White Paper

Information Technology
February 2023

IT@Intel: Accelerating Business Value and Improving Quality Through Large-Scale Test Automation

Through the use of agile methodologies and six test quality practices, Intel IT reduced the testing time of a software rebate management solution by 79 percent and created a reusable test-automation framework with an efficiency ratio of 75 percent.

Intel IT Authors

Ayyappadas Mohandas
Quality Leader

Biju M Aliyas
Director, Software Engineering

Michael P Bush
IT Program Manager

Jay Krishen
Industry Engagement Manager

Table of Contents

Executive Summary 1
Background 2
Challenges 2
Possibility Thinking 2
Solution 3
Frameworks, Methodology, and Accelerators 7
Results 8
Conclusion 9

Executive Summary

Intel IT planned the deployment of the Vistex rebate management solution, based on the SAP platform, over three releases. The solution was part of a portfolio of modernization initiatives through which Intel sought to standardize its contract structures, rebate accruals, and settlement and payment processes. After Release 1 of the program was deployed, new business requirements dictated that internal users and external customers needed the solution sooner than initially planned.

The team accelerated deployment of Release 2 and Release 3 by applying agile methodologies including test quality practices like end-to-end integration testing. As a result, Intel IT was able to meet the new schedule commitments. Compared to Release 1, the team reduced the testing time of Release 2 by 50 percent and Release 3 by 79 percent. The team achieved these results even as the number of test cases rose in each release: compared to Release 1, test cases increased by 25 percent in Release 2 and by 24 percent in Release 3.

The team attributed much of the reduction in delivery time to automation. By Release 3, Intel IT had automated 66 percent of the end-to-end integration tests and 94 percent of the regression tests. Test-automation efficiency (defined as the ratio of effort saved by automating testing to the effort required for manual testing) was 74 percent for both Releases 2 and 3. Intel IT achieved these levels of accelerated delivery and test efficiency while remaining within the original budget and resources. The team also created a reusable testing framework that can now be applied to other programs. Post-program efficiency was calculated at 75 percent.

By delivering Release 2 and Release 3 on schedule and with high quality, Intel's internal stakeholders were able to onboard customers faster and gain access to real-time financial information sooner. This allowed Intel's business groups to respond to market demands more rapidly with faster business decisions.

This paper shares how Intel IT accelerated business value and improved quality through a large-scale test automation program. It describes the team's use of possibility thinking to jumpstart the pivot to agile methodologies, and it outlines the six test quality practices that the team adopted. The paper also describes Intel IT's development of a test-automation framework, highlighting both automation efficiency and end-to-end integration testing. By sharing details of the program, Intel IT hopes to enable other teams to build a framework that delivers quality software faster, leading to more rapid realization of business value.

Abbreviations

API: Application programming interface

CI/CD: Continuous integration and continuous delivery

ERP: Enterprise resource planning

MVP: Minimum viable product

SDLC: Software development life cycle

SOA: Service-oriented architecture

Background

Intel is an industry leader that continuously works to advance customer experiences through the design and manufacturing of semiconductors and other products. Within its customer base, the company caters to multiple segments and manages an ever-increasing number of sales channels.

As part of its sales process, Intel sometimes provides rebates to its customers. To track these rebates, the company had used a set of discrete processes storing data in spreadsheets. This manual approach was time-consuming, effort-intensive, carried the risk of human error, and was difficult to manage and track. As its business grew, Intel determined that it should automate and streamline this business process. Transforming the process created the potential for:

- Automating claims and payouts
- Increasing visibility into go-to-market programs
- Enabling profitability analytics through automated and real-time accrual management

Intel IT began this modernization program initiative by aligning processes, data, architecture, and solutions with a reference architecture guideline that outlined how to deliver a Vistex solution embedded within the SAP platform. The industry practice is to deploy this type of software solution using a waterfall methodology because of the solution's complexity and many dependencies. Intel IT began the program using this approach.

The technical implementation of the program was broken into three phases: Release 1 (minimum viable product or MVP), Release 2, and Release 3. The program was complex due to the many business processes, stakeholder groups, and IT systems involved, including custom solutions.

After Release 1 was deployed, new business requirements dictated that both internal users and external customers needed the solution sooner than originally planned. To address this new schedule requirement while maintaining budget levels, Intel IT made a directional change away from the waterfall methodology used for Release 1. Because of Intel IT's extensive expertise with agile methodologies, in addition to years of experience deploying ERP software solutions, the team successfully pivoted to using an agile methodology with a focus on six test quality practices. These changes enabled Intel IT to deploy the solution on schedule while staying within budget. More importantly, Intel IT was able to create a testing framework that could be reused for subsequent programs.

