Avinash Budihal

A Guide for Implementing the Behaviour-Driven Development (BDD) Framework in IT Projects

Metropolia University of Applied Sciences

Master's Degree

Degree Programme in Business Informatics

Master's Thesis

26 May 2025

My master's study in business informatics started in the year 2024 at Metropolia University of Applied Sciences. During this stage, I gained a lot because the lecturers were knowledgeable and experienced. The curriculum gave an extensive understanding of the respective courses. Throughout the entire process of writing the thesis, my supervisor, Zinaida Grabovskaia’s help and support, was of paramount importance.

My time at Metropolia University of Applied Sciences was memorable and provided me many new things to learn professionally and personally. The teaching atmosphere was stimulating, and it set me on the right path for what I aimed to achieve. Being part of Finnish culture for the first time provided me with much to think about and new opportunities to grow personally. My mother has cheered me on and helped me throughout all the things I experienced as I learned and moved to a foreign country.

I would like to thank the experts for sharing their experience, which really helped improve my thesis. Thanks to the expert’s help, I was able to analyses my results along with the findings of others in the industry.

Avinash Budihal
Helsinki, Finland
May 26, 2025

Abstract

Author:	Avinash Budihal
Title:	A Guide for Implementing the Behavior-Driven Development (BDD) Framework in IT Projects
Number of Pages:	81 pages + 1 appendices
Date:	26 May 2024
Degree:	Master of Business Administration
Degree Programme:	Business Informatics
Instructor:	Zinaida Grabovskaia, PhL, Senior Lecturer

The thesis looks into the issues that come up when agile IT sector companies adopting Scrum try to write executable test cases from user stories whose core focuses are behavioral needs. If requirements are unclear to all parties, people involved in the project may misunderstand each other, and delays can happen, leaving more issues inside the software code. Therefore, the purpose of this study is to develop a step-by-step guide for Scrum teams to start using the Behavior-Driven Development (BDD) Test Framework.

By using behavior-driven acceptance criteria, BDD helps people on the scrum team to understand the requirements the same way. Discuss and agree on the rules for the project to let each person understand their duties and the tasks involved.

It also uses information gathered from talking with people who are actually trying out BDD with their Scrum teams. Many points in this paper were drawn from interviews with experts involved in the Scrum team during BDD. Experts looked over the first proposal to help perfect and finalize the document.

The thesis was developed by studying real cases from companies and advice from specialists in the field.

The goal of this thesis is to show organizations how they can improve their test cases, better meet user expectations, promote stronger teamwork between development, testing, and business teams, and supply software that fully fits the requirements of agile.

Contents

List of Figures

List of Tables

1 Introduction

In present days, the software development lifecycle usually takes place using agile methodologies. Scrum team members are the main players of the agile team. The system requirements are brainstormed, planned, developed, tested, and deployed by these team members. The main issue faced by the Scrum teams is related to the understanding of the requirements. Whenever the requirements are delivered by the product owner to the Scrum team, there are high chances of misunderstanding of the requirements by the Scrum team. This kind of communication gap drastically increases the project's overall budget as the Scrum team spends most of the time in understanding and clarifying the requirements. To address these kinds of problems, the best practices, such as Test-Driven Development (TDD), Behavior-Driven Development (BDD), and Acceptance Test-Driven Development (ATDD), have evolved, but still today most of the organizations have not adopted these practices in their projects.

To address this issue, this thesis provides a detailed guide for implementing the BDD in Scrum teams. The guide also provides detailed understanding of all aspects of BDD and its effective adoption to the teams who are willing to adopt it.

1.1 Business Context

In today's world, software development is iterative in nature, which will contribute to many domains such as the financial sector, travel, healthcare, and many more. The software requirements get evolved frequently, and the scrum teams have to understand these changes and do the development accordingly.

Mainly, it is particularly difficult in IT for Scrum teams to turn user stories into test cases that can run on the system. Getting requirements wrong often leads to problems among stakeholders, delays in releasing the product, flaws in the software, and greater project budgets.

This thesis is built from experiences from three different IT projects. This thesis analyzes three real-time projects from three global case companies from the years of 2022 to 2024.

1.2 Business Challenge, Objective and Outcome

A common challenge in many of organizations is a lack of shared understanding of the requirements under development among all the Scrum team members. It is very essential to eliminate or nullify this deviation in requirement understanding by both teams, which are responsible for development and other relevant stakeholders. For projects where business stakeholders have a specific and unique expectation related to IT system development, Behavior-Driven Development (BDD) is considered to be a highly effective approach. According to (Manekar 2024, May 20), BDD fosters collaboration between technical and non-technical teams, ensuring that everyone understands and has a shared common understanding of the requirement.

Behavior-Driven Development (BDD) is a powerful framework that can help to bridge these gaps very effectively. However, implementing BDD requires specialized and expert knowledge and understanding of how to adapt it to a new project or existing project needs. While many companies have started to adopt BDD, its widespread use is still emerging day by day. According to (Jacon 2023, July 10), successful BDD implementation demands a depth and vast understanding of key questions like 'what,' 'where,' 'how,' 'why,' and 'when.' Companies must also carefully assess the feasibility of BDD adoption within their teams and corresponding stakeholders.

For these challenges, today's creative firms rely on a newly developed method named Behavior-Driven Development (BDD). Using this approach increases the success of tests, lowers how far off requirements are and helps the entire team collaborate more smoothly. While many organizations want to use it, not having staff with direct experience or knowledge often stops them from starting.

The Objective of this thesis is *to develop a guide for implementing the Behavior-Driven Development (BDD) Test Framework in IT projects by Scrum teams,* based on the first-hand experience of the stakeholders in three IT projects done for global IT companies.

The Outcome of this thesis is *a guide for implementing the Behavior-Driven Development (BDD) Test Framework for Scrum teams in IT projects.*

Adopting such a practice-based BDD (Behavioral-Driven Development) test framework can increase the impact of the test cases by shaping the expected customer behavior with accuracy. Previous real-life project experience of the thesis researcher and hos

stakeholders shows that this approach fosters a shared understanding of test scenarios among all the scrum team members and helps in the streamlining of the communication and collaboration. By aligning development efforts with required user requirements, BDD can also contribute to the formation of more user-centric and relevant software product solution output.

1.3 Thesis Outline

This thesis aims to improve value creation by software engineering teams in an Agile environment. In theory, the BDD approach can be applied to any industry sector, but this thesis will focus on software development and testing departments, with the goal to develop a guide for implementing the Behavior-Driven Development (BDD) Test Framework in the IT projects in Scrum teams. The scope of the thesis is the creation of the guide for implementing BDD practice in the IT team of Scrum of IT companies.

This section is written in seven sections. Section 1 is the Introduction. Section 2 describes the research approach, data collection and analysis methods used in this thesis. Section 3 focuses on exploring available knowledge and best practice related to implementation of the Behavioral-Driven Development (BDD). First, it is crucial to understand its core principles, advantages, and limitations. Section 3 answers questions like what are the key components of BDD, how to build a BDD team, why use BDD, when BDD best fits, and what are the key advantages and disadvantages of BDD. Next, Section 3 overviews the BDD tools, how BDD tests can be integrated into the cloud pipeline, how the BDD test reports can be generated for analysis purposes. Finally, Section 3 also touches on how to implement the BDD framework in Scrum teams. This section ends with a conceptual framework on BBD implementation in Scrum teams.

To gain real-life insights from the past IT projects of the case company, Section 4 reports on three projects from multinational IT companies that used agile methodology for software development. By detailed analysis of the past projects, the common challenges, successful strategies, and emerging trends in BDD adoption are identified. This analysis is focused on identifying the tools and best practices of successfully adopting the BDD approach into the development process, including its design, implementation, and resource allocation.

Next, drawing on research and industry insights, the practical guide for implementing

BDD in Scrum teams in IT similar projects is created. This guide will suit the specific needs of IT testing organizations, also considering factors such as project complexity, team size, and existing tools in use. By following this guide, Scrim teams in IT projects can successfully adopt the BDD into their projects and utilize its full potential.

Thus, the scope of this thesis is limited to developing a framework guide that can be used in the implementation of BDD practices within Agile Scrum teams for software development. It focuses on how to bridge the communication gaps that always exist in requirements understanding by team members in the development and testing of software products. It would point to a number of critical components in real-time coordination and iterative product delivery concerning software development and testing teams in Agile Scrum teams, particularly for software applications of complex IT projects.

2 Method and Material

This section describes the research approach, research design, and the analysis methods used for this thesis.

2.1 Research Approach

There are multiple research approaches that can be used. Some of the basic research families are discussed below.

Firstly, research methods can be classified based on purpose and methodology. Basic research and applied research are purpose-based. Qualitative, quantitative, and mixed methods are methodology-based research methods. (Creswell 2018)

Secondly, applied research has various methods, such as interviews, surveys, and document analysis. (Creswell 2018)

The third one is the applied action research method. Collaboration and tackling the problem step by step are emphasized in this research. To adopt this type of research, one should work in cycles. First find out the issue, then introduce the change and monitor the progress. (Kananen 2013)

By using action research, the study creates a reference guide on using BDD in IT scrum teams. The timely collaboration was conducted with relevant stakeholders and industry experts to collect the data. The real-time project observations are considered.

This thesis first reviews the existing literature around this area, and second, a detailed analysis of the implementation of BDD in three case organizations. This phase is mainly captured in the current state analysis. These all combine to give a detailed picture of the BDD implementation in agile scrum teams.

Each case company project explains the project in perspective of the BDD need, how it is implemented, who participated, how it is achieved, and then its benefits and limitations. The stakeholders who are interviewed hold different roles in a team while implementing BDD. At the end, the initial proposal was created by keeping in mind all the above details.

In this research, the data is collected from stakeholders, technical diagrams, articles, figures, and data.

Interviews with relevant stakeholders are a very important part of this research. These interviews made a platform to collect the experience, knowledge, processes, and practices from the stakeholders. Observations allow the researchers to witness the interactions, processes, and practices that were experienced by conducting interviews and surveys. This information is then documented, which produces valuable policies, practices, and guides for understanding the current context.

A case study provides an in-depth understanding of the particular practice, process, and series of events. These case studies provide the detailed understanding of the case under research.

The strategy used in this thesis is applied action research. The main characteristic of this approach is to address the specific real-world context. In the current thesis, the context is to implement the BDD for the scrum team of IT.

The main aim of this applied action research is to enable the participants who are involved in the real-time context to learn, train, adopt, and effectively implement the practice.

2.2 Research Design

Figure 1 illustrates the research design for this study.

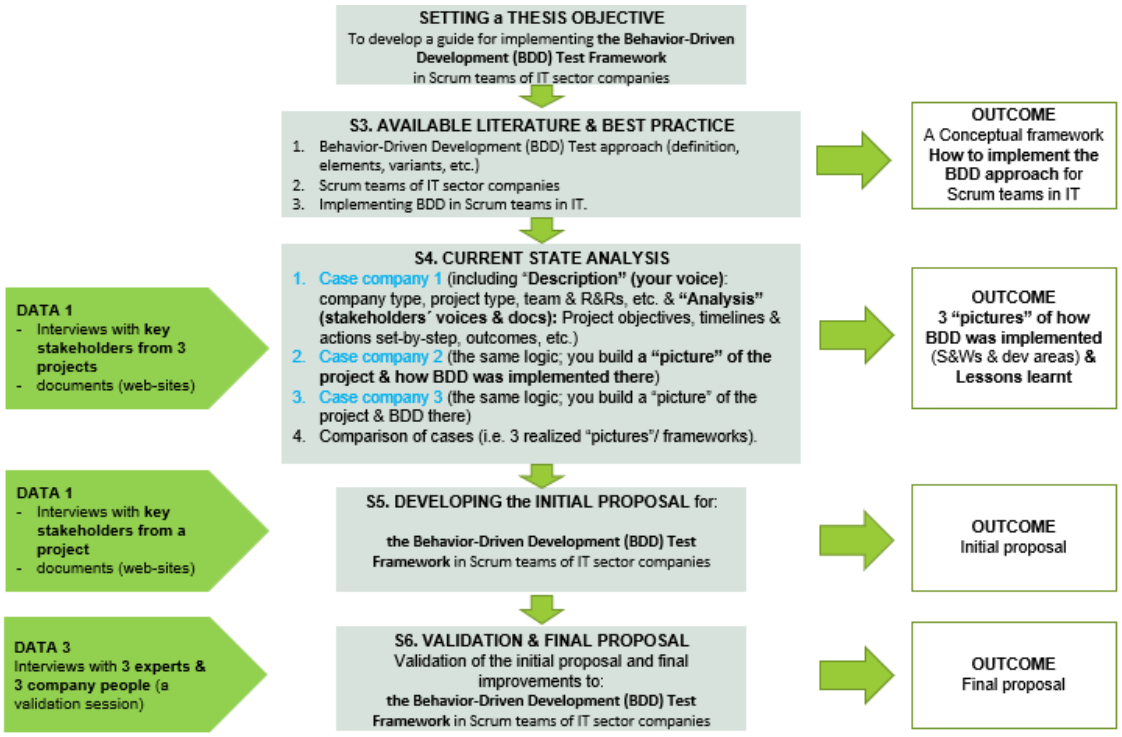

Figure 1. Research design of this thesis.

This thesis began by setting the objective for developing a guide how to implement the Behavior-Driven Development test framework in Scrum teams in IT Projects.

Second, a review of literature and best practice was performed. This entailed the review of BDD testing approaches taken from different articles and journals. The studies were industry-based, written by experts with experience in the relevant skills and domains. The literature and best practice review ends with the conceptual framework for implementing the Behavior-Driven Development test framework in Scrum teams in IT projects.

Third, the study continues to the current state analysis carried out in three organizations employing comparable approaches within their Scrum teams. The data collection was carried out through detailed discussions with the relevant stakeholders and analyses of internal documents within each Scrum team. The analysis focused on business domain, type of projects, objectives, timelines, development roadmaps, outcomes, limitations,

and weaknesses of each projects and the practices of implementing BDD in them. Additionally, for benchmarking purposes, a meeting with one of the stakeholders was also held with a company based in Finland for an outsider organizational input on their test framework approach to comparatively analyze BDD against other approaches.

Fourth, the initial proposal for the guide is developed, and fifth, it is validated through discussions with experts working in this domain. The final proposal for the guide is built based on refinement comments and validation feedback. The final proposal puts this document together, showing the final guidance on implementing BDD test framework.

2.3 Data Collection and Analysis

This thesis used three rounds of data collection. Their details are shown in Table 1 below.

Table 1. Details of Data collections 1-3 used in this study.

	Participants / role	**Data type**	**Topic, description**	**Date, length**	**Documented as**
	Data 1, for the Current state analysis (Section 3 or 4)				
1	Respondent 1: Technical architect	Online face to face meeting	The case company develops the visitor management application for schools in the USA.	Dec 2024, 85 min	Field notes
2	Respondent 2: Test manager	Online face to face meeting	The case company develops the health care applications for hospitals.	Jan 2025, 95 min	Field notes
3	Respondent 3: Automation Test Engineer	Online face to face meeting	The case company develops the application for data protection and security.	Feb 2025, 90 min	Field notes
	Data 2, for Proposal building (Section 5)				
8	Participants 1: Software Development Engineer in Test	Face to face meeting	Proposal building	Feb 2025, 90 min	Field notes
	Data 3, from Validation (Section 6)				
9	Respondent 4: Technical architect	Online face to face meeting	Validation, evaluation of the Proposal	April 2025, 90 min	Field notes
10	Respondent 5: Senior development engineer	Online face to face meeting	Validation, evaluation of the Proposal	April 2025, 90 min	Field notes

11	Respondent 6: Test automation engineer	Online face to face meeting	Validation, evaluation of the Proposal	April 2025, 90 min	Field notes

As demonstrated in Table 1, the data for this thesis, was obtained through interviews with the stakeholders in three past projects that included the implementation of the BDD framework within agile methodologies IT projects. The data collection occurred in multiple rounds.

In the first round, discussions were conducted with three stakeholders from three IT Scrum teams from three IT projects. The thesis research was closely familiar with all three projects and their stakeholders, as he used to previously work on these projects. The three companies specialize, respectively, in developing visitor management tools, health care sector solutions, and SaaS product development. The project stakeholders were one technical architect, one technical manager, and one automation test engineer from each of the projects. Detailed interviews were conducted, for which interview were recorded and field notes taken.

In the second data collection round, the textual data was analyzed using thematic analysis, and the internal architectural diagrams were analyzed, and the data was collected, which contains the details of software tools, techniques, processes, and diagrams.

Table 2. Internal documents used in the current state analysis.

	Name of the document	**Number of pages/other content**	**Description**
A	Roles involved in the BDD implementation of Project 1	1 diagram	Illustrate the roles involved in project 1
B	Phases of Project 1 in relation to BDD implementation	1 diagram	Illustrate the phases of project 1.
C	Project 1: Task-Phase Mapping	1 diagram	Illustrate the task-phase mapping of project 1.
D	Cloud application development and test process using Behaviour-Driven Development at Case company 1	1 diagram	Illustrate the cloud application development and test process at case company 1.

E	Roles involved in the BDD framework of Project 2	1 diagram	Illustrate the roles involved in the BDD framework of project2.
F	Phases of Project 2 in relation to BDD implementation	1 diagram	Illustrate the phases of project 2 in relation to BDD implementation.
G	Project 2: Task-Phase Mapping	1 diagram	Illustrate the task-phase mapping of project 2.
H	Continuous integration and automated testing workflow at Case company 2	1 diagram	Illustrate the continuous integration and automated testing workflow at Case company 2.
I	An Overview of the Backup service SaaS platform at Case company 3	1 diagram	Illustrate the overview of the backup service SaaS platform at Case company 3.
J	Roles involved in the BDD framework of Project 3.	1 diagram	Illustrate the roles involved in the BDD framework of Project 3.
K	Phases of Project 3 in relation to BDD implementation.	1 diagram	Illustrate the phases of Project 3 in relation to BDD implementation.
L	Project 3: Task-Phase Mapping	1 diagram	Illustrate the task-phase mapping of project 3.

As shown in Table 2, the data collection rounds included analysis of technical documents that visually represent each project´s best practices. These documents typically present the flow from the requirement-gathering stages to the deployment and delivery of the product. They also convey information regarding the software tool used in implementing a product in a BDD framework. The next section examines the implementation of the BDD framework in three technology companies, including interviews with relevant stakeholders.

