Developer's Guide of the CROSSMINER Advanced Integrated Development Environments Plug-in

2

Chapter 1

Development Processes

The general layout of the development and testing process are shown on Figure 1.1
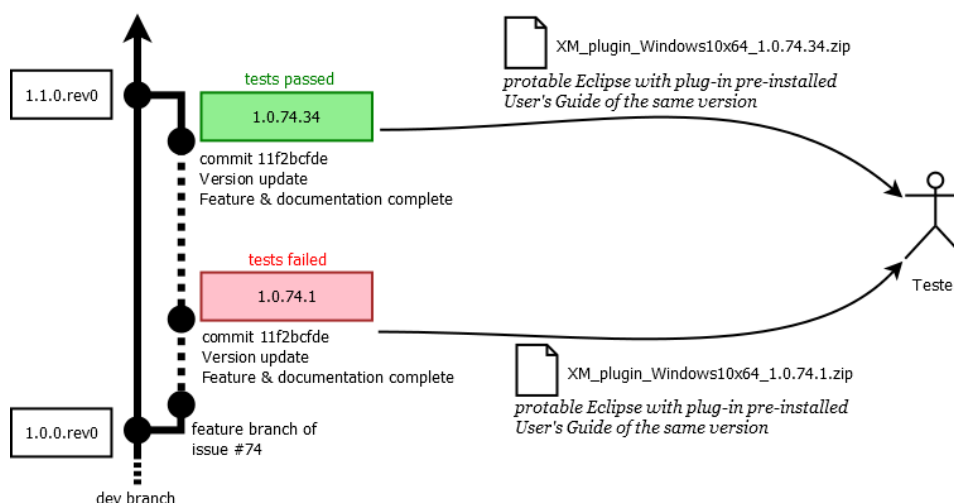

Figure 1.1: Overview of the testing and development process

1.1 Versions

The version number consists of three parts separated by period. It follows the layout $\langle main\rangle.\langle sub\rangle.\langle issue\rangle.rev\langle revision\rangle$. To understand when and how to change these parts please consider the following guide.

Increasing main version Request rights of lead developer or above. It will be increased when the current version are shipped and demonstrated to the (whole) consortium.

Increasing sub version Could be changed by any participant. Increase it by one respecting to the current version of the `dev` branch when a new feature are finished, passed all required tests and merged into the current development version (on `dev` branch). Reset it to 0 every time main version number increases.

Changing issue number You have to set this part to be equal the identification number of the current issue associated to the feature branch you are on. Set it 0 on non-feature branches.

Incrementing the revision Could be changed by any participant. Increment it by one, before you send a new version for testing. It could be the first version containing a testable new feature or any fixes applied due to testers concerns. Reset it to 0 at the beginning of each new feature branch.

There are some additional rules need to consider, partially implicated by the these statements. You have to follow these as well the previous ones.

Development versions Each version on the `dev` branch should follow this scheme: ⟨*main*⟩.⟨*sub*⟩.*0.rev0*, for example *1.12.0.rev0*.

Testable versions Each version sent for testing should follow this scheme: ⟨*main*⟩.⟨*sub*⟩.⟨*current-issue*⟩.*rev*⟨*revision*⟩, for example *2.23.42.rev11*.

Version tagging Each version should be explicitly tagged in the Git repository using the following commands (if your HEAD points to the relevant commit):
```
git tag ⟨version⟩
git push origin ⟨version⟩
```

Issue-branch-feature tracibility Each feature branch has to have a single and unique issue. Do not change (create new) version off the plugin after merging a branch contains only refactoring without any new features or fixes.

Tracability of User's Guide The User's Guide (this document) has to follow the version any version changes. You have to update the guide before you change the version number.

1.2   Structure of Releases

After chancing the version you have to assemble a new release package. By doing so please follow these rules.

Naming Each release package should be named by the following scheme:
XM_plugin_⟨operation-system⟩_v⟨version⟩.zip,
for example `XM_plugin_Windows10x64_2.23.0.rev0`

Content Each release should consist of a single, self-contained compressed folder, containing a portable Eclipse with the plug-in pre-installed.[1]

Placing Documentation The relevant version of this document should be placed in the root folder of the compressed package in PDF format.

1.3 Testing

Each feature and function should be tested against expected functionality described in this document. Any discrepancies should be reported.

Reporting failed tests All failed test should be reported using a dedicated issue recorded in the project. `https://git.sed.hu/geryxyz/crossminer/issues/new`

Content of the report Each report should contain a detailed description of the action and condition of test to be repeatable. You have to specify the concrete version in which the divergence occur and the exact place in this document where the feature was described.

Goal of testing As a tester you have to ensure that each features are working as described in this document with reasonable reliability.

---

[1]Except in the cases, when the goal of testing is installation.