Qualification and Selection of Open Source software (QSOS)

Version 2.0 - 2013-01-19

Contents

1 License **2**

2 QSOS Manifesto **3**
2.1 About the need of a method 3
2.2 About the need of a free method 4

3 History of modifications **4**

4 Introduction **4**
4.1 Purpose 4
4.2 Target audience 5

5 General approach **5**
5.1 Four steps 5
5.2 Iterative process 6

6 Step 1 : Define **6**
6.1 Purpose 6
6.2 Type of software reference 6
6.3 Type of license reference 8

6.4 Type of community reference 9

7 Step 2 : Evaluate **9**
7.1 Purpose 10
7.2 Evaluation of a version of a piece of software 10
7.2.1 Evaluation templates 10
7.2.2 Scoring 10

8 Step 3 : Qualify **11**
8.1 Purpose 11
8.2 Filters 11
8.2.1 Identity filter 11
8.2.2 Maturity filter 11
8.2.3 Functional coverage filter 12

9 Step 4 : Select **12**
9.1 Purpose 13
9.2 Select mode 13
9.2.1 Strict selection 13
9.2.2 Loose selection 13
9.2.3 Comparison 14

10 The QSOS Project **15**
10.1 A free and community project 15
10.2 Tools and Formats 15
10.2.1 Templates 16
10.2.2 Evaluations 17
10.3 How to contribute 22

11 Appendix A : QSOS Maturity criteria **22**

12 Appendix B : DrakkR framework **25**

1 License

the “Evaluate” section.

2 QSOS Manifesto

2.1 About the need of a method

From a company point of view, choosing to opt for a piece of software as part of its information system, that piece of software being open or closed source, is based on the analysis of the needs and constraints (technical, functional and strategic) and matching the piece of software with the requirements.

However, it is necessary to have a method of qualification and selection suited to free and open source software while studying its adoption.

It is indeed extremely important to scrutinize the specific constraints and risks of the free and open source software. This domain being broad and rich, it is also necessary to have a method of qualification that allows to differentiate the often many potential matching components, from a technical, functional and strategic point of view.

In addition to these “natural” questions :

- Which piece of software matches my current and anticipated technical needs ?
- Which piece of software matches my current and anticipated functional needs ?

Here are some questions that every company should ask before making a decision :

- What is the continuity of this piece of software ? What are the odds of a fork ? How to anticipate and manage it ?
- What is the level of stability to expect? How to manage dysfunctions ?
- What is the required and available level of support of this piece of software ?
- Is it possible to influence the software development (adding new features) ?

To serenely answer these questions and make an informed decision while managing the risks, it is required to have a method that allows to :

- qualify a piece of software by integrating the specificity of free and open source software ;
- compare several piece of software depending of the needs and weighting criteria to make a final decision.

Those different points led Atos to design and formalize the method of Qualification and Selection of Open Source software (QSOS).

2.2 About the need of a free method

In our opinion, such a method must be distributed to anyone under a free license. Only such a license is indeed able to guarantee the promotion of the free and Open Source movement, through :

- the possibility of the reuse of the evaluations by anyone ;
- the quality and objectivity of the generated documents, that can always be improved through transparency and peer review.

Hence, Atos decided to distribute the QSOS method and the generated documents (templates and identity cards) under the *GNU Free Documentation License*.

The tools developed to easily apply this method are distributed under the *GNU General Public License*.

3 History of modifications

Version	Date	Authors	Comments
1.0	2004	Raphaël Semeteys	Design and first draft.
1.1	2004	Olivier Pilot	Design and review.
1.2	2004	Laurent Baudrillard	Design and review.
1.3	2004-11-17	Raphaël Semeteys	First public release.
1.4	2005-11-23	Raphaël Semeteys Olivier Pilot	Licensing. History. New logo.
1.5	2006-01-19	Gonéri Le Bouder Raphaël Semeteys	Switch to LaTeX. GNU FDL License. QSOS Manifesto.
1.6	2006-04-13	Gonéri Le Bouder	Update of the Maturity section.
2.0	2013-01-19	Raphaël Semeteys Philippe-Arnaud Haranger	Switch to Markdown. Formats and tools. Update of the Maturity section.
2.0	2013-02-27	Thomas Moreau	Translation into English.

4 Introduction

4.1 Purpose

This document deals the method, named « QSOS » (Qualification and Selection of Open Source software), designed by Atos to qualify and select the Open Source software within the framework of software support and technology watch.

This method can be used as part of a more general approach of technology watch that is not discussed here, and describes a process of generation of identity cards and evaluation of free and open source software.

4.2 Target audience

The target audience is :

- people inquiring on the method either as professionals or as non-professionals ;
- free and open source communities ;
- IT experts wanting to know and apply this method in day-to-day activities of evaluation and selection of components in order to build software solutions meeting their own or their customers' needs ;

5 General approach

5.1 Four steps

The general approach of the QSOS Method is composed of four interdependent steps.

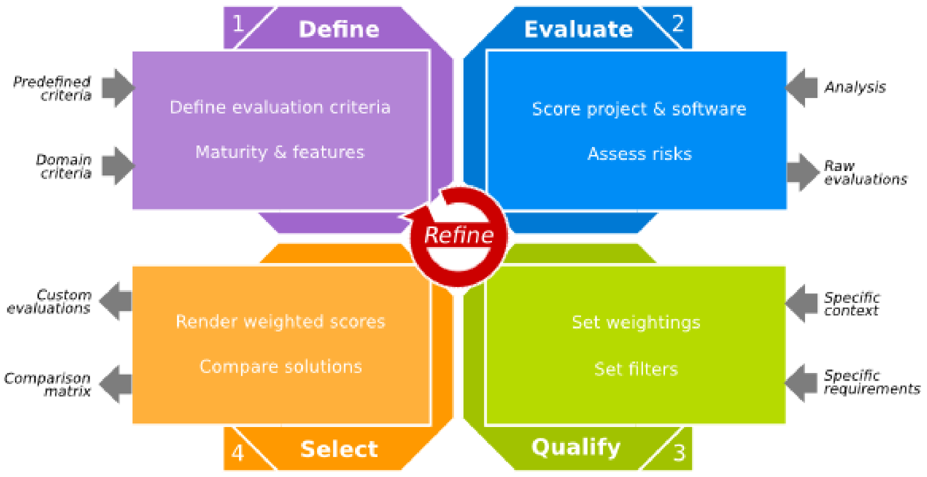


Figure 1: General approach of QSOS

Step	Description
Define	Definition and update of the reference used for the next steps

Step	Description
Evaluate	Evaluation of a version of a piece of software (functional coverage and maturity of the project).
Qualify	Weighting the criteria according to the context.
Select	Comparison and selection of software, based on previous steps' data.

