ENFOQUTE 2: 102-124

ISSN: 1390-6542

# ANÁLISIS DEL MÉTODO PARA CALIFICACIÓN DE SOFTWARE QSOS PARA LA SELECCIÓN DE SOFTWARE APLICABLE A PROCESOS EDUCATIVOS

*Galo Ramos*[1], *Jaime Páez*

## RESUMEN

En este documento se define y analiza el método QSOS, concebido para calificar, seleccionar y comparar el software de código libre y abierto de forma objetiva, trazable y argumentada, ampliando el trabajo realizado por Atos Origin. Se estudia también las posibilidades de extrapolar sus funcionalidades para analizar y seleccionar software para la aplicación en procesos educativos. El método QSOS está disponible bajo los términos de la GNU Free Documentation License.

## INTRODUCCIÓN

A nivel empresarial, la decisión de optar por una solución de software, representa implicaciones muy extensas, que van desde los aspectos económicos, legales, sociales e incluso de cultura organizacional. La decisión comúnmente se toma conforme al análisis de las necesidades y limitaciones (técnicas, funcionales y estratégicas) que éste software puede representar. Dentro del contexto educativo, a todos los aspectos antes analizados, se suman aspectos relativos a la docencia, como la adaptabilidad al proceso educativo y las funcionalidades investigativas, inherentes a cada pieza de software; estos aspectos extra, deben ser correctamente evaluados al momento de seleccionar herramientas que se integren en los procesos formativos, dadas las necesidades específicas de la actividad académica.

Cuando se planea implementar una solución de software a nivel tanto empresarial como educativo es necesario definir una metodología de calificación y selección objetiva, estructurada y funcional, basada en criterios técnicos, por ejemplo, resulta de vital importancia examinar las limitaciones y riesgos del software; también es necesario contar con un método de calificación para establecer de manera cuantitativa las diferencias existentes entre los distintos candidatos, tanto en aspectos técnicos como funcionales y estratégicos.

[1] Universidad Tecnológica Equinoccial, Facultad de Ciencias de la Ingeniería, Av. Occidental y Mariana de Jesús. Quito, Ecuador. gramos@ute.edu.ec

Para realizar éste tipo de análisis es necesario contemplar criterios relativos a:

- Requerimientos técnicos actuales y planificados.
- Requerimientos funcionales actuales y planificados.
- Sostenibilidad del software.
- Nivel de estabilidad del software.
- Gestión de los fallos de funcionamiento del software.
- Nivel de soporte disponible y previsto para el software
- Influencia en el desarrollo del software.
- Prospecciones funcionales del software evaluado.

Por lo tanto, para poder cuantificar y medir las posibilidades reales de implantación del software es necesario establecer un método de gestión de riesgos y calificación de software, que nos ofrezca la posibilidad de hacer comparaciones de software según los requisitos de las necesidades específicas que se generan dentro del contexto pedagógicos, en función de establecer criterios ponderados, en base a los cuales calificar el software y hacer una selección final de la manera más objetiva y beneficiosa.

## PROCESO GENERAL

El proceso general diseñado en QSOS se compone de cuatro etapas definidos en ciclos iterativos (figura 1).

