple times with the same or different set of data values.

Below figure elaborates the complete process of Progressive Automation implemented for one of the SOA projects.

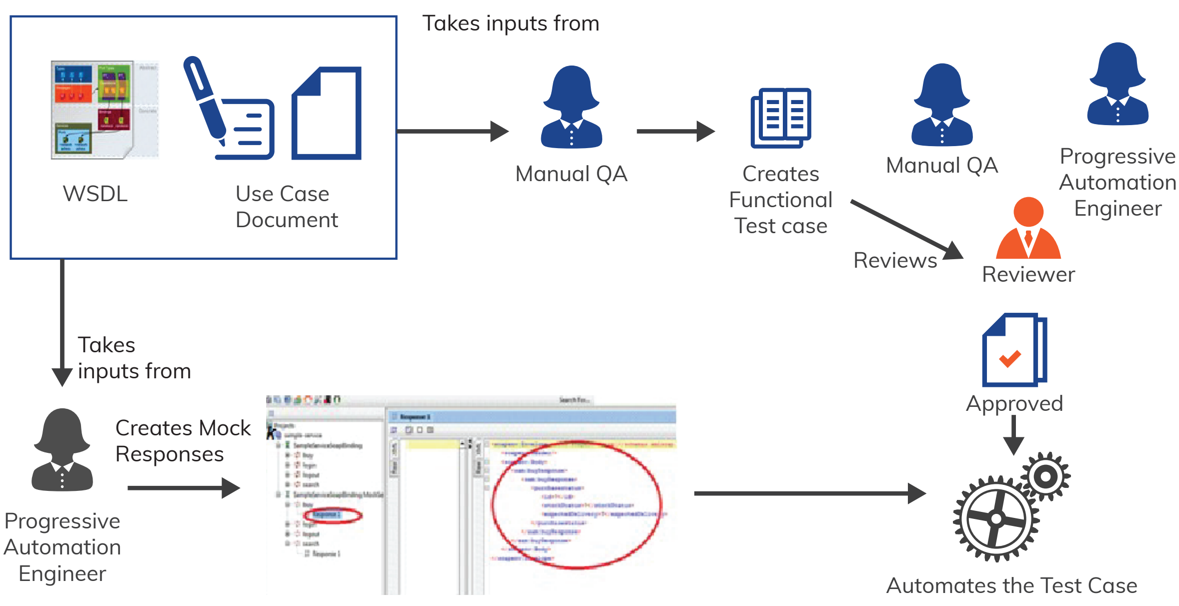

Fig.2: Process for Progressive Automation in one of SOA projects

Defects are logged against the test cases that failed. Once the defects are fixed the re execution of the test case with which it was associated needs to be done. The already automated test case is re executed and results analyzed again. This process goes on till all the defects are fixed for the spring or the required SLA is met.

Benefits

A few advantages seen by IGT in adopting the progressive automation approach in test automation are listed below:

About 40-50% of typical QA cycle is saved with this approach as depicted in graphical representation below. The saving is primarily in the execution phase with the use of automated scripts.