Every step is detailed further.

5.2 Iterative process

The general approach can be applied with different granularities in order to be more or less detailed. It is also possible to be more detailed in every iteration.

6 Step 1 : Define

6.1 Purpose

The purpose of this step is to define different elements of typology that will be used during the next three steps of process.

The different typological reference are :

- type of software : the hierarchical classification of types of software and the description of functional coverage linked to every type in the form of templates ;
- type of license : classification of types of free and open source licenses in use ;
- type of community : classification of types of community organizations around the software to ensure the life cycle.

6.2 Type of software reference

It's the reference that change the most because, as software change, it offer new features that have to be added.

The templates of this reference is composed of hierarchical criteria, grouped by axes :

- maturity analysis of the project in charge of the software development ;
- functional coverage analysis of the software.

The QSOS method defines and imposes the maturity criteria of a project.

These criteria must be used in every single QSOS evaluation. They are detailed in the appendix of this document.

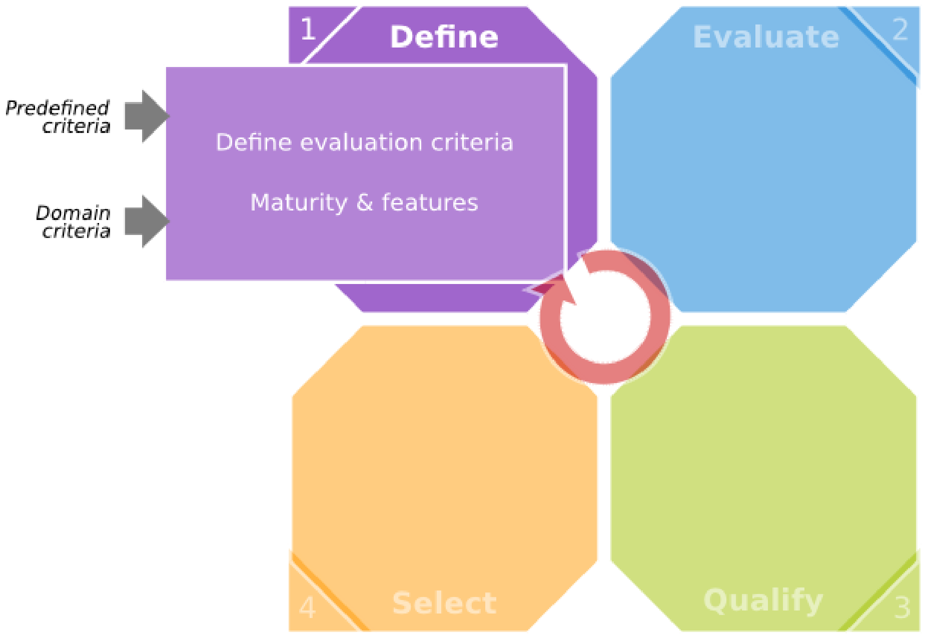


Figure 2: Position in the process

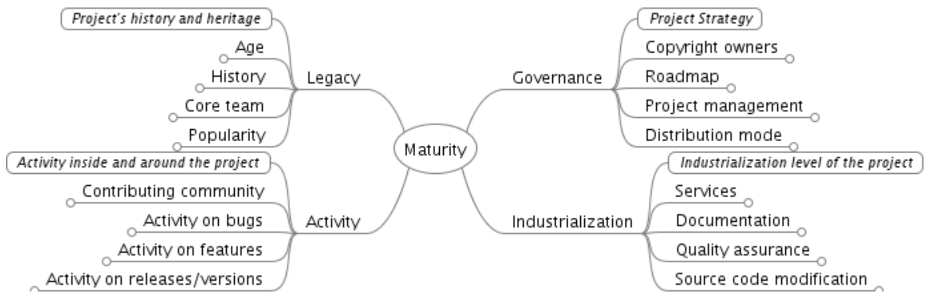


Figure 3: Maturity criteria of a project

The other criteria are specific to a functional domain of which the evaluated software belong.

Visit the website http://www.qsos.org to see the available templates and to be guided to the creation of new templates.

6.3 Type of license reference

There are many free and open source licenses. The purpose of this reference is to identify and categorize them according to the following axes:

- Copyleft : can derivative works become proprietary or have to stay under the same conditions ?
- Virality : does the use of the software from a module implies that this module has to be under the same license ?
- Inheritance : does the derivative work inherit from the license or is it possible to add restrictions ?

The following array list the mostly used licenses with the properties described above:

License	Copyleft	Virality	Inheritance
GNU Public License	Yes	Yes	Yes
CeCILL	Yes	Yes	Yes
LGPL	Yes	Partial	Yes
BSD and BSD-like	No	No	No
Artistic	No	No	No
MIT	No	No	No
Apache Software License	No	No	No
Mozilla Public License	Yes	No	Yes
Common Public License	Yes	No	No
Academic Free License	No	No	No
PHP License	No	No	No
Open Software License	Yes	No	No
Zope Public License	No	No	No
Python SF License	No	No	No

You can visit the website of *SLIC*[1] (Software LIcense Comparator) for a more detailed description of the free and open source software and their compatibilities.

NB: software can be distributed under several licenses, including proprietary ones.

