10 Steps to Enhancing BridgePoint

BridgePoint xtUML Editor and Model Compilers

Extending BridgePoint

- BridgePoint evolves through the efforts of community members
- Your enhancements and patches can play a role in the evolution
  a. Editor & Verifier
  b. Model Compilers
  c. Python-based Generator
- Start by completing the Developer Getting Started guide (https://github.com/xtuml/bridgepoint)

Step 1 - Enter a Bug

- Create account on support.onefact.net
- Add the new issue in the proper subproject of BridgePoint
- What to capture:
  a. Failure mode, reproduction steps, relevant data (model, project, etc)
  b. Known workarounds, if any
  c. Affected version

Step 2 - Create a Working Branch

- Create a new branch in your fork
  a. Merge xtuml/bridgepoint "master" into your fork's "master"
  b. Create a branch from your fork's master
  c. Configure the branch to push to your fork

- Name the branch using form
  `<issue>_<short_description>`
  (e.g. `7993_fix_jre_paths`)

Step 3 - Write a Design Note

- Start with the Design Note template:
  `doc-bridgepoint/process/templates/dnt_template.md`
- Copy to `doc-bridgepoint/notes` and rename for this issue (e.g. `./7745/7745_port_oal_dnt.md`)
- Fill in the sections of the template
- Likely will require looking into the code to get a feeling for what the design should be. Consider and explore alternatives.

Step 4 - Review Design Note

- Design notes should be reviewed
- Hop into the xtUML Community chat and ask for the design proposal to be reviewed
- Other BP developers will read the Design Note and provide feedback in a review minutes document
- Author updates Design Note to address review comments

Step 5 - Implement

- Roll up your sleeves and do the development
- Model a solution using xtUML when possible
- Write Java or Model Compiler RSL if necessary
- Commit early and often
  - Use commit comment `job #<issuenum> - <text>` to tie the comments into the issue tracker

Step 6 - Implement & Run Tests

- Create manual test case for promoter to run
- Or better, write new automated JUnit test cases for use in regression testing
- Run the existing JUnit tests in the affected area

Step 7 - Write Implementation Note

- Grab the Implementation Note template:

  `doc-bridgepoint/process/templates/int_template.md`

- Fill in the sections

- Capture any deviations from the original design documented in the Design Note

- Capture the list of changed files

Step 8 - Merge with Master Locally

- Merge latest `xtuml/<repo> master` into your development branch
- Assures there are no conflicts with other promotions since your branch was created

Step 9 - Submit Pull Request

- Log on to `github.com/xtuml/<repo>`, create a new pull request
  - Use “compare across forks” link to specify your fork as “head fork”
  - Left side is the target (xtuml), right side is source (your fork)
- Log in to SupportPro (`support.onefact.net`) and update the issue state to Resolved

Step 10 - Review and Promotion

- A BP Core Team member will review the code & run relevant tests
- Identified concerns are discussed and code re-worked by author until resolution
- Approved changes are merged into master by committer
- Issue is marked “Closed” by committer

Conclusion

- Open-source software is great. It reaches its full potential when a community is actively engaged
- The BridgePoint development team is eager to help new developers get started!
  a. Bug Tracker - support.onefact.net
  b. E-mail - support@onefact.net
  c. xtUML Community chat and forum on xtuml.org