QAG activity report

On behalf of the QAG
hep-project-grid-qag@cern.ch

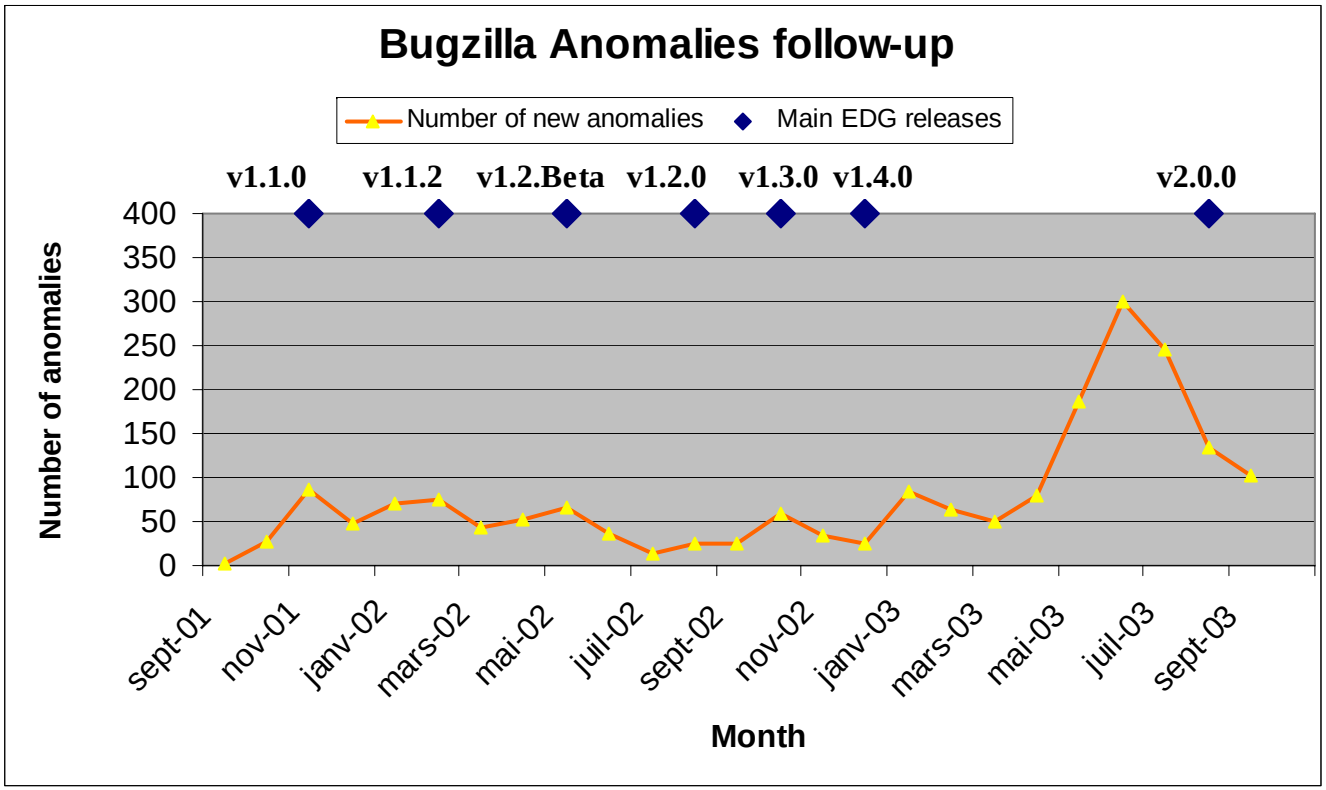
DataGrid is a project funded by the European Commission
under contract IST-2000-25182

