Méthode de Qualification et de Sélection de logiciels Open Source (QSOS) version 1.6

Avril 2006

Résumé

Ce document décrit la méthode QSOS, conçue pour qualifier, sélectionner et comparer les logiciels libres et open source de manière objective, traçable et argumentée.
Elle est mise à disposition de tous selon les termes de la **GNU Free Documentation Licence**.
Ce document ainsi que ses fichiers sources LaTeX sont disponibles sur `http://www.qsos.org`.

Table des matières

1 Note de licence

2 Manifeste QSOS

2.1 De la nécessité d'une méthode

Pour une entreprise, le choix d'opter pour un logiciel comme composant de son système d'information, que ce logiciel soit Open Source ou commercial, repose sur l'analyse des besoins et contraintes (techniques, fonctionnels et stratégiques) puis de l'adéquation du logiciel à ces besoins et aux contraintes exprimés.

Toutefois, dès lors que l'on envisage d'étudier l'adéquation de logiciels Open Source, il est nécessaire de disposer d'une méthode de qualification et de sélection adaptée aux spécificités de ce type de logiciels. En effet, il est, par exemple, tout particulièrement important d'examiner précisément les contraintes et les risques spécifiques à la nature même de ces logiciels. Le domaine de l'Open Source étant très vaste et très riche, il est aussi nécessaire de disposer d'une méthode de qualification permettant de bien différencier les différents logiciels candidats à un besoin, souvent très nombreux, tant sur les aspects techniques et fonctionnels que stratégiques.

En plus des questions « naturelles » comme :

- Quel logiciel répond le mieux à mes besoins **techniques** actuels et prévus ?
- Quel logiciel répond le mieux à mes besoins **fonctionnels** actuels et prévus ?

Voici quelques questions que devrait se poser toute entreprise avant de prendre une décision :

- Quelle est la pérennité du logiciel ? Quels sont les risques de Forks ? Comment les anticiper et les gérer ?
- Quel est le niveau de stabilité auquel s'attendre ? Comment gérer les dysfonctionnements ?
- Quel est le niveau de support requis et disponible sur le logiciel ?
- Est-il possible d'influer sur le logiciel (ajout de nouvelles fonctionnalités ou de fonctionnalités spécifiques) ?

Pour pouvoir répondre sereinement à ce type d'interrogations et ainsi faire un choix éclairé en maîtrisant les risques, il est impératif de disposer d'une méthode offrant la possibilité :

- de qualifier un logiciel en intégrant les spécificités de l'Open Source ;
- de comparer plusieurs logiciels en fonction de besoins formalisés et de critères pondérés pour être à même d'effectuer un choix final.

Ce sont ces différents points qui ont poussé Atos Origin à concevoir et formaliser la méthode de Qualification et de Sélection de logiciels Open Source (QSOS).

2.2 De la nécessité d'une méthode libre

Selon nous, une telle méthode ainsi que les résultats qu'elle génère, se doivent d'être mis à disposition de tous selon une licence libre. En effet, seule une telle licence est à même de garantir la promotion du mouvement open source, via notamment :

- la possibilité de réutilisation par tous des travaux de qualification et d'évaluation réalisés ;
- la qualité et l'objectivité des documents générés, toujours perfectibles selon les principes de transparence et de revue par les pairs.

A ce titre, Atos Origin, a décidé de placer la méthode QSOS et les documents générés lors de son application (grilles fonctionnelles, fiches d'identité et fiches d'évaluation) sous la licence libre GNU Free Documentation License.

3 Historique des modifications

version	date	auteurs	commentaire
1.0		Raphaël SEMETEYS (Atos Origin)	Conception et rédaction initiales.
1.1		Olivier PILOT (Atos Origin)	Conception et relecture.
1.2		Laurent BAUDRILLARD (Atos Origin)	Conception et relecture.
1.3	17/11/2004	Raphaël SEMETEYS (Atos Origin)	Première version officielle.
1.4	23/11/2005	Raphaël SEMETEYS (Atos Origin)	Corrections typographiques, ajout de la note de licence et de l'historique. Suppression de l'exemple d'évaluation au profit d'un lien vers `http://www.qsos.org`
		Olivier PILOT (Atos Origin)	Nouveau logo.
1.5	19/01/2006	Gonéri LE BOUDER (Atos Origin)	Passage à LaTeX Changement de licence vers la Gnu FDL
		Raphaël SEMETEYS (Atos Origin)	Ajout du Manifeste QSOS Renommage des axes "Risques du point de vue client" en "Risques de l'Utilisateur" et "Risques du point de vue Atos Origin" en "Risques du Prestataire de services"
1.6	13/04/2006	Gonéri LE BOUDER (Atos Origin)	Ajout du critère "Packaging", suppression du critère "Intégration" et du sous-critère "Exploitabilité/Installation, déploiement"

4 Introduction

4.1 Objet du document

Ce document présente la méthode, baptisée « QSOS » (Qualification et Sélection de logiciels Open Source), conçue par Atos Origin pour qualifier et sélectionner les logiciels Open Source dans le cadre de ses travaux de support et de veille technologique.

La méthode peut s'intégrer dans le cadre plus général d'un processus de veille technologique qui n'est pas présenté ici, et décrit un processus de constitution des fiches d'identité et d'évaluation de logiciels libres.