3 Available Knowledge and Best Practice on BDD Test Framework Implementation in Scrum Teams in IT Projects

This section highlights various aspects of the BDD guide that each Scrum team requires, focused on behavior-driven development for what and when; it also has to state the roles and involvement of the different stakeholders of the BDD process—consolidation of collaboration. It therefore underlines the strategy toward the alignment of a team with BDD so that all principles in place can be perfectly tailored inside the Scrum framework.

This guide covers major design patterns to implement BDD, best practices, and architecture considerations. The area also deals with essential tools and how they can connect with different programming languages. It covers both the positives and negatives of using BDD and highlights some of the main problems teams are bound to come across when adopting it. The report includes all the necessary information that will assist any Scrum team working on implementing BDD.

3.1 BDD Test Framework: Definition, Types and Key Elements

BDD is a collaborative approach involving everyone in order to come up with a shared understanding as to how the software should behave. In that it talks about conversations and examples between business stakeholders, developers and testers, in order to discover and make requirements clearer. This definition draws a line where communication and common vocabulary are additional concrete terms that can be used to remove ambiguity. (Lawrence 2019)

Facilitating collaboration and transforming business goals in executable specifications are the building blocks of implementing BDD in IT projects, and especially in an IT project under Scrum framework. The first step is to have the Product Owner to state what user stories and acceptance criteria should meet in terms of the behavior the system should have. After which they are elaborated on with the Development Team (developers and testers) collaboratively to derive concrete examples to depict what should happen. (Khomenko 2024)

The next level would be that the Development team takes these examples and turns them into structured scenarios using a domain language such as Gherkin. These are not just documentation, but executable specifications of how the software should behave

and hence can be automated to verify functionality. These scenarios will be used by developer to follow while they are coding and write the tests before or alongside the production code (the BDD framework and the TDD principles). In a good process, testers work hand in hand with developers to ensure the scenarios are complete and cover all the necessary edge cases. (Smart 2014)

It is the role of a Scrum Master to encourage this collaboration during the sprint, and to assure that BDD as a process is adhered to, and that impediments are removed when they occur. The BDD tests automating keep us on the path on the behavior that is desired. To execute these scenarios and generate living documentation, which is up to date information about the behavior of the system, tools and frameworks are used to automate the execution. This documentation can act as a communication tool for all the stakeholders who are outside the Scrum team. (Smart 2014)

The main feature of BBD is that it goes around defining the what (of the software's functionality, intended and from a business) as a result of cooperative discussions. The what is captured in user friendly scenarios, followed by the expected behavior they involve. This functionality is implicitly understood because of the why behind it, which creates these scenarios for the business value that they deliver. Finally, the resulting business requirements are translated into how by developing and testing the answer, which the software they build when development is done serves or does not serve the intended value for the stakeholders. (Deshmukh 2020)

3.1.1 BDD as a test approach

The BDD scenarios are one of the keys means for the team to understand, validate, and refine the functionality of the feature. These scenarios give crystal clear guidance for testers to conduct thorough testing. They also help developers understand the expected behavior, which helps in developing it efficiently and delivering it on time. (Manekar 2024)

As it comes with the built in clear, business readable language like Gherkin, it is naturally way easier to tie in with CI/CD pipeline because BDD scenarios, written in it, are executable specifications that can be automated using various BDD frameworks. The QA strategies of BDD and TDD help in allocating resources efficiently for they focus testing on the critical areas from very early in the process, automate the repetitive tasks to free up the testers to test the complex usage, provide the metrics of failing vs

succeeding on to the testers to guide on where to put the resource based on risk and value which results into minimizing the wasted effort and rework. (Tesvan 2024).

The BDD approach is also used at the coding level, where it helps developers to produce better-quality code, which is better tested, better documented, and easier to use and maintain. (Gopalakrishnan 2020). Where possible, such examples are automated as executable specifications that validate the software and provide automatically updated technical and functional documentation (Gopalakrishnan 2020). The figure below diagram will show the high-level overview of the BDD.

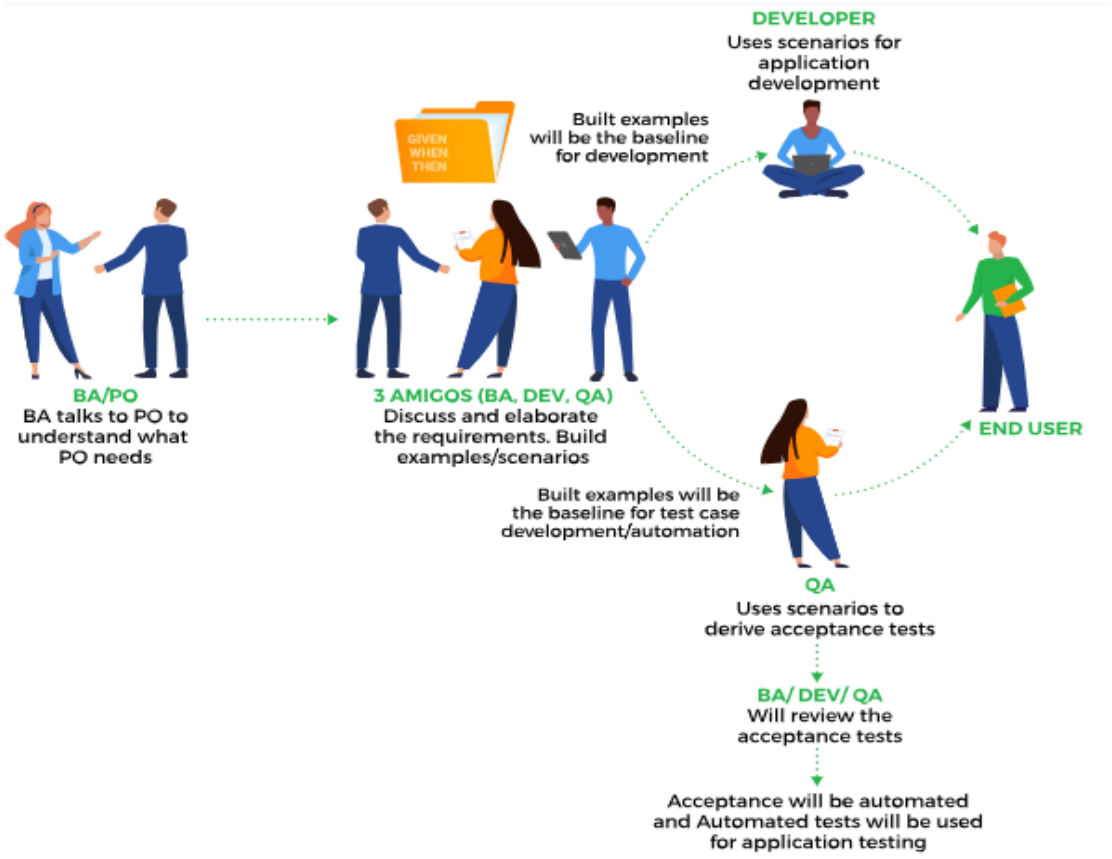

Figure 2. Software Development Life Cycle without Behavioral Driven Development. (Gopalakrishnan 2020)

As seen from Figure 2, the BDD practitioners start by identifying business goals and collaborating with other users. This diagram describes a software development process that is probably inspired by the "3 Amigos" approach, emphasizing collaboration and user-centric design. It starts with either a business analyst or product owner working in

tandem to understand the needs and requirements of the end user. (Gopalakrishnan 2020)

Next, a very important activity is the "3 Amigos" meeting: BA, Developer, and QA Engineer. These three get together to discuss and refine the requirements into solid examples and scenarios of how the application will be used. These examples and scenarios form one of the key bases for the development and testing processes. (Gopalakrishnan 2020)

Then, these scenarios are used by the developer to drive the development process of the application, which should meet the identified scenarios of interaction and functionality. In turn, these same scenarios would be used by the QA Engineer to write tests of acceptance—a set of verifications that check whether an application meets defined acceptance criteria. (Gopalakrishnan 2020)

After that, the 3 Amigos meet again when the acceptance tests have been developed for a joint review. Such a review helps ensure that the tests represent the user's requirements accurately and that the development team has a common understanding of the acceptance criteria. (Gopalakrishnan 2020)

Lastly, automation of the acceptance tests is done. Automation simplifies the tests in such a way that it increases efficiency by maintaining consistency. The goal is one of improving collaboration, making documentation understandable, increasing test coverage, automating to find defects earlier, as well as increasing feedback and value delivery through iteration. (Chaves 2023)

Summing up, BDD is a best test practice, especially for agile projects. It keeps the behavior of the feature being developed in focus and ensures common understanding by the whole team. The next section will discuss how BDD tests are integrated into the CI/CD pipeline.

3.1.2 Integration of BDD test cases into Continuous Integration/Continuous Deployment (CI/CD) pipelines

Configuring the pipeline to execute BDD tests as part of its build process for seamless integration within a Continuous Integration/Continuous Delivery pipeline. The major and well-known Continuous Integration/Continuous Deployment platforms include Jenkins,

GitLab CI, and Azure DevOps. They have strong mechanisms that can enable the execution of BDD tests upon every commit in your code in a proactive way to keep assuring testing on each change without having to do this manually. Firstly, add a build step to the Jenkins job. Run the SpecFlow tests by invoking the SpecFlow command-line tool and then include a pipeline step to execute the BDD tests during the release. This might be the running of SpecFlow tests as part of the pipeline definition. Integration of BDD means it includes the continuance of automated acceptance tests into your CI/CD. With every commit, CI/CD pipelines run automatically to execute predefined scenarios of BDD. It automatically does a deeper-level verification if the application behaviors are as expected per its specified requirements, which inherently adds reliability and quality to your application by means of a mechanism called continuous validation. (Gavandi 2024).

By automating BDD tests inside the CI/CD pipeline one will receive very fast feedback of the code to the development team ensures that the code is aligned with the defined behavioral specification. In the event that a commit brings new code that breaks the BDD tests, developers are being put on notice and can then work through the differences to fix the discrepancies. If the application's behavior is tested often, at every step of development, it enhances the code, quickens fixing issues and ensures only stable software moves through the deployment pipeline to be delivered. (Sheremeta 2025). The next section will discuss the generation of custom dashboard reports for BDD frameworks.

3.1.3 Custom Test Report Generation in BDD Frameworks

The figure below displays an example of a test report dashboard.

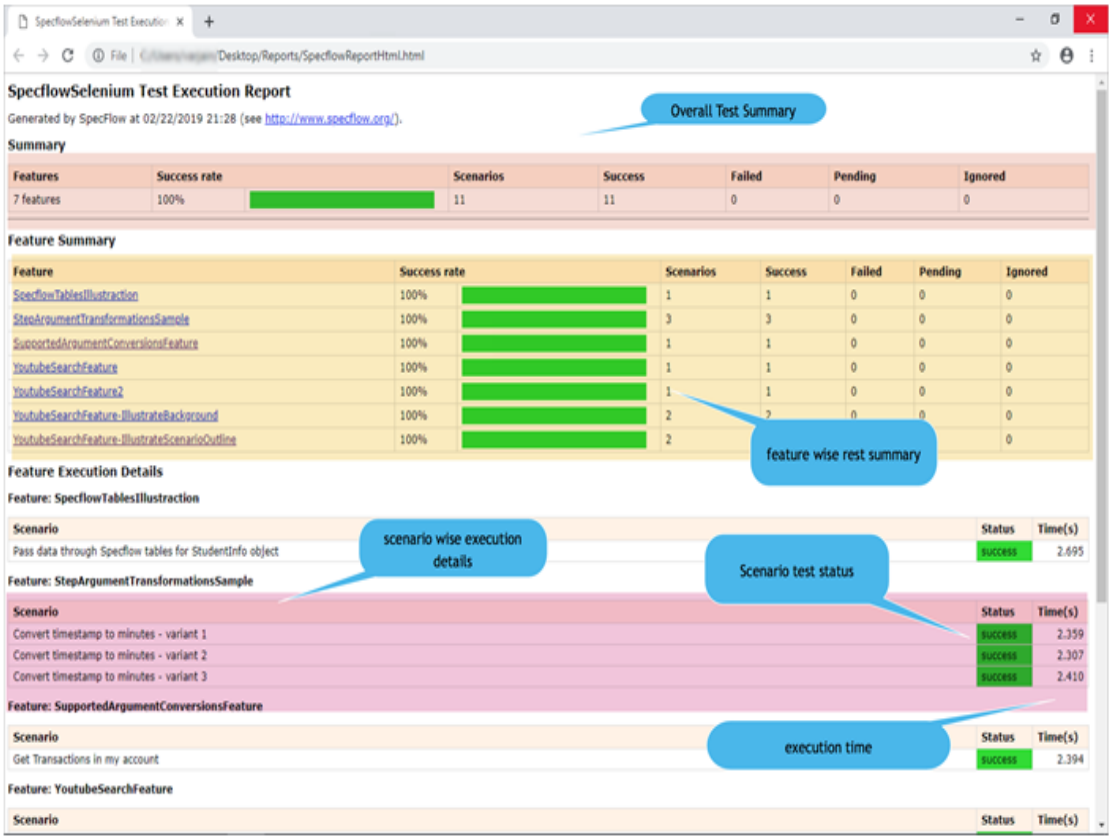

Figure 3. Specflow BDD HTML test report (Sruthy 2025).

As shown in Figure 3, the test report is generated using a reporting utility available with the BDD test automation framework. The generation of reports can be customized based on the user's dashboard requirements. Reports are generated based on the intended use case and analysis of test result data. (Sruthy 2025)

Figure 6 shows the detailed report which consists of information like feature name, success rate, screenshots that were captured, failed cases count, pending count, and ignored. It displays the feature summary, which contains the details that are clickable to view the failure reason with exception detail. This report helps the test engineers and development team to identify the root cause of the defect. For the tester, it will help to mention all the information in the defect, and for the developer, it helps to point out the exact point where the error has occurred and provide so many hints on where to fix the issue. This way, the report generation plays a very important role in the development and testing life cycle. (Sruthy 2025)

3.1.4 Recognized Advantages and Disadvantages of BDD

The recognized advantages of BDD typically include, first, *minimizing "waste".* The main purpose of BDD is to guide development to focus on those features that provide substantial business value and, therefore, should be prioritized. Therefore, it reduces waste associated with the development of a feature by making sure everyone understands the requirement. (Das 2024).

Second, another recognized advantages of BDD is *minimizing costs*. By enhancing the quality of application code through a focus on 'building the software right,' BDD contributes to reduced bugs. This reduced bug incidence translates into lower costs related to bug fixes, thus minimizing potential delays emanating from these bugs. (Das 2024).

Third, another important recognized advantages of BDD is *increasing the release velocity.* It happens because fully automated tests accelerate the release cycle significantly. Performing acceptance testing through automation minimizes the time needed for manual testing and releases the tester for higher strategic activities: exploratory testing, deep manual testing of a complex scenario, or any high-value activity needing human judgment and expertise. (Das 2024)

Fourth, Scenarios written in plain language help to understand the functionality and behavior that the system is supposed to support. (Akhtar 2024)

However, there are also some well-recognized disadvantages of BDD. First, BDD requires *very high business involvement* and *collaborative communication*. In other words, frequent open dialog with stakeholders is essential to ensure that development efforts actually deliver business value on a consistent basis. Without active stakeholder involvement in those discussions, successful implementation of BDD becomes considerably more difficult. (Das 2024)

Second, BDD thrives in a *collaborative environment* and struggles in isolation. Traditional development models have often included siloed teams where business analysts create detailed specifications, which then get thrown over the fence to development teams, who may be located remotely. In this respect, direct interaction is often lacking between the business analyst and developers; that iterative clarification and understanding of true requirements are so important to successful BDD implementation. (Das 2024)

Third, *writing automated acceptance tests* well, especially for complex web applications, requires *a special type of expertise,* which is usually one of the biggest hurdles facing most teams when trying to bring BDD into their work (Das 2024).

Fourth, If the project doesn't have much user interaction and the functionality is very straightforward, BDD may also be overkill which can result in over complication. (Akhtar 2024)

Summing up, the BDD provides several benefits, like reducing waste and boosting fast release. It also has some of the disadvantages, such as it demands more collaboration with all stakeholders and needs specialized expertise. The next section compares BDD with other contemporary testing best practices.

3.2 Implementing a BDD Test Approach in a Scrum Team

In IT sector, Scrum is the main driving engine for software development activity. The main importance of these scrum teams is the ability of the cross-functional skills that the members exhibit. In traditional methodology, such as the waterfall model, the development takes place phase by phase (Khomenko 2024). Scrum is time-boxed events to perform a particular set of items by a self-organized team called a scrum team. Typically scrum team members are cross-functional (Sutherland 2014).

3.2.1 Scrum teams of IT sector

In the modern agile methodology, all the software development processes, such as design, coding, test case writing, and data modelling, will take place in parallel. This approach makes the development process rapid and iterative in nature. The main importance of the scrum team is shared understanding of the requirements, project goals, and problem solving. (Martin 2012.)

Collaboration is the main aspect of a scrum team. The scrum ceremonies are the daily stand-up meeting, sprint planning, and retrospective. These ceremonies make the given scrum team collaborate and communicate effectively in a timely manner with transparency. It provides the stage to show the ownership, responsibility, progress, and blockers that the team faces on a daily basis. It also facilitates the scrum team to adhere to the changing requirements. (Shore 2008.)

Tools such as Jira, Confluence, and Slack are used for communication other than face-to-face meetings to document the requirements and assess the progress. This way, the agile scrum is shaping the modern development process. (Khomenko 2024)

Quality assurance is a part of the development life cycle. The quality assurance indicates the software that is under development is getting developed in the right direction in a timely manner. This approach reduces the cost of the project by ensuring error-free software at the end of the project and also minimizes the late defect detection at the customer end. (Crispin 2009.)

3.2.2 BDD framework as used by Scrum teams

The essence of BDD is all about user stories and scenarios to describe the behaviors of the software from the perspective of the end user. These are written in a structured format, usually in the syntax of Given-When-Then. (Gavandi 2024). The below table provides the meaning of Given-When-Then of the Gherkin scenarios.

Table 3. Notation used to express examples in BDD (Smart 2014).