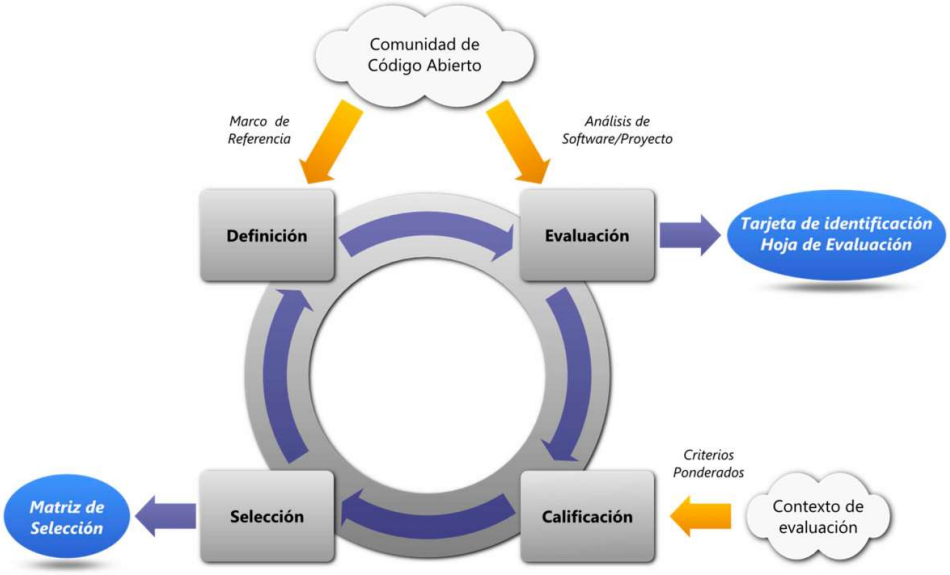


Figura 1. Etapas de QSOS.

El proceso general de QSOS se puede aplicar con diferentes niveles de granularidad, lo que permite establecer el nivel de detalle del proceso en función de los requerimientos individuales de cada organización, así como el avance del proceso en ciclos iterativos para ir perfeccionando las cuatro etapas definidas.

| Etapa | | Descripción |
|---|---|---|
| 1 | Definición | Construcción de los marcos de referencia utilizados en las etapas siguientes. |
| 2 | Evaluación | Evaluación realizada en los tres ejes de criterios: cobertura funcional, riesgos para el usuario y riesgos para el proveedor de servicios. |
| 3 | Calificación | Ponderación de criterios divididos en los tres ejes, modelando el contexto (requerimientos de los usuarios y/o estrategias establecidas por el proveedor de servicios). |
| 4 | Selección | Aplicación del filtro creado en la Etapa 3, "Calificación" en datos proporcionados por las dos primeras etapas, con el fin de continuar las consultas, comparaciones y selecciones de productos. |

Cuadro 1. Definición de etapas de QSOS.

## DEFINICIÓN

En ésta etapa se deben definir los distintos elementos de la tipología reutilizados por las tres etapas restantes del proceso general, los marcos referenciales son:

- **Familias de software:** clasificación jerárquica de los dominios de software y la descripción de cuadros de funcionalidades asociados a cada dominio.
- **Tipos de licencias:** clasificación de licencias de código libre y abierto.
- **Tipos de comunidades:** clasificación de las organizaciones comunitarias existentes en torno al software de código abierto o libre y que son responsables de su ciclo de vida.

### Familias de software

Este marco de referencia se desarrolla más, porque a medida que el software evoluciona ofrece nuevas funcionalidades que hay que añadir al marco de referencia.

## Tipos de licencias

Este marco de referencia lista y clasifica las licencias principales utilizadas para el software libre y de código abierto. Los criterios elegidos para describir dicha licencia son:

- **Propiedad:** Analiza si el código derivado puede convertirse en privativo o debe permanecer libre
- **Viralidad:** Analiza si alguno de los otros módulos vinculados al código fuente es inevitablemente afectado por la licencia
- **Herencia:** Analiza si el código derivado hereda su licencia obligatoriamente o es posible aplicar restricciones adicionales

El cuadro 2 muestra una comparación de las licencias más comunes que expone los criterios formulados anteriormente.

| Licencia | Propiedad | Viralidad | Herencia |
|---|---|---|---|
| **GPL** | No | Si | Si |
| **LGPL** | No | Parcial | Si |
| **BSD** | Si | No | No |
| **Artistic** | Si | No | No |
| **MIT** | Si | No | No |
| **Apache v1.1** | Si | No | No |
| **Apache v2.0** | Si | No | No |
| **MPL v1.1** | No | No | Si |
| **Common Public License v1.1** | No | No | No |
| **Academic Free License v2.1** | Si | No | No |
| **PHP License v3.0** | Si | No | No |
| **Open Software License v2.0** | No | No | No |
| **Zope Public License v2.0** | Si | No | No |
| **Python SF License v2.0** | Si | No | No |

Cuadro 2. Lista no exhaustiva de licencias de código abierto.

