Table of Contents

Table of Contents

Q&A related to Metrics Meeting

General Questions

Questions relating to all metrics

Q. What is the prioritisation on metrics? (UNICORE)

A. Bug-tracking related metrics:

- defect tracking takes precedence over feature tracking.
- currently, immediate and high priority bugs open bugs should be highlighted in the EMT.
- the trend in immediate and high priority closed bug turnaround time is much more important than medium and low priority bugs, in Priority Bugs and Backlog management Index (BMI).

A. In software metrics:

- High priority FindBugs and memory leaks in cppcheck are top priorities.
- Certain sections of PMD and Cyclomatic Complexity should be next.
- Checkstyle and other PMD sections should be last.

Q. What part of the metrics are given to the outside and which are for purely PT purposes? (dCache)

A. EU reports

- Anything mentioned in an EU report (rather than detailed report or EMT report) is pased outside.
- Also, the PEB reserves the right to look at any of the metrics at any time with the help of the SA2 lead.

Dashboard and Trend Graphs

Q. We need a dashboard, can we have one? (gLite)

A. To appear...

- This is on the to-do list of SA2.4 for the metrics group (SA2.3).
- It is absolutely necessary to have this, since trend graphs must be deduced and an S-curve needs to be provided for decreasing threshold until the release of EMI-2.

Specific Bug-tracking Issues

Q. Does "closed" state mean the same thing in all bug-trackers? (gLite)

A. Yes

- The idea of fix certified and reaching production in gLite is no longer valid. The request for change (RfC) policy as laid out in the SQAP is identical for all middlewares.

Q. Is the time from bug state "fixed" to "closed" a better measure of cumulative days to close a bug? (gLite)

A. Yes and No.

- The optimal solution seems to be to produce an average lifecycle of a set of bugs in a time-period per product team.
- The life-cycle is broken down into "open"->"accepted/rejected"->"fixed"->"verified/not-verified"->"closed" with average times for each transition.

Q. Is time to close a bug dependent on the type of bug and resulting in misleading metrics? (dCache)

A. No.

- This is where escalation of priority plays a very large roll. You must diminish items that are non-critical if you can work without them.
- Also, the time to close bugs must be compared with time to fix a bug, hence the lifecycle plots give a good overall view of what is going on in a product team.
- Diminishing items of high priority to give good metric statistics will not go unnoticed within the middleware itself.

Questions relating to all Software Metrics

Q. Are all components built in ETICS? (UNICORE)

A. No.

- Some UNICORE components are not in ETICS and therefore do not appear in metric software reporting.

Specific Metric Questions and Issues

Bug Density Distribution

Q. Is total bug density increasing over time and therefore uninterpretable? (UNICORE)

A. No

- You assess the total bug density in a time period, so you calculate the number of open or closed bugs in a certain period and then normalise with respect to the kLOC.

Q. Is total bug density a realistic metric when refactoring code results in a large decrease in kLOC? (UNICORE)

A. Yes and no.

- Not realistic if you forget to compare the total bug density with total bugs. If compared with total bugs, the refactoring will be noticed.
- Otherwise it is realistic, since after the code is refactored, you expect the bug density to decrease over time.
- If the refactoring is bad, the bug density will increase (in some cases dramatically) which will be

shown in the trend graphs.

FindBugs

Q. Is the threshold zero? (UNICORE)

A. No. Not yet...

- By the release of EMI-2 the goal is to have zero high priority FindBugs in each PT. This therefore infers a reducing threshold over the course of year 2 (using an S-curve).
- The documented threshold and formula is out of date. The plan is to only highlight high priority bugs, not medium or low priority bugs. False positives must be ignored or filtered.

Q. Is the threshold/output value relevant for? (UNICORE)

A. No.

- It should be just a simple count of the number of high priority bugs per kLOC, not the formula that now exists which is too complicated to "interpret at a glance".

PMD

Q. PMD seems hard to interpret, is it relevant? (UNICORE)

A. Only a subset is relevant.

- A subset of the total set should be agreed by the product teams, SA1 and SA2.
- It is definitely a lower priority metric than cppcheck and FindBugs.

Checkstyle errors

Q. Is checkstyle useful? (UNICORE, gLite)

A. Currently, not really.

- Since there are too many possible items to interpret. There is a suggestion to hone in on JavaDoc in the first instance from the complete set of results.

Python (pylint) errors

Q. Where are the python metrics? (gLite)

A. is due for release around the end of May.

Cyclomatic Complexity

Q. Should CC be related to kLOC? (UNICORE)

A. This remains to be seen.

- We need to analyse CC/kLOC per product team to investigate this.

Quality in Use Metrics (GGUS)

Q. Are user incident metrics of use? (UNICORE)

A. This remains to be seen.

- We need to plot the information received from GGUS that we need to make an interpretation of the results.
- The main issue is now is whether you can neglect something that is defined in the DoW and specified as KPIs from the start of the project.

Memory Leak warnings

Q. Should a PT attempt to use Valgrind/Helgrind on C/C++ code? (UNICORE)

A. No.

- Its a big overhead on the build system and is hard to interpret as a metric. It also requires direct intervention by the developers to include relevant header files into the code and tuning is needed.
- However, if a product team is suffering from memory leaks or threading problems that they can't track down, they should first use cppcheck to find easily spotted problems.
- If memory leaks still persist the PT should isolate the problem to a smaller code set, where Valgrind/Helgrind can be implemented.

SLOC (Source Lines of Code)

Q. Why define SLOC? (dCache)

A. For other metrics

- SLOC on its own is almost useless and difficult to interpret over time (even in trend graphs)
- However, when used in conjunction with Backlog management index or Bug Density distribution it becomes useful for comparing the size of product teams and the number of bugs open against them.

-- EamonnKenny - 13-May-2011

This topic: EMI > EmiSa2MetricsQandA
Topic revision: r4 - 2011-05-20 - unknown