1	Given:	<a context>
2	When:	<something happens>
3	Then:	<you expect some outcome>

As shown in Table 3, the BDD uses the given-when-then format to express the system behavior under development. Given that something is a prerequisite, when is the action, and then what is the outcome (Smart 2014). This approach used by BDD helps to ensure the same understanding of software requirements and expectations through the structured approach of all persons within a development process. This improves communication and reduces the chances of errors, hence producing quality software. (Gavandi 2024)

Additionally, BDD can be connected to automated testing tools to make specifications that can be run. As a result, developers can check the software's performance each time they write a piece of code. (Gavandi 2024) For example, when the new feature is going to be an "Add to Cart" functionality of an e-commerce application. They are supposed to

determine prerequisites, preconditions, and current systems. (Gopalakrishnan 2020) The below table shows an example for BDD scenario.

Table 4. Example of writing BDD scenarios using Gherkin (Gopalakrishnan 2020).

1	Given:	Given a customer has searched for a product
2	When:	When the customer adds a product to the cart
3	Then:	Customer should be able to see the shopping cart updated with the product name

As shown in Table 4, the customer searched for a product in the application. So that goes into the given statement. Now the actions associated with the scenario have to be decided upon, and the same is written as "customer adds product to the cart. It can be rewritten using the when statement. Finally, the outcome of the action was identified by the team and described as then the customer should be able to view the updated shopping cart with the product name, and the product quantity must be 1.

Summing up, this sub-sections discussed how the BDD scenarios are created using Given-When-Then statements. This way the Gherkin enables the user to write the BDD scenarios. The next section looks at the different tools that are available to perform BDD with various modern programming languages.

3.2.3 BDD tools

The table below illustrates the availability of BDD tools for various programming languages.

Table 5. BDD Tools (Manekar 2024).

1	Java	JBehave, Cucumber, Gauge
2	C#	Specflow
3	Python	Behave
4	Ruby	Cucumber
5	JavaScript	GaugeJS

6	PHP	Behat

As shown in the table 5 above, options to use for BDD implementation in Java are JBehave, Cucumber, and Gauge. Among these three, Cucumber is the most used with Java. In C#, the SpecFlow NuGet package supplied by Microsoft is mostly used for BDD scenario implementation. Behave is mostly used in the case of Python among all BDD tools. And for Ruby, Cucumber is used, similar to that in Java. (Manekar 2024.)

3.2.4 Implementing BDD test Framework for the first time

Implementing Behavior-Driven Development (BDD) for scrum teams of IT projects has many steps. The BDD approach is not just a test framework; rather, it is a collaborative approach in which all the team members of the scrum team are in sync and are involved in the development of the application as per requirements. These requirements should depict the behavior of the system under development. The core idea of BDD is to foster the communication and collaboration among the scrum team members and business stakeholders. By doing this, the scrum team member will better understand the requirements under development, and this will help to develop the right product. The ambiguity in requirements understanding can be nullified by this approach. (Jacon 2023)

According to Akhtar's 2024, the key elements of BDD are collaboration among scrum team members, emphasis on business value, and the requirements specifications are captured as user-friendly scenarios in simple plain English, which can be understood by both technical and non-technical staff. The format used to capture these specifications uses Gherkin syntax. According to Sheremeta's 2025, the companies can select behavioral-driven development practice if the requirements emphasize the behavior of the system, there is more scope for collaboration between the technical team and business stakeholders, and the development methodology should be agile.

To start with the BDD approach, all the stakeholders, including technical staff and business stakeholders, should be aware of BDD practice. If the BDD approach is new for the team, then initial training is required to learn it. To start with, the team can perform a pilot trial run to learn the BDD approach in a tiny timeline. (Jacon 2023)

The selection of BDD tools depends on the technology and programming language used by the development team for application development. If the programming language

used for development is Java, then the JBehave BDD tool is used; if C# is used for development, then the Specflow tool is used; if the development is happening in JavaScript, then the GaugeJS tool is used. The BDD tools are available for all advanced programming languages. Different roles involved in the BDD practice are as shown in the table below. (Manekar 2024)

Table 6. Roles involved in BDD (Khomenko 2024)

1	Business stakeholders such as customers, business analysts, and product owners.
2	Scrum master
3	Developers
4	Testers
5	DevOps engineer

As shown in Table 6, the main roles involved in BDD practice contain both technical and business stakeholders. These roles are collaborated on frequently in behavior-driven development practice. All these roles perform different responsibilities of BDD practice. The main intent of all the stakeholders in BDD practice is to develop the right product. The repeated interaction between these roles makes everyone understand the requirement specification clearly, which results in a product that is as required by the customer. (Khomenko 2024)

Table 7 illustrates the different responsibilities performed by each role in BDD practice.

Table 7. Responsibilities performed by each role in BDD (Khomenko 2024).

1	Business stakeholders such as customers and business analysts make the product owner understand the behavior of the system to be developed. The product owner analyses the requirements and writes the PBI's and acceptance criteria, which emphasize the behavior of the system.
2	Scrum masters facilitate the scrum team members to resolve any kind of blockers and hurdles. He communicates with the product owner to priorities the features and user stories for upcoming sprints.
3	Developers interact with other scrum members and the product owner to develop the application and also write the BDD unit tests. They push their code changes to CI/CD.

4	Testers start writing BDD test scenarios by reading the requirement specification and acceptance criteria provided by the product owner. They also write the BDD scripts and push the test code and test cases to CI/CD.
5	DevOps engineers are responsible for creating and maintaining CI/CD pipelines and agents to deploy the application code and test code.

As shown in Table 7, different roles have different responsibilities in the BDD approach. Table 8 provides the information about scrum ceremonies and the BDD approach.

Table 8. Behavior-Driven Development activities within scrum ceremonies (Khomenko 2024).

Product backlog refinement	In this phase the product owner, developers, and testers collaborate to understand and refine the product backlog items. The acceptance criteria, which were behaviorally centric, are discussed for better understanding.
Sprint planning	Product owner and team discuss the PBIs and turn them into little tasks for coding. It is decided by the team to develop the application in line with system behavior.
Daily scrum	The scrum team attends the daily meeting to talk about work progress, what has been completed on time and whether there are any problems stopping them. The primary objective in implementing their task will be to observe the behavior of the application development.
Sprint review and demo	At the end of the sprint, all scrum team members confirm the accomplishments of the current sprint with product owners, business analysts and customers. Stakeholders look at the sprint deliverables to confirm that what is being built is based on the BDD scenarios.
Sprint retrospective	In this meeting the team members will discuss what went well and what went badly. The team makes sure the main importance is given to the behavior of the system in the BDD retrospective. The future sprint changes and action items will be discussed.

As shown in Table 8, BDD tasks are performed in each scrum ceremony, starting from product backlog refinement till sprint retrospective.

According to (Smart 2014), implementing the BDD is a multi-step approach. First, it starts with understanding the BDD approach by training and piloting a small project if it is new to the scrum team. Then, the next step is to choose the BDD tool based on the programming language and technology used for the actual development of the application. Then, the 3 amigos, such as the product owner, developers, and testers, understand the user stories and come up with concrete expected behaviors. The test engineers will create the Gherkin scenarios in Given-When-Then format. Developers use

these scenarios for better understanding while developing and also write their unit tests in BDD format. Table provides the sample Gherkin syntax.

Table 9. Gherkin syntax in BDD (Smart 2014)

Given	Prerequisite
When	Action
Then	Output/Result

As shown in Table 9, Given-When-Then are the main building blocks of BDD scenarios.

Further on, the test engineers and developers push their code changes to the CI/CD pipeline for continuous deployment. In each sprint, the team will review and fix the development and test script errors to align with system requirement specifications. Finally, the scrum team and management access the BDD approach and advantages for their team and help other scrum teams to follow the same in their organization. (Gavandi 2014)

Summing up, the BDD development approach is a continuous process in which the technical and non-technical staff interact frequently to develop the system. The emphasis is provided on the behavior of the system under development. This approach is helpful for such projects that are following agile methodology. (Gavandi 2014)

The next section compares BDD with other contemporary testing best practices.

3.3 Comparison of BBD with Other Testing Best Practices

Table 10 below illustrates a comparison of BDD with TDD and ATDD frameworks. If the acronym BDD was used multiple times, the follow acronyms mean: TDD – Test Driven Development (Jacon 2023); ATDD - Acceptance Test-Driven Development (Jacon 2023).

Table 10. Comparison of BDD with TDD and ATDD frameworks (based on: Jacon 2023).

Parameters	**TDD**	**BDD**	**ATDD**

Definition	TDD is a development technique that focuses more on the implementation of a feature	BDD is a development technique that focuses on the system's behavior	ATDD is a technique similar to BDD, focusing more on capturing the requirements
Participants	Developers	Developers, Customers, QAs	Developers, Customers, QAs
Language used	Written in a language similar to the one used for feature development (Eg. Java, Python, etc)	Simple English, (Gherkin)	Simple English, (Gherkin)
Main Focus	Unit Tests	Understanding Requirements	Writing Acceptance
Tools used	JDave, Cucumber, JBehave, SpecFlow, BeanSpec, Gherkin Concordian, FitNesse, Jest, Jasmine, Protractor	Gherkin, Dave, Cucumber, JBehave, Spec Flow, BeanSpec, Concordian, Cypress	TestNG, FitNesse, EasyB, Spectacular, Concordian, Thucydides, Robot

Table 10 summarizes the difference between the advanced software testing practices currently available in the market. It shows the practices definitions, participants involved in development activity, programming languages used, and tools used to develop them. These details help to have an idea of which tools and techniques are available for different programming languages to adopt these testing best practices. Furthermore, Table 11 below illustrates a comparison between the BDD and traditional testing framework approaches.

Table 11. BDD approach and a comparison with traditional functional tests (Elyashevskyy 2018).

Feature	BDD (Cucumber)	Traditional Functional Tests (TestNG)
Readability of a test script	High	Low
Need for requirements	High	Medium
Ability to understand why a test fails	High	Medium / Low
Clarity of default reports	High	Medium
Costs and efforts to implement a scenario	High	Medium

Table 11 compares the BDD with traditional approaches on various features such as readability of test scripts, need for requirements, ability to understand why a test fails, clarity of test reports, and efforts required to implement test scenarios. This comparison facilitates the user's access to the advantages of BDD over traditional functional tests. The next section provides the conceptual framework for implementing BDD approach for scrum teams in IT.

3.4 BBD implementation as a phased approach

Based on the literature review presented above, the traditional BDD approach outlined below can be summarized as follows.

Table 12. Description of the traditional BDD approach (based on available literature and best practice discussed in Section 4): 5 phases of the BDD test framework and recommendations for its implementation by Scrum teams in IT projects.

Phase 1. Prerequisites, preparation and BDD tool selection	Phase 1 is about prerequisites and preparation for implementing BDD. It also provides the tools for BDD. (Akhtar 2024) To train Scrum team members and business stakeholders, the BDD sessions and pilot project can be organized. Every stakeholder participates to understand the Gherkin syntax (Given-When-Then). Everyone will discuss their points to adopt the BDD in their development work. (Chaves 2023) **BDD tools typically used** (Manekar 2024)**:** Java => JBehave, Cucumber, Gauge C# => Specflow Python => Behave Ruby => Cucumber JavaScript => GaugeJS PHP => Behat
Phase 2. Integration of the BDD approach into Scrum ceremonies	Phase 2 explains the integration of the BDD approach in scrum ceremonies. a) When doing backlog refinements, all parties involved agree on what the user stories should accomplish. The importance is given to how to develop it and why to develop it. b) Sprint planning: The team and product owner look at the PBIs and break them up into smaller items for the developers to work on. All team members agree to design the application to match the system's behavior. c) Daily scrum: Everyone on the scrum team comes to the daily scrum meeting to talk about their efforts, recent accomplishments and to mention any hurdles they have. Their main consideration in carrying out their tasks will be the behavior of the development application. d) Sprint review and demo: Members of the scrum team show the outcomes of the sprint to product owners, business analysts and customers. Stakeholders in the business review the work done in each sprint to verify it corresponds to the BDD scenarios created. e) Sprint retrospective: The team looks at the things that had success as well as the things that failed. The topic of guiding the development team to prioritize the system's behavior will be discussed. (Smart 2014; Khomenko 2024)
Phase 3. Implementation of BDD	Phase 3 is about implementing BDD in real time using a Gherkin scenario. (Das 2024)

	BDD scenario writing using Gherkin syntax: Business people, developers and testers are the main team in BDD. Everyone meet and discuss to write clear, readable, business scenarios that resemble plain English. **Roles involved in BDD:** (Gopalakrishnan 2020) 1. Business stakeholders such as customers, business analysts, and product owners. 2. Scrum master 3. Developers 4. Testers 5. DevOps engineer **An example scenario for "Add to Cart" functionality:** Given: Given a customer has searched for a product When: When the customer adds a product to the cart Then: Customer should be able to see the shopping cart updated with the product name In BDD, the developers will get insights and more understanding of the system behavior by understanding the BDD scenarios written by the manual test engineers. The automation test engineers will use BDD-relevant BDD tools to automate these scenarios. (Gopalakrishnan 2020)
Phase 4. Integration of BDD in the CI/CD pipeline & Continuous improvement	Phase 4 is about integrating BDD into the CI/CD pipeline for continuous improvements. One can select different CI and CD tools such as Jenkins, Azure or GitLab for your project. As soon as code is committed, a build job will automatically be processed on the pipeline in the cloud. New changes are now tested using BDD tests in the cloud pipeline. It checks the application in a quick and reliable way. (Gavandi 2024) Analyzing the behavior of the app at every stage of development increases the code's reliability, reduces the period needed for debugging and ensures that sailable software is deployed faster and more securely. (Sheremeta 2025) Continued communication and collaboration at each stage of development make all the stakeholders adhere to the behavior of the system under development. (Smart 2014)
Phase 5. Sharing best practice	Phase 5 is about sharing the best practices. It can be done in multiple ways. BDD wiki, workshops, and online sessions can be arranged to educate and help other teams. Start a BDD Wiki and make sure it is documented. Spread the materials among teams so all can benefit from using BDD. Learnings gained in one scrum team may assist other teams to accomplish their work. (Chaves 2023)

The above table 12 systematic phased approach helps the team better understand the implementation of BDD. This approach can help the companies here in Finland that are following agile methodology for application development and having the requirements that emphasize the behavior of the system. In the next section, the conceptual framework is described.

3.5 Conceptual Framework of This Thesis

Based on available knowledge and best practice discussed above, the framework for implementing the BDD approach by Scrum teams in IT projects can be roughly divided into five phases. The logic pointing to this split is pulled together into the conceptual framework for the next steps in this thesis.

Table 13. Conceptual framework: 5 phases of the BDD test framework and recommendations for its implementation by Scrum teams in IT projects.

Phase 1. Prerequisites, preparation and BDD tool selection	Initial setup for BDD adoption. Train Scrum team and stakeholders against BDD(Gherkin). - Carryout a BDD pilot project. - Conduct stakeholder discussion on implementation of BDD. – Choose adequate BDD tools (e.g., JBehave, Cucumber, Specflow). (Akhtar 2024; Chaves 2023)
Phase 2. Integration of the BDD approach into Scrum ceremonies	Embedding BDD within Scrum events. a) Backlog Refinement-Define behavioral acceptance criteria. b) Sprint Planning-Making development synonymous with system behavior. c) Daily scrum-The focus should be placed on behavior implementation progress. d) Sprint review & demo-Stakeholder review of enacted behavior. e) Sprint retrospective: Discuss and improve behavioral focus. (Smart 2014; Khomenko 2024)
Phase 3. Implementation of BDD	Practical application using Gherkin. Collaborative Gherkin scenario writing. Testers contribute scenarios. Automation engineers automate scenarios. Scenarios are the way developers come to understand system behavior. (Das 2024; Gopalakrishnan 2020)
Phase 4. Integration of BDD in the CI/CD pipeline & Continuous improvement	Automation of BDD test for continuous validation. Include BDD tests into CI/CD pipelines: for example, Jenkins or Azure. Trigger tests on code commits. – Use test results for feedback in a timely manner. Continuous communication on system behavior. (Gavandi 2024; Sheremeta 2025; Smart 2014)
Phase 5. Sharing best practice	Spreading knowledge about BDD and practices achieved. Develop a Wiki for guidelines for BDD. Organize BDD workshops/sessions. Share BDD materials across teams. Ensure BDD community of practice develops. (Chaves 2023)

As shown in Table 13, the conceptual framework for implementing the BDD for scrum teams in IT can be split into five phases. The first phase is *the Prerequisites and preparation*. In the first phase, Scrum team members and business stakeholders learn and perform a pilot project to understand the BDD approach. They will also discuss and choose the BDD tool for their projects.

In the second phase, *the Integration of the BDD approach into Scrum ceremonies* takes places. The Scrum ceremonies stand for the application of Scrum tools such as backlog refinements, sprint planning meeting, daily scrum meeting, sprint review and demo, and sprint retrospective meeting is described in detail.

In the third phase, *the Implementation of BDD* takes place. Here, the different roles of BDD are mentioned. This phase also contains an example Gherkin scenario example writing to illustrate the mode of implementation.

In the fourth phase, different *CI/CD Pipelines for continuous integration & Continuous improvements* in using BDD in development practice are applied.

In the fifth and last phase, Scrum teams share best practices learned while implementing BDD with other teams.

The next section reports on the results of 3 IT projects that implemented BDD within their agile software development teams. For the analysis of 3 projects this conceptual framework is applied.

4 Current State Analysis of Implementing the BDD Framework in 3 IT Projects

The current state analysis examines the BDD framework implementation in three IT Projects.

4.1 Overview of the Current State Analysis

This current analysis discusses how the BDD test framework was applied in three IT projects with the focus on their practices, stakeholders, methodologies, tools, architecture, advantages, and disadvantages related to BDD implementation. These IT projects were done within different domains, such as SaaS, healthcare, and visitor management tools, but were all big-scale ERP implementation projects and all implemented BDD in their Scrum teams, conducted for three different organizations.

The CSA was conducted in three steps. First, each project was described and, second, analyzed in terms of BDD implementation. Third, the cross-comparison among these projects was performed to identify the similarities and differences in BDD implementations.

The data for this current state analysis was collected from the stakeholders of these three different projects. The stakeholders involved in this discussion were from different geographical areas, mainly located in the USA and India. The 3 IT projects were analyzed together with their Scrum team participants, and interview field notes were recorded. Then, these notes were organized into a structured format to perform thematic analysis. The interviews had a similar list of questions related to the BDD implementation in the projects. The discussions covered everything from test report generation to cloud integration, then running automation test scripts, and a customer-centric approach in scenario development. Besides, the discussion also covered challenges with actionable steps that should be taken for effective maintenance of the system under development. In addition, key aspects of the BDD implementation process included teammate participation, meeting organization and scheduling, communication protocols established, documentation, and other relevant details. After discussing them individually, the points were systematically categorized into all these discussion points.