Challenges

Intel IT successfully delivered the foundational MVP capabilities of the solution in Release 1. However, during that time, various environmental dynamics were at play, which ultimately demanded that Intel IT reassess its program delivery for Releases 2 and 3. These dynamics included the following:

- Business requirements demanded accelerated delivery.
- New features needed to be assessed and added in Release 2 and Release 3.
- Budgetary constraints necessitated increased operational efficiency.

The magnitude and complexity of ERP software programs can make it challenging to apply agile techniques. Traditionally, companies have used a waterfall methodology to control and coordinate the many moving parts of ERP software deployments. However, given the business need to accelerate delivery, Intel IT changed direction and adopted an agile methodology.

Possibility Thinking

To achieve an accelerated deployment schedule, add new features, and maintain a flat budget, the Intel team made use of possibility thinking—that is, the widening of perspectives to embrace new possibilities that were previously beyond expectation or experience. The team:

- Brainstormed agile delivery practices that would provide testing feedback earlier, identify defects sooner, and enhance test coverage.
- Researched tighter quality-governance best practices for managing and monitoring testing across multiple teams.
- Planned for shift left testing—that is, beginning testing earlier within each development cycle and ensuring quality throughout the program.
- Strategized how to increase automation to validate all end-to-end scenarios where complex calculations were made.

- Outlined a defect-evaluation process. This process included monitoring trends for both open and closed defects. An increase in the gap between open and closed defects would indicate the need for resource augmentation. This process also included monitoring the variation of defect types. If one type of defect spiked, the team would be able to track the source to its cause, such as a change in design, code, or configuration.
- Planned for the use of CI/CD to support the agile methodology. Agile methodologies can result in multiple intermediate releases. These releases benefit from automated tests that continuously verify new functionality and automated deployment services that deliver new features to end users.

Solution

As a result of the team's possibility thinking exercises, Intel IT adopted six test quality practices for Releases 2 and 3 (see Figure 1). Each of these practices can be adopted individually. However, Intel IT implemented them as a set because they are connected and complement each other. These best practices are described in the following sections.

Figure 1. Intel IT adopted six test quality practices in its pivot to agile methodologies.

1. Test Planning

Intel IT had to adapt to a new method of estimating the size of each feature. Breaking features down into user stories and then further classifying them by size as small, medium, large, or extra-large was a radical departure from the waterfall methodology. This impacted the testing process because large and extra-large features were significantly more complex and required more pervasive integration testing.

To support this new approach, the team crafted a revised test plan for Releases 2 and 3. This step was essential for moving to an agile development model. Key elements of the revised test plan included a revised schedule, an updated scope definition, plans for resource allocation, and details on how the test environment would be shared. With a comprehensive and well-crafted plan, the Intel IT team could manage the new, variable, and dispersed timelines associated with agile development. The revised plan allowed the team to integrate testing into four-week sprints and include customer feedback as part of validation testing.

2. Quality Governance and Management

Intel IT updated its quality governance and management framework, a necessary precursor to pivoting to an agile methodology for Releases 2 and 3. This step included:

- Defining, communicating, and clarifying the roles and responsibilities of every team member.
- Updating definitions of the scope, test phases, and processes, including updating entry and exit criteria for each testing phase and defining environmental practices.
- Consistently evaluating test metrics. Tracking defect metrics visually in graphs allowed the team to gain insight into product and build quality and to gauge the overall health and effectiveness of each release. This step included visualizing defect trends, defect ratios, the number of defects per feature, and the cost of rework. This type of defined test-management approach can help speed up the deployment process.
- Implementing new program-management software platforms, including a defect-management tool that would track testing results and progress. The team published daily reports to all stakeholders.
- Having daily meetings during testing to discuss defects and progress. These meetings included periodic reviews of the scrums, in addition to bug-scrub meetings to review phase goals such as overall test progress. Reviewing these goals helped ensure smooth interactions among teams for testing.

3. Test Strategy

Intel focused its revised test strategy on shift left testing and on automation. The shift left testing concept is illustrated in Figure 2.

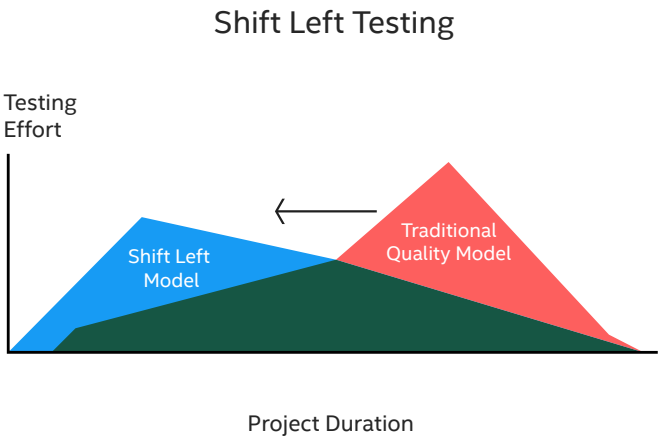

Figure 2. A shift left approach to testing increases testing effort earlier in the development cycle.

