Tricentis

The BDD Maturity Model

A Leveled Approach to Scale Behavior Driven Development

CONTENTS

INTRODUCTION

According to Digital.ai's 2020 State of Agile Report, nearly a third of companies are using test-driven development (TDD), with only 19 percent performing the more advanced behavior-driven development (BDD). That's because it's much more difficult to successfully perform BDD — but BDD can offer significant advantages for software quality.

While TDD calls for writing test cases before code is written, it does not specify things like where development should begin, what should be tested, or how those tests should be structured and named — all of which BDD prescribes — and all of which is much more difficult to achieve. But rapid adoption of TDD shows that the foundations are in place for a significant uptick in BDD adoption. Open source tools like Cucumber and SpecFlow are growing in popularity as teams adopt specification languages to improve collaboration between business testers, expert testers and developers, all in the name of creating better software — that meets the needs of its users and the business developing it — faster.

The following model is designed to help you assess where your team is on the journey to BDD maturity. This guide will give you a framework for understanding your team's progress as they gain experience and better define new processes. The model includes five maturity levels, which are based on the adoption patterns we've seen as organizations mature and progress through the BDD learning curve.

By assessing your team's current maturity and outlining a plan to progress to the next level, this guide will help you determine the incremental steps for your team to take to continue making progress towards BDD maturity. Implementing BDD involves significant process changes, both within and outside the QA team, but is well worth the effort. Teams with mature BDD practices also tend to be further along in their DevOps journey — DevOps calls for quality at speed, and by shifting testing left with BDD, these teams are delivering faster without sacrificing quality.

LEVEL ONE : **BDD COLLABORATION BEGINS**

> *"We collaborate, we record that collaboration in some form of specification and then we automate that specification to drive out the implementation."*
>
> *— Cucumber.io*

At level one, your team has started to dip their toes into BDD as a viable development and testing methodology. The team's initial focus should reside in making collaboration work across all team members. This means that they have some level of consistency around the roles of the product owner, developer, and tester in an agile environment. At this stage, the three roles work together to design acceptance criteria by exploring and discovering the needs of the business.

At this level, the team should be able to use their understanding of business goals to question the form (and at times, the validity) of the features the business is asking for. Once the team has mastered this skill, they have paved the way for true alignment between business requirements and the features being developed and tested.

Your team should be familiar with the concept of living documentation, but it's likely that there hasn't been any formal training for writing in the Gherkin syntax and team members are not yet familiar with advanced concepts of data tables and test tags. Even though team is documenting in the Gherkin style for features and test scenarios, there is no standardization of tools, and that's o.k. They may be writing acceptance criteria and test scenarios on note cards or a whiteboard. And manual execution of the scenarios is likely performed outside of any test automation.

Scenario Steps

Given	When	Then
Setup Initial State	Perform Action(s)	Check end state

Phase One Achievements

- √ Collaboration and alignment is improved with early sessions between the product owner, developer, and tester
- √ Example mapping sessions demonstrate visible progress and help gain executive level buy-in
- √ The team is highly focused on making sure the building blocks of the BDD process are in place

Next Steps

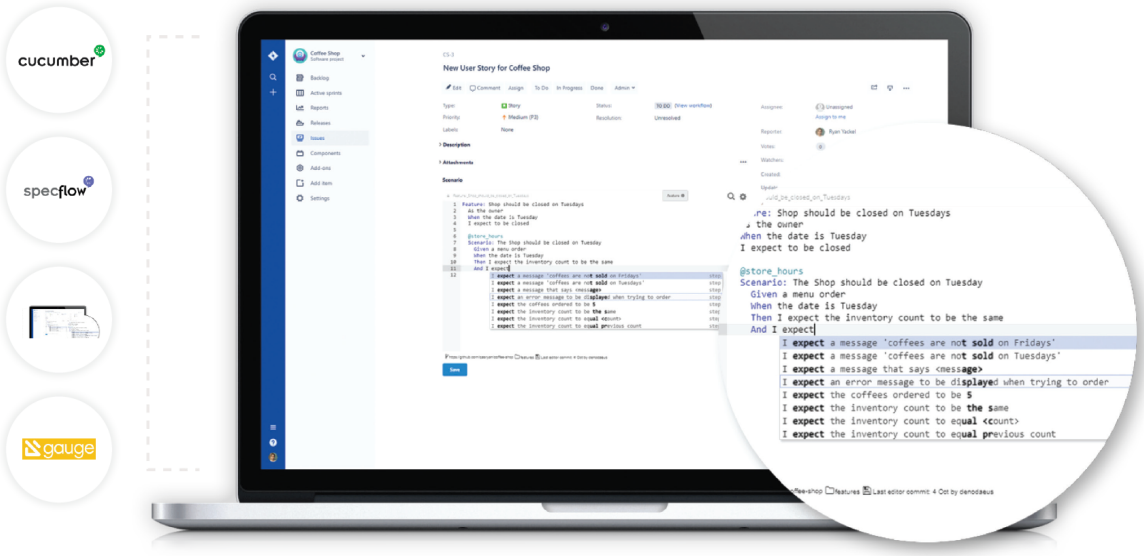
- √ You're writing Gherkin-style scenarios. Now it's time to put the proper tools in place
- √ User stories are clearly related to business outcomes. Now it's time to implement them with code and write automated tests.
- √ Proper use of Gherkin syntax is sporadic. Now that the team has had a chance to practice, it's time to introduce additional Gherkin concepts, like feature, background, and datatables, to increase test case writing efficiency.

