# Has Agile Helped or Hurt Testing?

**Jeffery Payne**
**@jefferyepayne**
**Coveros, Inc.**

**Robert Sabourin**
**@robertasabourin**
**Consultant, Professor**

**Johanna Rothman**
**@johannarothman**
**Rothman Consulting Group**

# Jeffery Payne

Jeffery Payne is CEO and founder of Coveros, Inc., a company that helps organizations accelerate the delivery of secure, reliable software using agile methods. Prior to Coveros, he was co-founder of application security company Cigital, where he was CEO for 16 years.

Jeffery is a recognized software expert and popular keynote speaker at both business and technology conferences on a variety of software quality, security, DevOps, and agile topics. He has testified in front of Congress on issues such as digital rights management, software quality, and software research.

Jeffery is also the technical editor of AgileConnection (**www.agileconnection.com**)

# Johanna Rothman

Johanna Rothman, known as the “Pragmatic Manager,” offers frank advice for your tough problems. She helps leaders and teams do reasonable things that work. Equipped with that knowledge, they can then decide how to adapt their product development.

With her trademark practicality and humor, Johanna is the author of 18 books about many aspects of product development. Her most recent books are the Modern Management Made Easy series, and From Chaos to Successful Distributed Agile Teams (with Mark Kilby).

See all her books, blog, and other resources at jrothman.com and createadaptablelife.com.

# Robert Sabourin

Rob Sabourin has more than thirty-nine years of management experience leading teams of software development professionals. A highly-respected member of the software engineering community, Rob has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization.

Rob authored **I am a Bug!**, the popular software testing children's book; works as an adjunct professor of software engineering at McGill University; and serves as the principal consultant (and president/janitor) of AmiBug.Com, Inc.

# Additional Resources

**What Project Managers Need to Know About Testing**
**https://well.tc/proj-mgr**
**by Johanna Rothman**
https://www.jrothman.com/articles/2007/12/what-project-managers-need-to-know-about-testing/

**How to Create a Career Ladder for Real People: Vertical & Lateral Movement**
**https://well.tc/career-ladder**
**by Johanna Rothman**
https://speakerdeck.com/johannarothman/how-to-create-a-career-ladder-for-real-people-vertical-and-lateral-movement

**Out of the Frying Pan into the Fire**
**An Independent Testers Transition to SCRUM**
**by Robert Sabourin**

Out of the Frying Pan into the Fire
An Independent Testers Transition to SCRUM

By: Robert Sabourin

I love SCRUM. SCRUM is a software development framework which helps teams deliver working shippable code in short iterations. I have worked in many lifecycle models from traditional waterfall methods through complex spirals to modern evolutionary approaches. SCRUM shines in turbulent project contexts in which business, organization and technical factors are constantly in flux.

The transition to SCRUM takes many traditional independent system testers out of their comfort zone. SCRUM challenges many common notions about development. If you have been testing as part of an independent team you may be in for quite a surprise moving to SCRUM. In traditional projects test teams find defects which escaped from the previous phases of development. In SCRUM testers are part of a self organized team charged with getting things done! Testers work hand in hand with developers and other team members. Testers are involved in all aspects of development: planning, elaborating stories, testing, debugging and delivering working code.

This article will describe three recent adventures I have had helping independent testers make a successful transition to SCRUM. In each case the transition to SCRUM met with different problems as system testers tried to improve chaotic turbulent projects. The testers took the dive from the frying pan and into the fire. All survived with different scars and some important lessons learned.

A bit about SCRUM

I use SCRUM frameworks to enable delivery of working shippable code in reasonably short periods. Some projects have one week iterations but most clients choose a SPRINT of between two and four weeks.

In SCRUM the project backlog is a live, prioritized, heap of potential requirements including User Stories, Capabilities, Constraints and Product Characteristics or Quality Factors. The product owner is responsible for actively managing the backlog. Potential entries come from customers, marketing ideas, architects and team members. The backlog evolves as development progresses. Backlog entries contain just enough information to prioritize them. Entries require further elaboration to implement. They are detailed as late as possible, often after an implementation decision is made.

The SCRUM team and product owner select high priority project backlog entries to implement in the SPRINT. During the planning meeting developers and testers estimate and scope out the work to do. The result is a selection of items moved from the project backlog into the SPRINT backlog. During the iteration the team focuses on fulfilling the SPRINT backlog.

