Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

TRABAJO DE FIN DE GRADO

Marco de caracterización y cálculo de las métricas de calidad de los componentes web para el desarrollo de aplicaciones de usuario final

Autor: David Herranz Fernández (r090008)

Email: davidf1991@msn.com

Tutora: Genoveva López Gómez

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software

Resumen.

En la Web 2.0, donde usuarios finales sin grandes conocimientos de programación pueden interactuar y crear aplicaciones web utilizando componentes publicados en Internet, ofrecidos por una gran variedad de proveedores de servicio. La selección de estos componentes requiere de un análisis exhaustivo por parte de los usuarios sobre las propiedades de estos componentes en referencia a su calidad.

En este proyecto, se presentan dos modelos de calidad según una estructura jerárquica, uno para componentes web como elementos autónomos y otro para componentes utilizados en aplicaciones de *mashup*, basado en un análisis de la emergente Web 2.0. Además, se define una herramienta de medición y anotación de calidad para los distintos niveles de los modelos. Estas herramientas pretenden ser un útil instrumento para los desarrolladores y usuarios de componentes y *mashups*.

Abstract.

In the Web 2.0, where end users without a technical programming background can interact and develop web applications leveraging web components published on the Internet, offered by a great diversity of service providers. This component selection requires an exhaustive analysis by these end users based on the requirements of these components related to their quality.

In this work, two quality models are presented according to a hierarchical structure, one for web components as independent elements and another one for web components as parts of web *mashup* applications, based on an analysis of the emerging Web 2.0. In addition, a measuring and write down quality *framework* is defined to weigh the quality of all the levels of the models. These tools intend to provide a useful instrument to components and *mashup* developers and end users.

Índice.

Capítulo 1.

Introducción.

Índice

Las aplicaciones web, y más concretamente las aplicaciones web 2.0, han experimentado un enorme crecimiento a lo largo de los años debido a la alta participación de usuarios finales en éstas [1][4].

Los usuarios participan aportando feedback a los creadores de estos desarrollos, pero no es ésta su única implicación, ya que pueden a partir de composiciones de fuentes de datos y servicios web ofrecidos por terceros crear páginas web por sí solos sin tener grandes conocimientos de programación, denominadas *mashups*, las cuales han ido ganando una gran popularidad a lo largo de los años.

Los *mashups* web son el resultado de la composición e integración de componentes publicados en la red. Entre dichos componentes pueden distinguirse servicios web, APIs web publicadas por terceros o fuentes de datos entre muchos otros, pudiendo ofrecer una interfaz visual o hacer las funciones de operadores sobre otros componentes que sí dispongan de una interfaz de usuario.

En lo que respecta a la calidad y al éxito que puede alcanzar un mashup web, éste se ve influenciado tanto por los beneficios que éstos aportan los componentes que lo forman por separado como por el valor añadido que ofrece la integración de éstos, ya que la calidad de estas aplicaciones depende inherentemente de la de la funcionalidad de dichos componentes de manera aislada y de la lógica de composición de estos elementos publicados por terceros.

La calidad de cada uno de estos componentes por separado afecta a la composición final de ellos, una calidad global que nunca podrá superar la calidad máxima de los componentes que forman una aplicación de *mashup* por separado.

Deberá tenerse en cuenta que la integración de estos componentes acarreará la aparición de problemas como inconsistencias entre los datos proveídos por componentes al integrarse dichos elementos en esta aplicación web, cuya resolución no será trivial.

La calidad de los componentes web como elementos independientes así como aquella que se deduce de su participación en la creación de mashups deberá ser estudiada y analizada de manera exhaustiva, ya que es un asunto de vital importancia para la creación de componentes y *mashups* de una mayor riqueza. Esta falta de calidad llevará a la creación de componentes web y a la composición de mashups de una calidad muy pobre.

A raíz de ésto, surge la motivación de desarrollar un modelo de medición y caracterización de la calidad de los componentes web como elementos autónomos, además de otro modelo de calidad de estos componentes [1][2] como parte del desarrollo de aplicaciones web de *mashup* creadas por usuarios finales sin conocimientos técnicos de programación para la creación de dichas aplicaciones.

Es en este ámbito donde nace la creación de un marco de caracterización de la calidad de los componentes web que permita la clasificación y asignación de un resumen de calidad a dichos componentes con el objetivo de calificar su potencial utilidad en desarrollos de *mashups* llevados a cabo por usuarios finales.

1.1 Objetivos.

El objetivo principal de este proyecto es el desarrollo de un marco de caracterización de métricas de calidad de componentes utilizados en el desarrollo de aplicaciones web de *mashup* por parte de usuarios finales.

Para la medición de la calidad de los componentes web y su utilidad en los *mashups* de usuario final, se llevará a cabo el desarrollo de dos propuestas de modelos de calidad, una de ellas se referirá a la medición de calidad de los componentes web como elementos autónomos e independientes y otra se centrará en la medición de los *mashup* como aplicaciones web de una única página, las llamadas Single Page Applications.

Cada una de estas propuestas conllevará un análisis exhaustivo de un conjunto de métricas en un alto nivel, además de un estudio de un desarrollo de un marco de métricas de bajo nivel para cada agrupación de métricas de alto nivel de carácter general.

Además de la creación de este marco de medición de calidad, es necesario la creación de un framework de anotado que permita ponderar la calidad de los componentes según su importancia relativa y lleve a cabo la realización de una asignación de un resumen de dicha calidad mediante técnicas de medición objetiva y no intrusiva mediante la utilización de técnicas de “caja negra”.

De esta manera, los objetivos generales de este proyecto son los siguientes:

- Aprendizaje y dominio de conceptos relacionados con los componentes web, el desarrollo de aplicaciones web de usuario final, modelos y métricas de calidad web asociados a componentes y SLAs.

- Diseño de un marco de métricas de calidad completo de alto nivel para la medición de la calidad de componentes web utilizados en el desarrollo de aplicaciones web orientadas a la calidad por parte de usuarios finales no profesionales, además de un modelo de calidad de *mashups* como aplicaciones web de usuario final.
- Definición de un subconjunto de métricas de bajo nivel para cada conjunto de métricas de calidad de carácter general para los dos modelos de calidad mencionados en el punto anterior.
- Creación de un framework de anotado de calidad que, mediante una serie de métodos y herramientas de medición, ponderación y evaluación de métricas, permita medir la calidad de un componente web (y un mashup), asociando dicha medición llevada a cabo por el framework a una especificación informativa de calidad para su consulta por parte de los usuarios finales interesados en su utilización.
- Aplicación del marco de caracterización de métricas definido a un caso de uso real, en el que un desarrollo llevado a cabo por un usuario final sin grandes cualificaciones técnicas de programación se beneficie de este marco de métricas de calidad, pudiendo dicho usuario construir un *mashup* web de mayor riqueza que uno que no se aproveche de este marco de calidad.

1.2 Descripción general del trabajo.

La realización del proyecto se divide en tres grandes secciones, la creación de los modelos de métricas de calidad para componentes web y *mashups*, el desarrollo de un framework de anotado de medición y ponderación de dichas métricas y la creación de un caso de uso real de aplicación de este marco de caracterización de calidad de componentes como elementos aislados y como potenciales participantes en la composición de *mashups*. Estas tres secciones se describen en detalle en los siguientes apartados.

1.2.1. Modelos de calidad para componentes web y mashups de componentes.

La realización de este proyecto tiene como una de las tareas principales el desarrollo de sendos modelos de calidad para componentes web como elementos autónomos e independientes y para componentes web utilizados en el desarrollo de *web mashups* cuyo desarrollo esté destinado a usuarios finales.

Estos modelos de calidad se basarán en las especificaciones definidas en estándares como el ISO 9126-1. Se llevará a cabo la creación de dos modelos de calidad basados en una estructura jerárquica, estructurada en torno a cinco dimensiones o factores de calidad en un alto nivel, las cuales serán desgranadas en varias subcaracterísticas o criterios de calidad de bajo nivel dependientes de aquellas que se encuentran en un nivel más alto de la jerarquía.

Estos criterios de bajo nivel se describirán y agruparán en torno a aquellas que se encuentran en un nivel más alto de la jerarquía. Además, se desarrollarán mecanismos de medición, ponderación y evaluación de estos criterios, tanto en el alto como en el bajo nivel de la jerarquía de los modelos.

1.2.2. Framework de medición de la calidad de los componentes web.

Para la evaluación de la métricas de calidad de los componentes, se desarrollará un framework de medición que, mediante técnicas objetivas y automatizables, permita realizar una ponderación de las métricas de calidad con el fin de asignar un resumen de calidad al componente, en los ámbitos de calidad global, calidad con respecto a los factores de alto nivel y de calidad de los criterios y sus métricas asociadas de bajo nivel.

Una vez asignado este resumen de calidad, el framework deberá asignar al componente una especificación de calidad que detalle a nivel informativo las características y funcionalidad de dicho componente o *mashup* que esté siendo objeto de estudio.

1.2.3. Caso de uso de aplicación del marco de calidad.

Para comprobar la utilidad de la aplicación del marco de caracterización de calidad a un caso de uso real se desarrollará un escenario de ejemplo que nos permitirá observar que, gracias a la utilización de este modelo de calidad para la creación de un *mashup web* por parte de un usuario final, éste podrá aprovecharse de dicho modelo para llevar a cabo la realización de una aplicación web de mayor calidad que ante la ausencia de dicho marco de calidad.

1.3 Estructura del documento.

Una vez se ha descrito de forma general en este primer capítulo de introducción el contenido del proyecto de fin de grado a realizar, se presentan a continuación los diferentes capítulos que detallarán el desarrollo del trabajo.

En el siguiente capítulo se expondrá el estado de la cuestión de los componentes web y mashups, presentándose una serie de conceptos relacionados con éstos. En primer lugar se presentarán las especificaciones de los componentes web y mashup junto a sus tecnologías asociadas.

Seguidamente, se describirá el paradigma de desarrollo de aplicaciones de usuario final orientado a las aplicaciones web, y en concreto los tipos de programación de usuario final que define dicho paradigma. Además de las características relacionadas con la ingeniería del software de usuario final.

A continuación, se detallarán los modelos de calidad software y como se adaptan al ámbito de los componentes de aplicaciones web 2.0. A continuación se realiza un estudio de la información que proveen catálogos colaborativos sobre componentes y mashups web. Por último, se presentan en detalle los acuerdos de nivel de servicio o SLAs y las tecnologías asociadas al mundo de los componentes web y *mashups*.

En los dos siguientes capítulos, se detallan los ambos marcos de caracterización de métricas de calidad para componentes web y para mashups, en los que se presentarán y se justificarán los diseños llevados a cabo de dichos modelos y se explicarán en detalle las métricas de alto nivel elegidas y las correspondientes métricas de bajo nivel asociadas a cada categoría de métricas general.

Seguidamente, en el siguiente capítulo se describirá en detalle la creación de un conjunto de herramientas de evaluación de las métricas de calidad definidas en el capítulo anterior según criterios de medición de su importancia relativa y se presentará un framework de anotado que proveerá de un resumen de calidad a los componentes web y mashups, asociando la calidad de éstos a una especificación de calidad.

Posteriormente, se detallará el desarrollo de un caso de uso real de aplicación del marco de caracterización de calidad de componentes a un escenario de ejemplo con el objetivo de comprobar los beneficios de la utilización de dicho marco para la creación de RIAs (Rich Internet Applications) que dispongan de una mayor calidad que ante la ausencia de estas métricas de calidad.

Por último, al final del documento se encuentran los dos últimos capítulos, los cuales detallan la bibliografía y los acrónimos y el glosario de términos utilizados en el desarrollo del proyecto.