Summary

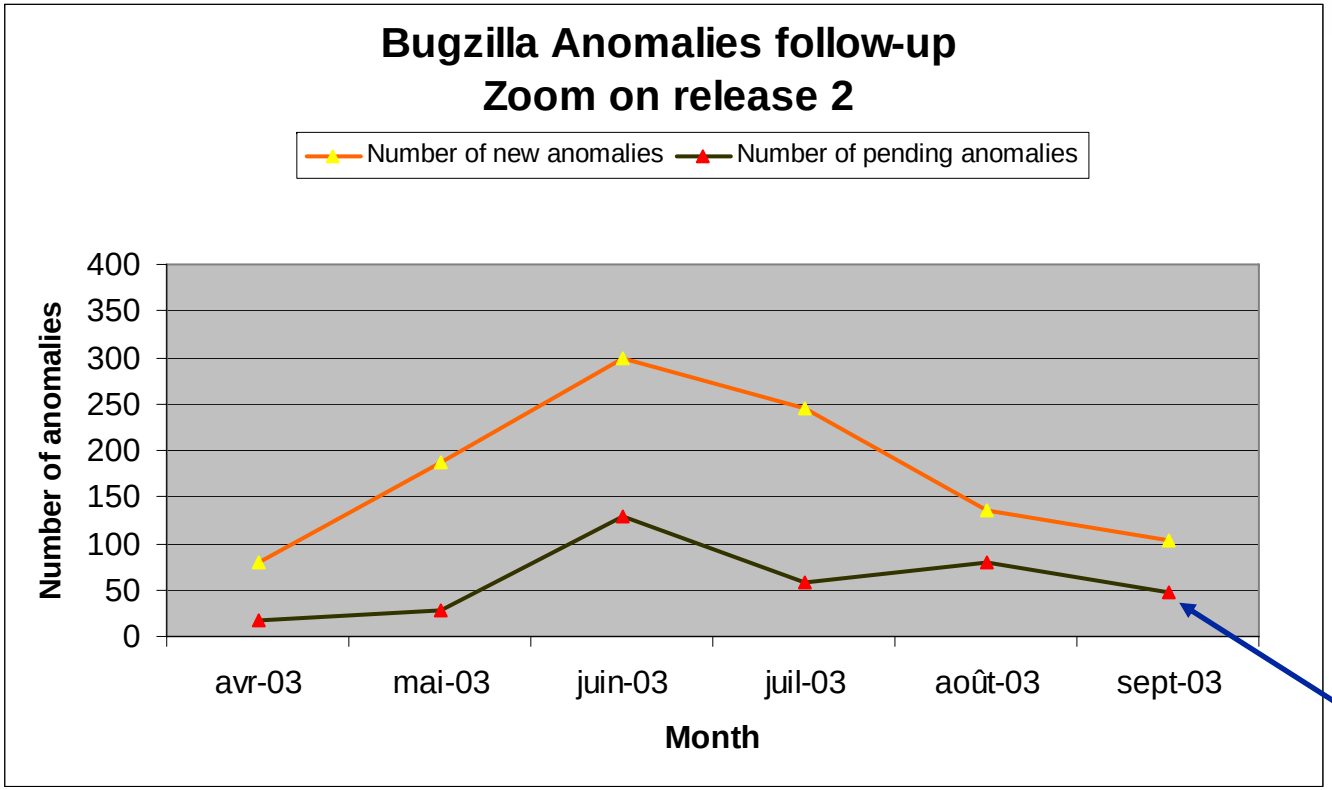
- Bugzilla anomalies follow-up
- Bugzilla procedure amendment
- Release procedure QA checklist
- Final deliverables & Proposal for the guidelines of the final deliverables report
- Performance indicators Definition & Implementation