4.2 Public visé

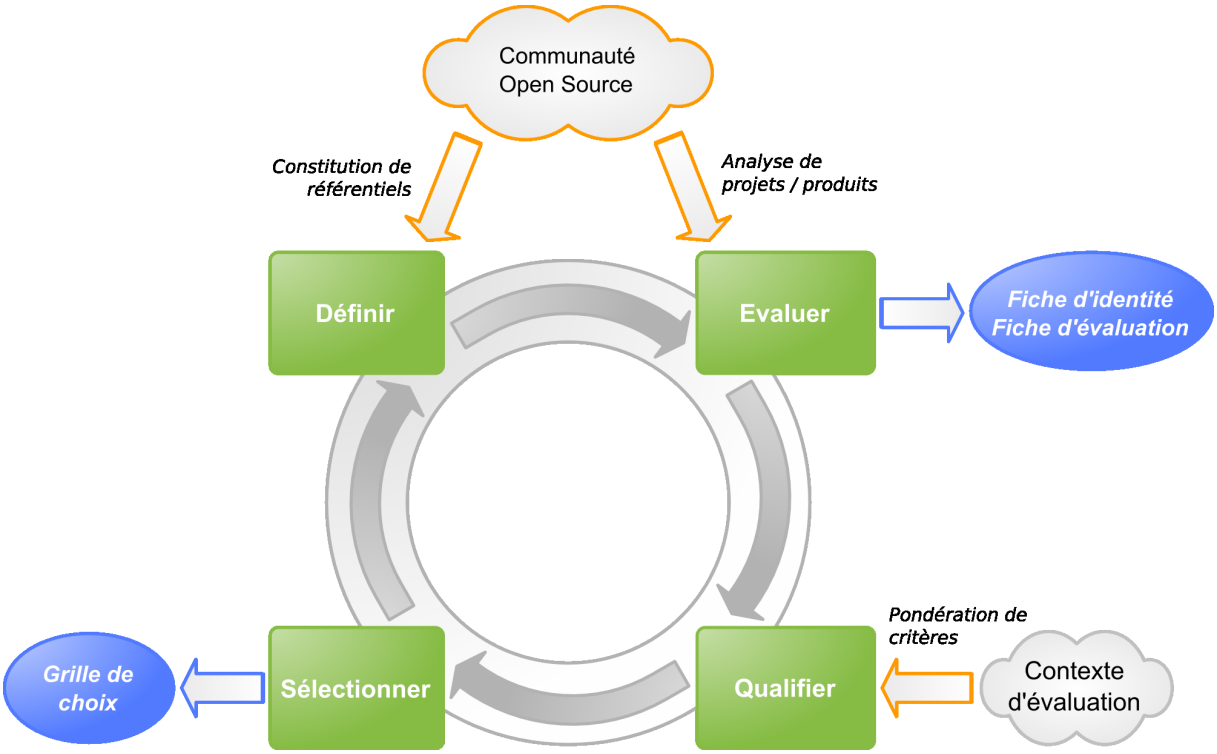
Le présent document vise les publics suivants :

- les personnes curieuses de se documenter sur la méthode à titre professionnel comme personnel ;
- les communautés des projets Open Source ;
- les experts du secteur informatique désirant connaître et appliquer la méthode dans leur travail quotidien d'évaluation et de sélection de composants dans l'optique de bâtir des solutions logicielles répondant à leurs besoins ou à ceux de leurs clients.

5 Processus général

5.1 Quatre étapes

Le processus général de QSOS se décompose en plusieurs étapes interdépendantes (figure 1).



1: Les 4 étapes du processus QSOS

Chacune de ces étapes est détaillée plus loin dans ce document.

5.2 Processus itératif

Le processus général présenté peut être appliqué avec des granularités différentes. Cela permet de s'adapter au niveau de détail souhaité dans le processus de qualification et de sélection ainsi que de procéder par boucles itératives pour affiner chacune des quatre étapes.

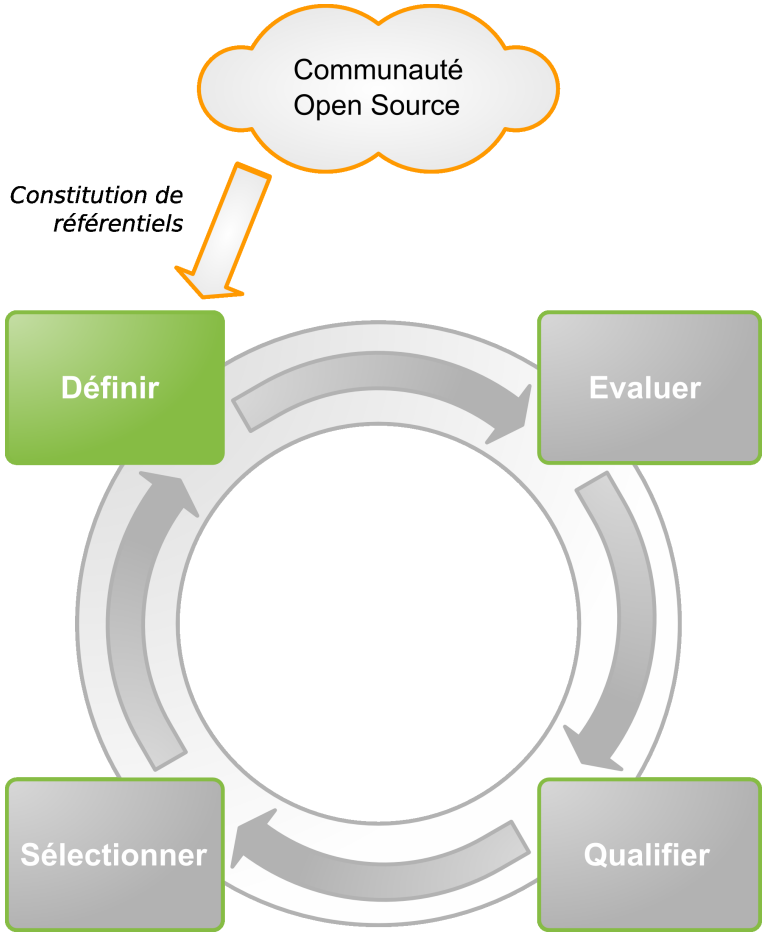
5.3 Outillage

Un outil unique est en cours de finalisation par Atos Origin pour permettre d'implémenter de manière cohérente l'application de la méthode présentée ici.

Cet outil, baptisé O3S (Open Source Selection Software), sera mis à disposition de la communauté via le site `http://www.qsos.org` pour coordonner la saisie, la modification et l'utilisation des évaluations réalisées avec la méthode QSOS.

étape	description
Définir	Constitution et enrichissement des référentiels utilisés par les autres étapes
Évaluer	Évaluation d'une version de logiciel par rapport aux trois axes de critères suivants : couverture fonctionnelle, risques côté utilisateur, risques côté prestataire de services (ceci indépendamment de tout contexte utilisateur ou client particulier).
Qualifier	Pondération des critères constituant les trois axes en fonction du contexte (besoins de l'utilisateur et/ou stratégie retenue par le prestataire de services).
Sélectionner	Utilisation du filtre constitué lors de l'étape de qualification pour procéder à des recherches, comparaisons et sélections de produits, basées sur les données des deux premières étapes.