[1]http://slic.drakkr.org

6.4 Type of community reference

So far, the type of identified communities are:

- sole developer : the software is developed by a sole person ;
- group of developers : several person working together without formal processes ;
- developers organization : a group a developers using a software lifecycle management system formalized and respected, based on roles (developers, validator, delivery manager...) and meritocracy ;
- legal entity : a legal entity, often a non for-profit, manages the community and the sponsorship and holds the copyrights.
- Commercial entity : a commercial entity employs the core developers of the project and gets revenue from the sale of services or commercial version of the software.

7 Step 2 : Evaluate

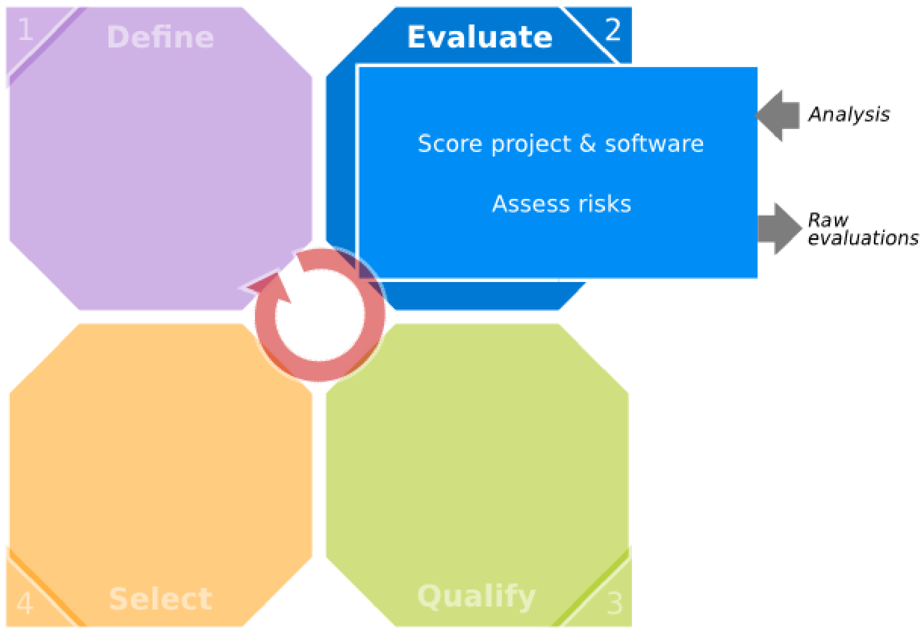


Figure 4: Position in the process

7.1 Purpose

The purpose of this step is to evaluate free and open source software. Information on the open source community are retrieved in order to score the software based on the criteria from the previous step. This analysis grid or template is then a tree of criteria.

This method can be used as part of a more general approach of technology watch that is not entirely discussed here.

7.2 Evaluation of a version of a piece of software

Every version of a piece of software is described in an identity card. This card contains, in addition to the name of the software and its version, information, a description and a detailed analysis of provided features.

7.2.1 Evaluation templates

The QSOS evaluations are done from the templates that describe the different criteria and their structure. The evaluation criteria of the maturity of the project are imposed by the method and described further. They are completed by criteria describing expected features of the type of the evaluated software.

7.2.2 Scoring

Criteria are assigned a discrete score from 0 to 2.

The evaluation templates contain the meaning of the three scores 0, 1 and 2 for every criterion. Regarding the functional coverage, the scoring rule is usually the following :

Score	Description
0	Functionality not covered.
1	Functionality partially covered.
2	Functionality fully covered.

These scores will be used in the selection step to compare and filter the software depending on the weighting specified during the qualification step.

It is possible to apply the general approach in an iterative way. At the evaluation level, it means to have the possibility to score the criteria several times. It then doesn't block the general process when only parts of the notes are available. When all criteria are evaluated, the score of the first two levels are then recalculated.

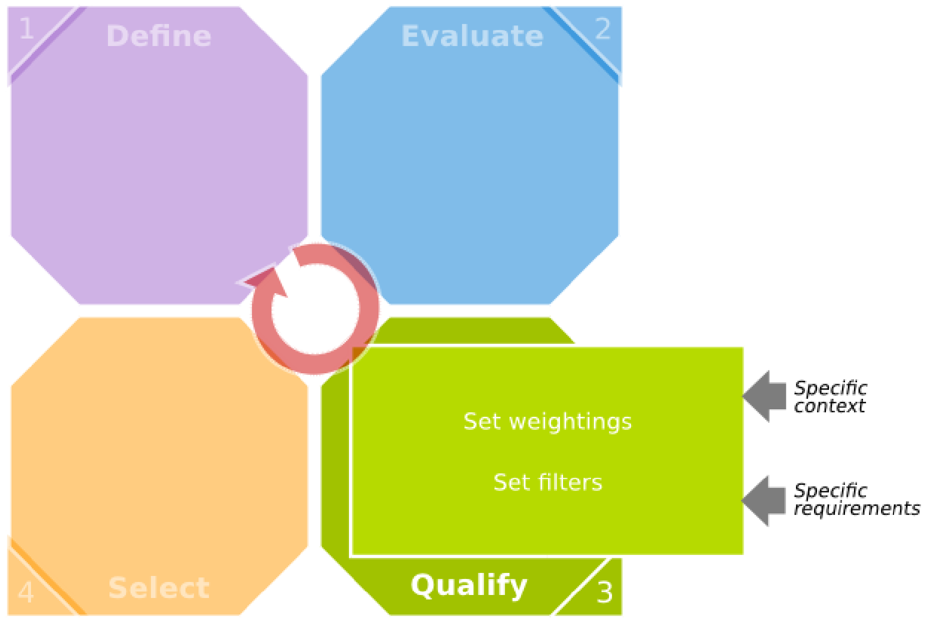


Figure 5: Position in the process

8 Step 3 : Qualify

8.1 Purpose

The purpose of this step is to define a set of elements translating the needs and constraints lined to the selection approach of a piece of open source software. The context in which the software will be used has to be set, in order to get a filter used in the Selection step.

8.2 Filters

8.2.1 Identity filter

The first level of filtering can be set on the data relative to the software identity. It can be, for example, to consider only the software of a certain type, or only distributed under a specific license.