Bugzilla anomalies follow-up until mid September

Better use of Bugzilla



Bugzilla: Zoom on release 2



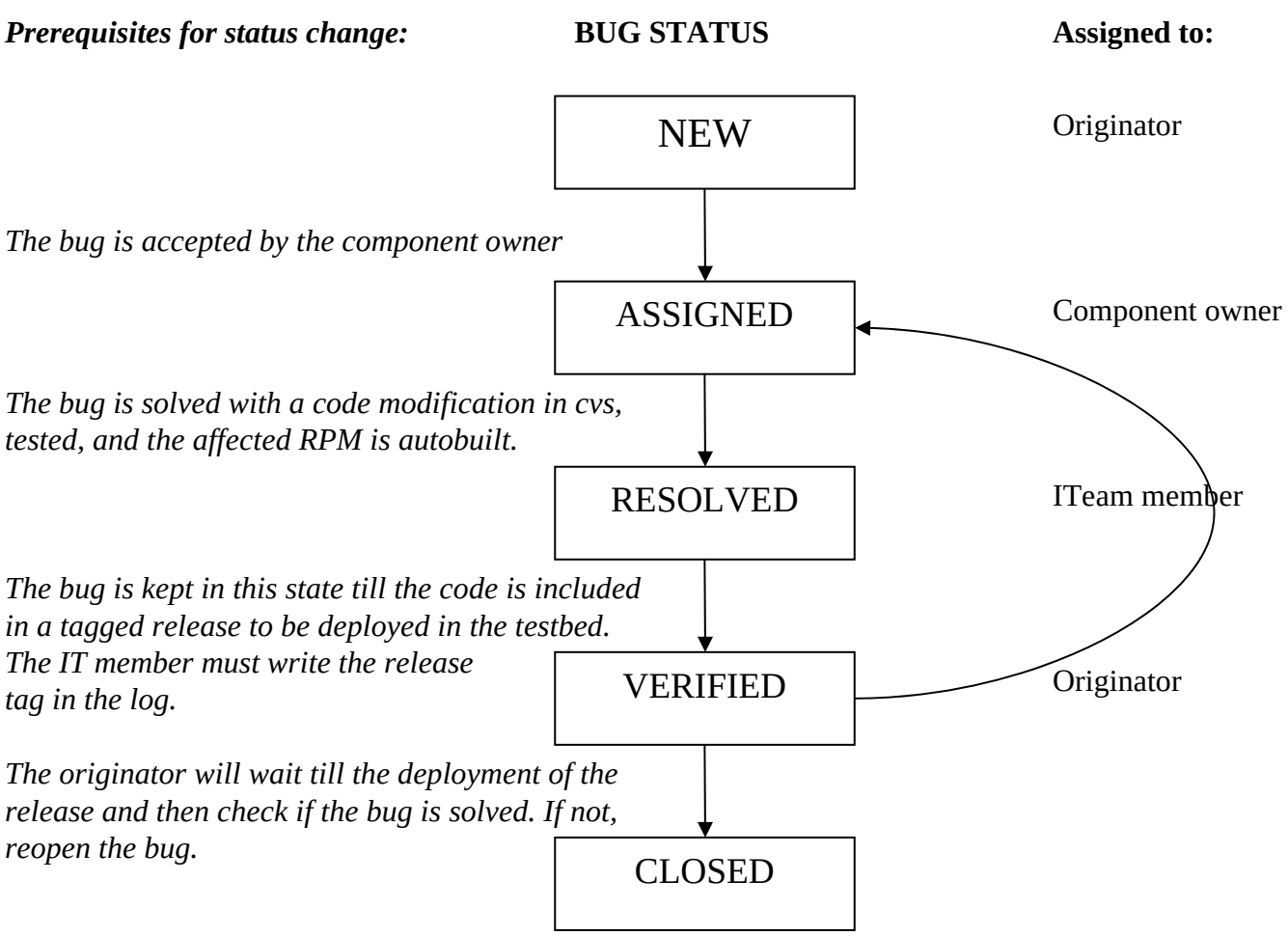
No critical bug

MTTR(Mean Time To Repair) during the period

Resolution Range	Bugs nb	%	Cumulated
1 day	258	28%	
3 days	80	9%	
7 days	133	14%	50%
longer	464	50%	
	935	100%	

Bugzilla procedure amendment

- The main improvement concerns the fact that the bug is automatically driven back until the originator. https://edms.cern.ch/document/386113



Release procedure QA checklist

- Describes the basic checks that has to be fulfilled for a component release verification. https://edms.cern.ch/document/387659

- **Adhere to the developers' guide**: e.g. Public interfaces should be in separate packages and versioned independently; package naming; Everything in central CVS; Validate if the correct version of the RPM is the flaged one in autobuild status page; Validate if the RPM is in RMP repository.

- **Documentation:** Verify Licence file; Readme file; Install file; Authors & Maintainers file; Dependencies file; Post-install doc; Code doc (Is there javadoc, doxygen ,….); End user documentation; Installation guide; Man pages (There must be a man page for each executable and you may provide one for configuration files).

- **Tests:** test report according to the test plans including numbers on performance and scalability. Verify if each item of the test plan is covered in the test report. Give the test report's URL.

- **Easily configurable** to allow a setup/usage in release 2.0 mode. Verify if new functionalities are optional.