6 Définir



2: Définir

6.1 Objectif

L'objectif de cette étape est de définir différents éléments de typologie réutilisés par les trois étapes suivantes du processus général.

Les différents référentiels typologiques concernés sont les suivants :

Types de logiciels : classification hiérarchique de types de logiciels et description des couvertures fonctionnelles associées à chaque type.

Types de licences : classification des types de licences libres et Open Source utilisées.

Types de communautés : classification des types d'organisations communautaires existant autour d'un logiciel Open Source pour en assurer le cycle de vie.

6.2 Référentiel des types de logiciels

Il s'agit du référentiel qui évolue le plus car, au fur et à mesure que les logiciels évoluent, ils offrent de nouvelles fonctionnalités qu'il est nécessaire d'y ajouter.

Propriétarisation : le code dérivé peut-il être rendu propriétaire ou doit-il rester libre ?

Viralité : l'utilisation du code du logiciel à partir d'un autre module se traduit-il ou non par la nécessité que ce module soit placé sous la même licence ?

Héritage : le code dérivé hérite-il obligatoirement de la licence où est-il possible d'y appliquer des restrictions supplémentaires ?

Le tableau 3 liste les licences les plus souvent utilisées en les comparant par rapport aux critères énoncés plus haut.

Licence	Propriétarisation	Viralité	Héritage
GPL	Non	Oui	Oui
CeCILL	Non	Oui	Oui
LGPL	Non	Partielle	Oui
BSD	Oui	Non	Non
Artistic	Oui	Non	Non
MIT	Oui	Non	Non
Apache v1.1	Oui	Non	Non
Apache v2.0	Oui	Non	Non
MPL v1.1	Non	Non	Oui
Common Public License v1.1	Non	Non	Non
Academic Free License v2.1	Oui	Non	Non
PHP License v3.0	Oui	Non	Non
Open Software License v2.0	Non	Non	Non
Zope Public License v2.0	Oui	Non	Non
Python SF License v2.0	Oui	Non	Non

3: Liste non exhaustive de licence

Il convient de noter qu'un même logiciel peut être assujetti à plusieurs licences différentes (y compris propriétaires).

6.3 Référentiel des types de communautés

Les types de communautés identifiés à ce jour sont :

Développeur isolé : le logiciel est développé et géré par une seule personne.

Groupe de développeurs : plusieurs personnes travaillant ensemble de manière informelle ou pas fortement industrialisées.

Organisation de développeurs : il s'agit d'un groupe de développeurs utilisant un mode de gestion du cycle de vie du logiciel formalisé et respecté, généralement basé sur l'attribution de rôles (développeur, validateur, responsable de livraison, ...) et la méritocratie.

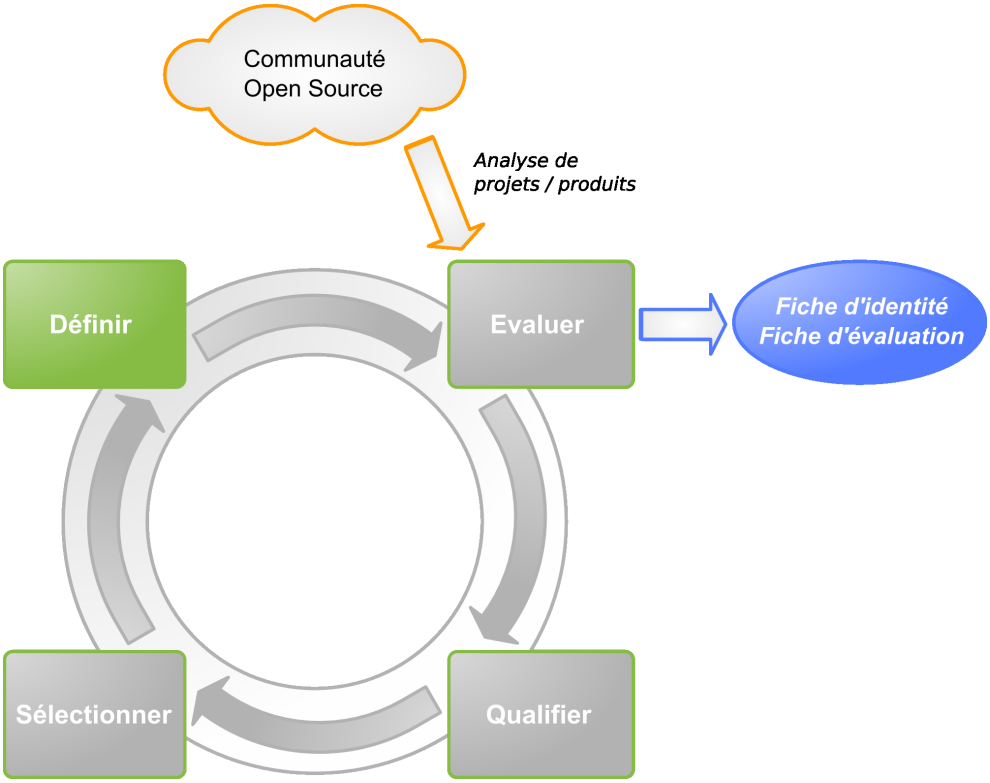
Entité légale : une entité légale, en général à but non lucratif, chapeaute la communauté pour généralement prendre en charge la détention des droits d'auteur ou gérer le sponsorat et les subventions associées.

Entité commerciale : une entité commerciale emploie les développeurs principaux du projet et se rémunère sur la vente de services ou de versions commerciales du logiciel.

6.4 Outil O3S

L'outil O3S est conçu de manière à pouvoir facilement saisir et modifier ces différents référentiels et à mesurer les impacts générés par les modifications sur les données déjà saisies lors des autres étapes du processus général.

7 Evaluer



4: Évaluer

7.1 Objectif

L'objectif de cette étape est de procéder à l'évaluation des logiciels Open Source. Elle consiste à récupérer des informations depuis la communauté Open Source, de manière à :

- Constituer les fiches d'identité d'un logiciel
- Constituer sa fiche d'évaluation : noter le logiciel selon des critères répartis sur trois axes majeurs :
  - Couverture fonctionnelle
  - Risques du point de vue de l'Utilisateur
  - Risques du point de vue du Prestataire de services

Ce travail d'évaluation est insérable dans une démarche plus large de veille technologique qui n'est pas décrite ici dans sa globalité.