8.2.2 Maturity filter

Filter on the maturity of the project

The degree of relevance of every maturity criterion is set depending on the context :

- not relevant criterion, not to be included in the filter ;

- relevant criterion ;
- critical criterion.

This degree of relevance will be translated into a weighting value in the next step of the process, depending on the chosen selection mode.

8.2.3 Functional coverage filter

Every functionality described in the evaluation template is assigned a level of requirement, in the following list:

- required functionality ;
- optional functionality ;
- not required functionality.

These requirements will be associated to weighting values during the Select step, depending on the chosen selection mode.

9 Step 4 : Select

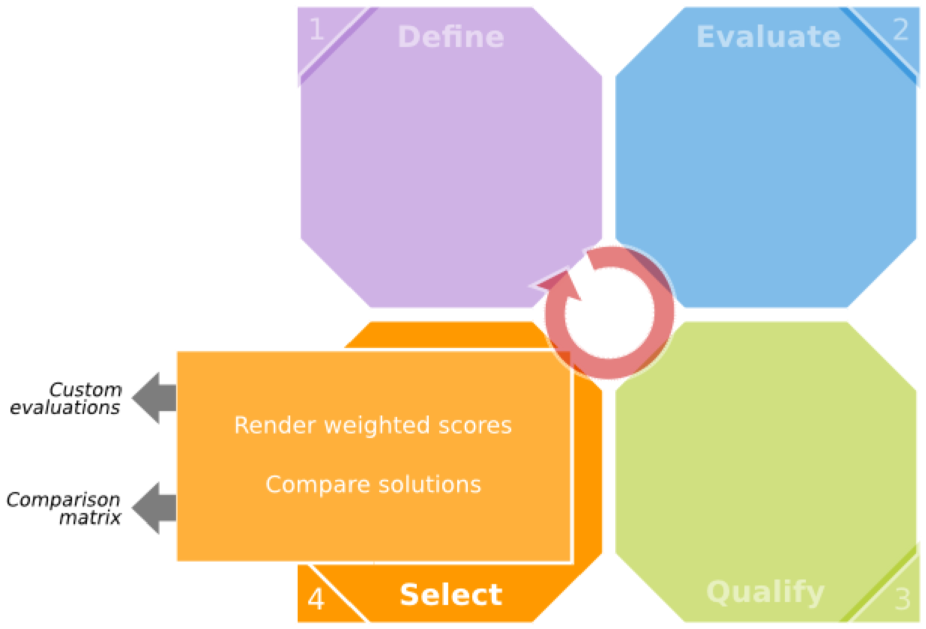


Figure 6: Position in the process

9.1 Purpose

The purpose of this step is to select the software matching the user's needs, or to compare the software of the same type.

9.2 Select mode

Two modes are available :

- strict selection ;
- loose selection.

9.2.1 Strict selection

the strict selection is made by a process of elimination as soon as a piece of software does not comply with the demands :

- elimination of the software that don't go through the identity filter ;
- elimination of the software that don't provide the required functionalities ;
- elimination of the software whose maturity criteria don't match with the degree of relevance defined by the user :
  - the score of a relevant criterion must be greater than or equal to 1 ;
  - the score of a critical criterion must be equal to 2.

Depending on the demands of the user, this strict selection can return no eligible software.

The software that went through the selection process are then assigned a global weighted score, in the same way as the loose selection.

9.2.2 Loose selection

This selection is less strict than the previous one because instead of eliminating software that are non eligible, it sort them while measuring the difference compared to the filters previously defined.

It is based on the weighting values whose rules are detailed in the following paragraphs.

Weighting of functionalities

The weighting factor is based on the level of requirements of every functionality of the functionality coverage.

Level of requirement	Weighting
Required functionality	3
Optional functionality	1

Level of requirement	Weighting
Not required functionality	0

Weighting of maturity

The weighting factor is based on the degree of relevance of every maturity criteria.

Degree of relevance	Weighting
Critical criterion	3
Relevant criterion	1
Not relevant criterion	0

9.2.3 Comparison

The software of the same domain may also be compared with one another according to the weighted score from the previous steps.

The following figures show possible synthesis. The O3S application, described further, allows to export the comparisons in different format (OpenDocument, HTML and SVG).

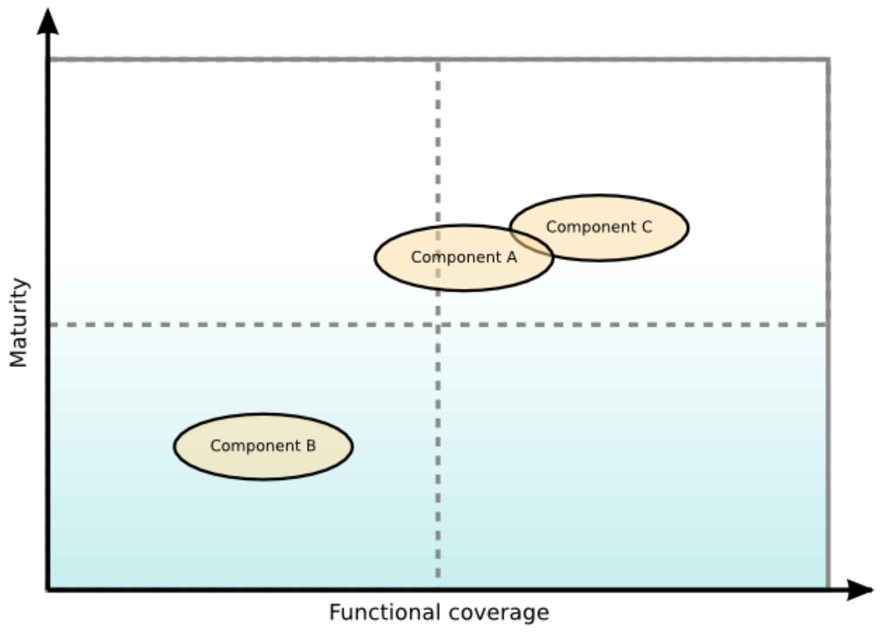


Figure 7: QSOS Quadrant (SVG format)

