Metric Review Q&A

Presenter: Eamonn Kenny

Outline

- Feedback: general remarks
- Metrics in the Metrics Report
  - Successful builds
  - SLOC count
  - Backlog management
  - Priority PMD/checkstyle violation density
  - Findbugs error density
  - Bug density distribution
  - Time to close bugs
  - Cumulative time to close bugs
  - Open bugs
  - Untouched bugs
  - Fixed bugs
- Metrics in the SQAP
  - Process metrics
    - Priority bug (time to handle a bug)
    - Fixed bugs
    - Open bugs
    - Untouched bugs
    - Bug severity distribution
    - Backlog
    - Successful build
    - Integration test effectiveness
    - Up-to-date documentation
    - Delay on release schedule
  - Product metrics
    - Unit test coverage
    - Supported platforms
    - Total bug density
    - Bug density per release
  - Static analysis
    - Cyclomatic complexity
    - C/C++
    - Java
    - Python
- New metrics

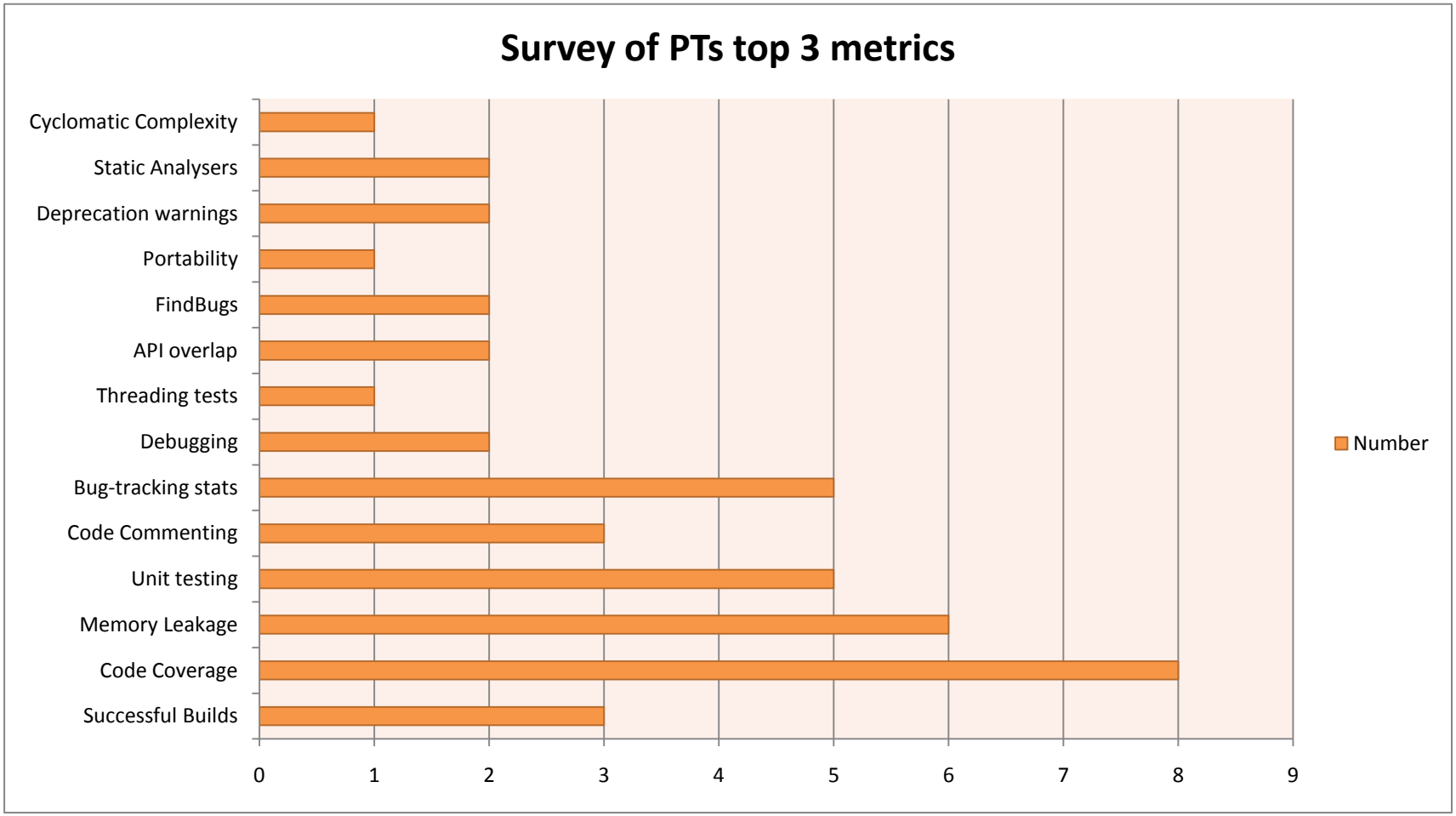
Feedback: general remarks

- UNICORE: Code metrics are based on ETICS but not all code is built with ETICS.
  - Ongoing discussion going on this point between UNICORE and SA2.4
- UNICORE: Metrics should have a priority so that PTs know where to put more effort. It's not the same checkstyle than findbugs.
  - Priority in metrics, look at top 3 metrics
- ARC: It's not clear who are the consumers of the metrics report.
