Behavior Driven Development (BDD) A Case Study in Healthtech

Stefania Bruschi, Le Xiao, Mihir Kavatkar, Gustavo Jimenez-Maggiora

bruschi@usc.edu, xiaole@usc.edu, kavatkar@usc.edu, gustavoj@usc.edu

Abstract

Behavior Driven Development (BDD) is a software development practice that leverages a simple domain-specific language to enhance the effectiveness of testing within the Software Development Life Cycle (SDLC). The biomedical research applications developed at the University of Southern California - Alzheimer's Therapeutic Research Institute (USC ATRI) are subject to multiple regulatory requirements. Compliance with these requirements must be tested formally following current software quality assurance best practices and methods. As part of a comprehensive test battery, BDD plays a critical role to ensure each software release meets internal quality standards and complies with applicable regulatory requirements.

The BDD infrastructure developed at USC ATRI provides an extensive set of functionalities: from testing data integrity and replicating user interactions to efficiently communicating test results, analytics, and dysfunctional operations. This scalable infrastructure can accelerate development, speed up the execution of cross-platform testing across multiple browser-device-operating system combinations, standardize validation, facilitate the redesign of existing User Interfaces (UIs) and enhance the readability of test documentation.

This case study demonstrates our journey of building and incorporating BDD in our day-to-day development workflows using a real-world use case. Also, we describe the infrastructure set-up, the test execution, automation and get a real-world perspective on the strengths and limitations of BDD.

In our view, a key advantage of the BDD methodology is that it seeks to build a robust communication channel between software quality engineers (SQEs) and business analysts (BAs) in which the expected behaviors of the system are described in a business readable and domain-specific language. By breaking down siloed communications between stakeholders, BDD allows SQEs to apply tests based on semantically valid business requirements without the need for translation to a distinct technical schema. BAs, for their part, can focus on ensuring that the cases being tested are meaningful, representative, and can easily be traced back to their requirements, leading to increased engagement in the testing process.

Biography

Stefania Bruschi, MS, MBA, is a Senior Programmer Analyst at USC ATRI with a broad range of computer programming experience and proven track record in team leadership and development.

Le Xiao (MS, MBA) is Programmer Analyst at USC ATRI focusing on full stack development and automation testing.

Mihir Kavatkar, MS is Programmer Analyst at USC ATRI. He is a Full Stack Software Engineer and deep learning enthusiast with experience in delivering modern web applications.

Gustavo Jimenez-Maggiora, MBA, is the Director of Informatics for USC ATRI. He is a recognized expert in the field of clinical research informatics in Alzheimer's Disease and Related Dementias.

1 Introduction

Software testing is a broad term collectively applied to a variety of activities along the software development life cycle and beyond, aimed at different goals (Tuteja and Dubey 2012). Historically, challenges in software development and real-time usage have served as motivation for the advancement of new software testing techniques and the technologies which underpin those techniques with the ultimate goal being able to detect flaws in an application before releasing it to consumers. As its benefits have become better understood, software testing has become a widely adopted and important practice, especially in industries that are highly regulated. Health Technology, or Healthtech, defined as "the application of organized knowledge and skills in the form of devices, medicines, vaccines, procedures, and systems developed to solve a health problem and improve quality of lives" (World Health Organization n.d.) is one such industry.

The Alzheimer's Therapeutic Research Institute (ATRI) at the University of Southern California (USC) is an academic organization leading a consortium of academic institutions dedicated to the acceleration of therapeutic interventions for Alzheimer's disease (AD). It is collectively committed to developing new models of AD to test, characterize biomarkers and develop and minimize variability through enhanced quality control of outcome measures by applying novel analytic methods and enact highly innovative regulatory pathways. USC ATRI is currently conducting 18 clinical trials via a network of clinical trial sites in multiple countries to accelerate the development of effective therapies for AD (Keck School of Medicine of USC n.d.). There are multiple sections and functional areas at USC ATRI that participate in different stages of a clinical trial's life cycle (Figure 1: Organizational Structure of USC ATRI). Among these, the Informatics section takes the lead in building the computing and data infrastructure that supports the scientific and operational aspects of the clinical trials conducted by USC ATRI.