The philosophy of shift left testing is based on the theory that most software defects can be identified during the requirements phase, the first phase of deployment, resulting in fewer defects emerging during the later development phase of the life cycle. Shift left testing can also help reduce costs. For example, if a vulnerability can be detected early in the development process, it might cost hundreds of dollars to correct. But if the same vulnerability is detected after the solution is moved to production, it might cost thousands of dollars to correct.

For this program, Intel IT applied the shift left approach by deploying releases into the SAP quality environment earlier. This paved the way for earlier testing and created a faster feedback cycle. For example, 60 percent of defects were identified in earlier functional and integration testing and 15 percent of defects were identified in later regression testing.

Intel IT implemented shift left testing in other important ways:

- The team broke down the development and test processes into sprints, or iterations, with a release to testing every four weeks. Each sprint resulted in either a prototype or a workable version of the final solution. At the end of a sprint, in addition to testing, customer or user feedback determined whether the solution was rejected or developed further in the next sprint. This enabled the development team to know much earlier whether changes were needed, thus accelerating the schedule.
- The automation team started scripts early so that automated tests were available for end-to-end integration testing and regression testing. The availability of automated tests contributed to shortening the schedule.
- Intel IT tested features as they became available, rather than waiting until the end of the development cycle. Developers demonstrated features to customers for validation almost immediately. These two changes facilitated faster feedback for the developers.
- The team partnered with both customers and stakeholders to understand end-to-end scenarios. They tested these scenarios using real-time use cases.

End-to-End Testing

Intel IT implemented end-to-end integration testing in Release 2 and Release 3. Because modules are often coded by different programmers, integration testing exposes defects in the interactions between software modules when they are integrated.

Validating the data flows from the first point where data is introduced and all the way to the last stage helped Intel IT simulate the data flow that users would experience. This deeper and more comprehensive level of testing, done at the integration testing stage, helped ensure that defects were caught earlier, saving both time and money. It also benefitted the team through reduced rework.

With advancements in automation tools and technology, Intel was able to create automation scripts for integration testing. These automation scripts allowed immediate testing. The team did not have to wait for all functionalities to be delivered. End-to-end integration testing was incorporated into the continuous delivery pipeline so that automated end-to-end scripts self-triggered, which helped save time and accelerate defect discovery.

4. Test Automation

Intel IT invested in test automation to reduce test time and accelerate the deployment of Releases 2 and 3. Automating tests for ERP deployments has traditionally been considered difficult. However, automation is a key component of agile methodology, so it was prioritized by the team.

The primary focus of automation for this program was to validate all end-to-end scenarios in which complex calculations were made. A secondary focus was to build accelerators—such as automating the creation of billing data—to reduce the time needed to create data and upload templates.

Intel IT adopted several tools and methods to increase automation. The team:

- Developed a test-automation framework.
- Developed robust end-to-end automation scripts that used manual scenarios and that could be used across all internal and external customers.
- Ensured that all possible validation and verification scenarios were covered.
- Automated the testing of complex rebate calculations.
- Used in-sprint script development to gain the benefits of automation in the initial phases of testing.
- Developed a script to support the creation of billing data.
- Developed API scripts using industry tools to support upstream data for activities like deal creation.
- Developed SOA scripts for interface testing to validate deal information.

In stark comparison to Release 1, in which none of the integration tests were automated, Intel IT automated 45 percent of the end-to-end integration tests in Release 2 and 66 percent in Release 3 (see Figure 3). Similarly, the team automated 85 percent of the regression tests in Release 2 and 94 percent in Release 3. The extensive use of test automation was a substantial factor contributing to the decrease in testing time.

5. Test Evaluation

Intel IT completely assessed the quality of the software during each test phase using defect trend analysis and category analysis. The team tracked trends to understand overall test execution progress and to compare planned and actual results. This tracking also enabled the team to identify any bottlenecks in the process, which could then be fixed.

Defect Trend Analysis

Intel IT assessed defect trends daily using defect trend analysis. This approach allowed the team to track the caliber of each build. If cumulative open defects spiked, the change triggered the team to take corrective action.

Figure 4 illustrates defect trends: cumulative open defects and cumulative closed defects. Points A and B in the cumulative open-count trendline illustrate spikes that indicate the occurrence of a systemic issue. To investigate these spikes, the team evaluated defect density (total defects divided by total test cases) to identify the specific test cases in which defects were increasing.

To close the gap, the team applied additional resources and fixed defects in their order of priority and severity. Corrective action included boosting smoke testing, improving quality gates through code reviews, conducting risk-based testing, and re-executing automated regression tests.

