Notes from the Meeting

Habi.suite External/Public

- Test of a released version -

--- Releasing Habanero will be done quarterly. These releases will be named 2.5, 2.6 etc. Tags made against this branch will be 2.5.1 etc.

-- Public Trunk - Source, dlls and synced. Potentially unstable.

-- Internal Alpha - Branches for specific projects. These can be any number starting from 1 and onwards.

--- Insuring you have all the parts of all the sub projects that you need you have to branch all the sub projects. You have to build them manually copy all dll's to the libs. CheckIn. You copy all these to your project.

-- Public Betas - Not fully tested in a large set of production environments. But should be reasonably stable. Only changes to a public beta will be bug fixes.

-- Public Stable - Fully tested in a large set of production environments. Should be stable. The only changes will be bug fixes.

1) Ensure that Hudson is working with the latest of all Dlls. Therefore if hudson works we know the latest work together.

2) Ensuring that a developer is using the latest Dlls. Run script.

Public release ensuring that all versions work together.

Public Trunk Code

- Should compile together and their tests should all pass.

Public RC suite

- Should compile together and their tests should all pass.

-- latest release source,

-- latest release suite.

Public RC indiv Habi projects

- Should only be faces, testability, smooth, naked, bindability.

Public RC to public stable.

Habanero .Net 2.0. Any CPU. 64 bit, 32 bit. Any CPU at moment built to any CPU.

- Formal testing of a set of projects. Agree on what these projects are and wait 1 month for public for public feedback.

We should aim for a quarterly release of the RC and Stable.

Use Version Updater for all database updates.

Notes from B4 meeting.

1) Habanero Versioning and Seperate projects.

The issue with this is that we now have a large number of seperate projects. I think this is good because it creates the impression of an eco system around Habanero.

The problems are keeping these all up to date in sourceforge and ensuring that users have a sensible path to understand the peaces and how they fit together.

The second problem is that these projects all use libs to store there references v.s. previously when they directly referenced the bin of the referenced project. The consequence is that if a change is made to Habanero.Core then no changes are propogated to Habanero.Faces etc. The bug or error will then only be discovered when a user compiles all the dlls and updates all the libs of all the related projects manually.

The third problem is how do we extend our community to include developers outside of chillisoft.

2) Production project versioning and management.

Currently whether the developer is using a branch of Habi or trunk of Habi the Production.project is referencing the projects lib.

This causes a number of problems

a) If an update/bug fix is made to the Habi.branch or to Habi.trunk and the developer does not recompile this Habi Project and copy the compiled dlls to the production lib then these changes and any affect from them will not be noticed untill a much later date. This would even be the case where a change caused the production project to not compile or tests to not pass let alone a more sublte change.

b) At what point does the production developer branch and tag their project and the referenced habi projects.

I think that we need to recognise that there are competing objectives here.

b.1) We really really want as many projects testing the latest changes in habi as soon as possible so as to ensure that any changes to Habi are good (i.e. don't break a whole lot of stuff). *If it is difficult do it often and early*.

b.2) We need to deliver high quality bug free production project on time. I.e. we want a stable environment.

b.3) If the developer finds that in the course of his/her project he/she needs to make an enhancement to Habi then he/she should be on trunk since you should only be doing bug fixes on a branch.

There are many other variables involved in making the decision as to when to move to a stable version of Habi, when to branch etc e.g. the experience of the developers involved, the amount of new functionality required i.e. if this project is using a lot of functionality that was not previously in Habi, The risks involved in the project, How tight the deadlines are, How conservative the developer is.

I think the decision as to when the developer changes from using Trunk Habanero to a stable version of Habanero needs to be left up to the Team Lead on the project and needs to made on a project by project basis.

From Mark - Here are my thoughts for the meeting:

The issue that arose is that we would like a more structured way of versioning Habanero and releasing the new versions as part of a roadmap, instead of some project’s release.

So, the one thing that needs to be discussed at some stage is this Habanero roadmap & versioning strategy, but that is not for this discussion.

The issue is that we have had project development happening on trunk that when it comes time to release, then there is pressure to release Habanero.

So, I think we should follow the following procedures:

- All projects should only have referenced components in the project’s local lib

- Project trunk folders should reference Habanero trunk dlls in the project’s lib and as part of their build process, these dlls should be copied from the Habanero trunk bin folder before compiling and running tests (this would be projects with an SLA only)

- We should start using branches for project ‘feature’ development (bug fixes should happen in the latest branch). So the procedure would be:

o Make the branch of the project (following the procedure as detailed in the wiki – I was working on drawing this up with Duncan)

o Choose a branched Habanero version (& others) to use and copy the dlls into the lib

o Document this version’s dependencies and planned development in the Wiki

o Do the development

o The lib shouldn’t need to be updated that many times at all because the Habanero branch should be stable

o If you need to make a bug fix to Habanero, then do that fix on the Habanero branch and merge to Habanero trunk immediately

o If you need to make a feature change to Habanero, create a new feature branch (from the branch you are working on) and do these changes here. Document this in the wiki and place the merging of this feature into the Habanero development roadmap (when should it be merged into **trunk**? When finished, or at a planned point?). The name of this feature branch should be something like “v2.4\_bobfeature”.

o When work on this project branch is tested and released a tag should be created.

o When development on the branch is completed, then all the changes should be merged back into the project trunk.

In summary, the idea is this: The main portion of development should happen on branches instead of the trunk. Is this a decent approach? Does this match the normal SVN model?

Does GIT offer us anything for this process, or are the only advantages of GIT for distributed version control (I haven’t looked much into it)?

Some other questions to ask:

If we are not wanting other projects to fuel Habanero releases, then should we follow this same procedure with other Habanero Community projects?

The versioning of Habanero Community projects should be separate to the versioning of dependant components, so this needs to be controlled & documented.

Production Proj version

- Green Fields

Where possible use trunk. At some point to a branched version of Habi branch 13 and branch project this must before release retesting big demo etc but weekly drop testing can be against trunk. Branch including FS. If you are the only developer working on that branch then do not need to tag. developing against trunk then you should not have to Tag. If find bug fix in branch and

Every time you interact with customer or submit for testing including demos should have a new version number for your project.

Developing against a public branch of habi.

When submit to kelly for testing or you can stay on Branch or move to a Tag or create a Tag.

Any Deployed/Production Release to customer you must be on a Tag of all Habi sub projects (This can be automated in future). You will know you are against that tag because you have changed all your scripts to reference that tag. Tag of bin folder will be checked in.

Every release to Kelly or customer for demo testing or any other purpose must be tagged.

Whenever you are releasing something for testing demos etc and you are working on you need to branch your project. This branch must work on the appropriate version of habi.

Maintenance/ Enhancements - Mark responsible for decision and updating wiki page.

Obligation to keep up to date with latest Stable release. Therefore any change must at least move to latest stable TAG/Talbot.

Project lead should make decision to move to RC or Trunk.

Releasing as above based on the scenario.

habi.Core, Faces :- Peter.

Testability, Smooth - Documentation UI gen Validation Gen, , Naked - Brett.

Bindability, - Brett.

Firestarter - Mark

- DB Schema

- DM Modelling

- Templating Generation.

The issue is to find tools or a mechanism for managing code reviews.