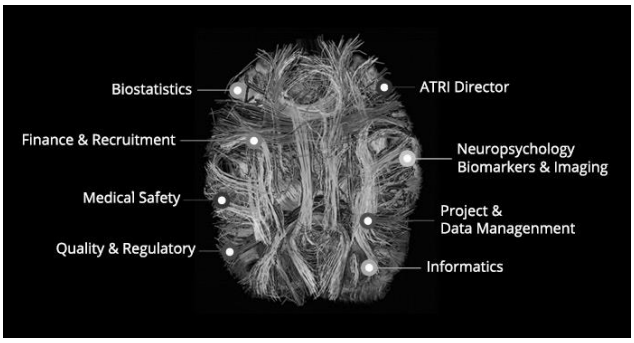

Figure 1: Organizational Structure of USC ATRI

A key system within USC ATRI's computing and data infrastructure developed by the Informatics section is the ATRI Electronic Data Capture (ATRI EDC) system. This web-based system provides clinical research teams with multiple applications to manage their daily scientific and operational workflows and data management activities. This system is an example of a Healthtech application that is subject to regulatory requirements regarding its development and operation, with a strong emphasis on software quality. To address these requirements more efficiently and maintain a feature development roadmap which is responsive to our project goals and user feedback, the Informatics section restructured its development workflow to incorporate the Behavior Driven Development (BDD) methodology. We describe our BDD implementation and discuss the impact this approach has had on our ongoing development of this system in the sections that follow.

2 Objectives

This case study describes the integration of Behavior-Driven Development (BDD) into our Healthtech software development lifecycle. Specifically, we (1) develop a cloud-based platform that supports the process of requirements gathering, testing and reporting; (2) develop CI/CD automation workflows; (3) describe the tools and services that are required to build our BDD platform.

3 Overview

3.1 Electronic Data Capture (EDC) System at USC ATRI

The ATRI EDC system is a software solution designed to support data management activities for a clinical trial, from study start-up through study completion and publication of results. The driving principle behind such a system is to provide research teams with a set of reusable tools for collecting, storing, and sharing clinical research data. The following are the key features of our system; (1) user authentication and role-based security; (2) electronic case report forms (eCRFs); (3) real-time data validation, integrity checks, and other advanced techniques to ensure data quality; (4) Audit capabilities; (5) Document storage and sharing; (6) Data Export functionality for use in statistical analysis workflows; (7) Reporting on the different operational data collected by the system; (8) Secure data storage and backup capabilities. Importantly, as an example of a Healthtech application used to establish new medical treatments, the development and use of this system for research purposes are subject to regulatory requirements.

3.2 Data Quality, Integrity and Regulatory Compliance

The development of new drugs and biologics is a capital-intensive, time-consuming, and highly regulated process. The clinical success rate for a new compound is estimated to be 12%, development costs per new approved drug or biologic are estimated to be $2.6 billion (2013 dollars), and development timelines are estimated to be 96.8 months (DiMasi, Grabowski and Hansen 2016). Regulatory authorities supervise the marketing of new compounds based on evidence generated in clinical trials. Regulators define quality standards for evidence presented in support of new drug applications. Quality standards for the construction and operation of the computer software and systems used to collect and manage clinical trial data are also regulated.

The ATRI EDC system is used to collect data in support of new drug and device marketing applications to the U.S. Food & Drug Administration and international regulatory agencies. When used for this purpose, a computer system is subject to multiple regulations including Title 21 of the Code of Federal Regulations Part 11 (U.S. Food & Drug Administration 2003) and The General Principles of Software Validation - Guidance for Industry and FDA Staff (U.S. Food & Drug Administration 2002) guidelines. The consequences of non-compliance with these standards are high; data collected in a computer system which is not compliant may be rejected by the regulatory agency, potentially putting the approval of the entire new drug application at risk

Title 21 of the Code of Federal Regulations Part 11, or Part 11, describes the record-keeping requirements that a computer system must meet to assure its electronic data and signatures will be eligible for submission to the FDA. These procedural and technical requirements ensure the authenticity, reliability, and integrity of system data and specify the use of written operating procedures, access controls, and immutable audit trails.

