BDD Test Integration Strategy

*John Ferguson Smart – BDD Advisor - john.f.smart@hsbc.com*

# Introduction

Up until present, the CDD project has been focusing on building an automated regression test suite, around stories that were delivered in previous sprints. This approach, which invaluable for regression testing and as a “safety net”, does not help the Scrum teams deliver new stories any faster. With the progressive introduction of BDD practices into the Scrum teams, we aim to provide both regression tests and allow the Scrum teams to benefit from BDD within their sprints, with the aim of helping them deliver working features sooner and more reliably.

As the Scrum teams start to adopt BDD and test-early practices, we will inevitably run into situations where the tests overlap. For example, a new BDD scenario may test a new story, but this story may also affect tests in the regression pack. In addition, because of the nature of the BDD tests, they are likely to be run earlier and against different environments to the regression tests. For these reasons, it is important to define a clear strategy regarding the implementation and execution of the new BDD tests, and how they relate to the existing regression pack.

# Guiding Principles

The strategy we propose is based on a number of simple guiding principles. While the way we deliver on these principles may evolve, we expect the principles themselves to remain relatively stable.

The three guiding principles are:

1. BDD Tests add value by providing feedback about stories under development
2. Regression Tests add value by providing feedback about delivered stories
3. BDD Tests become regression tests once a sprint is completed

## BDD Tests add value by providing feedback about stories under development

The BDD tests provide concrete value to a Scrum team in two ways:

* *Defining the acceptance criteria collaboratively*, during or around the sprint planning phase, gives a clearer focus on the work that needs to be done, and can help highlight uncertainty earlier, making it easier to manage.
* *Automating these acceptance criteria during the sprint*, which allows the team to use the automated acceptance criteria as part of the definition of done. This allows the (fewer) manual testers to perform more in-depth exploratory testing (rather than the traditional scripted variety), which is a more effective way of spotting hard-to-find bugs. The automated tests also give faster feedback to the developers about when a feature works, and when it doesn’t.

The overall effect of these two factors is to allow Scrum teams to work in a more focused way, get faster feedback on their work, and ultimately deliver stories sooner.

For these benefits to be realised, the teams need to be able to automate and run the new BDD tests as soon as possible, without waiting until a story has been completed and delivered. Neither of these benefits can be provided by a regression testing strategy where tests are implemented after the story is completed. The primary beneficiaries of these automated acceptance criteria, actively run during the development of the stories and not after their completion, are the scrum team members. The benefit to the project comes from empowering the scrum teams to deliver stories sooner and more reliably.

## Regression Tests test completed work

The regression test pack is valuable because it tests a large number of features fairly comprehensively. Since the tests relate to stories that are already delivered, they are not of immediate value or direct to the scrum teams. But they are of great value to the project as a whole, since they give broad feedback across the whole project.

## BDD Tests become regression tests once a sprint is completed

Once a story is completed (“done”) and delivered into SIT, the corresponding BDD acceptance tests become part of the regression test pack.

These principles are illustrated in Figure 1.