Table 14. Main themes in Project 1-3 stakeholder interviews.

1. Why BDD: What are the main reasons behind adopting BDD in the particular projects?
2. How BDD: How is BDD actually implemented in this project?
3. Development of framework: How long will the basic framework development take to implement BDD?
4. Applied Tools and Techniques: What are the tools and techniques used for BDD?
5. Team size: How many people participated in the agile project team?
6. Benefits of BDD: In what ways did the BDD contribute to the project with regards to various perspectives, such as development, testing, security, management, performance, customer, agility, and cost-effectiveness?

These data points were analyzed, and the BDD implementation framework in Projects 1-3 was identified. By doing this, several strengths and weaknesses were also brought to light in the BDD implementation in Projects 1-3.

4.2 Description of BDD Framework Implementation in 3 IT Projects

The description starts with an overview of each company domain, their respective areas of work, and their way of adopting Behavior-Driven Development (BDD) will be discussed.

First, *Project 1 at Case company 1* was focused on Visitor Management system, a high-tech security system to create completely new opportunities for schools to embrace their communities with peace of mind. The main work of the company was to provide visitor management tools for schools so that the school can be protected from people who have a criminal background. For example, "when a visitor arrives at the school," "when parents come to school, "when the user tries to check in with a valid ID card," and "when the front-end receptionist checks the user's criminal background using the service provided by Case company 1." These use cases outline the system being behaviorally driven.

The technology stack of this visitor management tool was formed using a front-end user interface developed using AngularJS, a service layer implemented using WCF (Windows Communication Foundation), a backend database used was SQL Server, and AWS cloud service was utilized. Generally speaking, in this context, the BDD framework is well suited since the system emphasizes more behavior of the system. The team had expert

automation test engineers, a developer engineer, cloud architects, a database administrator, security engineers, and manual test engineers. The company provided the subscription of Visual studio account to facilitate development activities. The scrum team had a very strong desire to adopt and implement modern best practices to achieve the project goal. Management was ready to take a calculated risk to support new practices. Given these factors, the company adopted BDD development practice.

At Case company 1, the interview was conducted with the technical architect, who led the design and implementation of the BDD framework for the project. The bulk of the discussion was around the adoption of BDD for the scrum team and the effective use of the SpecFlow BDD package for the Microsoft .NET environment.

Second, *Project 2 at Case company* 2 was devoted to extending a new generation of innovative healthcare solutions, which were strategically designed, carefully executed, and at the mercy of market insight and data to optimize the return on investment while essentially making life better for everyone who uses its service. Today, Case company 2, a transnational company, is trying to provide the best healthcare services, which were innovative, promising, and secure.

This Project 2 underwent the scope change due to a technology stack migration. This involved transitioning the front end from traditional ASP.NET three-tier architecture to modern Angular UI, introducing a service layer using microservices, and bifurcating the database calls to an API layer to make it loosely coupled. The angular front-end design would then interact with this microservice for data operations. This was how the underlying development stack itself was changed; the opportunity became available to build upon and further improve an automation testing framework. The proof of concept in the base framework allowed the team to do the implementation of BDD. Separation of concern was preserved by adhering to SOLID principles. The manual test case scenarios that were of business importance were chosen for automation. The scenarios were prioritized and ordered, and the relevant test suites and tags were added to each suite. The project budget provided enough resources to accommodate these improvements. Due to all these factors, the team decided to adopt and develop a BDD framework practice.

Third, Project 3 at Case company 3 was focused on multi-cloud data management and backup service. Case company 3 provides mainly data optimization, data protection, and

data availability to its clients who use their desktop and web applications. The company decided to use the automation testing using the BDD approach. This was because of the high involvement of business analysts within the scrum teams. The repeated changes in requirements and system behavior required an approach that would be flexible. The interaction between scrum team members such as product owners, developers, testers, and scrum masters were so frequent, which helped manual test engineers to create scenarios in Gherkin format. This methodology used simple English, which was easy to understand, helping bridge the gap in communicating evolving requirements within the development team. The company, being multinational with subscriptions to licensed software, gave flexibility to the scrum team to choose the best methodology.

4.2.1 Description and Analysis of BDD Framework Implementation of Project 1, Case company 1

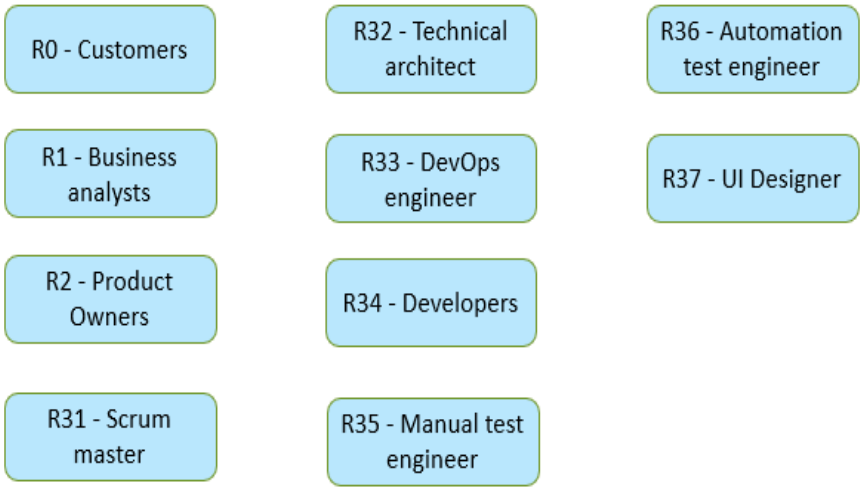

Figure 4. Roles involved in the BDD implementation of Project 1.

Figure 4 shows the roles involved in the implementation of BDD in Project 1, the different roles involved were Customers, Business Analyst, Product owner, Scrum Master, Technical architect, DevOps engineer, developers, Manual Test Engineer, Automation test, Churner, UI Designer.

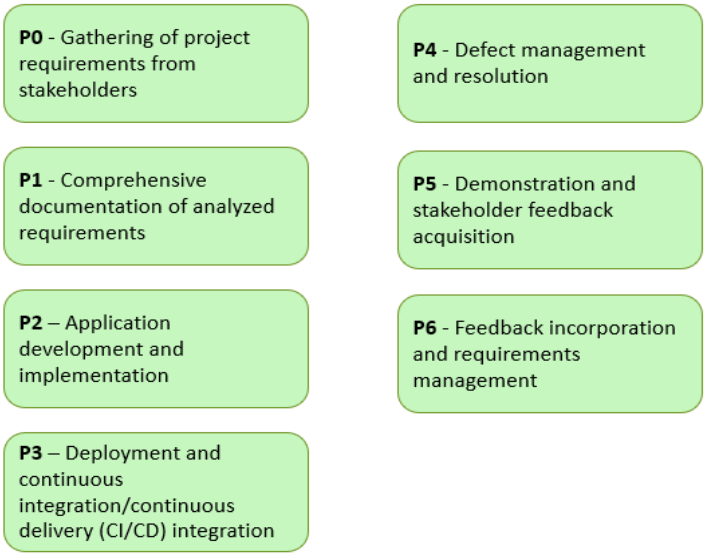

Figure 5. Phases of Project 1 in relation to BDD implementation.

This above figure 5 describes the sequential development of applications in Case company 1 (also implemented in Project 1). This approach involves collecting project requirements with P0, and all this was well documented in P1. The application development and implementation (P2) portion of the core development phase transitions immediately into deployment, followed by CI/CD integration (P3) to bring about an iterative and automated release process. After deployment, it focused on defect management and resolution (P4), i.e., application quality. It was crucial for gathering insights and validation to therefore acquire demonstration and stakeholder feedback (P5). Thus, the process, consisting of feedback incorporation and requirements management (P6), indicates an adaptive approach with feedback loops back being involved in refining the application and its requirements.

Table 15 provides an overview of all tasks performed and corresponding inputs and outputs.

Table 15. Role-based tasks 1-7 in Project 1.

T1		
Inputs	**Task performed**	**Outputs**
R1 collects Functional requirements for School Visitor Management System (Emphasis on System Behavior) from R0	Features and functionalities required from the software	Functional requirements specifications (FRS)

R1 collects Schools non-functional requirements of Visitor Management System from R0	Understand performance, security, usability, scalability, and other quality	Documented non-functional requirements (NFRs)
T2		
Inputs	**Task performed**	**Outputs**
R2 get detailed functional and non-functional requirements for the security of the school from R0 and R1.	Reviewing and clarifying detailed requirements to ensure alignment with the school's safety vision and security needs	Alignment of the product's vision and roadmap with school security and safety goals
R2 get market research, competitive analysis, and school safety trends from R0, R1, and other sources	Incorporating market insights into the product strategy and roadmap	Prioritization of features
T3		
Inputs	**Task performed**	**Outputs**
R31 prioritized Product Backlog, Sprint Goal suggestions from R2	Collaborating with the R2 to ensure the Product Backlog was refined and ready for sprint	Create sprint goals and PBIs
R31 publishes product backlog items to the scrum team in the presence of R2.	Refinement meetings, ensuring the development team understands the PBIs.	Refined Product Backlog Items with clear acceptance criteria with emphasis on the behavior of the system.
T4		
Inputs	**Task performed**	**Outputs**
Product backlog items selected for the sprint and corresponding user stories	R32 Decide the technical stack for product development and select the BDD plugin. R33 will create the CI/CD agents and pipeline. R34 will implement the requirements and develop BDD unit tests. R35 will author BDD functional end-to-end test cases using Gherkin statements. R36 will utilize the plugin provided by the R32 to automate the BDD test cases created by the R35. R37 will implement the front-end code.	Tech stack, technical assistance, and tools CI/CD setup Dev code and BDD unit tests BDD test cases Automated BDD test scripts User interface
T5		
Inputs	**Task performed**	**Outputs**
Dev code and BDD unit tests BDD test cases Automated BDD test scripts User interface	R34, R35, R36, and R37 push the changes to CI/CD pipelines.	Implemented dev code, test cases, and test scripts, and UI code reflects in CI/CD.
T6		
Inputs	**Task performed**	**Outputs**
Dev code bugs Test case changes	R34 fixes Dev bugs. R35 edit test case changes	Backend code with bug fix Edited test cases

Test script errors UI code bugs	R36 fixes script errors. R37 fixes UI bugs.	Test script with error fix UI code with bug fix
T7		
Inputs	**Task performed**	**Outputs**
Application code Test code Test cases CI/CD pipelines	R2, R1, and R0 review and accept these deliverables.	PBI marked as accepted and closed.
T8		
Inputs	**Task performed**	**Outputs**
Developed applicationTest artifacts such as test cases and test automation frameworks	R2, R31, R35, and R34 collectively look into feedback and incorporate changes in requirements, test code, and dev code.	Feedback incorporated development code and test code.

Table 15 provides an overview of the tasks performed and corresponding inputs and outputs. It describes the different roles and their contribution in performing tasks from requirement collection to final delivery. It helps to understand the roles and tasks performed by scrum team members in typical behavior-driven development at each stage. The main focus here was on the behavior of the system under development. It provides comprehensive information related to implementation and validation in BDD.

Figure 6 below shows a workflow where different phases involve specific tasks.

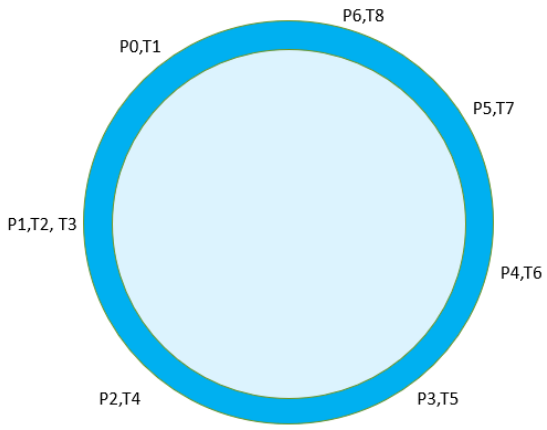

Figure 6. Project 1: Task-Phase Mapping

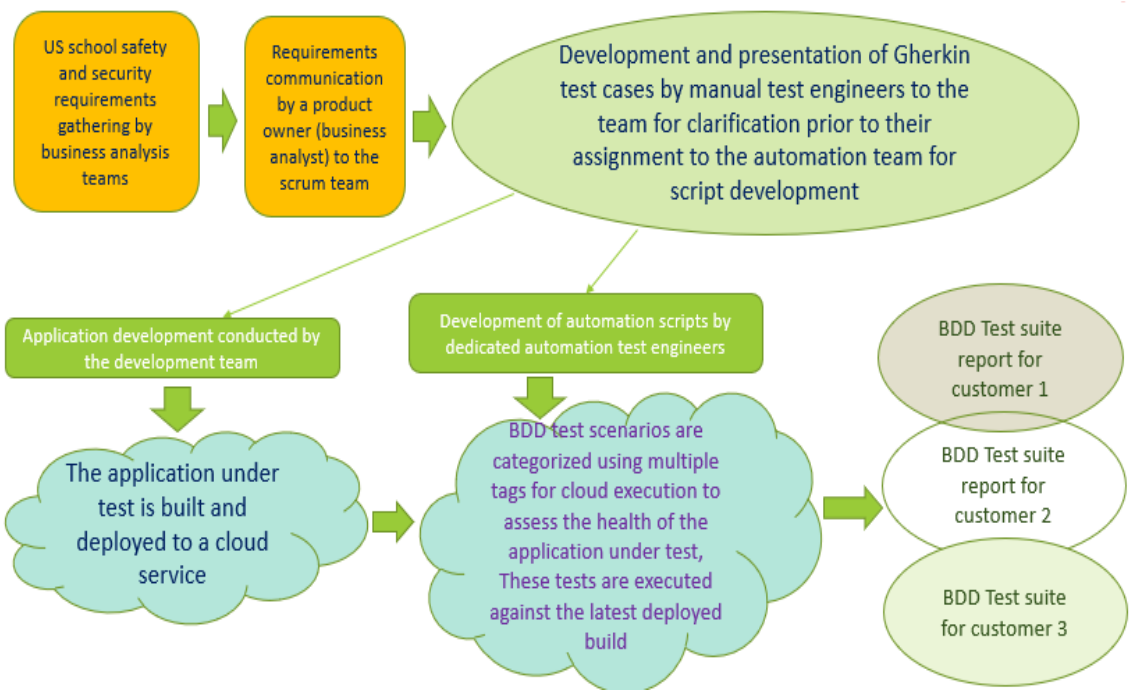

Figure 7. Cloud application development and test process using Behavior-Driven Development at Case company 1.

The aim of the project was to build the visitor management application for schools in the United States. To gather the requirements, many business analysts were deployed to the multiple schools across the country. These analysts' primary job was to collect and analyze the data, then identify the similarities and differences in use cases. This analysis resulted in detailed documentation of requirements in the form of user stories. These user stories were then assigned to scrum teams to develop the application using agile methodology.

Even though the whole team participated and contributed to the BDD implementation, the responsibility of writing the Gherkin scenario was mainly performed by manual test engineers. These scenarios were then presented and reviewed by the whole team. The product owner also reviewed the scenarios to realize how the behavior of the system was captured in test scenarios. These test scenarios were then automated by automation test engineers, and the developed scenarios were executed in the Azure test pipeline. These automation test suites were run against each build to ensure the health of the application. The failed test scenarios were analyzed by the testing team. The tests that actually failed due to application issues were mapped to the corresponding bugs and assigned to the development team for fixing.

The main strength of the scrum team was the clarity of the requirements that every team member had. The shared understanding of the requirements facilitated the developers writing the accurate unit tests, which led to the creation of high-quality software code methods.

The test scripts were executed in the Azure test pipeline, and HTML test reports were generated, which helped the test team to identify the errors in the application and track the bugs on a daily basis. The failed tests were mapped to the corresponding bugs in Azure DevOps, which helped the testers to see the progress of the build for fixed bugs. This process reduced the testing efforts and provided the testers with more quality and productive time to write new test cases and test scripts. The test pipeline was configurable to select the required browser to run the user interface tests, which facilitated the cross-browsing testing based on customer requirements.

All BDD functionalities were utilized, such as hooks functionality in SpecFlow. This enabled the team to do the test prioritization, ordering, and assigning tags to test cases.

There were multiple customers of the application, and each customer had a different tenant database. The test data to feed to test scripts was retrieved from this database at run time. The mechanism to fetch the test data based on the corresponding tenant build was handled in the underlying automation test framework. The tenant name and tenant database name were configurable as Azure test pipeline variables. This enabled the same set of test suites to run against different builds of different tenants. The team faced many challenges in designing and performing exploratory tests that depended heavily on workflows. This required a lot of manual work.

In project 1, the team first went through training related to the BDD approach. Every team member took participation, including the product owner, to understand the basics of BDD. The team decided to use the SpecFlow NuGet package for writing BDD Gherkin scenarios. The scrum master encouraged every team member to use BDD as effectively as possible. The testing team implemented the behavior-driven test cases, and the developers also wrote the unit tests using the BDD format. These code changes were pushed in Git and run in Azure Pipeline to check the build quality. The team was able to deliver on a frequent basis due to increased collaboration with business stakeholders and clear requirements.

In real time, the schools that were going to consume this application use many more third-party applications. This made it challenging to extend the BDD test framework to other applications. It was because of licensing restrictions of those third-party tools, and those were very unique in nature. The BDD implementation in project 1 made frequent delivery of new builds. Due to increased communication team members were able to discuss on daily basis and it helped to achieve the sprint goals on time.

4.2.2 Description and Analysis of BDD Framework Implementation in Project 2, Case company 2

At Case company 2, the main aim of the project was to migrate the application from outdated legacy technology to the latest best available technology. The project goal was to introduce the service layer, which was the most important part of the development. It used the agile methodology as a development practice, which involved the important ceremonies such as brainstorming and planning with all the stakeholders of the team, such as scrum master, product owner, business analysts, user interface designers, testers, business layer developers, cloud engineer, database engineers, and service layer development engineers. The detailed interview was held with the project manager who was involved in this process at all the stages of product development.

The detailed brainstorming was held with all the stakeholders in choosing the best development strategy, and the team assessed many of the best practices that are in use across the companies. Since the system was more reliant on behavior, the BDD was chosen as a development practice. The team weighed the detailed pros and cons based on available knowledge on this practice.