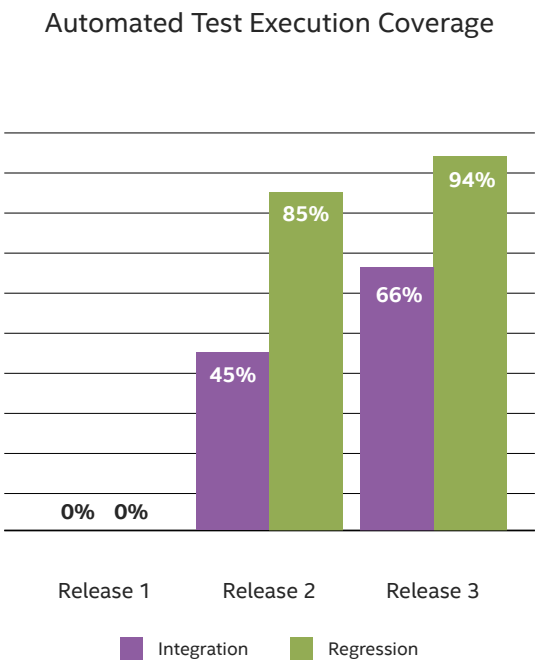

Figure 3. Intel IT increased automation of integration and regression testing from Release 1 to Release 3.

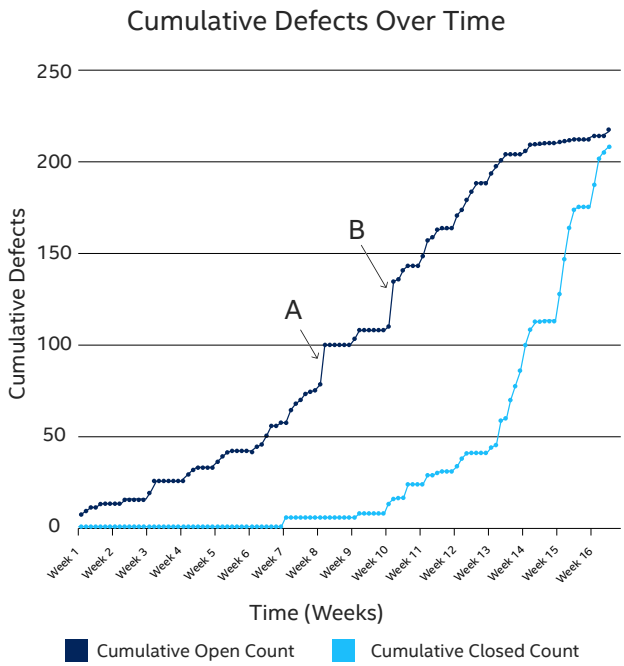

Figure 4. Cumulative open defects trends and cumulative closed defects trends.

Defect Category Analysis

Intel IT also used defect category analysis to understand the variation of defects. The team investigated the root cause of each defect and charted this information (see Figure 5). This approach gave the team a tool to understand whether a change in test strategy—whether in design, code, or configuration—was inducing defects. For example, Figure 5 shows 115 defects in the "Code Change" category.