During planning testers and developers determine how to code and test each story. The team begins the SPRINT with a strong idea of how it will end. To paraphrase Steve Covey “Begin with the End in Mind – First things First”.

The dynamic of development is facilitated by the SCRUM master. The SCRUM Master ensures the team is in sync. The SCRUM master represents the team to external stakeholders. Daily stand up meetings facilitate communications and keep team members aware of progress. The SCRUM master will block distractions and help keep the team remains on track. The SCRUM master facilitates and supports the team’s self-organization.

Testers work hand in hand with developers throughout the SPRINT. Testers prepare test scenarios, test data and work with builds prepared on the fly during the iteration. Developers are often challenged to test and testers may also be asked to develop code as part of the SPRINT. Traditional roles are set aside as required to adapt to the work at hand.

The SPRINT ends with a demonstration of the working code which has passed all tests set out during the planning meeting. Any changes, clarifications or refocusing have been done with full involvement of the product owner to ensure no surprises at the end. A SPRINT retrospective is held by all team members to identify changes to improve subsequent iterations.

Role of Testing in SCRUM

I see the traditional role of testing as falling into two broad camps gatekeepers and information providers.

- Gatekeepers act as the guardian of quality and blocking the release of weak, ineffective or dangerous products.

- Information Providers offer massive objective information about the state of the project under development. What works? What does not work? What bugs must be resolved?

In SCRUM some team members may be primarily testers, some may be primarily developers. Some may take other roles. But all team members may be called upon to test or to support testing during the SPRINT.

The tester works with the developer. The tester is both a consultant and active participant. Before the code is written the tester works with the developer to decide what it means to confirm the code has been implemented correctly. Elaboration of story tests before the coding starts is a very important blend of design and requirement analysis. The tester is helping the developer decide what it means to fulfill the Story.

Basis of Testing in SCRUM

In system testing testers often base testware on many sources of information. Typical sources include requirements, designs, usage scenarios, code, support information and fault models.

In a SCRUM projects the basis for testing is very different. Strategies to assess correctness, must be determined during planning. The amount of documentation available is minimal. The tester must use their creativity and judgement to come up with effective ways to assess correctness. Side by side testing may be used to identify inconsistencies and differences. Subject matter experts and customers can help assess correctness. Tester can use a series of heuristics to guide validation. I have seen many cases in which the tester points out an inconsistency to the developer and then work to understand and resolve the concern. The developer can inspect and modify code while the tester varies conditions to build a good understanding of the problem at hand.

Results of Testing

Traditional testing focuses a lot on documentation. Test documentation is a rich deliverable which can be used to demonstrate in detail what parts of the application were exercised, what problems were identified and what other findings were observed and reported.

In SCRUM projects I see a minimum of test documentation. When a tester finds a bug he or she works directly with the developer to resolve the problem on the spot. No bug list is used. The only bugs documented are those which are not corrected on purpose.

Care is taken to ensure that test documentation required to conform to regulatory requirements are kept. For example in a medical project I worked on a detailed log of how drug data was validated was required to conform to relevant domain regulations.

The Work of Testing

I encourage SCRUM testers to hone their skills in exploratory testing. Testers need to quickly assess the stability of the system and then explore changes or emergent behaviours. Exploration involves navigating

though the new features concurrently designing and executing tests as the tester learns about the implementation.

Testers can explore the typical, alternate and error paths associated with each story being implemented. The typical path of buying a book on Amazon.com involves buying the book based on the title. An alternate path would be buying the book based on author name or the ISBN number. An error path could include attempting to buy a book which does not exist, is out of inventory or with an invalid credit card.

Testers should also focus on exploring "what if" questions about the implementation. What if there is not enough disk space? What if a process runs out of system resources? What if the input is invalid? Exploring failure modes can expose weakness in design.

I encourage testers to use automation, generally controlling the application via a well defined API, using scripted languages such as Perl or Ruby. I avoid traditional GUI based test automation tools since updating scripts when GUIs change takes a lot of time.

Some Case Studies

I help fix broken development projects. Several companies have consulted me recently regarding problems in their implementation of SCRUM. Task analysis of these stories highlight insights into avoiding problems when migrating to SCRUM from traditional lifecycle models.