Capítulo 2.

Estado de la cuestión.

Índice

A lo largo de este capítulo se describirá el estado actual de los componentes web y el desarrollo de aplicaciones de *mashups* de usuario final, así como sus tecnologías asociadas e información sobre catálogos web que disponen de todo tipo de información técnica y social sobre éstos.

Al estar enfocado este trabajo en la creación de un marco de métricas de calidad para componentes usados en el desarrollo de aplicaciones web desarrolladas por usuarios finales sin una gran cualificación técnica de programación, se detallarán también los conceptos relacionados con este paradigma de programación.

Además, se presentará la situación actual sobre los modelos de calidad software y las estrategias de diseño que se llevarán a cabo para la realización de los modelos de calidad software para componentes web como elementos independientes y como piezas de composición de los llamados web mashups.

Seguidamente, se explicará en detalle en qué consisten las SLAs (Service Level Agreements) o acuerdos de nivel de servicio y cuál es su utilidad como contratos formales de especificación de calidad entre proveedores y clientes para un determinado servicio.

Por último, se describirán una serie de tecnologías y servicios ofrecidos por desarrolladores o instituciones de prestigio que tienen relación con el desarrollo de componentes web y mashups y el análisis y evaluación de características de éstos para su extrapolación a métricas de calidad.

2.1 Componentes web.

Los componentes web son elementos HTML que cumplen con una serie de nuevos estándares de la W3C, los cuales permiten crear a partir de las tecnologías web convencionales (HTML, CSS y JavaScript) nuevos elementos o etiquetas HTML que podrán ser reutilizadas en todo tipo de aplicaciones web por parte de los usuarios. Esta característica de reutilización aporta a estos nuevos elementos HTML una propiedad de componetización que hace que reciban esta denominación de componentes web.

Las tecnologías relacionadas con los componentes web permiten a los usuarios definir piezas que forman parte de aplicaciones web, con un potencial de interactividad y usabilidad que no es posible alcanzar mediante otras tecnologías disponibles actualmente.

Estas y muchas otras características son ofrecidas por los componentes de manera declarativa, customizando las etiquetas de los componentes directamente desde el *markup* HTML, lo que permite que una gran cantidad de usuarios que no disponen de grandes conocimientos técnicos de programación puedan desarrollar componentes con las mismas facilidades que los grandes desarrolladores de componentes web, e integrar dichos componentes en aplicaciones de mashup creadas por ellos.

Dichos componentes, como se ha mencionado anteriormente, pueden ser reutilizados por los usuarios e integrados con otros en la creación de aplicaciones web de *mashup*. La correcta integración de estos elementos web en la composición de un *mashup* viene influida por la calidad individual de éstos como elementos autónomos y por la calidad en conjunto de varios componentes al unirse.

2.1.1 Especificaciones de la W3C.

Las cinco piezas o estándares de la W3C en los que se apoya el modelo de componentes web [5] se detallan a continuación:

- Templates. Son etiquetas HTML que contienen plantillas de *markup* HTML, las cuales se renderizan en el navegador cuando necesiten ser utilizadas por una página web.
- Custom elements. Esta especificación establece nuevos tipos de etiquetas HTML definidas por los usuarios, por lo que estos nuevos elementos pueden ser reutilizados con sólo utilizar de manera declarativa el elemento en un fichero HTML y pueden configurarse mediante atributos expuestos públicamente por el desarrollador de éste. Además de esto, es posible utilizar estos "custom elements" para extender elementos HTML ya creados, extendiendo en consecuencia su funcionalidad.
- Shadow DOM. Define un nuevo árbol DOM (Document Object Model) propio un componente web, distinto al DOM principal de un página o aplicación web HTML. Este nuevo árbol de nodos tiene un ámbito independiente y local al componente web al que pertenece, pudiendo sobreescribir definiciones y estilos desarrollados en este último sin afectar a aquellos propios del DOM principal.

- Imports. Esta característica permite que componentes web creados sobre este conjunto de especificaciones puedan ser cargados desde ficheros externos en documentos HTML, con el fin de ser reutilizados por multitud de usuarios finales en aplicaciones web.

2.1.2 Librerías para el desarrollo de componentes web.

Existen una serie de librerías que implementan las especificaciones de la W3C con respecto a componentes web, las cuales vienen detalladas a continuación:

- Polymer [6]. Librería de programación para crear componentes web creada por Google, la cual implementa los estándares relacionados con éstos mediante los llamados *polyfills*, pedazos de código fuente que proveen la implementación de dichos estándares para los navegadores que no los soportan de manera nativa. Esta librería se apoya en el nuevo diseño de Google, el "Material Design" [7] y para ello, el grupo de desarrolladores de esta librería ha creado un conjunto de elementos que implementan este diseño, bautizados como "Paper elements".
- AngularJS [8]. *Framework* de desarrollo front-end el cual implementa un modelo MVC (Modelo Vista Controlador) y utiliza directivas para hacer la programación de aplicaciones más sencilla a los programadores, lo cual acerca a los usuarios finales a la creación de aplicaciones. Debido al enorme potencial de este *framework* y sus directivas, es posible crear mediante directivas personalizadas componentes web que pueden ser integrados en aplicaciones web de *mashup*. Un ejemplo de componentes web creados enteramente mediante este *framework* son los aquellos de la suite AngularUI [9].
- X-tags [10]. Librería con una funcionalidad análoga a Polymer escrita en el lenguaje JavaScript y mantenida por los desarrolladores de Mozilla. Ha sido desarrollada utilizando como base un subconjunto de polyfills de Polymer para la creación de componentes web con compatibilidad para los navegadores web modernos.
- Bosonic [11]. Utiliza un compilador fuente a fuente para traducir los componentes web creados en el lenguaje HTML a código fuente en JavaScript y CSS para ofrecer soporte para los navegadores web actuales. Para el desarrollo de los componentes web utiliza, al igual que x-tag, un grupo de *polyfills* de la libería Polymer e incluye una implementación propia de la especificación del estándar W3C de Shadow DOM para componentes web.

2.2 Desarrollos de usuario final (EUD).

El desarrollo de aplicaciones de usuario final o End-user development (EUD) [12] es un paradigma de programación que permite a usuarios y desarrolladores software no profesionales, mediante una serie de herramientas y técnicas crear, modificar o extender la funcionalidad de un programa software.

El número de usuarios de software de ordenadores ha crecido a lo largo de los años, incluyendo personas de muy diversas profesiones como ingenieros, científicos, hombres de negocios, profesores o alumnos entre muchos otros.

Las necesidades de cada una de estas personas con respecto al software son muy complejas y variadas, por lo que necesitan de herramientas con un potencial software suficiente para satisfacer dichas necesidades. Son desarrolladores y usuarios sin una gran cualificación técnica de programación pero con un gran conocimiento del dominio de sus trabajos, por lo que estas herramientas que ofrece el paradigma de desarrollo de usuarios finales les servirán a estos para crear aplicaciones situacionales que satisfagan esas necesidades inmediatas.

EUD no es un paradigma tradicional de desarrollo de software ya que los desarrolladores que lo utilizan no tienen tantos conocimientos técnicos como un programador de software profesional, ni disponen muchos del tiempo y motivación para aprender cómo utilizar estas herramientas, por lo que las técnicas utilizadas en los desarrollos de usuario final deben ser reutilizables y de fácil aprendizaje y uso por parte de dichos usuarios finales.

Existen dos conceptos que se explicarán en detalle a continuación, los cuales son la programación de usuario final (EUP) y la ingeniería de software de usuario final (EUSE), claves en este paradigma de programación de aplicaciones por parte de usuarios finales.

Un ejemplo de desarrollo de usuario final en el ámbito de la Web 2.0 que se apoya en estos dos conceptos es el de la creación de aplicaciones de *mashup* basadas en componentes web, las cuales se explicarán en la siguiente sección. La programación de un *mashup* de calidad se conseguirá mediante la aplicación mediante técnicas de Ingeniería del software de modelos de métricas de calidad que enriquecerá estos desarrollos de una manera significativa, ayudando a la creación de aplicaciones que se beneficien de estas técnicas y adquieran una mayor calidad que ante la ausencia de este modelo de calidad.

2.2.1. Tailoring.

El *tailoring* es una técnica que permite a usuarios modificar una determinada aplicación para hacerla más usable o extender su funcionalidad. Al permitir a los usuarios crear o modificar aplicaciones para dotarlas de una mayor riqueza y funcionalidad, el *tailoring* es considerado como una técnica EUD.

Ejemplos de utilización de esta técnica en programas software son la creación de macros en procesadores de texto para dotar de mayor funcionalidad a dichos programas o la interacción de los usuarios con editores de hojas de cálculo.

2.2.2. Programación de usuario final (EUP).

La programación de usuario final es un tipo de programación en la que el usuario tiene como objetivo satisfacer una determinada necesidad puntual, que conseguirá mediante la creación de ese programa. Dicho usuario, se centra en el desarrollo de un programa para llevar a cabo la consecución de un resultado, teniendo en cuenta en ninguna o un poco medida la calidad del código del programa en cuestión.

El objetivo del programador es el de usar personalmente ese programa, por lo que la programación de usuario final difiere de la programación profesional en la que los desarrolladores de software construyen programas para que otros usuarios los utilicen. EUP dispone de distintos estilos de interacción, explicados a continuación:

- Programación mediante atributos visuales: en este tipo de interacción EUP se utilizan atributos visuales para encapsular la semántica de funcionamiento del programa que se quiera desarrollar. Dichos atributos van desde la posición de un determinado elemento, su color o tamaño hasta atributos relacionados con la interacción entre los elementos del programa.
- Programación por demostración: la programación EUP por demostración es una técnica de programación que consiste en la construcción por parte del usuario de la lógica del software a desarrollar, a partir de la cual un entorno de programación deducirá un programa que utilice dicha lógica para su funcionamiento. Algunos entornos de programación por demostración son capaces de inferir por completo el programa, mientras que otras necesitan de la ayuda del usuario para obtener la información restante para construir el programa. Por este motivo esta técnica EUP es utilizada junto a lenguajes gráficos o textuales.

  La principal problemática de la programación por demostración es la potencial utilidad que pueda tener para el usuario final la creación por parte del entorno del programa inferido, con el fin de que ese usuario pueda revisar y testear dicho programa.
- Programación por especificación: es un estilo de interacción que permite a usuarios finales que deseen desarrollar una aplicación hacer uso de una herramienta que, dada una descripción del funcionamiento de dicha aplicación como entrada, genera un programa para ese usuario a partir de esa descripción. Existe la problemática del desconocimiento acerca del lenguaje de salida en el que se generará el programa, ya que si la descripción se realiza en un lenguaje como el inglés el entorno de programación podría generar una aplicación en lenguaje Java, por lo que el usuario debería conocer tanto el lenguaje de entrada de descripción del programa como el lenguaje de salida.

  Además de esto, para conseguir que el entorno que genera la aplicación sea más susceptible de entender las entradas que les proveen los usuarios, estos entornos suelen obtener su entrada mediante formularios que restrinjan dichas entradas, en lugar de una interfaz textual.
- Programación con texto: es la técnica de programación EUP más extendida debido a que, a pesar de la existencia de otras alternativas de estilos de interacción que restringen los datos de entrada que un entorno de programación recolecta de los usuarios finales, dicha técnica permite crear aplicaciones web a partir de conceptos abstractos que provea el usuario.

2.2.3. Ingeniería del software de usuario final (EUSE).

En el desarrollo de aplicaciones de usuario final, y en concreto en el caso de creación de *mashups* de usuario final, la calidad es un aspecto vital a tener en cuenta. Los usuarios finales, al no disponer de una gran cualificación técnica de programación, pueden implementar aplicaciones a partir de códigos fuente potencialmente dañinos en relación a pérdidas de datos o agujeros de seguridad.