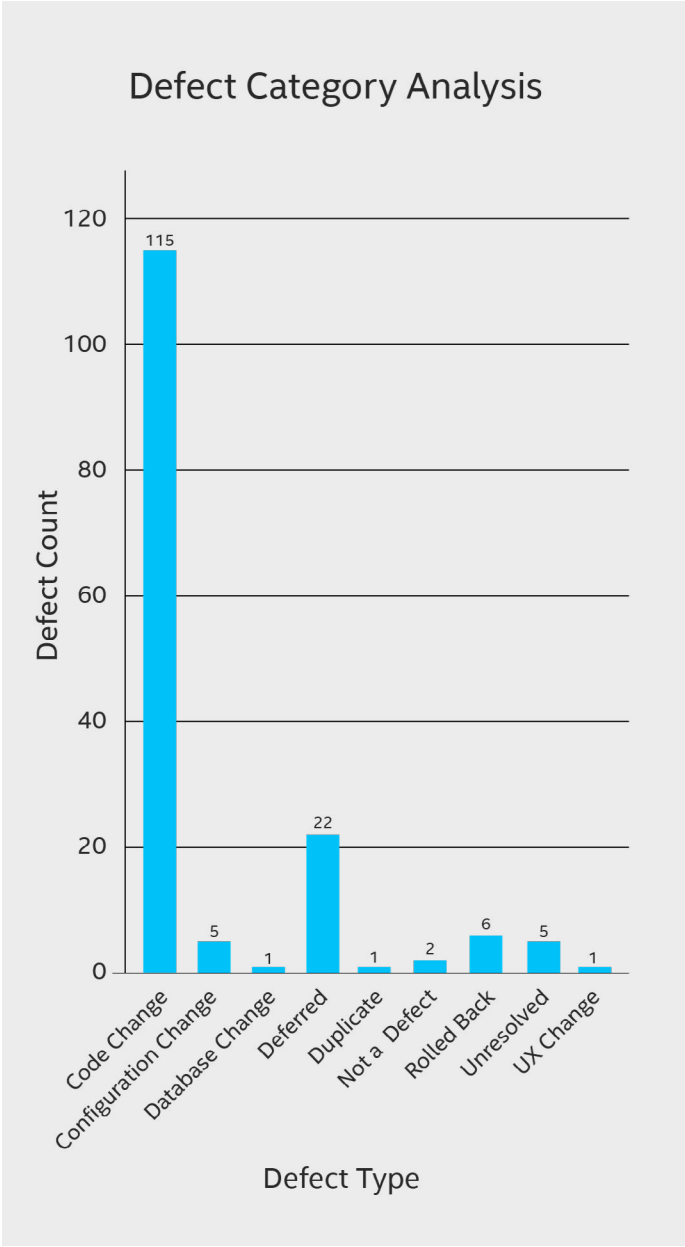

Figure 5. Defect category analysis provided the team with a visual indication of which areas needed attention.

Defect category analysis clearly identified which problem areas needed attention. Depending on the category, the team could apply corrective actions like creating detailed designs incorporating real-world scenarios, improving prototyping, improving technical unit testing, evaluating test scenarios with end users, or executing test scenarios.

The team used defect trend analysis and category trend analysis to identify defects and their underlying causes faster and to correct them swiftly. As a result, rework was significantly reduced.

6. Continuous Testing

Intel IT employed a third-party CI/CD tool in Releases 2 and 3. As Intel IT adopted the agile methodology with its associated multiple intermediate releases, CI/CD was a critical component of the SDLC that helped the team discover defects earlier and accelerate release cycles.

CI/CD helped bridge the gap between development and operation activities (DevOps) by enforcing automation in the building, testing, and deployment of applications. CI/CD services compiled the incremental code changes made by developers and then linked and packaged those changes into software releases. The team then used automated tests to verify the software functionality and automated deployment services to deliver software to internal and external users.

Figure 6 illustrates how CI/CD combines the development-related activities, such as designing, coding, building, and testing, with the operational aspects, such as deploying, operating, monitoring, and managing. The standard DevOps cycle is enhanced with an additional focus on developing code that is both security-enabled and optimized for performance.

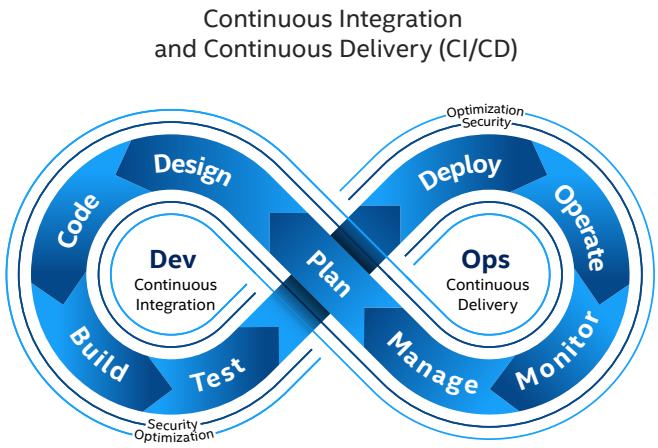

Figure 6. The use of CI/CD helped Intel IT uncover defects earlier.

Frameworks, Methodology, and Accelerators

Intel IT used a collection of frameworks, methodologies, and accelerators to support automated testing of the solution. The framework that Intel IT developed integrated various functions such as libraries, test data, and reusable modules. This section describes the team's selection of a hybrid automation framework. It also explains how the program benefited by using an automation framework.

Selecting an Automation Framework

Intel IT selected a hybrid automation framework, combining a data-driven framework with a keyword-driven framework. A data-driven framework involves fetching input and output values from data files such as Microsoft Excel, recordset, comma separated values (CSV), and Open Database Connectivity (ODBC) sources.

A keyword-driven framework, on the other hand, is created with a set of keywords and data tables that do not depend on the automation tool used by the organization or the test scripts powering the data and the application being tested.

In its framework, the Intel IT team used both the built-in recordset feature of its automation tool and Microsoft Excel files to pass data into the application under test (AUT). Intel IT designed scripts by using the built-in keywords of its automation tool. The framework was a robust solution that successfully addressed existing challenges, delivered optimum test coverage, and helped ensure time and cost efficiencies. Notably, the solution was designed in such a way that it could be reused in future programs. Figure 7 illustrates the components of the automation framework created by Intel IT to deploy the solution on an accelerated schedule.

