

Metrics definition inside the software QA process

Andres Abad Rodriguez

CERN

Inverted CERN School of Computing, 3-4 March 2011

Contents

- What is a metric?
- Why are they needed?
- Different types and how to collect them
- Reporting examples
- Standards

Personal highlights

- Software engineer education background
- Positions at CERN:
 - *2007-08*: SCADA developer for the LHC Gas Control System project



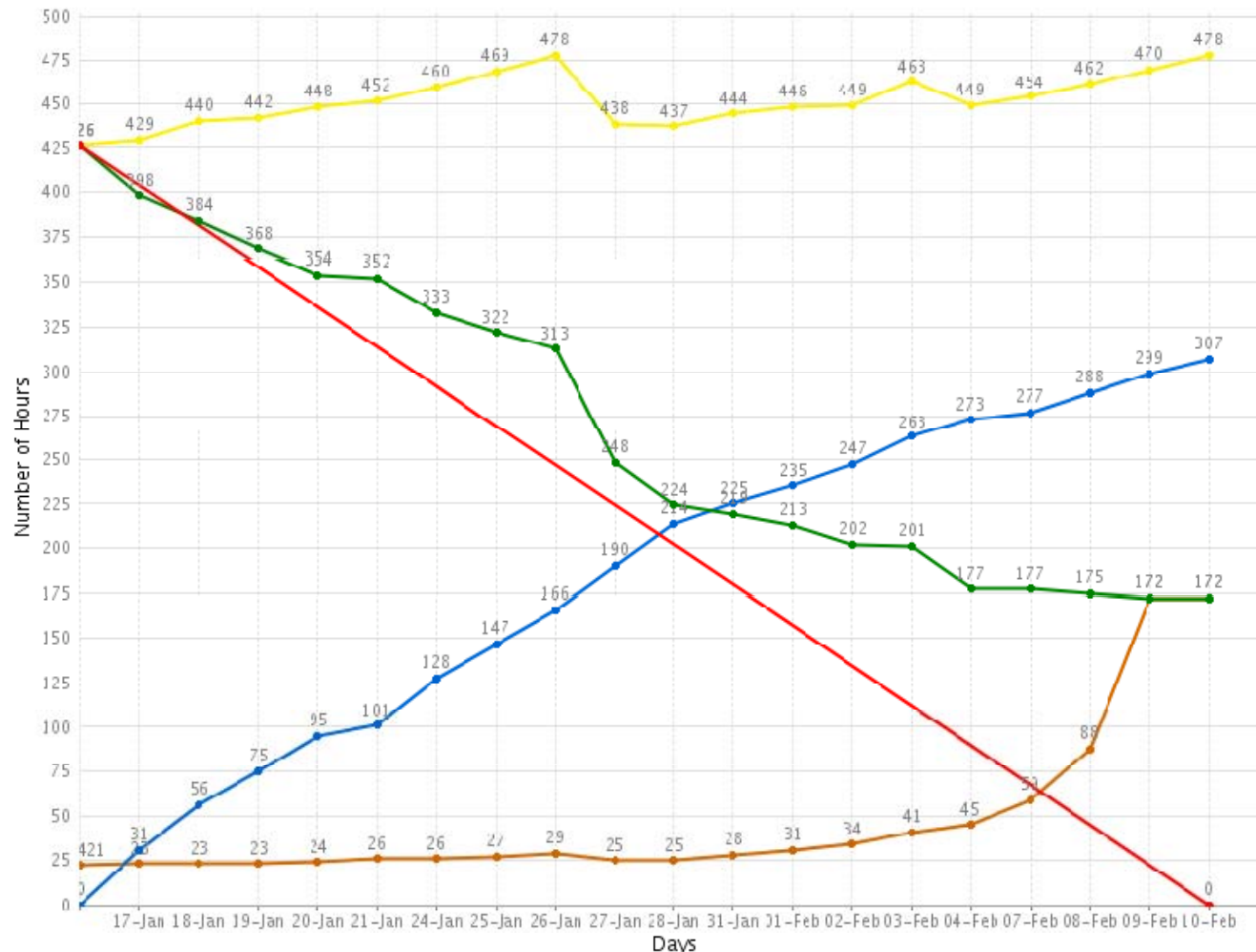
- *2008-10*: Java, AXIS and GWT developer for the ETICS project



- *2010-11*: Continue with ETICS as part of EMI. **Also involved in the quality assurance team.**



Real example: SCRUM



- Metrics are useful for **big** but **also** for **small teams**
- Different metrics can be used depending on the goal of their analysis

What is measurement?

