EUROPEAN MIDDLEWARE INITIATIVE

QA METRICS REPORT

EU DELIVERABLE: DXN.M

Document identifier:	EMI-METRICS-REPORT_EU
Date:	23/02/2011
Activity:	
Lead Partner:	
Document status:	
Document link:	

Abstract
This document provides a report on the metrics collected within the SA2 activity. It provides information to the Project Executive Board and other project decisional bodies on the status of the software as an instrument to take corrective actions
.

Delivery Slip

	Name	Partner / Activity	Date	Signature
From		SA2	01/02/11	
Reviewed by				
Approved by				

Document Log

Issue	Date	Comment	Author / Partner
1			
2			
3			

Document Change Record

Issue	Item	Reason for Change
1		
2		
3		

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to provide information related to the quality of the processes used to develop, certify and release the EMI software and the quality characteristics of the software itself. This information should provide the base on which corrective actions are taken, with the final goal of improving the processes and the quality of the products delivered by EMI.

1.2. DOCUMENT ORGANISATION

Chapter 2 provides a high-level summary of the content of the document.

Chapter 3 presents the results of *process* related metrics; for each metric one or more charts are presented to offer different perspectives of the collected data.

Chapter 4 presents the results of *product* related metrics; for each metric one or more charts are presented to offer different perspectives of the collected data.

Chapter 5 is reserved to *quality in use* metrics, which are not yet available.

Chapter 6 presents the conclusions that can be made from a first analysis of the data presented in Chapter 4 and Chapter 5.

1.3. REFERENCES

R1 https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2ChangeManagementGuidelines

R2 The EMI Project Description of Work, EMI, April 2010

https://twiki.cern.ch/twiki/pub/EMI/EmiDocuments/EMI-Part_B_20100624-PUBLIC.pdf

R3 DSA1.1 - Software Maintenance and Support Plan, EMI, to be released

http://cdsweb.cern.ch/record/1277556

1.4. DOCUMENT AMENDMENT PROCEDURE

This document can be amended by the authors further to any feedback from other teams or people. Minor changes, such as spelling corrections, content formatting or minor text re-organisation not affecting the content and meaning of the document can be applied by the authors without peer review. Other changes must be submitted to peer review and to the EMI PEB for approval.

When the document is modified for any reason, its version number shall be incremented accordingly. The document version number shall follow the standard EMI conventions for document versioning. The document shall be maintained in the CERN CDS repository and be made accessible through the OpenAIRE portal.

1.5. TERMINOLOGY

KLOC Thousands lines of code.

2. EXECUTIVE SUMMARY

In this section some background information related to the organization of the metrics are provided.

2.1. METRIC CATEGORIZATION

According to the ISO 9126 standard, we separated the metrics in 3 basic categories:

1. External or process metric: concerning the bug tracking service for each middleware and associated turnaround times on fixing bugs. In addition, metrics for the reliability of the repository generation, install scripts usability and associated end user documentation are also collected.
2. Internal or product metrics: metrics derived mainly from static code analyzers, to show the presence of anomalies and the complexity of the code.
3. Quality in use or user incidents metrics: concerning GGUS user tickets related to the use of the EMI production infrastructure affecting both the software process and software development.

2.2. PROCESS METRICS

The bug-tracking process can be broken down into a number of sub-processes containing inputs, operations internal to each sub-process and measureable outputs. So taking the request for change (RfC) as in Figure below, there are clear transitional states that can be assessed in an attempt to produce some guidelines on how best to achieve timely reactions to each part of the process.

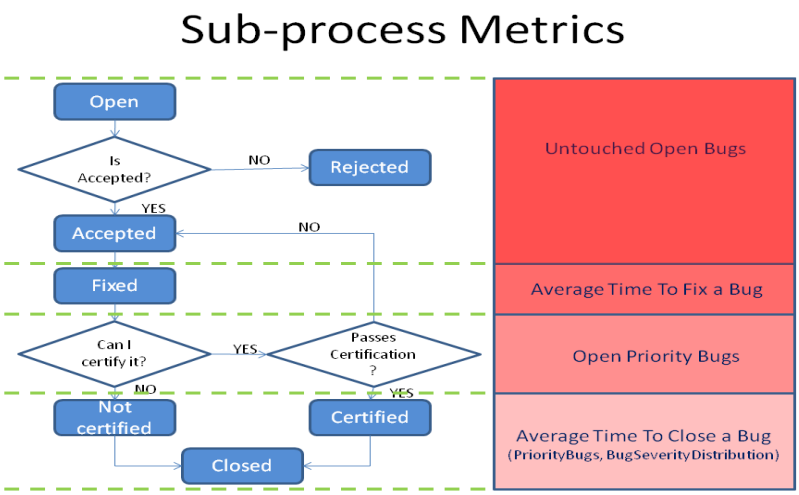


Figure : The process can be split into a number of distinct sub-processes

Purely looking at Figure in terms of streamlining the process to increase turnaround time in bug reporting and processing, it is clear that accessing the progression of bugs in terms of Table is advantageous.

Metric ID	*Starting State*	*Finishing State*	*Metric Description*	*Action to take*
Untouched Open Bugs	*Open*	*Rejected /Accepted*	Bugs in state open longer than 14 days without transitioning to state accepted/rejected must be highlighted per product category	*Speed up transition to accepted/rejected over time*
Fixed Bugs	*Open*	*Fixed*	The average time to fix a bug per product category	*Decrease average time in each quarter*
Open Priority Bugs	*Open*	*Accepted /Fixed*	Priority Bugs still open but not certified in a specific time period.	*Fix and certify priority bugs in a more timely fashion*
Priority Bugs / Bug Severity Distribution	*Open*	*Closed*	Average Time to deal with a closed bug of a particular type severity or priority	*Deal with Priority bugs/High severity bugs in a more timely fashion*

Table : Metrics related to sub-processes that appear in Section 5

Streamlining the overall process turnaround time is not sufficient. It's also very important to address bug-tracking incidences in terms of the categorizations starting with those deems most important.

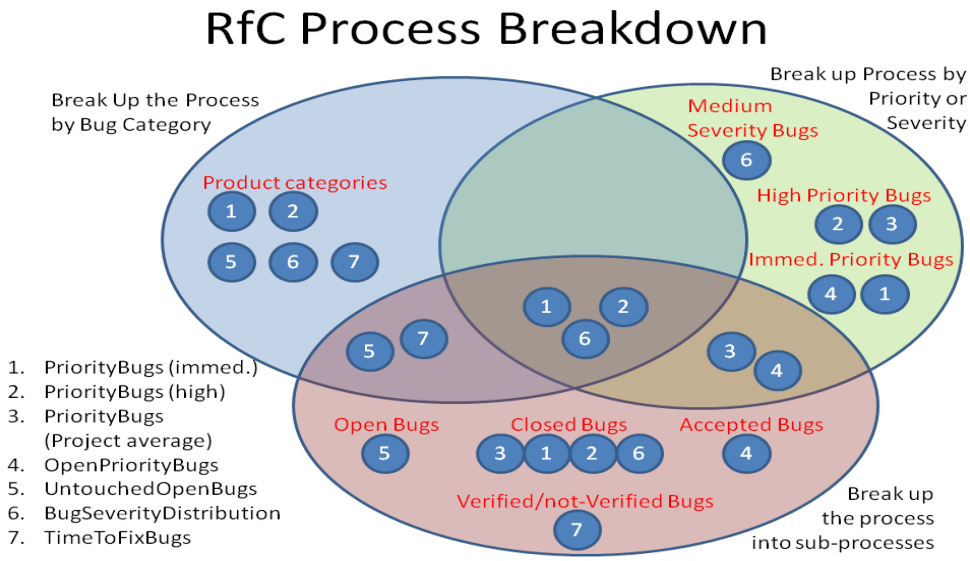


Figure : The process broken down into sub-processes/states, priorities/severities and product categories

Figure shows how the process can be broken down into particular classes of problem. Many of the metrics will be used to compare the results from different product teams (and their associated ‘categories’). Others will give preference to the immediate and high priority and severity bugs, since these are the most urgent pending issues that the middleware must address to provide a good quality service to the user. The breakdown into sub-processes as stated in Figure is suggests a very obvious way to streamline the process.

3. PROCESS METRICS

The goal of process metrics is to identify improvement opportunities in the way the software is developed and tested, from a process point of view. In our context the main sources of process metrics are the Defect Tracking tools used in EMI. A common definition for Defect Tracking states and transitions is available in the Change Management Guidelines [R1].