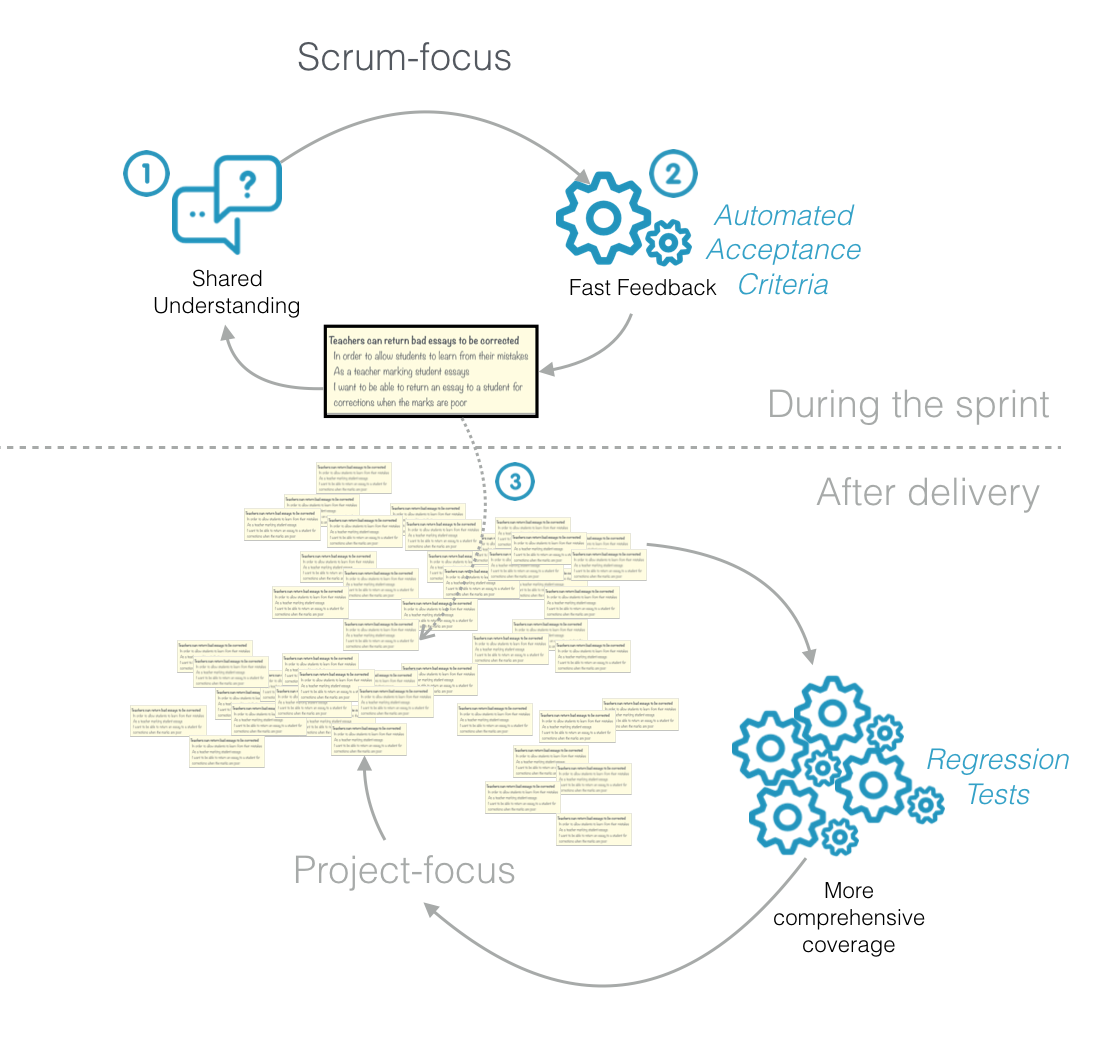


Figure : The relationship between BDD tests and regression tests

# Recommended guidelines

Based on these principles, we can define a number of general guidelines that should help the BDD tests and the Regression Pack live in relative harmony.

These guidelines are:

* BDD Tests should be run against DIT or similar “early” environments
* BDD Tests should be isolated from regression tests until the story is deployed to SIT
* BDD Tests should not interfere with other tests running on the same environment
* BDD Tests become regression tests once a sprint is completed
* Scrum-related BDD regression tests will be run in the existing test jobs wherever possible/appropriate.
* Scrum teams should be able to easily see both the tests for the current sprint, and the full set of test they have worked on.
* BDD Tests should add new scenarios by building on the existing tasks
* If you break it, you fix it

## BDD Tests should be run against DIT or similar “early” environments

Scrum teams benefit from BDD automation when they get feedback about the stories in progress during the sprint, not once the stories have been completed. This means running the acceptance tests against DIT (on Jenkins) rather than against SIT.

Developers may which to participate in automating the acceptance criteria (after appropriate training in Serenity and the CDD test architecture). This will be done initially by pairing with or being coached/supported by specialised automation testers. We encourage this as it tends to accelerate both automation and development, and encourages developers to take a more active ownership in code quality.

The acceptance tests for a sprint (say 5-6 stories) would take on average between 30 and 60 minutes to complete (depending on the number and nature of the stories). This may be a bit long to run before each commit (unless the tests could be run in the background leaving the developer free to work on something else in the meantime). On the other hand, it would be quite feasible to run these as part of the build pipeline whenever the changes are committed. Each scrum team should agree on the schedule that would work best for them, with a minimum of nightly builds against DIT.