### Tipos de comunidades

Los tipos de comunidades identificadas de software libre se pueden clasificar así:

- **Desarrollador aislado:** el software es desarrollado y gestionado por una sola persona.
- **Grupo de desarrolladores:** varias personas colaboran de una manera informal.
- **Organización de desarrolladores:** un grupo de desarrolladores gestiona el ciclo de vida del software de una manera formal, generalmente basada en la asignación de funciones (desarrollador, tester, administrador de entrega...) y la meritocracia.
- **Entidad legal:** una persona jurídica (generalmente sin fines de lucro) que gestiona la comunidad, en general, posee derechos de autor y también gestiona el patrocinio y subvenciones vinculadas.
- **Entidad comercial:** la organización comercial emplea a los desarrolladores principales del proyecto que son remunerados mediante la venta de servicios o de versiones comerciales del software.

## EVALUACIÓN

En ésta etapa se debe llevar a cabo la evaluación del software. Consiste en la recopilación de información de la comunidad de código abierto, con el fin de:

- Generar la tarjeta de identificación del software
- Generar la hoja de evaluación del software, al anotar criterios divididos en tres ejes principales:
    - Cobertura funcional
    - Riesgos desde la perspectiva del usuario
    - Riesgos desde la perspectiva del proveedor de servicios

### Tarjeta de Identificación

Los datos que constituyen la tarjeta de identificación son directos y objetivos, no se evalúan directamente sin embargo, se utilizan como base para el proceso de registro descrito a continuación.

Las partes principales de una tarjeta de identificación son:

#### Información General

- Nombre del software.
- Referencia, fecha de creación, fecha de publicación de la tarjeta de identificación.
- Autor.

- Tipo de software.
- Breve descripción del software.
- Licencias a las cuales se encuentra sujeto el software.
- URL del proyecto y sitio de demostración.
- Sistemas operativos compatibles.
- Origen de la bifurcación (si el software es una bifurcación).

### Servicios existentes

- Documentación.
- Cantidad de oferentes de soporte por contrato.
- Cantidad de oferentes de servicios de capacitación.
- Cantidad de oferentes de servicios de consultorías.

### Aspectos funcionales y técnicos

- Tecnología(s) de implementación.
- Requisitos técnicos.
- Funcionalidades detalladas.
- Plan de trabajo.

### Síntesis

- Tendencia general.
- Comentarios.

## Hoja de evaluación

Cada versión del software se describe en una hoja de evaluación. Este documento incluye información más detallada que la tarjeta de identificación, ya que se centra en identificar, describir y analizar en detalle cada progreso presentada por la nueva versión.

### Puntuación

Los criterios se puntúan de 0 a 2. Estas puntuaciones se utilizarán en la Etapa 4, "Selección" para comparar y seleccionar el software de acuerdo a las ponderaciones, representando las necesidades del usuario especificadas en la Etapa 3, "Calificación".

En los párrafos siguientes se describen los criterios utilizados para la puntuación de cada eje. Cabe señalar que el mismo criterio o criterios pueden parecer similares en distintos ejes.

## Cobertura funcional

El cuadro de funcionalidades se determina por la familia del software y el producto en el marco de referencia de la Etapa 1, "Definición". Para cada elemento del cuadro, la norma de puntuación es la siguiente:

| Funcionalidad | Puntuación |
|---|---|
| **No Cubierta** | 0 |
| **Parcialmente cubierta** | 1 |
| **Completamente Cubierta** | 2 |

En algunos casos es necesario utilizar varios cuadros de funcionalidades para el mismo software, por ejemplo, cuando pertenece a más de una familia de software. En este caso, los criterios de funcionalidad se distribuyen en ejes separados con el fin de poder evaluar claramente la cobertura funcional para cada familia.

Los cuadros de funcionalidades deberán ser definidos conforme a los requerimientos de la organización y deberán ser aplicados en igualdad de condiciones para todo el software evaluado

## Riesgos desde la perspectiva del usuario

Este eje de evaluación incluye criterios para estimar riesgos incurridos por el usuario cuando adopta software libre o de código abierto. La puntuación de los criterios se realiza independientemente de cualquier contexto de un usuario particular (el contexto es considerado después en la Etapa 3, "Calificación").