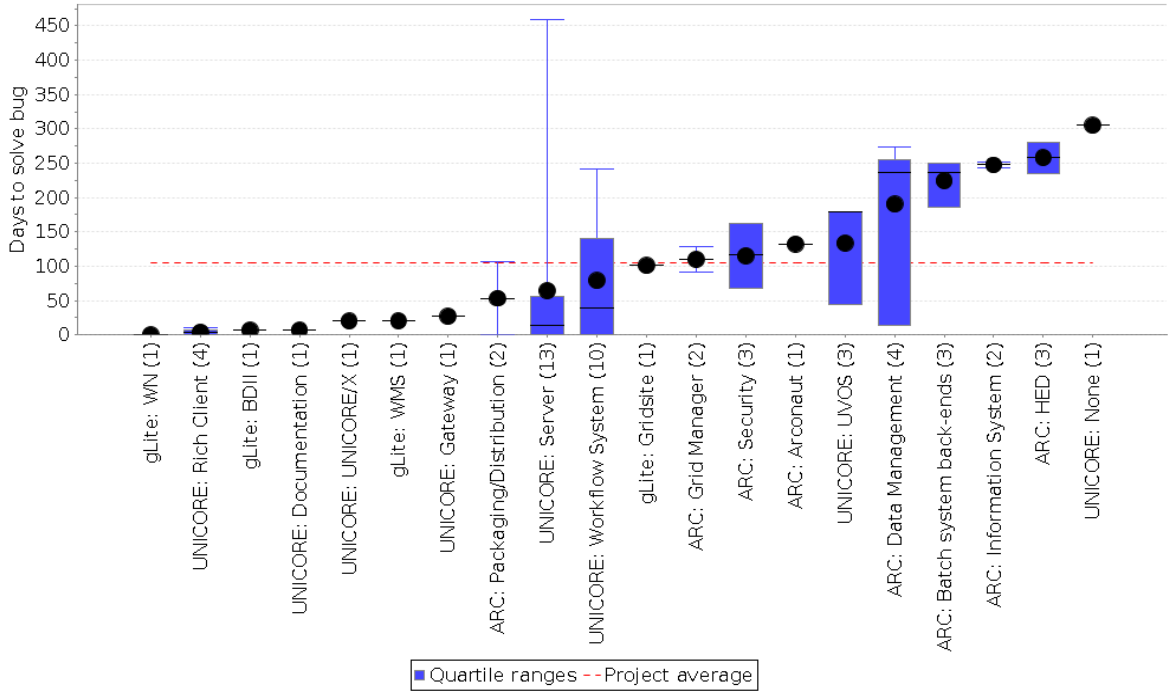
In some cases the metrics will be displayed using a type of box plots that is explained in Appendix A.

3.1. PRIORITYBUG

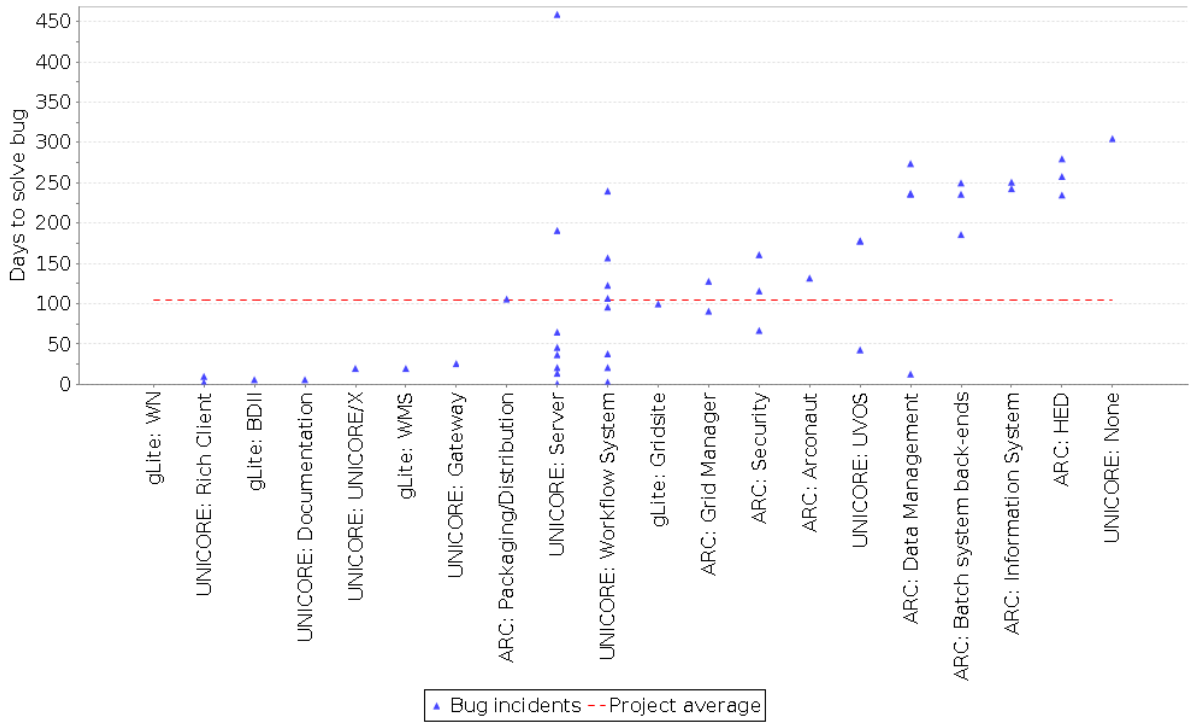
This metric describes the average time needed to provide a fix to a bug with a given priority. The time is calculated starting from the time a bug is submitted to the bug tracker, and the time the bug fix is closed.

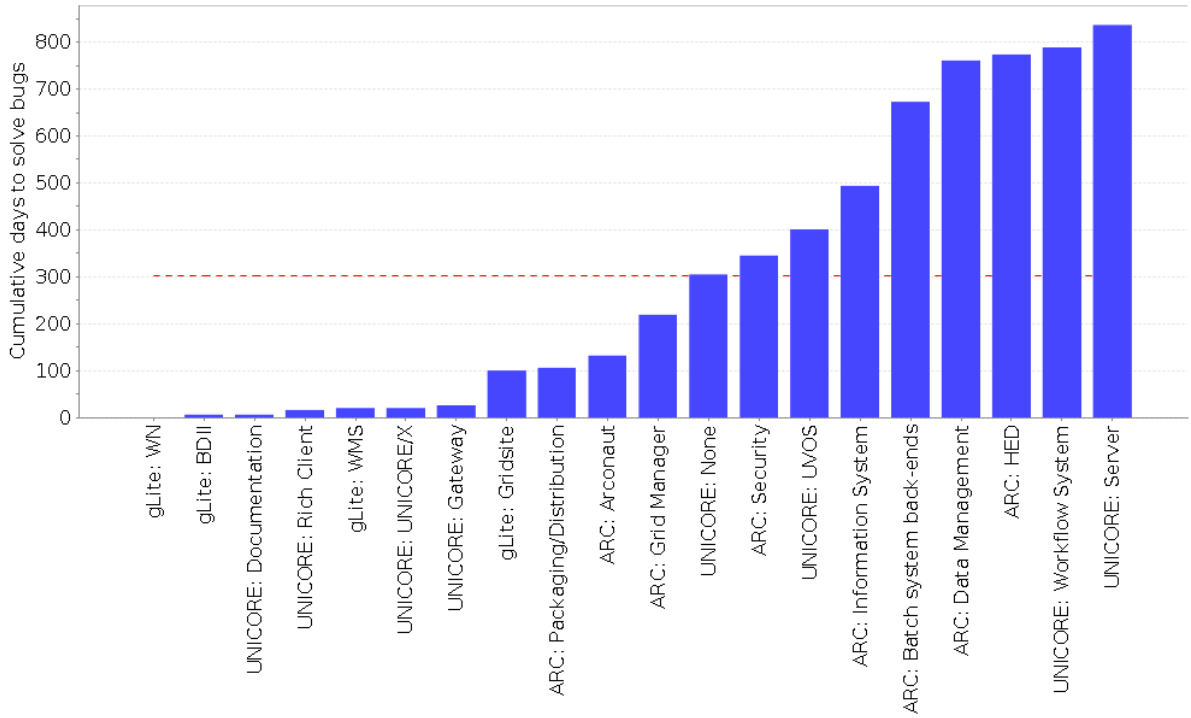
The metric will be calculated for three levels of priority: ‘Immediate’, ‘High’ and ‘Medium’ and it will be displayed for each Defect Tracker ‘Category’. The calculation is done considering all the bugs that have been closed within a specified interval.