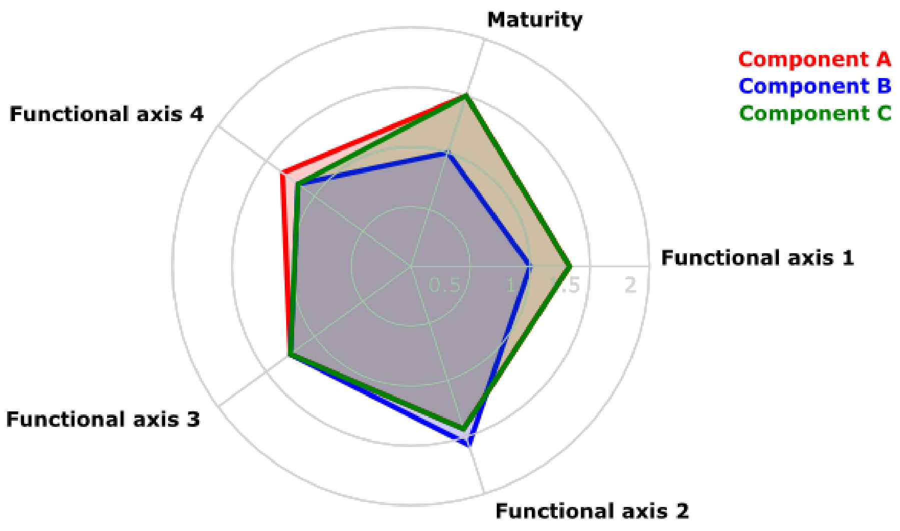


Figure 8: QSOS Radar (SVG format)

10 The QSOS Project

10.1 A free and community project

In addition to the method, QSOS is also a free and community project dedicated to the collaborative watch of free and open source software.

Hence, the main purposes of the project are :

- manage the evolution of the method and of the evaluation file format ;
- centralize the references, notably the storage of templates, the identity cards and evaluations ;
- provide tools to help apply the QSOS method ;
- Help user using the method via best practices and communication hubs.

10.2 Tools and Formats

The free project QSOS also provide tools to apply the process of the method and to make the collaboration easy.

The diagram below describes the existing tools and formats.

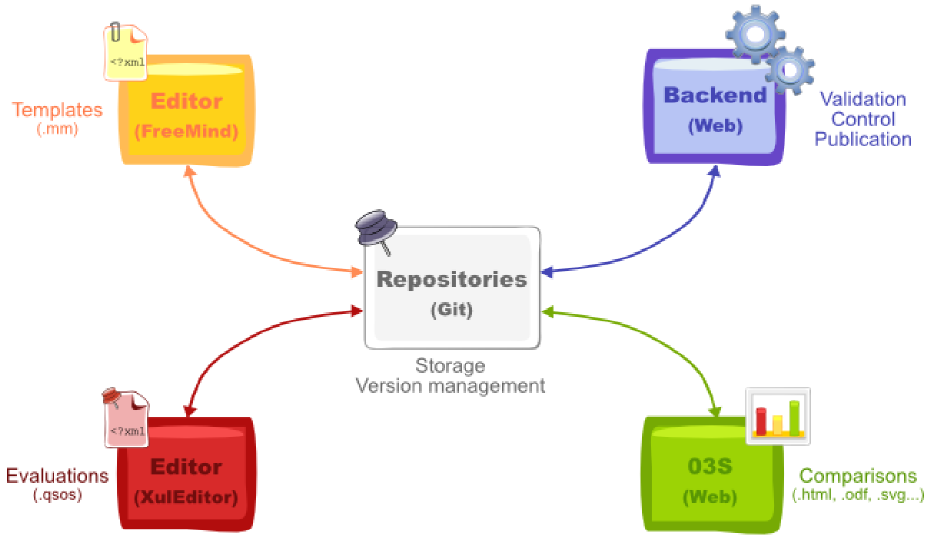


Figure 9: QSOS formats and tools

10.2.1 Templates

FreeMind

The template are functional coverage grids specific to a software family. Before evaluating a piece of software, a well-suited template is needed.

The QSOS project uses mind maps to design and document its templates. The free software FreeMind[2] has been chosen because of its portability and its XML format allowing the transformation of the templates into `.qsos` format (described hereafter) thanks to XSL.

`.mm` Format

The template are described and stored in the format used by FreeMind (`.mm`).

This format is described on the freemind official website . It's an XML format used by QSOS as a pivot format for templates. The blank evaluations used to analyze software are generated from this format via XSL transformations.

The mind maps representing QSOS templates must comply with a specific formalism in order to be transformed in evaluations.

1. the criteria descriptions must be bubbles (Format/Bubble with FreeMind) ;
2. the score descriptions (0, 1 and 2).

[2]http://freemind.sourceforge.net

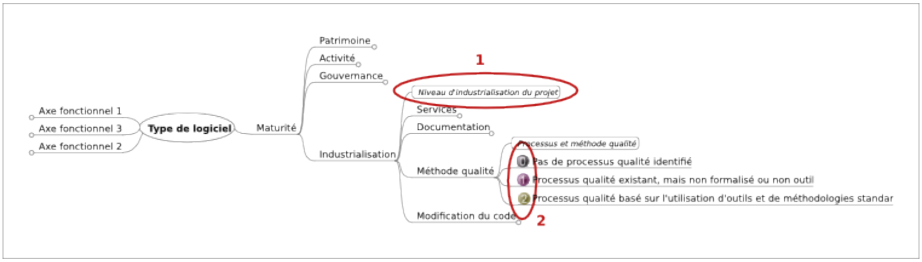

Patrimoine
Activité
Gouvernance
1
Niveau d'industrialisation du projet
Axe fonctionnel 1
Axe fonctionnel 3
Axe fonctionnel 2
Type de logiciel
Maturité
Services
Documentation
Industrialisation
Processus et méthode qualité
Méthode qualité
0 Pas de processus qualité identifié
1 Processus qualité existant, mais non formalisé ou non outil
2 Processus qualité basé sur l'utilisation d'outils et de méthodologies standar
Modification du code
2

Figure 10: Criteria description

The XSL file allowing to transform templates into evaluations is available on the QSOS project website. FreeMind allows to apply the transformation via the menu File/Export/Using XSLT. . .

