CASE STUDY
QA Automation in an Agile Development Process

THE CLIENT
An Illinois-based insurance company that creates diverse software applications to meet the needs of their various lines of business.

THE CHALLENGE
Historically, the client has always used a manual testing method; however, they recently instituted a customized agile methodology with monthly sprints for new development projects. This change required integrating automated QA testing into the development cycle. The client needed to figure out a way to create a new process, adapt to the new monthly sprints, adjust their story boarding sessions, and develop automation scripts in parallel. Since IBS has provided QA support at this company for the past three years, the client knew they could rely on IBS to create a process that would allow them to adapt an agile approach. The client recognized the IBS QA team’s excellent domain and application knowledge, and wanted to utilize IBS’ blended onshore/offshore model to meet their needs, as they had done in the past. The IBS QA testing team would work offshore while the client’s agile team worked onsite.

Implementing QA automation into an agile process posed several challenges to both teams:
• What is the best configuration for the agile team?
• What are the Time Zone needs?
• Who determines the feasibility of taking the stories within the sprint?

THE SOLUTION
IBS understood the client’s need right away and quickly arrived at a process flow that satisfied the client. IBS determined that the team configuration was key to implementing a successful agile process; hence, a QA automation analyst from offshore was included in the agile team. This project required three agile teams, so IBS allocated one automation QA expert to each.

The agile teams consisted of:
• Agile Project Manager (Scrum Master)
• Business Analyst
• UI Designer/Functional Design Analyst
• Developers (offshore and onsite)
• QA Automation Analyst
• User Representative (for UAT and User Story validation purposes)

Taking into consideration the new monthly sprints, IBS adjusted the timelines for story boarding (one week),

TECHNOLOGIES USED
• Selenium Java
• Rational Functional Tester
design and development (two weeks), and testing/fixing the release (two weeks), with a week overlap between development and testing. After the application was released, the team worked on QA automation - primarily regression scripts.

The teams also quickly determined that, for the initial few release cycles, they would work in the same time zone to ensure availability while they fine-tuned the process.

The dedicated QA automation analysts worked full-time throughout the project, which helped validate the user story. The analysts understood how prioritizing the user stories and including them in the sprints held equal importance. The analysts' knowledge of the functional domain and the application were key to determining the interface touch points, thus aiding both white box and black box test script creations.

Once the user stories for the sprint were determined, the team discussed the functional validations, interface and interactive needs, and the UI needs. Upon determining these, the automation QA analyst began working on the functional and validation related scripts. While the user Interface design occurred, the QA automation analyst identified and developed the test data needed (both reference and transactional) for testing the user story. Once this was finished, an automation skeleton script was developed as a data element to be part of the UI, which was known at the end of the first week.

During the second week, the UI elements that were developed/modified during the day were sent to the QA analyst, who promptly developed/updated the skeletal links in the scripts. This process bridged the need to handle the fluidity of the agile process and the rigid needs of a QA automation process.

Finally, at the end of the second week, the automation script was tested for accuracy and functional validity. Even though it was not 100% ready, the agile process allowed usage of the completed scripts. The automation scripts continued to be developed during the third week, but there were parts of the applications that were simultaneously tested as well. The fourth week was primarily devoted to test execution (QA and UAT) and defect fixes, work that was very similar to the normal testing process. By the middle of the fourth week, a Go/No-Go decision was made and accepted, so the story was marked as ready for release.

**THE RESULT**

IBS created a customized agile development process with offshore QA automation. Over the span of 12 months, this process went through only minor refinements. After gaining experience with this customized process, IBS was able to optimize working in multiple time zones and use it to their advantage: the QA automation was always a step ahead of the development to assist in a hurdle-free agile development process.

### The Current Work Process:

1. **Client provides the manual test case documents and the test data to create the automation scripts.**
2. **Offshore team helps client identify regression scenarios and test cases, transforming workflows into test scripts and starting development.**
3. **A daily standup call is used to discuss the script work status, and if there are any road blocks.**
4. **QA team uses Rational Functional Tester and Selenium Java as automation tools.**
5. **QA team executes regression automation scripts for every new build, ensuring the regression test cases pass. If any errors in the script’s execution are identified, the team examines the execution report and verifies whether it is a script error or an application error. If it is an application error, the respective developer will be assigned the defect using the test management tool. If it has something to do with automation script, the QA team will work on and maintain the scripts.**
6. **The execution report is sent to the onsite QA manager. The report includes: line of business (workflows), business type, platform (web or mainframes), function (new policy creation or claims), number of policies created, minutes taken to create each policy, time taken to create all policies, test result status, created policy numbers, and detailed defect description.**
7. **We use data-driven, page object modeling design, keyword-driven, and hybrid frameworks for automation scripts. The frameworks provide an outline of overall test structure, ensures testing consistency, minimize the amount of code for development (less maintenance), maximize reusability, reduce the exposure of non-technical testers to code, and enable test automation using data.**