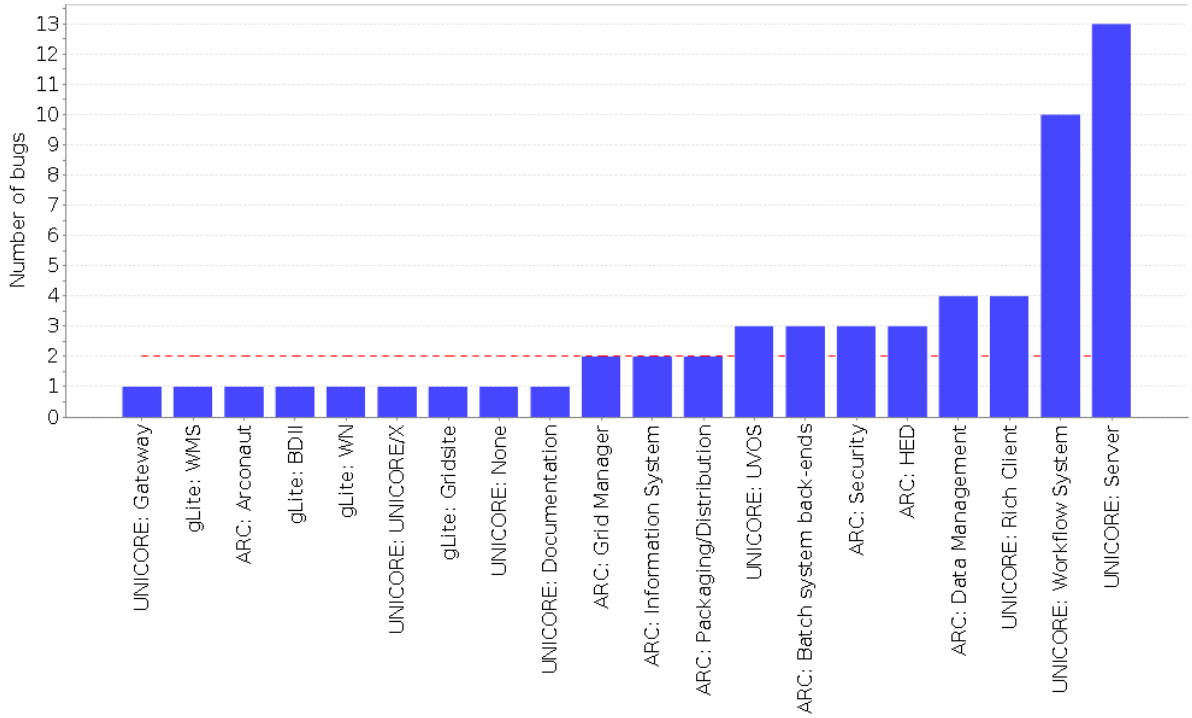
3.1.1 Immediate priority bugs closed between 1 January 2010 and 20 January 2011



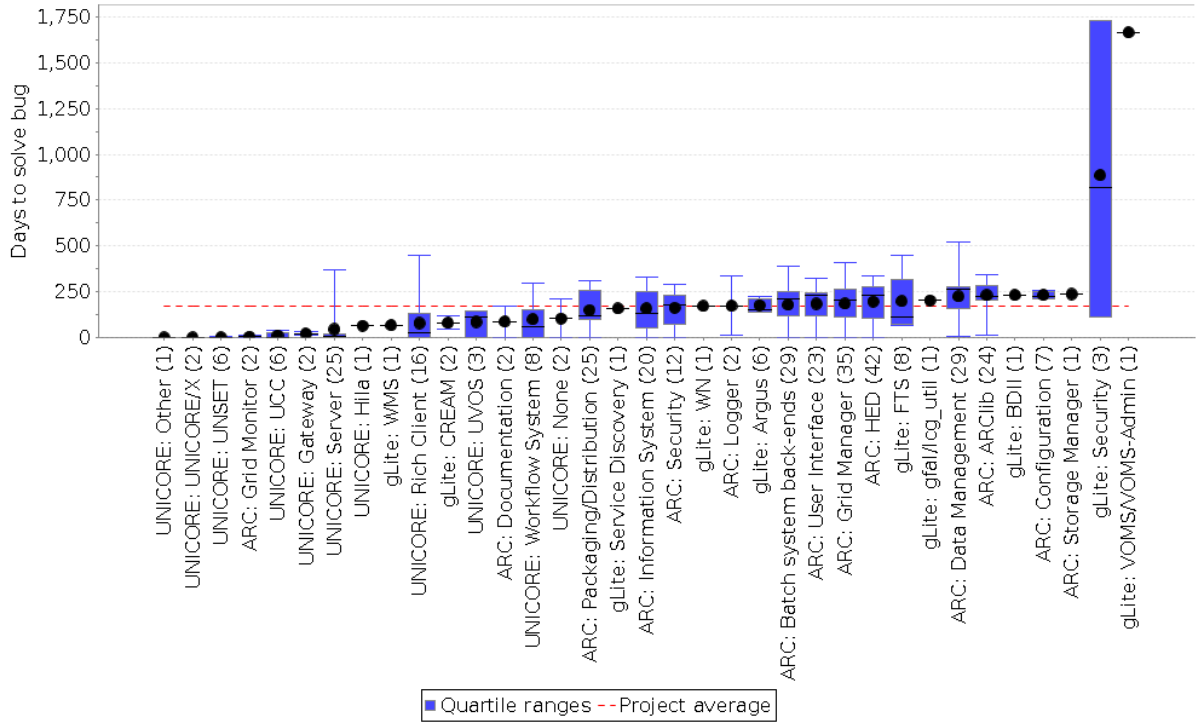
The box plot has the format explained in Appendix A, and it shows also the project average.



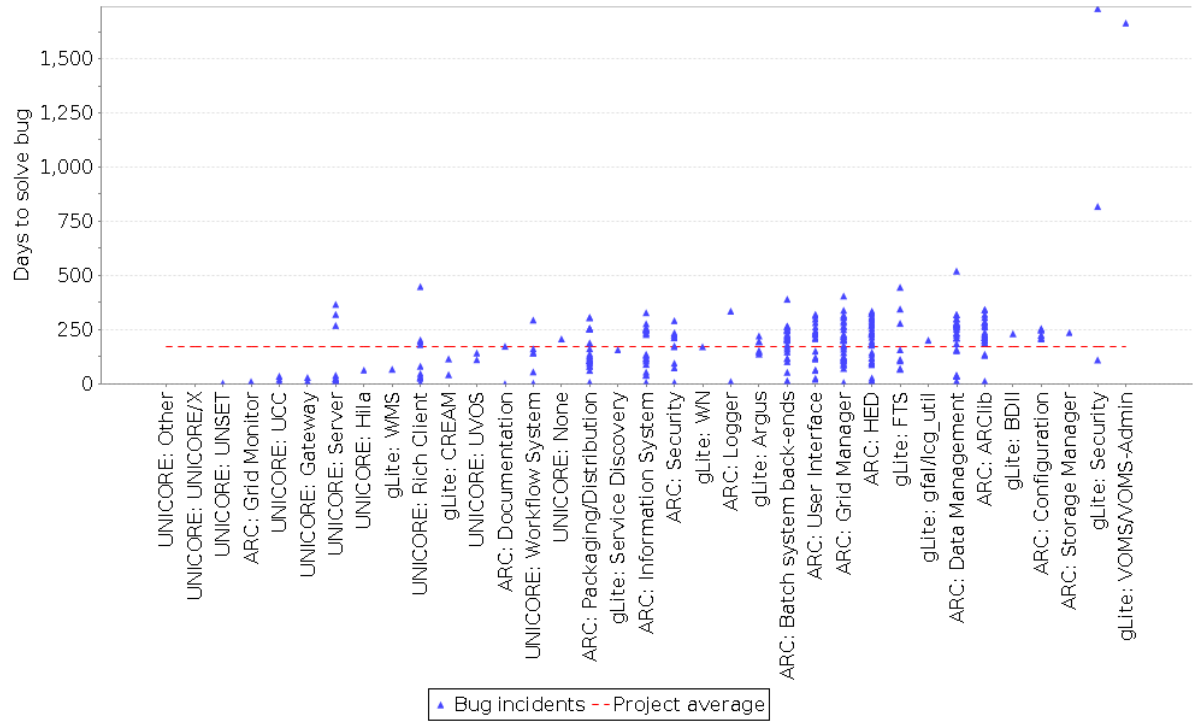

Cumulative days to solve bugs
0
100
200
300
400
500
600
700
800
gLite: WN
gLite: BDII
UNICORE: Documentation
UNICORE: Rich Client
gLite: WMS
UNICORE: UNICORE/X
UNICORE: Gateway
gLite: Gridsite
ARC: Packaging/Distribution
ARC: Arconaut
ARC: Grid Manager
UNICORE: None
ARC: Security
UNICORE: UVOS
ARC: Information System
ARC: Batch system back-ends
ARC: Data Management
ARC: HED
UNICORE: Workflow System
UNICORE: Server