Por este motivo se deben desarrollar técnicas de ingeniería del software en desarrollos de usuarios finales, similares a las técnicas y herramientas utilizadas por los programadores profesionales de aplicaciones relativas a la calidad del ciclo de software en estas aplicaciones. Este ciclo de software incluye las fases de requisitos, diseño, validación, depuración y reutilización, las cuales se detallan a continuación:

- Las fases de requisitos y diseño van de la mano en el caso de los desarrollo de aplicaciones de usuarios finales, ya que en la mayoría de los casos los usuarios no conocen los requisitos previamente al desarrollo de la aplicación y no tienen como objetivo crear un diseño detallado, sino que aspiran a crear una aplicación que satisfaga una necesidad inmediata. Uno de los principales objetivos de la EUSE es que estos requisitos queden perfectamente definidos antes del comienzo de la fase de implementación de las aplicaciones de usuario final.

  Para ello, se han realizado estudios con el objetivo de solucionar o reducir esta problemática. Una aproximación para este problema ha sido la creación de un software que hiciera las veces de crítico de diseño y llevara a cabo la revisión de un diseño llevado a cabo por un usuario final y sugiriera mejoras a implementar en él, mientras que otra solución tenida en cuenta ha sido la consulta por parte de usuarios finales desarrolladores de aplicaciones a otros desarrolladores con más experiencia profesional con el objetivo de conseguir la asistencia de éstos en referencia a temas relacionados, por ejemplo, con técnicas de buenas prácticas.
- Con respecto a la validación y testeo del software, existen dos alternativas principales: el método de testeo "What you see is what you test" y la comprobación de errores en el código mediante aserciones. Con respecto a la primera, se utilizan colores para resaltar partes de la aplicación y atraer a los usuarios a testear los componentes de las aplicaciones que necesitan probarse. Una vez los usuarios acceden a testear un determinado componente, les aparecerán cuadros de ayuda con la información acerca de cómo realizar ese testeo y la recompensa que conseguir si lo hacen. En referencia al segundo método de validación mediante aserciones, éste buscará en el código inconsistencias de tipos de datos o unidades de medida, mostrándose mensajes de error si en un componente existen dichas inconsistencias.
- La fase de depuración se encuentra integrada con la fase de validación en muchas herramientas EUP, proveyendo técnicas de depuración como las aserciones de la fase de validación anteriormente explicada. Una nueva técnica ha sido utilizada satisfactoriamente a lo largo de los últimos años, consiste en la consulta de preguntas por parte de un usuario a una aplicación que dicho usuario esté usando, en caso de tenga dudas acerca del comportamiento de la aplicación. Dicha aplicación contestará con una explicación del comportamiento, previa consulta del histórico de ejecución de la aplicación.

- En referencia a la fase de reutilización o reusabilidad del código fuente de una aplicación de usuario final, supone una dificultad añadida para estos desarrolladores de aplicaciones crear aplicaciones reusables, ya que la mayoría de estos no disponen de la cualificación técnica adecuada para hacerlo. Se han llevado a cabo algunas aproximaciones para reducir esta problemática, como la creación de repositorios que alojan aplicaciones reusables que los usuarios finales podrán buscar para reutilizarlas y satisfacer sus necesidades sin crear aplicaciones desde cero.

2.3 Mashups.

Los *mashups* [1][2][4], en el ámbito de la programación web y de la Web 2.0, son aplicaciones web desarrolladas por usuarios finales que se basan en una unión de componentes ofrecidos por terceros como fuentes de datos, APIs o servicios web, con el objetivo de crear nuevas aplicaciones de mayor potencial y valor que aquellas que disponen de APIs o servicios de componentes web como elementos autónomos sin interacción entre ellos.

Como hemos comentado anteriormente, en la Web 2.0 surgen las aplicaciones desarrolladas por usuarios sin grandes conocimientos técnicos sobre programación, que unen servicios y componentes de terceros en una aplicación de mayor riqueza que la que ofrecen cada uno de esos servicios por separado. Dichas aplicaciones, como se ha comentado, son denominadas como aplicaciones de *mashup*.

La calidad de estas aplicaciones necesita ser estudiada, ya que el éxito de un determinado *mashup* depende de la calidad de los componentes que lo forman como elementos independientes y autónomos, además de la calidad que se puede conseguir al integrar estos elementos en la construcción de dicho *mashup*.

2.3.1. Categorías.

Los *mashups* se pueden dividir en tres tipos o categorías [13]: *mashups* de negocios o de empresa, *mashups* de clientes o consumidores y *mashups* de datos. Cada uno de ellos se explica a continuación:

- *Mashups* de empresa: este tipo de *mashups* definen aplicaciones web en las que se realiza una composición de componentes y servicios internos a la empresa con servicios web oferecidos por terceros.
- *Mashups* de cliente: este tipo de *mashup* está enfocado al público general combinando fuentes de datos, servicios web y APIs públicas en una única interfaz gráfica que representa esta composición. Los *mashups* que están siendo caso de estudio en este trabajo, relacionados con el mundo de la Web 2.0 y la implicación de los usuarios finales en la creación de aplicaciones web de componentes, son *mashups* de cliente
- *Mashups* de datos: estas aplicaciones web surgen de la composición de distintas fuentes de datos de origen, como por ejemplo datos provenientes de feeds RSS, y de la aplicación posterior de filtros y operadores que modifican, filtran y reestructuran esos datos de origen, proveyendo a dichas aplicaciones de una mayor riqueza que aquellas que utilizan esos datos de origen en crudo.

2.3.2. Herramientas para la creación de mashups.

En Internet existen muy diversas herramientas que permiten la creación de mashups mediante la composición de componentes web de terceros, como las que vienen explicadas a continuación:

- Polymer designer [14]: esta herramienta permite la creación de *mashups* de componentes Polymer, tanto de los elementos definidos por el grupo de desarrolladores de esta librería como de elementos creados por los usuarios. Para este último caso, es necesario descargarse el código fuente del designer y añadir manualmente cada uno de los elementos que queramos utilizar.

  Con respecto a la interfaz, ésta permite añadir mediante una técnica de "drag&drop" los elementos al *workspace* de trabajo para poder hacer una composición de ellos. Disponemos de un cuadro en el que se pueden modificar los atributos de los elementos que usemos, así como sus estilos. También, podremos visualizar el código fuente del *mashup* que creemos, guardarlo en el sistema de ficheros local o exportarlo a un Gist de la plataforma Github, así como lanzar una demo para comprobar su funcionamiento.

- Yahoo Pipes [15]: es una herramienta de desarrollo de *mashups* web creada por Yahoo!, la cual permite la creación de *mashups* de datos utilizando componentes web basados principalmente en fuentes de datos en formato CSV o feeds RSS entre muchos otros. El objetivo principal de esta herramienta es el de crear y publicar Rich Internet Applications (RIAs) que surgen de la composición de todos estos componentes.

  Permite en su interfaz gráfica la creación de *mashups* mediante técnicas de "drag&drop" de componentes representados gráficamente como cajas que se conectarán entre ellas, con la utilización de operadores que podrán ser aplicados sobre esos componentes para modificar sus entradas y salidas.

- Wirecloud [16]: plataforma de creación de *mashups* de componentes web centrado en usuarios finales, los cuales sirven una necesidad situacional, es decir, la necesidad de construcción de una aplicación para un propósito inmediato.

  Al igual que las anteriores alternativas, permite la creación de mashups mediante la composición de componentes web construidos como widgets. Es una herramienta muy completa que permite la composición de componentes en *mashups*, previa adquisición e instalación de éstos en el *marketplace* de Wirecloud. Podremos conectar estos widgets mediante el *wiring* que ofrece la plataforma y aplicar sobre éstos operadores para alterar las entradas y salidas de estos widgets.

2.4 Catálogos web colaborativos sobre componentes web y mashups.

Existen una serie de catálogos y repositorios en Internet con una gran cantidad de información sobre componentes webs y mashups. Esta información abarca desde características técnicas como protocolos y estándares utilizados en su desarrollo, aspectos de la arquitectura software o tipos de autenticación utilizados en sus APIs hasta información de los desarrolladores que toman parte en la creación de estos componentes y *mashups* y la URL donde éstos se encuentran almacenados.

Un claro ejemplo de la gran comunidad que se encuentra detrás del desarrollo de estos *mashups* y los componentes web que los forman se encuentra en la aparición de estos catálogos web, actuando algunos como fuente de información y noticias, otros como recubrimiento de los repositorios donde dichos componentes se almacenan y algunos otros como repositorios de dichos componentes. Algunos de los catálogos web con más éxito entre la comunidad de desarrolladores de componentes web y de *mashups* vienen detallados en las siguientes subsecciones.

2.4.1. ProgrammableWeb.

Programmable Web [17] es un portal web líder en información y noticias sobre una gran cantidad de componentes web publicados a través de APIs. En esta web se puede encontrar todo tipo de información sobre características que ofrecen cada una de estas tecnologías, tutoriales de aprendizaje y sobre la comunidad de desarrolladores que trabajan con éstas en el día a día.

Existen alrededor de 15000 APIs en la plataforma, tales como la API de Google Maps, Twitter, Facebook, eBay, Amazon S3, Google App Engine y Flickr entre muchas otras, accesibles desde el directorio de búsqueda de APIs (API directory).

Existe un directorio de búsqueda donde podemos encontrar todo tipo de APIs con sus características técnicas, documentación y comunidad de desarrolladores entre otras. Al realizar una búsqueda de información sobre una API el portal web nos ofrecerá un resumen de éste con el nombre de la API, una descripción y categoría asociadas y la fecha de publicación de la API. Al pinchar en el enlace del nombre de la API buscada, la página nos redirigirá a una ficha de información de las características técnicas y documentación de la tecnología, explicadas en detalle en la siguiente sección.

Además de este repositorio de una gran cantidad de APIs, en el portal Programmable Web también se encuentra un directorio de búsqueda de mashups, en el que aparecerá la misma información de resumen de cada mashup al realizar una búsqueda de éstos en el directorio al igual que en el caso anterior de búsqueda de APIs.

- **APIs.**

  En el directorio de búsqueda de APIs se podrá encontrar todo tipo de información sobre las características de cada API que esté publicada en el portal web:

  - Especificaciones técnicas. En este apartado se muestran características comunes a todas las APIs como:
    - Proveedor de servicio de la API.
    - Endpoint.
    - Página web principal.
    - Categorías principales y secundarias en las que aparecen indicadas las temáticas y propósitos de la aplicación a la que da soporte la API.
    - Protocolos que utiliza (REST/XML-RPC) y formato de los mensajes que intercambia (XML, JSON).
    - Soporte SSL.

        - ApiKits o lenguajes utilizados en la implementación de estas APIs.
        - Foros en los que los usuarios pueden consultar a otros sobre dudas y problemas que les surjan con la realización de aplicaciones que se basen en esas APIs o presentar bugs.
        - URL de cuentas de Twitter para información, noticias y actualizaciones sobre estas APIs o plataformas de desarrolladores que usan estas APIs en el desarrollo de aplicaciones y widgets.
        - Email de contacto con los responsables de la API.
        - Información del autor que ha subido la API a Programmable Web.
        - Modos de autenticación (Api Key, OAuth).