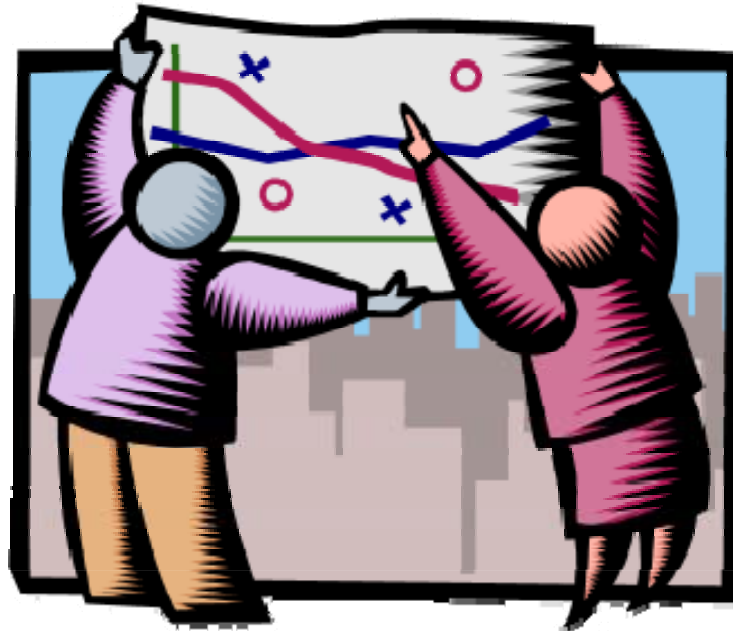
- Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules



Measurement is to assign values to things or processes to compare them

What is a metric? And a SW metric?

- **Metric:** a standard of measurement
- **Software metric:** is a measure of some properties of a piece of software (including all its process) or its specifications



Why to collect them? Reasons

- **Better understand, track and control** of software projects, processes and products
- Developers get **feedback** about their program
- **Guidelines** for refactoring
- Way to measure the **progress** of code **during development**

Why to collect them? Benefits

- The goal is to become better at:
 - Schedule and budget planning
 - Cost estimation
 - Quality assurance testing
 - Software debugging
 - Software performance optimization
 - Optimal personnel task assignments



Different approaches

- We can select one of the two next approaches (as described in Westfall[1]):
 - Measure **everything** that is measureable, report on it and try to correlate the information



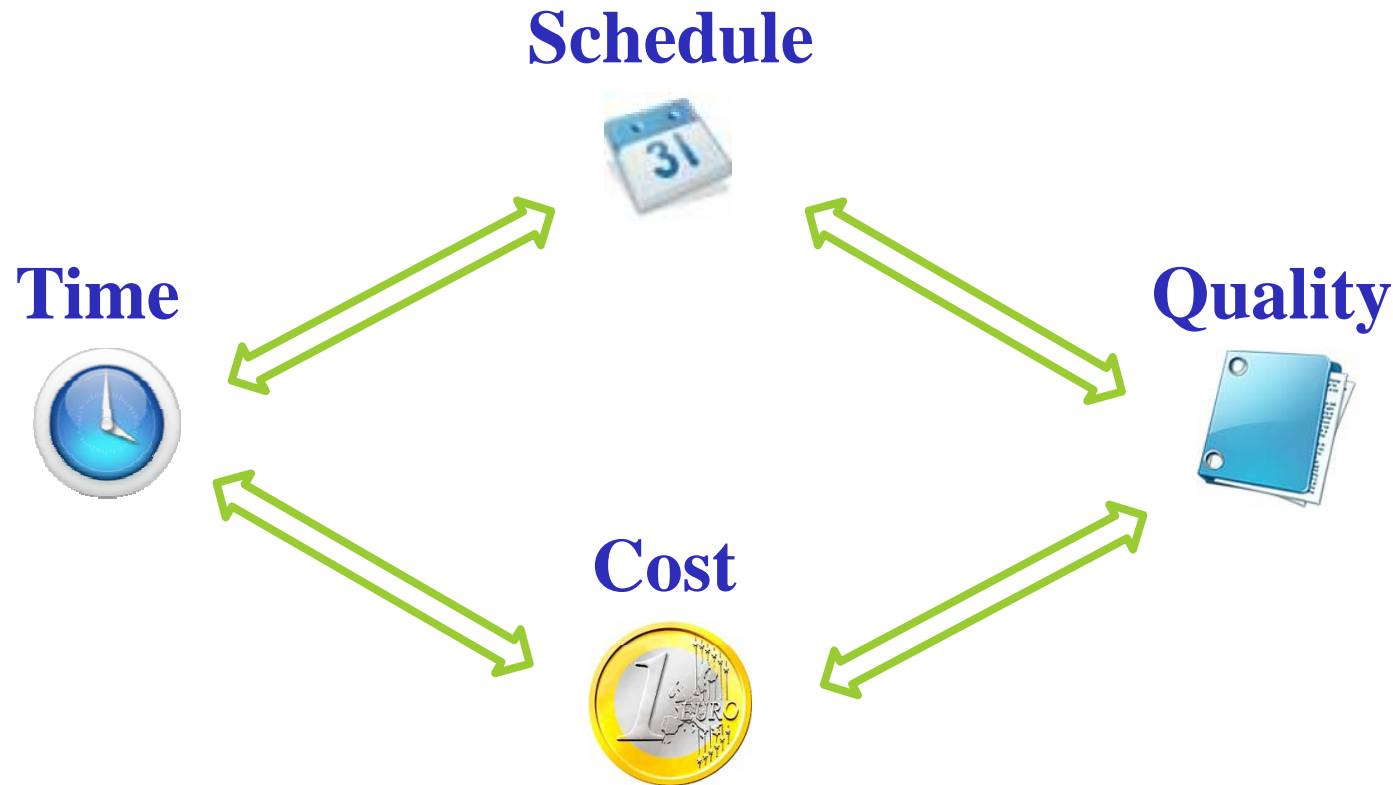
- Collect what is seen to be the currently most important metric



[1]Linda Westfall, "12 steps to useful software metrics", http://www.westfallteam.com/Papers/12_steps_paper.pdf