Los criterios se dividen en cinco categorías:

- Durabilidad intrínseca
- Solución industrializada
- Integración
- Adaptabilidad técnica
- Estrategia

Los cuadros siguientes detallan cada una de estas categorías, especificando la regla de notación que se utilizará para cada criterio.

| **Durabilidad Intrínseca** | | **Puntuación** | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Madurez** | Edad | Por ejemplo, menos de 3 meses | Por ejemplo, entre 3 meses y 3 años | Por ejemplo, mayor a 3 años |
| | Estabilidad | Software inestable con numerosas versiones o parches que generan efectos secundarios | Existen versiones de producción, estables pero antiguas. Dificultades para estabilizar versiones de desarrollo | Software estabilizado. Las versiones proporcionan correcciones de errores, pero sobre todo nuevas funcionalidades |
| | Historial, problemas conocidos | Se conoce varios problemas de software que pueden ser prohibitivos | No hay problemas conocidos o crisis | Buena gestión histórica de situaciones criticas |
| | Probabilidad de bifurcación, origen de la bifurcación | Es muy probable que el software se bifurque en un futuro | Software proviene de una bifurcación, pero tiene muy pocas posibilidades de ser bifurcado en el futuro | Software cuenta con muy pocas posibilidades de ser bifurcado. Tampoco viene de una bifurcación |

| Durabilidad Intrínseca | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Adopción** | Popularidad (en relación a: público en general, especializados...) | Muy pocos usuarios identificados | Uso detectable mediante Internet (SourceForge, Freshmeat, Google.) | Numerosos usuarios, numerosas referencias |
| | Referencias | Ninguno | Pocas referencias, usos no críticos | A menudo implementado para aplicaciones críticas |
| | Contribuciones de la comunidad | Ninguna comunidad o sin actividad real (foro, lista de correo...) | Comunidad existente con una notable actividad | Fuerte comunidad: gran actividad en los foros, numerosos colaboradores y defensores |
| | Libros | Ningún libro sobre el software | Al menos de cinco libros sobre el software disponibles | Más de 5 libros sobre el software están disponibles en varios idiomas |
| | Manuales y Tutoriales | Muy pocos o ningún manual o tutorial sobre el software, o los tutoriales son poco relevantes y confusos. | Manuales o tutoriales con contenido medianamente relevante y poco confuso | Manuales o tutoriales sobre el tema con contenido relevante y disponibles en varios idiomas |

| Durabilidad Intrínseca | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Dirección de Desarrollos** | Equipo Principal | 1 a 2 individuos involucrados, no claramente identificados | Entre 2 y 5 personas independientes | Más de 5 personas |
| | Estilo de gestión | Dictadura total | Despotismo ilustrado | Consejo de arquitectos con un líder identificado (por ejemplo: ASF...) |

| Durabilidad Intrínseca | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Actividad** | Número de desarrolladores, identificación, volumen de negocios | Menos de 3 desarrolladores, no claramente identificados | Entre 4 y 7 desarrolladores o más, no identificados con volumen de negocios significativo | Más de 7 desarrolladores claramente identificados, equipo muy estable |
| | Actividad en errores | Lenta o ninguna reacción en el foro o en la lista de correo con respecto a las correcciones de errores en las notas de lanzamiento | Actividad detectable, pero sin un proceso claramente expuesto, tiempo de reacción y resolución largo | Reacción fuerte, basado en roles y asignación de tareas |
| | Actividad en funcionalidades | Ninguna o pocas funcionalidades nuevas | Progreso del software impulsado por el equipo central o por solicitud del usuario sin ningún tipo de proceso claramente explicado | Herramienta(s) para gestionar las peticiones de características, una fuerte interacción con el plan de trabajo |
| | Actividades en publicaciones | Muy débil actividad en ambas versiones, producción y desarrollo | Actividad en versiones de producción y desarrollo. Frecuentes versiones menores (corrección de errores) | Importante actividad con frecuentes versiones menores (corrección de errores) y versiones principales planificada en relación con el plan de trabajo |

| Durabilidad Intrínseca | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Independencia de desarrollo** | Independencia de desarrollo | Desarrollos realizados al 100% por los empleados de una sola empresa | Máximo 60% | Máximo 20% |

| Solución industrializada | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Servicios** | Capacitación | Ninguna oferta de capacitación identificada | Existe oferta, pero está limitada geográficamente, en términos de lenguaje o suministrado por un proveedor único | Amplia oferta, brindada por varios proveedores en varios idiomas y se divide en módulos de niveles graduales |
| | Soporte | Ninguna oferta de soporte, excepto a través de foros públicos y listas de correo | Existe oferta, pero se limita a un único proveedor sin un fuerte compromiso con la resolución oportuna de los incidentes | Múltiples proveedores de servicios con un fuerte compro miso (por ejemplo: tiempo de resolución garantizado) |
| | Consultoría | Ninguna oferta de servicios de consultoría | Servicios ofrecido por un proveedor único, limitado a nivel del idioma y la geografía | Servicios de consultoría prestados por contratistas distintos en varios idiomas |