      Otras características técnicas que pueden variar de unas APIs a otras son:

        - Campo en el que se indica si la persona que ha subido la API al portal Programmable Web es el dueño o no de la aplicación.
        - Email o enlace a una página de soporte para desarrolladores.
        - Enlace a una consola para desarrolladores vía web.
    - How-to's o tutoriales sobre cómo utilizar estas APIs para crear aplicaciones y widgets.
    - Desarrolladores que están utilizando esas APIs para sus aplicaciones.
    - Comentarios de usuarios del portal web sobre la API y seguidores que monitorizan las APIs.
    - Mashups en los que participa la API que está siendo visualizada.
- **Mashups.**

  El directorio de búsqueda de mashups podemos encontrar información similar a la que ofrece el directorio de APIs, la cual es la siguiente:

    - Especificaciones técnicas. specificaciones técnicas. En este apartado se muestran características comunes a todas las APIs como:
        - APIs que componen el mashup que está siendo visualizado.
        - Etiquetas que describen la funcionalidad del mashup.
        - URL de la aplicación web donde están alojadas.
        - Empresa que ha desarrollado el mashup.
        - Tipo del mashup (usualmente web).
    - Desarrolladores que están utilizando el mashup en cuestión.
    - Comentarios de usuarios del portal web sobre el mashup.
    - Followers del mashup que está siendo visualizado.

2.4.2. Github.

Github [18] es una plataforma que funciona como un repositorio distribuido vía web que utiliza Git como sistema de control de versiones. Utiliza planes de pago de repositorios tanto gratuitos como de pago para particulares y también para empresas, los cuales ofrecen una serie de repositorios gratuitos y de pago en función de la tarifa que se seleccione.

A lo largo de los años, Github se ha convertido en una de las comunidades de desarrolladores más grandes del mundo, siendo utilizada por muchos de éstos como currículum vitae en lugar del convencional. Esta plataforma es utilizada mayormente como repositorio de código fuente de todo tipo de programas y aplicaciones, mayormente del ámbito web, aunque también es usada por algunos usuarios como catálogo de documentación acerca de un determinado tema.

Al ser una comunidad *open-source*, desarrolladores ajenos a un repositorio pueden ejercer una petición de participación en un proyecto mediante una *pull request*, que las personas a cargo de ese repositorio pueden o no aceptar, permitiendo la colaboración entre desarrolladores para una mayor productividad y calidad de los repositorios de código.

Otra funcionalidad que ofrece Github es la utilización de Gists, pequeños snippets o pedazos de código fuente que pueden contener scripts de instalación de una determinada herramienta o parches de código de repositorios de la plataforma, entre muchas otras funcionalidades.

En el ámbito de los componentes web, la mayoría de ellos son subidos a Github para su reutilización por parte todo tipo de desarrolladores. Al estar subidos estos componentes a Github, muchas páginas que actúan como catálogos web y repositorios de información obtendrán esta información directamente de los repositorios de estos componentes en Github, como veremos en los siguientes apartados.

2.4.3. Bower.

Bower [19] es un gestor de paquetes web para la parte front-end de aplicaciones web. Además, actúa como repositorio de una enorme cantidad de librerías, entre las cuales se encuentran componentes web que podrán ser almacenados y reutilizados por parte de los usuarios en sus aplicaciones.

Cualquier desarrollador de librerías web y en concreto de componentes web puede subir su componente a Bower para ponerlo a disposición del resto de usuarios para que éstos los utilicen en sus proyectos. Este gestor de paquetes permite, mediante un fichero en el formato JSON, almacenar toda la información relativa a las dependencias de librerías web que se están utilizando en un determinado proyecto.

Ésto permite que todos los miembros de un equipo de trabajo compartan ese fichero de configuración con información sobre las dependencias web de la parte front-end y las versiones de cada uno de ellas. En consecuencia, los desarrolladores de un proyecto dispondrán de la misma versión de ese componente, eliminando inconsistencias de versiones que puedan surgir entre dichos desarrolladores.

Para la instalación de dependencias mediante el fichero de configuración, denominado necesariamente bower.json, se instalará mediante la línea de comandos del sistema operativo que corresponda las dependencias web de un proyecto en una carpeta del proyecto, pudiendo ser referenciadas posteriormente en el código fuente.

Por último, cabe decir que este gestor se utilizará conjuntamente con otros gestores de dependencias de la parte del back-end para cubrir esta necesidad de gestión de dependencias web de este módulo.

2.4.4. Custom Elements.

Custom Elements [20] es un repositorio de componentes web subidos a la plataforma Github que actúa como recubrimiento de esta plataforma en este ámbito. Esta web tiene como principal funcionalidad la búsqueda de un subconjunto de componentes web previamente subidos a la plataforma Github, cuyas referencias a sus URLs quedan almacenadas en el site de Custom Elements. Actualmente se encuentran las referencias de 700 componentes web de todo tipo de librerías como Polymer o x-tags entre otras.

Al llevar a cabo la búsqueda de un componente, nos aparecerá información sobre éstos relacionada con la plataforma de Github, información como el nombre del repositorio en Github del componente, su descripción, el número de seguimientos o stars, el número de descargas o forks y el autor del componente web. Podremos ordenar la búsqueda por seguimientos o descargas del componente para comprobar qué componentes son más populares o cuáles son los más descargados por los usuarios.

En lo referente al almacenamiento de las referencias de los componentes web, éstas son previamente subidas por los creadores de los componentes u otros usuarios. Cuando se realice la subida de la URL del repositorio de un componente en Github se creará una issue en el repositorio de Custom Elements de la misma plataforma en la que se indicará que se quiere añadir un nuevo componente a dicho repositorio y se adjuntará el enlace de la referencia al componente en Github.

Esta issue quedará pendiente de revisión por parte del equipo de desarrolladores a cargo de este repositorio, que podrán aceptar este componente en cuestión si cumple con una serie de requisitos. El principal objetivo de utilizar este sistema de control de subida de componentes web es el de disponer de un repositorio público de componentes web funcionales y que cumplan con los estándares definidos por la W3C relacionados con los componentes web.

2.4.5. Component Kitchen.

Component Kitchen [21] es una página web que actúa como un portal de información sobre componentes web de manera análoga a Custom Elements, sin permitir la subida de componentes por parte de los usuarios.

Esta página dispone de un timeline con entradas de noticias sobre componentes web, además de un catálogo de componentes con información detallada de cada uno de ellos relacionada principalmente con los repositorios de Github como por ejemplo información del autor del repositorio, de la documentación, el número de seguimientos del componente o la licencia del repositorio entre otros.

Como podemos ver, esta información es similar a la que ofrecen otros repositorios de componentes que hemos visto anteriormente, añadiendo en algunos componentes (en los que exista) un demo del componente que nos permitirá comprobar el funcionamiento de éste sin acceder a la página web que contiene esa demo.

Esto nos permitirá disponer de un repositorio con información centralizada sobre un componente web, lo que hará que dispongamos de una visión detallada del componente y su funcionalidad, en vistas a la creación de métricas que exploten esta información para la creación de mashups de usuario final.

2.4.6. Built with Polymer.

Built with Polymer [22] es una página web con información acerca de websites que están construidas con Polymer. La creación de dicha página es relativamente nueva, datando del 1 de septiembre de 2014 y es un proyecto open-source que se encuentra en la plataforma Github.

Una funcionalidad interesante de este repositorio es la de presentación de información acerca de qué componentes Polymer son utilizados en cada una de las páginas web de las que guarda información. También ofrece una lista con los componentes web usados en estas páginas y cuántas veces se han utilizado, además de un ránking con los componentes web más populares.

2.5 Modelos de calidad software.

Como hemos visto anteriormente, el desarrollo de componentes y *mashups* por parte de usuarios finales ha crecido en popularidad a lo largo de los últimos años con los avances de la Web 2.0.

Debido a que el número de usuarios que desarrollan estas aplicaciones es cada vez mayor, es necesaria la creación de modelos de calidad para enriquecer los recursos y servicios que estas ofrecen y permitir que dichas aplicaciones sean reutilizables por los demás usuarios.

Para ello, se deberá construir un modelo de calidad que se apoye en los cimientos de estándares establecidos por organizaciones internacionales como la organización ISO en temas relacionados con la calidad del software en general, adaptando estas características de calidad a los desarrollos de aplicaciones de usuario final de *mashup* y a los componentes web que las forman.

2.5.1. Calidad del software.

La calidad del software hace referencia a dos conceptos bien diferenciados, que son la calidad funcional y la calidad estructural del software. Con respecto a la calidad funcional, ésta hace referencia al grado de conformidad del software con un diseño en base a las especificaciones de funcionamiento definidas para éste.

Para el caso de la calidad estructural del software, ésta aborda el análisis de su estructura interna, de requisitos no funcionales del software como su seguridad o mantenibilidad. La calidad del software necesita ser gestionada y, a lo largo de su historia, se han definido modelos de calidad para solucionar esta problemática, basados en el estándar de calidad ISO 9126, el cual se explicará en detalle en la siguiente sección.

Estas aportaciones realizan un análisis de las características del software y presentan un conjunto de atributos de éste que describen y evalúan su calidad. Algunos de estos modelos se organizan de manera jerárquica en base a medidas de calidad o métricas en diversos niveles.

2.5.2. Norma ISO/IEC 9126.

El estándar ISO 9126 [23] ha sido durante muchos años el estándar internacional de evaluación para la calidad del software hasta el surgimiento de la norma ISO 25010, la cual se presentará en la siguiente sección y por la cual ha sido la primera recientemente reemplazada. La ISO 9126 define la calidad del software como la combinación de seis características que representan atributos del software cuya calidad puede ser medida y evaluada [1]. Dicho estándar distingue además entre tres categorías de calidad del software:

- Calidad interna: se basa en un modelo de "caja blanca" que mide las características internas del software relativas al código fuente y a su flujo de ejecución, independientemente del entorno en el que se ejecute dicho software.
- Calidad externa: está basada en un modelo de "caja negra" relativo al comportamiento de los métodos de la API del software. Esta categoría de calidad es la más apropiada para el desarrollo de aplicaciones de *mashup* de componentes web creadas por usuarios finales, ya que al componer dichas aplicaciones las propiedades externas del software son tenidas más en cuenta para la composición de componentes que las internas, que se esconden en estos desarrollos debido a la componetización de sus piezas.
- Calidad en uso: se refiere a la capacidad del software para permitir a usuarios alcanzar diversos objetivos en un determinado contexto y se encuentra íntimamente relacionada con la usabilidad.

Las características de calidad que define el estándar ISO 9126 y a las subcaracterísticas en las que éstas se dividen son las siguientes:

- Funcionalidad: conjunto de atributos que mide el grado en el que las funciones, junto con sus propiedades, satisfacen las necesidades para las que ha sido desarrollado el software. El conjunto de subcaracterísticas en las que se divide esta característica de funcionalidad son la adecuación, precisión, interoperabilidad, conformidad y seguridad.
- Confiabilidad: conjunto de atributos que determinan el ratio en el que el software es capaz de mantener un cierto nivel de rendimiento en unas condiciones definidas durante un determinado período de tiempo. Las subcaracterísticas en las que se divide la confiabilidad son la madurez, tolerancia a fallos y recuperabilidad.

- Usabilidad: conjunto de atributos que abordan la calidad de uso del software por parte de un conjunto determinado de usuarios. El conjunto de subcaracterísticas en las que se divide la usabilidad son el entendimiento, aprendizaje, y operabilidad del software.
- Eficiencia: conjunto de atributos del software que miden la correspondencia entre el rendimiento de éste y los recursos utilizados para su funcionamiento, en unas determinadas condiciones. El conjunto de subcaracterísticas en las que se desglosa esta característica de eficiencia son el comportamiento con respecto al tiempo y el comportamiento con respecto a los recursos utilizados.
- Mantenibilidad: conjunto de atributos que evalúan el esfuerzo empleado en el mantenimiento y actualización en el software. Las subcaracterísticas en las que se divide la mantenibilidad son la analizabilidad, cambiabilidad, estabilidad y testabilidad.
- Portabilidad: conjunto de atributos que cuantifican la capacidad del software para ser migrado de un entorno a otro y para funcionar en cada uno de estos entornos en las mismas condiciones. El conjunto de subcaracterísticas en las que se desglosa la característica de portabilidad son la adaptabilidad, instalabilidad, conformidad y reemplazabilidad.