7.2 Fiche d'identité de logiciel

Les données constituant la fiche d'identité du logiciel sont des données brutes et factuelles qui ne sont pas directement notées mais sur lesquelles se base en partie la notation des critères présentés plus loin.

Les principaux points de la fiche d'identité de logiciel sont les suivants :

7.2.1 Informations générales

- Nom du logiciel
- Référence, dates de création et de mise jour de la fiche
- Auteur de la fiche
- Domaine fonctionnel du logiciel
- Description succincte du logiciel
- Licences auxquelles est soumis le logiciel
- URL du site principal du projet Open Source et de démonstration du logiciel
- Systèmes d'exploitation compatibles
- Origine du fork (si le logiciel provient d'un fork)

7.2.2 Services existants

- Documentation
- Nombre d'offres de support contractuel
- Nombre d'offres de prestations de formation
- Nombre d'offres de prestations de conseil

7.2.3 Aspects fonctionnels et techniques

- Technologie(s) d'implémentation
- Pré requis techniques
- Fonctionnalités détaillées
- Plan de développement ou « roadmap»

7.2.4 Synthèse

- Tendance générale
- Commentaire synthétique

7.3 Fiche d'évaluation de logiciel

Chaque version d'un logiciel est décrite dans une fiche d'évaluation. Cette fiche comporte des informations plus détaillées que la fiche d'identité car elle s'attache à identifier, décrire et analyser en détail chaque évolution apportée par la nouvelle version.

7.3.1 Notation

Les critères sont notés de 0 à 2. Ces notes serviront, lors de l'étape de sélection, à comparer et filtrer les logiciels en fonction des pondérations précisées lors de l'étape de qualification des besoins de l'utilisateur.

Les paragraphes suivants décrivent les critères utilisés pour chaque axe d'évaluation. Il est à noter que le même critère ou des critères similaires peuvent apparaître sur des axes différents.

7.3.2 Couverture fonctionnelle

La grille de couverture fonctionnelle à utiliser est déterminée par le type de logiciel évalué et est issue du référentiel de l'étape « Décrire ». Consultez le site `http://www.qsos.org` pour le détail des grilles fonctionnelles par type de logiciel.

Pour chaque élément de la grille, la règle de notation est la suivante :

fonctionnalité	**note**
Non couverte	0
Partiellement couverte	1
Totalement couverte	2

Dans certains cas il peut être nécessaire d'utiliser plusieurs grilles de couverture fonctionnelle pour un même logiciel, lorsque ce dernier se classe dans plusieurs types du référentiel. Dans ce cas les critères de couverture sont répartis sur des axes différents de manière à pouvoir évaluer de manière distincte les niveaux de couverture par rapport à chaque type de logiciel.

7.3.3 Risque de l'Utilisateur

Cet axe d'évaluation comprend des critères permettant d'estimer les risques encourus par un utilisateur lors de l'adoption d'un logiciel Open Source. La notation des critères se fait indépendamment de tout contexte utilisateur particulier (cet aspect est traité dans l'étape « Qualifier »).

Les critères sont regroupés en cinq catégories :
- Pérennité intrinsèque
- Solution industrialisée
- Intégration
- Adaptabilité technique
- Stratégie

Les tableaux suivants détaillent chacune de ces catégories, en précisant pour chaque critère la règle de notation.

Pérennité intrinsèque		**Note**		
		0	**1**	**2**
Maturité	**Âge**	Par exemple moins de 3 mois	Par exemple entre 3 mois et 3 ans	Par exemple plus de 3 ans
	Stabilité	Logiciel instable avec de nombreuses releases ou correctifs générant des effets de bords	Existence d'une version stabilisée pour la production mais ancienne. Difficulté à stabiliser la version en cours de développement	Logiciel stabilisé. Les releases intègrent toujours des corrections mais surtout des nouvelles fonctionnalités
	Historique, problèmes connus	Le logiciel connaît plusieurs problèmes qui peuvent être rédhibitoires	Aucun problème majeur connu ou historique de mauvaise gestion des situations de crise	Historique de bonne gestion des situations de crise
	Probabilité de FORK, provenance d'un FORK	Le logiciel a beaucoup de chances d'être forké à l'avenir	Le logiciel provient d'un fork mais a peu de chances d'être forké à l'avenir	Le logiciel a très peu de chance d'être "forké". Il ne provient pas d'un fork non plus.

Pérennité intrinsèque		Note		
		0	1	2
Adoption	**Popularité (à rapporter à la cible visée : grand public, niche, ...)**	Très peu d'utilisateurs	Utilisation décelable sur Internet (sourceforge, freshmeat, google, ...)	Nombreux utilisateurs, nombreuses références
	Références	Aucune	Quelques références sur des applications non critiques surtout	Souvent mis en oeuvre, dans des applications critiques
	Communauté des contributeurs	Pas de communauté ou communauté sans réelle activité (forum, mailing list,...)	Communauté existante avec une activité notable	Forte communauté : grosse activité sur les forums, nombreuses contributions, nombreux articles
	Publications	Aucun livre traitant directement du logiciel n'existe sur le marché	Moins de 5 livres traitant du logiciel sont disponibles	Plus de 5 livres traitant du logiciel sont disponibles dans plusieurs langues

Pérennité intrinsèque		Note		
		0	1	2
Direction des développements	**équipe dirigeante**	1 ou 2 personnes pas clairement identifiées	Entre 2 ou 5, indépendants	Plus de 5
	Mode de direction	Dictature totale	Despote éclairé	Collège d'architectes avec tête identifiée (type ASF ou autre)

Pérennité intrinsèque		Note		
		0	1	2
Activité	**Nombre de développeurs, identification, turnover**	Développeurs < 3, pas clairement identifiés	Développeurs entre 4 et 7 ou plus nombreux mais pas clairement identifiés ou avec un turnover important	Plus de 7 développeurs clairement identifiés, peu de turnover
	Activité autour des bugs	Faible réactivité dans le forum ou sur la mailing list, aucune information sur ce sujet dans les releases notes	Activité décelable mais sans processus clairement expliqué, temps de réaction/résolution longs	Fore réactivité avec identification de l'état et de l'affection des corrections
	Activité autour des fonctionnalités	Peu ou pas de nouvelles fonctionnalités	Évolution du produit sur démarche de l'équipe coeur ou sur demande de la part des utilisateurs sans démarche clairement exposée	Outils de gestion des demandes de nouvelles fonctionnalités, interaction forte avec la roadmap
	Activité sur les releases	Très peu d'activité que cela soit au niveau des releases de production ou de développement	Activité existante aussi bien sur les releases de production que de développement. Mises à jours fréquentes des versions mineures (corrections de bugs)	Activité importante avec des releases mineures fréquentes (corrections de bugs) et des releases majeures planifiées et respectant la roadmap

