The Polarsys Maturity Model: Assessing and Improving Software Project Quality

B. Baldassari, G. Blondelle, J. M. Gonzalez-Barahona, D. Izquierdo

https://polarsys.org/wiki/Maturity_Assessment_WG

Polarsys Meeting
Ludwigsburg (Germany), October 27st 2014
Some rights reserved. This presentation is distributed under the “Attribution-ShareAlike 3.0” license, by Creative Commons, available at http://creativecommons.org/licenses/by-sa/3.0/
Structure of the presentation

1. The Big Picture
2. The Quality Model
3. Mining Software Repositories
4. The PolarSys Dashboard
5. Next Steps
The Big Picture
The PolarSys ecosystem
PolarSys in a nutshell

PolarSys Working Group

- Open Innovation to create better methods and tools
- Software tools for critical systems
- Interoperability based on Open Standards
- Fostering of exchanges between academics and industrial partners
- Quality and maturity assessment
- Very Long Term Support for more than 10 years
Objectives...

What we want..

- Assess maturity of components to ensure that the full stack is safe.
- Help people better understand, know and manage their project.
- Propose guidance for good practices and Eclipse processes.
- Involve people to make it a community-driven project.

What we DO NOT want..

- Judge or blame projects or people.
- Address individual characteristics.
The Quality Model
There is no single definition of quality.

- Usual definitions of quality, existing standards and norms are not enough.
- OSS has an impact on the different quality concerns.
- We have to define our own quality requirements for Eclipse and PolarSys.
  - Process: predictable outputs, well-defined structure of projects.
  - Ecosystem: nurture communities, good citizenship behaviour.
  - Product: maintainability, reliability, good practices.
Composition of the Quality Model

Quality Attributes:

- Ecosystem
  - Activity
  - Diversity
  - Responsiveness
  - Support
  - Visibility
  - Usage
  - User feedback

- Project Quality
  - Configuration Management
  - Change Management
    - Intellectual Property Management
    - Planning Management
    - Test Management

- Process
  - Analysability
  - Changeability
  - Reliability
  - Reusability

- Product
We further define:

- **Quality attributes**: define the concerns of quality.
- **Measurement goals**: generic measurement goals that can be applied to all projects.
- **Base metrics**: raw numbers adapted to each project’s characteristics.
The full quality model

Polarsys Maturity Model
Mining Software Repositories
Eclipse Software Repositories

The metrics used in the process are retrieved from several sources:

- **Source Code**: SonarQube, rule-checking tools.
- **SCM, ITS, Mailing lists**: Eclipse Dashboard *(Grimoire)*.
- **PMI JSON files** from the Project Management Infrastructure project.
- **Analysis of Forums, Web sites, or even manual surveys.**
Source code metrics

• Source code metrics are retrieved from SonarQube, but other inputs may be easily added.

• Metrics are computed as a ratio to limit the effect of size of software:
  ▶ **Comment rate**: number of comments per KLOC.
  ▶ **Complexity density**: average number of paths per function.
  ▶ **Duplicated lines density**: amount of duplicated lines per KLOC.

• Some metrics are kept absolute because they are self-explicite:
  ▶ **Source Lines of Code**: representative of the size of the software.
  ▶ **Depth of Inheritance Tree** for OO languages.
Rule-checking metrics

- Rule-checking tools define **good practices** elaborated by the Open-Source Community.
- PMD 5.2.1 and FindBugs 3.0.0 are used for now, but other tools can be easily included in the framework: e.g. CheckStyle, FxCop, Lint, CppCheck.
- Examples of metrics include:
  - NCC (NCC\_ANA, NCC\_CHA, etc.): number of violations (related to analysability, changeability, etc.)
  - ROKR (ROKR\_ANA, ROKR\_CHA, etc.): percentage of acquired practices (related to analysability, changeability, etc.)
Software Configuration Management metrics are extracted from tools metadata.

Supported tools are Git, Subversion, CVS, Bazaar, Mercurial.