2.5.3. Estándar ISO/IEC 25010.

El estándar ISO 25010 [24], como se ha mencionado anteriormente, sustituye a la ISO 9126. La norma ISO 25010 revisa el modelo de calidad de su estándar predecesor, denominado por dicho estándar como modelo de calidad del producto software, y añade otro modelo, el modelo de calidad en uso del software.

En lo referente al modelo de calidad del producto, el cual presenta unas características de medición de la calidad similares al modelo de su predecesor y aporta nuevas características como la seguridad y la compatibilidad, realiza una medición de la calidad de métricas internas y externas de la calidad del software. Las características de calidad de dicho modelo quedan especificadas por la norma de la siguiente manera:

- Adecuación funcional: característica de calidad análoga a la de su estándar predecesor, la cual mide el grado en el que un producto software aporta las funciones que cumplen con los requisitos establecidos para éste. El conjunto de subcaracterísticas en las que se divide dicha característica son la adecuación, precisión, interoperabilidad, conformidad y seguridad, al igual que se establece en la ISO 9126.
- Confiabilidad: conjunto de atributos que de manera análogo a su estándar predecesor miden la capacidad del software para llevar a cabo su funcionamiento bajo unas condiciones específicas durante un período de tiempo. Al igual que en la ISO 9126, el conjunto de subcaracterísticas en las que se divide la confiabilidad son la madurez, tolerancia a fallos y recuperabilidad.
- Operabilidad: conjunto de atributos que especifican el grado en el que el software y su funcionamiento pueden ser aprendido, entendido y usado por los usuarios, bajo unas condiciones específicas. El conjunto de subcaracterísticas en las que se divide la operabilidad son la adecuación, reconocibilidad, facilidad de uso, entendimiento, atractividad, accesibilidad técnica y conformidad del software.

- Eficiencia del rendimiento: al igual que la ISO 9126, la eficiencia del software se mide en un conjunto de atributos que evalúan su rendimiento en relación a los recursos utilizados por él. Las subcaracterísticas en las que se desglosa esta característica de eficiencia son el comportamiento con respecto al tiempo, los recursos utilizados y la conformidad del software en relación a la eficiencia del rendimiento.
- Seguridad: conjunto de atributos de medición del grado de calidad de la información y los datos del software en términos de confidencialidad, integridad, disponibilidad y autenticidad de éstos. El conjunto de subcaracterísticas de la seguridad del software son la confidencialidad, la integridad, la no repudiación, la responsabilidad, la autenticidad y la conformidad.
- Compatibilidad: mide el grado en el que un software puede compartir información con otros programas o sistemas software. Las subcaracterísticas en las que se desglosa esta característica de compatibilidad son la reemplazabilidad, la coexistencia, la interoperabilidad y la conformidad en realción a la compatibilidad.
- Mantenibilidad: al igual que en su norma ISO predecesora, esta característica se refiere al conjunto de atributos que miden el grado de calidad en que el software se puede actualizar y mantener. El conjunto de subcaracterísticas en las que se divide esta la mantenibilidad son la modularidad, reusabilidad, analizabilidad, cambiabilidad, estabilidad de las modificaciones, testabilidad y conformidad del software con respecto a su mantenibilidad.
- Transferibilidad: evaluación del grado de calidad en el que el software puede ser migrado de un sistema software a otro. El conjunto de subcaracterísticas en las que se desglosa esta característica de transferibilidad son la portabilidad, adaptabilidad, instalabilidad y la conformidad del software en este ámbito.

Con respecto al modelo de calidad en uso del software, éste enfatiza para dicha categoría de calidad en una medición de ésta mediante un conjunto de cinco características de calidad, desglosadas en una serie de subcaracterísticas que las describen:

- Efectividad: evalúa la calidad del software en relación a su capacidad para que los usuarios que lo utilicen puedan alcanzar una serie de metas. Esta característica mide de manera general la calidad en uso en el ámbito de la efectividad, por lo que no se desglosa en ninguna subcaracterística.
- Eficiencia: esta característica lleva a cabo la medición de la correspondencia de los recursos utilizados por un software para que sus usuarios alcancen sus objetivos. La eficiencia mide de manera general la calidad en uso en este ámbito, por lo que no se desglosa en ninguna subcaracterística.
- Satisfacción: lleva a cabo la cuantificación del grado de experiencia de uso del software por parte de los usuarios en un determinado contexto. Esta característica de calidad se desglosa en las subcaracterísticas de agradabilidad, placer de uso, comfort y confianza.

- Seguridad: esta característica mide en qué grado el uso del software puede conllevar a un estado de peligro para la seguridad de carácter humano o del entorno en el que dicho software participa. El conjunto de subcaracterísticas en las que se divide la seguridad en el modelo de calidad en uso de este estándar son el riesgo de daño económico, el riesgo de daño a la seguridad y a la salud y el riesgo de daño al entorno.
- Usabilidad: esta característica lleva consigo la evaluación de la calidad de uso de un software por parte de los usuarios con el fin de la consecución de sus objetivos. Esta característica de calidad se desglosa en las subcaracterísticas de aprendizaje, flexibilidad, accesibilidad y conformidad con el contexto.

2.5.4. Modelos de calidad web.

Existen modelos de calidad web que se basan en las especificaciones definidas por los estándares ISO que han sido mencionados anteriormente. A pesar de la existencia de dichos modelos de calidad, escasean las propuestas de modelos de calidad específicos para componentes web y desarrollos de usuario final de aplicaciones de la Web 2.0 de *mashup*.

Esta falta de propuestas de calidad se debe a la consideración de la calidad en uso como caracterización de la calidad de dichos *mashups* [1], la cual ha sido cubierta por la investigación llevada a cabo a lo largo de los años en el ámbito de la usabilidad web.

Recientes estudios en este ámbito toman en consideración, además de esta caracterización de la calidad en uso otros aspectos de la calidad de los componentes web que forman estos *mashups* y de los componentes web como elementos independientes, aspectos como la calidad de los datos o de presentación de los componentes web y los *mashups* que forman al combinarse.

2.6 SLAs (Service Level Agreements).

Los acuerdos de nivel de servicio o SLAs (*Service Level Agreements*) [22]* son contratos en los que se establece la especificación de un servicio mediante un contrato acordado entre el proveedor o proveedores del servicio y el grupo de usuarios que lo utiliza. Este contrato puede llevar consigo obligaciones legales o penalizaciones económicas en caso de que una de las partes contratantes no cumpla una serie de requisitos definidos en la SLA.

Dichas SLAs abordan muchos temas como la especificación de los servicios, de su rendimiento, garantías de su cumplimiento, gestión de problemas o términos de rescisión de ese contrato. Debido a que existen un gran número de componentes que dependen de servicios de terceros, es interesante analizar como los términos de servicio y las métricas asociadas a estos acuerdos de nivel de servicio pueden afectar a la percepción de calidad de un componente web determinado.

2.6.1. Niveles de las SLAs.

Las SLAs se definen en tres niveles, los cuales se detallan a continuación:

- SLAs basadas en el consumidor: este acuerdo de nivel de servicio consiste en un contrato entre el proveedor o proveedores de un determinado servicio y un cliente o grupo de clientes en el que se especifican todos los servicios que los clientes utilizarán.

- SLAs basadas en el servicio: este contrato consiste en un acuerdo entre el proveedor o proveedores del servicio y todos los consumidores de dicho servicio sin excepción.
- SLAs multinivel: este tipo de acuerdo de nivel de servicio se divide en tres niveles, cubriendo cada uno de ellos un conjunto diferente de clientes para los mismos servicios. Estos tres niveles son los siguientes:
    - SLA en el nivel corporativo: este nivel aborda la gestión del nivel de servicio de forma genérica, aplicada a cada grupo de clientes que participa en el acuerdo.
    - SLA en el nivel de cliente: dicho nivel aborda las problemáticas de la gestión del nivel de servicio relacionadas con un grupo de clientes concreto, sin tener en cuenta los servicios que estén usando.
    - SLA en el nivel de servicio: este nivel de SLA cubre los temas relacionados con servicios específicos para un grupo concreto de clientes.

2.6.2. Métricas asociadas.

Los acuerdos de nivel de servicio pueden definir una serie de métricas para gestionar el rendimiento de un determinado servicio, con el objetivo de obtener una serie de objetivos a nivel de servicio. Algunas de estas métricas utilizadas en las SLAs se explican a continuación:

- Tiempo de funcionamiento del servicio (uptime): es una de las métricas más comunes en las SLAs y se refiere al tiempo de funcionamiento del servicio y de la red de comunicaciones que éste utiliza.
- Tiempo de caída del servicio (downtime): métrica relacionada con el tiempo total de pérdida de conectividad con un determinado servicio.
- Crédito financiero a abonar en caso de no cumplimiento del servicio: Esta métrica aborda el dinero que deberá abonar una de las partes en caso de incumplimiento de alguno de los requisitos del servicio definidos en la SLA. Se detallarán los porcentajes que deberá abonar la parte contratante en función de que se incumpla con una determinada condición. Un ejemplo de esta situación sería que la parte que no cumpla con un uptime de un 95% abone un 40% del pago del servicio para un determinado período de tiempo a la otra parte.
- Ventana de tiempo de mantenimiento del servicio: como su mismo nombre indica, esta métrica hace referencia al período de tiempo que transcurre durante el mantenimiento de los servicios definidos en la SLA, tiempo durante el cual el servicio no estará disponible para su utilización.
- Instancias del back-end del servicio que funcionan correctamente: métrica que indica el número de instancias de back-end que responden correctamente a los test de balanceo de carga que se realicen por parte del proveedor o proveedores de los servicios definidos en la SLA.

2.7 Tecnologías asociadas.

Las tecnologías de programación relacionadas con los componentes web y su utilización en los desarrollos de aplicaciones de *mashup* de usuario final se detallan a continuación en las siguientes secciones.

2.7.1. Polymer.

Polymer [6] es una librería web creada por Google construida sobre los estándares establecidos por la W3C con respecto a la creación de componentes web. Permite crear tanto componentes web aislados como complejas aplicaciones web de escritorio o móviles.

Debido a que los estándares de la W3C no disponen de soporte por parte de todos los navegadores web, Polymer trae consigo un conjunto de implementaciones de los estándares de la W3C referentes a componentes web (templates, custom elements, shadow DOM e imports).

Estas implementaciones son denominadas *polyfills* y proveen soporte como se ha comentado a navegadores que no soportan de manera nativa dichos estándares, permitiendo a los navegadores que sí los soporten usar su implementación nativa.

Polymer dispone de elementos visuales y no visuales, apoyando estos últimos el funcionamiento de los primeros. Con respecto a los elementos visuales un subconjunto de ellos, los paper elements, son implementan el diseño llevado a cabo por Google, el *Material Design*.

Dicho diseño, se inspira en los elementos del papel y la tinta para crear un diseño interactivo, visual y lleno de movimiento, además de multiplataforma. Android en su versión 5.0, además de las webs de documentación relacionadas con los servicios ofrecidos en la nube por Google, está adaptando este nuevo diseño en este sistema operativo para móviles.