## BDD Tests should be isolated from regression tests until the story is deployed to SIT

During a sprint, the BDD automated tests run against stories that are not yet deployed to SIT. Each Scrum team will have a dedicated Jenkins build job that runs only the tests for that scrum team and for the current sprint. These jobs will create dashboards to help the Scrum teams keep track of progress and issues during the sprint.

This means that the tests run in this job may fail or not behave correctly when run against SIT.

There are two main ways that we can prevent the BDD tests from adversely affecting the regression pack:

1. The BDD teams can send their changes to *develop* (using the current Pull-Request based process), and using tags to avoid running the BDD scenarios for the current sprints;
2. The BDD teams can use separate branches, a branch for each scrum team, merging from *develop* at least daily but only merging to *develop* once the stories are deployed to SIT.

The first approach would work will if the BDD scenarios did not affect existing tests at all. It allows for smaller, more frequent commits and easier merges. However, there are cases where the requirements for a new story do affect existing tests, and these need to be updated accordingly (or replaced by the BDD scenarios if the original requirement is superseded).

The second approach isolates changes to existing tests in a branch for each scrum team. This approach is illustrated in Figure 2.

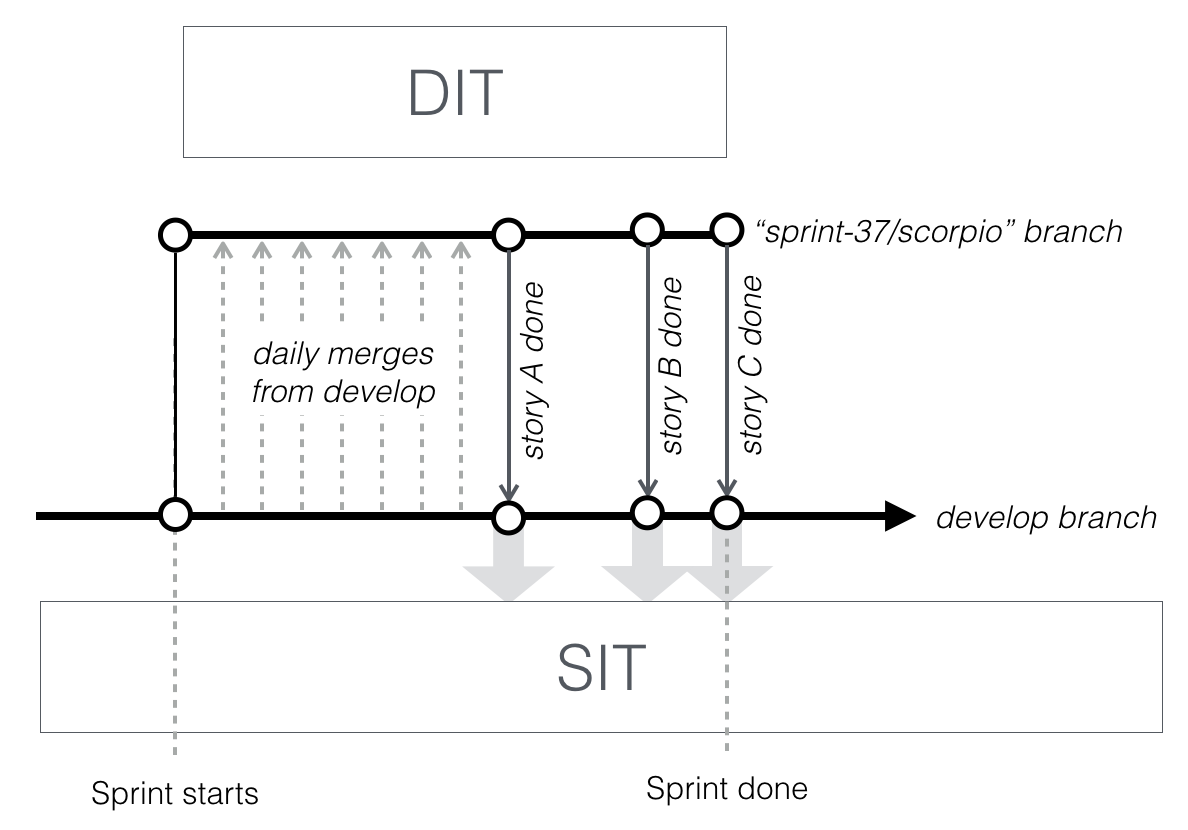


Figure : Scrum-specific branching strategy

Our recommendation would be to use the first approach (working directly against *develop*) as long as no existing tests are affected by the stories in the sprint, and using the second approach (using a separate branch) only if there are existing tests that need to be updated.