The General Principles of Software Validation guideline promotes the use of good software engineering practices and risk management when developing software used in clinical and research applications. The guideline recommends the Software Development Life Cycle (SDLC) approach, which consists of discrete phases such as planning, testing, traceability, and configuration. In this approach, testing, verification, and validation are considered distinct activities. Software verification "provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase". Software validation, on the other hand, provides "evidence that all software requirements have been implemented correctly and completely and are traceable to system requirements". Based on the software's risk profile, this evidence allows us to develop an "acceptable level of confidence" that the software satisfies all requirements and user expectations before release.

We rely heavily on the concepts and methods defined in the aforementioned guidelines and the expert guidance of quality assurance consultants to support the conclusion that the ATRI EDC system is validated and compliant with Part 11. Historically, however, the implementation of these methods was

reliant on a costly, error-prone, manual approach which limited the number of times we could realistically revalidate the system. This approach also suffered from poor reproducibility and scalability. Given the rapidly evolving needs of science and clinical research which drive our system requirements, we decided to find a new approach to validation.

4 BDD Methodology and Strategy

Behavior-Driven Development (BDD) (Lazar, Montogna and Parv 2010) is an agile software development methodology that encourages collaboration between developers, software quality engineers (SQEs), business analysts (BAs) and non-technical stakeholders in a software project. It encourages teams to use conversation and concrete examples to formalize a shared understanding of how the application should behave. BDD combines the techniques and principles of Test-Driven Development (TDD) with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.

Despite these potential benefits, adopting the BBD methodology can be a daunting task for development teams. Incorporating additional effort into use case discovery and testing as well as integrating specialized tools and infrastructure required to support BBD carry risks and costs that may dissuade some teams. Automation, however, can play an important role in recovering these upfront costs and mitigating risks. These considerations are especially important to small organizations such as ours which have limited resources.

To de-risk our team's adoption of BDD, we made several strategic choices: 1) we chose to focus on a specific software project, the ATRI EDC, 2) we chose to focus on a specific goal - validation and regulatory compliance - which BDD was capable of solving effectively, 3) we scoped the project to ensure that automation was a requirement, thus allowing us to recoup our upfront costs rapidly, and 4) we chose to build our BDD infrastructure by using a modular architecture that integrates various cloud-based component services.

As a next step, we enlisted the help of a medical device quality assurance consultant to provide an objective review of our plans to introduce BDD into our existing validation model, which is based on the V-model (Mathur and Shaily 2012), an approach which envisages the SDLC as a series of design, development, and testing stages (Figure 2: V-model). Each design and development stage (e.g. planning, requirements, specifications) is associated with a corresponding testing stage (e.g. testing, verification, validation). Importantly, this model promotes a high degree of traceability between related stages.