- **Api changes** (and new apis) must follow the api change procedure. If the API changed: give the list; has ATF procedure been followed

QA checklist: release 2.x

- **VOMS - Slot: Starts week of September 1st, ends September 12th**
  - QAG checklist delivered on September 12th
  - https://edms.cern.ch/document/404428
- **LCMAPS - Slot: Starts September 10th, ends September 19th**
  - QAG checklist delivered on September 10th
  - https://edms.cern.ch/document/404431
- **RLS - Slot: Starts September 16th, ends September 25th**
  - QAG checklist delivered on September 19th
  - http://cern.ch/edg-wp2/release2.1/index.html#quality

Deliverables for PM33

- Title: User documentation for release 2.x (the production one)
  - D6.7=(D1.6,2.5,3.5,4.5,5.5,6.7)
  - This deliverable is presented as a set of links gathering together the various user guides from WPs (1,2,3,4,5,6,7):
    - Installation guide
    - Users guide
    - Developer guide
- Editor: Eric Fede fede@cppm.in2p3.fr
- Timescale:
  - The various link should be sent to Eric before 6/10/2003
  - The review will start 6/10/2003
  - The document should be sent to EU before the end of October
- Reviewers: Users Guides: WP8 (D. Boutigny), Installation Guides (WP6): Site Administrators

QR11 Quarterly report for PM33

- D12.15: Quarterly report – Period: July to September 2003

- Timescale:
  - Should be sent to Gabriel before 14/10/2003
  - The document should be sent to EU before the end of October

Deliverables for PM36

D1.7	Final evaluation report
D2.6	Final evaluation report
D3.6	Final evaluation report
D4.6	Final evaluation report
D5.6	Final evaluation report
D6.8	Final evaluation of testbed operation
D7.4	Final Report on network infrastructure and services
D7.7	Security report on final project releases
D8.4	Report on results of HEP applications at Run #2 and Final Application Report
D9.5	EO application processing testbed demonstration and final report
D10.4	Final report including report on the 2nd bio-testbed release
D11.7	Final conference
D11.9	Contribution to international standards

- Timescale:
  - The review process will start 24/11/2003
  - PTB approval 15/12/2003
  - The document should be sent to EU before the end of January 2004
- Moderators & reviewers: To be nominated

Final Quarterly & Annual report

- D12.16: QR12 Quarterly report – Period: October to December 2003
- D12.19: Third Annual report

- Timescale:
  - Should be sent to Gabriel before 14/01/2004
  - Should be sent to EU before the end of January 2004

Proposal for the guidelines of the final deliverables report

- The motto of the deliverables should be:

  'what has been achieved - what needs to be done in future'

- If applicable, the deliverables should relate to the first 2 deliverables of the project (Dx.1:Current technology report, Dx.2: Design documents), and report fulfilments and deviations (explaining why) of the original plans.
- A thorough evaluation of the WP achievements, again if possible using criteria defined in the first 2 deliverables should also be part of it.
- Finally, a discussion of future developments, in particular pointing out what seems to be a promising way and what not, is needed.
- So, there should be three major chapters:
  - Achievements
  - Evaluation
  - Future Directions

Performance and definitions of Efficiency

Rough draft at https://edms.cern.ch/document/386039

There is a need to quantify how well grid projects are working

Both in terms of providing the user with the service they require and in terms of utilising the resources available to it an efficient manner.

Traditional benchmarking is inadequate, its works well in a homogeneous stable environment. Less well in \heterogeneous dynamic environment.

Performance and definitions of Efficiency

To this end we described a set of “efficiencies”

Properties of the efficiencies:

-efficiency of 1.0 should reflect the ideal case.

-efficiency of 0.0 should reflect that it does not work.

-(Ideally) the efficiencies should be linear.

Performance and definitions of Efficiency

These efficiencies are designed to test how well the infrastructure is providing what is needed.

These efficiencies should be general/non-architecture specific as possible. Essential if they are going to be used to compare between grid projects or the same project at different stages of its evolution.

Obviously these efficiencies will depend on the nature of the job

These should be measured during real operation when things really do go wrong. Results shown will be from the 1.4 Testbed with Jobs run through the Imperial College RB. Lots of help by Paul Crosby from UCL

User Efficiencies

Most importantly, the jobs should complete returning the correct output.

Easily turned into an efficiency:

$$E_{crude} = \frac{\text{Number of jobs successfully completed}}{\text{Total Number of jobs submitted}}$$

This should be measured applications testers, however can be estimated from the information in the LB.

Sadly, this was only ~0.6 for release 1.4

User Efficiencies

The reasons for failure were:

-Information systems could not cope with the scale of resources causing jobs not to be matched even when resources were available

-Known problem with the compiler version that was used causing problems with the jobs database. Cleaning caused all current jobs to be lost.