The following three stories are true. Only the names have been changed to protect the innocent. These are projects in which system testers stepped out of the frying pan and into the fire.

The Case of the Weak Pilot

TBU, a major data processing corporation, had decided to implement SCRUM in an effort to improve their failing lifecycle model. Previously projects were implemented with a waterfall approach. TBU has being experiencing turbulence in product requirements and new market pressures from small niche competitors who were able to come up with competitive solutions quickly. Rework due to changing requirements was getting out of hand. The heavy delivery cycle although rigorous was not fast enough to compete.

TBU has a small Software Engineering group dedicated to refining the lifecycle model and very excited about possibilities of implementing SCRUM. SCRUM indeed seems ideal for TBU. Short incremental delivery cycles with working shippable code would be an important part of meeting the new market pressures.

TBU runs several dozens of concurrent development projects. The organization has a rich history in delivering high quality solutions and has come to depend on it's independent testing team. Indeed the TBU system testing team has saved the day many times by finding important showstopper bugs before release. The TBU system testing team has considerable political power and has a tendency to block initiatives which they feel will risk quality of product delivery.

After piloting SCRUM for over six months with a series of three week iterations several problems occurred. TBU called me in to perform a task analysis and recommend changes to help put the pilot project back on track. I looked under the hood and here is what I found.

TBU Selects a Pilot

TBU was very careful in selecting a pilot project to experiment with SCRUM. TBU was inherently conservative. They chose a project which would have minimal impact on the business if it failed. A failed pilot project would, in the worst case, involve absolutely no lost business.

TBU Kicks off SCRUM

The developers were given basic training on SCRUM. One of the developers was assigned the role of SCRUM Master over and above his normal day by day development activities.

Testing Discovers they are doing SCRUM

After the first SCRUM iteration started the TBU testing team was informed of how SCRUM would work. Testers were not aware of the process. There was a lot of confusion as to what was expected of testers in the iteration.

Testing Team Location

The testing team continued to work in a testing lab far away from the work area of the developers. Testers could not easily meet or interact with developers.

Double Entry of Bugs

This was an experimental Pilot implementation of SCRUM. The director of testing at TBU wanted to ensure that product bug or test data was not lost at the end of the project. The testers were instructed to enter all bugs found in both the TBU traditional testing bug tracking as well as in the new SCRUM tools. Testers managed the double entry of all bug and test data as well as redundant documentation of all test cases run.

Product Owner Absent

This was a very low priority business project for TBU. The product owner was involved with many other responsibilities. At the onset of the SCRUM implementation the product owner actively participated in the SCRUM planning meetings but was not available to the team whenever prioritization or clarification was needed.

Team Interrupted by Support Issues

The development team was constantly interrupted by support issues often related to basic system operation. Developers provided second line support but were basically transferred calls directly from the help desk continuously. The SCRUM Master was seemingly unaware of this problem which chewed by a lot of the teams development capacity. It was treated by developers as a necessary evil and a part of life.

SCRUM Master was also a Developer

The SCRUM Master was a developer on the team. The SCRUM Master took on important development activities as well as coordinating the team, representing the team to stakeholders and escalating issues to management. Often the work of the SCRUM Master was conflicted. Should he spend the time writing critical code or helping unblock another team member. The SCRUM Master made sure the mechanics of SCRUM worked fine but he neglected ensure the dynamics of the team were fluid and effective. Escalating a problem to management is not the same as solving it.

Tester got Builds too Late

Testers did not get builds from developers until very late into the SPRINT. Developers delayed giving code to testers effectively implementing a microscopic waterfall model within the SPRINT. Developers did not coordinate work with testers.

Test Automation

Very little effort was spent by testers automating tests. Very little effort was spent by developers building a framework to support automation. No Application Program Interface (API) was established to support

automation by developers for their unit testing or by testers to help exercise the user stories. Testability issues were not found in the project backlog and were never prioritized as part of an iteration.

Over reliance on Hardening Iterations

In some SCRUM projects an iteration will be dedicated to hardening the codebase. A Hardening iteration focuses on bug fixing. Hardening iterations could be used to explore how applications work on different platforms, in different development contexts or when different co-resident third party applications are running at the same time. TBU used hardening iterations as a placebo for traditional system testing.