The team had some engineers who already practiced the BDD in their previous employment, but the company needed more engineers due to the need for new human resources who have the knowledge of implementing the BDD. The hiring of the new engineers took place, and the scrum teams were formed. These parallel scrum teams were given user stories that were interdependent. In the program increment meeting, all the team members had discussed the major issues and project timelines.

The customer was not only interested in the migration of existing application functionality to the new application but also had the new requirements to implement. These functionalities should be aligned with their present health management system.

The product owner actively participated frequently in scrum ceremonies since the new requirements behavior was very complex. The team interacted with the product on a frequent basis to come up with Gherkin scenarios for the API service layer. The team did not have direct contact with the customers since the clients were very confidential and placed in other geographic regions.

In the beginning of product development, the service layer deliverables were demonstrated to both customer and management. Some of the unseen issues were found while doing exploratory testing. To address these issues, the management managed to borrow extra time for delivery since there were core-level changes. This was possible due to repeated discussions with the customer in a timely manner, which BDD brought to the product development life cycle.

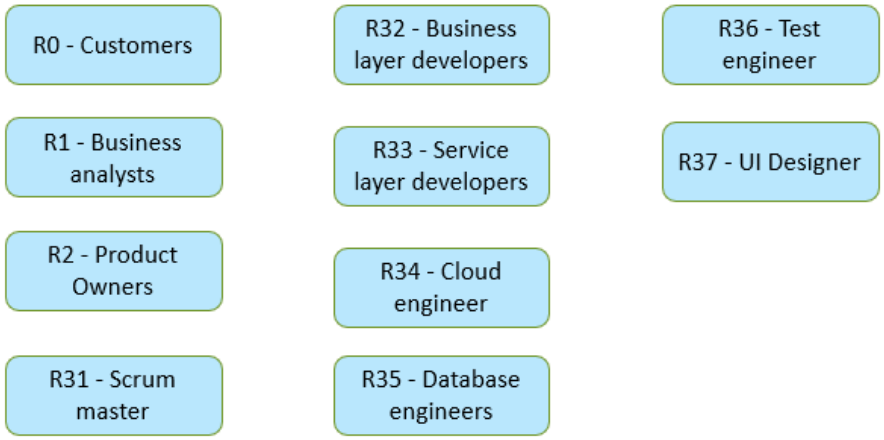

Figure 8. Roles involved in the BDD framework of Project 2.

Figure 8 shows the roles involved in the implementation of BDD in Project 2.

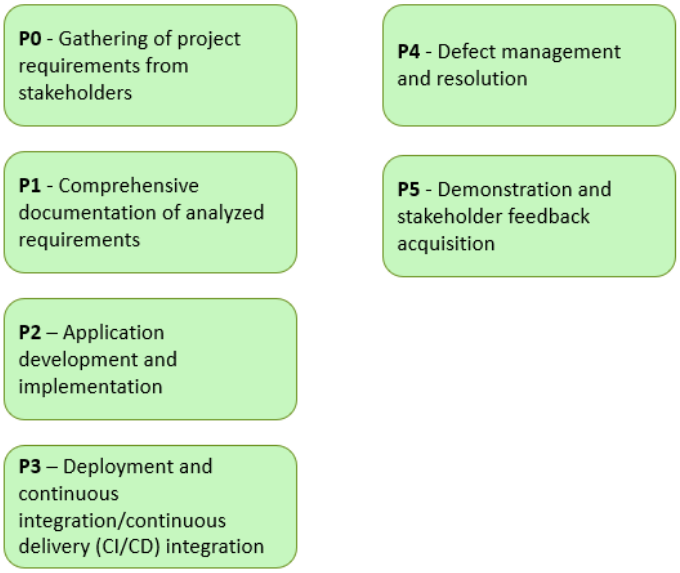

Figure 9. Phases of Project 2 in relation to BDD implementation.

This above figure 9 describes the sequential development of applications in Case company 2 (also implemented in Project 2). This included the collection of requirements from the stakeholders, developing and testing the system, deploying it, and then starting to gather feedback.

Table 16 provides an overview of all tasks performed and corresponding inputs and outputs.

Table 16. Role-based tasks 1-7 in Project 2.

T1		
Inputs	**Task performed**	**Outputs**
R1 collects Functional requirements for clinical app (Emphasis on System Behavior) from R0	Features and functionalities required from the software	Functional requirements specifications (FRS)
R1 collects non-functional requirements of clinical app from R0	Understand performance, security, usability, scalability, and other quality	Documented non-functional requirements (NFRs)
T2		
Inputs	**Task performed**	**Outputs**
R2 get detailed functional and non-functional requirements for the clinical app from R0 and R1.	Reviewing and clarifying detailed requirements to ensure alignment with the health care customer needs	Alignment of the product's vision and roadmap with customer goals

R2 get market research, competitive analysis, and clinical app trends from R0, R1, and other sources	Incorporating market insights into the product strategy and roadmap	Prioritization of features
T3		
Inputs	**Task performed**	**Outputs**
R31 prioritized Product Backlog, Sprint Goal suggestions from R2	Collaborating with the R2 to ensure the Product Backlog was refined and ready for sprint	Create sprint goals and PBIs
R31 publishes product backlog items to the scrum team in the presence of R2.	Refinement meetings, ensuring the development team understands the PBIs.	Refined Product Backlog Items with clear acceptance criteria with emphasis on the behavior of the system.
T4		
Inputs	**Task performed**	**Outputs**
Product backlog items selected for the sprint and corresponding user stories	R32 will implement business layer code and BDD unit tests. R33 will implement service layer code and BDD unit tests. R34 will create the CI/CD agents and pipeline. R35 will create data model and implement SQL stored procedures. R36 will write BDD Gherkin manual test cases and also write the BDD automation test scripts. R37 will implement the front-end code.	Business layer and Service layer Dev code and BDD unit tests CI/CD setup SQL stored procedures BDD test cases and scripts User interface
T5		
Inputs	**Task performed**	**Outputs**
Business layer and Service layer Dev code and BDD unit tests SQL stored procedures BDD test cases and scripts User interface	R32, R33, R35, R36, and R37 push the changes to CI/CD pipelines.	Implemented dev code, test cases, and test scripts, and UI code reflects in CI/CD.
T6		
Inputs	**Task performed**	**Outputs**
Service layer dev code bugs Business layer dev code bugs Test case and script changes SQL changes UI code bugs	R33 fixes service layer dev code bugs. R32 fixes service layer dev code bugs. R36 edit test case and fix script changes R35 fixes SQL changes. R37 fixes UI bugs.	Service layer bug fixes. Business layer bug fixes. Test case and script fixes Data base changes UI code with bug fix
T7		
Inputs	**Task performed**	**Outputs**
Application code Data base model Test code	R2, R1, and R0 review and accept these deliverables.	PBI marked as accepted and closed.

Test cases CI/CD pipelines		

Table 16 provides an overview of the tasks performed and corresponding inputs and outputs. It explains the Behavior-Driven Development (BDD) process across multiple stages of software development lifecycles, such as writing BDD Gherkin manual tests, highlighting focus on the behavior of the system while writing acceptance criteria, developing software with alignment with system behavior, and improving communication and shared understanding.

The figure 10 below shows a workflow where different phases involve specific tasks.

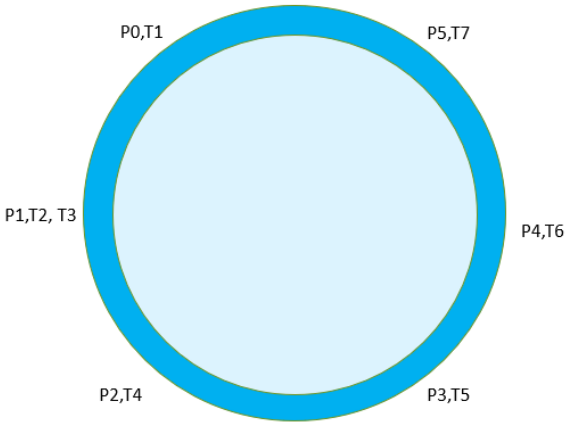

Figure 10. Project 2: Task-Phase Mapping

Project 2 explains the BDD implementation. It promotes early communication with stakeholders as well as scrum team members so that system behavior under development was understood clearly. That explains the early service layer demos and the focus on BDD acceptance criteria and testing. Ongoing communication brings more product delivery timelines.

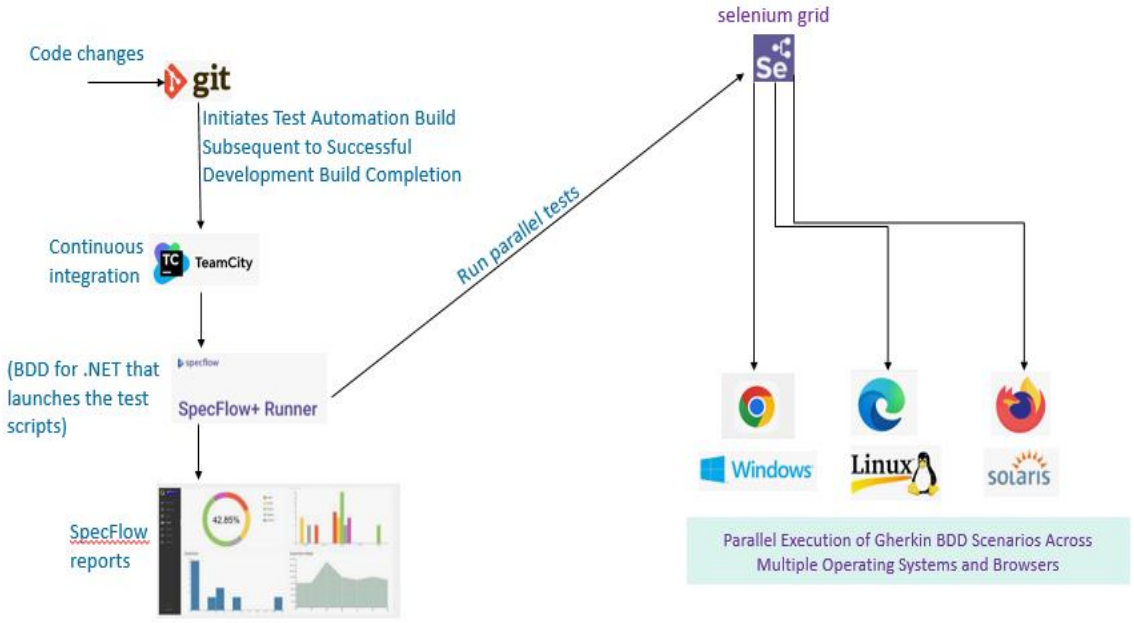

Figure 11. Continuous integration and automated testing workflow at Case company 2

The new automation framework, which was built using SpecFlow BDD, was configured to run against the latest build of the application under test using TeamCity as a continuous integration tool. Test execution was performed by Selenium Grid, which enabled parallel execution of test suites with multiple tags. This optimization of parallel execution resulted in faster testing of the application. The frequent releases took place due to an efficient testing mechanism. Finally, exploratory test execution was of high priority since the new functionalities were tested for the first time. The unseen issues were identified in the testing phase, which earned the client's confidence in the scrum team. This credit should go to BDD practice, which enabled this.

Overall, there was huge pressure on automation test engineers since the script maintenance of BDD scenarios was huge since the regression test suites had a huge number of test cases. The team managed to handle this workload due to reusable step methods for Gherkin statements.

Main aim of the project in this organization was to migrate the outdated application stack to new modern application. The team contains some people who already worked in previous companies on similar approach. The company also provided some level of training to the scrum members. Every team member actively participated in implementing BDD scenario from the beginning till end of all scrum phases to come up with software

which adhere to BDD best practices. The behavior of the system under development was retained in all the stages of software development.

4.2.3 Description and Analysis of BDD Framework Implementation in Project 3, Case company 3

The scope of the project at Case company 3 was to provide the backup service and data protection of data stored on Salesforce, Microsoft 365, physical drives, and cloud platforms. The company developed the Software as a Service (SaaS) application to facilitate the customer with these functionalities. The main aim of the product was to back up the data when the data gets lost due to any reason.

The main goal of the automation testing scope was to provide comprehensive test coverage and test each corner case. The test automation aimed to run against each development build to check the health of the application. The build quality was measured in repetitive passion.

The test scripts and test cases were mapped using the mapping functionality provided by Microsoft Visual Studio. This mapping saved the testers time to mark the test case result against each test case for every build. It also attached the test logs for each of the builds, enabling the software engineers to find out the root cause of the issue.

The technology stack used for automation testing was C#, the SpecFlow NuGet package, the Rest-Sharp NuGet package, Selenium WebDriver, and Azure as a cloud service. The BDD API tests were executed for each new build of the service layer to ensure the contract testing. This approach decreased the load on UI developers to consume contracts at the UI level without bothering about the readiness of the APIs.

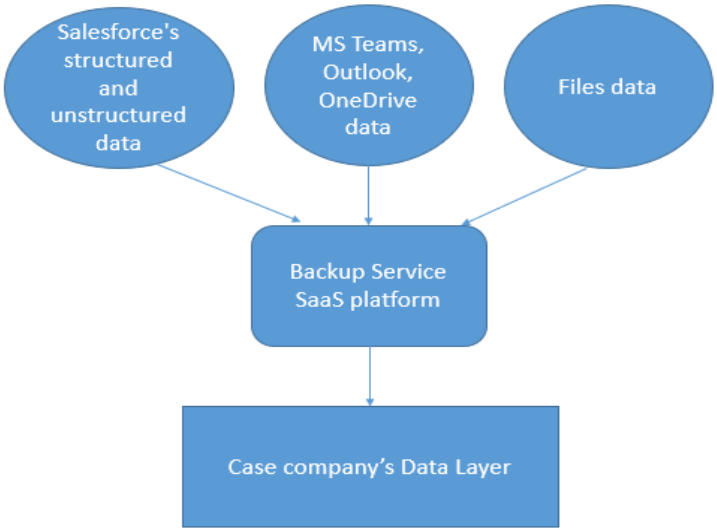

Figure 12. An Overview of the Backup service SaaS platform at Case company 3

The above figure 12 displays the overview of the high-level architecture diagram at the company where the back service, which was built as Software as a Server, takes a backup of Salesforce, Microsoft Teams, Outlook, OneDrive, and file data backup and saves it in the storage unit of the company where the data was stored and protected. If the customer loses any of the data in these portals, the company will provide backup service.

The product owner and business analysts discussed and delivered the required understanding to the scrum team on a daily basis, which helped the development team to align the development activity with respect to the user requirement. The manual test engineer contributed in the BDD scenario writing and automation test engineers write scripts for the same. The technical architect constantly provided the guidance to adhere to the BDD framework so that the system was developed in the right direction.

The team members who were new to BDD have gone through an understanding seminar held by the BDD experts at an organizational level. The resources were trained in BDD basics like writing Gherkin statements, understanding and interpreting requirements in a behavioral-driven pattern, and frequent interaction with business analysts to make sure the system was in line with requirements. Another scrum team that was not following BDD faced issues while having scrum of scrum due to a gap in understanding of the requirement. The team that followed the BDD approach was more aligned with requirements than others. This indicated to everyone the power of the BDD approach in a given condition.

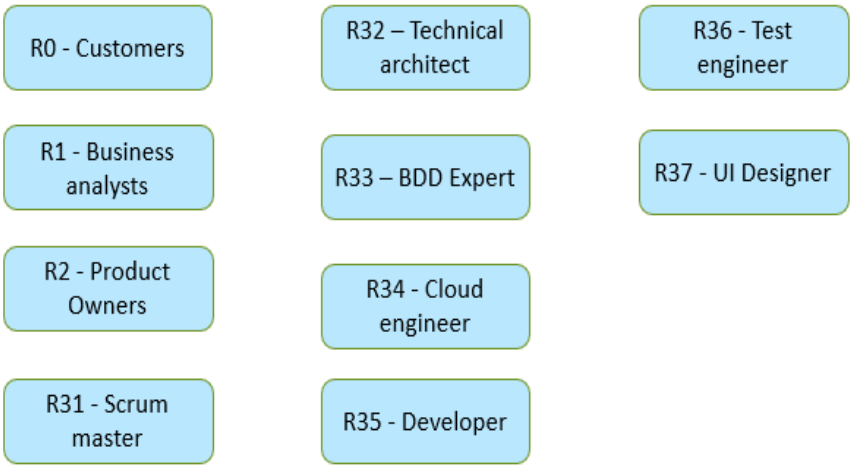

Figure 13. Roles involved in the BDD framework of Project 3.

Figure 13 shows the roles involved in the implementation of BDD in Project 3.

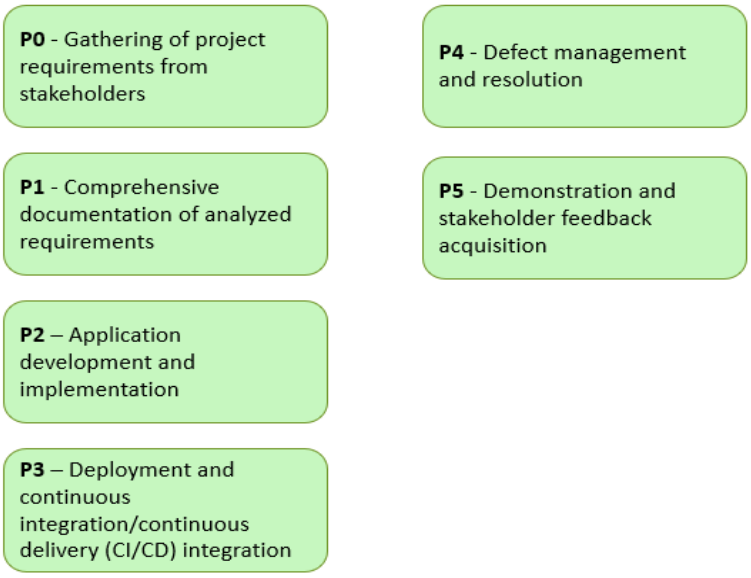

Figure 14. Phases of Project 3 in relation to BDD implementation.

This above figure 14 describes the sequential development of applications in Case company 3 (also implemented in Project 3). This approach involves collecting system requirements from customers, then development, testing, deployment, and receiving feedback.

Table 17 provides an overview of all tasks performed and corresponding inputs and outputs.

Table 17. Role-based tasks 1-7 in Project 3.

