TFS 2010 New Agile Template Release Management Guidance

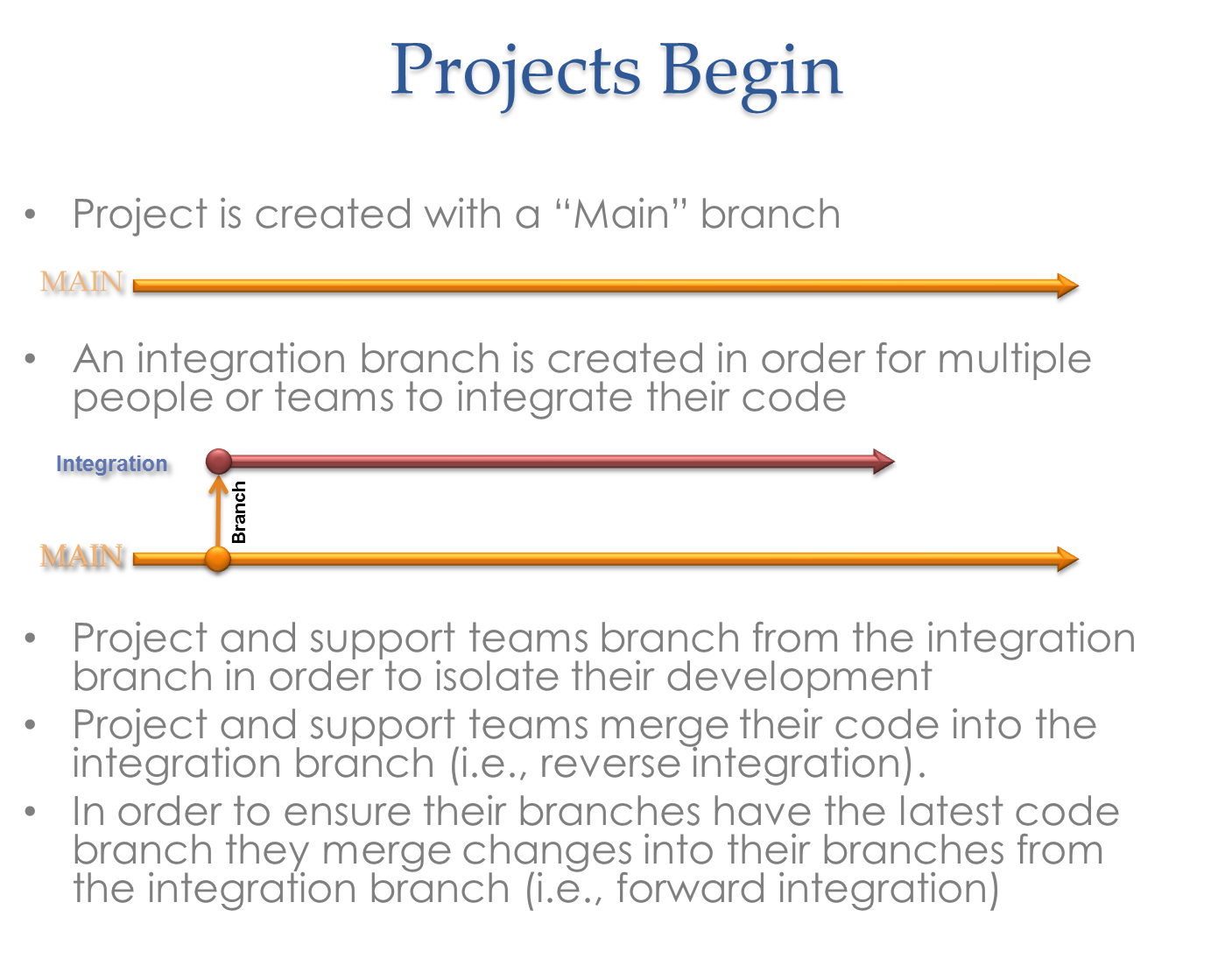
# Introduction

The upgrade to TFS 2010 and move to the new TFS 2010 Agile template and Microsoft Test Manager has resulted in numerous improvements. As part of these changes, we have elected to give development more control and accountability in regards to the management of source code. This document describes the changes to the branching, release and DB code review processes.