Esta completa librería permite la creación de componentes web y aplicaciones web de mashup con características como la implementación del Material Design de Google o la utilización de *Custom Elements* creados por todo tipo de usuarios que convierten a esta librería en una tecnología front-end muy completa para este tipo de desarrollos.

2.7.2. AngularJS.

AngularJS [8] es un framework open-source para desarrollo web desarrollado por Google en el lenguaje JavaScript. La característica principal de este potente framework es la creación de aplicaciones web front-end mediante le Modelo Vista Controlador (MVC) de una sola página o Single Page Applications (SPAs).

Este framework utiliza la programación declarativa como característica para construir aplicaciones robustas de manera sencilla. Mediante la utilización de directivas HTML una aplicación Angular puede llevar a cabo, mediante un número no tan elevado de líneas de código fuente, funcionalidades de un gran potencial.

Dicho framework permite la inclusión en sus aplicaciones de componentes web a modo de plugins y widgets visuales, que dotan a esta herramienta de desarrollo web de un gran potencial. Angular, unida a la utilización de Polymer para el desarrollo de componentes que siguen los estándares más modernos del mercado, puede llevar a la creación de aplicaciones del lado front-end muy potentes y robustas, de manera sencilla y orientadas al usuario final no profesional.

2.7.3. Google Cloud Platform.

Google Cloud Platform [25] es un conjunto de servicios en la nube ofrecidos por Google que ofertan productos o servicios de Infraestructura como servicio (IaaS) o Plataforma como Servicio (PaaS), además de servicios de bases de datos (tanto relacionales como no relacionales) y de creación de APIs. La familia de servicios ofertados por Google son los siguientes:

- Google App Engine [26]: servicio ofrecido en la nube por Google como una Plataforma como Servicio (PaaS) utilizada para alojar aplicaciones web que se ejecutarán en la infraestructura de Google.
- Google Compute Engine [27]: servicio en la nube ofrecido como Infraestructura como Servicio (IaaS) el cual permite lanzar máquinas virtuales bajo demanda.
- Google Cloud Storage [28]: servicio REST ofrecido por Google en la nube para almacenamiento de ficheros.
- Google Cloud SQL [29]: servicio de bases de datos relacionales ofrecido en la nube. Se caracteriza por la alta disponibilidad de dichas bases de datos, su flexibilidad y su facilidad de uso.
- Google Cloud Datastore [30]: servicio de bases de datos no relacionales proveído por Google caracterizado por la alta disponibilidad de sus servicios y su fácil administración.
- Google BigQuery [31]: servicio REST ofrecido en la nube que consiste en una Infraestructura como Servicio utilizada para realizar consultas de cantidades ingentes de datos sin coste de hardware y software adicional.
- Google Cloud Endpoints [32]: servicio ofrecido por Google en la nube que permite crear servicios web REST para que sean accedidos por clientes web de escritorio y móviles. Permite la administración de las claves de clientes y la compatibilidad con el sistema de autenticación Auth 2.0.
- Google Cloud DNS [33]: servicio de DNS alojado en la infraestructura de Google de alta disponibilidad y escalabilidad.

2.7.4. Google Analytics.

Google Analytics [34] es un servicio web ofrecido por Google que genera estadísticas sobre páginas web relativas al tráfico generado y a la interacción de los usuarios con dichas webs. En el ámbito de las métricas de calidad se puede utilizar este servicio para recolectar información relacionada con éstas y para su posterior consulta mediante el acceso a las APIs de la plataforma Analytics.

Para llevar a cabo la vinculación de la información de las métricas de calidad con Google Analytics se utilizarán las APIs y SDKs de esta plataforma, la cual está dividida en cuatro secciones: collection, encargada de recolectar la información de la interacción del usuario con la aplicación; configuration, cuya función es la de establecer cómo se va a gestionar la información que se va a procesar; processing, la cual procesa la información que se va a enviar a Analytics en conjunto con la información de la sección de configuration y reporting, que permite el acceso a toda la información procesada por la anterior sección.

Con respecto a las APIs y SDKs a utilizar, se consideran las siguientes:

- Collection:

  Este SDK se encarga del envío de datos a la plataforma Analytics, datos que podrán ser consultados posteriormente por la API de la sección Reporting (Core Reporting API). Esto se consigue mediante la inclusión en cada página de la web o vista de la aplicación web de la que queramos enviar datos a Analytics de un fragmento de código JavaScript, el cual cargará de forma asíncrona del código de seguimiento de Google Analytics en la página y cuya función principal es la de medir las interacciones de los usuarios con la aplicación. La API de Collection utiliza varios componentes de medición de datos:

  - Web Tracking, el cual mide las interacciones de los usuarios con las páginas y aplicaciones web mediante el código de seguimiento JavaScript analytics.js.
  - Android, el cual evalúa interacciones de los usuarios con aplicaciones Android.
  - IOS, que evalúa las interacciones con aplicaciones iOS de los usuarios.
  - Measurement Protocol, un protocolo de bajo nivel que mide las interacciones de usuarios en los lados front-end y back-end.

  Mediante la API de Web Tracking de Collection podremos enviar datos de los llamados *hits* que se pueden llevar a cabo en la página (en el lado front-end), los cuales son:

  - *Screenview* o visita de pantalla (aplicaciones web).
  - *Pageview* o vista de página (páginas web).
  - *Event* o *hit* de evento (clicks sobre anuncios, clicks sobre botones de descargas, enlaces a páginas externas, etc.).
  - *Social* o *hit* de interacciones sociales (Me gusta de Facebook, Retweet de Twitter o +1 de Google+).
  - *Timing*, relacionada con tiempos de ejecución (tiempos de carga de librerías o tiempos de ejecución de contenido de la aplicación).

- Configuration.

  Mediante la API de configuración podemos acceder a toda la información de gestión de Google Analytics, la cual se divide en dos componentes:

  - La Management API es utilizada para acceder y gestionar cuentas de usuario de Google Analytics, así como muchas otras características como permisos de usuarios, objetivos a conseguir por la aplicación o importación y exportación de datos a bases de datos.
  - La Provision API usada para crear nuevas cuentas de Google Analytics.

- Reporting.

  Este SDK permite el acceso a la información procesada por los anteriores SDKs, la creación de informes personalizados y la intergación de datos de Google Analytics con otras aplicaciones de terceros.

El acceso a la información se realiza mediante peticiones a la API organizadas según unas dimensiones y métricas. Las dimensiones son atributos medibles de la aplicación a los que se les puede asignar, según un determinado criterio, diferentes valores o métricas. Los informes estarán poblados con información sobre estos dos parámetros, las dimensiones y las métricas, los cuales pueden relacionarse entre ellos dando coherencia y un mayor significado a los datos. Cabe destacar que se pueden crear dimensiones y métricas personalizadas, algo que dota de mucho potencial a la plataforma Analytics.

2.7.5. Big Query.

Big Query [31] es un servicio web REST que permite realizar consultas de datos masivas de manera muy rápida utilizando la infraestructura de Google. Se puede definir como una infraestructura como servicio que permite manejar cantidades enormes de datos sin coste de hardware y software adicional.

Este servicio permite gestionar información almacenada en servicios de Google anteriormente explicados como App Engine Datastore, datos de Google Analytics o gestión de ficheros extensos de logs de App Engine. Para el ámbito de las métricas de calidad, Big Query al manejar cantidades ingentes de datos, podrá utilizarse para el procesamiento de métricas de una gran cantidad de componentes web software cuyo cálculo requiera de una infraestructura muy potente.

Capítulo 3.

Propuesta de un marco de caracterización de métricas de calidad para componentes web.

Índice

A lo largo de los años [1], ha aflorado el desarrollo por parte de usuarios finales con pocos conocimientos de programación de aplicaciones web basadas en componentes. La creación y selección de componentes ha sido basada en aspectos puramente funcionales, sin tener en cuenta la calidad de los componentes web y las aplicaciones o *mashups* que surgen de la integración de éstos.

La calidad de las aplicaciones web de usuario final se encuentra influenciada por el valor añadido que la combinación de los componentes es capaz de ofrecer. Por otro lado, también depende enormemente de la calidad de los componentes web como elementos aislados e independientes, debido a que dicha aplicación surge de la integración de componentes o piezas independientes.

De esta manera, podemos establecer un modelo de calidad para componentes web compuesto por atributos que evalúen la calidad de éstos, basados en las características recogidas por los estándares de calidad tradicionales, adaptando y concretando dichos atributos para evaluar la calidad de las características externas y de uso de los componentes web.

Por este motivo, se considera a los componentes web como cajas negras que exponen públicamente sus APIs para que los desarrolladores de aplicaciones de integración de componentes puedan utilizarlos en sus aplicaciones web.

Las características que definen dichas APIs son su usabilidad, funcionalidad o accesibilidad entre muchas otras, las cuales contribuyen al éxito de un componente, y por ende, a su calidad.

3.1. Diseño del modelo de calidad para componentes web.

Desde el punto de vista del desarrollo de aplicaciones web de *mashup* de usuario final, la calidad interna de los componentes web, la cual está basada en un modelo de caja blanca apoyado en la calidad del código fuente y su flujo de funcionamiento, tiene una gran importancia en el desarrollo de componentes web como elementos autónomos pero no en la creación de *mashups*.

Para la integración de determinados conjuntos de componentes para la creación de *mashups* de usuario final la calidad que debe ser tenida en cuenta es la calidad externa de los componentes, más concretamente de sus APIs, debido a que esta categoría de calidad se basa en un modelo de caja negra que evalúa las características externas de un componente web que lo hace candidato a ser interoperable con otros, para la creación de *mashups*. Además de dicha calidad externa, la calidad en uso de los componentes, cubierta por la enorme investigación llevada a cabo acerca de la usabilidad de las aplicaciones web, también debe tenerse en mente a la hora de diseñar un modelo de calidad de componentes web y *mashups*.

Existen pocas propuestas de modelos de calidad de componentes web y *mashups* con motivo de la investigación en usabilidad web llevada a cabo a lo largo de los años. Dichos modelos atribuyen la calidad de los *mashups* y los componentes que los forman a características funcionales de estos y a la usabilidad que ofrecen, aunque atributos externos de los componentes como la disponibilidad de documentación acerca de su funcionamiento, su impacto social o la calidad de los datos que ofrece entre otros, no son contemplados para la evaluación de la calidad por estos modelos.

Teniendo todo esto en consideración, se ha llevado a cabo el desarrollo de un modelo de calidad derivado del estándar sucesor de la norma ISO 9126-1 [23], la norma ISO/IEC 25010 [24]. El modelo de calidad propuesto, sigue una jerarquía organizada según unos niveles de importancia de calidad. Dichos niveles se clasifican, de mayor a menor importancia de calidad en el modelo, en calidad global del componente, factores, criterios y métricas de calidad. La calidad global del componente se encuentra enfocada en torno a cinco aspectos principales: calidad de los datos, de la API del componente, de presentación, de impacto social y del proveedor y hosting del componente, los cuales se dividen en otras características de grano más fino y éstas a su vez serán evaluadas mediante métricas de calidad.