| Solución industrializada | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Documentación** | Documentación | Ninguna documentación de usuario | Existe documentación, pero ha cambiado a través del tiempo, limitada en términos de idioma o le falta detalle | Documentación siempre actualizada y traducida, posiblemente adaptada para distintos lectores objetivo (usuario final, gerente, administrador de sistemas, etc.) |

| Solución industrializada | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Aseguramiento de la Calidad** | Aseguramiento de la Calidad | Ningún proceso de control de calidad | Se identifican procesos de control de calidad, pero no muy formalizados y sin hacer uso de ninguna herramienta | Proceso automático de pruebas incluido en el ciclo de vida del código con la publicación de resultados |
| | Herramientas | Ninguna herramienta de gestión para reportes de errores o solicitud de funciones | Proporcionada herramientas estándar (por ejemplo, un repositorio SVN), pero mal utilizadas | Uso muy activo de herramientas de asignación de funciones/tareas y seguimiento de avances |

| Solución industrializada | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Empaquetado** | Código Fuente | El software no se puede instalar desde el código fuente sin mucho trabajo | Instalación desde la fuente es limitada y depende de condiciones muy estrictas (sistema operativo, archivos, bibliotecas...) | La instalación desde el código fuente es fácil |
| | Debian | El software no está empaquetado para Debian | Existe un paquete Debian, pero tiene problemas importantes o no tiene soporte oficial | El software se empaqueta en la distribución |
| | RedHat/Fedora | El software no está empaquetado para RedHat/Fedora | Existe un paquete, pero tiene problemas importantes o no tiene soporte oficial | El software se empaqueta en la distribución |
| | Otra distribución Linux Requerida | El software no está empaquetado para la distribución requerida | Existe un paquete, pero tiene problemas importantes o no tiene soporte oficial | El software se empaqueta en la distribución |
| | MacOS X | El software no está empaquetado para MacOS X | Existe un paquete, pero tiene problemas importantes o no tiene soporte oficial | Existe un paquete oficial de instalación para MacOS X |
| | Windows | El proyecto no se puede instalar en Windows | Existe un paquete, pero es limitado o tiene problemas importantes o sólo cubre versiones específicas de Windows (por ejemplo: Windows 2000 y Windows XP) | Es totalmente compatible con Windows y se proporciona un paquete de instalación |
| | Otro S.O. | El software no está empaquetado para el S.O. requerido | Existe un paquete, pero tiene problemas importantes o no tiene soporte oficial | Existe un paquete oficial de instalación para el S.O. requerido |

| **Solución industrializada** | | **Puntuación** | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Explotabilidad ergonomía** | Facilidad de uso, | Difícil de usar, requiere un conocimiento en profundidad de la funcionalidad del software, ergonomía austera y muy técnica | Facilidad de uso dada por la asistencia entre usuarios. Presencia de Interfaces Hombre-Máquina. | Software muy orientado al usuario: ayuda contextual, Interfaz de usuario atractiva y, posiblemente, gestión de temas |
| | Administración, Supervisión | Ninguna funcionalidad de administración o de supervisión | Existen funcionalidades pero están incompletas y requieren ser mejoradas | Funcionalidades administrativas y de supervisión, completas y fáciles de usar. Posible integración con herramientas externas (por ejemplo: a través de SNMP, etc.) |

| **Adaptabilidad Técnica** | | **Puntuación** | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Modularidad** | Modularidad | Software Monolítico | Presencia de módulos de alto nivel que permiten un primer nivel de adaptación de software | Concepción modular, lo que permite una fácil adaptación del software mediante la selección de módulos o incluso el desarrollo de módulos nuevos |

