Git: Working on same story in parallel

|  |  |
| --- | --- |
| Author: | Richard Stevenson |
| Document Name: | Git: Working on same story in parallel |
| Document Location: | * [\\lukpc210\Projects\Product Projects\Operations\Vitic 2017.1\Project Documents\POC\VXRW-359-git-working-on-same-story-in-parallel](file:///\\lukpc210\Projects\Product%20Projects\Operations\Vitic%202017.1\Project%20Documents\POC\VXRW-359-git-working-on-a-story-with-mutiple-sub-tasks) |
|  |
| Date Created: | 27/03/2017 |
| Date Last Modified: | 28/03/2017 |
| Status: | Draft |
| Confidentiality: | <Select/enter confidentiality status> |
| Circulation: | <Detail circulation list> |
| Document Version: | 0.2 |

Contents

[1. Overview 2](#_Toc478459011)

[2. What should the strategy achieve? 3](#_Toc478459012)

[2.1. Multiple people should be able to work on a story at once 3](#_Toc478459013)

[2.2. Individuals who are working on a task must be able to work in isolation 3](#_Toc478459014)

[2.3. Merging our changes into the main line of development must occur in one single merge 3](#_Toc478459015)

[3. A definition of a story 4](#_Toc478459016)

[4. A branching strategy for our story 5](#_Toc478459017)

# Overview

This document has been written as part of a number of documents to identify the best practises which should be adopted along with Git in order to implement it successfully. Part of implementing Git is learning how it works and the useful commands. This will not be covered in depth in this document.

This document specifically addresses how we should work when we have a story that contains multiple strands of work which should be done in parallel.

# What should the strategy achieve?

The following are things to consider when coming up with a strategy to enable a story to be completed:

# Multiple people should be able to work on a story at once

A story may have several sub tasks that cross multiple disciplines in a team. Ideally parts of the story that exist in different elements, say front end, back end, api tests and acceptance tests should be worked on individually.

# Individuals who are working on a task must be able to work in isolation

Work should be carried out so that it doesn’t affect anyone else’s code. As an example a team member responsible for writing acceptance tests for a web based application should be able to write those independently to the person writing the front end code, as there will be a standard naming convention for html elements and it will be defined what each element should do. In this case it is perfectly possible that all the acceptance tests are finished before the front end is fully developed.

# Merging our changes into the main line of development must occur in one single merge

When we merge back into the main line of development this should be done as a single commit. This makes our history easier to understand and ensures that a story is a whole, complete unit. Doing this means that we don’t get half-finished items into the main line of development, all items have the right level of testing and the main line remains a potentially shippable product. It also mean that when we look back at the history we can see all the code for that change in one, allowing us to reason more easily about what the purpose of that individual change was.

# A definition of a story

When we are working on a story it is normal for there to be a number of different strands of work that must be done by different people. Consider the following story “JIRA-1 – User Story”, that is split up into several sub-tasks:

* JIRA-2 – Server side changes
* JIRA-3 – Front end changes
* JIRA-4 – API tests for server side changes
* JIRA-5 – Acceptance tests for end to end function

Working within scrum it is desirable that each of the subtasks can be worked on separately by different member of the team. To avoid the “mini waterfall” scenario where tests are pushed back to the end of the sprint it is also good if the person writing tests can do so as soon as possible, this will inform the person who is writing the code as to what they need to do to fulfil the acceptance criteria of the user story.

Clearly if a story simply contains subtasks to be carried out by one person, or it’s a simple bug fix that only requires some extra unit testing, then it is possible to simply have a single branch for the whole story.

# A branching strategy for our story

Let us assume that for our story, JIRA-1, four different members of the team will be working on the four different sub tasks. The member of the team who first picks the Jira story should assign that story to themselves, they become the “co-ordinator” of the story. They should create the story branch from the main line of development and a branch from the story branch for their own sub task. Then other people who work on the other sub tasks can create their own sub task branch from the story branch when they start to work on their own sub task.

It would then be possible for the git tree to look something like this.

JIRA-5

JIRA-1

JIRA-3

JIRA-4

Main Line of Development

Last commit on main line

JIRA-2

Now each of the people who are working on a sub task is able to do work as they please and commit to their individual branch.

4 C1

2 C1

JIRA-1

JIRA-2

JIRA-3

JIRA-4

JIRA-5

2 C2

2 C3

3 C1

5 C1

5 C2

When each individual is happy that they have done their work then they should raise a pull request which will merge their code into the story branch, this obviously has to be reviewed by another person and pass all usual coverage and static analysis criteria. At this point we have done no squashing commits and no rebasing of any branch involved in the story.

5 C2

JIRA-1

2 C1

Main Line of Development

2 C3

4 C1

3 C1

5 C1

2 C2

Task 4 Merge

Task 3 Merge

Task 2 Merge

Task 5 Merge

In an ideal world all the tests on this story branch will pass, it might be that when everything is merged together then things might not work so neatly. At this point it is up to the group who have worked on it to determine how to fix the issue and who is going to fix it. Creating another branch from the story branch and applying fixes to this in the usual manner, then when all the tests pass merge this change into the story ticket branch.

JIRA-1

2 C3

2 C2

4 C1

3 C1

5 C2

5 C1

2 C1

JIRA-1-integration

INT C1

Now that the story, in isolation, has been finished our branch will look like this

Main Line of Development

JIRA-1

2 C3

2 C2

4 C1

3 C1

5 C2

5 C1

2 C1

INT C1

Now, the co-ordinator for the story must squash the changes of the story into a single commit.

Main Line of Development

JIRA-1

Last commit on main line

JIRA-6

JIRA-10

Squashed commits

Then rebase any changes from the main line so that all tests can be run with all code, here the co-ordinator may need help from those who have done the work to merge changes in.

Main Line of Development

JIRA-10

JIRA-6

JIRA-1

Now merge the story into the main line of development. It is worth noting that if there have been any conflicts while rebasing then there is an opportunity for things to slip through a code review. As the co-ordinator needs to create a pull request at this point they can indicate if any changes have had to be made and how extensive they are. We will rely on people’s judgement to determine how to proceed.

JIRA-1

JIRA-10

JIRA-6

Main Line of Development

This approach means that each story is one discrete unit, no story will be committed that will break the main line of development, so it will always be potentially shippable product.