T1		
Inputs	**Task performed**	**Outputs**
R1 collects Functional requirements for back up service (Emphasis on System Behavior) from R0. R33 helps R1 to link BDD scenarios back to business goals and value.	Features and functionalities required from the software	Functional requirements specifications (FRS)
R1 collects non-functional requirements of backup service from R0	Understand performance, security, usability, scalability, and other quality	Documented non-functional requirements (NFRs)
T2		
Inputs	**Task performed**	**Outputs**
R2 get detailed functional and non-functional requirements for the backup service from R0 and R1. R33 helps R2 to align with BDD practices in requirement analysis.	Reviewing and clarifying detailed requirements to ensure alignment with the back service customer needs	Alignment of the product's vision and roadmap with customer goals
R2 get market research, competitive analysis, and back service trends from R0, R1, and other sources.	Incorporating market insights into the product strategy and roadmap	Prioritization of features
T3		
Inputs	**Task performed**	**Outputs**
R31 prioritized Product Backlog, Sprint Goal suggestions from R2	Collaborating with the R2 to ensure the Product Backlog was refined and ready for sprint	Create sprint goals and PBIs
R31 publishes product backlog items to the scrum team in the presence of R2.	Refinement meetings, ensuring the development team understands the PBIs.	Refined Product Backlog Items with clear acceptance criteria with emphasis on the behavior of the system.
T4		
Inputs	**Task performed**	**Outputs**
Product backlog items selected for the sprint and corresponding user stories	R32 decides the tools and technologies to be used for development activity. R33 helps with BDD approach to R32, R35, R36. R34 will create the CI/CD agents and pipeline. R35 will develop the application and also write the BDD unit tests. R36 will write BDD Gherkin manual test cases and also write the BDD automation test scripts. R37 will implement the front-end code.	Tools and technology BDD approach and practices help guide. CI/CD setup Dev code BDD test cases and scripts User interface code
T5		

Inputs	Task performed	Outputs
Dev code BDD test cases and scripts User interface code	R35, R36, and R37 push the changes to CI/CD pipelines.	Implemented dev code, test cases, and test scripts, and UI code reflects in CI/CD.
T6		
Inputs	**Task performed**	**Outputs**
Dev code bugs UI dev code bugs Test case and script changes	R35 fixes dev code bugs. R37 fixes UI dev code bugs. R36 edit test case and fix script changes	Dev code bug fixes. UI code bug fixes. Test case and script fixes
T7		
Inputs	**Task performed**	**Outputs**
Application code both UI and backend Test code and test cases CI/CD pipelines	R2, R1, and R0 review and accept these deliverables.	PBI marked as accepted and closed.

Table 17 provides an overview of the tasks performed and corresponding inputs and outputs. It shows a BDD driven project through which the system behavior was driven from requirements gathering and through acceptance. The practices of BDD were helping to analyze the requirement, refine the backlog with the behavior criteria, and generate behavior base tests (unit, manual automated by Gherkin).

The figure 15 below shows a workflow where different phases involve specific tasks.

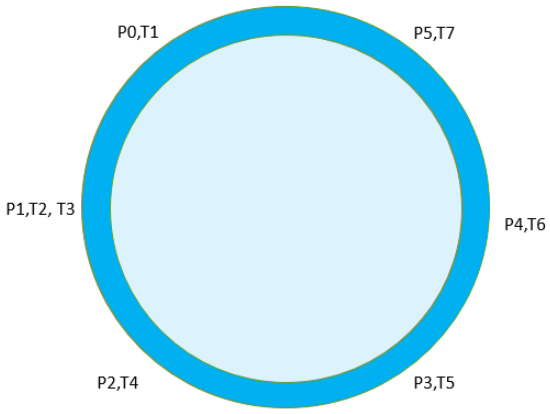

Figure 15. Project 3: Task-Phase Mapping

In this project the team had a BDD expert for initial understanding and set up. The technical architect and developers took suggestions from the expert in choosing tool for implementing scenarios in Gherkin format. The BDD expert facilitated everyone in all the phases of software development with respect to BDD approach. The product owner

communicated with team on timely manner which helped the team to better understand the business need and deliver the software soon. The BDD implementation helped the team to take the full use of BDD approach due to presence of BDD expert. The expert helped to make sure the development of software adheres to business objective and goals.

In Project 3, the BDD implementation itself took an approach that was centered on behavior from the very beginning. Its intent was to make sure that the gathered requirements emphasize the system behavior in respect to the business value. Also, the requirements analysis was consistent with the traditional BDD practices. Behavior driven acceptance criteria were used and recorded in the product backlog, which then directly drove development (BDD unit tests) and testing (manual and automated tests).

4.3 Key Findings from the Analysis of BDD Framework Implementation in Scrum Teams of 3 IT Projects

The key findings of BDD implementation in Scrum teams of 3 IT projects were summarized in Table 18 below.

Table 18. Common Characteristics of BDD Implementation in 3 IT Projects

Feature	Project 1	Project 2	Project 3
Project Goal	Visitor management application for schools	Application migration and new feature implementation	Backup service and data protection SaaS application
BDD Approach	Scrum team adoption, SpecFlow for .NET	Agile methodology, SpecFlow	SaaS application development, SpecFlow, C#
Requirements Gathering	Business analysts collected user stories from schools	Brainstorming with stakeholders, user stories	Product owner and business analysts provided daily guidance
Gherkin Scenario Creation	Primarily by manual test engineers, reviewed by the team and product owner	Developed collaboratively, focus on API service layer	Manual test engineers contributed, guided by technical architect
Test Automation	Automated scenarios executed in Azure DevOps pipeline	SpecFlow, TeamCity, Selenium Grid for parallel execution	C#, SpecFlow, Rest-Sharp, Selenium WebDriver, Azure
Testing Process	HTML test reports, bug tracking in Azure	Exploratory testing prioritized, frequent	Comprehensive test coverage, mapping of

	DevOps, cross-browser testing, Testing third-party application integrations, exploratory testing of workflows	releases, Maintaining BDD scenarios due to large regression suites	tests, API contract testing
Challenges	Clarity of requirements, high-quality code, reduced testing effort, test prioritization, tenant-specific test data	Improved communication and collaboration, early issue detection, increased client confidence	Comprehensive test coverage, API contract testing, improved team alignment on requirements
Strengths/Benefits	No specific training mentioned	New engineers hired with BDD experience	BDD training provided to new team members
Training	SpecFlow, Azure DevOps, Azure Test Pipeline	SpecFlow, TeamCity, Selenium Grid	C#, SpecFlow, Rest-Sharp, Selenium WebDriver, Azure
Key Tools	Whole team participated, roles defined for scenario writing, automation, etc.	Parallel scrum teams, all stakeholders involved	Scrum teams, product owner, business analysts, technical architect

Table 18 shows how BDD facilitates the shared understanding of requirements among all the team members, such as developers, business analysts, scrum masters, automation test engineers, manual test engineers, and also database administrators.

Project 1 (Case company 1) had the main strength in the clarity of the requirements among all the stakeholders. This shared understanding enabled the developers to write the proper unit tests to improve the code quality. This clearly shows how BDD eliminates the understanding gaps between the business and technical teams. The use of Gherkin scenarios in all three organizations illustrates the systematic approach in adopting the BDD in their IT teams. This approach also ensures everyone has the same understanding of the given team, leading to higher quality software.

Project 2 (Case company 2) involved the migration of the legacy application to introduce a new service layer. The complexity of the new requirements needed the frequent interaction between the business and technical teams to come up with relevant Gherkin scenarios. This interaction between technical and non-technical staff was very important for the success of the project. The analysis of BDD adoption at Case company 2 shows that the exploratory test was also important apart from the BDD implementation, which uncovered the unseen issues. This indicates that BDD was not the ultimate solution for the communication, but it enhances the interaction, which leads to more understanding of the business needs.

Project 3 (Case company 3) had two teams involved, the scrum team, which was following the BDD approach, and another team, which was not following it. The study clearly indicates the issues faced by the team, which was not following the BDD related to the understanding of the requirements among the technical team. This solid evidence provides the positive impact of BDD implementation in the agile development team. This alignment between the business and technical teams increases the efficiency of the development team to achieve a higher level of quality.

4.3.1 BDD facilitates early testing and continuous integration, leading to improved build quality and faster feedback cycles.

The participants experience challenges when integrating their old legacy system into the new BDD approach. This can be seen by the following quote by Respondent 1, who was a technical architect at Project 1:

> *"We are seeing that the old legacy system is monolithic and did not well aligned with BDD practice. It requires the detailed refactoring to create a testable interface. The team is struggling to break the old big legacy into smaller ones to make it align so that it can be managed using the BDD approach."*

This indicates the importance of evaluating system architecture while adopting the BDD for the project. Particularly in the company where the technology was very old. The transformation not only requires the new practice implementation but also altering the existing technology to new, latest, maintainable applications.

Furthermore, BDD can be integrated with continuous integration (CI) pipelines, automating the Gherkin scenarios with each build. Case company 2 uses TeamCity and the Selenium Grid for parallel test execution, enabling testing on frequent builds.

At Project 3, the main focus of implementing BDD was to cover all the corner test cases and generate comprehensive test coverage for the application using BDD implementation. The use of the BDD API test scenario indicates the proactive service layer testing, which resulted in reduced errors at UI integration. This helped the UI developers to easily integrate the API endpoints with the UI layer without worrying much about the data being bound because this data was already tested by BDD layer test scenarios. The mapping of the test scripts with Visual Studio and the generation of test logs increased the efficiency of the testing team to do the thorough testing of the

application under development. Running the test scripts against each new development build validated the health of the application at all the phases, ensuring the quality of the product. This was how BDD contributed to continuous integration and testing.

4.3.2 BDD requires investment in training and adaptation to address specific project challenges.

Effective implementation of BDD requires substantial investment in training the resources of the organization. This can be very easily seen by all the case companies discussed in this thesis. The transition of the BDD framework not only needs the tools but also a detailed understanding among the scrum team members about how to do it. This education and training for the scrum teams requires an adoption of some of the team culture and processes. Once the team was able to adhere to this practice, it was easy to implement the BDD effectively in the company. To handle the situation of third-party integration and handling tenant-specific data, it requires very skillful and creative changes in the BDD framework. The ability to learn all these techniques was a key component in adopting the BDD successfully.

4.3.3 Phased approach to implementing BDD Implementation and other strengths and weaknesses

This section summaries the findings identified in implementing the phased approach to Behavior-Driven Development (BDD) framework in Scrum Teams of the 3 IT projects.

Project 1 (Case company 1) had the most robust way of gathering and understanding the requirements compared to the other projects. It also had better testing efficiency which focused more on Azure integration.

Project 2 (Case company 2) exhibited stronger collaboration as compared to the other projects. Its Scrum team and the product owner exhibit strong communication which ensures the right product was being developed.

Project 3 (Case company 3) proactively invested in BDD training to benefit from its advantages. It shows Case company 3's long-term vision for their application which was missing in the other two case companies. The below table 19 shows the overview of key comparisons between the case companies.

Table 19. Summary of the findings in the Projects 1-3 in BDD implementation in terms of a phased approach.

	Project 1, BDD implementation	**Project 2, BDD implementation**	**Project 3, BDD implementation**
Phase 0	Requirements Gathering (Emphasis on System Behavior)	Requirements Gathering (Emphasis on System Behavior)	Requirements Gathering (Emphasis on System Behavior)
Phase 1	Detailed Requirement Review & Clarification (Security Focus) (Prioritized Backlog Refinement, Sprint Goal Creation, PBI Publication with Acceptance Criteria)	Detailed Requirement Review & Clarification (Healthcare Focus) (Prioritized Backlog Refinement, Sprint Goal Creation, PBI Publication with Acceptance Criteria)	Detailed Requirement Review & Clarification, Alignment with BDD Practices (Prioritized Backlog Refinement, Sprint Goal Creation, PBI Publication with Acceptance Criteria)
Phase 2	Sprint Backlog Implementation (Dev, CI/CD, Tests, UI)	Sprint Backlog Implementation (Business Layer, Service Layer, CI/CD, Data, Tests, UI)	Sprint Backlog Implementation (Tools & Tech, CI/CD, Dev, Tests, UI)
Phase 3	CI/CD Pipeline Execution (Code, Tests, UI)	CI/CD Pipeline Execution (Code, Tests, Data, UI)	CI/CD Pipeline Execution (Code, Tests, UI)
Phase 4	Bug Fixing (Dev, Tests, UI)	Bug Fixing (Code, Tests, Data, UI)	Bug Fixing (Code, Tests, UI)
Phase 5	Sprint demo and feedback	Sprint demo and feedback	Sprint demo and feedback
Phase 6	Feedback incorporation and requirement management	no Phase 6	no Phase 6
Other Strengths in BBD implementation	Strong Requirement Clarity, Efficient Testing Process, Effective Use of BDD Features, Tenant-Specific Testing	Improved Communication & Collaboration, Agile Approach, Parallel Execution (Selenium Grid)	Comprehensive Test Coverage, Improved Team Alignment, Investment in Training, Effective Tooling
Other Weaknesses in BBD implementation	Challenges with 3rd Party Integrations, Limited Information on Collaboration	Maintenance Overhead (Large Regression Suites), Limited Direct Customer Interaction	No Major Weaknesses Highlighted (Potential challenges with multi-team consistency & evolving requirements)

As for the other weaknesses identified across all the projects, *Project 1* had issues with 3rd party integration in their application and exhibit limited collaboration. *Project 2* had maintenance overhead of huge regression test scenarios and also there was limited

customer interaction for required understanding. *Project 3* did not exhibit any major weaknesses but challenges with hybrid teams and evolving requirements.

Overall, it was identified that extensibility, collaboration, and training of the resources provided major opportunities across the companies to effectively implement the BDD in their Scrum teams. In the next sections, these points will be kept in mind to come up with a robust guide for implementing BDD. The next section provides details related to building the proposal.

5 Building Proposal for implementing the Behavior-Driven Development (BDD) Framework in IT Projects

In this section, the initial proposal was created by considering current state analysis, conceptual framework, and co-creation knowledge.

5.1 Overview of the Proposal Building Stage

This proposal explains the details for implementing BDD in the scrum teams. This proposal tries to address the gaps which were identified across case companies. This way it will create the comprehensive and extensible proposal for implementing the BDD framework.

The goal of this section is to create a comprehensive proposal that can be utilized to increase the collaboration among team members of Scrum, reducing the misunderstanding in requirement understanding between the business and technical teams.

The conceptual framework, which developed based on literature review and available knowledge, contributes to creating this proposal.

The BDD implementation in case projects helped to create this proposal. The insights from industry experts are taken into consideration. The proposal uses the phased approach to implement BDD.

To develop this proposal, first the conceptual framework knowledge is taken into consideration. Secondly, the current state analysis of three case companies with co-creation expertise was utilized. Third, the expert suggestion is taken into consideration.

5.2 Findings from Data 2 (pulling together CSA, CF and Data 2)

The first one is the current state analysis on the implementation of the BDD approach for three separated IT projects in the SaaS, healthcare, and visitors' management domains. This analysis through a comparison of their practices, stakeholders, methodologies, tools, and identified advantages and disadvantages, shows that there is both similar motivations to adopt BDD, like improved communication and better testing, and context

particular implementations people within how the project requires and the technological landscape. What key findings have identified for their role in enabling BDD, and what challenges were uncovered with regard to BDD's integration with complex testing scenarios and integration with external systems? This paper provides a basis for a practical guide that will help IT companies implement BDD in an effective way in their process of development.

The second is conceptual framework. In this, there is a 6-step process to integrate BDD into Scrum. In this case, the BDD comes in this way: a team training and selection of tools, and then the launch of BDD to all (or most) Scrum events to say what is needed those Scrum events to sound like. The 'three amigos' write together clear scenarios that require development and continuously run on the CI/CD pipeline. Current validation runs constantly and drives continuous improvement of code quality and delivery. In the end, this framework makes the knowledge of BDD spread even more to the teams. Essentially, it just brings BDD's behavioral emphasis to increase collaboration, promote development, and ensure the quality of the Scrum cycle.

The table 20 provides the information related to key stakeholder suggestions.

Table 20. Key stakeholder suggestions (Data 2) for the Proposal building, shown against the CF ideas and findings from the CSA (Data 1).

Key focus areas from CSA (Data 1)	**Inputs from literature (CF)**	**Suggestions from stakeholders for the Proposal (Data 2)**	**Descriptions of their suggestions, in more detail (Data 2)**
1. A clear approach and guidance are needed to BDD implementation for all Scrum team members	1. A phased approach to BDD implementation	a) Construct a framework and guidelines which can be informed to all Scrum team members for BDD implementation	Identify the context-specific implementation based on available technical expertise.
2. Extensibility (not enough 3rd party integrations, not enough tenant specific data).	The extensibility framework calls for time discussing their approach to the API first and configuration.	a) Construct a framework which can be extended for the purpose of integrating 3rd party tools. b) Implementation of tenant-specific configuration	a) "We want to have a plug and play integration system." "I thought, well if we can just have a central API layer that you could easily extend that would solve many of our problems." - Technical Architect. "We have to have a configuration service, where we will be able to manage tenant data without touching core code." - Database Administrator. b) Stakeholders advised creation of reusable components and a loose coupling design. stakeholders

			also recommended well documented third-party integration.
3. Limited collaboration between teams and stakeholders (Collaboration)	Regular communication and shared understanding among cross-functional teams.	a) Regular meetings called 'BDD refinement' should be held which involves all stakeholders. c) Centralized knowledge repository for improving the community practice of shared documentation.	a) "But we need more regular sessions where all our business analysts, developers and testers can get together and review and refine scenarios." "This will make sure everyone is on the same page." - Business Analyst. "One thing that we should have would be a wiki or similar tool, where all BDD scenarios, requirements, and test results are easily accessible to everyone." - QA Lead. b) Stakeholders recommended formalizing the '3 Amigos' meetings and having clear agendas and action items for meetings as well. Additionally, stakeholders suggested that the project stakeholders should have access to an identical set of collaboration tools.
4. Training (Varied levels of BDD expertise and understanding)	Better training programs, continuous learning, and mentorship.	a) Create a structured BDD training with workshops, practice and learning by doing exercises. b) Setup a mentorship for new team members from experienced BDD practitioners.	a) 'There needs to be more practical training.' Reading about BDD alone will not be enough. But we need to really write scenarios and automate them." - Junior Developer. 'I definitely think a mentorship program would be great.' And stakeholders answer our questions and will guide us if we know experienced people. - Junior Tester. b) Stakeholders recommended creating the knowledge sharing platform and having regular lunch and learn sessions.