LEVEL TWO : **BDD FRAMEWORKS & TOOLS ARE CHOSEN & IMPLEMENTED**

> *"We collaborate, we record that collaboration in some form of specification and then we automate that specification to drive out the implementation."*
>
> *— Cucumber.io*

At level two, your team should evaluate tools and select one that can tie the BDD process together. Prior to this, team's might have written their acceptance criteria and scenarios in notepad or physically on note cards. Tools like SpecFlow, Cucumber, JBehave, and Gauge are all valid candidates your team should discuss when determining which tool is best for them. For simplicity's sake, in this maturity model, we are going to assume that Cucumber has been chosen as the collaboration tool for BDD transformation.

To minimize re-work, teams will adopt a domain specific language required by their particular BDD methodology. For example, if your team has chosen to work with Cucumber, the team will also learn and become familiar with the Gherkin syntax structure during level two. This lays the groundwork for test driven development and test automation.



Phase Two Achievements

- √ Proper tools are in place and connected
- √ You've laid the groundwork for automating test scenarios
- √ The proper syntax is now familiar with all team members

Next Steps

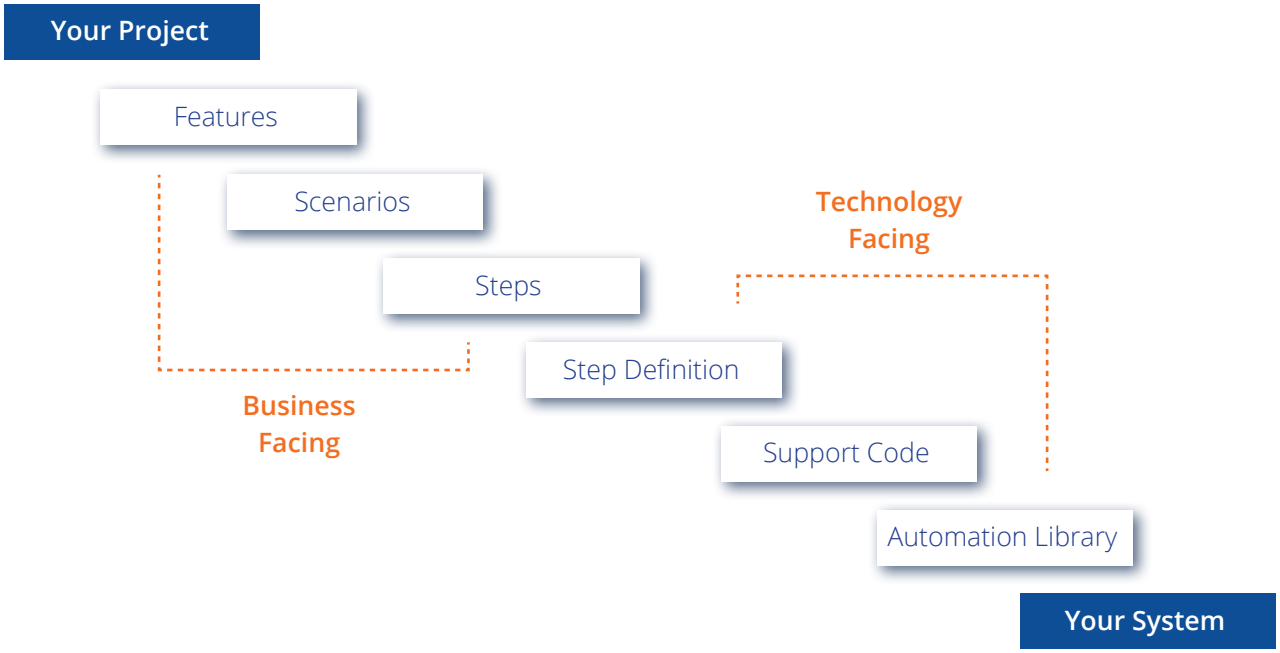
- √ Features need to be versioned with source code for version control and test automation
- √ Reuse of test scenarios should increase.
- √ Features should be connected with enterprise agile planning tools (i.e. Jira Software, Version-One)

LEVEL THREE : SYSTEMS CONNECT FOR DEVELOPMENT & AUTOMATION

Up to this point, all features and scenarios have yet to be connected with source code (i.e. Git) and typical project management tools (i.e. Jira Software). Integration with source code enables developers to implement features and testers to add automated step definitions.