| Adaptabilidad Técnica | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Trabajos derivados** | Modificación de código | Todo a mano | Recompilación posible, pero compleja sin ninguna herramienta o documentación | Recompilación con herramientas (por ejemplo: make, ANT...) y documentación proporcionada |
| | Extensión de código | Cualquier modificación requiere la recompilación del código | Arquitectura diseñada para extensión estática pero requiere recompilación | Principio de plugin, arquitectura diseñada para la extensión dinámica sin necesidad de recompilar |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Licencia** | Protección contra bifurcaciones propietarias | Licencia muy permisiva como BSD o Apache License | Licencia moderadamente permisiva, situada entre ambos extremos (GPL y BSD), licencia dual en función del tipo de usuario (persona, empresa.) o de sus actividades | Licencia muy estricta, como la GPL |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Propietarios del copyright** | Propietarios del copyright | Derechos en manos de unos pocos individuos o entidades, por lo que es más fácil cambiar la licencia | Derechos en manos de muchas personas, poseen el código de una manera homogénea, lo que hace muy difícil modificar la licencia | Derechos en manos de una persona jurídica en quien confía la comunidad (por ejemplo, la FSF o ASF) |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Modificación del código fuente** | Modificación del código fuente | Ninguna forma práctica para proponer modificaciones del código | Proporciona herramientas para acceder y modificar el código (como CVS o SVN), pero no se han usado realmente para el desarrollo del software | El proceso de modificación del código está bien definido, expuesto y respetado, mediante la asignación de funciones |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Plan de trabajo** | Plan de trabajo | Plan de trabajo no publicado | Existe un plan de trabajo sin proyección | Plan de trabajo versionado, con proyección y medida del retrasos |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Patrocinador** | Patrocinador | Software no tiene un patrocinador, el equipo principal no es remunerado | El software tiene un patrocinador único el cual determina las estrategias | El software es patrocinado por la industria |

| Estrategia | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Independencia estratégica** | Independencia estratégica | Estrategia no detectable o fuerte dependencia de un actor único (persona, compañía, patrocinador.) | Visión estratégica compartida con otros proyectos de código libre y abierto, pero sin un fuerte compromiso de los propietarios de derechos de autor | Independencia fuerte del equipo principal, la entidad titular de los derechos legales participa activamente en el proceso de normalización |

## Riesgos desde la perspectiva del proveedor de servicios

Este eje incluye los criterios de evaluación para los riesgos incurridos por un proveedor de servicios alrededor del software de código abierto (conocimientos, integración, desarrollo, apoyo.). Es especialmente sobre esta base se puede determinar su nivel de compromiso.

| **Prestación de servicios** | | **Puntuación** | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Mantenibilidad** | Calidad del código fuente | Código no muy legible o de mala calidad, incoherencia en los estilos codificación | Código legible, pero realmente no comentado en detalle | Código legible. Patrones de diseño estándar, implementados y comentados. Políticas de codificación, coherentes y respetadas |
| | Dispersión tecnológica | Uso de múltiples lenguajes distintos | Un lenguaje principal con ciertos módulos codificados en otros lenguajes para requisitos específicos y limitados | Lenguaje Único |
| | Complejidad Intrínseca | Código muy complejo que requiere alto nivel de conocimientos para llevar a cabo modificaciones sin generar efectos secundarios | Código no muy complejo, pero requiere conocimientos en lenguajes de programación y diseño de software | Codificación y diseño simples, fácil de modificar |
| | Documentación técnica | Ninguna documentación (guía de desarrollo o documentación generada automática mente como javadoc) | Documentación incompleta o antigua, sin consideraciones de arquitectura integradas | Documentación detallada y actualizada, incluyendo consideraciones de arquitectura, diseño y codificación |

| Prestación de servicios | | Puntuación | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| **Dominio del código** | Directo | Ninguna experiencia directa con el código fuente | Dominio del código, pero limitado a una sola persona o una sola parte del código fuente | Varios individuos dominan el código y cubren así la totalidad del código fuente |
| | Indirecto | Ninguna experiencia indirecta con el código fuente | Fuerte dominio a través de expertos externos proporcionados por socios | Colaboración con el propietario de los derechos de autor y/o el equipo principal |

### Granularidad del registro

Como se mencionó anteriormente es posible repetir el proceso general de manera iterativa. En la fase de evaluación esto trae como consecuencia la capacidad de evaluar criterios en tres ciclos, cambiando el nivel de granularidad dependiendo de la evaluación realizada, por lo tanto, es posible evaluar:

1. Categorías principales
2. Subcategorías de cada categoría
3. Criterios restantes (es decir, todo lo que no se encuentra incluido en los dos primeros ítems)

Esto permite no bloquear el avance del proceso general cuando no se dispone de todas las calificaciones.