Table 20 explains the key stakeholder's suggestions for developing the proposal. The stakeholders discussed all the key findings of three IT projects and all the gaps found in the current state analysis. Stakeholders looked at the steps in the overview of BDD included in the conceptual framework. During this meeting, the participants used the input provided and imagined unique suggestions to build the BDD framework.

5.3 Initial Proposal

This proposal presents a guide for implementing the Behavior-Driven Development (BDD) test framework for Scrum teams working on IT projects.

The Outcome of this thesis is *a guide for implementing the Behavior-Driven Development (BDD) Test Framework for Scrum teams in IT projects.* This proposal consist of 2 parts: (1) the BDD test framework to implement, and (2) the guide for test framework.

5.3.1 Element 1: The BDD test framework, in phases (for which the Guide is created)

This version of BDD approach was co-created with the stakeholders based on (a) the "traditional" BDD approach that was discussed in literature review above, and (b) stakeholder's experiences in the previous 3 projects. This version presents **the BDD framework** based on the views of practitioners, developed in this thesis.

The below table 21 describe the practice-based BDD framework in 5 phases.

Table 21. Description of **the practice-based BDD framework** in 5 phases for implementation by Scrum teams in IT projects.

Phase 1. Prerequisites, preparation and BDD tool selection	This phase provides information about BDD tools that can be used. This is the initial setup for the BDD framework. Different programming languages use different tools that enable users to create BDD scenarios. **BDD tools typically used** (Manekar 2024)**:** Java => JBehave, Cucumber, Gauge C# => Specflow Python => Behave Ruby => Cucumber JavaScript => GaugeJS PHP => Behat
Phase 2. Integration of the BDD approach into Scrum ceremonies	This phase explains how BDD can be integrated into scrum a) When doing backlog refinements, all parties involved agree on what the user stories should accomplish. The importance is given to how to develop it and why to develop it. (Smart 2014; Khomenko 2024) b) Sprint planning: The team and product owner look at the PBIs and break them up into smaller items for the developers to work on. All team members agree to design the application to match the system's behavior. (Smart 2014; Khomenko 2024) c) Daily scrum: Everyone on the scrum team comes to the daily scrum meeting to talk about their efforts, recent accomplishments and to mention any hurdles they have. Their main consideration in carrying out their tasks will be the behavior of the development application. (Smart 2014; Khomenko 2024) d) Sprint review and demo: Members of the scrum team show the outcomes of the sprint to product owners, business analysts and customers. Stakeholders in the business review the work done in each sprint to verify it corresponds to the BDD scenarios created. (Smart 2014; Khomenko 2024) e) Sprint retrospective: The team looks at the things that had success as well as the things that failed. The topic of guiding the development team to prioritize the system's behavior will be discussed. (Smart 2014; Khomenko 2024)
Phase 3. Implementation of BDD	This phase provides implementation details for BDD test scenarios. BDD scenario writing using Gherkin syntax: The business people, developers, and testers are the three amigos of the BDD approach. They

	communicate and collaborate to come up with clear, readable, plain English-like business scenarios. (Gopalakrishnan 2020) **Roles involved in BDD:** (Gopalakrishnan 2020) 1. Business stakeholders such as customers, business analysts, and product owners. 2. Scrum master 3. Developers 4. Testers 5. DevOps engineer **An example scenario for "Add to Cart" functionality:** Given: Given a customer has searched for a product When: When the customer adds a product to the cart Then: Customer should be able to see the shopping cart updated with the product name (Gopalakrishnan 2020)
Phase 4. Integration of BDD in the CI/CD pipeline & Continuous improvement	This phase provides information about integration of BDD into CI/CD pipeline. One can select different CI and CD tools such as Jenkins, Azure or GitLab for project. As soon as code is committed, a build job will automatically be processed on the pipeline in the cloud. New changes are now tested using BDD tests in the cloud pipeline. It checks the application in a quick and reliable way. (Gavandi 2024) Analyzing the behavior of the app at every stage of development increases the code's reliability, reduces the period needed for debugging and ensures that sailable software is deployed faster and more securely. (Sheremeta 2025)
Phase 5. Sharing best practice	This is last phase which embrace sharing of BDD practice. Create a BDD Wiki and document it. Share these materials across different teams to help them with the BDD approach. This sharing of the knowledge may help other scrum teams. (Chaves 2023)

5.3.2 Element 2: The Guide for the practice-based BDD framework for implementation by Scrum teams in IT projects

The Guide supports **the practice-based BDD framework** presented above. It stresses the approach to bring the stakeholders to be more collaborative, so that to build quality in the development of the software and ensure a common understanding of requirements by all the stakeholders.

Table 22. Guide for the Behavior-Driven Development (BDD) test framework for Scrum teams working on IT projects.

Phase 1: To prepare and prerequisites for adopting a BDD
Get acquainted with the BDD framework. The training procedure should include members of scrum teams and business stakeholders at the beginning of the development. Use the following steps for training the team:

a) explain the information about key concepts, ideas, and benefits of a BDD approach to everyone. b) Conduct a workshop can be conducted for developers and testers for writing Gherkin scenarios. Which helps them while writing unit and functional tests. Third, the team can have a BDD expert who can intervene in the process in all the phases of software development. The BDD expert act as catalyst for following BDD best practices. Fourth, practical practice using BDD tool like C#: SpecFlow.
Phase 2: Introducing a BDD approach in Scrum ceremonies a) The BDD needs to be presented in every single ceremony like the backlog brainstorming, sprint planning, daily scrum, sprint review, and sprint retrospective. b) In backlog grooming and sprint planning, the team should perform the activity by keeping in mind the behavior system, which is under development. c) The daily scrum meeting in which the team will update the current sprint work tasks, if any, and related blockers. The behavior of the system in every team member will be in their minds as they interact. d) The sprint tasks should be checked in a review meeting by relevant stakeholders. e) While retrospectively meeting, the team discusses what went well and what went badly in the sprint goal. If there is any deviation or compromise with respect to system behavior, the appropriate action item will be formulated.
Phase 3: Creation of Gherkin scenarios Plain English-like Gherkin test scenarios were written by manual test engineers, and those should be automated by automation testers. The developers can also write the unit tests using Gherkin plain English. Non-technical staff can read and understand these scenarios.
Phase 4: CI/CD pipeline and a continuous improvement in BDD The test code and development code, with the BDD automated scenarios, can be plugged into the CI/CD pipeline that runs a daily execution of the latest build in order to check the behavior of the system under development in relation to acceptance criteria. Various modern tools such as Azure DevOps, GitLab, and Jenkins have these configurations.
Phase 5: Sharing BDD best practices with other teams in the organization a) The BDD best practices can be shared with the teams that are agile in nature, heavily dependent on system behavior, and teamed with people to explore new development methodologies. b) The company wiki pages can be updated with a BDD approach so that everyone in the company can access its information. c) The BDD expert can arrange the knowledge transfer session with other employees of the organization who are willing to learn this methodology. d) The developers and testers who are experts in writing the test script in the form of Gherkin can arrange a mentoring program for other technical staff in the company.

Table 22 shows the Guide for implementing BDD in Scrum teams. It explains the BDD framework as a phased approach, starting from prerequisites to sharing the best practices with other teams in the organization.

5.4 Summary of the Initial Proposal

The proposal provides (a) **the practice-based BDD framework** and (b) **the guide** for implementing **this BDD framework** in Scrum teams of IT. The increased communication and collaboration make sure the software being developed is in sync with the behavior of the system. It meets the unique business requirements of the organization. In other words, these programs result in more successful outcomes of IT projects.

Table 23. Summary of initial proposal.

<table>
<tr><th>Element 1: The practice-based BDD framework for Scrum teams in IT projects</th><th>Element 2: The Guide for implementing the practice-based BDD framework for Scrum teams in IT projects</th></tr>
<tr>
<td>
<table>
<tr><td>Phase 1. Prerequisites, preparation and BDD tool selection</td><td>This phase provides information about BDD tools that can be used. This is the initial setup for the BDD framework. Different programming languages use different tools that enable users to create BDD scenarios.

BDD tools typically used (Manekar 2024):
Java => JBehave, Cucumber, Gauge
C# => Specflow
Python => Behave
Ruby => Cucumber
JavaScript => GaugeJS
PHP => Behat</td></tr>
<tr><td>Phase 2. Integration of the BDD approach into Scrum ceremonies</td><td>This phase explains how BDD can be integrated into scrum
a) When doing backlog refinements, all parties involved agree on what the user stories should accomplish. The importance is given to how to develop it and why to develop it. (Smart 2014; Khomenko 2024)
b) Sprint planning: The team and product owner look at the PBIs and break them up into smaller items for the developers to work on. All team members agree to design the application to match the system's behavior. (Smart 2014; Khomenko 2024)
c) Daily scrum: Everyone on the scrum team comes to the daily scrum meeting to talk about their efforts, recent accomplishments and to mention any hurdles they have. Their main consideration in carrying out their tasks will be the behavior of the development application. (Smart 2014; Khomenko 2024)
d) Sprint review and demo: Members of the scrum team show the outcomes of the sprint to product owners, business analysts and customers. Stakeholders in the business review the work done in each sprint to verify it corresponds to the BDD scenarios created. (Smart 2014; Khomenko 2024)
e) Sprint retrospective: The team looks at the things that had success as well as the things that failed. The topic of guiding the development team to prioritize the system's behavior will be discussed. (Smart 2014; Khomenko 2024)</td></tr>
<tr><td>Phase 3. Implementation of BDD</td><td>This phase provides implementation details for BDD test scenarios.
BDD scenario writing using Gherkin syntax: The business stakeholders, developers, and testers are the three amigos of the BDD approach. They communicate and collaborate to come up with clear, readable, plain English-like business scenarios. (Gopalakrishnan 2020)
Roles involved in BDD: (Gopalakrishnan 2020)
1. Business stakeholders such as customers, business analysts, and product owners.
2. Scrum master
3. Developers
4. Testers
5. DevOps engineer
An example scenario for "Add to Cart" functionality:
Given: Given a customer has searched for a product
When: When the customer adds a product to the cart
Then: Customer should be able to see the shopping cart updated with the product name (Gopalakrishnan 2020)</td></tr>
<tr><td>Phase 4. Integration of BDD in the CI/CD pipeline & Continuous improvement</td><td>This phase provides information about integration of BDD into CI/CD pipeline.
One can select different CI and CD tools such as Jenkins, Azure or GitLab for project. As soon as code is committed, a build job will automatically be processed on the pipeline in the cloud. New changes are now tested using BDD tests in the cloud pipeline. It checks the application in a quick and reliable way. (Gavarcik 2024)
Analyzing the behavior of the app at every stage of development increases the code's reliability, reduces the period needed for debugging and ensures that satable software is deployed faster and more securely. (Bhagwanda 2025)</td></tr>
<tr><td>Phase 5. Sharing best practice</td><td>This is last phase which embrace sharing of BDD practice.
Create a BDD Wiki and document it. Share these materials across different teams to help them with the BDD approach. This sharing of the knowledge may help other scrum teams. (Chaves 2023)</td></tr>
</table>
</td>
<td>
Phase 1: To prepare and prerequisites for adopting a BDD
Get acquainted with the BDD framework. The training procedure should include members of scrum teams and business stakeholders at the beginning of the development.
Use the following steps for training the team:
a) explain the information about key concepts, ideas, and benefits of a BDD approach to everyone.
b) Conduct a workshop can be conducted for developers and testers for writing Gherkin scenarios. Which helps them while writing unit and functional tests. Third, the team can have a BDD expert who can intervene in the process in all the phases of software development. The BDD expert act as catalyst for following BDD best practices. Fourth, practical practice using BDD tool like C#: SpecFlow.

Phase 2: Introducing a BDD approach in Scrum ceremonies
a) The BDD needs to be presented in every single ceremony like the backlog brainstorming, sprint planning, daily scrum, sprint review, and sprint retrospective.
b) In backlog grooming and sprint planning, the team should perform the activity by keeping in mind the behavior system, which is under development.
c) The daily scrum meeting in which the team will update the current sprint work tasks, if any, and related blockers. The behavior of the system in every team member will be in their minds as they interact.
d) The sprint tasks should be checked in a review meeting by relevant stakeholders.
e) While retrospectively meeting, the team discusses what went well and what went badly in the sprint goal. If there is any deviation or compromise with respect to system behavior, the appropriate action item will be formulated.

Phase 3: Creation of Gherkin scenarios
Plain English-like Gherkin test scenarios were written by manual test engineers, and those should be automated by automation testers. The developers can also write the unit tests using Gherkin plain English. Non-technical staff can read and understand these scenarios.

Phase 4: CI/CD pipeline and a continuous improvement in BDD
The test code and development code, with the BDD automated scenarios, can be plugged into the CI/CD pipeline that runs a daily execution of the latest build in order to check the behavior of the system under development in relation to acceptance criteria. Various modern tools such as Azure DevOps, GitLab, and Jenkins have these configurations.

Phase 5: Sharing BDD best practices with other teams in the organization
a) The BDD best practices can be shared with the teams that are agile in nature, heavily dependent on system behavior, and teamed with people to explore new development methodologies.
b) The company wiki pages can be updated with a BDD approach so that everyone in the company can access its information.
c) The BDD expert can arrange the knowledge transfer session with other employees of the organization who are willing to learn this methodology.
d) The developers and testers who are experts in writing the test script in the form of Gherkin can arrange a mentoring program for other technical staff in the company.
</td>
</tr>
</table>

Table 23 shows the summary of the initial proposal, including the practice-based BDD framework the guide for its implementation. It touches many points, such as preparation before BDD adoption, training scrum teams, communication among team members, writing of Gherkin scenarios, CI/CD pipeline integration, and spreading the BDD to other teams in the organization. The next section explains validation of the proposal.

6 Validation of the Proposal

In this section the validation of the initial proposal is performed. At the end of this section, the final proposal is created.

6.1 Overview of the Validation Stage

The main goal of this validation stage is to make a proposal stronger and more effective. Multiple experts are consulted to get more data to make the proposal robust.

First, the expert who is already practicing the BDD in their organization is interviewed. The expert designation is technical architect. Discussion is held on their process and approach to the BDD implementation. The comparison is done between the initial proposal and the approach used in their organization. The collected data is analyzed related to generating automated Gherkin scenarios for corresponding living documentation. This particular approach can further enhance the BDD approach guide in the initial proposal.

Second, the next expert who is working in the advanced cloud data management industry is interviewed. The expert designation is senior development engineer. The talks went on about the BDD approach, which is currently being practiced in their organization. The main focus of the topic in discussion was on cloud integration of BDD scenarios. Integration of scenarios with pipelines such as AWS Pipeline and Azure DevOps Pipelines was discussed. The discussion yielded one interesting enhancement related to the use of cloud-based reporting and monitoring tools.

Third, the last expert who was interviewed is a test automation expert. The expert possessed a good idea on technical areas associated with the development and maintenance of the BDD test framework. The discussion was mainly concentrated around making the framework more robust for future adoptability of new enhancements and system failure. The expert suggested how the customized exception handling mechanism can help the developers and testers to easily identify the root cause for script failure. The expert suggested the use of a logger mechanism in the framework. This particular spot is highly relevant if one wanted to make further improvements to the

original proposal. In the next section, the development of the proposal based on data collection 3 is described.

6.2 Developments to the Proposal (based on Data Collection 3)

The data collection 3 concentrates on finding improvements for the proposal based on validation from experts. The points that are very useful to enhance the effectiveness of the initial proposal are considered. This section is completely dependent on experts who have real-time industry experience in similar technology. By considering these validation points, the original proposal will become more efficient.

The below table 24 provides the expert suggestions for the initial proposal. This table is very useful in terms of making the proposal more robust and relevant.

Table 24. Expert suggestions (findings of Data 3) for the Initial proposal.

	Element 1 of the Initial proposal	*Parts commented in Validation*	*Description of the comment/ feedback by experts (in detail)*	*Development to the Initial proposal*
1	The BDD test framework, in phases	a) In Phase 1, Practical practice using different BDD tools. Java: Cucumber, JBehave, Gauge C#: SpecFlow, Python: Behave, JavaScript: Gauge JS, PHP: Behat.	Apart from practicing BDD tools. Introduce advanced BDD features particular to extending the framework for future enhancement using API interface and configuration.	Take advantage of hooks, custom formatters, user friendly step definitions, method functions bindings and enhancing things for specific frameworks to keep improving your approach over time.
		b) In Phase 4, The test code and development code, with the BDD automated scenarios, can be plugged into the CI/CD pipeline that runs a daily execution of the latest build.	Propose tenant-specific pipeline for validating the dev code and test code instead of testing it on the same pipeline daily. This will reveal more errors in the developing system.	The pipeline will allow developers to execute their code specifically for the concerned tenant.

		c) In Phase 3, The plain English-like Gherkin test scenarios were written by manual test engineers, and those should be automated by automation test engineers.	Use scenario parameterization functionality to use the same step definition for writing multiple Gherkin statements. This will bring reusability and easy extension.	Use scenario outline and parametrization functionality.
	Element 2 of the Initial proposal	*Parts commented in Validation*	*Description of the comment/ feedback by experts (in detail)*	*Development to the Initial proposal*
2	The Guide for the practice-based BDD framework for implementation by Scrum teams in IT projects	a) In Phase 1: To prepare and prerequisites for adopting a BDD	Workshop can be conducted for developer engineers, testers.	The workshop can be conducted for developers, testers, and business stakeholders for writing Gherkin scenarios. This session will enhance the interaction between team members.
		b) In Phase 2: Introducing a BDD approach in Scrum ceremonies	On overall phase 2 introduction of BDD experts in all scrum ceremonies.	The BDD expert can provide help to all scrum team members in the first sprint for practical practice of BDD.
		c) In Phase 5: Sharing BDD best practices with other teams in the organization	Organize hackathons and workshops	The developers and testers who are experts in writing the test script in the form of Gherkin can arrange a mentoring program for other technical staff in the company. Organize hackathons and workshops.

As shown in the above table, 24. The validation comments from experts are collected for each of the elements of the initial proposal. These validation comments are considered

to enhance the effectiveness of the initial proposal in the further section. In the next subsections, the developments to each element of the initial proposal are discussed.