Criteria

- Metrics are under the same limitations that the software they measure:



Types of measurements

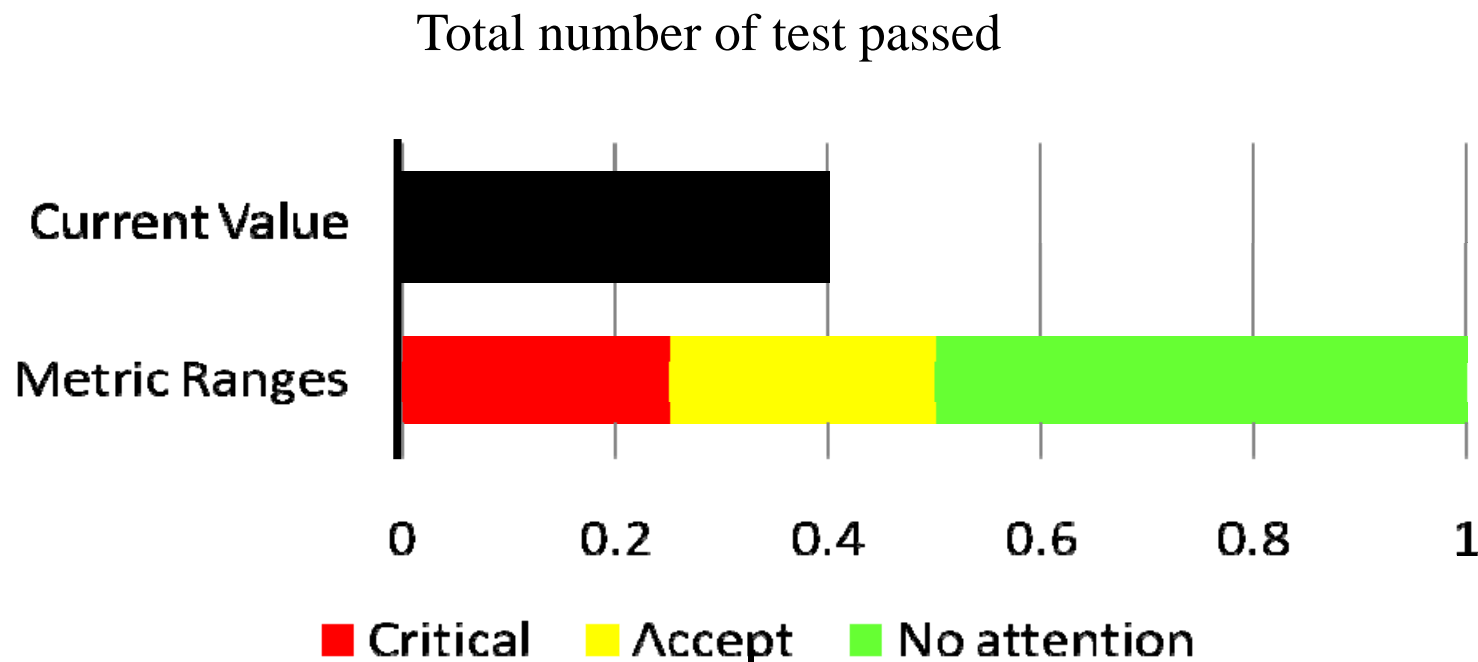
- **Interval**
 - *Example:* if the number of people assigned to a module is between X and Y is consider correct

- **Ratio**
 - *Example:* the current number of bugs for this module is 50% of total

- **Absolute**
 - *Example:* the software has 350.000 lines of code

Indicators

- They can be plotted as a horizontal bar-chart, typically using a normalized scale [0,1] or [0%,100%] which is split into:
 - **Red**: Critical, needs improvement or immediate attention
 - **Yellow**: reasonable or needing some attention
 - **Green**: Very good and needing no immediate attention

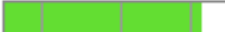


Indicators (II)


IT Services

17 Feb 2011 Thu 01:24:05


Administrative applications



88% available [\(more\)](#)


Services for physics



98% available [\(more\)](#)


For developers and engineers



91% available [\(more\)](#)


Windows Services



81% available [\(more\)](#)


Infrastructure services


99% available [\(more\)](#)


Databases



100% available [\(more\)](#)

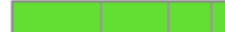
Mail and Web Services



100% available [\(more\)](#)


Worldwide LHC Computing Grid



98% available [\(more\)](#)


Networking



100% available [\(more\)](#)

Documents and collaborative services


97% available [\(more\)](#)

EU-funded projects



100% available [\(more\)](#)