Una vez que todos los criterios se han evaluado, las calificaciones de los dos primeros niveles se calculan utilizando una media ponderada de las calificaciones de los niveles anteriores.

## CALIFICACIÓN

En ésta etapa se busca definir un conjunto de elementos que reflejan las necesidades y limitaciones relacionadas con el proceso de selección de un software de código abierto. Esto es para describir el contexto en el que se propone utilizar software libre, con el fin de obtener un filtro que se utiliza posteriormente en la Etapa 4, "Selección" del proceso general.

### Filtro en la tarjeta de identificación

Un primer nivel de filtrado se puede definir en los datos de tarjeta de identificación del software.

Por ejemplo, se podría considerar el software sólo de una determinada familia o software que es compatible con un sistema operativo determinado.

En general, aunque no es obligatorio, este filtro no incluye ninguna ponderación, sino que se utiliza sobre todo para eliminar el software inadecuado en el contexto específico del usuario, es decir, se utiliza para determinar características altamente limitantes (críticas), las cuales, por el hecho de no estar presentes, anulan totalmente la posibilidad de uso del software dentro del entorno de la organización.

## Filtro de cuadros de funcionalidades

Todas las características del cuadro de funcionalidades se atribuyen un nivel de exigencia seleccionado entre los siguientes:

- Funcionalidad requerida
- Funcionalidad opcional
- Funcionalidad no requerida

Estos niveles de exigencia estarán relacionados a los valores de ponderación en la Etapa 4, "Selección", según el modo elegido de selección.

## Filtro sobre los riesgos del usuario

La relevancia de cada criterio de este eje se coloca de acuerdo al contexto del usuario, como se indica en el cuadro siguiente:

| Relevancia |
| --- |
| Criterio irrelevante, excluidos del filtro |
| Criterio relevante |
| Criterio crítico |

Esta relevancia será convertida en un valor de ponderación numérica en la siguiente etapa, según el modo elegido de selección.

## Filtro sobre los riesgos del proveedor de servicios

Este filtro es utilizado por un proveedor de servicios para evaluar software y servicios que serán integrados en su oferta y para determinar los niveles de compromiso relacionados.

# SELECCIÓN

En ésta etapa se busca seleccionar el software para satisfacer las necesidades del usuario, o más generalmente para comparar software similar.

Es posible realizar la selección mediante dos modos:

- Selección estricta
- Selección flexible

## Selección estricta

La selección estricta se basa en la eliminación directa tan pronto como el software no cumple con los requisitos formulados en la Etapa 3, "Calificación":

- Eliminación de software incompatible con el filtro de la tarjeta de identificación
- Eliminación de software que no proporciona la funcionalidad requerida por el filtro en el cuadro de funcionalidades
- Eliminación de software en que los criterios de riesgo utilizados no se ajustan a las normas establecidas por el usuario
    - La puntuación de un criterio pertinente debe ser al menos igual a 1
    - La puntuación de un criterio fundamental debe ser al menos igual a 2

Este método es muy selectivo y puede, en función de la exigencia del consumidor, no devolver ningún software como resultado.

El software que ha pasado la selección se le asigna una puntuación global determinada mediante la ponderación de la misma manera que en la selección flexible.

## Selección flexible

Este método es menos estricto que el anterior, porque en lugar de eliminar el software no elegible, lo clasifica mientras mide la desviación en los filtros aplicados.

Las reglas de ponderación a utilizar se detallan en los párrafos siguientes.

### Ponderación de las funcionalidades

El valor de ponderación se basa en el nivel de exigencia definida en cada funcionalidad del cuadro de funcionalidades.

| Nivel de exigencia | Ponderación |
|---|---|
| Funcionalidad requerida | +3 |
| Funcionalidad opcional | +1 |
| Funcionalidad no requerida | 0 |

### Ponderación de riesgo en el eje del usuario

El valor de ponderación se basa en la relevancia de cada criterio en el eje de riesgo del usuario.

| Relevancia | Ponderación |
|---|---|
| Criterio irrelevante | 0 |
| Criterio relevante | +1 o -1 |
| Criterio crítico | +3 o -3 |

El signo del valor ponderado representa un impacto positivo o negativo en relación a los requerimientos del usuario.