The Remedy

As a result of my detailed task analysis I suggested that TBU change a few things in order to implement SCRUM effectively.

1. Make sure SCRUM Master is not on critical path of project.
2. Eliminate redundant bug tracking system.
3. Have testers work directly with developers and in close proximity.
4. Do everything possible to ensure active product ownership. Have the role seconded if the product owner is distracted by other responsibilities
5. *Choose a more important project for next pilot*
6. Allocate time to support issues before planning
7. Have testers engaged in the planning meeting
8. Ensure testability issues are in the project backlog. Trust me they will bubble up in priority quickly.

What TBU is doing now?

TBU has wisely scraped the initial SCRUM pilot. It was deemed better to try again fresh. A group of three pilot projects are underway in a much more important product area with very active product ownership. There is less redundant work but the corporate inertia of the traditional testing role is hovering like vultures waiting for the next weakness to be exposed.

The Case of the Distracting Documentation

BoxedIn is a major leader in the domain of information management. BoxedIn has a long track record of leadership in their niche. BoxedIn has a rich tradition of corporate responsiveness. When a customer has an urgent problem or highly specialized request BoxedIn has always prided themselves in the timely delivery of a high quality solution. BoxedIn are the “Ghostbusters” of “Information”!

BoxedIn had trouble resuming projects interrupted by urgent opportunities. BoxedIn often shifts all developers and testers to an emergency leaving “normal” ongoing projects in a lifeless limbo.

SCRUM is a development framework that could be quite effective for BoxedIn. During the iteration a team would be dedicated to completing the SPRINT without interruption. New opportunities could become high priority requirements for the next iteration thus eliminating the need to constantly cancel, refocus and reprioritize corporate projects.

Before SCRUM, BoxedIn had an independent test team which relied on detailed manual scripted testing. The team was composed of subject matter experts with a spattering of “professional” testers. The testing team dedicated a lot of effort to ensuring several notions of coverage were maintained and documented. Anytime a support person asked “Have you tested this feature” the system test team could respond with an answer quoting the build, tester, date, test description and even the actual test data used. The system testing team kept track of a lot of detailed data.

The development team was torn apart and reassembled in response to corporate emergencies. Developers were heroic knights in shining armour. When a problem came up with put on armour and mounted beautiful white horses. They rode into battle. They made whatever tradeoffs and compromises they could in order to save the day. They proudly rode home victorious, into the sunset, with yet another new configuration to maintain and support for years to come.

Development had a rich tradition of documenting and validating requirements, functional specifications and designs before diving into a solution. Each developer had his own special style of unit testing and the developers did tons of integration testing before builds were released to the independent system testing team.

Product Managers were customers advocates. They worked in the same team as the system testers. Product managers gained consensus between many project stakeholders in order to prioritize requirements.

BoxedIn Kicks off SCRUM

An urgent critical business project was selected as a SCRUM pilot implementation. Team members were not trained. Roles were not clearly defined.

Storyboards replaced traditional requirements for the new project. Product managers were thrust upon the role of product owners. Development managers were made into SCRUM masters and developers and testers attempted to deal with the new reality.

Testing Team Uncomfortable

At first the testing team was completely lost. Where was the test documentation based on formal requirement or design documentation? Testing was to be based on running code and a storyboard. The team was able to use storyboards and running code to identify many important problems but they were never sure of how much testing was actually done. They did not have a notion of test coverage.

Discovering Exploratory Testing

Testers were able to discover unexpected emergent behaviours and identified many critical bugs as they learned about the application. The team was able to find many dozen critical bugs without having to document line by line in advance each test case or procedure. They felt something was wrong and were afraid that they would not be able to answer customer support if queried about whether a certain feature had been tested. The testing team was out of their comfort zone. In order to compensate they spent many hours working overtime trying to create documentation to match the testing done.

Product Management Passive Not Active

The role of product management did not evolve into the role of product ownership. BoxedIn Product Managers wanted to gain consensus from stakeholders before prioritizing requirements or making product decisions. Traditional product management documents were not needed in the new process.

The system architect, development lead, product manager and SCRUM Master maintained separate requirement lists. All were conflicting.

The Remedy

As a result of my detailed task analysis I suggested that BoxedIn change a few things in order to implement SCRUM effectively.