Pérennité intrinsèque		Note		
		0	1	2
Indépendance des développements	**Indépendance des développements**	Développements à 100% assurés par des personnes payées par une société	60% maximum	20% maximum

Solution industrialisée		Note		
		0	**1**	**2**
Services (quel que soit le fournisseur)	**Formation**	Aucune offre de formation sur le logiciel	Offre existante mais limitée géographiquement, au niveau de la langue ou fournie par un seul prestataire	Offres étendues, offertes par plusieurs prestataires, dans plusieurs langues et décomposées en modules de niveaux graduels
	Support	Aucune offre de support hormis les forums et mailing lists	Offre existante mais limitée à un seul prestataire sans engagement fort sur les délais de résolution d'incident	Offre proposée par plusieurs prestataires avec fort engagement
	Conseil	Aucun service proposé	Service proposé par un seul prestataire, limité au niveau de la langue et de la géographie	Services offerts par différents prestataires dans plusieurs langues

Solution industrialisée		Note		
		0	**1**	**2**
Documentation	**Documentation**	Documentation utilisateur inexistante	Documentation existante mais décalée avec le temps, limitée au niveau de la langue ou peu détaillée	Documentation toujours à jour, traduite et éventuellement adaptée au type d'utilisateur (utilisateur final, administrateur, ...)

Solution industrialisée		Note		
		0	**1**	**2**
Méthode qualité	**Assurance qualité**	Pas de processus d'assurance qualité autour du logiciel	Existence d'une démarche qualité mais peu formalisée et pas outillée	Processus de test automatique intégré au cycle de vie du code avec publication des résultats
	Outillage	Pas d'outil de gestion de bug ou de gestion des demande de nouvelle fonctionnalités.	Outils fournis en standard (exemple : sourceforge) mais peu utilisés.	Outils existants et utilisation très active avec basée sur l'affectation de tâches et le suivi de leur avancement

Solution industrialisée		Note		
		0	**1**	**2**
Packaging	**Source**	Le logiciel requiert beacoup d'efforts pour être installé depuis le code source	L'installation depuis le code source est limitée et fait l'objet de restrictions très strictes (OS, architectures, librairies, ...)	Installation facile depuis le code source
	Debian	Le logiciel n'est pas packagé pour Debian	Il existe un paquet Debian mais avec des problèmes importants ou non officiellement supporté	Le logiciel est packagé dans la distribution
	FreeBSD	Le logiciel n'est pas packagé pour FreeBSD	Il existe un portage mais avec des problèmes importants ou non officiellement supporté	Un portage officiel existe dans FreeBSD
	HP-UX	Le logiciel n'est pas packagé pour HP-UX	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté	Un paquet testé est fourni pour HP-UX
	MacOSX	Le logiciel n'est pas packagé pour MacOSX	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté	Le logiciel est packagé dans la distribution
	Mandriva	Le logiciel n'est pas packagé pour Mandriva	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté	Le logiciel est packagé dans la distribution

Solution industrialisée		Note		
		0	**1**	**2**
Packaging	**NetBSD**	Le logiciel n'est pas packagé pour NetBSD	Il existe un portage mais avec des problèmes importants ou non officiellement supporté	Un portage officiel existe dans NetBSD
	OpenBSD	Le logiciel n'est pas packagé pour OpenBSD	Il existe un portage mais avec des problèmes importants ou non officiellement supporté	Un portage officiel existe dans OpenBSD
	RedHat/Fedora	Le logiciel n'est pas packagé pour RedHat/Fedora	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté	Le logiciel est packagé dans la distribution
	Solaris	Le logiciel n'est pas packagé pour Solaris	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté (ex : SunFreeware.com)	Le logiciel est supporté par Sun pour Solaris
	SuSE	Le logiciel n'est pas packagé pour SuSE	Il existe un paquet mais avec des problèmes importants ou non officiellement supporté	Le logiciel est packagé dans la distribution
	Windows	Le logiciel ne s'installe pas sur Windows	Il existe un paquet mais limité, avec des problèmes importants ou ne couvrant que ceryaines versions de Windows (ex : Windows2000 et WindowsXP)	Windows est complètement supporté et un package est disponible

Solution industrialisée		Note		
		0	**1**	**2**
Exploitabilité	**Facilité d'utilisation, ergonomie**	Utilisation difficile nécessitant une connaissance approfondie du fonctionnement du logiciel. Ergonomie austère et très technique	Utilisation facilité par l'existence d'aide à l'utilisateur. Présence d'IHM.	Logiciel très orienté utilisateur : aide (contextuelle), IHM attirante et éventuellement gestion de skins
	Administration / Supervision	Absence de fonctionnalités d'administration. Pas d'aspect supervision.	Des fonctionnalités d'administration sont proposées mais demandent à être améliorées ou complétées.	Intégration possible avec d'autres outils (via SNMP, ...)