Benefits of an Automation Framework

The program benefitted from key characteristics of the automation framework. These benefits included:

- **Accelerated script development:** Automated development of scripts and test cases reduced development efforts by up to 80 percent, as compared to traditional test-automation approaches.
- **Modularity and reusability of framework components:** Because the framework supported reusable components, it helped reduce test-automation maintenance costs.
- **Easy test script maintenance:** The framework made it easy to execute tests and easy to make changes and maintain scripts. It also improved error and exception handling.
- **On-demand testing:** The automation framework supported smoke, integration, functional, and regression testing, in addition to other testing processes. It provided effective validations on the go.
- **Hybrid model:** The automation framework was modular and could be integrated with several other automation tools, including CI/CD.
- **Scheduling and reporting:** The framework could be used to schedule tests using a third-party CI/CD tool and to e-mail reports to stakeholders when testing was complete.
- **Automated testing:** The framework was integrated with a third-party CI/CD tool to enable automated testing.

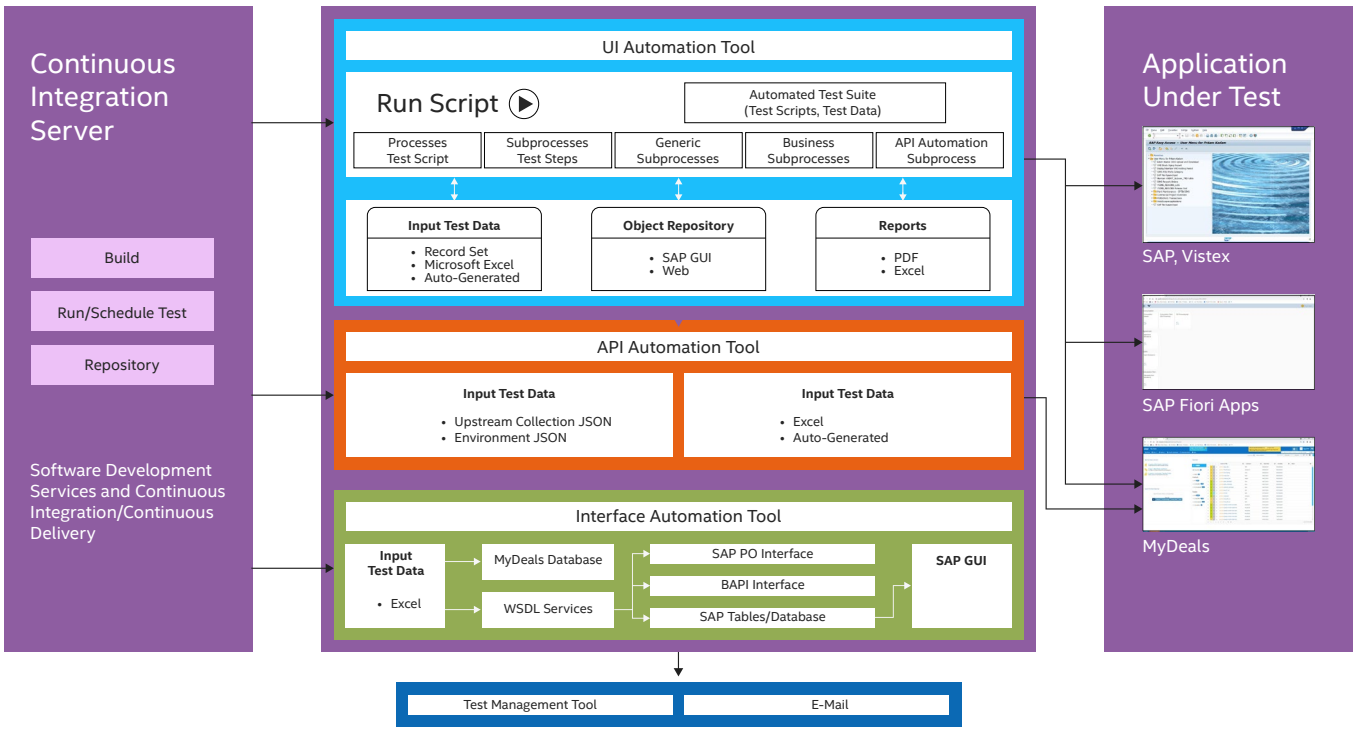

Figure 7. Intel IT created an automation framework to support automated testing.

Results

Intel IT achieved significant results by adopting an agile methodology and using test quality practices. Figure 8 illustrates the dramatic decrease in testing time (in weeks) for Release 2 and Release 3, relative to Release 1.

Compared to Release 1, the Intel IT team was able to reduce the testing time for Release 2 from 14 weeks to 7 weeks—a 50 percent decrease—even as test cases increased by 25 percent (see Figure 9). Compared to Release 1, the Intel IT team was able to further reduce the testing time for Release 3 from 14 weeks to 3 weeks—a 79 percent decrease—even as test cases increased by 24 percent.