SLS by CERN IT/CF

SLS.Support@cern.ch

Software metric types

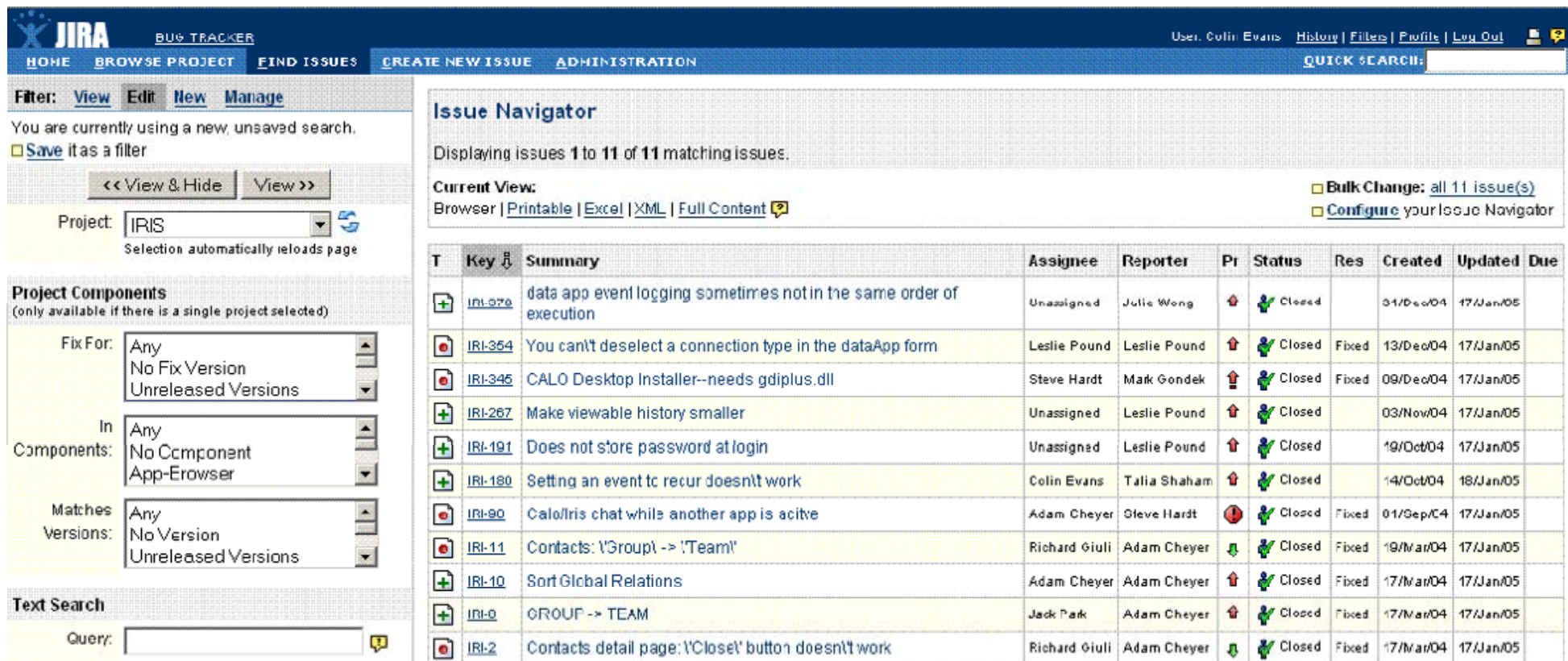
- Last step to define metrics is to select which one of them are important in the project
- We can divide them in three groups:
 - **process metrics**
 - *Example:* average time to handle high severity bugs
 - **product metrics**
 - *Example:* unit test coverage
 - **quality in use metrics**
 - *Example:* Training incident per period of time
- Additionally we can add an extra group called “**Additional metrics to consider**” to add any other necessary metric

Metric Types (II): Process metrics

- Average time to handle a high severity bug
 - Divide the total time spent in the high severity bugs and by the number of high severity bugs
- Bug severity distribution (grouped by severity level)
 - Amount of bugs
- % of failed nightly builds
- Certification test effectiveness
 - % of bug found during integration test of a release over the number of bugs found in production of the same release)
- Up-to-date documentation

Metric Types (III): Process metric tools

- Most of the information should be obtained from the different bug trackers used in the project
- Another information such as the result of the nightly builds can be obtained from the continuous integration tool (ex: Maven)



The screenshot shows the JIRA Bug Tracker interface. The top navigation bar includes links for HOME, BROWSE PROJECT, FIND ISSUES, CREATE NEW ISSUE, and ADMINISTRATION. The user is identified as Colin Evans. The main content area displays an 'Issue Navigator' for the project 'IRIS', showing 11 matching issues. The issues are listed in a table with columns for Key, Summary, Assignee, Reporter, Priority, Status, Resolution, Created, Updated, and Due.

T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	IRI-070	data app event logging sometimes not in the same order of execution	Unassigned	Julie Wong		Closed		01/Dec/04	17/Jan/05	
	IRI-354	You can't deselect a connection type in the dataApp form	Leslie Pound	Leslie Pound		Closed	Fixed	13/Dec/04	17/Jan/05	
	IRI-345	CALO Desktop Installer--needs gdipplus.dll	Steve Hardt	Mark Gondek		Closed	Fixed	09/Dec/04	17/Jan/05	
	IRI-267	Make viewable history smaller	Unassigned	Leslie Pound		Closed		03/Nov/04	17/Jan/05	
	IRI-191	Does not store password at login	Unassigned	Leslie Pound		Closed		19/Oct/04	17/Jan/05	
	IRI-180	Setting an event to recur doesn't work	Colin Evans	Talia Shaham		Closed		14/Oct/04	18/Jan/05	
	IRI-90	Calo/Iris chat while another app is active	Adam Cheyer	Steve Hardt		Closed	Fixed	01/Sep/04	17/Jan/05	
	IRI-11	Contacts: 'Group' -> 'Team'	Richard Giuli	Adam Cheyer		Closed	Fixed	19/Mar/04	17/Jan/05	
	IRI-10	Sort Global Relations	Adam Cheyer	Adam Cheyer		Closed	Fixed	17/Mar/04	17/Jan/05	
	IRI-0	GROUP -> TEAM	Jack Park	Adam Cheyer		Closed	Fixed	17/Mar/04	17/Jan/05	
	IRI-2	Contacts detail page: 'Close' button doesn't work	Richard Giuli	Adam Cheyer		Closed	Fixed	17/Mar/04	17/Jan/05	