1 Train team in SCRUM, roles, responsibility, framework
2 Implement active product management
3 Consolidate backlog, one for project and one for current SPRINT

4 Involve testers in SPRINT planning sessions
5 Coaching testers and developers on Exploratory Testing
6 Developers should consistently use unit test framework
7 Use "stories" for requirements and testing
8 Ensure team is committed during iteration

What BoxedIn is doing now?

BoxedIn has effectively implemented SCRUM. Testers are no longer focused on documenting the testing effort but rather they are focused on implementing exploratory testing based on storyboards and failure modes. They use automation tools supported by the development team. It took three iterations to get the team in sync. (one month per iteration) The testers had access to expert coaching and some on site training. All key team members received SCRUM training.

BoxedIn management is now using SCRUM on more projects. They are delighted with the fact that projects can progress and deal with urgent changes without having to scrap disrupted project work.

The Case of the Graceful Goose

TimeSoft is a world leader in human resource scheduling and management software. TimeSoft management has established an important mission to dramatically improve time to market without risking product quality.

TimeSoft was often caught following the “tyranny of the urgent” projects and opportunities leaving incomplete or inconsistently documented software in it’s wake. Projects were constantly interrupted and the company was at risk of loosing market share to smaller niche players.

TimeSoft was forcefully merged with another related software company. This merger raised management awareness of the software project problems and led to some dramatic changes.

Some members of the new merged TimeSoft management team had previous successes with SCRUM frameworks in other companies. SCRUM was a natural fit for TimeSoft.

TimeSoft Kicks off SCRUM

TimeSoft management first identified internal champions to support the move to SCRUM. They trained the internal organization leaders including all product owners, development leads and test leads. Once this was done developers and testers were offered a blend of in house custom developed training and external public training in SCRUM. The in house training was designed to focus developers and testers on the differences between their traditional roles and what would be expected of them in the SCRUM implementation.

With one Machiavellian fell swoop TimeSoft implemented SCRUM in all business units concurrently. There were no pilot projects. It was a complete commitment to change that was supported by all levels of management. Business units had no choice in the matter.

TimeSoft adopted a philosophy of creating “barely adequate” documentation for all requirements, designs and project tests. After each SCRUM iteration the team decided if they required more, less or different levels of detail in project documentation.

TimeSoft Pain

The transition to SCRUM was painful and abrupt. It was very difficult for teams to deliver working shippable code for the first few iterations. They did a lot of learning and a lot of self organizing to find the acid mix of development testing and documentation required. SPRINT durations were varied and teams

were established to implement “integration stories” so that multiple small SCRUM teams could work in parallel from a common project backlog.

During the first few SPRINTS testers relied too much on manual testing. They also started testing too late into the iterations.

The Remedy

1 Test Automation
2 Eliminate bug report
3 Learn from other teams
4 Institutionalize learning from mistakes

Test automation was made the highest priority backlog item. The ability to control and observe applications using automated tools enabled automated regression testing. The effect was to have a few iterations which do not add new external features but added testing hooks and frameworks ready for action. This lead to a dramatic acceleration in development and increased confidence in code delivered at the end of each iteration.

What TimeSoft is doing now?

TimeSoft continues to learn and adapt. The testing team is learning new ways to apply visual models and systematically implement exploratory testing in each iteration. The team has a wonderful dynamic. Developers and testers collaborate in all aspects of planning, coding and testing in each SPRINT. In TimeSoft learning and adapting are critical success factors.

Some Common Threads

Testing should be an active role throughout the sprint.

Testing priorities should be in the project backlog and can include frameworks, hooks and test data. It is also a good idea to include stress testing experiments in the backlog.

Iteration planning should decide and prioritize the focus of testing. We should estimate testing effort based on the work to do. Planning should help us decide what we will test and what we will not test as part of the iteration.

During the SPRINT testers work directly with developers. Testers can coach developers in building effective unit and regression tests. Testers can collaborate with developers as code is developed. Testers can communicate with developers to describe, isolate and fix bugs immediately.

Testers should learn to practise exploratory testing. Concurrently design and implement tests as you learn about the application.

Testers should learn to minimize paper work.

Make sure we learn and adapt as a result of every iteration. SCRUM teams are self organizing and very effective at dealing with turbulence and reacting to change. In order to avoid corporate inertia it is a good idea to continuously learn from your mistakes and to also capture excellence.