BDD Tests should not interfere with other tests running on the same environment

If several of the scrum-specific build jobs are running against the same environment, care must be taken to ensure the profiles do not conflict. This can be done with a scrum-specific set of profiles, or Scrum-specific DIT environments. Note that when they go into the regression pack (see below), they will also need to play nicely with the other test jobs, which generally means running as part of one of the existing regression build jobs.

## BDD Tests become regression tests once a sprint is completed

When a story is marked as done (including having passing automated acceptance tests), the BDD scenarios associated with this story become part of the Regression pack. The simplest way to include the BDD scenarios in the regression test pack is to mark them as done (using the @done tag).

The regression pack is made up of many different build jobs, configured to work with different profiles so that they can run in parallel. For new BDD scenarios, we will need to determine which job corresponds to the profiles required for the scenarios. If no existing jobs have all of the necessary profiles, we can either (following discussion with the team):

* Add new profiles to one of the existing job configurations (e.g. xLOB), or
* Create a new build job and profile configuration for these profiles

Once the build job is decided, we simply add the corresponding tag to the scenario or feature. Each scenario and/or feature will also be tagged by Scrum team (e.g. @scrum=scorpio) and by sprint (e.g @sprint=33) rather than by type-specific build job.

### Test Execution

The new BDD tests will be executed via dedicated JUnit/Cucumber test runner classes. Since they will not initially be tagged by job, they will not be accidentally run with the original Regression Pack tests, but only via dedicated Scrum build jobs (see below).

### Profiles

Some of the new tests may need profiles, or combinations of profiles, that do not currently exist in the profile configuration files. Each Scrum team will be responsible for identifying, collecting and defining any new profiles they need for these tests. In each sprint, team members will, as required, collaborate with the Scrum manual testers to identify/obtain the necessary profiles for the new stories.

### Aggregate reports

Sprint-specific and Scrum-specific aggregate reports will be made available from the results of the nightly SIT build jobs.

### Build jobs

Overall, each Scrum will have two BDD-specific jobs running on Jenkins:

* The BDD scenarios for the current sprint, run against DIT (on commit or nightly)
* The BDD scenarios for all of the sprints so far, run against DIT (excluding the tests in the Regression Pack) (on commit or nightly)

### Triage

Triage for the Scrum-specific jobs will be done in the same way as the other build jobs, with collaboration via SameTime. We expect that it will largely be the responsibility of the Pune-based BDD specialists to maintain these tests.

### Cohabitation

Once the new tests are integrated into the regression test suite, they will use the same underlying Task classes as the existing tests (see below), and will effectively be just another form of regression test. The test automation team will also be briefed on how the Cucumber tests work.

## BDD Tests should add new scenarios by building on the existing tasks

Two key principles behind the CDD test architecture are the **Single Responsibility Principle** and the **Open-Closed Principle**. In practice, this means that each Screenplay task and action should have a single, focused job (Single Responsibility Principle), and that new tests should either use the existing tasks unchanged, or create new tasks that do something new (Open-closed principle). Modifying a task class should only happen if the underlying requirements or UI have changed, not because a new test is being added.

This makes the test framework quite resilient, and makes it easier to know what evidence of impact need be provided when changes are made.

The same principle applies for the new BDD tests. The BDD tests (which may be written as Cucumber scenarios when the Scrum teams prefer) use the existing tasks or create new ones, just like the regression tests. There is just a thin layer of “glue code” to bind the classes to the Cucumber feature files. When a new scenario is added, just like for the regression/backlog tests, the priority should be to reuse existing tasks (without modifying them) before creating new tasks (see Figure 3).