-Limit of version and method of use of CondorG. If reached, cleaning caused all running jobs to be lost

-Many smaller problems

However what is important is to the user is the base figure of 0.6

User Efficiencies

Hopefully, in release 2 $E_{Crude} \approx 1$. Then a more subtle definition is required.

“From a user perspective an efficiency of 1 corresponds to all their jobs running immediately on resources of sufficient scale and which have instant access to all the data required by the jobs.”

“It is important to note that even a very large, centralised computing centre would not achieve perfect efficiency as there will always be a certain overhead taken to by the batch system to process the job etc.”

User Efficiencies

We have chosen:

$$E_{User} = \frac{\text{Time while job is running}}{\text{Total time between submission and completion}}$$

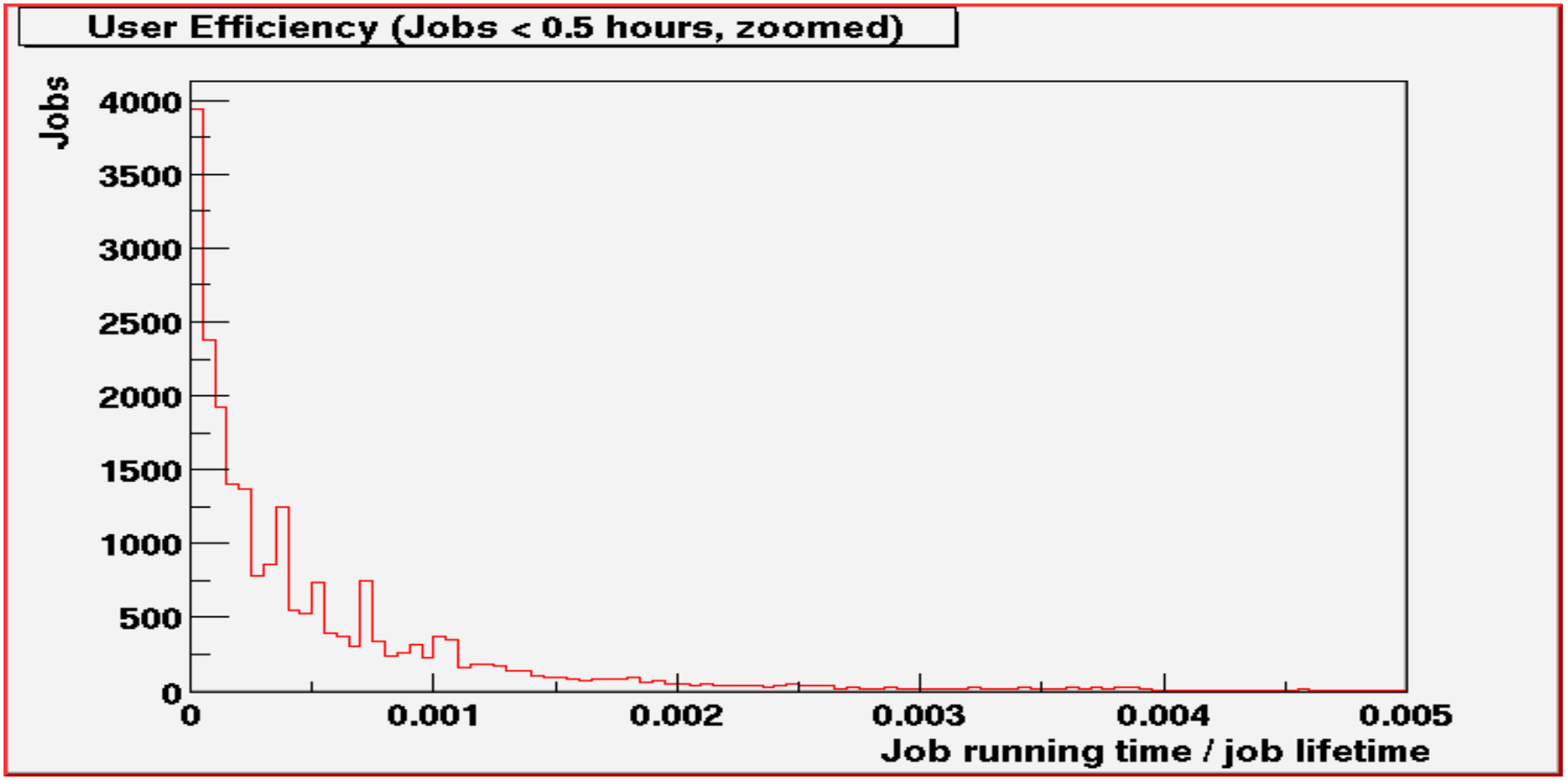
In a heterogeneous system there are many subtle problems with this definition.