10.2.2 Evaluations

XulEditor

XulEditor is a QSOS evaluation entry and management tool. It allows to perform the following actions :

- create a new evaluation from a template in `.mm` format (local template or from the QSOS reference) ;
- open and modify an existing evaluation (local evaluation or from the QSOS reference) ;
- apply a new version of a template on an evaluation (without losing existing data) ;
- save an evaluation (locally or in the QSOS reference).

XulEditor doesn't allow to modify a `.mm` template and only deals with evaluations in the `.qsos` format.

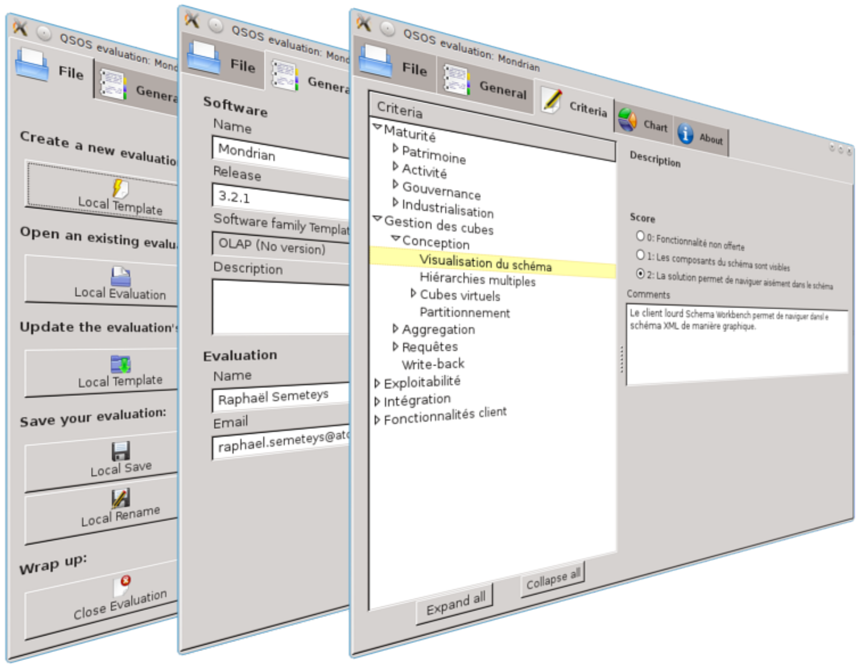
It is an application using the Mozilla technology. It can be deployed as an extensions for the web browser Firefox or as a XulRunner application.

Visit the QSOS project website for more details on the installation of XUlEditor.

O3S (Open Source Selection Software)

03S is a web application allowing to visualize, weight and compare QSOS evaluations according to the method process. It allows to visualize, compare and export the QSOS evaluations in OpenDocument format, and to generate charts in SVG format.

It is available online at this URL : http://www.qsos.org/o3s/.



Software

Name: Mondrian

Release: 3.2.1

Software family Templat: OLAP (No version)

Description

Evaluation

Name: Raphaël Semeteys

Email: raphael.semeteys@at

Description

Score

0: Fonctionnalité non offerte

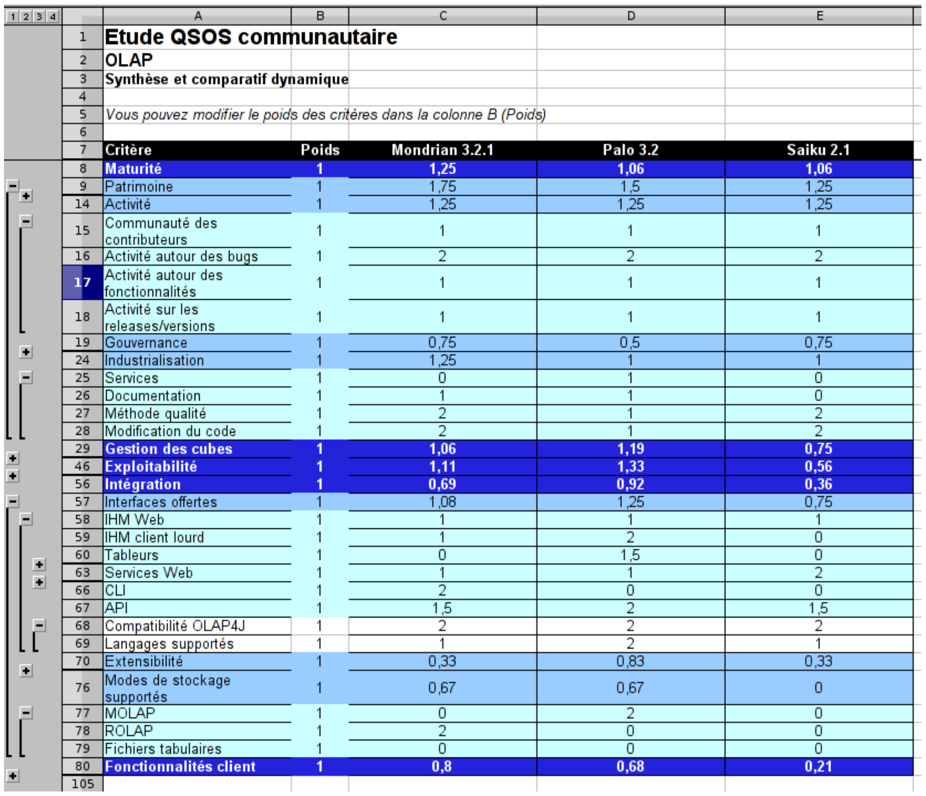
1: Les composants du schéma sont visibles

[✓ 2: La solution permet de naviguer aisément dans le schéma]

Comments

Le client lourd Schema Workbench permet de naviguer dansl e schéma XML de manière graphique.