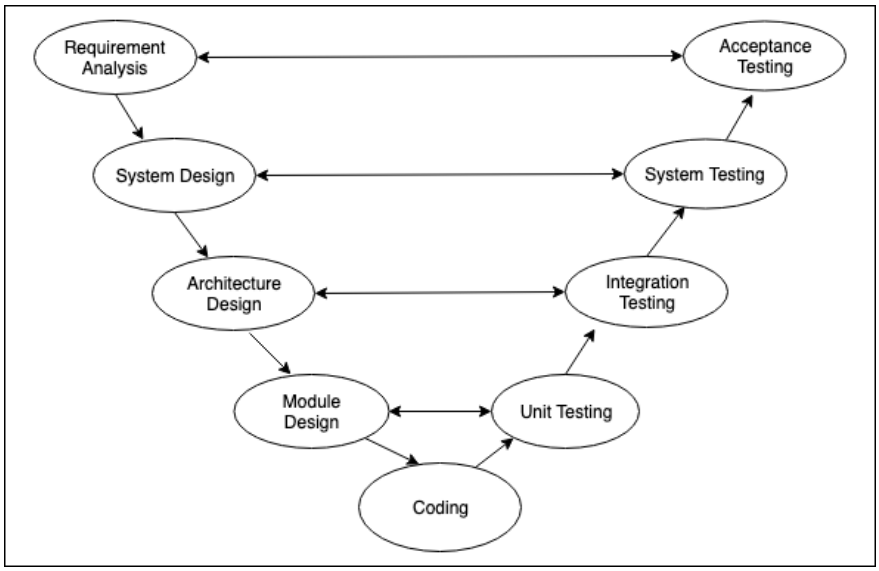

Figure 2: V-model (Mathur and Shaily 2012)

Our BDD-enabled validation exercise was rooted in the fundamental principle of BDD, which posits that user behavior should drive application functionality. User-oriented use cases and test scenarios were developed using Gherkin syntax (cucumber.io n.d.), a highly accepted standard in the software testing community used to describe user behavior utilizing business readable, domain-specific language. These artifacts serve as inputs to our BDD infrastructure, which automates test generation, execution, reporting, and documentation.

Once the initial validation of the ATRI EDC system was completed we continued to work to further incorporate BDD into our development workflow. Ultimately, we established a process where new development starts with a requirements discovery process which seeks to be inclusive of multiple stakeholder perspectives. Requirements are broken down into use cases and test scenarios using Gherkin syntax to ensure the full project team can participate in the review process. The final Gherkin documents are used by our BDD infrastructure to support automated testing, documentation, verification, and validation. As new user requirements emerge, this cycle can be repeated efficiently.

5 Infrastructure Overview

The Behavior Driven Development (BDD) infrastructure enhances the testing framework of a software application and must be integrated into the culture and software development life cycle of an organization. We implemented the BDD infrastructure as a plugin that can be used across multiple projects with minimal training. When developing a BDD infrastructure, it is crucial to consider each of the following aspects: transparency in the organization, scalability of the overall system and processing pipelines, automation in the integration with a software implementation, version control, maintainability, simplicity, compatibility, and cross-browser support. Finally, the BDD infrastructure must automate the creation and distribution of reports and traceability matrices.

Figure 3 provides an overview of BDD infrastructure we developed.