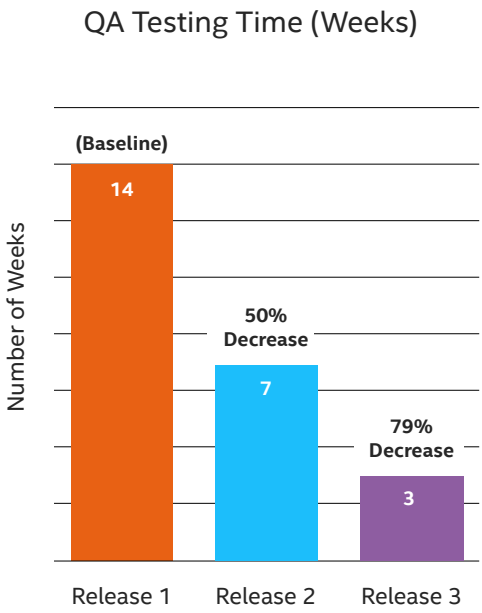

Figure 8. Decrease in testing time (in weeks) over the three releases.

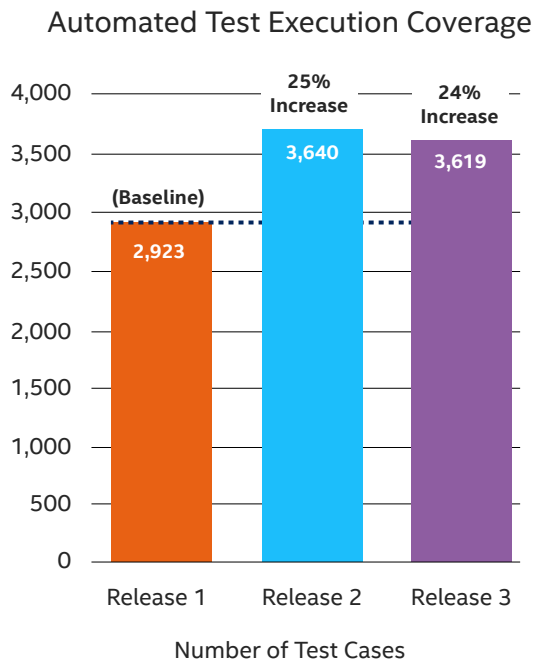

Figure 9. The number of test cases was higher for Releases 2 and 3, as compared to Release 1.

The decreases in testing time helped ensure that the team got the new solution into the hands of internal and external customers on schedule.

Automation Efficiency

Intel IT invested approximately 1,200 person-hours to develop automation testing for both Release 2 and Release 3.

Test-automation efficiency is defined as the ratio of the effort saved by automating testing to the effort required for manual testing. Figure 10 illustrates the efficiency of automating the development of billing data. Across Releases 2 and 3, the overall efficiency was 74 percent. The team saved a total of 13,511 person-hours, which is equivalent to 1,689 person-days. This time would otherwise have been spent creating billing data for the program, and so the automation effort yielded a positive return on the investment.

Integration Testing Efficiency

Although the average number of features delivered across all three releases was similar, the complexity of the features was not. In Release 1, 65 percent of the features were designated as low complexity and 35 percent were high complexity. During Release 3, on the other hand, only 22 percent of the features were of low complexity and 78 percent were high complexity.

This skew towards an increasingly complex set of features should, ordinarily, have resulted in an increase in testing time. Instead, the team completed the testing for Release 3 in 3 weeks as compared to 14 weeks for Release 1. Testing more complex features in Release 3 in a reduced amount of time is counterintuitive, but it demonstrates the extreme efficiency that the team was able to achieve during the integration testing process. This efficiency can be attributed to the combined effect of using all six test quality practices in tandem.

The team's testing framework will continue to serve Intel IT well into the future. The suite build-out will be reusable across all types of testing. For example, the regression suite built for Releases 2 and 3 has been reused eight times since the completion of the program, achieving an efficiency of 75 percent.

Conclusion

Intel IT met its goals of accelerating the solution deployment of Releases 2 and 3, meeting the needs of both internal and external customers. The dramatic reduction in testing time, despite an increase in the number and complexity of test cases, was possible only because the team used an agile methodology, a shift left testing approach, and increased automation with a greater depth of testing. This resulted in fewer errors, issues, and defects, which led to less rework and faster deployment.