Adaptabilité technique		Note		
		0	**1**	**2**
Modularité	**Modularité**	Logiciel monolithique	Existence de modules de haut niveau permettant un premier niveau d'adaptation du logiciel	Conception modulaire, permettant d'adapter le logiciel aux besoins, par choix de certains modules ou développement de nouveaux

Adaptabilité technique		**Note**		
		0	**1**	**2**
Travaux dérivés	**Facilité technique de modification du code existant**	Tout à la main	Conception modulaire, permettant d'adapter le logiciel aux besoins, par choix de certains modules ou développement de nouveaux	Recompilation possible, outillée (ex : make, ANT) et documentée
	Facilité d'extension du code	Mode ajout + recompilation	Zones prévues pour extension mais recompilation	Principe de plug-in, zones d'extension prévues et recompilation non nécessaire

Stratégie		Note		
		0	**1**	**2**
Licence	**Permissivité (NB : à pondérer uniquement si l'utilisateur désire un jour ne pas redistribuer le code)**	Licence très stricte type GPL	Licence moyennement permissive située entre les deux extrêmes (GPL et BSD), existence de modes de licences multiples en fonction du type d'utilisateurs (particulier, entreprise, ...) ou de son activité	Licence très permissive type licence BSD ou Apache licence
	Protection contre des forks commerciaux	Licence très permissive type licence BSD ou Apache licence	Licence moyennement permissive située entre les deux extrêmes (GPL et BSD), existence de modes de licences multiples en fonction du type d'utilisateurs (particulier, entreprise, ...) ou de son activité	Licence très stricte type GPL

Stratégie		Note		
		0	**1**	**2**
Détenteur des droits	**Détenteur des droits**	Les droits sont détenus par des personnes individuelles et peu nombreuses, cela facilite le changement du mode de licence	Les droits sont détenus par de nombreuses personnes et couvrent de manière homogène le code, rendant plus difficile le changement de mode de licence	Les droits sont détenus par une entité légale en qui la communauté a confiance (type FSF ou ASF)

Stratégie		Note		
		0	1	2
Modification du code	**Modification du code**	Pas de possibilité pratique de proposer une modification de code	Modifications du code outillée (type CVS, SVN) mais pas véritablement utilisée pour faire évoluer le logiciel	Le processus de modification du code est clairement défini et respecté. Basé sur l'attribution de rôles.

Stratégie		Note		
		0	1	2
Roadmap	**Roadmap**	Aucune roadmap publiée.	Existence d'une roadmap sans planning prévisionnel ou à très court terme	Roadmap versionnée, intégrant un planning prévisionnel et mesure des écarts

Stratégie		Note		
		0	1	2
Sponsor	**Sponsor**	Le projet ne dispose d'aucun sponsor, l'équipe coeur n'est pas payée	Le logiciel possède un sponsor unique et identifié risquant de faire le jeu de sa stratégie	Le logiciel est sponsorisé par l'industrie

Stratégie		Note		
		0	1	2
Indépendance stratégique	**Indépendance stratégique**	Pas de vision stratégique décelable ou forte dépendance vis à vis d'un acteur ou sponsor unique	Vision stratégique partagée avec plusieurs autres projets Open Source mais sans engagement fort de la part des détenteurs des droits	Forte indépendance de l'équipe coeur ou de l'entité légale détentrice des droits. Forte présence au niveau des instances de normalisation

7.3.4 Risque du Prestataire de services

Cet axe d'évaluation comprend des critères permettant d'estimer les risques encourus par un prestataire en matière de services autour d'un logiciel Open Source (expertise, intégration, développement, support, ...). C'est notamment sur cette base que le niveau d'engagement du prestataire vis à vis de ses clients peut être déterminé.

Stratégie		**Note**		
		0	**1**	**2**
Maintenabilité	**Qualité code**	Code peu lisible ou de mauvaise qualité. Pas de cohérence au niveau des styles de codage.	Code lisible mais pas commenté de manière détaillée.	Code très lisible, commenté et mettant en oeuvre des design patterns classiques. Politique de coding cohérente et respectée.
	Dispersion technologique	Utilisations de nombreux langages différents	Utilisation d'un langage principale avec certains modules codés dans d'autres langages pour des besoins précis et limités	Langage unique
	Complexité intrinsèque	Forte complexité du code nécessitant un haut niveau de maîtrise pour pouvoir effectuer des modifications sans générer d'effet de bord	Code peu complexe mais nécessitant une bonne maîtrise du langage et des principes d'algorithmique	Code simple et facile à modifier
	Documentation technique	Documentation inexistante (guide de développement, architecture, ...)	Documentation existante mais pas complète ou pas à jour et n'intégrant pas de vision architecturale	Documentation détaillée et à jour, intégrant les aspects conception/architecture et coding style

Stratégie		Note		
		0	**1**	**2**
Maîtrise du code	**Directe**	Aucune maîtrise directe du code en interne	Maîtrise directe du code mais limitée soit au niveau d'une seule personne soit au niveau de la couverture du code	Existence de plusieurs ressources internes possédant une maîtrise du code
	Indirecte	Aucune maîtrise indirecte du code	Maîtrise indirecte via un partenariat	Partenariat fort avec le détenteur des droits et/ou l'équipe coeur

7.3.5 Granularité de la notation

Comme indiqué plus haut, il est possible d'appliquer le processus global de manière itérative. Au niveau de l'évaluation, cela se traduit par la possibilité de noter les critères en trois fois, en calquant le niveau de détail sur celui de l'évaluation réalisée. Ainsi il est possible d'évaluer :

- Au niveau des cinq catégories principales
- Au niveau des sous catégories de chaque catégorie
- Au niveau de chaque critère

Cela permet ainsi de ne pas bloquer le déroulement du processus général lorsque l'on ne dispose pas de l'intégralité des notes.

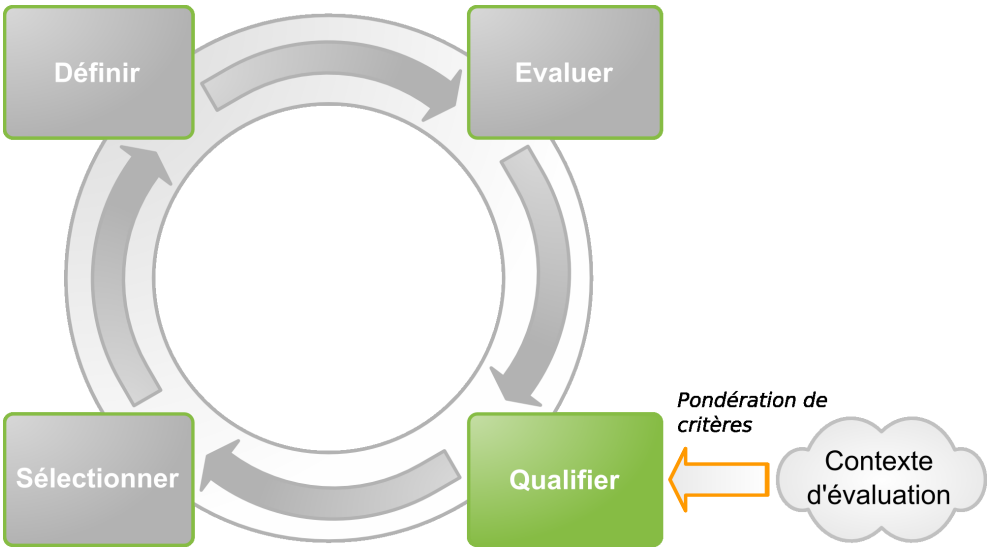
Une fois tous les critères évalués, les notes des deux premiers niveaux sont alors calculées par moyenne pondérée des notes attribuées ou calculées aux niveaux précédents.