Examples of metrics include:

- Number of commits during last month.
- Number of committers during last month.
- Number of files committed during last month.
ITS metrics

- Issue Tracking System metrics are extracted from tools metadata.
- Bugzilla, Jira, Redmine, Launchpad, GitHub and SourceForge ITS are supported.
- Examples of metrics include:
  - Number of bugs per thousands Lines of Code during last month.
  - Number of users active on the ITS during last month.
  - Number of updates to the ITS during last month.
  - Median time to fix issues during last month.
MLS metrics

- Mailing list metrics are extracted from the project’s archives.
- MBox and GMane are supported for now (forums are planned).
- Examples of metrics include:
  - Number of mails exchanged during last month.
  - Number of threads exchanged during last month.
  - Number of active authors during last month.
  - Median ratio of answers on questions during last month.
PMI metrics

The Project Management Infrastructure maintains a list of characteristics for all projects, with:

- Information about repositories: SCM, ITS, MLS.
- Links for published content: web site, wiki, getting started, build process, official documentation.
- Description of releases, date, scope, status of review...
The PolarSys Dashboard
Architecture of the platform
Welcome to the Maturity Assessment dashboard!

Maturity Assessment

PolarSys is an Eclipse Industry Working Group created by large industry players and by tools providers to collaborate on the creation and support of Open Source tools for the development of embedded systems. Members of PolarSys started working on a Maturity Assessment task force back in 2013, to assess and help improve quality of projects entering the PolarSys umbrella.

This page is automatically generated from the definition files stored in a git repository, so they can not be modified directly. Instead, please send your remarks to the PolarSys mailing list or contact me (Boris Baldassari).

Projects
List of projects analysed.
- CDT
- Pajima
- Sirius.

We also set up an example project to demonstrate what it should look like when all information is available for the analysed projects. Most of its information is borrowed from CDT, and missing parts have been manually filled.

Documentation
This process is self-documented: this documentation is automatically generated from the definition files themselves.
- The Quality Model defines how quality attributes are organised, and mapped to measurements concepts and metrics.
- Measurement concepts are generic measurement units, that can be potentially applied to all types of languages or software development processes. From these measurement concepts, the quality model defines metrics targeted to a specific context: language, target, etc.
- Metrics are measured together and the project environment process, source code, unit, etc. They are thoroughly described in this section with their description, source, and impact.
- Rules are computed by well-known rule-checking tools PMD and FindBugs. They are software development practices known to be good.
The Dashboard: Quality Model for CDT
### Measures for [example]

<table>
<thead>
<tr>
<th>Name</th>
<th>Mmerno</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment rate</td>
<td>COMR</td>
<td>16.4</td>
</tr>
<tr>
<td>Number of downloads on the web site</td>
<td>DL_REPO_1M</td>
<td>572801</td>
</tr>
<tr>
<td>Number of downloads on update site</td>
<td>DL_UPDATE_SITE_1M</td>
<td>3359</td>
</tr>
<tr>
<td>Function cloning</td>
<td>FU_CLONE</td>
<td>115314</td>
</tr>
<tr>
<td>ITS authors</td>
<td>ITS_AUTH_1M</td>
<td>55</td>
</tr>
<tr>
<td>Defect density</td>
<td>ITS_BUGS_DENSITY</td>
<td>0.08530612869851283</td>
</tr>
<tr>
<td>Median time to fix bug</td>
<td>ITS_FIX_MED_1M</td>
<td>14.95</td>
</tr>
<tr>
<td>ITS updates</td>
<td>ITS_UPDATES_1M</td>
<td>237</td>
</tr>
<tr>
<td>Number of favourites on the Marketplace</td>
<td>MKT_FAV</td>
<td>7</td>
</tr>
<tr>
<td>Number of failed install on the Marketplace</td>
<td>MKT_INSTALL_FAILED_1M</td>
<td>301</td>
</tr>
<tr>
<td>Number of successful install on the Marketplace</td>
<td>MKT_INSTALL_SUCCESS_1M</td>
<td>3058</td>
</tr>
<tr>
<td>Developer ML authors</td>
<td>MLS_DEV_AUTH_1M</td>
<td>32</td>
</tr>
<tr>
<td>Developer ML response ratio</td>
<td>MLS_DEV_RESP_RATIO_1M</td>
<td>2.71875</td>
</tr>
<tr>
<td>Developer ML subjects</td>
<td>MLS_DEV_SUBJ_1M</td>
<td>32</td>
</tr>
<tr>
<td>Developer ML posts</td>
<td>MLS_DEV_VOL_1M</td>
<td>119</td>
</tr>
</tbody>
</table>

**More info**

You can get more information by clicking on the metric name or mnemonic.

**Download data for this project**

- Metrics [JSON]
- Concepts [JSON]
- Attributes [JSON]
Welcome to the Maturity Assessment dashboard.

