Tagging Information

CMT Tags

For what concerns the format of tags recognized by CMT, it appears that it is a lot more liberal that what one generally thinks. Actually, CMT only focus on the numbers in the tag names. It is up to the collaboration to choose one format or the other. The only advice given by CMT authors is to avoid dots, if the CMT is used together with CVS, because CVS will not like dots in the tag names.

The format which is generally used in the CMT documentation can be expressed thanks to this perl expression : v[0-9]+(r[0-9]+(p[0-9]+)?)?. For ease of reading, we will call it v*r*p* below, but one must note that v*r* can be used, but not v*p*. In some situations, CMT only focus on numbers in the tag name, so v3p1 could be considered equivalent to v3r1...

Also, CMT is meant to be used together with a version management tool, such as CVS or subversion, but it tries not to depend on specific aspects of any of these tools. In particular, CMT is not aware of CVS branches. Badly enough, CMT developers uses the term "branches" for the subdirectories of a package.

CMT only cares for tags, and it get those tags for a CVS package thanks to the command "cmt cvstags". The way the tags are ordered is unclear and linked to the implementation of the cmtcvs pluggin which is installed in the CVS repository. My current understanding is the following :

- For a given package, all the tags except the first one are given in the order they appear in the cmt/requirements file in the repository ; it should be the same order as the command "cvs rlog", and I guess it is chronological.
- The tag given first is expected to be the "latest" or more "recent" depending on the source ; actually, it seems to be the tag which have greater numbers in the corresponding CVS file release number.

CMT is not aware of the eventual collaboration policy for tags, so it is considering all the tags, whatever they look like, and it does not care for the CVS branches :

- If a branch has just been made for a given package, and no new tags have been applied to the CVS trunk, then the last tag on the branch will be displayed first by the "cmt cvstags" command.
- If the first tag returned by "cmt cvstags" must be filtered-out, then the following tags are ordered chronologically, which means the branch ones could come first if they have been made lately.

This make me think that it is dangerous to rely on the ordering of the output of "cmt cvstags", and dangerous to modify the cmtcvs pluggin without consequences on the general behavior of cvs. Perhaps we could try to affect only the selection of the first tag, but this would need to be done again each time a new version of the pluggin is delivered by CMT authors.

Release Manager LATEST

The Release Manager is filtering out all the tags which do not match v[0-9]+(r[0-9]+(p[0-9]+)?)?. This is a way not to take into account the tags which are applied on CVS branches, provided the developers do not use this format of tags on the branches !

LATEST is said to be only for GlastRelease, and not for other check out packages. Actually, I feel it as a place where to check that the head of each package stay up-to-date with the head of its providers and clients. I would not say it is GlastRelease oriented or Engineering oriented. It is rather "cvs head" oriented, so to remember to each package owner he must keep in touch with other packages evolutions.

Since LATEST is meant to take the latest trunk tag of each package, a way to get those tags is to ask developers to use v*r*p* only on the CVS trunk, and just select the last one.

Release Manager HEAD

The tags which are used in a HEAD build are explicitly defined one by one, under the controll of Navid. There are HEAD version for every check out package : BeamtestRelease, EngineeringModel, GlastRelease and ScienceTools. By default, the selected tags should follow the same rules as for LATEST, but there are situation where one could need a branch :

- We are preparing a patch for a given version, for example GlastRelease v9r0, but one of the package to be patched has always been strongly modified for some future big version. Then we need to start a branch from the tag used for v9r0.
- Two checkout packages are relying on two different version of a given low level package. Typical case : EngineeringModel requires a different Event from GlastRelease. Then we again need a branch for Event.

What we are developing on those branches should not affecty LATEST, yet we should establish a policy so to easily recognize the branches and tags which are meant for next HEAD and official versions. For a given <package>, whose last GlastRelease HEAD tag was v<i>r<j>p<k>, the current policy is :

- The branch should start from <package> v<i>r<j>p<k>,
  and the branch tag should be called GR-v<i>r<j>p<k>.
- The future tags on the branch should be called v<i>r<j>p<k>gr0, v<i>r<j>p<k>gr1,
  and so on.

In some place, a confusion is made between the tags which are defining a branch, and the tags which are applied to some revision which is on a branch. Try to always distinguish betweenthe former, which I would call "branch tags", and the latter, which I would call "tags on branches".

The Art of Use in Requirements

Where to use specific tags ?
Where to use generic ones such as v12r* ?
To be written.

Places where there is information about tags

About CVS branches.
About Release Manager.
Within CMT introduction, when clicking on "Tagging Conventions".