It is also open to user stupidity in their Job Definition

However, it was the best working definition that we come up come up with. Captures most inefficiencies.

User Efficiencies

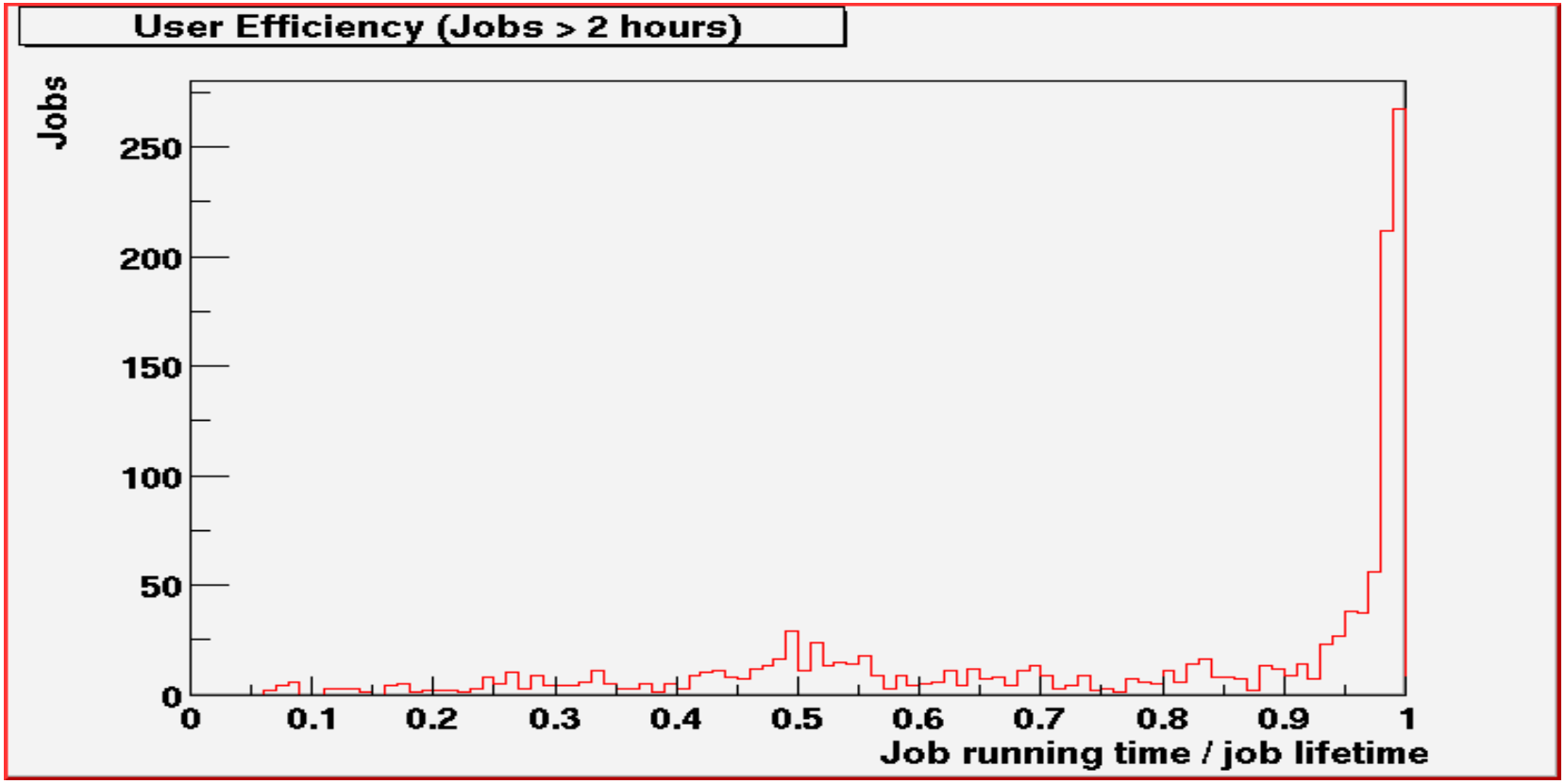
For short jobs…



Still many very short “Hallo World” jobs which are impossible to run efficiently…

User Efficiencies

However for longer jobs…



Still *some* jobs with low efficiencies…

User Efficiencies

As $E_{Crude} \approx 1$ it becomes more meaningful to measure the performance from the lbserver.