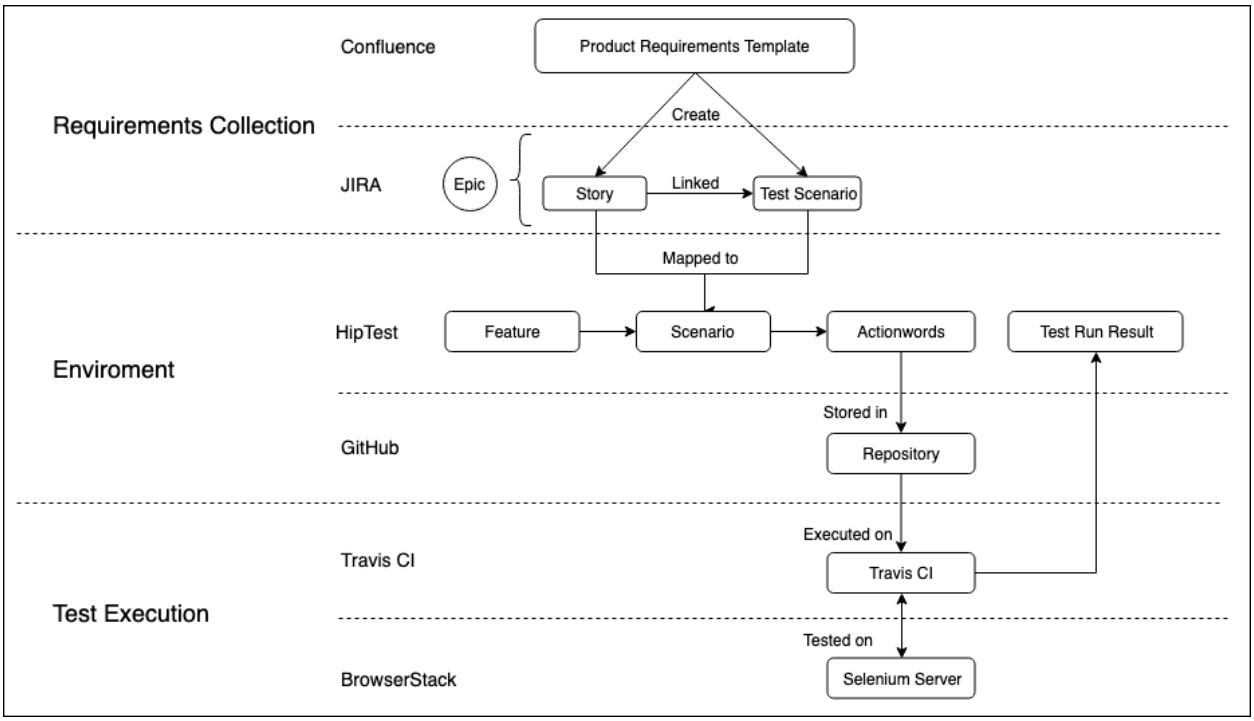

Figure 3: BDD Infrastructure

6 Requirements Collection

The integration of the BDD methodology within the early stages of the SDLC process is critical. Following the V-Model shown in Figure 2, planning BDD tests during the requirement analysis stage is an effective strategy to increase collaboration between stakeholders and to define linkages between requirements and acceptance tests.

To increase transparency and facilitate communication between different cross-functional teams, we use an integrated project workspace made up of Atlassian Confluence, a knowledge base tool, Atlassian Jira, an agile project management tool. Atlassian tools allow teams to initialize linkages between requirements and BDD test scenarios and maintain the association between tasks generated at any stage in the SDLC and requirements and BDD test scenarios. This process minimizes disruption to the coding and testing phases but helps software quality engineers (SQEs) and business analysts (BAs) automatically generate reports that can be distributed to project stakeholders.

6.1 Requirements Hierarchy

Defining a requirements hierarchy is one approach to break down the requirements into understandable and readable parts.

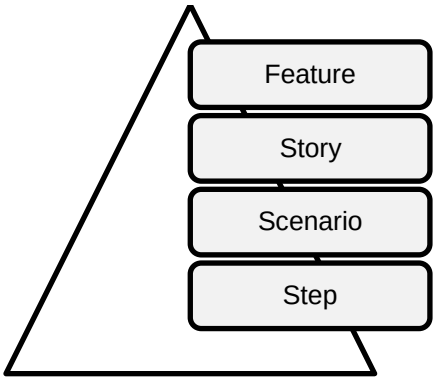

Figure 4: Feature Structure

In Figure 4, we define the following hierarchy:

- **Feature** is a summary of new functionality to be developed and tested.
- **Story** is the end-user use case that needs to be addressed and the expected outcome is interpreted from the software or system requirements.
- **Scenario** is the basic unit of functionality which scopes out the user behaviors. A story could have many scenarios to test different outcomes.
- **Step** in a scenario is a simulation of user behavior which is verified against the expected behavior. There can be many steps to satisfy a test scenario.

A similar convention is followed by HipTest (SmartBear Software n.d.), a collaborative testing platform in the cloud from SmartBear Software that allows the software delivery team to co-design acceptance tests.

7 Environment

7.1 Following standards

Writing test scenarios must follow standards that ensure readability and maintainability. We used Gherkin language, a highly accepted standard in the testing community, to describe the test cases. “Gherkin is designed to be non-technical and human-readable, and collectively describes use cases relating to a software system. The purpose behind Gherkin's syntax is to promote behavior-driven development practices across an entire development team, including business analysts and managers. It seeks to enforce firms, unambiguous requirements starting in the initial phases of requirements definition by business management and in other stages of the development lifecycle”. (cucumber.io n.d.)