Metric Types (IV): Product metrics

- Unit test coverage
- Memory leak warnings
- Number of supported platforms (from the total defined to be supported)
- Total bug density (number of bugs per KLOC*)
- Total bug density per release
- Cyclomatic complexity

* KLOC: 1000 lines of code

Metric Types (V): Product metric tools

- **Cyclomatic complexity:** CCCC (java, C and C++)
- **Code analysis:** Checkstyle (java), ckjm (java), findbugs (java), PMD(java), Pylint (python), cppcheck (C++)
- **Unit testing:** Xunit (multiple languages)
- **Test coverage:** pycoverage (python)
- **RPM integrity:** rpmlint
- **Number of lines of code:** sloccount (multiple languages)
- **Security and other tools:**
http://security.web.cern.ch/security/recommendations/en/code_to_ols.shtml

Metric Types (VI): Product metric tools

- Findbugs example
Summary

Warning Type	Number
Bad practice Warnings	33
CORRECTNESS Warnings	87
Internationalization Warnings	22
Malicious code vulnerability Warnings	4
PERFORMANCE Warnings	121
STYLE Warnings	120
Total	387

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

CodeWarning

- DE** org.etics.buildsystem.ui.pat.command.RemoveCommand.doWork(String) might ignore java.lang.Exception
- Eq** org.etics.buildsystem.ui.datamodel.helper.platform.PlatformGroup defines compareTo(PlatformGroup) and uses Object.equals()
- Eq** org.etics.buildsystem.ui.datamodel.helper.SystemPropertyDefinition defines compareTo(SystemPropertyDefinition) and uses Object.equals()
- Eq** org.etics.buildsystem.ui.datamodel.Platform defines compareTo(Platform) and uses Object.equals()
- Eq** org.etics.buildsystem.ui.wsclient.buildsystem.patched.history.AbstractHistoryEntry defines compareTo(AbstractHistoryEntry) and uses Object.equals()
- HE** org.etics.buildsystem.ui.datamodel.Configuration defines equals and uses Object.hashCode()
- J2EE** Store of non serializable org.etics.buildsystem.ui.datamodel.User into HttpSession in org.etics.buildsystem.ui.authn.AuthnFilter.doFilter(ServletRequest, ServletResponse, FilterChain)

Metric Types (VII): Product metric tools

- Checkstyle example

Coding Style Check Results

Summary	
Total files checked	21
Files with errors	21
Total errors	781
Errors per file	37

The following are violations of the Sun Coding-Style Standards:

File: /home/condor/execute/dir_21040/userdir/org.etics.submission.vmloder/src/org/etics/submission/vmloder/bootstrapper/Main.java

Line Number	Error Message
0	File does not end with a newline.
6	Missing an import control file.
9	Wrong order for 'org.apache.log4j.Level' import.
21	Utility classes should not have a public or default constructor.
23	Missing a Javadoc comment.
24	Missing a Javadoc comment.
24	'+' should be on a new line.
27	Missing a Javadoc comment.
28	'+' should be on a new line.
31	Missing a Javadoc comment.
32	'+' should be on a new line.
33	'+' should be on a new line.
34	'+' should be on a new line.
35	'+' should be on a new line.
36	'+' should be on a new line.
37	'+' should be on a new line.
38	'+' should be on a new line.
39	'+' should be on a new line.
40	'+' should be on a new line.

Metric Types (VIII): Quality in use metric

- Total user incidents per period of time.
- Training and support incident per period of time.
- Average time to deal with an incident



‘Incident’ is not the ITIL term. We consider it as all user communication that is not a software bug

Tools: main tools are again the bug trackers or any other tool used to track user problems