  - SA2, JRA1 PTs, SA1 QC, JRA EMT, PEB, EU reports/deliverables

Feedback: general remarks

- ARC: Current report is too long and too frequent.
  - Frequency is currently for reassessment purposes internally.
  - EMT report must be weekly, not true for others.
- dCache: only make sense to PT if they know implications derived from metrics, and what metrics are exactly needed for.
  - Quality product is the end goal, harmonisation, reduced bugs
- gLite: dashboard with metrics results needed.
  - Reports only
- SA1 QC: prioritise metrics.

Top 3 Metrics



1. Code Coverage. 2. Bug-tracking Statistics 3. Memory Leaks

Proposed Metrics Priorities in Reports

1. PTs/EU: FindBugs high priority bugs and cppcheck errors (not warnings)
   - Generate filters over time to remove false positives
   - Reducing over course of year 2 (threshold?)
2. SA2/EMT/Bug-tracking: Full life cycle of Immediate and High priority bugs
   - (open→accepted/rejected→fixed→tested/not)
3. PTs/EU reports/SA2/SA1: Coverage testing

Periodicity is important & customer specific

Metrics in the Metrics Report

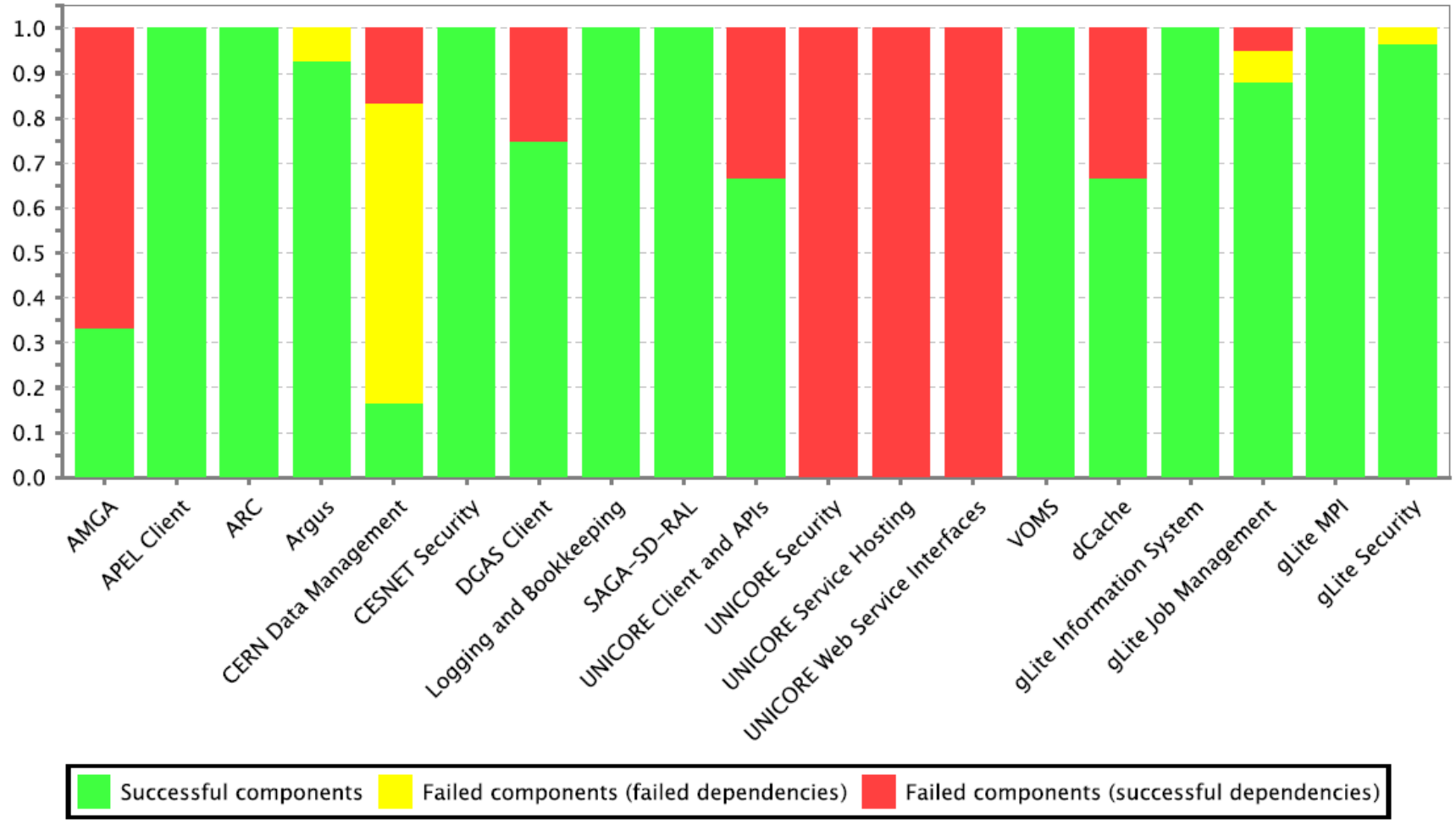
- Successful builds
- SLOC count
- Backlog management
- Priority PMD/checkstyle violation density
- Findbugs error density
- Bug density distribution
- Time to close bugs
- Cumulative time to close bugs
- Open bugs
- Untouched bugs
- Fixed bugs