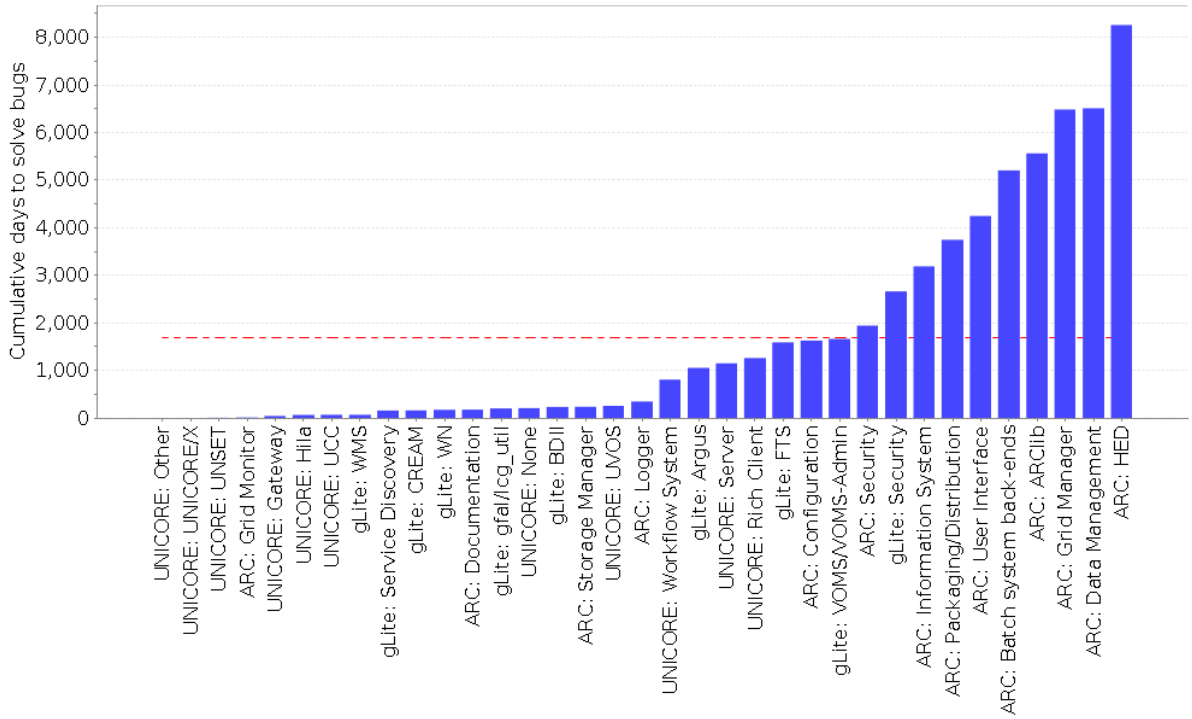

Number of bugs
0
1
2
3
4
5
6
7
8
9
10
11
12
13
UNICORE: Gateway
gLite: WMS
ARC: Arconaut
gLite: BDII
gLite: WN
UNICORE: UNICORE/X
gLite: Gridsite
UNICORE: None
UNICORE: Documentation
ARC: Grid Manager
ARC: Information System
ARC: Packaging/Distribution
UNICORE: UVOS
ARC: Batch system back-ends
ARC: Security
ARC: HED
ARC: Data Management
UNICORE: Rich Client
UNICORE: Workflow System
UNICORE: Server

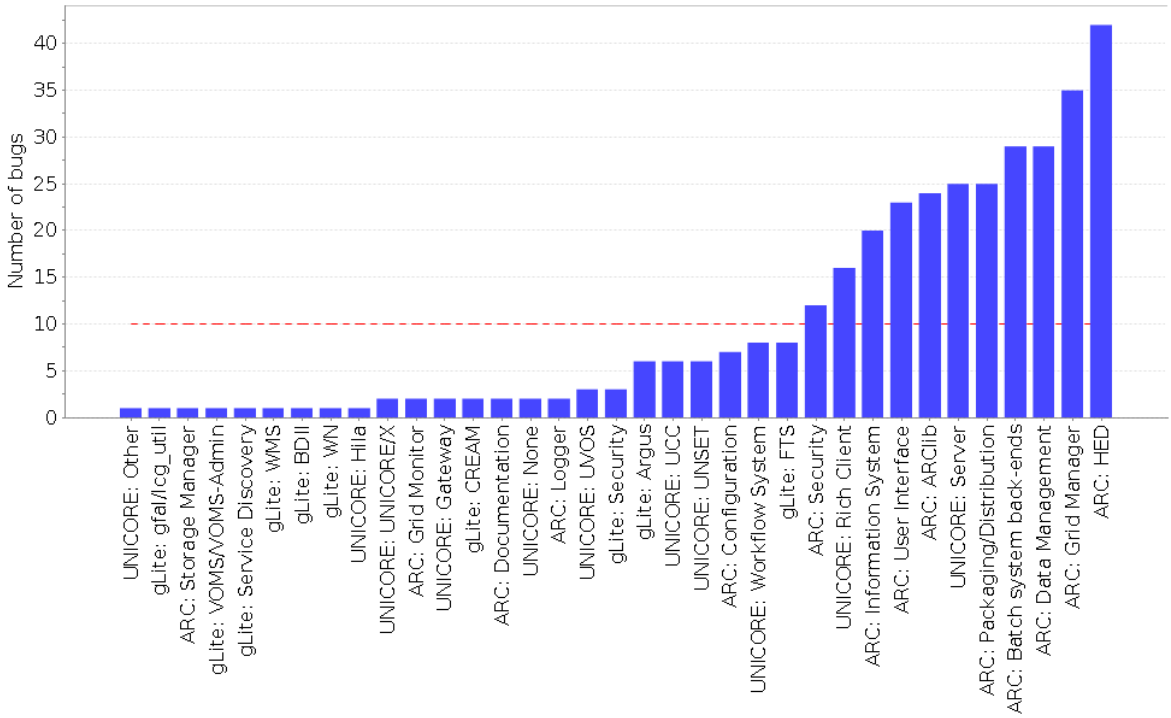
3.1.2 High priority bugs closed between 1 January 2010 and 20 January 2011



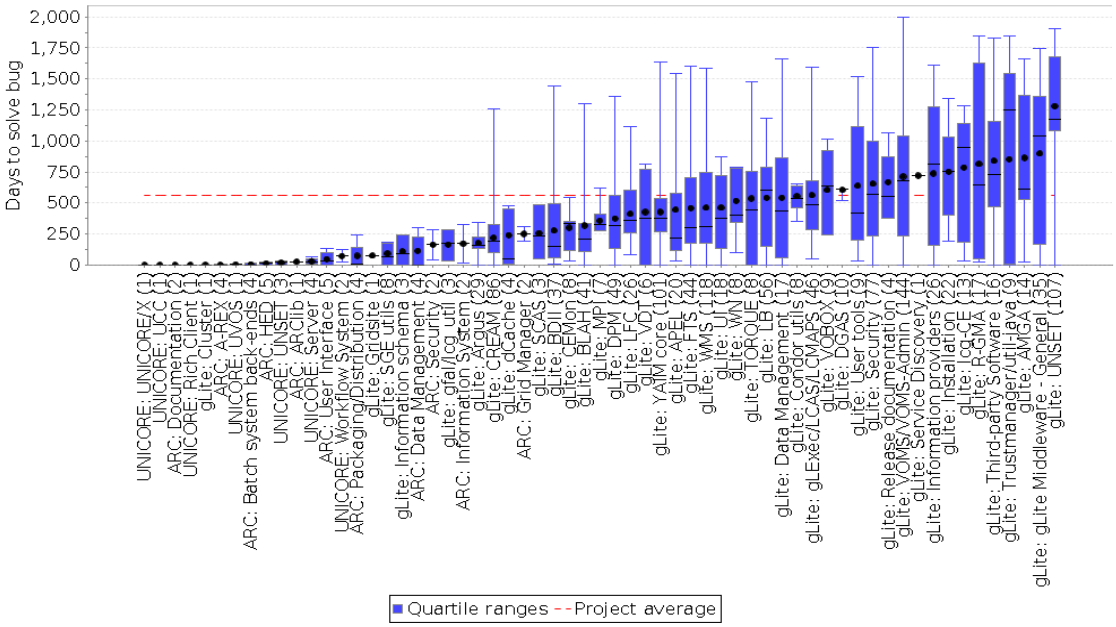
The box plot has the format explained in Appendix A, and it shows also the project average.