Figure 11: XulEditor screenshots



	A	B	C	D	E
1	**Etude QSOS communautaire**				
2	OLAP				
3	**Synthèse et comparatif dynamique**				
4					
5	*Vous pouvez modifier le poids des critères dans la colonne B (Poids)*				
6					
7	**Critère**	**Poids**	**Mondrian 3.2.1**	**Palo 3.2**	**Saiku 2.1**
8	**Maturité**	**1**	**1,25**	**1,06**	**1,06**
9	Patrimoine	1	1,75	1,5	1,25
14	Activité	1	1,25	1,25	1,25
15	Communauté des contributeurs	1	1	1	1
16	Activité autour des bugs	1	2	2	2
17	Activité autour des fonctionnalités	1	1	1	1
18	Activité sur les releases/versions	1	1	1	1
19	Gouvernance	1	0,75	0,5	0,75
24	Industrialisation	1	1,25	1	1
25	Services	1	0	1	0
26	Documentation	1	1	1	0
27	Méthode qualité	1	2	1	2
28	Modification du code	1	2	1	2
29	**Gestion des cubes**	**1**	**1,06**	**1,19**	**0,75**
46	**Exploitabilité**	**1**	**1,11**	**1,33**	**0,56**
56	**Intégration**	**1**	**0,69**	**0,92**	**0,36**
57	Interfaces offertes	1	1,08	1,25	0,75
58	IHM Web	1	1	1	1
59	IHM client lourd	1	1	2	0
60	Tableurs	1	0	1,5	0
63	Services Web	1	1	1	2
66	CLI	1	2	0	0
67	API	1	1,5	2	1,5
68	Compatibilité OLAP4J	1	2	2	2
69	Langages supportés	1	1	2	1
70	Extensibilité	1	0,33	0,83	0,33
76	Modes de stockage supportés	1	0,67	0,67	0
77	MOLAP	1	0	2	0
78	ROLAP	1	2	0	0
79	Fichiers tabulaires	1	0	0	0
80	**Fonctionnalités client**	**1**	**0,8**	**0,68**	**0,21**
105					

Figure 12: Export in OpenDocument Spreadsheet format

It is also possible to install a local 03S instance in your organization. Visit the QSOS project website to go further.

`.qsos` format

The evaluations are described and stored in an XML pivot format specific to QSOS. The XML schema is available on the QSOS project website. This chapter describes the structure principles.

The file extension is `.qsos`.

The main tag is `<document/>`, it contains :

- a header `<header/>` containing the metadata related to the evaluation (author, language, used template, QSOS version, template version, date of creation, date of validation. . .) ;
- one or several sections (`<section/>`) :
  - each section is composed of evaluation criteria (`<element/>`) that can be nested, and descriptions (`<desc/>`) ;
  - in this tag tree, the leaves (criteria that don't have children) contain the meaning of the scores O, 1 and 2 (`<desc0/>`, `<desc1/>` and `<desc2/>`), the score (`<score/>`) and a comment to give the reason and to cite the sources(`<comment/>`).

An example of the structure is below :