Moving from traditional models of testing to SCRUM can be exciting. If we can overcome our fears and hesitations SCRUM can be a fun and productive way to enable the development of solid shippable “test inspired” code. Remember it’s all about people … and the occasional bug!

About the author

Robert Sabourin, P. Eng., has more than twenty-five years of management experience leading teams of software development professionals. A well-respected member of the software engineering community, Robert has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization. Robert is the author of I am a Bug!, the popular software testing children's book; an adjunct professor of software engineering at McGill University; and the principle consultant (and president/janitor) of AmiBug.Com, Inc. Contact Robert at rsabourin@amibug.com.

# QUESTIONS?

**Jeffery Payne**
**@jefferyepayne**
**Coveros, Inc.**

**Robert Sabourin**
**@robertasabourin**
**Consultant, Professor**

**Johanna Rothman**
**@johannarothman**
**Rothman Consulting Group**

# Coveros Approach to Agile Testing

## A Pragmatic Approach for Improving Agile Testing Practices

### INTRODUCTION

While many companies are attracted to the concept of agile, they fail to embrace a true agile methodology. When the whole process is agile, organizations are able to iteratively release quality software at a much more rapid pace than traditional waterfall methods. Unfortunately, oftentimes organizations push testing to the end of their agile development cycles, meaning that their cycles are now effectively just smaller waterfall cycles. In these situations, software is blindly handed to the testers for verification at the end of a sprint without the organized, logical direction of a test plan or even a detailed understanding of feature requirements. Testers are left to complete work on weekends or outside of normal work hours, all at once, at the end of the sprint. This scenario allows no opportunity to fix bugs within the same sprint in which they are detected. We hear stories like this all the time from testers - experiences which negatively impact quality and prevent the benefits of agile from being realized. *To fully achieve the benefits of agile development, appropriate testing methods are needed.*

**The focus of agile testing is on having the entire team participate in software testing activities throughout the development cycle.** The role of testers in agile software development encompasses working with product owners, developers, and end users or end user representatives. Product owners and testers collaborate to define acceptance criteria and the definition of done. Agile software testers actively work with developers to clarify the intent of feature requirements, and to ensure their testability and completeness. Involving testers in the early stages of planning allows the organization to accurately predict risks as well as the amount of time needed to successfully complete a feature. Earlier awareness of these details can prevent future unexpected costs or delays and lead to the creation of a higher-quality product. Finally, testing is defined as a part of every user story, and the story is not considered done until all story-level tests (unit, smoke, acceptance, functional, and non-function) have been written and successfully executed.

### COVEROS GUIDELINES FOR AGILE TESTING

Coveros focuses on empowering testers to work on stories in conjunction with developers and non-technical stakeholders. A key point of agile testing is to understand how to integrate software testing early in the development and planning process. Testers are included in all sprint planning and team retrospectives as a proxy for the user, they help the team to properly identify risks, and work closely with development to provide a short feedback loop. This early integration ensures well-planned user stories that address stakeholder needs, thereby increasing quality and significantly reducing the risk of unplanned work.

Coveros focuses the following activities when coaching and implementing tests to improve agile testing practices:

***Shifting testing activities left***—Aligning the work of developers and software testers so that testing activities begin in the planning stages of the development process.

***Integrating testing activities into a definition of done***—Coaching clients who are struggling with identifying the key criteria that comprise a useful definition of done including properly-scoped feature testing.

***Having the right integration and unit tests for applications***—Focusing on creating tests that are applicable for the system and reflect the team's definition of done.

***Integrating tests into a CI/CD pipeline***—Automating continuous testing helps to shorten the feedback loop by providing immediate feedback about code quality based on test results or other metrics as defined in the application's quality thresholds.

***The ultimate focus is not on processes, but on delivering high-quality software at the end of every sprint.***

Here are some important things that testers can do to promote agile testing and allow the team to realize the benefits of agile:

- Get involved in initial development, design, planning, and scoping
- Assess the testability of features being considered for development
- Help developers design effective unit tests
- Design and execute test cases for all aspects of testing
- Automate test cases (when appropriate) to allow for more efficient testing
- Ask questions that help evaluate user stories for appropriate size and scope

### HOW COVEROS DRIVES AGILE TESTING

Our experienced team will help your organization to adopt agile testing principles, standards, and best practices. We prefer to work alongside existing teams to design, develop, and implement testing frameworks. These extensible, reusable frameworks allow other team

members to work with the tests by either creating or modifying as the software project develops.