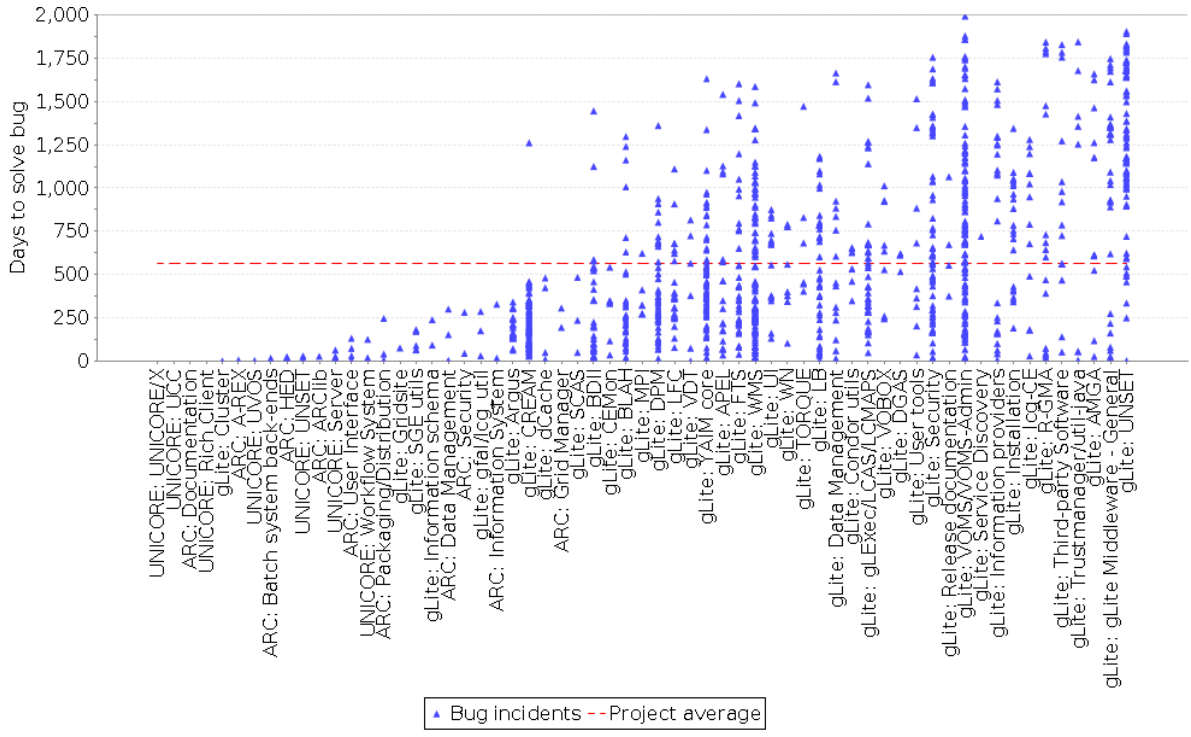

Cumulative days to solve bugs
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
UNICORE: Other
UNICORE: UNICORE/X
UNICORE: UNSET
ARC: Grid Monitor
UNICORE: Gateway
UNICORE: Hila
UNICORE: UCC
gLite: WMS
gLite: Service Discovery
gLite: CREAM
gLite: WN
ARC: Documentation
gLite: gfal/lcg_util
UNICORE: None
gLite: BDII
ARC: Storage Manager
UNICORE: UVOS
ARC: Logger
UNICORE: Workflow System
gLite: Argus
UNICORE: Server
UNICORE: Rich Client
gLite: FTS
ARC: Configuration
gLite: VOMS/VOMS-Admin
ARC: Security
gLite: Security
ARC: Information System
ARC: Packaging/Distribution
ARC: User Interface
ARC: Batch system back-ends
ARC: ARClib
ARC: Grid Manager
ARC: Data Management
ARC: HED

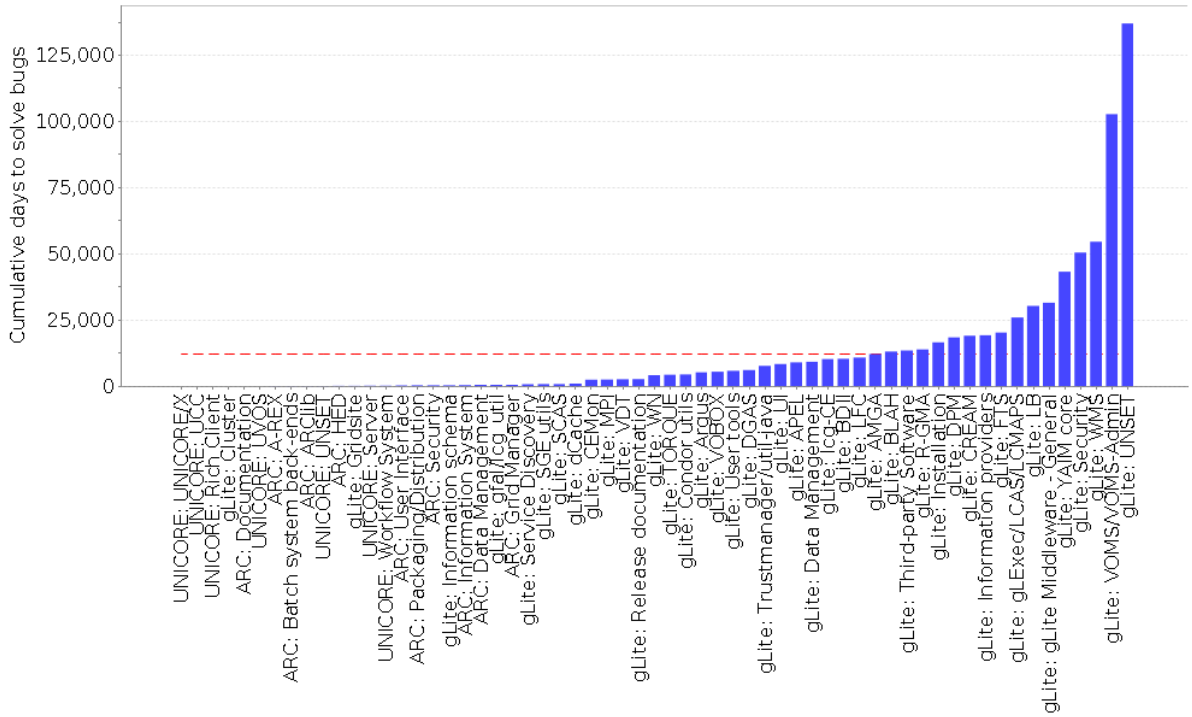

Number of bugs
40
35
30
25
20
15
10
5
0
UNICORE: Other
gLite: gfal/lcg_util
ARC: Storage Manager
gLite: VOMS/VOMS-Admin
gLite: Service Discovery
gLite: WMS
gLite: BDII
gLite: WN
UNICORE: Hila
UNICORE: UNICORE/X
ARC: Grid Monitor
UNICORE: Gateway
gLite: CREAM
ARC: Documentation
UNICORE: None
ARC: Logger
UNICORE: UVOS
gLite: Security
gLite: Argus
UNICORE: UCC
UNICORE: UNSET
ARC: Configuration
UNICORE: Workflow System
gLite: FTS
ARC: Security
UNICORE: Rich Client
ARC: Information System
ARC: User Interface
ARC: ARClib
UNICORE: Server
ARC: Packaging/Distribution
ARC: Batch system back-ends
ARC: Data Management
ARC: Grid Manager
ARC: HED

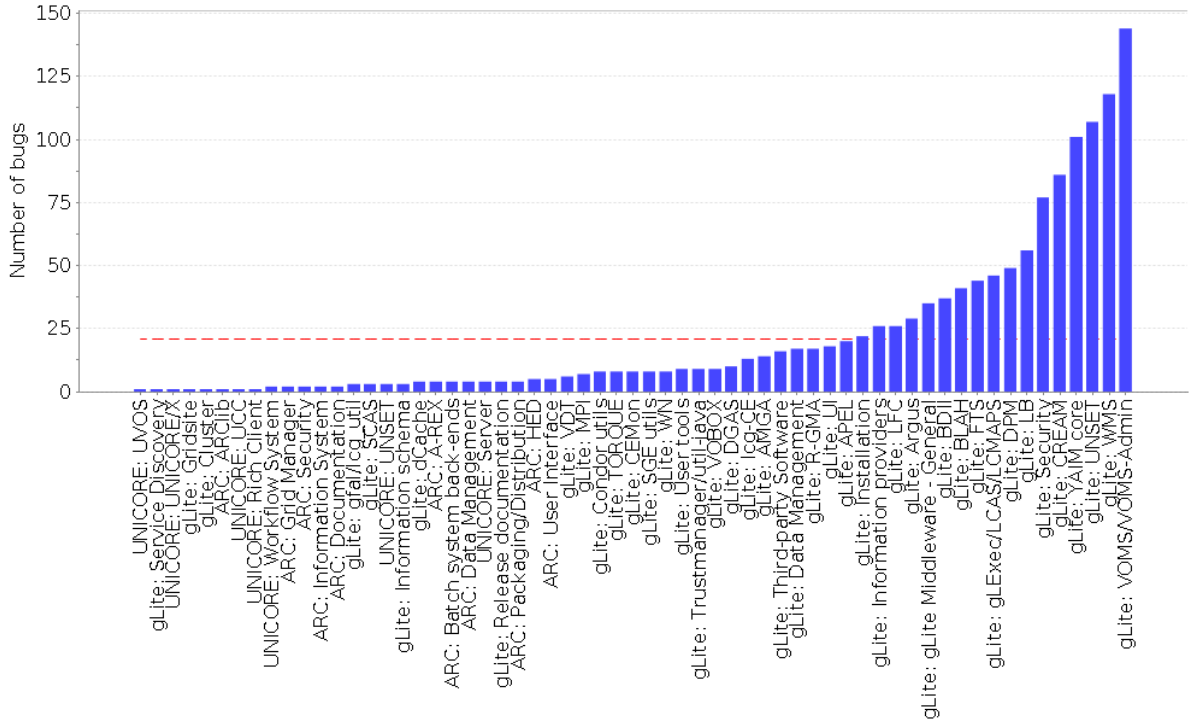
3.1.3 Medium priority bugs closed between 1 January 2010 and 20 January 2011



The box plot has the format explained in Appendix A, and it shows also the project average.