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
   <header>
      <authors>
         <author>
            <name>Name of the author</name>
            <email>Email address of the author</email>
         </author>
         <!-- Other <author/>  -->
      </authors>
      <dates>
         <creation>Date of Creation</creation>
         <validation>Date of Validation</validation>
      </dates>
      <appname>Application/software name</appname>
      <desc>short description of the software</desc>
      <release>Software version</release>
      <licenseid>Identifier of the main license</licenseid>
      <licensedesc>Name of the main license</licensedesc>
      <url>URL of the software website</url>
      <demourl>URL of the demo website</demourl>
      <language>language of the evaluation : en, fr...</language>
      <qsosappname>CPE identifier of the version</qsosappname>
      <qsosformat>used QSOS format, currently: 2.0</qsosformat>
      <qsosspecificformat>template version</qsosspecificformat>
      <qsosappfamily>template name</qsosappfamily>
   </header>
   <section name="maturity" title="Maturity">
      <!-- <section/> imposed by QSOS -->
   </section>
   <section name="Unique-identifier-1" title="Section name">
      <element name="Unique-identifier-2" title="Criterion name">
         <desc>Description of the criterion</desc>
         <element name="Unique-identifier-3" title="Sub-criterion name">
         <desc>Description of the sub-criterion</desc>
            <desc0>meaning of the score 0</desc0>
            <desc1>meaning of the score 1</desc1>
            <desc2>meaning of the score 2</desc2>
            <score>Score : 0, 1 or 2</score>
            <comment>Reason and sources</comment>
         </element>
         <!-- Other <element/> -->
      </element>
      <!-- Other <element/> -->
   </section>
   <!-- Other <section/> -->
</document>
```

So, it is an XML tree composed of a header (`<header/>`) and of sections (`<section/>`) containing elements (`<element/>`). The leaves of this tree are criteria that can be scored *0*, *1* or *2*.

This format is used as a pivot by the tools provided by the QSOS project to export into other formats, such as HTML, SVG or OpenDocument.

The detailed structure of this format is described in an XSD schema, available on the QSOS project website.

QSOS reference and engine

The QSOS engine is a chain of tools to validate, control and publish the QSOS evaluations and the templates stored in the reference.

The reference is composed of two Git repositories dedicated to the storage of the evaluations templates :

- the *Incoming* repository : reserved for publication, sharing and manipulation of the evaluations and templates by the community. It is available to everyone via O3S and only requires the creation of a user account ;
- the *Master* repository : dedicated to the storage of the evaluations and templates considered as good quality elements and that have been approved

by a moderator of the QSOS community.

In addition to these two repositories reserved for the documents generated and used by the QSOS method, the project also use a Git repository for the development of its tools and another one for its documentation.

The documentation is written in Markdown[3], used as a pivot format by Pandoc[4] to export into PDF and HTML, and by Gitit[5] for the wiki of the project.

To sum up, the Git repositories are :

Repository	Purpose
QSOS.git	Tools and formats
QSOS-Incoming.git	Templates and evaluations in sandbox mode
QSOS-Master.git	Templates and evaluations approved by the community
Drakkr.git	Documentation of QSOS and the other projects within Drakkr[6]

Visit the QSOS project's website to clone these repositories.

10.3 How to contribute

The purpose of the QSOS project is to mutualize the effort on the free and open source software watch. It is then resolutely a community project: the more contributors there are, the greater are the number, the quality and the objectivity of the evaluations.

You can contribute to the project by :

- creating or updating templates and evaluations ;
- translating templates, evaluations or the documentation ;
- getting involved in the development of the tools ;
- promoting the method and the project.

Visit the QSOS project's website for more details on the governance of the QSOS community.

11 Appendix A : QSOS Maturity criteria

This set of criteria, named "Maturity", is compulsory for every template and every evaluation in the QSOS 2.0 format.

[3]http://daringfireball.net/projects/markdown/
[4]http://johnmacfarlane.net/pandoc/
[5]http://gitit.net
[6]See appendix B for more details on the Drakkr framework

- Legacy : Project's history and heritage
  - Age :
    - 0 : Less than three months
    - 1 : Between three months and three years
    - 2 : More than three years
  - History :
    - 0 : The software has many problems which can be prohibitive
    - 1 : No major crisis, or unknow history
    - 2 : Good past experience in crisis management
  - Core team :
    - 0 : Very few identified core developers
    - 1 : Few active core developers
    - 2 : Important and identified core development team
  - Popularity :
    - 0 : Very few identified users
    - 1 : Usage can be detected
    - 2 : Many known users and references
- Activity : Activity inside and around the project
  - Contributing community :
    - 0 : No real community nor activity (forum, mailing lists...)
    - 1 : Community with significant activity
    - 2 : Strong community with vivid activity in forums, with many contributors and supporters
  - Activity on bugs :
    - 0 : Low reactivity in forums and mailing lists, or no mention about bugfixies in release notes
    - 1 : Existing activity but without any clearly defined process or with long resolution times
    - 2 : Strong reactivity based on roles and task assignments
  - Activity on features :
    - 0 : Few or no new features
    - 1 : Product's evolution is led by a dedicated team or by users, but without a clearly stated process
    - 2 : Feature request process is industrialized, an associated roadmap is available
  - Activity on releases/versions :
    - 0 : Very low activity on the production or development versions (alpha, beta)
    - 1 : Activity on production or development versions (alpha, beta) with frequent minor corrective versions
    - 2 : Important activity with frequent corrective versions and planned major versions linked with the roadmap
- Governance : Project's strategy
  - Copyright owners :
    - 0 : Rights are being held by a few individuals or commercial entities

    - 1 : Rights are uniformly held by many individuals
    - 2 : Rights are held by a legal entity or a foundation that the community trust (ex: FSF, Apache, ObjectWeb)
  - Roadmap :
    - 0 : No roadmap is published
    - 1 : Roadmap without planning
    - 2 : Versioned roadmap with planning and delay measurements
  - Project management :
    - 0 : No clear and apparent project management
    - 1 : Project managed by an individual or a single commercial entity
    - 2 : Strong independance of the core team, rights held by a recognized entity
  - Distribution mode :
    - 0 : Dual distribution with a commercial version along with a functionally limited free one
    - 1 : Subparts are only available under proprietary license (core, plugins. . .)
    - 2 : Completely open and free distribution
- Industrialization : Industrialization of the project
  - Services : Existing service offerings (support, training, audit. . .)
    - 0 : No service offering identified
    - 1 : Limited service offering (geographically, to a single language, to a single provider or without warranty)
    - 2 : Rich ecosystem of services provided by multiple providers, with guaranteed results
  - Documentation :
    - 0 : No user documentation
    - 1 : Documentation exists but is partly obsolete or restricted to one language or to few details
    - 2 : Documentation up to date, translated and possibly adapted to several target readers (enduser, sysadmin, manager. . .)
  - Quality assurance : QA process
    - 0 : No QA process identified
    - 1 : Existing QA processes, but they are not formalized or equiped
    - 2 : QA process based on standard tools and methodologies
  - Source code modification :
    - 0 : No convenient way to propose source code modifications
    - 1 : Tools are provided to access and modify the code (eg SCM, forge. . .) but are not really used by core team to develop the product
    - 2 : The contributing process is well defined, exposed and respected, it is based on clearly defined roles

12 Appendix B : DrakkR framework

QSOS is a sub-project of the DrakkR initiative aiming to build a free framework dedicated to the open source governance within companies and administrations.

In addition to QSOS, dedicated to the assessment of the free and open source software, DrakkR is also composed of other methods and tools to implement such a governance.

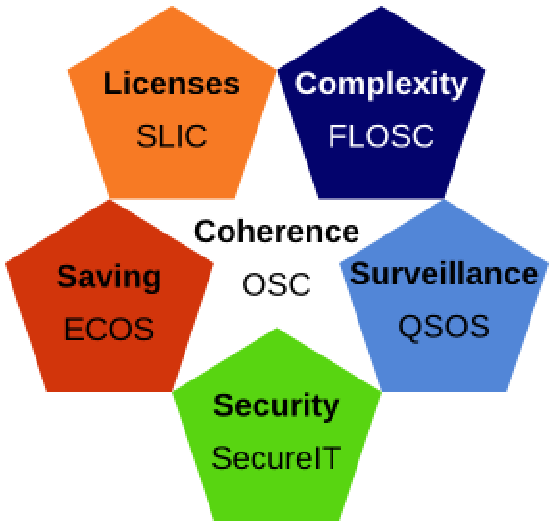


Figure 13: DrakkR Framework

- **OSC** (Open Source Cartouche) : sub-project dedicated to the unique identification of a version of a piece of open source software and the management of its metadata ;
- **ECOS** (Evaluation of the Costs linked to Open Source) : sub-project regarding the evaluation and the calculation of the Total Cost of Ownership of a piece of open source software as well as the Return Of Investment of a migration ;
- **FLOSC** (Free/Libre Open Source Complexity) : sub-project providing method and tools to evaluate intrinsic complexity of open source components ;
- **SLIC** (Software LIcense Comparator) : sub-project dedicated to the formal description of open source licenses and its relative compatibilities ;
- **SecureIT** : sub-project dedicated to the management of security notifications in open source software.

Visit the web site of the project to go further : http://drakkr.github.io.