Once source code is connected, development uses the test scenarios to drive the implementation of the feature. Testers can then execute the test scenarios that validate the defined behavior and drive automated deployments.

At this stage, your team also needs to link features with project management tools in order provide traceability for sprint progress. For example, you may have epics and user stories to that are being tracked in typical scrum environments on To Do," "In Progress," and "Done" status'. Communicating your BDD processes back to these key artifacts enables you to justify



Phase Three Achievements

- √ Features are stored with source code for source of truth
- √ Results and status rolled up to overall project plan for acceptance criterias
- √ Features status help progress issues into production members

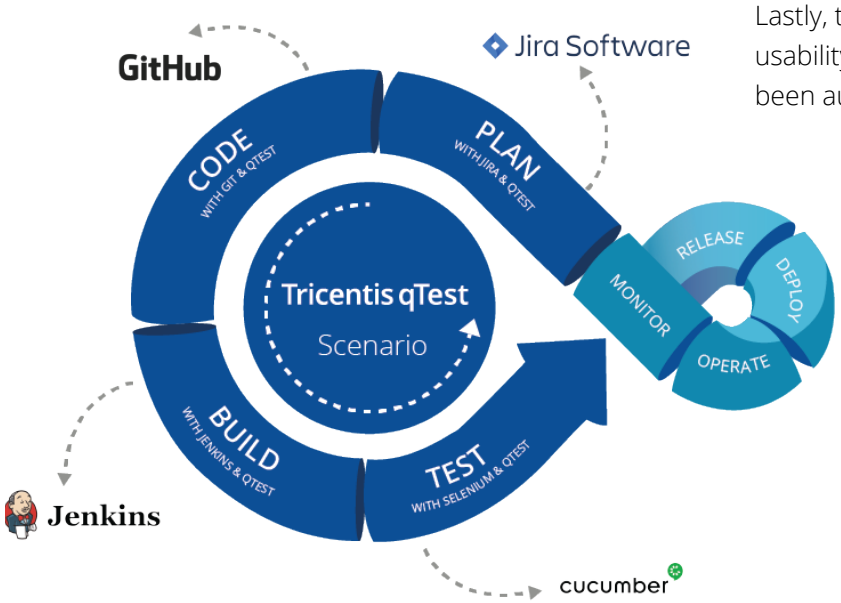
Next Steps

- √ A proper mix of manual vs. automated test scenarios has not been established
- √ Automated test scenarios are not yet tied into continuous integration (Jenkins).
- √ ChatOps notifications have not been enabled for systemic communication.

LEVEL FOUR : **CI & SYSTEMATIC COLLABORATION BECOME STANDARD**

One of the main reasons teams give up on BDD is a lack of automated test scenarios. Teams wil stop short of adding automated step definitions and simply run their scenarios manually after development has implemented the feature. However, to achieve true BDD maturity, automated test scenarios are a must.

At level 4, teams need to start thinking about marrying their development team's test-driven development (TDD) process, if applicable, with BDD processes in testing. Not all teams take on TDD, but if the development team is practicing TDD, it's typically happening at the time application code is being written, and BDD can be a good partner here. However, lots of teams successfully implement BDD and ATDD without ever moving all the way to TDD. If that's the case, after code commits and stay focused on functional automation related to the test scenarios.



Teams taking on test-driven development should be generating automated code stubs once a feature has been created. This practice sends failing step definitions into a Git repository, which mandates attention to the feature. Since the code repository is connected to the CI pipeline, the team cannot progress until they add test automation and ensure the automated tests pass.

Lastly, teams must also include Exploratory and usability testing after their test scenarios have been automated. As mentioned earlier, BDD usually breaks down when the test scenarios are not automated and simply ran through like traditional manual test steps. However, this does not mean that we throw out any test planning that included exploratory and usability testing.

Phase Four Achievements

- √ You've mastered test driven development (TDD as part of your BDD process
- √ Feature file results are reported back through the CI pipeline for full continuous integration
- √ ChatOps workflows instantly alert team members when results fail
- √ A proper mix of exploratory testing has been added to your test cycle

Next Steps

- √ Team-level reporting
- √ Executive-level reporting for BDD progress

LEVEL FIVE : **MEASURING BDD SUCCESS WITH REPORTING**

It seems that every maturity model ends with a focus on reporting, and for good reason. Reporting is the best way to establish a consistent method for measuring your progress against your goals. When reporting on BDD, too many teams stop short at reporting which scenarios passed and which scenarios failed. While that's an important metric that allows teams to rapidly respond to broken builds, it's also important to report on some higher-level metrics that will help you measure and communicate business value. After all, that's what BDD is all about — aligning the business with the engineering team, and delivering business value with each release.