7.2 Collaborative Testing Platform

The BDD infrastructure requires a collaborative testing platform allowing the design of the test scenarios, test execution, and test refactoring.

Our selection was based on the following criteria:

- Improve trackability – the testing platform is integrated into our agile project manager tools, Jira, via plugins.
- Automate report (i.e. traceability matrix) – the testing platform provides standard reports and APIs.
- Minimize disruption of SDLC process – the testing platform generates executable code and feature files that can be translated into many programming languages including all the scenarios and action words.

- Automate maintainability across multiple releases – the testing platform supports test refactoring and automated report generation synchronizing test scenarios and keeping feature files up to date.
- Visibility and One stop shop – the testing platform provides a rich reporting framework that aggregates test results and Gherkin-based descriptions.
- Keep the overall system lean – the testing system provides reusable action word across features.
- Security – the testing platform requires encrypted communications between component services.
- Cost-effectiveness and Total Cost of Ownership – the testing platform requires modest subscription costs and minimal or no licensing fees.

Based on the listed selection criteria, HipTest (SmartBear Software n.d.) was selected as the best match for our requirements.

7.3 BDD Plugin

The BDD plugin is the core of the BDD Infrastructure where all the preliminary work of creating stories and scenarios comes together. Our BDD plugin uses the Python-based Django framework. The BDD plugin includes several modules as shown in Table 1. The code structure of the plugin is crucial for its future reusability.

Feature Files	The HipTest command line interface (CLI) helps to build a set of feature files which describe the scenarios and the steps that are required to accomplish them. These feature files are the entry point to the tests (SmartBear Software n.d.).
Step	A step is a user behavior description. In the plugin, it builds a mapping between a step described in a feature file and implementations which are referred to as action words.
Action Word	An action word is a set of actions taken to implement user behaviors. Developers use an action word to define the logic of the test (SmartBear Software n.d.).
Page Object	The page object pattern is used in the context of web testing for abstracting the application's web pages to reduce the coupling between test cases and application under test (Spadaro, et al. 2013).
Fixtures	A fixture is a collection of pre-defined sample data that can be used to support development and testing (Django n.d.).

Table 1 - BDD Plugin module

8 Test Execution

Figure 5 illustrates the environment utilized by our BDD infrastructure to support cross-platform, high performance test execution.

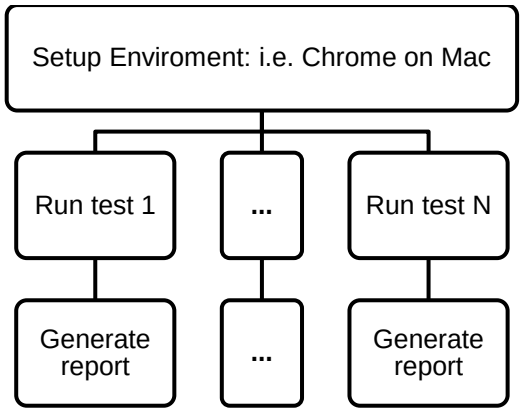

Figure 5: Test Execution

8.1 Parallel Test Running

Monitoring multiple test runs is time-consuming and involves manual work. Automating the parallel execution of tests is a crucial requirement for an effective BDD infrastructure. Travis CI (Travis CI n.d.) is a continuous integration (CI) tool that can be integrated with source code version control systems such as GitHub to streamline test execution at different stages enabling parallel testing runs. Each concurrent job has a makefile which defines the BDD test environment's build directives and dependencies.

8.2 Test Running on a Cross-Browser Platform

For web development, the BDD Infrastructure must enable developers to execute their tests across multiple combinations of operating systems, devices, and browsers. This need has become increasingly important as the smartphone market continues to grow and diversify (Prasad and Mesbah 2011).

To support this requirement, we chose BrowserStack, a cloud-based cross-browser testing solution, that allowed us to manage our costs, provide our developers with reasonable test execution times, and collect video-based troubleshooting and execution reports (Gowthaman and Kaalra 2014).