En las siguientes secciones se explicarán en detalle estos cinco aspectos de calidad, que quedan reflejados en el modelo de la siguiente figura:

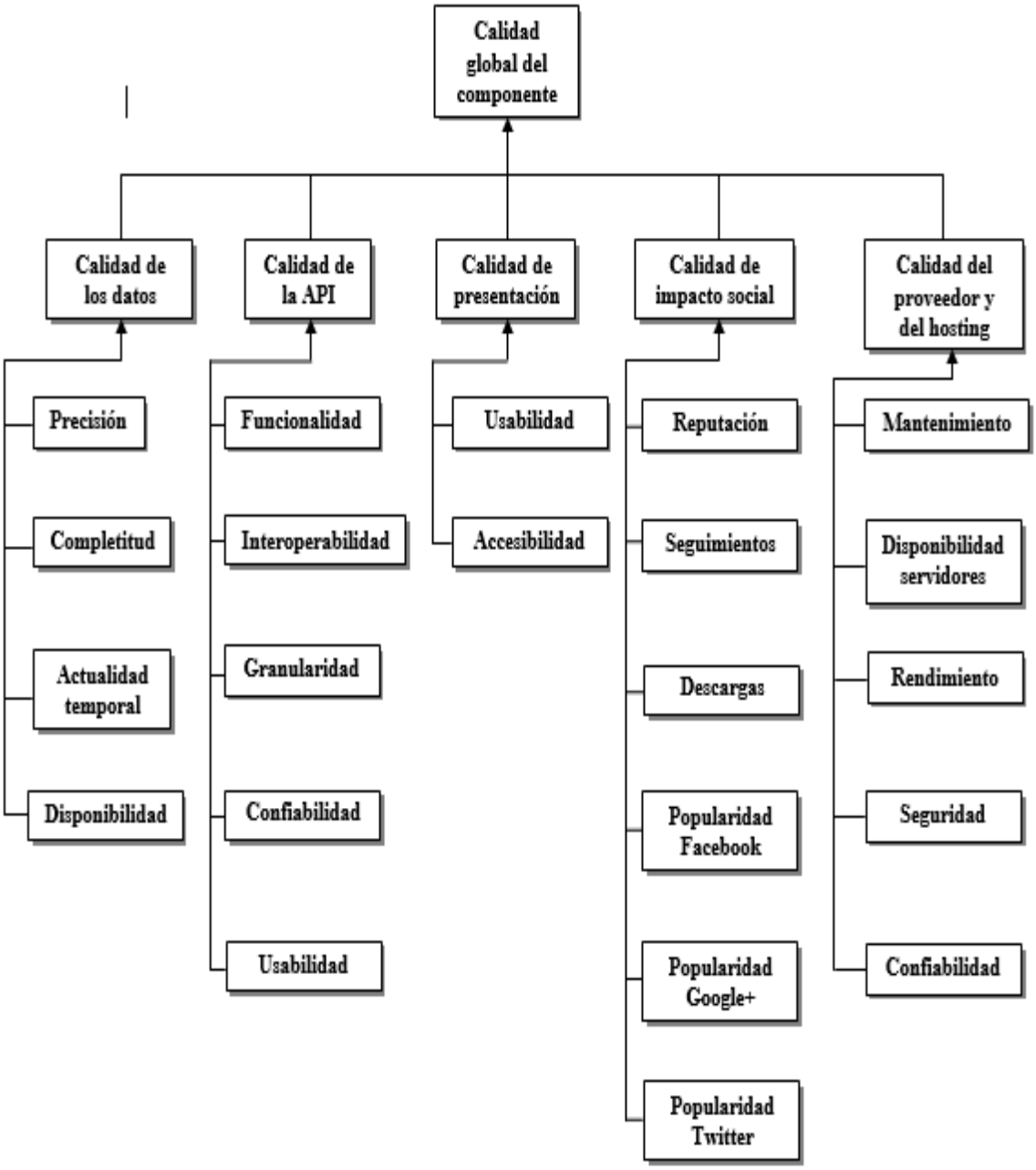

Figura 1: Modelo de calidad de componentes web

En referencia a los factores, criterios y métricas de calidad del modelo para componentes web, quedarán descritos en la siguiente tabla y explicados en las siguientes secciones de manera detallada:

Factores	**Criterios**	**Descripción**	**Métricas**
Calidad de los datos	Precisión	Probabilidad de que el funcionamiento con los datos que provee el componente es correcto.	p(dcorr)
	Completitud	Grado de completitud en la implementación requerida para el componente.	(CDR / CDI) * 100
	Actualidad temporal	Frescura de los datos obtenidos.	(edad / validez)^{s} * 100
	Disponibilidad	Probabilidad de que un componente pueda proveer datos.	(∑ petExito / ∑ petTotales) * dispon_API
Calidad de la API	Funcionalidad	Número de RIAs en las que participa el componente.	cantRIAs + conformidad + segAcceso
	Interoperabilidad	Número de componentes interconectables con uno dado en función de sus entradas y salidas	(IO_cw / atributos_parametrizables) * 100
	Granularidad	Grado de detalle de la funcionalidad del componente.	buena, regular, mala
	Confiabilidad	Habilidad de un componente para llevar a cabo sus funciones bajo unas condiciones durante un período de tiempo específico en términos de madurez.	(actualidad / edad_media_vers) * 100
	Usabilidad de la API	Facilidad de uso de la API de un componente. Medido en términos de aprendizaje y entendimiento (disponibilidad de documentación, ejemplos, blogs o forums) y operabilidad (complejidad de protocolos, lenguajes, formatos de datos y mecanismos de seguridad).	buena, regular, mala
Calidad de presentación	Usabilidad	Usabilidad de la interfaz visual en términos HCI.	(∑directrices_cumplidas /|directrices|) * 100

	Accesibilidad	Accesibilidad visual a personas discapacitadas	nivel_WCAG2.0 U grado_WAI-ARIA1.0
Calidad de impacto social	Reputación	Valoración de componentes por parte de usuarios finales	∑valoracion_usuarios / |valoraciones |
	Seguimientos	Popularidad del componente relativa a seguimientos de éste por parte de los usuarios	buena, regular, mala
	Descargas	Popularidad del componente relativa a descargas por parte de los usuarios	buena, regular, mala
	Popularidad en Facebook	Popularidad entre los usuarios de Facebook	buena, regular, mala
	Popularidad en Google+	Popularidad entre los usuarios de Google+	buena, regular, mala
	Popularidad en Twitter	Popularidad entre los usuarios de Twitter	buena, regular, mala
Calidad del proveedor y del hosting	Mantenimiento	El proveedor proporciona actualización y reparo de bugs	∑(bugs_resueltos / bugs_totales)[x versión] |versiones|
	Disponibilidad	Los servicios web y recursos empleados están accesibles	UP + MT + [(∑ petEx / ∑ petTot)*dispon_SRV]
	Rendimiento	Los tiempos de acceso a servidores no superan el tiempo mínimo de sesión de la RIA	buena, regular, mala
	Seguridad	Los servidores tienen certificados y firmas digitales de confianza válidos para el cliente	buena, mala
	Confiabilidad	Los servicios web y recursos aseguran la implementación del no repudio para las comunicaciones	buena, mala

Figura 2: Tabla de factores, criterios y métricas de componentes web

3.2. Calidad de los datos.

La calidad de los datos se centra en la pertinencia de los datos que provee el componente y si éstos representan correctamente a su representación equivalente en la vida real. Las características de grado fino que se tienen en cuenta para representar este aspecto de calidad son la precisión, la completitud, la actualidad temporal y la disponibilidad, las cuales se verán detalladas a continuación [1]:

3.2.1 Precisión.

Esta característica de calidad aborda la exactitud de los datos del componente web y la consistencia entre éstos y sus homólogos en el mundo real. Es posible realizar una evaluación de la precisión de los datos del componente mediante una métrica que calcule la probabilidad de que los datos sean correctos, la cual es la complementaria de la probabilidad de error de los datos, mediante la siguiente fórmula:

metrica_precision = p(dcorr) , con

p(dcorr) = 1 – p(derr) , donde

p(dcorr) es la probabilidad de que los datos sean correctos

p(derr) es la probabilidad de que haya un error en los datos

Los errores en los datos pueden ocurrir debido a representaciones de datos erróneas, errores tipográficos o actualizaciones de los datos inexistentes, los cuales quedan recogidos en el histórico de uso del componente.

3.2.2 Completitud.

La completitud de los datos se define como el grado en el que un componente implementa la funcionalidad requerida para éste. Para cumplir con los requisitos definidos para un determinado componente, es necesario que dicho componente sea capaz de proveer un conjunto de datos al usuario para que éste lo pueda utilizar.

En un ámbito teórico, la intención del desarrollador de un determinado componente es la de proveer un conjunto de datos completo, el conjunto de datos ideal (CDI). Desde el punto de vista práctico, solamente un subconjunto de este CDI denominado conjunto de datos situacional (CDS) es necesario para cumplir con los requisitos de un componente.

Por este motivo, la completitud de los datos de un componente se mide a través de una completitud situacional de los datos, la cual evalúa como dicho componente es capaz de proveer la información deseada. De este modo, es posible utilizar una métrica que cuantifique esta completitud situacional, con la siguiente fórmula:

$$\textbf{metr\_completitud (situacional)} = \frac{\textbf{CDS}}{\textbf{CDI}} * 100$$, donde

CDS es el conjunto de datos situacional del componente

CDI es el conjunto de datos ideal del componente

3.2.3 Actualidad temporal.

La actualidad temporal de los datos define el grado de frescura de los datos que provee un determinado componente web, es decir, cómo está de actualizada la información de dicho componente de cara a los usuarios. Para realizar una evaluación de la frescura de los datos del componente se puede utilizar una métrica de calidad que viene determinada por la siguiente fórmula:

metr_frescura = max (0,1 – (actualidad_datos / volatilidad_datos))^{s} * 100, donde

actualidad_datos es el período de tiempo que ha transcurrido desde que los datos fueron creados o actualizados por última vez

volatilidad_datos se refiere a una dimensión estática que representa el período de tiempo de validez de los datos

s es el exponente que controla la sensibilidad de esta actualidad de los datos, cuyo valor se estima de manera subjetiva por parte del ente que analice los datos, dependiendo del contexto en el que se encuentren éstos

3.2.4 Disponibilidad.

La disponibilidad de los datos de un componente web hace referencia a la probabilidad de que dicho componente pueda proveer datos, es decir, la probabilidad de que un determinado componente esté disponible para su uso por parte de los usuarios.

Una práctica común en el desarrollo de componentes es la utilización de licencias de usuario con restricciones de uso. Estas limitaciones, como por ejemplo las restricciones por IP de peticiones de ubicaciones a servicios web como Google Maps, son vistas como limitaciones desde el punto de vista de los usuarios, pero son necesarias para que no exista un abuso de determinados servicios de terceros por parte de los componentes web. Se puede asociar una métrica de calidad a esta cuantificación de la disponibilidad de los datos que ofrece un determinado componente, mediante el cálculo del grado en el que las peticiones de datos con éxito se corresponden con el número de peticiones totales por día por parte de los usuarios. Ésto, queda plasmado mediante la siguiente fórmula:

metr_disponibilidad_componente = $\frac{\sum \text{petExito}}{\sum \text{petTotales}}$ *** disponAPI (por día)** , donde

petExito indica la cantidad de peticiones de datos por día en las que el componente responde con los datos que espera el usuario

petTotales indica la cantidad total de peticiones de datos diarias realizados por los usuarios a un componente

disponAPI hace referencia a la disponibilidad que ofrezca la API que provea los datos; si el proveedor de la API no ofrece un valor de disponibilidad se asumirá un valor del 100% (un valor de 1 para el cálculo de esta métrica)

3.3. Calidad de la API del componente.

En esta sección se abordan los temas relacionados con las características software evaluables sobre la API software de los componentes web. Las características de grado más fino que se consideran para representar este factor de calidad son la funcionalidad, interoperabilidad, granularidad, confiabilidad, y usabilidad de la API, que se explicarán en detalle a continuación:

3.3.1. Funcionalidad.

El aspecto de funcionalidad de la API de un componente depende de los siguientes tres atributos o métricas de calidad, las cuales vienen detalladas a continuación:

- Cantidad de RIAs en las que participa un componente.

