Shift-left Testing

- Overview
- Shift-Left Journey
  - Early Testing
  - Suggested Guidelines to Accepting Stories:
    - Accelerate Automation
    - Test Maintenance
    - PR Quality
  - Timelines
  - Conclusion

Overview

Shift left testing is a highly effective and innovative approach to software testing. It involves commencing testing activities earlier in the software development lifecycle (SDLC). Unlike the traditional method, where testing is performed after the development process is completed, the shift left approach involves running testing in parallel with the development process. The purpose of embarking on a shift-left journey is to improve the overall quality, efficiency, and effectiveness of software development. By identifying and addressing issues at an earlier stage, the ultimate objective is to enhance the quality of software, reduce time-to-market, and promote better collaboration among development teams.

Advantages

- This approach enables the detection of issues at an early stage in the SDLC.
- Testing saves time and money in the long run.
- Improved software quality earlier, allowing fixes before the release deadlines.
- It helps focus on preventing defects from occurring instead of just detecting them.

Shift-Left Journey

We're excited to kick off our shift left process! To make it happen, we're starting with a few key initiatives that will help us get there. As we implement these initiatives and gain experience, they will continue to evolve. For all official changes, we'll keep everyone in the loop by sending out email updates and keeping this wiki up-to-date.

These initiatives include the following.

1. Early testing - IN Sprint testing.
2. Accelerate Automation - Focus AQA team more on automation, Centrally manage automation work.
3. (future) Test Maintenance – The teams fix daily test failures of existing tests
4. (future) PR Quality – Each PR will execute e2e tests and Karate tests.

Early Testing

When conducting early testing, the aim is to finalize the coding and testing of the story in the designated sprint without any unresolved bugs that may require attention in the subsequent sprints. The objective is to conduct early testing in a bug fest environment to prevent the late discovery of issues during bug fest testing.

Starting: Sprint 168 Early Testing process is adopted.

The sprint timeline is split into 70% development and 30% hardening. The development teams are prioritizing their coding activities until the second Wednesday of the sprint. Afterward, they redirect their attention toward testing and improving the features, hardening the code, and bringing all testing IN sprint.

On the second Wednesday of the sprint, the Kitfox team refreshes the Sprint Test environment with the bug fest dataset. The automated integration (Karate) and UI end-to-end (Cypress) tests are executed and analyzed in this environment.

The MQA team continues to develop test cases and review them with POs throughout the sprint. MQA test stories in both Snapshot (built from scratch with reference dataset) and Sprint Test (migrated Bugfest dataset) environments to cover new customer scenarios as well as existing customer (upgrade) scenarios.

For now, the AQA team continues to analyze test failures from the daily nightly tests and Sprint testing environment.

Product Owners are expected to work with Devs and Testers in the same fashion as before. A PO can continue testing, reviewing, and approving stories in Snapshot or, if they choose to, in Sprint Test (if, for instance, they feel the migration can impact the feature)

POs will CLOSE stories after QA complete their testing (we should never close untested stories!). For example, say the PO reviewed and approved the story in Snapshot, then they may want to look at it again in Sprint Test after QA is finished before closing it, OR they can review the results of QA testing and if no bugs have been reported the PO can go ahead and close the story. If bugs have been reported, then the PO needs to decide whether the found bugs need to be fixed before closing the story or the story can be released as is (so the PO then goes ahead and closes the story). In the latter case, the bugs found will be deferred to the next release.

Jira Workflow: Once the code review is complete, the developer moves the story to “In QA” status. QA tests these stories in Snapshot and Sprint Test environments and moves them to “In Review” status. POs accept and close these stories.
Suggested Guidelines to Accepting Stories:

1. The development team has demonstrated the story to POs. This should be done as soon as the feature is demo'able (no need to wait until it's fully tested). This can be done in Snapshot (or even Rancher). Basically, no changes to the exiting process. EXCEPT - the POs should not accept this story at this point.
2. The POs perform business-level testing in Snapshot to ensure the story meets their criteria. QA will likely start their test cycle once the PO approves it, but they can also start sooner if they choose to.
3. Once QA has completed testing of the story, the PO can re-review the feature in Sprint Testing env OR review the QA test results to assess whether the story meets the expected level of quality.
4. If there are no bugs, PO closes the story. If bugs have been reported, then the PO needs to decide whether the found bugs need to be fixed before closing the story or the story can be released as is (so the PO then goes ahead and closes the story). In the latter case, the bugs found will be deferred to the next release.