7.4 Outil O3S

L'outil O3S permet de saisir les données brutes et d'évaluer le logiciel selon les trois axes. Il offre également la possibilité de générer les fiches d'identités des logiciels évalués.

La granularité de l'évaluation est gérée comme suit : tant que tous les critères composant une sous catégorie ne sont pas notés, sa note est laissée à l'appréciation de l'utilisateur. Dès que l'intégralité des critères est évaluée, la note de la sous catégorie est déduite automatiquement de celles des critères. Il en va de même pour les catégories et les sous catégories.

8 Qualifier



5: qualifier

8.1 Objectif

L'objectif de cette étape est de définir un ensemble d'éléments traduisant les besoins et les contraintes liés à la démarche de sélection d'un logiciel Open Source. Il s'agit ici de qualifier le contexte dans lequel il est envisagé d'utiliser le logiciel libre, de manière à obtenir un filtre utilisé par la suite dans l'étape « Sélectionner » du processus général.

8.2 Filtre sur la fiche d'identité

Un premier niveau de filtrage peut être posé au niveau des données constituant la fiche d'identité des logiciels.

Il peut s'agir, par exemple, de ne considérer que les logiciels d'un type donné du référentiel, ou fonctionnant sur un système d'exploitation donné.

En général, bien que cela ne soit pas obligatoire, ce filtre ne comporte pas d'information de pondération ; il sert principalement à éliminer les logiciels inadéquats dans le contexte de l'utilisateur.

8.3 Filtre sur la couverture fonctionnelle

Chaque fonctionnalité de l'axe fonctionnel est affectée d'un niveau d'exigence, choisi parmi les suivants :

- fonctionnalité requise
- fonctionnalité optionnelle
- fonctionnalité non requise

Ces exigences seront associées à des valeurs de pondération lors de l'étape « Sélectionner », en fonction du mode de sélection retenu.

8.4 Filtre sur les risques Utilisateur

Le degré de pertinence de chaque critère de l'axe des risques Utilisateur est positionné en fonction du contexte, comme indiqué dans la figure 8.4. Cette pertinence sera traduite par une

Pertinence du critère
Critère non pertinent, à ne pas intégrer au filtre
Critère pertinent
Critère critique

valeur numérique de pondération à l'étape suivante du processus en fonction du mode de sélection utilisé.

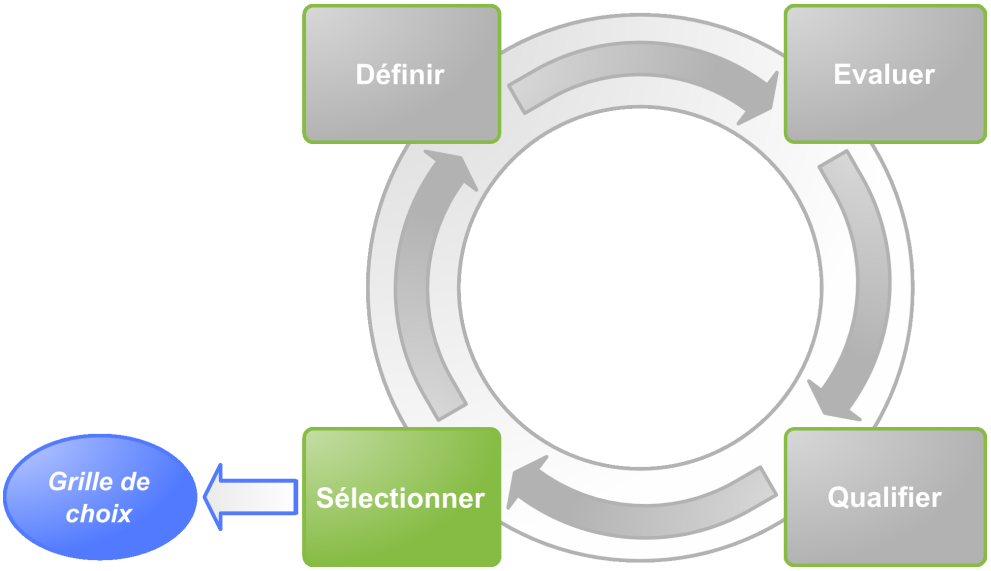
8.5 Filtre sur les risques Prestataire de service

Ce filtre est utilisé par un prestataire de services pour évaluer les logiciels et les prestations à intégrer dans son offre et déterminer les niveaux d'engagement associés.

8.6 Outil O3S

L'outil O3S permet de définir ces différents filtres, tout en étant guidé à la saisie.

9 Sélectionner



6: Sélectionner

9.1 Objectif

L'objectif de cette étape est de sélectionner le ou les logiciels correspondant aux besoins de l'utilisateur, ou plus généralement de comparer des logiciels du même type.

9.2 Sélection

Deux modes sont possibles :
- la sélection stricte
- la sélection souple

9.2.1 Sélection stricte

La sélection stricte se base sur un processus d'élimination directe dès qu'un logiciel ne répond pas aux exigences formulées dans l'étape :
- Élimination des logiciels ne correspondant pas au filtre sur la fiche d'identité
- Élimination des logiciels n'offrant pas les fonctionnalité requises par le filtre sur la couverture fonctionnelle
- Élimination des logiciels dont les critères de risques Utilisateur ne satisfont pas aux pertinences définies par ou avec l'utilisateur :
  - La note d'un critère pertinent doit être au moins égale à 1
  - La note d'un critère critique doit être au moins égale à 2

Cette méthode est très sélective et peut, en fonction du niveau d'exigence de l'utilisateur, ne retourner aucun logiciel éligible.

Les logiciels ayant passé la sélection sont ensuite affectés d'une note globale déterminée par pondération, de la même manière que dans la sélection souple.