Coveros practitioners work one-on-one with your development team to improve the quality of your unit tests. Developers will be introduced to concepts such as **test-driven development (TDD)**, which ensures that the development of every new feature begins with writing tests for that feature. We can also help non-technical stakeholders to participate in specifying how an application should behave functionally by introducing **behavior-driven development (BDD)** methodologies and tools. BDD uses a simple, 'Given, When, Then' format to allow all members of the team to collaborate and agree upon feature functionality. To validate the features, tests will then be written using those same steps, which document the agreed-upon behaviors. In an agile software testing environment, testers are able to begin their work at the same time as developers since the user stories are appropriately planned out before being added to the sprint.

**Successful agile testing embraces automation as a long-term investment.** As an application grows, so will the regression test suite, which increases the amount of time it takes to verify releases. Automating part of your regression tests and integrating them into your CI/CD pipeline allows testers to focus on validating new features being developed in the sprint. The team's initial investment in writing these tests will yield time savings in the future. While we can help organizations at any point along the test automation maturity spectrum, we have a proven process for taking an organization with poor or nonexistent automation and starting them on the path to reaching full agile testing maturity.

## SUMMARY

Implementing agile testing methods results in the entire team sharing responsibility for quality while allowing testers to grow in their roles. Empowering testers to work with product owners and developers to articulate feature requirements during the planning phase provides the entire team with an appropriate definition of done for each story. This reduces the risk of re-work and shifts testing activities towards the very beginning of the development cycle. Testers and developers working in tandem throughout the sprint produces a more unified feature matching the definition of done. Investing in automated testing allows for faster verification of new releases while allowing testers to focus on newer features. Our approach is unique in its belief that every organization is different and benefits from a tailored agile testing process befitting its culture, market, organizational structure, people, and business environment.

## RELEVANT CERTIFICATIONS

Coveros has partnered with the International Consortium for Agile (ICAgile) and International Software Testing Qualifications Board (ISTQB) to provide certifications to agile professionals. Our training program offers specific, proven courses for improving your team's agile testing practice. Coveros provides virtual, public, and on-site training offerings, offering you the flexibility to choose the delivery option that works for you and your team. Whether you need to enhance the skills of 1 or 1,000, our experienced agile testing instructors can help.

In addition to simply improving their role-specific knowledge through our training, many software testers seek to complete an industry recognized certification.

Through our training program, your organizational staff can receive certifications in:
**Fundamentals of Agile (ICP certification)**
**Agile Testing (ICP-TST certification)**
**Agile Test Automation (ICAgile)**
**Fundamentals of Test Automation**

Coveros also offers five conferences a year for software professionals via our TechWell brand. These include the STAR software testing series, widely recognized as North America's premier software testing events and EPIC Experience conference which focuses on agile testing and automation. For more information about our training and conferences, please visit training.coveros.com and techwell.com/software-conferences.

## ABOUT COVEROS

For over a decade, Coveros has helped organizations accelerate their software delivery utilizing the latest DevOps and agile methods. Our full complement of cutting-edge services help organizations assess and improve their software development, testing, DevOps, and application security practices.

Our clients include such leading organizations as UnitedHealth Group, Delta Dental, Department of Homeland Security, Symantec, US Air Force, Fannie Mae, RSA, WorldBank, and Advent.

Coveros is headquartered in Northern Virginia.

To learn more about Coveros and our Agile Testing Services, visit www.coveros.com.

**Connect** with **Coveros**

coveros.com
929.341.0139
linkedin.com/company/coveros
info@coveros.com
twitter.com/coveros
youtube.com/CoverosInc

coveros

# Coveros Eliminates a Software Bottleneck

## by Developing an Automation Strategy for Functional and Security Testing

CASE STUDY

CASE STUDY

# Coveros Eliminates a Software Bottleneck by Developing an Automation Strategy for Functional and Security Testing

## CHALLENGES

- A DevOps pipeline with no automation
- Struggles verifying deployments and features
- Manual testing
- Insufficient security testing

## SOLUTIONS

- Test automation
- Integration with CI/CD pipelines
- Base suite of API and functional tests
- Security testing strategy

Coveros worked with a software consulting company that provided security, enterprise cloud, big data analytics, financial project management, and strategy consulting solutions. One of the main focuses of the company's work was providing many web applications for different government agencies. The success of the company was dependent on customer satisfaction with regard to the functionality of these web applications.