Accelerate Automation

This initiative is to significantly improve our automated test coverage. Increasing automation test coverage is crucial as it has been proven to be a reliable and efficient method for detecting bugs at an earlier stage of the testing process.

**Starting:** Sprint 170, automation work will be managed centrally.

To increase the AQA team’s focus on automation, the AQA engineers are organized in a separate team with all the scrum team’s ceremonies, Jira board, etc, and a dedicated scrum master. Yogesh will lead the team. POs will provide test case priority, such as Smoke/Critical/Extended, for automation. Obviously, the AQA engineers are free to work with feature teams and individual developers/testers as they see fit. They can also participate in feature teams’ DSUs/refinements/reviews/demos as they see fit.

The AQA team will develop new tests from the backlog or new features based on Smoke/Critical/Extended priority and will continue to maintain existing e2e test cases.

Furthermore, an assessment of the test automation backlog will be conducted to classify which tests ought to be automated in either end-to-end, unit, or integration suites.

Our end-to-end test suites will now include automated testing for user journeys to ensure better coverage for the end-user tasks.

Test Maintenance

This initiative is to have developers take ownership of the existing test maintenance from the AQA team. This is a very effective and efficient approach for the automated tests, as well as the industry standard. Since developers are very familiar with codebase and troubleshooting, they can quickly identify the root cause of failures by examining the code (or better yet, adjust the tests as they’re adjusting the screens), allowing for faster resolution of test issues. When developers make changes to the application code, they can maintain synchronization between code and tests, reducing the risk of outdated and ineffective tests. Developers can closely share insights and knowledge to improve the test suites with the automation testers.

**Starting:** TBD

PR Quality

This initiative is to improve our Pull Request Quality gate. Currently, PR Quality includes static analysis, code review, and unit tests. We will include running relevant integration and end-to-end tests along with modified code. Performing testing during PRs is essential to proactively detect and address issues before they penetrate the main codebase. The implementation of this PR quality gate is crucial to prevent regressions and ensure that developers receive timely feedback on the impact of their code changes. Furthermore, code reviewers can rely on the test results as an objective evaluation of code quality. This approach saves both time and effort, while also guaranteeing early stability.

**Starting:** Sprint 169 - Only Kitfox and SA

- Feature teams - TBD

PR quality gate has infrastructure development work to be done by the Kitfox team. The team will implement a test environment and pipeline to run selective tests. AQA team members create selective test groups with relevant end-to-end tests to run based on the code changes. An automated pipeline will be designed and implemented to run selective tests.

For feature teams, Step by step guide shall be created on how developers can run these selective tests.

Timelines

1. Early Testing – Started Sprint 168, Teams (Firebird, Foliojet, Spitfire, Thunderjet, Vega, and Volaris)
2. Accelerate Automation - Starting Sprint 170 (July 17th) - Teams (Firebird, Foliojet, Spitfire, Thunderjet, Vega, and Volaris)
3. Test Maintenance – TBD (on hold)
4. PR Quality – Started sprint 168, Teams(Kitfox, Solutions Architecture (SA), Feature Teams (NO ACTION required - TBD

Conclusion
The shift-left journey emphasizes proactive approaches, early testing, and continuous integration to catch and address issues as early as possible, leading to more efficient and successful software development projects. As we embark on this journey, we shall continue to collect feedback and evolve as we gain experience.

It involves shifting tasks traditionally performed later in the development process to earlier stages. The goal of the shift-left journey is to improve the quality, efficiency, and effectiveness of software development by detecting and addressing issues earlier. Overall, the shift-left journey aims to improve software quality, reduce time-to-market, and enhance collaboration among development teams. It emphasizes