## Comparación

El software de una misma familia (con un cuadro de funcionalidades común) también puede ser comparado en base a los puntajes ponderados determinados antes.

La figura 2 ilustra el tipo de síntesis disponibles.

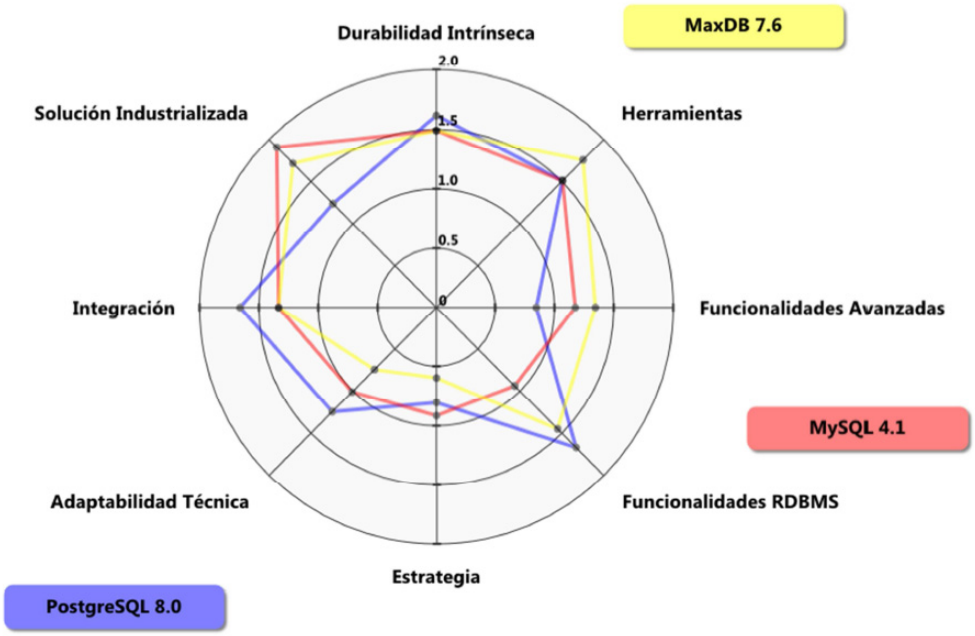


Figura 2. Comparación. Esta figura se presenta como un ejemplo, por lo que las ponderaciones en los distintos ejes no son representativas de todos contextos de uso de un RDMBS.

Así, evaluando dentro de un diagrama radial los resultados obtenidos, en cada uno de los dominios de software relevantes, podemos visualizar las fortalezas y debilidades de cada producto evaluado, y comparar las funcionalidades requeridas que sean más relevantes para organización.

## CONCLUSIÓN

Dentro del ambiente académico el método QSOS debe ser ampliado para poder ser utilizado como parte del proceso de selección de software, dado que existen variables de entorno, referentes al contexto de aplicación profesional de las herramientas que utilicen los docentes, que el método QSOS no analiza; por ejemplo, la penetración del software a nivel empresarial, los formatos, estándares y capacidades tecnológicas propias de cada organización, los cuales deben ser anexados a los cuadros de funcionalidades propuestos de QSOS.

Adicionalmente, como una etapa previa a la aplicación de la metodología QSOS dentro de los ambiente académicos es necesario complementarla añadiéndole una etapa previa al Proceso General, que realice un análisis costo/beneficio, el cual, mida las implicaciones económicas de uso de software libre frente al software privativo, dado que, a diferencia de la creencia general, el uso de software libre o de

fuentes abiertas, también ocasiona gastos a las organizaciones, simplemente que éstos gastos se encuentran focalizados en áreas distintas a las del software privativo.

Finalmente, el método QSOS, debe ser adaptado para cada una de las organizaciones donde se lo pretenda implantar tomando en cuenta las variables económicas, legales, sociales, organizacionales, tecnológicas y estratégicas particulares que puedan intervenir en el proceso. Por ejemplo, dentro de los países que hayan ratificado acuerdos internacionales o tengan legislaciones internas referentes a patentes de software, se deberá revisar el aspecto legal de aplicación de ciertas soluciones de software libre que han recibido cuestionamientos en este sentido.