Whitepaper:
Supporting ATDD/BDD with SpecLog, SpecFlow and Team Foundation Server

Christian Hassa, May 4th 2011 (ch (at) techtalk (dot) at, Twitter:chr99ha) – v1.0

This whitepaper provides a brief overview of a possible requirements and development flow supported by SpecLog (www.speclog.net), SpecFlow (www.specflow.org) and Microsoft Team Foundation Server, in the context of agile software development and ATDD/BDD (acceptance test driven development/behavior driven development).

Note: The flow described is just a proposal; adjust as needed. None of the tools mentioned in this document should dictate a specific flow for your team. The proposed flow is based on many ideas in the agile community, including but not limited to the following:

- Jeff Patton: story maps
- Gojko Adzic: “Specification by Example”
- Brian Marick, Lisa Crispin: agile testing quadrants
- Mike Cohn: the test automation pyramid
Deriving scope from goals: maintaining and pruning the product backlog with story maps

There are many different methods in agile development for maintaining and pruning product backlogs. Many methods recommend visualizing and collaborating on the product backlog with story maps, which provide a better overview than simple product backlog lists. You can create story maps to build a release map, analyze journeys through the system or prepare stories for implementation. Jeff Patton came up with the original idea of story maps.

SpecLog introduces the concept of workspaces to the product backlog, where you can map requirements in story maps. Requirements can be arranged in multiple workspaces and you can visualize dependencies and refinements of requirements on the map:

Adding details just-in-time: specification by example

User stories selected for implementation need to be refined with details and made testable. The team needs to know when the story is accepted as “done”. “Done” criteria that are specific to a story are called “acceptance criteria”. Using examples is a good practice for describing acceptance criteria. Examples can be given for various aspects of the story such as the user interface, data or business logic. We recommend involving all team members (developers, testers, UI designers) as well as the business stakeholders to help collect examples. Gojko Adzic has written extensive papers and books on specification by example.

SpecLog supports collecting acceptance criteria and examples for existing user stories. You can insert UI scribbles and screen shots as well as data and business rule examples. A SpecLog server that
synchronizes requirements across multiple clients is available to support collaboration. You can also flexibly tag requirements to support your example collection flow:

**Automating without changing the specification: formalizing to Gherkin**

When stories are prepared for implementation, the team formalizes the specified acceptance criteria into business readable, automated Gherkin scenarios that serve as an automated acceptance test for the user story.

SpecLog can export the collected acceptance criteria in the Gherkin file feature format:
The team can further refine acceptance criteria for automation using SpecFlow and Visual Studio (or any other development stack compatible with Gherkin).

SpecFlow provides a powerful implementation for executing and even debugging Gherkin in the .NET runtime (including Silverlight and Windows Phone 7). SpecFlow integrates with Visual Studio and MonoDevelop to offer a rich editing experience including autocomplete for steps, go-to-step definition from Gherkin and automated table formatting.

The refined acceptance criteria are captured in .feature files that are source code artifacts of the solution. Refinement starts with the information captured in SpecLog:

![Feature File Example](image)

The originally defined acceptance criteria are then extended with examples that can be automated. The important part is that the acceptance criteria are not changed and remain business readable when formalized for automation.
Once the .feature file for a user story is hosted in a source code repository, it can be linked to the original SpecLog requirement user story, so that business stakeholders can review the formalized acceptance criteria. SpecLog allows .feature files to be linked to various sources like a TFS or GIT source code repository or a central source code drop location (to support any other kind of source code repositories).

Reviewing the formalization is an important step to keep business stakeholders involved and keep automated scenarios business readable. SpecLog provides both the originally captured planning notes and the formalized acceptance criteria from the linked .feature files. Stakeholders can compare the original notes and assert that they match the formalized acceptance criteria:
Visualizing implementation flow with TFS

Once acceptance criteria have been formalized and reviewed by all stakeholders, the team can start with development.

If you are using TFS, you can synchronize SpecLog requirements to TFS work items, so that the team can capture tasks in TFS associated with the user stories of the Sprint backlog:

SpecLog currently supports synchronizing requirements to product backlog work items for the following TFS 2010 templates:

- Microsoft Visual Studio Scrum 1.0
- MSF for Agile Software Development 5.0
Outside-in development: extending from TDD to ATDD

Based on the acceptance criteria, the team starts to implement the user stories with the formalized features. They start by automating the first acceptance criterion of the first user story, which fails to pass as the system is not supporting it yet (the test returns red).

Automating with Gherkin and SpecFlow is beyond the scope of this article. You can automate different architectures such as Web clients, rich clients and Windows Phone 7 applications using SpecFlow and optional other UI automation libraries (such as Selenium, Watin, White or CodedUI).

After automating the first acceptance criterion, the team extends the system so that it fulfills the first acceptance criterion (e.g. the test turns green). They continue with the next acceptance criteria of the story until the whole story is done. Depending on technical restrictions, the team tries to work on as few stories in parallel as possible. This practice extends the already known test-driven development cycle to form an acceptance test-driven development cycle:

In addition to the usual burn down diagram, stakeholders of the project can track work-in-progress and completed work through automated reports generated from the SpecFlow acceptance tests:
Testing

While describing agile testing strategies is not the purpose of this article, the following points regarding testing are important to note:

- Automated acceptance tests are not a replacement for a proper testing strategy. They can be an important aspect for testing, although their primary purpose is to serve as executable specifications. Grigori Melnik and Robert C. Martin describe this in their paper Tests and Requirements, Requirements and Tests: A Möbius Strip.

- Testers play an important role in defining acceptance criteria. They usually discover exceptional cases and find edge cases to be considered in the system that can be illustrated with examples and refined to executable specifications. This activity lies in the business facing/supporting the team quadrant of the agile testing quadrants as described by Brian Marick and Lisa Crispin.

- Using the Gherkin sentences defined, testers can easily automate additional scenarios for a given user story without needing to deal with automation code.

- Having all acceptance criteria automated as an executable specification leaves the tester more time for other testing tasks that are harder to automate, as described by Mike Cohn in his article on the test automation pyramid.
Merging completed work with the feature tree: building a living documentation system

Since completed acceptance criteria are continuously validated through automated tests, they become the living documentation of the current system. They are business readable and always up to date, offering a unique advantage compared to manually maintained system documentations or automated testing suites.

However, user stories do not provide a good organizational structure for the living documentation. They describe a change from a current state of the system to a desired new state, focusing on a specific behavioral aspect. They may therefore only make sense at the time they are written, while the system can evolve in the meantime. Also, acceptance criteria of user stories may contradict acceptance criteria of the existing system.

It is therefore important to merge acceptance criteria of user stories that are done into a living documentation tree that provides a functional overview of all current aspects of the system. Individual acceptance tests of the completed user stories are moved from the current sprint test suite to the feature tree test suite.
In SpecLog, you can build story maps to represent such a feature tree that describes the existing system through business readable, automated acceptance tests:
Automated acceptance tests can be attached to individual features in the tree. They provide a business readable detail description of the current system that is always up-to-date: