The SQO-OSS quality model: measurement-based open source software evaluation

Ioannis Samoladas, George Gousios,
Diomidis Spinellis, Ioannis Stamelos
Aristotle University of Thessaloniki, Greece
Athens University of Economics and Business, Greece

This work was partially supported by the European Community's
Sixth Framework Programme under the contract IST-2005-033331
Software Quality Observatory for Open Source Software

Introduction

- Software evaluation is a critical task for software professionals:
  - Decide which software is suitable for their needs
  - Decide when to deploy a new software product (when it is mature enough to release?)

- Advent of FLOSS rendered current models not applicable to some extend
  - They cannot be tuned in a FLOSS environment

Introduction

- Here we present SQO - OSS, a measurement based framework for FLOSS evaluation:
  - support for automated evaluation
  - metric oriented variables, with minimal human intervation
  - evaluates both community and code
  - weights and criteria can be tuned by evaluator, while a set of predefined profiles is available

- Rest of the presentation:
  - related work
  - model definition and evaluation process
  - example
  - the Alitheia system

Models in software evaluation

- There are many models in software evaluation, which usually are hierarchical
  - they decompose quality into an hierarchy (tree) of criteria and attributes (branches) which eventually lead to metrics (leaves)
- Examples: McCall, Boehm, ISO/IEC 9126 or ISO 2500:2005
- These models are not suitable for FLOSS evaluation:
  - open access to source code
  - peer review
  - asynchronous global development
- The need for FLOSS specific models was soon identified but these models are:
  - purpose specific
  - require substantial human intervention

OSMM

- Open Source Maturity Model assumes that FLOSS quality is proportional to its maturity
- Maturity is decomposed into six criteria:
  - Software, Support, Documentation, Training, Integration, Professional Services

- The evaluation is done through weight sum of the scores of the above criteria

- OSMM does not take into account important aspects of FLOSS such as the code itself

OpenBRR

- Open Business Readiness defines a model and a process for evaluating FLOSS
- Assessment is done in four phases:
  - Quick assessment filter
  - Target usage assessment
  - Data collection and Processing
  - Data translation
- The assessment process is organized into twelve categories, such as *Functionality, Usability, Quality, Community, Adoption and Support*
- Again result is achieved with weighted aggregation
- The problem with OpenBRR is that it requires effort from evaluator, who also assigns the marks in scores

QSOS

- Qualification and Selection of Open Source Software model has four iterative phases
  - Definition of the elements to be used by the next three steps
  - Evaluation of the software by collecting information from the community and building an identity card and an evaluation sheet
  - Qualification which involves the specifications and needs in order to select an open source software
  - Selection of the software that fulfills user's requirements
- The whole process is not too flexible and difficult to handle

Why SQO - OSS model?

- was built with focus on automation
- is the core of a continuous quality monitoring system
- does not evaluate functionality - functionality requires evaluator to play an important role in the process
- focuses on source code
- also considers community
- allows for user intervention

Model construction

There were two phases:

- Phase one: *Definition of the evaluation model*
  - Definition of the model criteria
  - Definition of the metrics

- Phase two: *Definition of the aggregation method*
  - Definition of the evaluation categories
  - Definition of the profiles of those categories

We tried to focus only on attributes that can be measured with minimal human intervention (automation)

Model definition

- We assumed that FLOSS quality depends on two critical factors: *Code* and *Community*
- In order to measure those factors and construct the model we used a simplified version of the Goal - Question - Metric (GQM) approach
- Two ultimate goals where:
  - *analyze the source code of an open source project*
  - *analyze the community of an open source project*

Model definition

For these goals we formulated questions iteratively:
"*How is source code quality measured?"*

Then by answering the questions we formulated new ones:
"How is maintainability, reliability and security measured?"

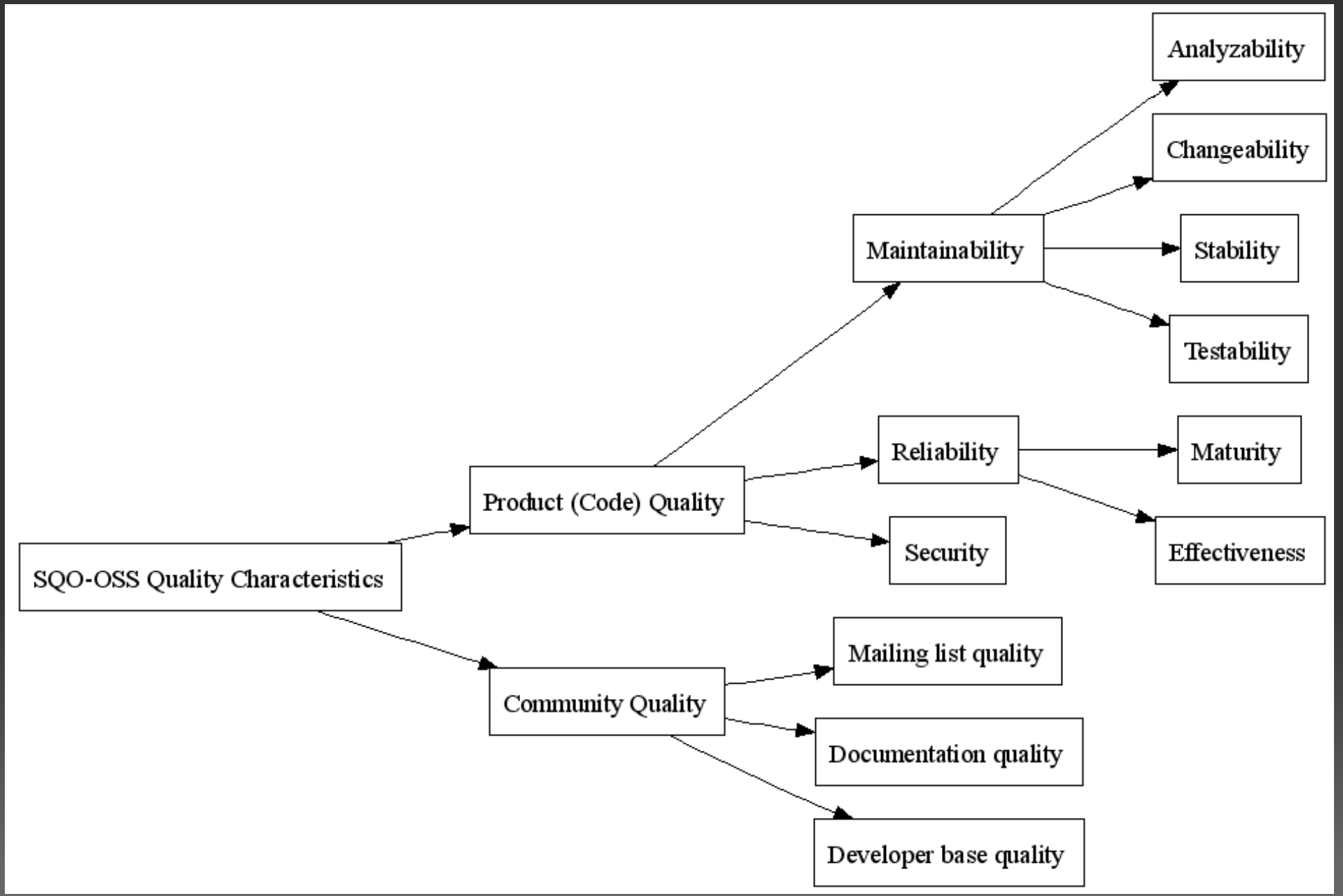
And for maintainability we followed the ISO/IEC 9126 approach:
"How is analyzability, changeability, stability and testability measured?"
We kept on formulated questions until we reached a level where attributes could be measured directly (metrics)

We chose only wide acceptable metrics and metrics suitable for automation

Model definition

- After initial construction, the SQO - OSS consortium partners (both FLOSS developers and academia) offered their comments and suggestions for further improvement
- Wiki technology allowed us for model review
- Our system allows partial evaluation (e.g. only Testability) thus we have used the same metrics in more than one category

Model hierarchy



Model metrics

Attribute	Metric
Analyzability	Cyclomatic Number
	Number of statements
	Comments frequency
	Average size of statements
	Weighted methods per class (WMC)
	Number of base classes
	Class comments frequency
Mailing list	Number of unique subscribers
	Number of messages in user/support list per month
	Number of messages in developers list per month
	Average thread depth

Evaluation process

- In order to have a result we have to combine all metrics
- For this we have used profile based evaluation, instead of a weighted average sum method such as Analytical Hierarchy Process
- The reason for doing so was that we wanted ordinal scale measures instead of interval scales that WAS uses
- We wanted results in the form of *excellent, good, fair* and *poor* - These are also our four evaluation categories (for now on *E, G, F* and *P*)

Evaluation process

- Our method, for four categories, requires the definition of three quality profiles - *E, G,* and *F*
- These profiles represent the least measurement values required for each category and they are defined seperately for each composed criterion of the model
- Thus, in order to characterize the product quality of a product as *E*, Maintainability, Reliability and Security must be also characterized as *E*.
  - When it comes for the metrics there are profiles with specific threshold values - these thresholds come from existing peer reviewed literature
- Users of the model can modify the profiles according to their needs - also they can alter the weights, although this is not recommended

Profile example

Composed Criterion	Criterion	Profile E	Profile G	Profile F	Scale
Analyzability	Cyclomatic number	4	6	8	Less is better
	Number of statements	10	25	50	Less is better
	Comments frequency	0.5	0.3	0.1	More is better
	Average size of statements	2	3	4	Less is better
Changeability	Average size of statements	2	3	4	Less is better
	Vocabulary frequency	4	7	10	Less is better
	Number of unconditional jumps	0	0	1	Less is better
	Number of nested levels	1	3	5	Less is better

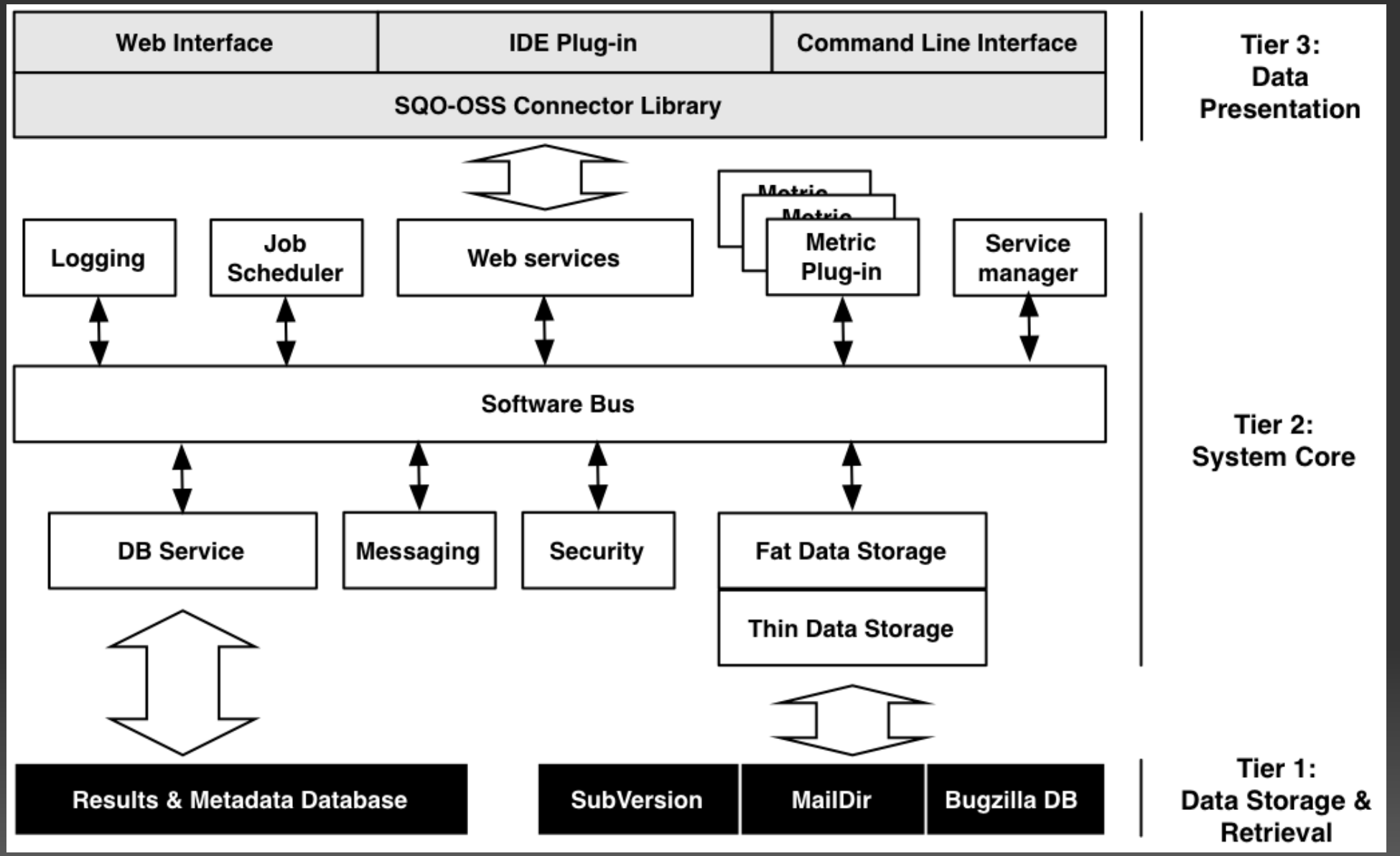
Aggregation process

- The aggregation process is done with the use of specific outranking relations iteratively with all the given profiles - *they express our decision of comparing the artifact with the profiles*
- An artifact *x* is considered to be *at least as good as* the *y* profile if and only if the “weighted” majority of the criteria agree so - there are specific tests which represent the *strength* to be reached in order to come to such decision
- There is another kind of assignment, which identifies the profile which is surely worse than *x* and assigns *x* to the previous one (for example if *x* is strictly worse than *E* then it is assigned to *G*) but this is not taken into account

Evaluation example

Composed Criterion	Project evaluation		
	CVS	**Perl**	**FreeBSD**
Analyzability	[5.63, 37.8, 0.34, 1.89]	[3.0, 36.17, 0.08, 1.02]	[3.82, 29.99, 0.12, 1.99]
Evaluation	*Good*	*Fair*	*Excellent*
Changeability	[1.89, 2.91, 0.24, 1.13]	[1.02, 2.42, 0.28, 0.53]	[1.99, 2.87, 0.51, 0.86]
Evaluation	*Good*	*Excellent*	*Excellent*
Stability	[0.24, 3.28, 1.08, 4.49]	[0.08, 3.21, 0.78, 4.75]	[0.25, 3.99, 1.23, 4.10]
Evaluation	*Poor*	*Fair*	*Poor*
Testability	[0.57, 5.62, 1.13, 0.24]	[0.46, 3.0, 0.53, 0.08]	[0.55, 3.82, 0.86, 0.25]
Evaluation	*Good*	*Good*	*Good*
Maintainability	**Good**	**Fair**	**Good**

The SQO-OSS platform



The model as SQO-OSS plugin

- Combination of:
  - Plug-in to precalculate data per project on each revision
  - (Web) UI component to apply weights
- Core Plug-in
  - Use other plug-ins to retrieve low-level metrics
  - Store results of measurements per version
- Use the project's ~700 fully mirrored project infrastructure to calibrate parameters
- Will be available from demo.sqo-oss.org soon

Conclusions - Future work

- We presented a new FLOSS quality evaluation model which focuses on automation and it is used in a real system
- Profile based evaluation, allows better selection decisions

Future work includes:

- Immediate empirical validation of the model
- Calibration of the profiles

THANK YOU FOR YOUR ATTENTION

QUESTIONS AND REMARKS ARE HIGHLY WELCOMED

Ioannis Samoladas: ioansam@csd.auth.gr