Furthermore, BrowserStack supports integration with Travis CI and enables automation testing to run through Selenium. After each test execution is complete, the test results are sent to HipTest through integration with Travis CI.

9 Reporting: Requirements Traceability Matrix and Test Results

A critical component of a software or system validation project is the creation of a requirements traceability matrix (RTM) (Wikipedia, the free encyclopedia 2019). In this context, the purpose of the RTM is to ensure that all requirements are covered by the test protocol.

Our BDD infrastructure generates a new RTM report for each software release which supports backward- and forward-traceability between requirements and related tests. Links to detailed cross-browser platform test results are also provided to facilitate quality reviews.

9.1 Current process

The current reporting process relies on Jira, HipTest, and Confluence APIs and the BDD plugin's environment file and test-specific makefiles to build and upload a detailed report of use cases, tests, and test results to Confluence. This information is subsequently used to generate the RTM. These reports are reviewed by the QA team and project stakeholders to confirm requirements and review discrepancies.

9.2 Potential Improvements

The current reporting process displays the RTM as a set of reports that can be viewed in Confluence or exported as a PDF document. This static, table-based approach limits our ability to analyze historical data using statistical or visualization tools. Platforms such as Amazon Web Services (AWS) or Google Cloud Platform (GCP) provide reliable, scalable, and cost-effective cloud computing services that could be used to optimize our reporting process. The use of these services would allow us to store, analyze, and draw inferences from the rich testing data we are collecting. Efforts to implement these improvements are already underway.

10 Discussion/Lessons Learned

The integration of the BDD methodology into our software development life cycle began with a comprehensive review of our processes, practices, and tools. Once complete, this review revealed several gaps which we addressed over 3 months, from September to December 2017. In the following 3 months to March 2018, we collaborated with a medical device quality assurance consultant to retrospectively apply the BDD approach to validate the ATRI EDC system and ensure compliance with 21 CFR Part 11 guidelines. After this, all subsequent development of the ATRI EDC system has been conducted using the BDD approach. This experience has provided us with a better understanding of the advantages and shortcomings of the BDD approach, which we discuss below.

In our experience, a key advantage is that BDD increases communication which results in better collaboration. It helps multiple teams agree on a standard for formalizing the test requirements which are used to automate testing and document generation, elements which are important for system validation. Every step, from requirements gathering to test execution, is traceable and transparent to all stakeholders. Time-consuming tasks, such as documentation and testing, are largely automated by BDD allowing teams the flexibility to focus on other priorities. To ensure continuous compliance before every release, comprehensive testing reports are generated which allow all project stakeholders to review and confirm that defined test scenarios are meaningful and cover product requirements. The automation available using BDD saves time and cost, critical to small organizations such as ours. Finally, significant changes can be deployed with thorough validation and test requirement coverage so that the development process is accelerated.

Despite the above advantages, BDD also has a few shortcomings. First, the Gherkin syntax attempts to make tests human-readable and therefore constrains human language. Because of this, it may not able to capture nuance and intention. Second, test execution follows a top-down approach within a feature file and hence ignores conditional situations. In other words, “if-else” conditions cannot be used to cover alternate use cases. A solution to this limitation is to replicate the test scenario for each conditional use case. Finally, different browsers have their respective web drivers and test scenarios rely on these to perform user behaviors. Some web driver versions may not be compatible with the latest browser and the tests may fail for these browsers. A solution to this problem is to restrict the version of the browser where the automated tests are executed. A limitation to this solution is that the tests are not executed on the latest browser which can result in unexpected results.