Cumulative days to solve bugs
125,000
100,000
75,000
50,000
25,000
0
UNICORE: UNICORE/X
UNICORE: UCC
UNICORE: Rich Client
gLite: Cluster
ARC: Documentation
UNICORE: UVOS
ARC: A-REX
ARC: Batch system back-ends
ARC: ARClib
UNICORE: UNSET
ARC: HED
gLite: Gridsite
UNICORE: Server
UNICORE: Workflow System
ARC: User Interface
ARC: Packaging/Distribution
ARC: Security
gLite: Information schema
ARC: Information System
ARC: Data Management
gLite: gfal/lcg_util
ARC: Grid Manager
gLite: Service Discovery
gLite: SGE utils
gLite: SCAS
gLite: dCache
gLite: CEMon
gLite: MPI
gLite: VDT
gLite: Release documentation
gLite: WN
gLite: TORQUE
gLite: Condor utils
gLite: Argus
gLite: VOBOX
gLite: User tools
gLite: DGAS
gLite: Trustmanager/Util-Java
gLite: APEL
gLite: Data Management
gLite: lcg-CE
gLite: BDII
gLite: LFC
gLite: AMGA
gLite: BLAH
gLite: Third-party Software
gLite: R-GMA
gLite: Installation
gLite: DPM
gLite: CREAM
gLite: Information providers
gLite: FTS
gLite: gLExec/LCAS/LCMAPS
gLite: LB
gLite: gLite Middleware - General
gLite: YAIM core
gLite: Security
gLite: WMS
gLite: VOMS/VOMS-Admin
gLite: UNSET


Number of bugs
150
125
100
75
50
25
0
UNICORE: UVOS
gLite: Service Discovery
UNICORE: UNICORE/X
gLite: Gridsite
gLite: Cluster
ARC: ARClib
UNICORE: UCC
UNICORE: Rich Client
UNICORE: Workflow System
ARC: Grid Manager
ARC: Security
ARC: Information System
ARC: Documentation
gLite: gfal/lcg_util
gLite: SCAS
UNICORE: UNSET
gLite: Information schema
gLite: dCache
ARC: A-REX
ARC: Data Management
UNICORE: Server
gLite: Release documentation
ARC: Packaging/Distribution
ARC: HED
ARC: User Interface
gLite: VDT
gLite: MPI
gLite: Condor utils
gLite: TORQUE
gLite: CEMon
gLite: SGE utils
gLite: WN
gLite: User tools
gLite: Trustmanager/Util-Java
gLite: VOBOX
gLite: DGAS
gLite: lcg-CE
gLite: AMGA
gLite: Third-party Software
gLite: Data Management
gLite: R-GMA
gLite: UI
gLite: APEL
gLite: Installation
gLite: Information providers
gLite: LFC
gLite: Argus
gLite: gLite Middleware - General
gLite: BDII
gLite: BLAH
gLite: FTS
gLite: gLExec/LCAS/LCMAPS
gLite: DPM
gLite: LB
gLite: Security
gLite: CREAM
gLite: YAIM core
gLite: UNSET
gLite: WMS
gLite: VOMS/VOMS-Admin

3.2. BUG SEVERITY DISTRIBUTION

Not available yet.

3.3. BACKLOG MANAGEMENT INDEX

Not available yet.

3.4. FAILED BUILDS

Not available yet.

3.5. INTEGRATION TESTS EFFECTIVENESS

Not available yet.

3.6. UP-TO-DATE DOCUMENTATION

Not available yet.

3.7. DELAY ON THE RELEASE SCHEDULE

Not available yet.

4. PRODUCT METRICS

Product (or internal) metrics measure certain characteristics of the product like complexity, changeability and testability. At the moment we rely on static code analyzers to measure these characteristics.

4.1. UNIT TESTS COVERAGE

Not available yet.

4.2. NUMBER OF SUPPORTED PLATFORMS

Not available yet.

4.3. TOTAL BUG DENSITY

Not available yet.

4.4. BUG DENSITY PER RELEASE

Not available yet.

4.5. STATIC CODE ANALYSIS

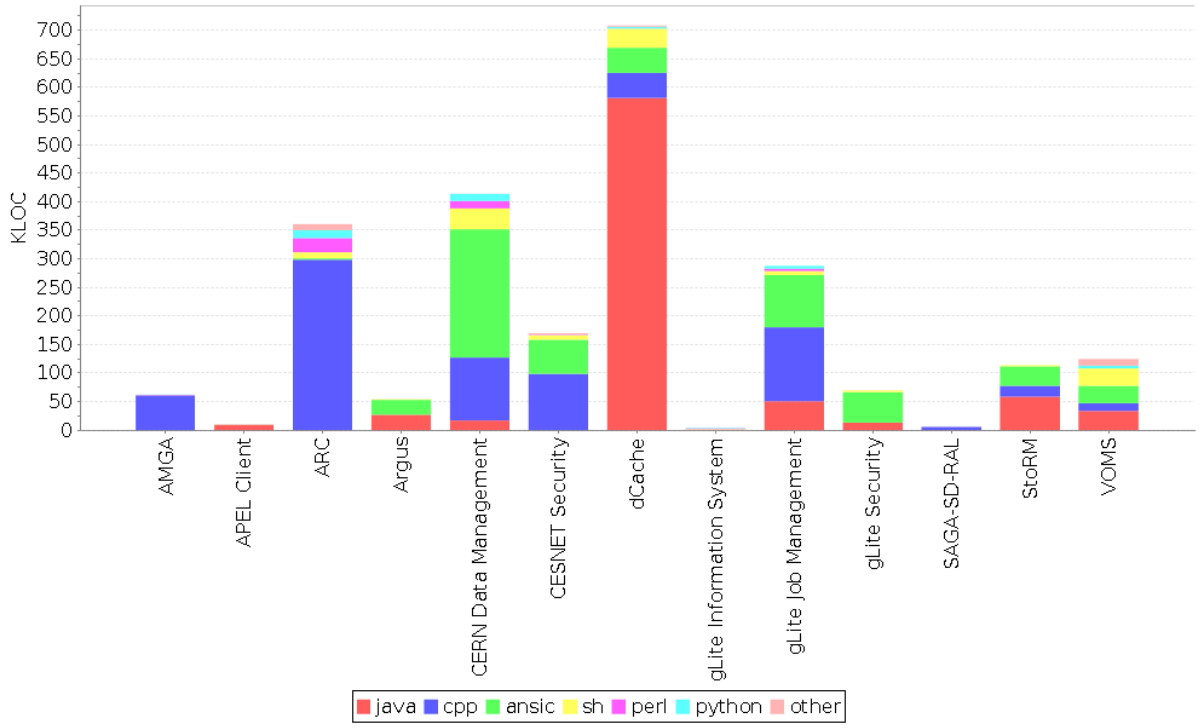
Static analysis metrics are metrics that can be collected analyzing the source code of a software project without executing it. Examples of such metrics are: Lines of code, Code Complexity, and Style checks. Some tools available for static analysis are also able to find bugs in the code by checking for specific patterns that can be configured before running the analysis.

Static analysis measurements are often subjective, and therefore difficult to apply in a distributed development environment with several programming languages. Nonetheless, they are very useful, and an effort should be done in order to promote their application in a development environment.

4.5.1 SLOC Count

This metrics shows, for each ETICS subsystem, the amount of lines of code and the programming language used.

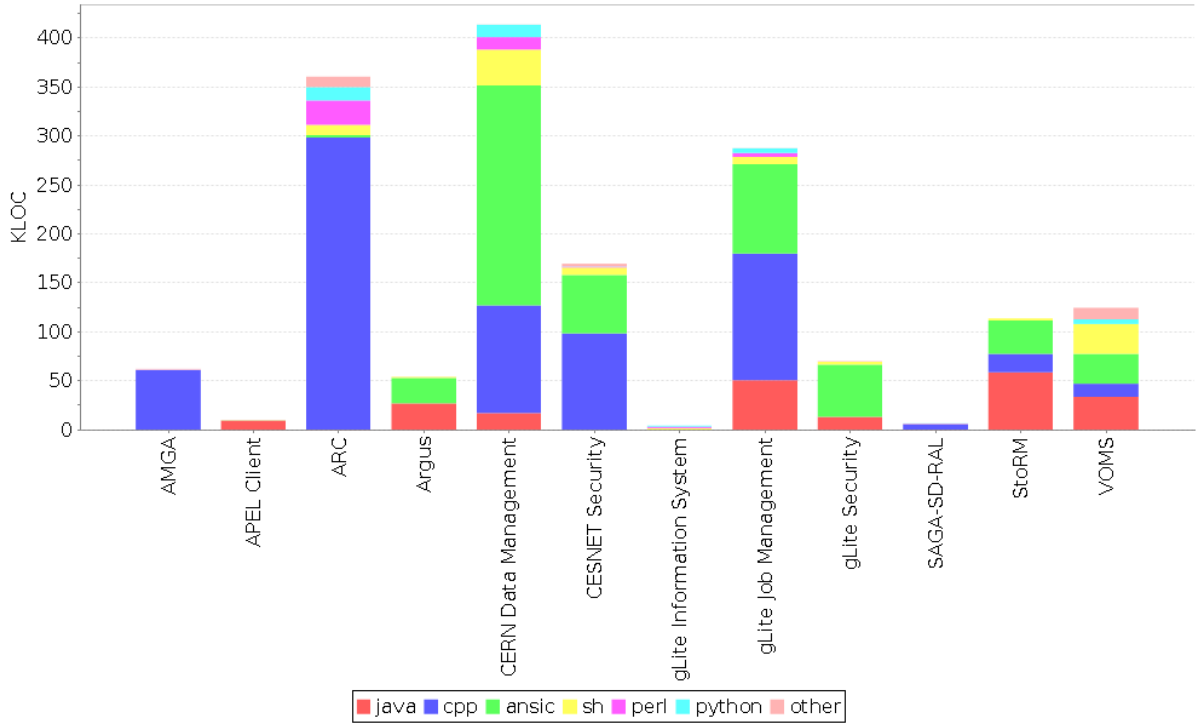
4.5.1.1 SLOC Count per subsystem at 20 January 2011.