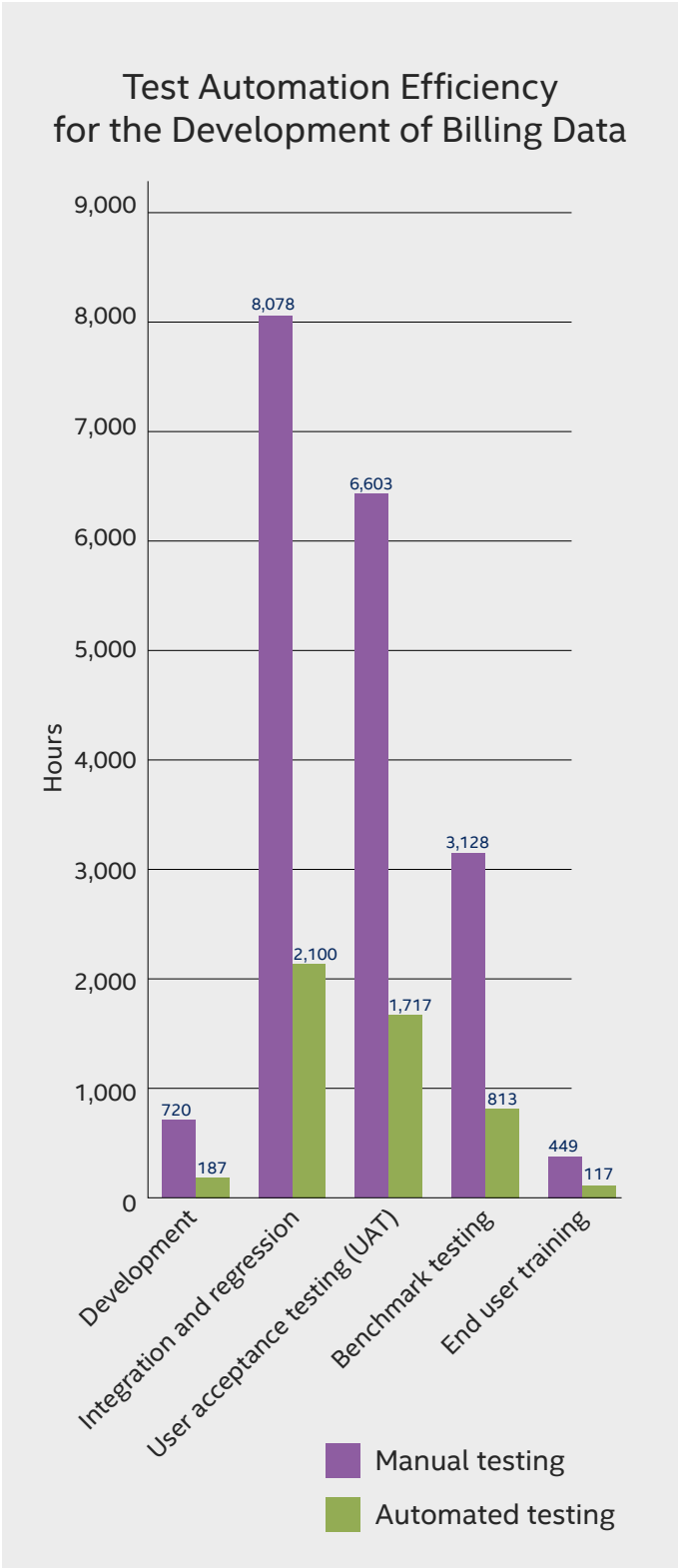

Figure 10. Automating tasks like the development of billing data saved time.

The investment in automation yielded 74 percent efficiency, confirming that Intel IT had made a sound, successful business decision. With increased automation, fewer team members were needed for testing, which freed up resources to work on new features. Finally, the use of standard tools and accelerators allowed for greater interchangeability of resources.

The test framework demonstrated that automation and accelerators in enterprise applications can increase production reliability. Automation can handle tasks that are easily prone to error, which might require additional staff or that might be too tedious for staff to perform.

Enhancing quality through large-scale automation also enabled Intel IT to enhance the skillsets of its developers and create a growth mindset. The result was better software with faster execution, leading to rapid realization of business value.

Because of the predictable deployment time and high-quality software, Intel's internal stakeholders were able to onboard customers faster and gain access to real-time financial information sooner. Intel's business groups were able to respond to the market's demands with more rapid business decisions.

Related Content

If you liked this paper, you might be interested in these other stories:

- Intel IT's Agile Journey Toward Scalability and Transformation
- Master Data - Managed!
- Accelerated Analytics Drives Breakthroughs in Factory Equipment Availability

For more information on Intel IT test-automation capabilities, visit intel.com/IT or contact your Intel account manager.

IT@Intel

We connect IT professionals with their IT peers inside Intel. Our IT department solves some of today's most demanding and complex technology issues, and we want to share these lessons directly with our fellow IT professionals in an open peer-to-peer forum. Our goal is simple: improve efficiency throughout the organization and enhance the business value of IT investments. Follow us and join the conversation on Twitter or LinkedIn. Visit us today at intel.com/IT if you would like to learn more.

Printed in USA 0223/DP/PRW/PDF Please Recycle 353691-001US