METCS673

PLM - Project Lifecycle Management

Software Confiugration Management Plan

Author:
Christian Heckendorf

September 21, 2013

1 Proposed Configuration Management Plan

1.1 Configuration Items and Tools

All configuration items will be stored in a git repository hosted on GitHub. The following configuration items are required:

- Documentation directory
  - Planning documents
  - Code documentation
  - User level documentation
- Front-end code directory
  - User interface code
- Back-end code directory
  - Services code
- Database code directory
  - Database create scripts
  - Sample insert scripts

Be careful what you add to the repository. In general, binaries and temporary files should not be added. Binary documentation is allowed in the repository but the plain text source documents should accompany them.

1.2 Change Management and Branch Management

The general outline is as follows:

- The master branch should be at demo quality at all times.
- Unstable code can be merged into a development branch.
- New features should go in their own branches, which will at some point be merged into the development branch.
- Approximately every three weeks, the development branch will be merged back to the master branch to prepare for the presentation.

This system should allow us to always have a state available for a demo while continuing to add features in feature branches and fixing bugs in the development branch. To begin, one person should commit the basic layout of the project. From there, each member can begin working on feature branches based on that initial commit. At some point, a feature will be complete and can be merged back to the development branch. We may want to have someone review the code before merging to minimize bugs creeping into the development branch. At the end of a development iteration, the developemnt branch will be merged back to the master branch to prepare for the presentation.

1.3 Code Commit Guidelines

Code that produces validation or build errors should not be pushed to github. I don't have a problem with committing code that's broken but it should stay in your local repository until the appropriate fixes are committed as well.

2 Reference

2.1 Git Basics

Since I don't know what git client you'll be using, I'll summarize the basic command line utility commands.

When you first start working on the project, you'll want to make a local copy of the repository to keep track of your changes:

```
git clone https://github.com/heckendorfc/plm.git
```

Before you begin working, you should create a feature branch based on the development branch:

```
git checkout -b somefeature origin/development
```

Once you have some code written and want to save your work to the repo, tell git which files you would like to commit to the repo and commit them:

```
git add file1 file2 ...
git commit -m "commit message here"
```

To get the latest code that other team members have shared on the github repo:

```
git fetch origin
```

After you have one or more commits in your local repo, you'll want to share them with the rest of the team:

```
git push -u origin somebranch
```

Further reading: http://git-scm.com/book