Normalization

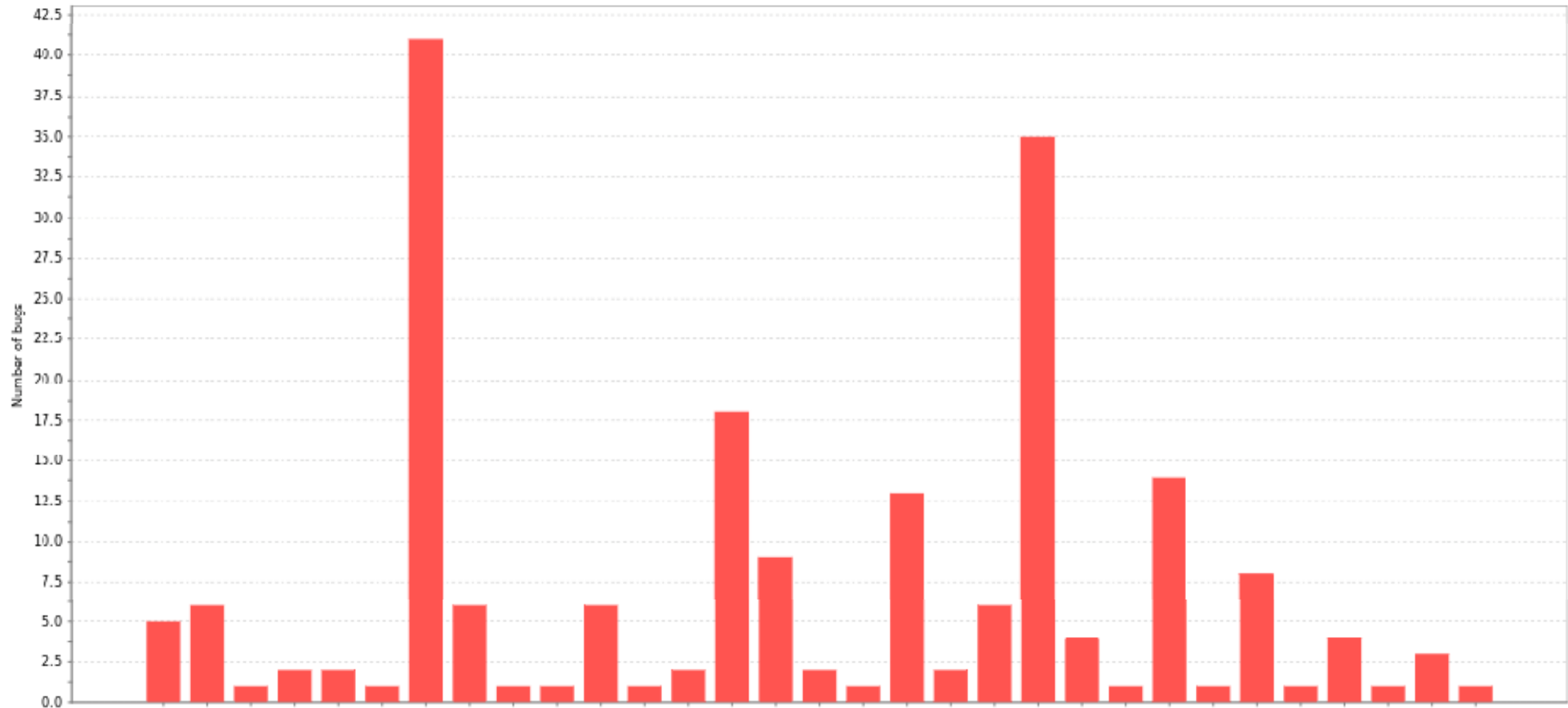
- Some parameters have to be normalized to be able to compare them
- Can be normalized using:
 - **LOC**: Number of lines of code
 - **MLOC**: Method lines of code
 - **NOC**: Number of classes
 - **CLOC**: Class lines of code
 - **NOP**: Number of packages

Example of formal definition

Metric Id	TotalUserIncidents		
Name	Total user incidents per month		
Description	This metric covers incidents not only in the software but also in the documentation, training and user support processes, per month.		
Input(s) & Units	This metric will be collected through GGUS		
Output(s) & Units	GGUS tickets per per month		
Scope	GGUS 3 rd level support unit, Product Team		
Thresholds/Target Value	It is difficult to state a threshold valid for all the product teams, in general a decreasing trend would show positive results.		
Tools	GGUS		
Availability	ARC	dCache	gLite UNICORE
	It should be possible to collect this metric for all the middlewares.		
Quality factor	Usability		
Goals	Keep at minimum the amount of user incidents by providing tested and documented software.		
Risks	<p>This metric does not relate its values with the complexity of the software or the number of users actually using it.</p> <p>The use of mailing lists such as log-rollout is damaging for tracking incidents related metrics. All incidents reports from users should be reported through GGUS.</p> <p>In order to track incidents per component and per product team a clear assignment of 'type of problem' (in GGUS) to components and product teams has to be made.</p>		
Special Notes	None		

Example of final report (I)