Other languages: exp, php, jsp, yacc, lex, awk, csh, sed, ruby, lisp, fortran, pascal

4.5.1.2 SLOC Count per subsystem, zoomed at 20 January 2011.

This graph zooms into the lower part of the previous graph, clarifying the situation for subsystems that were not clearly visible in the previous graph.



4.5.2 Cyclomatic complexity

Short metric explanation and reference to more detailed info.

Not available yet.

4.5.3 C/C++

4.5.3.1 Cppcheck errors

Short metric explanation and reference to more detailed info.

Not available yet.

4.5.4 Java

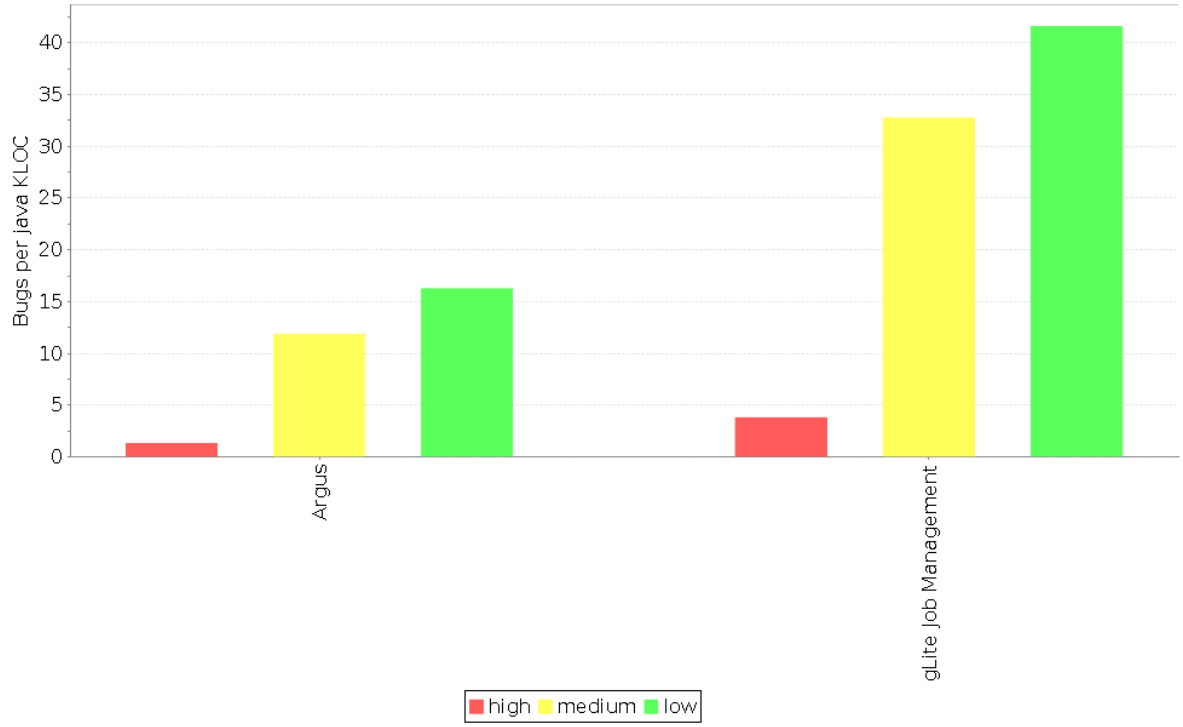
4.5.5 FindBugs errors

FindBugs is a tool for java static analysis specifically targeted for potential bugs:

http://findbugs.sourceforge.net/bugDescriptions.html

4.5.5.1 Reported errors have three levels of importance: "High", "Medium" and "Low". The graphs report, for each subsystem, the number of bugs found per KLOC.

Findbugs density per Java subsystem at 20 January 2011.

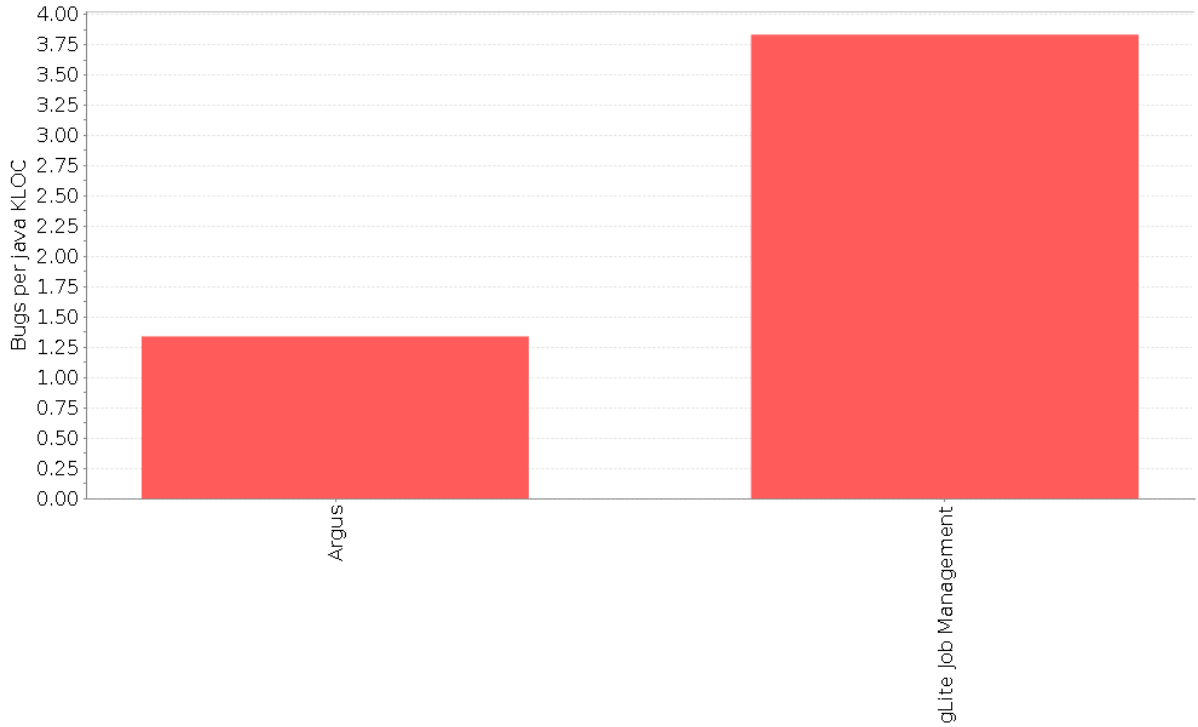


Product teams and products that have Java components not included in FindBugs metric:

- APEL Client: APEL parser [emi.apel.core, emi.apel.sge, emi.apel.lsf, emi.apel.condor, emi.apel.pbs], gLite-APEL [emi.apel.core, emi.apel.publisher]
- CERN Data Management: FTS [emi.fts.transfer-fts, emi.fts.transfer-interface]
- SAGA-SD-RAL: SAGA Service Discovery [emi.saga-adapter.context-java]
- StoRM: StoRM [storm.backend, storm.checksum]
- VOMS: VOMS-Admin [org.glite.security.voms-admin-server]
- dCache: dCache (server and clients) [emi.dcache.server, emi.dcache.srmclient]
- gLite Job Management: WMS [emi.wms-ui.wmproxy-api-java]
- gLite Security: Delegation Java [emi.delegation.delegation-service-java, emi.delegation.delegation-java], Hydra [emi.hydra.catalog-interface], Trustmanager/Util-java [emi.java-security.trustmanager, emi.java-security.trustmanager-axis, emi.java-secu-

rity.trustmanager-tomcat]

High priority Findbugs density per Java subsystem at 20 January 2011.