Individual Metrics

- For each metric we now present:
  - Feedback from developers
  - Our comments and some actions

Successful builds

- No comments.



SLOC

- ARC: Doesn't need to be calculated every night. Not even every week. No useful information for daily development.
  - It's calculated every week, but that's too much for PTs.
- dCache: little variations throughout EMI, so what does it really mean?
  - Will become more important in harmonisation tasks. Code will appear/disappear. Needed for other metrics.

Backlog management

- ARC: Wrong definition. *backlog* is not the open_bugs – closed_bugs, but number of non expected open bugs after the grace period, which can vary per component.
  - Correct! But this is BMI not Back Log
  - It doesn't grow over time by default. E.g: If you close more bugs than you open in a month, then it decreases.

Error or Style violation density

- ARC: useful to keep code clean but doesn't need to be evaluated nightly or weekly. Not helpful in bug fixing.
  - Totally agreed. Less periodic reports required.
- dCache: % of comments is not really useful since it doesn't check the quality of the comments.
  - This points to priority. Some items more useful than others.
- gLite: priority to Javadoc.
  - Agrees with previous comment. Need tailored CheckStyle.

Error or Style violation density

- UNICORE: PMD lower importance for Java. Use CPD. Filter warnings. Too many! Different type of checks; Checkstyle minimal importance for Java. It doesn't help bug detection. Rules not clear. Too many errors of little importance. No worth the effort of fixing them. Use javadoc or remove metric.
  - Not sure that CPD is going to improve this. However, it does need some investigation.
  - Agree the point about too many warnings.
  - Will consider javadoc. Needs to be more tailored and analysed manually from time to time.

Findbugs

- dCache: they already run it and improve code when feasible. So metric report doesn't bring anything extra.
  - Can we get the information out of your findbugs
  - Can you try tuning your code with the variables?
- gLite: focus on java configuration that developers find useful. Same for other languages.
  - ??

FindBugs

- UNICORE: no threshold given. Sometimes the give false-positives. Proposed formula is incomplete. Output is quite short. Proposal to change metric.
  - Include high priority in year 2.
  - We will analyse to find false positives.
  - We will attempt to remove meaningless info.

Time to close bugs

- ARC: doesn't need to be calculated nightly. When to start evaluating it?
  - Monthly, quarterly?
- Better *time to resolve a bug.* But does resolve mean the same for all middlewares?
  - No. We are not so interested in the closed state, but when defects/features are fixed/available or released.

Time to close bugs

- dCache: manual intervention is needed to explain certain deviations. Is this feasible?
  - Yes, the reports will be analysed with a synopsis given for each. This includes the EMT report presented each week...

Cumulative time to close bugs

- ARC: not useful.
- gLite: is the meaning of closed bug the same for all mw? Better differentiate: open to fix and to certified and then the time to go to production.

