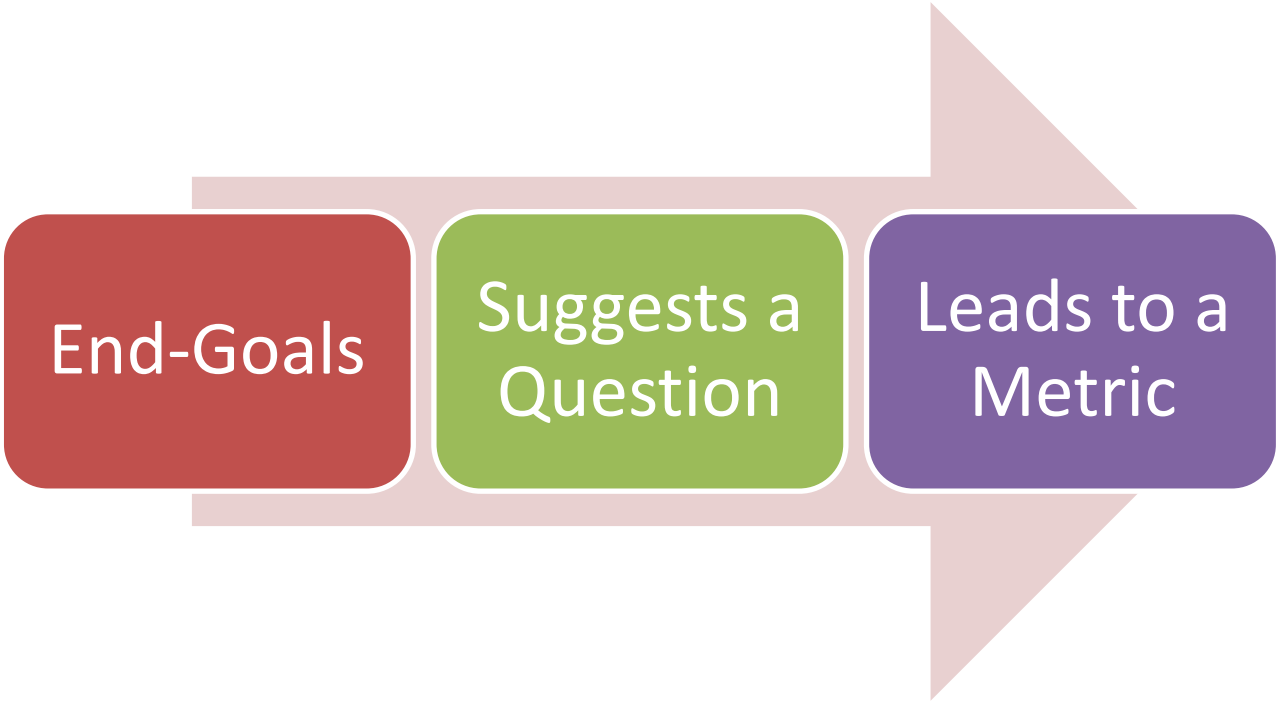
SA2.3 Metrics Report

Eamonn Kenny (TCD)

Gianni Pucciani (CERN)

Why collect Metrics?

- For goodness sake! Not for statistics sake!
- Governed by the end goals of the project



Metrics Categories

- Process Metrics (Bug-tracking related)
  - Related to priority, severity, open/closed bugs
- Quality in Use Metrics (Bug-tracking/EGI)
  - 3rd level GGUS related metrics (KPIs for SA1)
- Product Metrics (Software related)
  - Unit tests, supported platforms, bug density
- Static Analysis Metrics (Static analysers)
  - Cyclomatic Complexity, Code commenting, etc.
- Optional Metrics (Valgrind/Helgrind)

Process/Bug-tracking related Metrics

- Priority Bugs (closed), Open Priority Bugs
  - Graphs of immediate and high priority bugs
- Bug Severity Distribution
- Failed Builds metric
- Backlog Management Index
- Integration test effectiveness metric
- Up-to-date documentation metric
- Delay on release schedule metric
- Total user incident metric (KSA1.1)
- Training and support incident metric
- Average time to deal with an incident (KSA1.2)
- User Support level ticket distribution