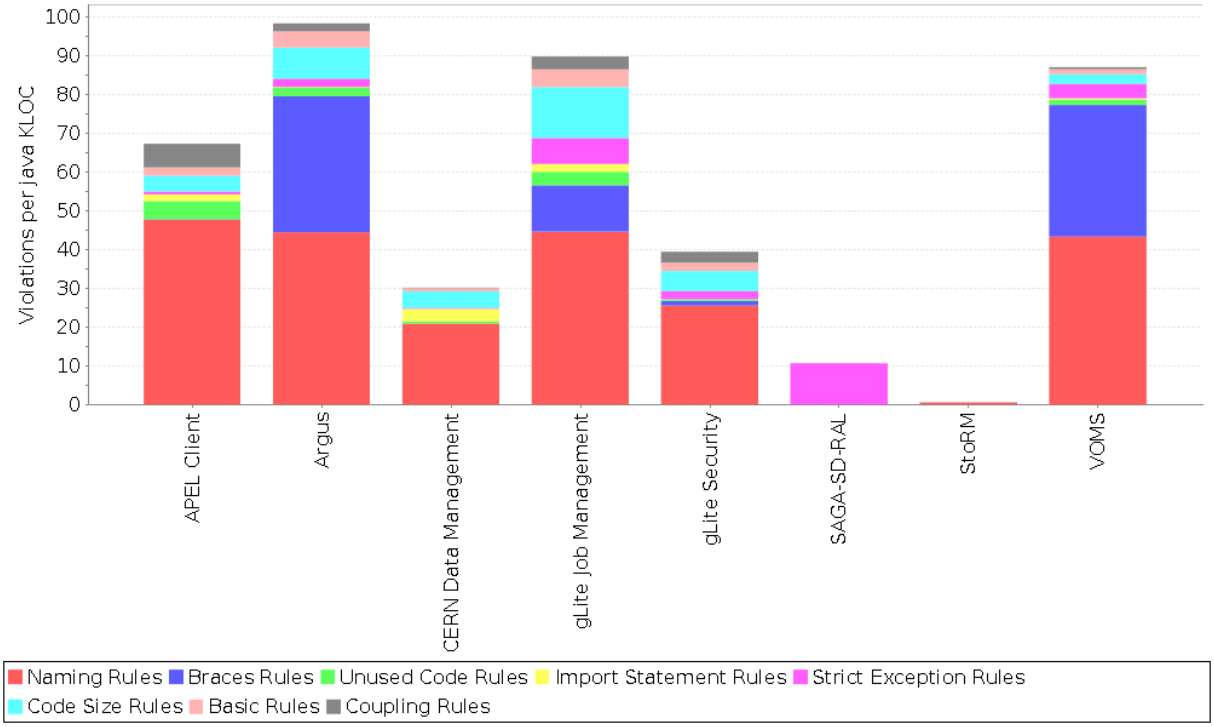
4.5.5.2 PMD errors

PMD is a tool for java static analysis specifically targeted for bad practices:

http://pmd.sourceforge.net/rules/index.html

These metric shows, for each subsystem, the amount of violations per KLOC and the type of violation with different colours.

PMD density per Java subsystem at 20 January 2011.



Product teams and products that have Java components not included in PMD metric:

- CERN Data Management: FTS [emi.fts.transfer-interface]
- StoRM: StoRM [storm.backend]
- dCache: dCache (server and clients) [emi.dcache.server, emi.dcache.srmclient]
- gLite Security: Delegation Java [emi.delegation.delegation-service-java], Hydra [emi.hydra.catalog-interface]

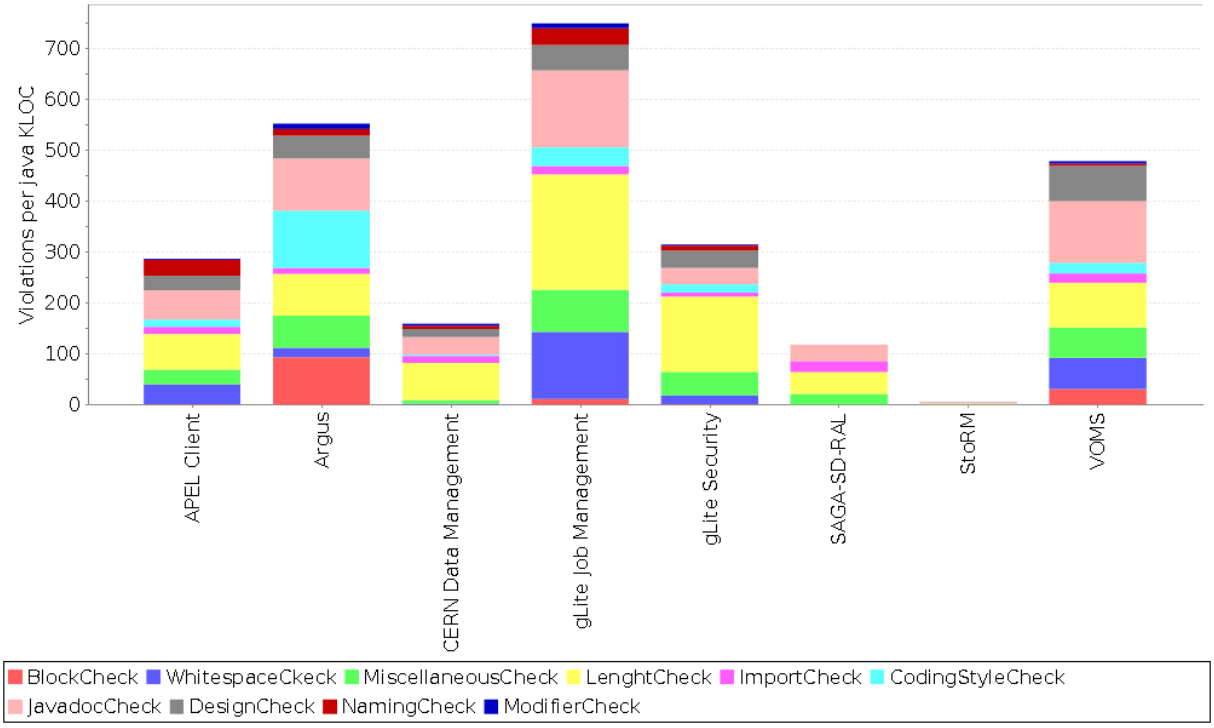
4.5.5.3 Checkstyle errors

Checkstyle is a tool for java static analysis specifically targeted for finding conventions violations:

http://checkstyle.sourceforge.net/checks.html

The following graphs show the amount of violations per KLOC in each subsystem, and the type of violation are showed in different colors.

Checkstyle density per Java subsystem at 20 January 2011.



Java components not included in Checkstyle metric:

- CERN Data Management: FTS [emi.fts.transfer-interface]
- StoRM: StoRM [storm.backend]
- dCache: dCache (server and clients) [emi.dcache.server, emi.dcache.srmclient]
- gLite Security: Delegation Java [emi.delegation.delegation-service-java], Hydra [emi.hydra.catalog-interface]

4.5.6 Python

4.5.6.1 Pylint errors

Not available yet.

4.5.7 Code commenting

Not available yet.

5. QUALITY IN USE METRICS

Quality in use metrics refer to the quality of the software as perceived by the end users in a production environment.

End users should provide feedback mainly through the GGUS portal, where they can open tickets for incidents related to the deployment and use of the software. An incident reported in GGUS can then lead to a problem (bug in the bug tracker) or not.

This metrics have been defined to cover the KPIs KSA1.1 and KSA1.2 defined in the Description of Work [R2] and in the Software Maintenance and Support Plan [R3].

5.1. TOTAL USER INCIDENTS (KSA1.1)

This metric is collected through the GGUS report generator. We need to find the way to integrate it here.

5.2. TRAINING AND SUPPORT INCIDENTS

Not available yet.

5.3. AVERAGE TIME TO DEAL WITH AN INCIDENT

This metric is collected through the GGUS report generator. We need to find the way to integrate it here.

5.4. USER SUPPORT LEVEL TICKET DISTRIBUTION

This metric could be dropped due to incompatibility with the GGUS model.

6. CONCLUSIONS

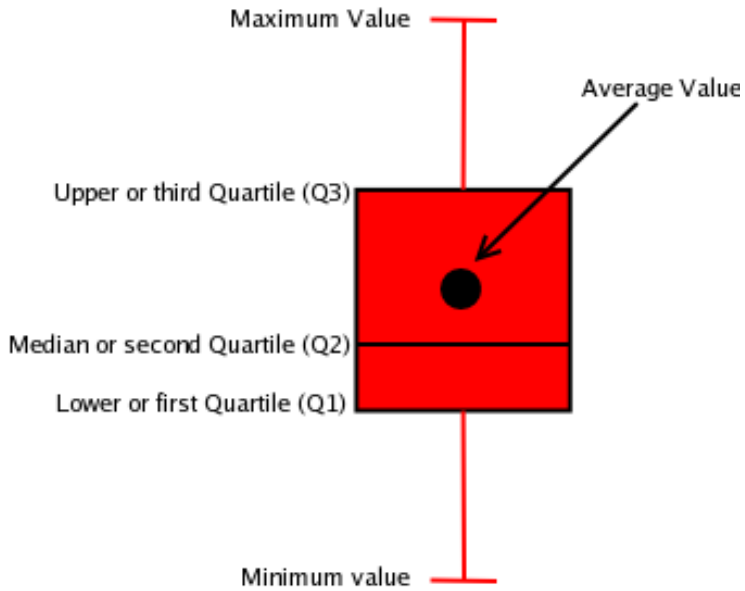
To be done.

Appendix A: Box plots

In this appendix we explain the format of the box plot used in this document.

In descriptive statistics, a **box plot** (also known as a **box-and-whisker diagram** or **plot**) is a convenient way of graphically depicting groups of numerical data through their five-number summaries: the smallest observation (sample minimum), lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation (sample maximum).

To this standard definition, we also added to our boxes the average value. A typical box will look like the following:



- **first quartile** (**Q1**) = cuts off lowest 25% of data
- **second quartile** (**Q2**) = median = cuts data set in half
- **third quartile** (**Q3**) = cuts off highest 25% of data