# Branching & Deployment

For projects using the Agile template and the new branching structure, Release Management will no longer be branching based on change sets. Release Management will be merging the entire branch into the release branch upon request from development. Because RM will be taking the whole branch, it is critical that a lead developer be responsible for keeping those branches clean. Code that is not ready for release should never be moved into the Integration or Main branches. Once code is those branches, it is on the release train – on its way to production.

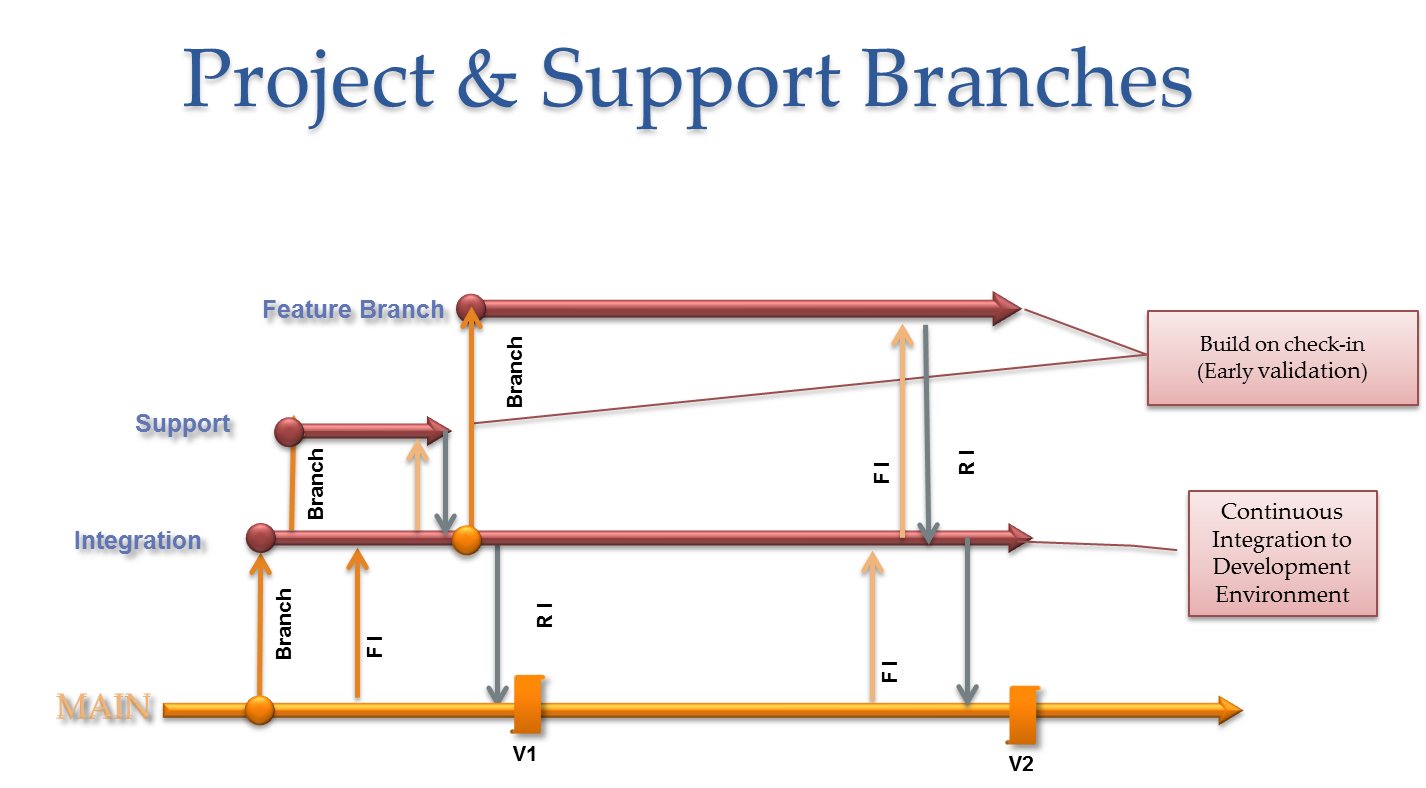
In order to keep the Integration and Main branches clean, projects and support branches will be used for nearly all development. People should not be checking directly into Integration or Main. Outlined below are the various branches and their usage.