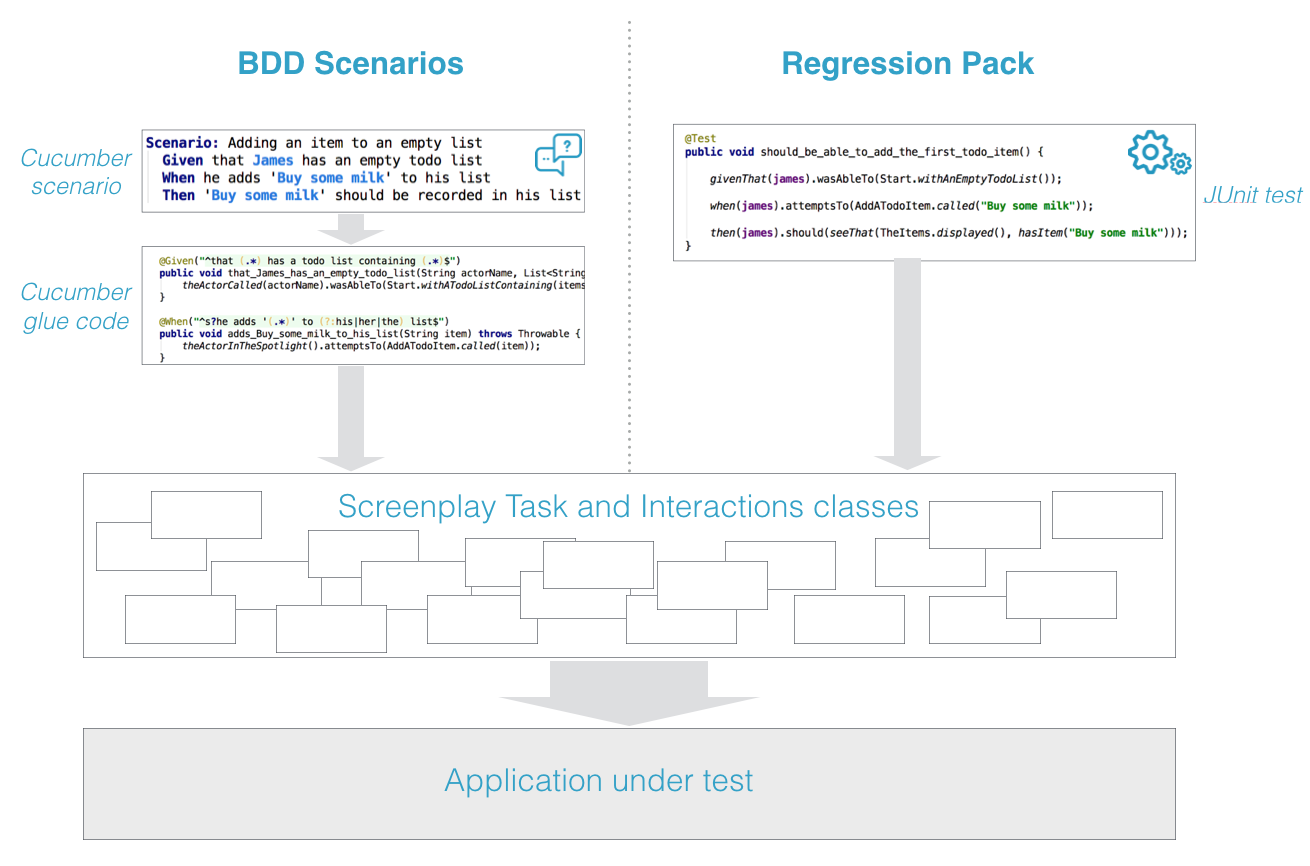


Figure : BDD scenarios and Regression Pack tests both use the same underlying components

## If you break it, you fix it

The scope of a new BDD scenario also includes updating any existing tests that are affected by the new or changed requirement. This can happen in two ways:

* **The new scenario replaces an existing test**. Existing tests are superseded by new requirements and the corresponding BDD scenarios provides equivalent or more coverage than the previous tests. In this case, the existing tests can be decommissioned and replaced by the BDD tests.
* **The new scenario affects existing tests**. For example, the new scenario involves a mandatory field that is also used by other scenarios. In this case, the existing Regression Pack test cases will need to be updated and merged into *develop* at the same time as the new scenario.