Product Team (PT) related Metrics

- Unit test coverage metric
- Number of supported Platforms
- Total bug density
- Bug density per release
- Cyclomatic complexity
- C/C++ metrics - cppcheck
- Java metrics - FindBugs, PMD, Checkstyle
- Python metrics - pylint
- Code commenting metrics
- Memory Leakage warnings (**optional**)
- Open POSIX test suite metric (**dismissed**)

Metrics Description Layout

- Metrics Id
  - E.g: PriorityBugs
- Name
- Description
- Measurement Calculation
  - Complete formula
- Input(s) & Units
  - E.g: time range in days
- Output(s) & Units
  - E.g: Average time in hours
- Scope
- Thresholds/Target Value
- Tools
- Availability
  - Per middleware availability
- Goals
- Quality factor
  - Follows the McCall factors
- Risks
- Special Notes

Thresholds

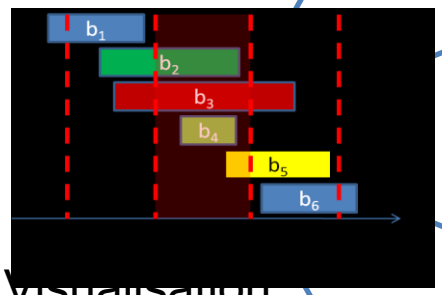
Thresholds in many cases don't make sense yet! Why?

- We are not starting afresh to produce a middleware stack → Some product teams will have good/poor statistics starting off
- Some programming languages strongly support static metrics
- Some bug-trackers are full of features
- Interoperation & reusability only beginning
- We expect improving trends…

Metric Guidelines & Repository

- Twiki Page
  - https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2MetricsGuidelines
- Subversion Repository
  - https://svnweb.cern.ch/trac/emisa2/browser/metrics/trunk
    - Internal metrics documentation directory
    - XML schemas (XSDs) directory
    - Per middleware Bug-Listing (in XML format)
    - Per middleware Bug-Mapping (in XML format)
    - Scripts to convert bug-tracking data into XML format, full validated by XML schemas

Why bug-tracking info in XML?



Visualisation

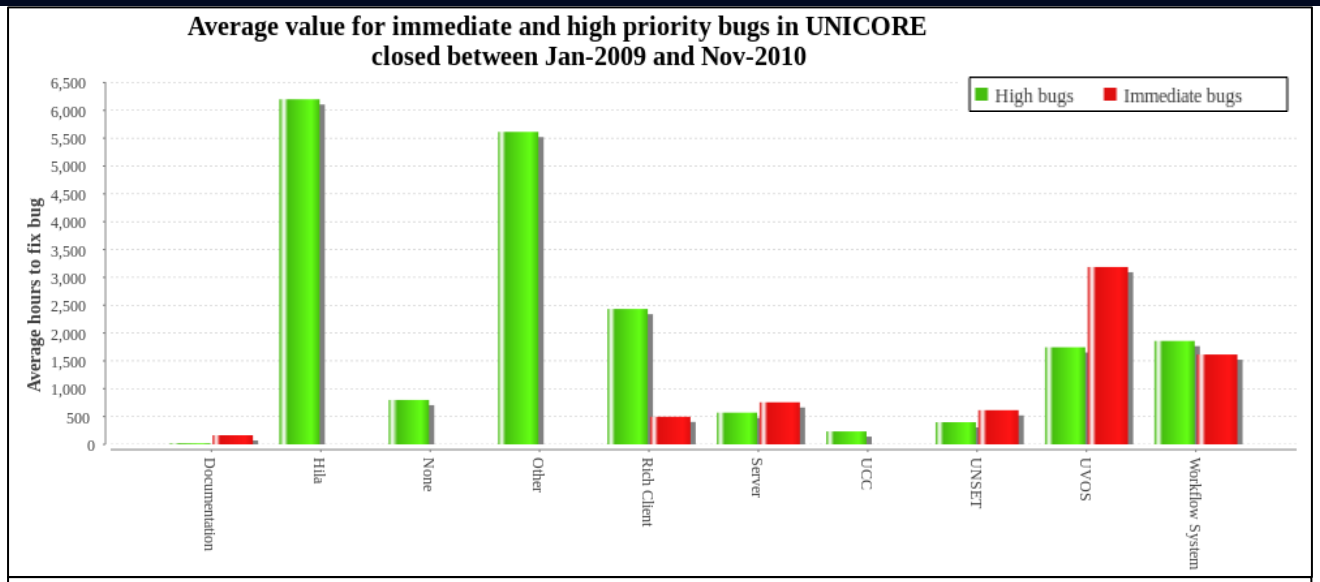
- ARC
  - Bugzilla
  - Bug-Mapping XML
- dCache
  - RT/Review Board
  - Bug-Mapping XML
- gLite
  - Savannah
  - Bug-Mapping XML
- UNICORE
  - SourceForge
  - Bug-Mapping XML