The branch types are as follows:

* Project Branches – branches used for projects and new features
* Support (Maintenance) Branch - branches for bug fixes and minor enhancements
* Integration Branch – branches where project and support changes are merged together in preparation for release (Release Candidate)
* Main Branch (Pre-Release) – the branch used as the hand-off from development to RM. Development merges changes into this branch in preparation for the production release. If code gets into this branch, it is going to production.
* Release – branch that is cut from main in preparation for release. It represents what is going into production
* Hot Fix – In cases where new code has been moved into main and an issue needs to be resolved either before or after production, a hotfix branch is created. Changes made in Hot Fix need to be merged back up to Main and Integration

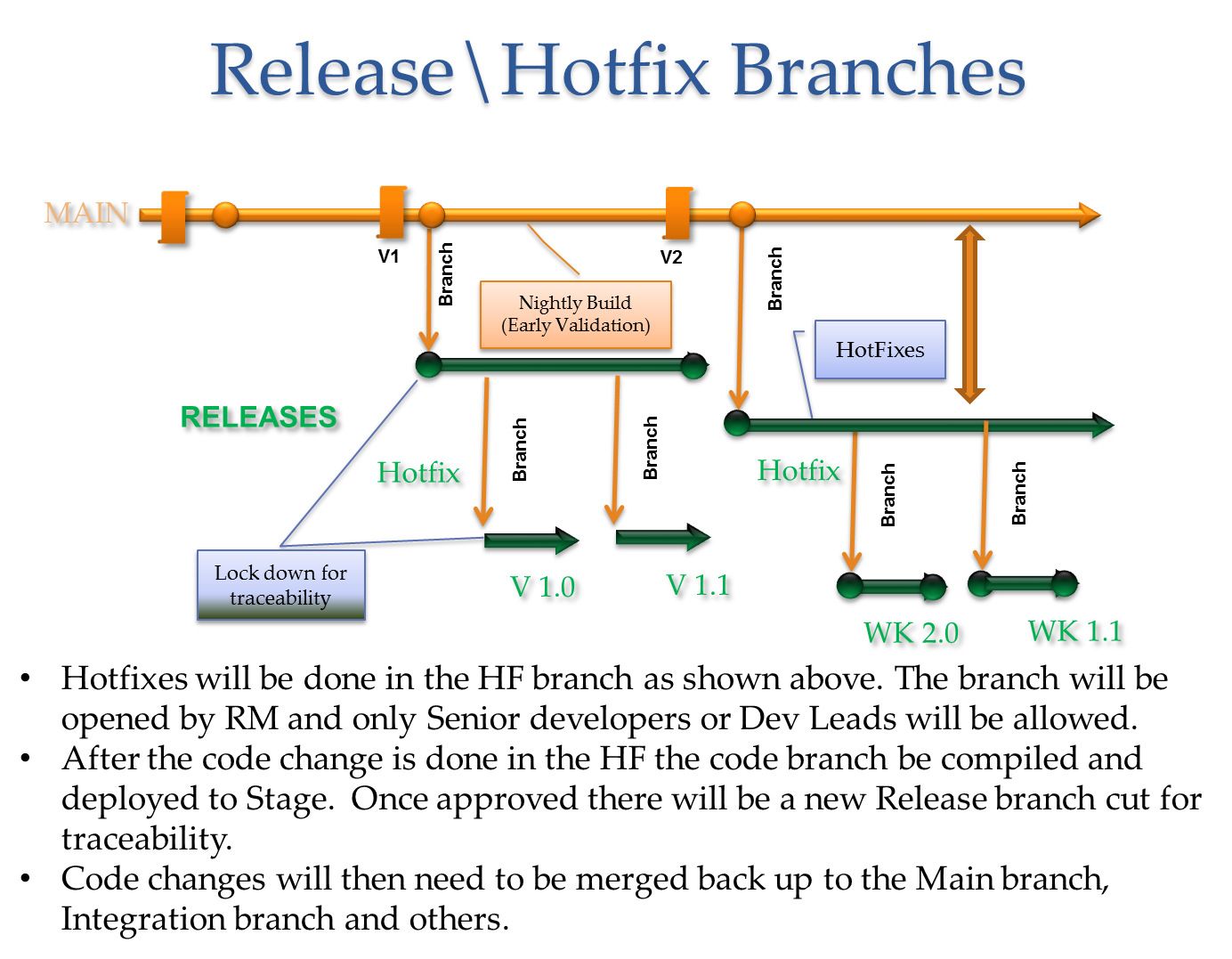
Development and maintenance teams work in project and support branches. All check-ins must be associated with TFS Tasks which associated with User Stories. When development is complete and validated in the project and support branches, code is then merged into the integration branch. The various project and support changes merge into the integration branch in preparation for release. Once changes are validated in the development environment against a build of the integration build, the changes in the integration branch get merged into the Main branch. A development lead is responsible for the Integration and main branches to ensure it remains clean and all conflicts are resolved.