### Attributes for [cdt]

<table>
<thead>
<tr>
<th>Mnemo</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QM_ACTIVITY</td>
<td>Activity</td>
<td>5</td>
</tr>
<tr>
<td>QM_ANA</td>
<td>Analysability</td>
<td>5</td>
</tr>
<tr>
<td>QM_CHA</td>
<td>Changeability</td>
<td>4</td>
</tr>
<tr>
<td>QM_COMMUNITY</td>
<td>Community</td>
<td>3</td>
</tr>
<tr>
<td>QM_DIVERSEITY</td>
<td>Diversity</td>
<td>4</td>
</tr>
<tr>
<td>QM_FEEDBACK</td>
<td>Feedback</td>
<td>4</td>
</tr>
<tr>
<td>QM_IT'S</td>
<td>Change Management</td>
<td>4</td>
</tr>
<tr>
<td>QM_PLAN</td>
<td>Planning Management</td>
<td>3</td>
</tr>
<tr>
<td>QM_PROCESS</td>
<td>Process</td>
<td>3</td>
</tr>
<tr>
<td>QM_PRODUCT</td>
<td>Product</td>
<td>2</td>
</tr>
<tr>
<td>QM_QUALITY</td>
<td>Maturity</td>
<td>4</td>
</tr>
<tr>
<td>QM_REL</td>
<td>Reliability</td>
<td>2</td>
</tr>
<tr>
<td>QM_RESPONSIVENESS</td>
<td>Responsiveness</td>
<td>3</td>
</tr>
<tr>
<td>QM_REU</td>
<td>Reusability</td>
<td>1</td>
</tr>
<tr>
<td>QM_SCM</td>
<td>Configuration Management</td>
<td>5</td>
</tr>
<tr>
<td>QM_SUPPORT</td>
<td>Support</td>
<td>2</td>
</tr>
<tr>
<td>QM_TST</td>
<td>Test Management</td>
<td>2</td>
</tr>
<tr>
<td>QM_USAGE</td>
<td>Usage</td>
<td>2</td>
</tr>
</tbody>
</table>

More info

You can get more information by clicking on the metric name or minemo.
The Dashboard: Practices for CDT

### Practices (rule violations) for [example]

This rules have been extracted from PMD 6.1.2 and FindBugs 3.0.0.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mnemo</th>
<th>Priority</th>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Class Without Abstract Method</td>
<td>AbstractClassWithoutAbstractMethod</td>
<td>3</td>
<td>ANA</td>
<td>28</td>
</tr>
<tr>
<td>Abstract Class Without Any Method</td>
<td>AbstractClassWithoutAnyMethod</td>
<td>3</td>
<td>CHA</td>
<td>7</td>
</tr>
<tr>
<td>Accessor Class Generation</td>
<td>AccessorClassGeneration</td>
<td>3</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Assignment To Non Final Static</td>
<td>AssignmentToNonFinalStatic</td>
<td>3</td>
<td>REL</td>
<td>99</td>
</tr>
<tr>
<td>Avoid Branching Statement As Last In Loop</td>
<td>AvoidBranchingStatementAsLastInLoop</td>
<td>2</td>
<td>ANA</td>
<td>42</td>
</tr>
<tr>
<td>Avoid Constants Interface</td>
<td>AvoidConstantsInterface</td>
<td>3</td>
<td>ANA</td>
<td>509</td>
</tr>
<tr>
<td>Avoid Deeply Nested If Statements</td>
<td>AvoidDeeplyNestedIfStatements</td>
<td>3</td>
<td>ANA</td>
<td>1102</td>
</tr>
<tr>
<td>Avoid Instance0() Checks In Catch Clause</td>
<td>AvoidInstance0()ChecksInCatchClause</td>
<td>3</td>
<td>CHA</td>
<td>11</td>
</tr>
<tr>
<td>Avoid Protected Field In Final Class</td>
<td>AvoidProtectedFieldInFinalClass</td>
<td>3</td>
<td>ANA</td>
<td>5</td>
</tr>
<tr>
<td>Avoid Protected Method In Final Class Not Extending</td>
<td>AvoidProtectedMethodInFinalClassNotExtending</td>
<td>3</td>
<td>ANA</td>
<td>24</td>
</tr>
<tr>
<td>Avoid Reassigning Parameters</td>
<td>AvoidReassigningParameters</td>
<td>3</td>
<td>CHA</td>
<td>1877</td>
</tr>
<tr>
<td>Avoid Synchronized At Method Level</td>
<td>AvoidSynchronizedAtMethodLevel</td>
<td>3</td>
<td>EFF</td>
<td>631</td>
</tr>
<tr>
<td>Avoid ThreadGroup</td>
<td>AvoidThreadGroup</td>
<td>3</td>
<td>REL</td>
<td>3</td>
</tr>
<tr>
<td>Avoid Using Hard Coded IP</td>
<td>AvoidUsingHardCodedIP</td>
<td>3</td>
<td>CHAR</td>
<td>19</td>
</tr>
<tr>
<td>Avoid Using Non-final Values</td>
<td>AvoidUsingNon-finalValues</td>
<td>3</td>
<td>REL</td>
<td>ANA</td>
</tr>
</tbody>
</table>