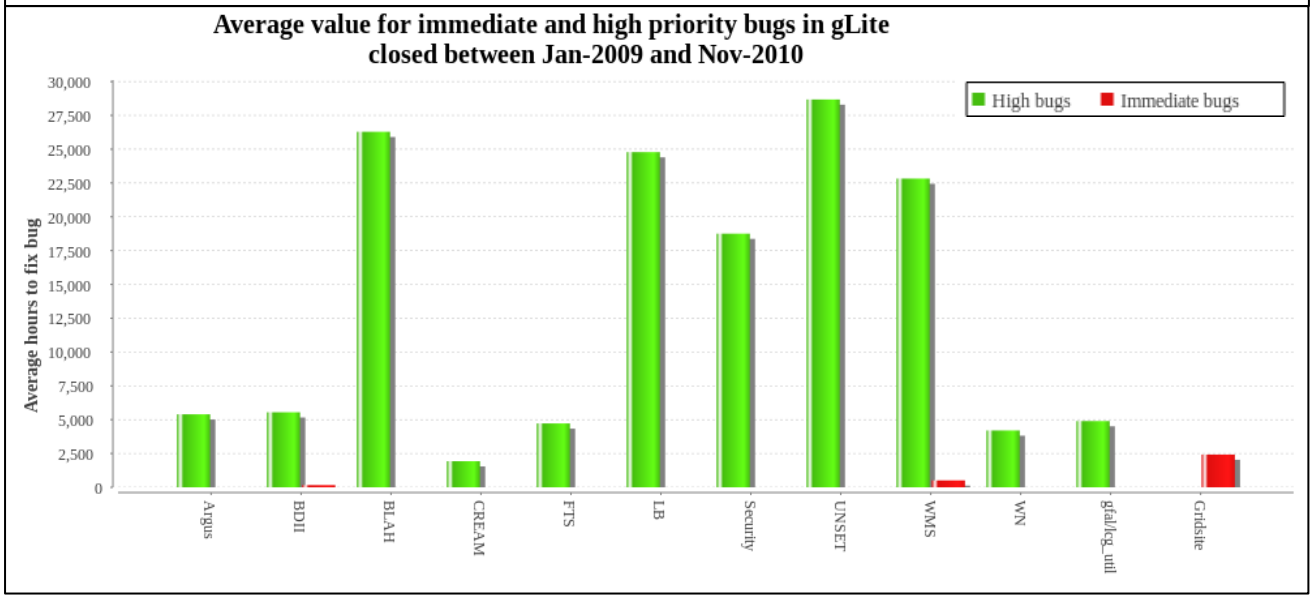
We want a tool that uses all bug-trackers

- Exposed XML to ETICS (run anytime)
- Middleware Bug-Mapping (needs periodic intervention)
- Middleware Bug-tracking (Snapshot taken daily)

We need a common interface

Sample Metric Plots of Priority Bugs





Achieved so far

- Bug Listing XSD, Bug Mapping XSD, XML examples, python sample scripts defined for extracting bug-tracking data
- Validated UNICORE and gLite XML listings
- Prototype ETICS plots/statistics for process and static analyser metrics
- Pseudo code made available to SA2.4 to produce metric calculations

Pending/Outstanding Issues

- 2 of 4, GGUS 3rd level user incident metric reports are still to be produced
- Awaiting dCache-BugListing.xml and ARC-BugListing.xml for combined bug-tracking metrics
- Metrics may need to evolve over time to track interoperation & reusability

QUESTIONS?