Hope to dump and collect the contents of these databases from the different lb machines.

System Efficiency

System efficiency is a quantification of how well the system is utilising the resources available to it.

Could be used to quantify differences between middleware projects or to show funding agencies that a distributed system really does use resources efficiently!

System Efficiency

System Efficiency is harder to define than user efficiency, but one possible candidate for a system efficiency of 1 would be the case when all the resources are in the same location and requests for the resources are instantly dealt with, up to the limit that all the available resources were being used.

In the case when the system was oversubscribed this would correspond to the fractional CPU usage and would still be meaningful for systems where this was not the case.

System Efficiency

I.e.

$$E_{System} = \frac{\text{Resources Delivered}}{\text{Min (Resources Requested,Total Resources Available)}}$$

Perhaps an integrated value over a given period is more meaningful…

$$I_{System} = \frac{\int_{t_1}^{t_2} E_{System}\,dt}{\int_{t_1}^{t_2} dt} = \frac{\int_{t_1}^{t_2} E_{System}\,dt}{t_2 - t_1}$$

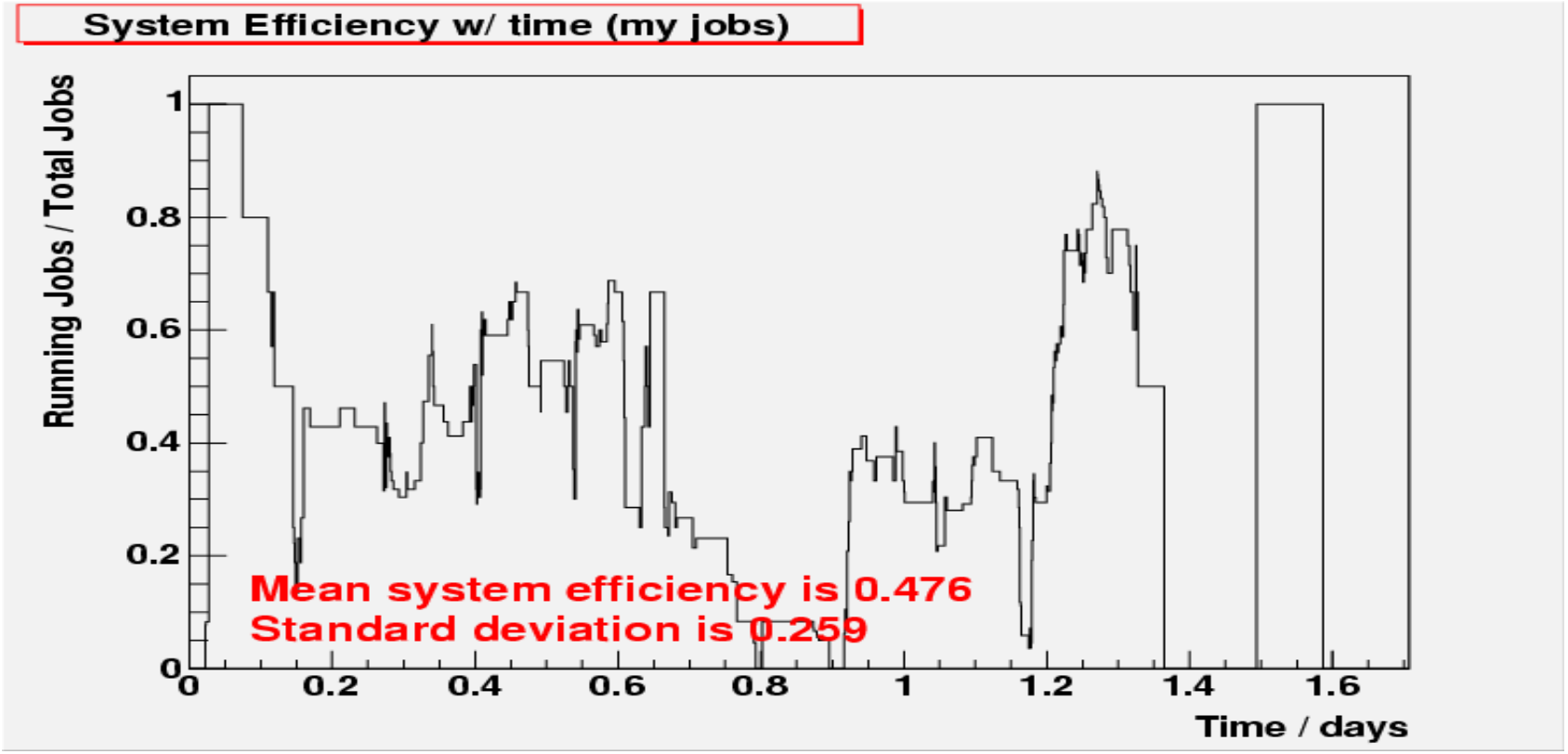
System Efficiency

This is harder to measure, not only requires knowledge of what resources are available but also:

-What resources satisfy the users needs (as specified in the jdl)
-What resources is the user allowed to run on
-Etc etc

System Efficiency

To avoid assumption we (Paul Crosby) measured for a series of his own jobs. Submitted in bursts and monitoring what resources were available to him via MDS.



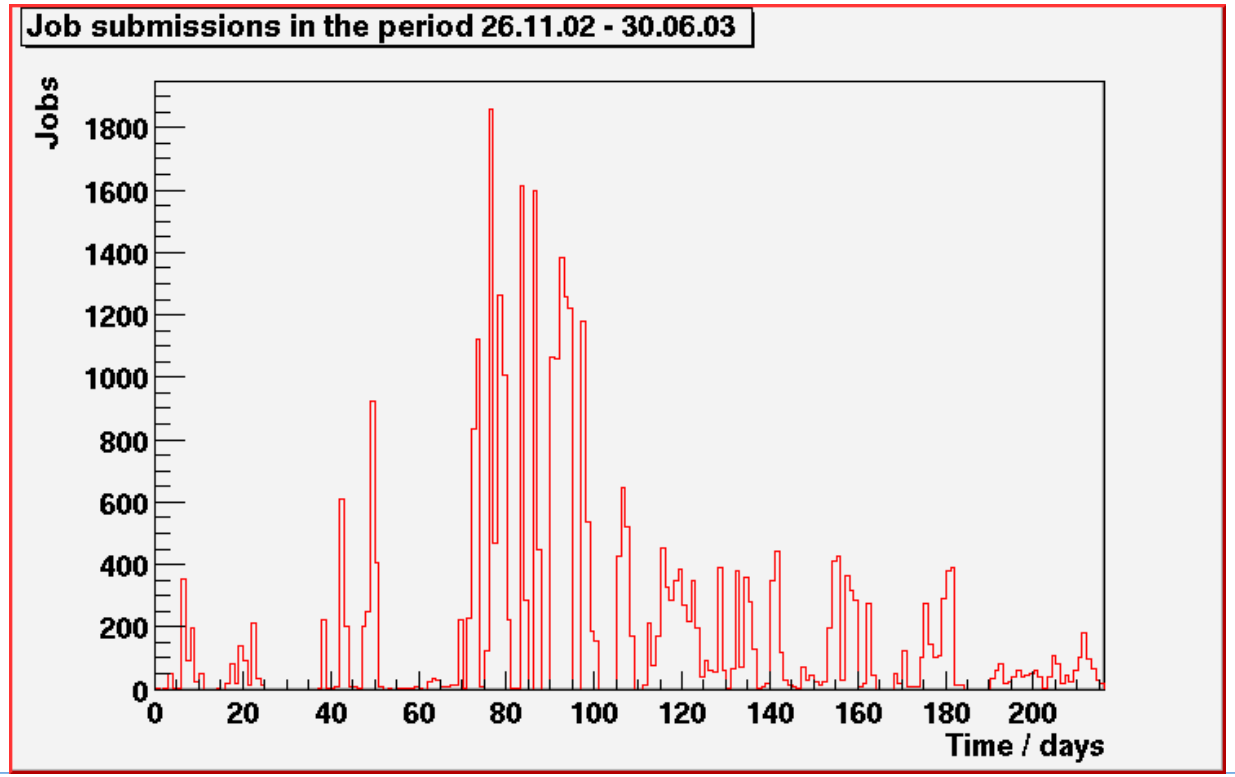
Clearly not always 100%

Non-efficiency metrics

Users…

“build it and they will come…”

But if it doesn’t work well, they will soon go away again…



Conclusions

-Changes in Bugzilla procedure and usage

-Release verification procedure

-Still lots of deliverables

-We are trying to quantify performance