More info

Rule violations are retrieved from well-known rule-checking tools like PMD and FindBugs. Rules can be assimilated to good and bad practices. They are all attached to a category (quality attribute, check the quality model for more information) and have a priority.

Download data for this project

- Metrics [JSON]
- Concepts [JSON]
- Attributes [JSON]

Plot it!
### The Dashboard: Actions for CDT

#### Process

Process actions impact the **organisational** maturity of the project: predictability of outputs, traceability, best management practices, and more generally any of the key process areas defined in the CMMI.

<table>
<thead>
<tr>
<th>Action</th>
<th>Mnemo</th>
<th>Priority</th>
<th>Category</th>
<th>Where?</th>
</tr>
</thead>
<tbody>
<tr>
<td>The source repository is not provided in the Project Management Infrastructure. Filling this field helps people easily find your sources, which is good for collaboration and good-citizenship behaviour.</td>
<td>PMI_SRC_REPO</td>
<td>1</td>
<td>Process</td>
<td>See PMI page</td>
</tr>
<tr>
<td>The bugzilla entry is not provided in the Project Management Infrastructure. Filling this field helps people easily find your issue tracking system, which is good for getting feedback (testing) and good-citizenship behaviour.</td>
<td>PMI_SRC_BUGS</td>
<td>1</td>
<td>Process</td>
<td>See PMI page</td>
</tr>
<tr>
<td>The developer mailing list entry is not provided in the Project Management Infrastructure. Filling this field helps people easily find your mailing list, which is good for collaboration and good-citizenship behaviour.</td>
<td>PMI_SRC_ML_DEV1</td>
<td></td>
<td>Process</td>
<td>See PMI page</td>
</tr>
</tbody>
</table>

#### Product

These practices have a level 1 criticality and are considered important stuff. You should take corrective action.

<table>
<thead>
<tr>
<th>Action</th>
<th>Mnemo</th>
<th>Priority</th>
<th>Category</th>
<th>How often?</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC: Impossible cast</td>
<td>BC_IMPOSSIBLE_CAST</td>
<td>1</td>
<td>COR REL</td>
<td>1</td>
</tr>
<tr>
<td>Avoid Branching Statement As Last In Loop</td>
<td>AvoidBranchingStatementAsLastInLoop</td>
<td>2</td>
<td>ANA</td>
<td>42</td>
</tr>
<tr>
<td>Boolean Instantiation</td>
<td>BooleanInstantiation</td>
<td>2</td>
<td>ANA</td>
<td>127</td>
</tr>
<tr>
<td>Broken NullCheck</td>
<td>BrokenNullCheck</td>
<td>2</td>
<td>REL</td>
<td>11</td>
</tr>
<tr>
<td>DMI: Invocation of hashCode on an array</td>
<td>DMI_INVOKING_HASHCODE_ON_ARRAY</td>
<td>2</td>
<td>COR</td>
<td>1</td>
</tr>
</tbody>
</table>
Next Steps
Next steps

- Scales: 1-5 scales to instantly get an idea of what numbers mean.
- Improve presentation of results: visualisation, ergonomy.
- Add more content: metrics, rules, actions.
- Improve automation, ease the adoption setup for new-coming projects.
- Add more projects. What about yours?
Join us!

- The PolarSys wiki: polarsys.org/wiki
- Maturity Assessment on the wiki: polarsys.org/wiki/Maturity_Assessment_WG
- PolarSys Mailing list: dev.eclipse.org/mailman/listinfo/polarsys-iwg
- Look at the live dashboard prototype: castalia.camp/dl/dashboard/
References