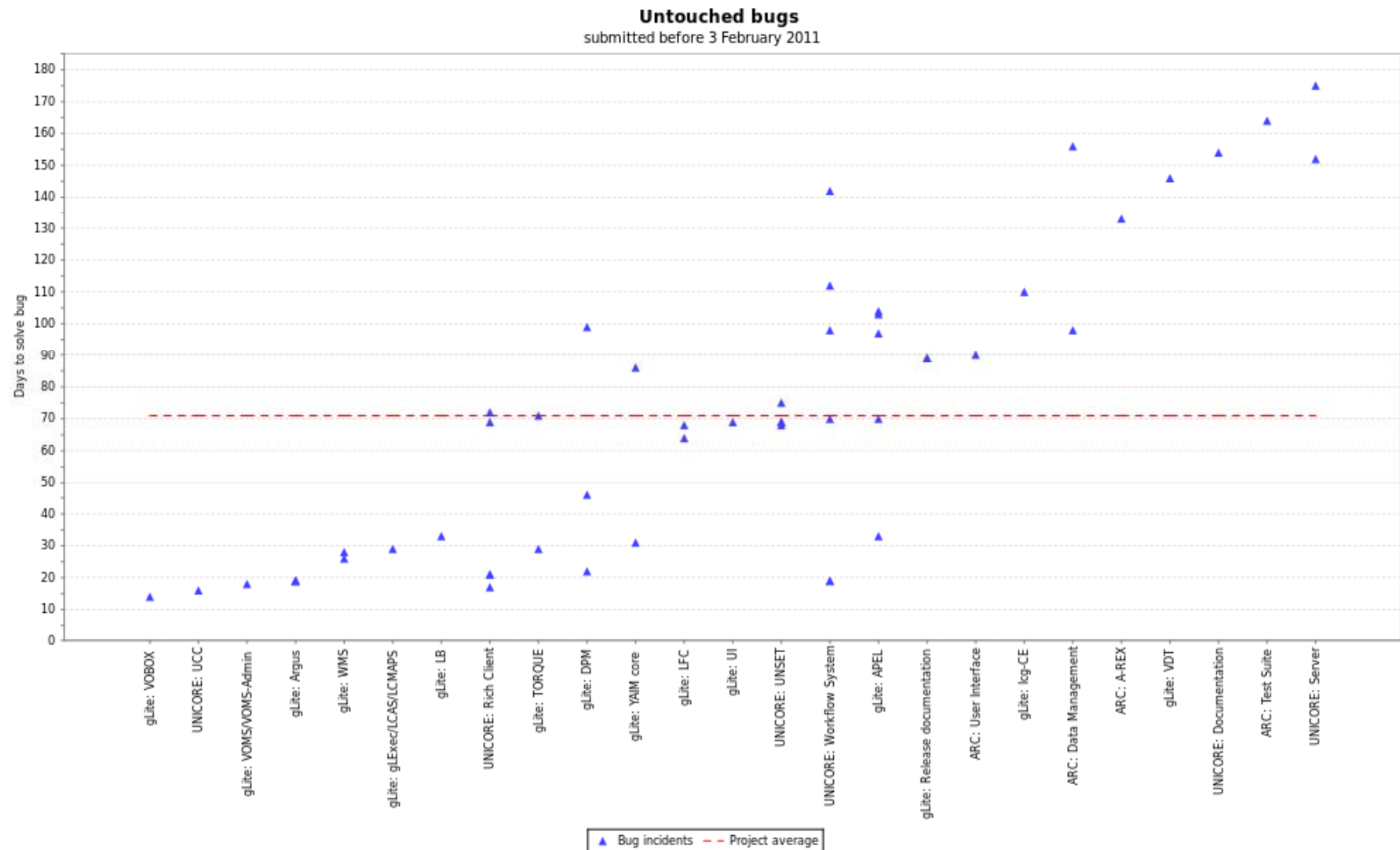
Number of untouched bugs
submitted before 10 January 2011



- In this example, X axis could contain different modules of the project or the same module in different periods of time
- We can use a logarithmic scale to be able to see small values

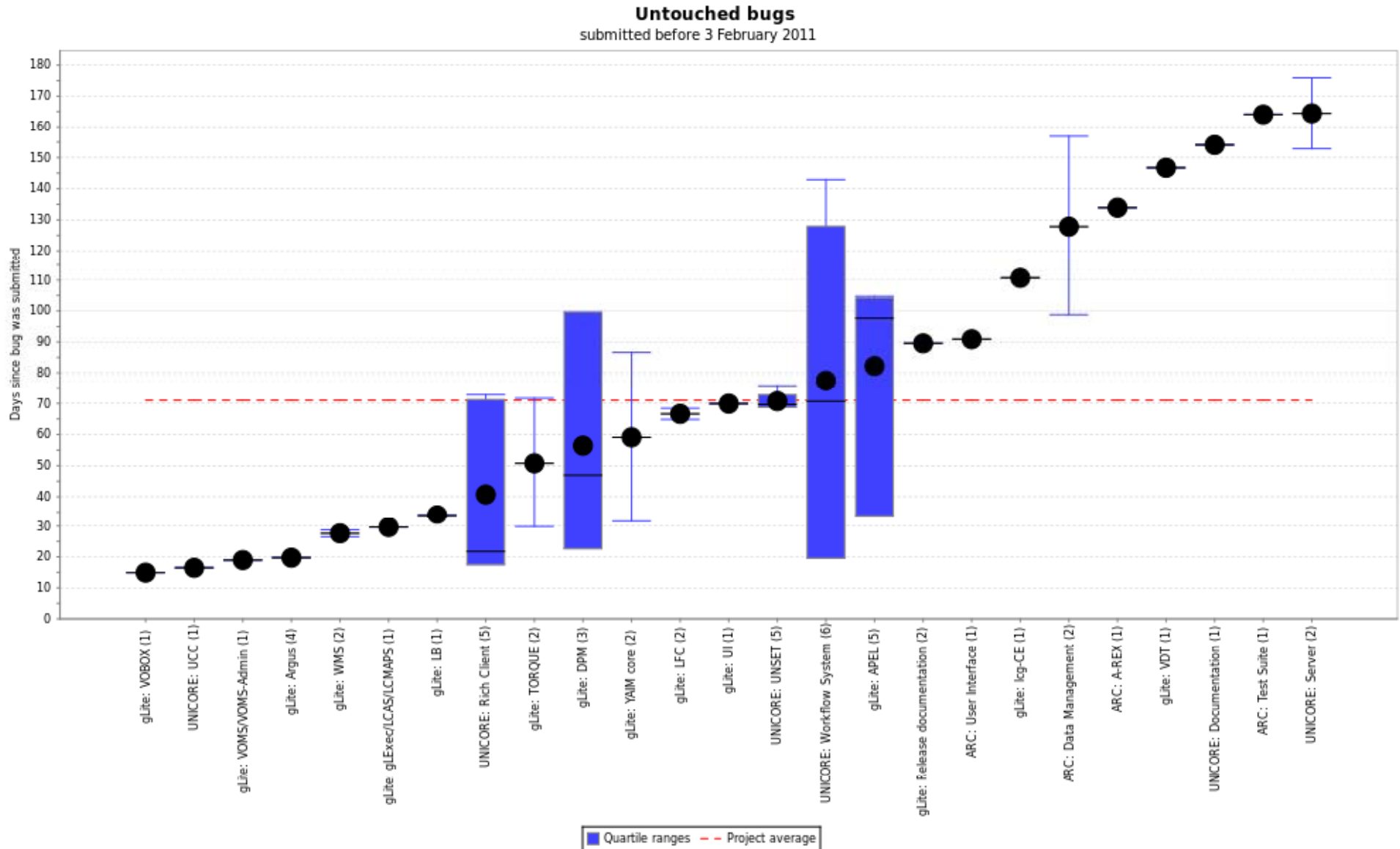
Example of final report (II)