6.2.1 Developments to Element 1 of the Initial Proposal

Notable areas of initial proposal that feedback experts gave addressed include the integration and configuration options for improving extensibility. Suggestions for Phase 1 built on advanced BDD functionalities and API interfaces which would inform subsequent framework improvements. The recommendation for Phase 4 is to migrate into tenant-specific CI/CD pipelines so that as development and test code can be systematically validated against each tenant every day to allow the better identification of errors. To overcome manually written Gherkin scenarios in Phase 3, the team suggested adding scenario outline and parameterization for better automation and extendibility. The revised proposal will endorse these suggestions by the implementation of advanced BDD capabilities, tenant-specific CI/CD pipelines, and Gherkin scenario outline with parameterization for enduring extensibility and more robust testing.

6.2.2 Developments to Elements 2 of the Initial Proposal

This initiative is based around BDD with a view to systematically introducing a phased approach that will enhance communication and close understanding gaps all through the organization. Phase 1 workshops enable collaboration of Gherkin scenarios during the development process by developers, testers, product owners, and business analysts to enhance overall team interaction. In the subsequent phase, Phase 2, BDD is integrated into regular scrum practices, a BDD specialist supports scrum team to bring in best practices during the first sprint. In Phase 5, As an expert, one has the chance to guide junior technical colleagues on Gherkin scripting skills by organizing hackathons and workshops to provide a more collaborative culture.

6.3 Final Proposal

By carefully considering all the validation comments from industry experts, the final proposal is created. The proposal provides (a) the practice-based BDD framework and (b) the guide for implementing this BDD framework in Scrum teams of IT.

(a) the practice-based BDD framework

Table 25. Description of **the practice-based BDD framework** in 5 phases for implementation by Scrum teams in IT projects.

Phase 1. Prerequisites, preparation and BDD tool selection	This phase provides information about BDD tools that can be used. This is the initial setup for the BDD framework. Different programming languages use different tools that enable users to create BDD scenarios. **BDD tools typically used** (Manekar 2024)**:** Java => JBehave, Cucumber, Gauge C# => Specflow Python => Behave Ruby => Cucumber JavaScript => GaugeJS PHP => Behat Utilize the features like hooks, custom formatters, flexible step definitions, bindings, and framework-specific enhancements for long-term extensibility.
Phase 2. Integration of the BDD approach into Scrum ceremonies	This phase explains how BDD can be integrated into scrum a) Backlog refinement: When doing backlog refinements, all parties involved agree on what the user stories should accomplish. The importance is given to how to develop it and why to develop it. (Smart 2014; Khomenko 2024) b) Sprint planning: The team and product owner look at the PBIs and break them up into smaller items for the developers to work on. All team members agree to design the application to match the system's behavior. (Smart 2014; Khomenko 2024) c) Daily scrum: Everyone on the scrum team comes to the daily scrum meeting to talk about their efforts, recent accomplishments and to mention any hurdles they have. Their main consideration in carrying out their tasks will be the behavior of the development application. (Smart 2014; Khomenko 2024) d) Sprint review and demo: Members of the scrum team show the outcomes of the sprint to product owners, business analysts and customers. Stakeholders in the business review the work done in each sprint to verify it corresponds to the BDD scenarios created. (Smart 2014; Khomenko 2024) e) Sprint retrospective: The team looks at the things that had success as well as the things that failed. The topic of guiding the development team to prioritize the system's behavior will be discussed. (Smart 2014; Khomenko 2024)
Phase 3. Implementation of BDD	This phase provides implementation details for BDD test scenarios. BDD scenario writing using Gherkin syntax: The business stakeholders, developers, and testers are the three amigos of the BDD approach. They communicate and collaborate to come up with clear, readable, plain English-like business scenarios. (Gopalakrishnan 2020) **Roles involved in BDD:** (Gopalakrishnan 2020) 1. Business stakeholders such as customers, business analysts, and product owners. 2. Scrum master 3. Developers 4. Testers

	5. DevOps engineer **An example scenario for "Add to Cart" functionality:** Given: Given a customer has searched for a product When: When the customer adds a product to the cart Then: Customer should be able to see the shopping cart updated with the product name (Gopalakrishnan 2020) Use scenario outline and parametrization functionality.
Phase 4. Integration of BDD in the CI/CD pipeline & Continuous improvement	This phase provides information about integration of BDD into CI/CD pipeline. One can select different CI and CD tools such as Jenkins, Azure or GitLab for project. As soon as code is committed, a build job will automatically be processed on the pipeline in the cloud. New changes are now tested using BDD tests in the cloud pipeline. It checks the application in a quick and reliable way. (Gavandi 2024) Analyzing the behavior of the app at every stage of development increases the code's reliability, reduces the period needed for debugging and ensures that sailable software is deployed faster and more securely. (Sheremeta 2025) The CI/CD pipeline can be adjusted to execute the develop and test code against a specific tenant.
Phase 5. Sharing best practice	This is last phase which embrace sharing of BDD practice. Create a BDD Wiki and document it. Share these materials across different teams to help them with the BDD approach. This sharing of the knowledge may help other scrum teams. (Chaves 2023) The developers and testers who are experts in writing the test script in the form of Gherkin can arrange a mentoring program for other technical staff in the company. Organize hackathons and workshops.

The above table 25 describe the practice-based BDD framework in 5 phases for implementation by Scrum teams in IT projects.

(b) the guide for implementing this BDD framework in Scrum teams of IT

Below table provide the guide for BDD test framework which consider the validation comments from experts.

Table 26. Guide for the Behavior-Driven Development (BDD) test framework for Scrum teams working on IT projects.

Phase 1: To prepare and prerequisites for adopting a BDD
Get acquainted with the BDD framework. The training procedure should include members of scrum teams and business stakeholders at the beginning of the development. Use the following steps for training the team: a) explain the information about key concepts, ideas, and benefits of a BDD approach to everyone.

b) Conduct a workshop can be conducted for developers, business stakeholders, and testers for writing Gherkin scenarios. Which helps them while writing unit and functional tests. Third, the team can have a BDD expert who can intervene in the process in all the phases of software development. The BDD expert act as catalyst for following BDD best practices. Fourth, practical practice using BDD tool like C#: SpecFlow.
Phase 2: Introducing a BDD approach in Scrum ceremonies a) The BDD needs to be presented in every single ceremony like the backlog brainstorming, sprint planning, daily scrum, sprint review, and sprint retrospective. b) In backlog grooming and sprint planning, the team should perform the activity by keeping in mind the behavior system, which is under development. c) The daily scrum meeting in which the team will update the current sprint work tasks, if any, and related blockers. The behavior of the system in every team member will be in their minds as they interact. d) The sprint tasks should be checked in a review meeting by relevant stakeholders. e) While retrospectively meeting, the team discusses what went well and what went badly in the sprint goal. If there is any deviation or compromise with respect to system behavior, the appropriate action item will be formulated. f) The BDD expert can help all scrum team members in the first sprint for practical practice of BDD.
Phase 3: Creation of Gherkin scenarios Plain English-like Gherkin test scenarios were written by manual test engineers, and those should be automated by automation testers. The developers can also write the unit tests using Gherkin plain English. Non-technical staff can read and understand these scenarios.
Phase 4: CI/CD pipeline and a continuous improvement in BDD The test code and development code, with the BDD automated scenarios, can be plugged into the CI/CD pipeline that runs a daily execution of the latest build in order to check the behavior of the system under development in relation to acceptance criteria. Various modern tools such as Azure DevOps, GitLab, and Jenkins have these configurations.
Phase 5: Sharing BDD best practices with other teams in the organization a) The BDD best practices can be shared with the teams that are agile in nature, heavily dependent on system behavior, and teamed with people to explore new development methodologies. b) The company wiki pages can be updated with a BDD approach so that everyone in the company can access its information. c) The BDD expert can arrange the knowledge transfer session with other employees of the organization who are willing to learn this methodology. d) The developers and testers who are experts in writing the test script in the form of Gherkin can arrange a mentoring program for other technical staff in the company.

The above table 26 provide the guide for the BDD test framework for Scrum teams working on IT projects. In the next section the recommendations are discussed.

6.4 Recommendations

This proposal supports the adoption of five-phase Behavior-Driven Development (BDD) together with the Scrum framework. The assurance of a strong foundation is found in favoring early broad training and constant expert advice. This is possible by pairing BDD into all Scrum events, thereby ensuring continuous concretion of development with the intended system behaviors. Identifying who implements and automates Gherkin coupled with CI/CD connectivity offers a never-ending supply of quality assurance. What's more, promoting a sense of sharing will give additional teams a better chance to implement the proposed solution successfully. Use of this strategy renders a practical way of improving software quality, communication, and meeting the deliverables to the business needs. The next section concludes this thesis.

7 Conclusion

This section provides the conclusion of this thesis. It contains the executive summary, managerial implications, thesis evaluation, and closing words.

7.1 Executive Summary

The main aim of this thesis is to create a guide for introducing BDD to IT companies that apply the agile methodologies. If the behavior of the system is of the highest priority for application development, then this is a better fit for those projects. It offers an all-around outline and support for the implementation of BDD.

For the creation of this guide, an in-depth analysis of the literature that is already existing as well as a current state analysis is performed. It is based on knowledge of wide literature review.

The current state analysis is performed on three real-time projects on which the real-time work is performed. Also, the relevant stakeholder interview is performed to gather the required information for developing this thesis.

The initial proposal has been developed based on a conceptual framework and state-of-the-art analysis. The initial proposition of the preliminary design of the proposed system has received verification by subject-matter experts. The comments from the expert review are being used to shape the proposal into the final version. In addition, the supervising lecturer offered various important directives aimed at strengthening the efficacy of the process of developing the proposal.

Application of the BDD guide in agile IT companies increases collaboration since there is commonly understood conduct of a system using clear, executable examples. This alignment minimizes any misunderstandings, minimizes rework, and guarantees that the software aligns with what business needs. In addition, due to early testing generated by BDD, it also means improvement in the quality of the software and its development cycles.

The guide also advocates for “living documentation” in which executable specifications are both understanding and automated tests, hence accuracy and relegation of documentation overhead. Through its attention to user-centric behavior, BDD makes sure that the work the team puts in to develop a unit of work does provide real value, resulting in greater satisfaction levels of stakeholders as well as a more efficient agile process. In next section the managerial implications are discussed.

7.2 Managerial Implications (Next Steps and Recommendations toward Implementation)

There are several important considerations and steps to take, from a managerial point of view, to successfully implement the proposed guide on applying the BDD in a given IT company that uses the agile methodologies.

First, managers should support an integrated atmosphere, as a joint definition of the system behavior and BDD training for teams should be invested.

Second, an opportunistic gradual adoption is recommended. Test pilot BDD on appropriate projects, modify the guide to the context of the company, and make sure that necessary tooling is present.

Third, set metrics to measure the impact of BDD (for instance, defect rates) and receive feedback from team on a continuous basis.

Above these, managers should drive long-term learning, regularly update the guide; foster sharing of internal knowledge to see that effectiveness of BDD is sustained.

7.3 Thesis Evaluation

To make a prudent consideration of the prime goal of this thesis, which was to define a practical guide towards the integration of behavior-driven development into an agile IT organization. Some of the existing scholarly work on BDD and agile methodologies was effectively synthesized by the research; therefore, a theoretical framework for the guide was adopted. In addition, the addition of a current state analysis based on the real-time observations of the projects and the stakeholder interviews offered real empirical insights on actual considerations on the grounds for the BDD adoption. The involvement of subject-matter experts throughout the review process also added up to the sharpening of the original proposition.

Lastly, this thesis sets fundamental bases for implementation of BDD in the agile environments. Notwithstanding, future research efforts can build on this work, and they can conduct a more thorough empirical analysis as well as develop a more granular and actionable guide. Overcoming the analyzed shortcomings with the help of extended analysis and specific practical suggestions will make this work even more valuable and applicable for organizations willing to use the advantages of behavior-driven development.

7.4 Closing Words

Ending this discussion on Behavior-Driven Development (BDD) in Scrum teams in IT, particularly when behavior of the system is most important. The integration of BDD promotes increased collaboration, common understanding, and a better vision of delivering business value. BDD effectively reduces the risks of misinterpretations and rework by overcoming the communication gap between the technical and non-technical stakeholders with real-life examples and executable specifications. The focus on “system

behavior is important" just further the importance of BDD in ensuring the delivered software is used as intended from the user's perspective.

Eventually, the effective implementation of BDD by Scrum teams and the ones that are focused on system behavior results in better quality of software, enhanced level of team's efficiency, as well as greater alignment of the different development initiatives with the objectives of the business. Although there might be some initial challenges of implementation, in the long run, BDD is a worthwhile and strategic investment for IT teams that work in a Scrum-oriented environment due to the benefits of clearer communication, less ambiguity, and a common understanding of system behavior.

References

Taherdoost, H. (2022, March). What are Different Research Approaches? Comprehensive Review of Qualitative, Quantitative, and Mixed Method Research, Their Applications, Types, and Limitations. Available from: https://www.researchgate.net/publication/360110506_What_are_Different_Research_Approaches_Comprehensive_Review_of_Qualitative_Quantitative_and_Mixed_Method_Research_Their_Applications_Types_and_Limitations

Smart. (2014, September). BDD in Action. https://learning.oreilly.com/library/view/bdd-in-action/9781617291654/?sso_link=yes&sso_link_from=metropolia-university

Lawrence. (2019). Behavior-Driven Development with Cucumber: Better Collaboration for Better Software. https://metropolia.finna.fi/Record/nelli15.4100000008492493?sid=4999371413

Sutherland. (2014). Scrum : the art of doing twice the work in half the time. Available from: https://metropolia.finna.fi/Record/3amk.89101?sid=4999378583

Martin. (2012). Agile software development: principles, patterns, and practices. Available from: https://metropolia.finna.fi/Record/3amk.253378?sid=4999388993

Shore. (2008). The art of Agile development. Available from: https://metropolia.finna.fi/Record/3amk.105764?sid=4999393361

Khomenko. (2024, Aug 28). Scrum Testing Fundamentals for Your Agile Team's Success. Available from: https://testomat.io/blog/scrum-testing-fundamentals-for-your-agile-teams-success/

Krystalli, Hoffecker, Leith, Kim Wilson. (2021, Sep 22). Taking the Research Experience Seriously: A Framework for Reflexive Applied Research in Development. Available from: https://academic.oup.com/isagsq/article/1/3/ksab022/6374182?login=false

Manekar, A. (2024, May 20). BDD – Be Agile, Create Value & Build Highly Visible Test Automation.[Blog post]. Nitor Infotech. Retrieved from https://www.nitorinfotech.com/blog/bdd-be-agile-create-value-build-highly-visible-test-automation/

Jacon, T. (2023, July 10). Mastering Behavior-Driven Development (BDD). Medium. Available from: https://medium.com/@taynarajacon/mastering-behavior-driven-development-bdd-c69eb3acf3b6

Das, S. (2024, September 18). Benefits of Test Management and BDD in Software Testing Process. Available from: https://www.browserstack.com/guide/benefits-of-test-management-and-bdd

Gavandi, R. (2024, September 15). Behavior Driven Development in Continuous Integration/Continuous Deployment (CI/CD) Pipelines. Available from: https://roshancloudarchitect.me/behavior-driven-development-in-continuous-integration-continuous-deployment-ci-cd-pipelines-05e70ddd5daa

Onukwube. (2024, Aug 22). 10 Best BDD Testing Tools Reviewed For 2024. Available from: https://thectoclub.com/best-tools/

Gopalakrishnan, A. A. (2020, June 5). How to align your team with BDD & automate testing. Available from: https://www.getxray.app/blog/align-your-team-with-bdd-automate-testing

Elyashevskyy, O. (2018, July 5). Does BDD Testing Live Up to its Hype? eleks. Retrieved from https://eleks.com/blog/does-bdd-testing-live-up-to-its-hype/

Sruthy (2025, Jan 30). Specflow Reporting: How to Generate Test Reports and Execute Selective Tests. Retrieved from https://www.softwaretestinghelp.com/specflow-reporting/

Crispin (2019). Agile testing: a practical guide for testers and agile teams. Retrieved from https://metropolia.finna.fi/Record/3amk.241532?sid=4999398542

Deshmukh (2020). Cucumber BDD Made Easy + Automation Framework Design. Retrieved from https://metropolia.finna.fi/Record/nelli15.4100000011551647?sid=4998603269

Chaves, G. S (2023, March 17). Tips for Successfully Implementing Behavior-Driven Development in Your Team. Retrieved from https://gorillalogic.com/blog-and-resources/5-tips-for-successfully-implementing-behavior-driven-development-in-your-team

Tesvan (2024, August 15). Reducing Development Costs with Effective QA Strategies. Retrieved from https://tesvan.com/blog/reducing_development_costs

Sheremeta (2025, Jan 29). Behavior-Driven Development (BDD): Meaning, Benefits & Best Practices. Retrieved from https://testomat.io/blog/mastering-a-robust-bdd-development-workflow/

Creswell, John W., kirjoittaja (2018). Research design: qualitative, quantitative, and mixed methods approaches. Retrieved from https://metropolia.finna.fi/Record/3amk.270727?sid=5024170886

Kananen, Jorma (2013). Design research (applied action research) as thesis research: a practical guide. Retrieved from https://metropolia.finna.fi/Record/3amk.259093?sid=5024387120

Akhtar (2024, Dec 17). Know everything about BDD Framework Retrieved from https://www.browserstack.com/guide/what-is-bdd

WRITTEN STATEMENT
on the use of AI-based tools in this thesis
by ____Avinash Budihal_____, the student of BI Master´s Degree Programme
Thesis title: A Guide for Implementing the Behavior-Driven Development (BDD) Framework in IT Projects

According to the "Guidance for addressing the use of AI-based tools in studies at Metropolia Business School (for written submissions)" from August 2023, I make this statement on the use of AI-based tools in my submitted Master´s thesis.

1) Which AI-bases large language models or other AI-based tools I used
Google Gemini
2) In which parts of the thesis which tools were used, and for which tasks (please make a list)
Section 1, Section 2.
3) What portion of the text was helped with these tools, for each use
Sentences
4) Which prompts were asked, exactly (please indicate the page number in the text where used)
"Please check grammar:"
5) Here, I describe what continues an ethical and reliable use of AI-based tools that I used
As per documents from MBS Guidance
6) Here, I describe how ethically and reliably I used the AI-based tools in my thesis submission
Use of AI tools was minimal for English language grammar checks.

This written statement makes part of my thesis and is done to help in evaluation and assessment.

_____26.05.2025_____
(Data and place)
(Signature)__Avinash Budihal__