Once the Main branch is validated and identified as ready for release, RM is notified by creating a TFS Task with an Activity set to “Non-production Deployment” and a description of the request. In this case the request should be to deploy the build from main into the test environment. Note that there would not be a re-build of the main branch, just a deployment. This task must be associated with the “deployment user story”. RM deploys the build from Main into the Test environment based on the details in the request.

Once the build from Main is validated, the scrum team requests that RM create the release branch, build and deploy to stage. The request is done by creating a new TFS Task with an activity of Non-production deployment. RM creates the release branch based on the deployment task and deploys to Stage. If code needs to be merged to release and redeployed after the initial deployment, the deployment task must be re-opened. (Note: During transition to the new process, the requestor should send RM an email upon creation and re-activation of the deployment task).

In cases where issues are identified late in the process or after release, RM will create a Hot Fix branch at the request of development and open it up for specific developers to make the change. This would be done based on a conversation between RM and the development team. After the changes are completed in the hotfix branch and validated, it is critical that the changes get merged into the other branches (main, integration, etc.).



# Deployment User Stories

When a team determines that an application is going to be released to production, a “Deployment User Story” (DUS) must be created. Ideally, the Deployment User Story is created by the Scrum Master as part of the sprint planning at (or before) the start of a sprint. This is done to track work associated with the release process and for tracking the production release workflow. The Deployment Users Story is a regular User Story that is used for tracking deployments, integration testing and release to production.

Initially, the deployment user story will contain the list of user stories that the deployment represents within the details of the DUS. Development will need to ensure the list of user stories associated with the deployment user story is complete. In the future (after updates to the WIM tool or move to DB Pro), the deployment story will be linked (using the TFS work item link capability) to the other user stories being deployed in that package. When creating the Deployment User Story, it is critical that development verifies that all the user story links are correct. This is done to ensure we can trace user stories from development through production release.

The Deployment User Story will also be given a release date in order to provide RM a method for tracking what is targeted for a release. In order to be included in a release, the code to be released must be merged into the main branch and build successfully, one week before the release. See “*Weekly Release Cycle*” below.

The tasks that may be included in the deployment user stories are:

* Code Merge (not required, developer merges code to Main)
* Non-production Deployment (i.e., Test, Stage or Auth)
* QA in Test Environment
* UAT (if applicable, it should be done in Test)
* QA in Stage Environment (RM merges code to Release or Hotfix & deploys to Stage)
* Production Validation
* Production Deployment

From a release management standpoint, the critical tasks are the deployment tasks. It is important that these user stories have the right type of tasks to ensure the deployment requests are in RM’s work queue.

Below is a high level overview of the release process.



When the Scrum team determines that code is ready to move into the Test environment, the code is merged into the Integration branch. Development will deploy the integration branch to the development environment or request RM to deploy to Test depending on how significant the change is, what other code is in those environments and where we are in the weekly release cycle. For example, if code is currently being tested in Test from a build from the Main (Release Candidate) branch, it would be best to run the initial integration tests against the Development environment. If there is a minor change that is planned to release on its own, it would make more sense to test build the integration branch and deploy it to Test.

Once the build from the integration branch is validated, the code needs to be merged down to the main branch by the development lead. A deployment task is assigned to the Release Manager (Raul Lorenzo) that is associated to the “Deployment User Story”.

# Weekly Release Cycle

In order to facilitate more rapid release and control within the development teams, the weekly release cycle will be changed as described below. In the schedule “T” is the release date. “T minus” or “T –“ is used to designate the number of business days before the release.

* T minus 5 (Wed.) Build from Main branch and deploy to Test environment
* T minus 4 (Thurs.) Approve the Build from Main; RM cuts Release Branch and deploys it to Stage
* T minus 3 (Fri.) Approve build from Release; Test in stage (primarily automated)
* T minus 2 (Mon.) Release build is packaged
* T minus 1 (Tues.) Change management meeting
* T (Wed.) Production Release

# FAQ

***If multiple teams are deploying the same application in a release, should each have their own DUSt?*** Yes, but a development lead should coordinate across the teams to ensure deployment requests are coordinated and code is merged properly.