9.2.2 Sélection souple

Cette méthode est moins stricte que la précédente car plutôt que d'éliminer les logiciels non éligibles au niveau de la couverture fonctionnelle ou des risques Utilisateur, elle se contente de les classer tout en mesurant l'écart constaté par rapport aux filtres définis précédemment.

Elle se base sur des valeurs de pondération dont les règles d'attribution sont détaillées dans les paragraphes suivants.

Pondération fonctionnelle La valeur de pondération se base sur le niveau d'exigence de chaque fonctionnalité de l'axe de la couverture fonctionnelle.

Niveau d'exigence	Pondération
Fonctionnalité requise	+3
Fonctionnalité optionnelle	+1
Fonctionnalité non requise	0

Pondération des risques Utilisateur La valeur de pondération se base sur le degré de pertinence de chaque critère de l'axe des risques Utilisateur.

Pertinence du critère	Pondération
Critère non pertinent	0
Critère pertinent	+1 ou -1
Critère critique	+3 ou -3

Le signe de la pondération est fonction de l'impact positif ou négatif du respect du critère sur la capacité au logiciel à répondre aux besoins.

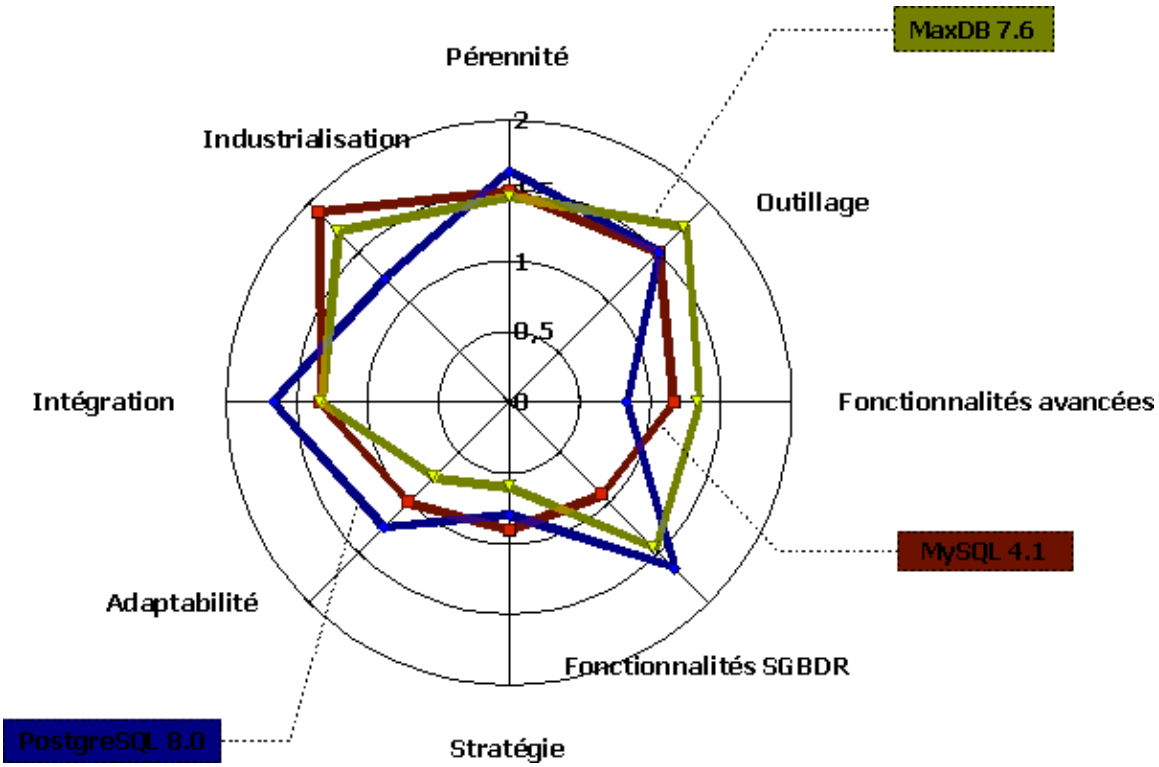
9.3 Comparaison

Les logiciels d'un même domaine peuvent également être comparés entre eux selon les notes pondérées obtenues lors des étapes précédentes.

La figure 7 illustre ce qu'il est alors possible d'obtenir en synthèse.

9.4 Outil O3S

Outre les modes de sélection stricte et de sélection souple, l'outil O3S permet également les opérations de visualisation simple ou filtrée des informations relatives à un logiciel donné (fiche d'identité et critères d'évaluation), ainsi que de comparaison intégrale, filtrée ou différentielle de plusieurs logiciels de même type.



7: Comparaison de fiches. Cette figure est fournie à titre d'exemple, ainsi les pondérations des sous-critères utilisées pour calculer les notes sur les différents axes représentés, ne sont pas représentatives de tous les contextes d'utilisation d'un SGBDR.

10 Site Internet

Le site `http://www.qsos.org` centralise la mise à disposition de documents et d'informations relatifs à la méthode QSOS ainsi que la création, la modification et la certification des grilles fonctionnelles, des fiches d'identité et des fiches d'évaluation.

Glossaire

Fork Un fork est un événement qui apparaît parfois dans le développement d'un projet informatique, typiquement dans des projets communautaires (cas de beaucoup de logiciels libres), quand les opinions au sein de l'équipe de développement divergent sur le chemin à prendre et qu'elles ne sont pas conciliables. Le développement du logiciel part alors dans deux directions différentes, sous l'impulsion des deux camps.

Méthode QSOS Méthode de Qualification et de Sélection de logiciels Open Source, conçue et utilisée par Atos Origin pour ses travaux de support et de veille technologique. Elle est mise à disposition - sous licence libre - sur le site `http://www.qsos.org`.

O3S Open Source Selection Software. Outil informatique développé par Atos Origin implémentant la méthode QSOS, qui sera utilisé sur le site `http://www.qsos.org` pour créer, modifier et visualiser les fiches d'identité et d'évaluation.

Prestataire de services Toute entreprise désireuse d'offrir des services autour de logiciels libres ou Open Source (expertise, intégration, développement, support, ...).

Utilisateur Toute personne physique, entité, entreprise ou administration utilisatrice ou future utilisatrice de logiciels libres ou Open Source.