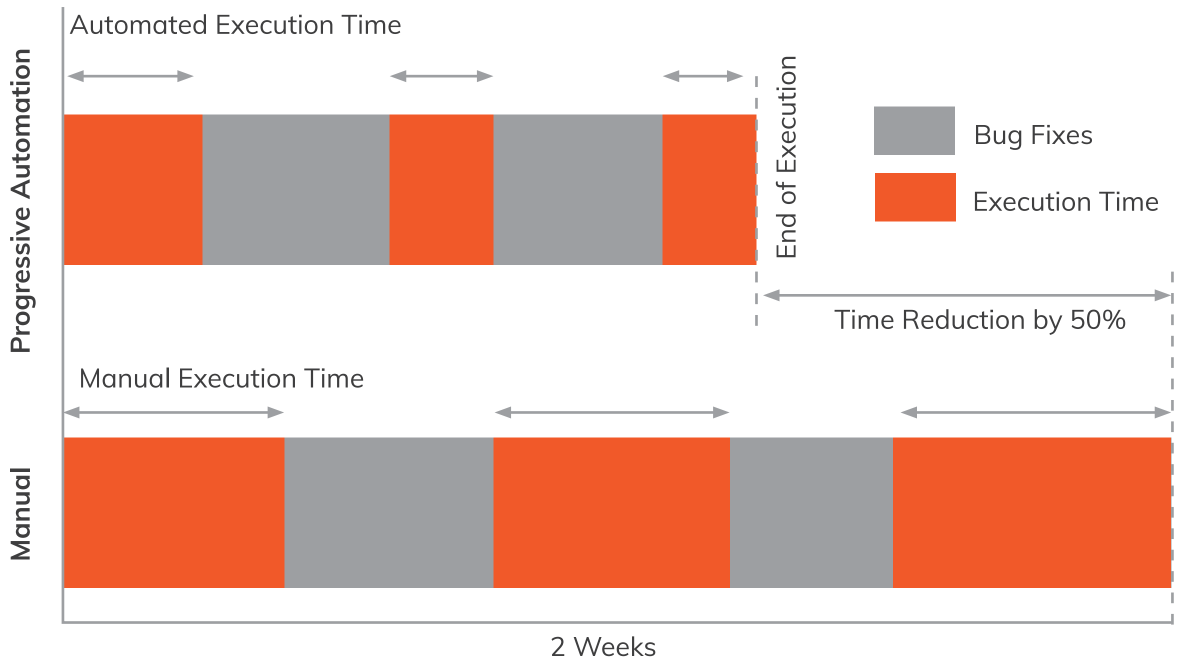

Fig.3: Automation time saving using progressive approach

- Progressive automation helps in effort saving in creating automated test cases for regression if compared to regressive automation. In case of progressive automation, effort would not be required to automate the regression suite; a subset of automated test cases from the progressive test suite itself can be utilized as regression suite. Thus, automated regression suite is built automatically.
- Early executions done on mock environment help to stabilize the automation scripts and apply validations. If any are missed a lot of time is saved in execution and analysis of results done on actual environment since test cases are already automated and verified and the automated executions are faster. Only the fail status in the results needs to be analyzed.
- The identified defects during the early stage of the development cycle reduce the effort and cost spent in fixing and retesting the same after fixing the reported bugs, the scripts can be re-executed multiple times in lesser time.
- Multiple Iterations of the test cases can be executed with any 'N' number of data values.
- By delivering operational and tested scripts on an incremental basis, progressive approach delivers increased value, visibility, and adaptability early in the development life cycle, significantly reducing project risk.

Challenges in Progressive Automation

As progressive automation testing teams work iteratively and in a different manner than other software teams, they must adapt their techniques of working to manage any challenges that might arise. Some challenges that might be faced are listed below:

- The completion and integration of the tasks performed by team members must occur in a limited time frame.
- The automation time and effort might be wasted if the functional test cases are not base-lined and reviewed at the initial stages.
- Repetitive maintenance might be required due to updates in the schema structures, data set or functionality

Suggested steps to address above challenges

Some measures that can be taken to address the challenges in progressive automations are listed below:

- Prompt decision making and prioritizing the tasks depending on the criticality of the application to avoid any unexpected failures.
- The scenarios as well as functional test cases created must be base-lined and reviewed at the beginning of the life cycle, reducing automation time and effort.
- Using parameterization and maintaining test data in separate excel sheets and using features such as wsdl refactoring maintenance to save time

Conclusion

We can easily determine that adopting the Progressive automation testing approach is cost effective and provides an optimal solution to testers and developers as well as organizations to meet today's racing and ever changing business needs. However proper planning and consideration of the implications of a longer term strategy to overcome the challenges it provides must be implemented by selecting an appropriate automated testing tool, automation framework and approach to be followed.

IGT Solutions (IGT) is a leading BPM, Technology and Digital Services and Solutions Company committed to deliver innovation and business excellence across the entire spectrum of Travel, Transportation and Hospitality domain.

Established in 1998, with 100% focused on the Travel industry, we have more than 70 marquee customers globally. IGT serves 4 in top 5 Airlines, 5 out of Top 5 Travel Companies, 4 out of Top 5 Hospitality companies. We provide digital contact center services, travel technology and innovative digital services and solutions for 100+ travel processes including Reservations and Sales, Customer Service, IROPS Management, Baggage Helpdesk, Crew Helpdesk, Chatbots, Robotic Process Automation, Travel Analytics and Social Media Services.

IGT
SOLUTIONS

IGT Solutions Pvt. Ltd.

Echelon Building, Plot No. 49,
Sector-32, Gurgaon - 122 001,
Haryana, India

T +91 (0)124 458 7000
F +91 (0)124 458 7198
mktg@igtsolutions.com
www.igtsolutions.com