- Open scatter chart



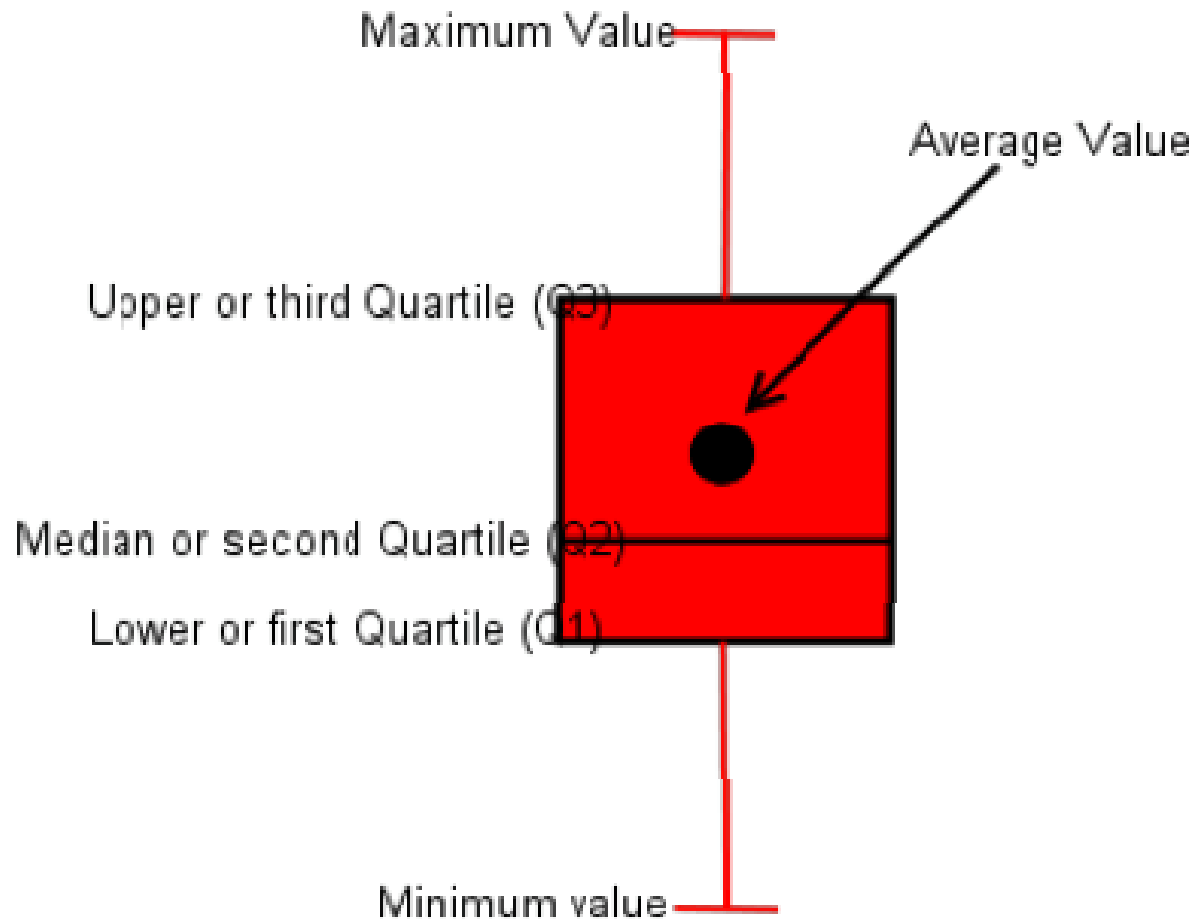
Example of final report (III)

- Box and whisker chart



Example of final report (IV)

- Box and whisker chart (detailed explanation)



ISO 25000 standard

- There is an international standard that will help us to define our metrics inside the software QA process: **ISO 25000**
 - Also known as “Software Quality Requirements and Evaluation (SQuaRE)”
 - Assumes that the software architecture design, or software development plan is already in place
 - Includes the previously existing ISO 9126 standard



Realistic goals

- Avoid impossible thresholds



Conclusion: Key points

- A software development organisation must make accurate cost estimations to improve productivity and quality
- If you do not know where you are now you certainly won't know where you will be in the future
- **To achieve accurate measurements of productivity and quality requires metrics collection and analysis**



Questions