So what does that mean for your reporting processes? The DevOps manager will need traceability reports to understand which features (code) are ready to be pushed to production. Executives will appreciate a high-level overview of progress for each project and team. The executive-level report should include measurements like risk profile, time spend and average number of test scenarios per feature, as well as scenario automation progress.. Not only will this help you keep your executive team in the loop, it will help get you and your team into the mindset of delivering and communicating business value with each release.

Phase Five Achievements

- √ Pipeline analysis is given in live dashboard as a key indication around current state of business value delivery
- √ Phase analysis is given through drill down detail into certain handoffs/areas in the pipeline to analyze why they are speeding up/slowing downs
- √ Retrospective views summarize data for review after the completion of a sprint or major release

Next Steps

- √ Present business case to management for extending BDD adoption
- √ Remove silos and extend BDD process across teams

GETTING STARTED

Now that we have discussed some steps to maturity, how should you get started down the path to BDD? Here are some practical steps to jump-start BDD at your organization.

CREATE A SMALL GROUP

Find team members that want to do something different. That's a simple statement, but the journey to BDD requires a huge mindshift with lots of experimentation. Teams that were the first to move from waterfall to agile might be a good place to start. Strengthen your chances that the group will succeed by including business stakeholders as well as developers and testers.

FOCUS ON THE PROCESS

Before you start "doing BDD," it is critical to understand the additional level of trust that the process requires. If your team is not bought into the process, they will instantly become frustrated. That's because BDD requires a more tightly integrated scrum team than you have ever had in the past. In typical scrum teams, you have mixtures of the scrum master, testers and developers. However, most of the time, team members can quickly create the mini siloes where requirements, code and test are not connected.

PERFORM EXAMPLE MAPPING

Example mapping will quickly identify potential communication breakdowns. Set up a whiteboard, stick post-it notes on a wall, or use an agile planning tool like Jira Software to create concrete examples through continuous conversation. Each example mapping session includes four types of information objects, which build on each other: stories, rules, examples and questions. As your team talks through how the examples will behave in the system, they will naturally form the basis for acceptance tests.

Example mapping will also help you accurately define the project's scope. If a single story ends up having 50 rules, for example, it could be broken into multiple smaller stories. Or if the team has more questions than examples, you may need to reconsider that feature.
Read more about Example Mapping here.

LEARN CORRECT GHERKIN

Before you start "doing BDD," it is critical to understand the additional level of trust that the process requires. If your team is not bought into the process, they will instantly become frustrated. That's because BDD requires a more tightly integrated scrum team than you have ever had in the past. In typical scrum teams, you have mixtures of the scrum master, testers and developers. However, most of the time, team members can quickly create the mini siloes where requirements, code and test are not connected.

ADOPT CUCUMBER AND START GETTING TECHNICAL

Now is the time to start figuring out the technical aspects of your BDD process: choosing your BDD collaboration framework, performing source code integration, setting up continuous integration and tying it all together with reporting. Having the non-technical, foundational elements in place first will ensure you can continue your progress toward BDD maturity — and save your team a lot of frustration.

ABOUT TRICENTIS

Tricentis is the global leader in enterprise continuous testing, widely credited for reinventing software testing for DevOps, cloud, and enterprise applications. The Tricentis AI-powered, continuous testing platform provides a new and fundamentally different way to perform software testing. It addresses both agile development and complex enterprise apps, enabling enterprises to accelerate their digital transformation by dramatically increasing software release speed, reducing costs, and improving software quality. Tricentis has been widely recognized as the leader by all major industry analysts, including being named the leader in Gartner's Magic Quadrant five years in a row. Tricentis has more than 1,800 customers, including the largest brands in the world, such as McKesson, Accenture, Nationwide Insurance, Allianz, Telstra, Moet-Hennessy-Louis Vuitton, and Vodafone.

To learn more, visit https://www.tricentis.com

AMERICAS

2570 W El Camino Real,
Suite 540
Mountain View, CA 94040
Unites States of America
office@tricentis.com
+1-650-383-8329

EMEA

Leonard-Bernstein-Straße 10
1220 Vienna
Austria
office@tricentis.com
+43 1 263 24 09 – 0

APAC

2-12 Foveaux Street
Surry Hills NSW 2010,
Australia
frontdesk.apac@tricentis.com
+61 2 8458 0766