## CHALLENGES

- The company had a DevOps pipeline but no automated tests, leading to unnecessary time spent manually verifying deployments or, even worse, verifying only new features while blindly accepting older features that may have been broken by new development
- Improvements were needed in test process, code coverage, and test data
- Tests were mainly manually executed
- Security tests were performed sparsely and by hand

Without automated testing, many hours were spent regression testing features, cutting into time that could be better spent testing new features. As an application grows, manual testing becomes more and more prone to error, as it becomes almost impossible to regression test all previously implemented features during a sprint. This affects software quality when bugs are introduced by adding new features that may have interactions with older features.

Contributing even more to the testing bottleneck was the fact that there was only one test engineer at the company who performed only manual tests on applications. In an attempt to make up for the lack of test engineers, business analysts without testing expertise were standing in for test engineers to manually test features.

The company needed a means to automate testing for its web applications to support development of new functionality, along with ongoing maintenance and support activities.

## SOLUTION

Coveros provided automation expertise to rapidly assess the current development process, then gave automation strategy recommendations, drafted an automated test proof of concept, and created and implemented test automation to support development and production release activities.

Coveros recommended the Selenified testing framework, an open source testing framework that provides traceable reporting for both web and API testing, wraps and extends Selenium calls to more appropriately handle testing errors, and supports testing over multiple browsers locally or in the cloud (Selenium Grid or SauceLabs) in parallel. Coveros quickly integrated the Selenified testing framework into existing build tool configurations to allow any developer to create and run functional tests on web applications as well as web services.

Selenified™
SELENIUM SIMPLIFIED

OWASP
Open Web Application Security Project

sqlmap®

SecureCI™
INTEGRATED BUILD SERVER

CASE STUDY

# Coveros Eliminates a Software Bottleneck by Developing an Automation Strategy for Functional and Security Testing

coveros

Test engineers at the company were then mentored by Coveros employees, who taught them how to write and maintain automated tests with Selenified. Company employees took ownership of functional testing for an application, creating more than a hundred front-end functional tests.

Next, the automated tests were added to existing CI/CD pipelines and build tool configurations. This gave developers rapid feedback and insight into software quality as they developed new features. The tests also acted as a quality gate, ensuring that all features, new and old, worked as intended before promoting a build to a later environment in the release cycle.

To take full advantage of the CI/CD pipelines already in place, Coveros added security tests. These security tests used OWASP ZAP and sqlmap to actively and passively scan web applications for common vulnerabilities. Including these security tests in the pipeline meant moving them to earlier stages in the development cycle, which allows vulnerabilities to be recognized more quickly, shortening cycle times and reducing the risk of releasing vulnerable software. The new security tests realized vulnerabilities in the software, allowing developers to fix the issues before they could be exploited.

Coveros also developed a SonarQube plugin to report the findings of sqlmap scans and alert developers when they had added code that exposed vulnerabilities.

## TECHNOLOGY SOLUTIONS

**Test Automation**

- Front-end and API testing with Selenified
- Security testing with OWASP ZAP and sqlmap

**Continuous Integration**

- Customized SecureCI (Trac, Jenkins, Git, Tomcat, Apache)
- sqlmap SonarQube plugin

## BUSINESS VALUE

Coveros was able to help the company achieve a number of business benefits, including increased ability to implement system improvements while reducing risk, escaped defects, and test cycle time.

Coveros wrote more than fifty API tests and more than a hundred functional tests across three applications. Having a base suite of API and functional tests allowed the CI system to block code from being added to the project when these tests did not pass. This empowered developers to refactor and improve the codebase, while giving them confidence that existing features remained functional. Including security tests into their CI/CD pipelines also reduced the risk of introducing security vulnerabilities and cut down on expensive rework caused by discovering security vulnerabilities later in the software development lifecycle.


CONNECT WITH COVEROS

coveros.com
info@coveros.com
929.341.0139
twitter.com/coveros
linkedin.com/company/coveros