11 Conclusions

In the fast-paced environment of medical research and information technology, our development teams historically struggled to keep up with rapidly changing user scenarios and requirements. This served as the initial motivation for moving our software development model from waterfall to agile. The agile development approach has provided our teams with numerous advantages such as short iterations, focus on consumer requirements, and increased developer engagement. Despite these gains, challenges still exist; being able to rapidly add a new feature to a project doesn't necessarily mean that the new feature will meet user requirements or stakeholder expectations. More often than not, this mismatch is due to gaps in communication between end-users, stakeholders, and developers. This is one of the areas where

the process of Behavior Driven Development, a methodology that extends traditional agile approaches, truly shines. By design, BDD strives to keep the whole project team in sync, maintaining one standard language (Gherkin), to describe use cases and test requirements. Just as importantly, BDD facilitates the inclusion of both the end-user defined requirements and the technical perspectives required to achieve them.

Interestingly, traditional BDD implementations consider automation an optional requirement. This case study demonstrates different strategies and services we used to automate BDD. By automating the execution of our BDD test battery, we attained several benefits including tighter collaboration and communication, faster feedback, and continuous compliance with 21 CFR Part 11 guidelines. Adding automation allowed us to create a sustainable BDD-based validation model which is a critical requirement for Healthtech projects.

Finally, the key advantage of including BDD in our software development approach has been the creation of a more inclusive environment where all project stakeholders are engaged in a more frequent and meaningful manner. This ongoing engagement has ultimately reduced the number of misunderstandings and inconsistencies that can occur at the end of the project. BDD has strengthened the interaction between the entire project team leading to significant efficiencies.

12 References

cucumber.io. n.d. *Gherkin Reference.* Accessed 08 23, 2019. https://cucumber.io/docs/gherkin/reference/.

DiMasi, Joseph A, Henry G Grabowski, and Ronald W Hansen. 2016. "Innovation in the pharmaceutical industry: New estimates of R&D costs." *Journal of Health Economics* 20-33.

Django. n.d. *Providing initial data for models.* Accessed 08 23, 2019. https://docs.djangoproject.com/en/2.2/howto/initial-data/.

Gowthaman, Dr. K, and Bhavnesh Kaalra. 2014. "Cross Browser Testing Using Automated Test Tools." *International Journal of Advanced Studies in Computer Science and Engineering* (IAASSE).

Keck School of Medicine of USC. n.d. *About the Alzheimer's Therapeutic Research Institute.* https://keck.usc.edu/atri/about-atri/.

Lazar, Ioan, Simona Montogna, and Bazil Parv. 2010. "Behaviour-Driven Development of Foundational UML Components." *Electronic Notes in Theoretical Computer Science (IEEE).*

Mathur, Sonali, and Malik Shaily. 2012. "Advancements in the V-Model." *International Journal of Computer Applications.*

Prasad, Mukul, and Ali Mesbah. 2011. "Automated Cross-Browser Compatibility Testing." *International Conference on Software Engineering.* ICSE. 561-570.

SmartBear Software. n.d. *Execute the tests and create feature files.* Accessed 08 23, 2019. https://hiptest.com/docs/create-feature-files-for-automated-execution/.

—. n.d. *Hiptest.* https://hiptest.com/.

—. n.d. *HipTest at a glance.* Accessed 07 17, 2019. https://hiptest.com/docs/about-2/.

—. n.d. *HipTest: Use action words.* Accessed 08 23, 2019. https://hiptest.com/docs/use-action-words/.

Spadaro, Cristiano, Filippo Ricca, Diego Clerissi, and Maurizio Leotta. 2013. "Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study." *Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation.* IEEE.

Travis CI. n.d. *Travis CI.* Accessed 08 23, 2019. https://travis-ci.com/.

Tuteja, Maneela, and Gaurav Dubey. 2012. "A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models." *International Journal of Soft Computing and Engineering (IJSCE).*

U.S. Food & Drug Administration. 2002. "General Principles of Software Validation." *www.fda.gov.* January. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-principles-software-validation.

—. 2003. "Part 11, Electronic Records; Electronic Signatures - Scope and Application." *www.fda.gov.* 08. Accessed 08 14, 2019. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/part-11-electronic-records-electronic-signatures-scope-and-application.

World Health Organization. n.d. "Health technology assessment: What is a health technology?" *World Health Organization.* Accessed 08 13, 2019. https://www.who.int/health-technology-assessment/about/healthtechnology/en/.