  – Drop this, we never had it defined as a metric anyway. It came out of discussions at the metric meetings.

Open bugs

- ARC: open bugs over period of time and time to solve useful but definition contradictory. Is open how can you calculate time to solve?
  - Can be opened in a previous period.
- Comparison with previous periods may be interesting.
  - Will be covered by lifecycle of a bug plots

Open Bugs

- UNICORE: bug density (bugs per KLOC) not useful.
  - The issue raised here was refactoring.
  - Refactoring in some period, reducing lines of code, requires a parallel plot of “total bug”
  - After refactoring, if refactoring is good, bug density reduces, otherwise it increase (and sometimes more rapidly).
  - Do we really need Bug Density in light of the average bug life cycle per product team/product?

Untouched bugs

- ARC: useful. Moreover it helps to monitor whether priorities are properly assigned.
  - Untouched immediate priority value of 2 weeks defined in recommendations of DSA1.1 is too high for immediate priority bugs!!!

Fixed bugs

- ARC: is it properly calculated? No double counting?
  - Not sure here?

Metrics in the SQAP

- Process metrics
  - Priority bug (time to handle a bug)
  - Fixed bugs
  - Open bugs
  - Untouched bugs
  - Bug severity distribution
  - Backlog
  - Successful build
  - Integration test effectiveness
  - Up-to-date documentation
  - Delay on release schedule

Metrics in the SQAP

- Product metrics
  - Unit test coverage
  - Supported platforms
  - Total bug density
  - Bug density per release
- Static analysis
  - Cyclomatic complexity
  - C/C++
  - Java
  - Python

UNICORE feedback

- Integration test effectiveness: Impossible to calculate in UNICORE.
  - If junit and code coverage are such well done takes then why is integration test effectiveness so difficult?
- Bug density per release: difficult to calculate.
  - Should probably be per release only. Frequent plots of this are not useful.
- Cyclomatic complexity: how is it going to be normalised? As it is now doesn't make sense.
  - Its an indicator more than a point for action.
  - Very high cyclomatic complexity may require action.

New metrics

- dCache: functionality duplication in EMI.
  - CPD needs some investigation.
- dCache: Testing coverage results.
  - It seems it's a high priority on the PTs list.
  - Is it an aspiration or a value?
  - We will obtain the results manually or extra with ETICS?

QA feedback

- Inform PEB which metrics can't be calculated because of limitations in trackers, etc.
- Separate reports for EC, SA2, SA1 QC, EMT (JRA1) and PTs (JRA1) .
- GGUS has to be used!

EMT Reports Proposal (1)

Frequency

- Every Friday for every Monday

Major Changes

- Remove cumulative days. Feedback poor!
- Remove medium bugs. Is it a priority?
- Remove scatter plots? Wasteful plot?
- Remove number of bug plots
  - Add cardinality to averages plot? Wasted plot?

Remove All the above. Accepted!

EMT Reports Proposal (2)

Prioritized Listing

1. Successful Builds (plot only)
2. Untouched Immediate/High Priority Bugs
   (a) Plot first
   (b) listing second
3. Accepted Immediate/High Priority Bugs
   (a) plot first
   (b) listing second

5 sections in all

PT Reports Proposal (1)

Frequency

- Every Month or every Quarter?

Major Changes

- Remove cumulative days. Feedback poor!
- Remove medium bugs. Is it a priority?
- Remove scatter plots? Wasteful plot?
- Remove number of bug plots
  - Add cardinality to averages plot? Wasted plot?
- Define better priorities

PT Reports Proposal (2)

Prioritized Listing

1. Static Analysers: High Priority FindBugs, PyLint & cppcheck errors
2. Average Bug Lifecycle per product team
   - Breakdown per product?
   - Breakdown per component?
3. Other analysers:
   - Tailored PMD, checkstyle

References

Reports

http://etics-repository.cern.ch/repository/reports/name/emi-qa-reports/-/reports/index-custom-QA-reports.html

Detailed Reports

http://etics-repository.cern.ch/repository/reports/name/emi-qa-reports/-/reports/index-custom-QA-reports.html

EMT Reports

http://etics-repository.cern.ch/repository/reports/name/emi-qa-reports/-/reports/EticsQAReports/EMT_report.pdf

EU Reports

http://etics-repository.cern.ch/repository/reports/name/emi-qa-reports/-/reports/EticsQAReports/EMI-METRICS-REPORT_EU.odt