  Este atributo mide el número de aplicaciones de *mashup* en las que participa un componente determinado, lo cual se podrá cuantificar cuantificando el número de aplicaciones o el número de *workspaces* de una herramienta de desarrollo de *mashups*.

  Podrá llevarse a cabo una clusterización de la cantidad de RIAs en las que participa un componente. Mediante este procedimiento, será posible clasificar un componente determinado en alguna de las clases resultantes de la aplicación de dicho procedimiento al conjunto de componentes web que esté siendo analizado.

- Conformidad. Atributo de calidad que evalúa si un determinado componente utiliza un formato de datos estándar para la comunicación de datos con otros componentes. La métrica de calidad asociada con la conformidad viene definida por un valor booleano, cuyo valor binario tendrá el siguiente significado:
  - La métrica llevará asociado el valor 1 para un determinado componente si alguno de los formatos de datos que soporta dicho componente es estándar.
  - La métrica de calidad obtendrá un valor de 0 para un componente si ninguno de los formatos de datos soportados por el componente para la comunicación de datos es estándar.

- Seguridad de la API. La característica de seguridad de la API de un determinado componente web viene determinada por el mecanismo de acceso a las funcionalidades de dicho componente. Se distingue entre dos aspectos, el soporte SSL y los mecanismos de autenticación. En relación al soporte SSL, puede o no ofrecerse una comunicación con encriptación por parte de un componente, algo que mejorará considerablemente la seguridad de su API. Con respecto a los mecanismos de autenticación, se distinguen los siguientes: ausencia de autenticación, API key, clave de desarrollador o cuenta de usuario, en orden creciente de seguridad. Si el objetivo del desarrollador de una aplicación que desea integrar componentes web es el de utilizar una API key, generará una clave (la cual normalmente puede ser generada en el sitio web del proveedor del componente) única y específica del *mashup* que va a utilizar un componente determinado.

  En caso de que el desarrollador de un *mashup* desee utilizar una clave de desarrollador, éste deberá registrarse como desarrollador en el sitio web del proveedor del componente.

Por otro lado, si desea utilizar el mecanismo de autenticación de la cuenta de usuario, éste deberá registrarse con ese rol en la página web del proveedor del componente. La métrica de seguridad de la API del componente quedará especificada por medio de la siguiente fórmula:

seguridad_API = uso_SSL + tipo_autenticacion , donde

uso_SSL es un valor booleano que indicará si un componente utiliza o no SSL como mecanismo de encriptación para las comunicaciones

tipo_autenticacion es un valor del 1 al 4, en orden creciente de seguridad, siendo el valor 1 la indicación de que no se utiliza un mecanismo de autenticación para el acceso al componente web

La valoración de la métrica de seguridad depende de los requisitos de los desarrolladores de aplicaciones de *mashup*. Si un individuo desea utilizar componentes que disponga como mucho de una clave de desarrollador como mecanismo de autenticación y soporte para encriptación SSL, la utilización de un componente que imponga el uso de una cuenta de usuario como mecanismo de autenticación no proporcionará ningún valor añadido. Por otro lado, un componente que no disponga de un mecanismo de autenticación o disponga de uno con API key, no cumplirá con los requisitos del desarrolador de *mashup*. De este modo, se asignará un valor de calidad a la métrica de seguridad según las expectativas del desarrollador de *mashups*, constante según se alcancen o se superen las expectativas de éste, o menor para los componentes que no alcancen dichas expectativas.

Una vez han quedado definidos estos tres atributos de medición de la calidad de funcionalidad de un componente determinado, dichos atributos serán calificados y sus valores transformados posteriormente a la escala uniforme de medición del site, la cual se detallará en capítulos posteriores. Se realizará la ponderación de los pesos referentes a la importancia que tendrán cada uno de estos atributos respecto al cálculo final de la métrica de funcionalidad, de la siguiente manera:

metr_funcionalidad (componente) = metr_cantidadRIAs + metr_conform + metr_seg , donde

metr_cantidadRIAs es el valor de la interoperabilidad de un componente

metr_conform supone el valor de conformidad de un componente

metr_seg supone el v**a**lor de la métrica de seguridad de acceso a la API de un componente

3.3.2. Interoperabilidad.

Es un atributo que define el grado en el que un determinado componente es capaz de integrarse con otros en un ambiente totalmente heterogéneo en función de las entradas y salidas de éste. La cuantificación de la interoperabilidad de un componente se puede realizar a través de una métrica en el que se cuantifique el grado en el que los atributos parametrizables de un componente web determinado se corresponden con entradas y salidas de éste. Para ello, se utilizará la siguiente fórmula:

$$\textbf{metr\_interoperabilidad (componente)} = \frac{|\,\textbf{entradasYsalidas\_cw}\,|}{\textbf{attr\_parametrizables}} * 100$$

3.3.3. Granularidad.

Esta característica de calidad se refiere al grado de detalle, en lo que a la funcionalidad que implemente se refiere, que tiene un componente web determinado. Dicho componente surge de la integración de múltiples APIs o servicios de terceros, y cada uno de ellos enriquecerá a éste con funcionalidad.

Debido a esto, la granularidad de un componente web dependerá del grosor de funcionalidad que las APIs que integre el componente ofrezcan a este último. Por ello, los conjuntos de datos que ofrezcan dichas APIs, y por ende, el conjunto de datos ofrecido por el componente para su utilización en composiciones de aplicaciones de mashups deberá ser evaluado mediante una métrica de calidad que comparará el conjunto de datos ofrecido por el componente con el resto de componentes que estén siendo objeto de estudio.

Podrá llevarse a cabo una clusterización de estos conjuntos de datos que ofrecen los componentes, y mediante este procedimiento clasificar un componente determinado en alguna de las clases resultantes de la aplicación de este proceso al conjunto de componentes web que está siendo analizado.

Según esta clusterización, si un determinado componente se encuentra en una de las clases resultantes de dicho proceso de clusterización en las que la funcionalidad de los componentes tenga una granularidad muy gruesa, este componente dispondrá de una buena calidad para esta métrica.

Por el contrario si el componente se clasifica según este proceso en una clase de componentes de granularidad más fina, dispondrá de una calidad calificada como "mala". De este modo la métrica que clasifica la granularidad quedará determinada de la siguiente manera:

metr_granularidad = buena, regular, mala

3.3.4. Confiabilidad.

Las técnicas de caja negra por las que se mide la calidad de un componente no permiten evaluar el rendimiento de un componente midiendo las características que ofrece el código fuente y su flujo de trabajo, por lo que esta fiabilidad del componente se medirá según la madurez de éste, considerando las estadísticas de su uso y su frecuencia de actualizaciones, mediante la siguiente fórmula:

$$\text{metr\_confiabilidad (componente)} = \max\left(1 - \frac{\text{actualidad\_cw}}{\text{edad\_cw}}\right) * 100 \;;\; 0$$

, donde

actualidad_cw es el tiempo transcurrido desde el último uso del componente

edad_cw es el período de tiempo desde que se creó el componente hasta la actualidad

|num_rel| se refiere a la cantidad de releases (versiones) que se han publicado del componente en cuestión

3.3.5. Usabilidad de la API.

Esta característica de calidad hace referencia a la facilidad de uso de la API del componente por parte de los usuarios. Dicha usabilidad puede ser medida en torno a los siguientes aspectos: entendibilidad, aprendizaje y operabilidad. La intención de este modelo de calidad es el de evaluar la calidad externa del componente, la cual mediante una aproximación basada en técnicas de caja negra, se refleja en la calidad de la API del componente web.

Mediante dicho modelo de caja negra, los aspectos de entendibilidad y aprendizaje de la API no pueden ser evaluados sobre el código fuente y el flujo de control (modelo de caja blanca), por lo que deberá cuantificarse la calidad de la documentación que ofrece el proveedor del componente web por medio de wikis, tutoriales, ejemplos, o blogs entre otros. La disponibilidad de este tipo de documentación ayuda a aumentar la usabilidad de la API del componente web.

En lo que respecta al aspecto de operabilidad, éste depende de la complejidad de las tecnologías utilizadas en los niveles de aplicación y de datos, además de los mecanismos de seguridad usados en los componentes web:

- En el nivel de aplicación, es posible utilizar una métrica de calidad que mida la difusión de la API en relación a la complejidad de la interacción de otros componentes web con ésta. La usabilidad de la API será mayor cuanto menos compleja sea la interacción con ésta.
- En la capa de datos, de manera similar al nivel de aplicación, es posible medir la calidad en este nivel por medio del análisis de los formatos de datos que ofrece un componente, extrapolando esta información a una métrica de calidad que evaluará si el formato de datos es estándar o no lo es, y si un determinado componente necesitará de una transformación en el formato de sus datos cuando sea integrado en un *mashup*.
- En lo referente a la operabilidad de la seguridad de la API, ésta y la seguridad real del componente son inversamente proporcionales. Esto es debido a que la mayor complejidad que conlleva utilizar un mecanismo de seguridad más restrictivo.

Una vez que se han clasificado estas tecnologías, según unos criterios determinados, puede definirse una clusterización en la que se definen una serie de clases, cada una con un "nivel de operabilidad". Cada tecnología que pertenezca al mismo clúster tendrá asignado un idéntico nivel de operabilidad. Debido a que un componentes puede integrar multitud de APIs y tecnologías, la métrica de calidad de operabilidad viene definida por la siguiente fórmula:

operabilidad_componente = max(op(prot U leng)) + max(op(df)) + max(op(seg)) , donde

prot son los protocolos que intervienen en las APIs de un componente

leng son los lenguajes de programación usados en las APIs que integra un componente

df son los formatos de datos utilizados por los componentes en las comunicaciones de datos con otros componentes web

seg son los mecanismos de seguridad empleados en las APIs de un componente

La fórmula anterior considera que la operabilidad total de un determinado componente viene determinado por el sumatorio de las operabilidades máximas de protocolos y lenguajes, formatos de datos y mecanismos de seguridad de todas las APIs que integra dicho componente.

Acorde a esta clusterización de operabilidad para este ámbito, si un componente se encuentra en una de las clases resultantes de dicho proceso de clasificación en las que su operabilidad sea elevada, este componente tendrá asociada una buena calidad en relación a esta métrica. En caso contrario, si el componente pertenece a una clase con escasa operabilidad, llevará asociada una calificación de mala calidad. De esta manera la métrica que clasifica la usabilidad de la API de un componente quedará determinada de la siguiente manera:

metr_usabilidad_API (componente) = buena, regular, mala

3.4. Calidad de presentación.

La calidad de presentación tiene en cuenta la usabilidad del componente en términos de HCI (Human Computer Interaction), la accesibilidad y la reputación que aportan los usuarios finales al componente. Las subcaracterísticas de grado fino consideradas para evaluar la calidad de presentación de los componentes web son las anteriormente mencionadas de usabilidad y accesibilidad. Estas subcaracterísticas de calidad se detallan a continuación:

3.4.1. Usabilidad.

La usabilidad del software [35] y de la interfaz que se presenta al usuario se evalúa en torno a cinco componentes de gran importancia: el aprendizaje, la eficiencia, la memorabilidad, los errores y la satisfacción de los usuarios que lo utilizan.

Los componentes web, en especial los UI widgets, presentan una interfaz de usuario en la que se muestran los datos que produce el componente, además de mecanismos de interacción con éste. Aunque las interfaces sean muy simples, la usabilidad debe ser tenida en cuenta, y aspectos como el entendimiento y el aprendizaje, la memorabilidad o la provisión de interfaces gráficas fáciles de entender toman una gran importancia en el desarrollo del componente.

Debido a que estos componentes web pueden ser tratados como aplicaciones web, los atributos y métricas de calidad a tener en cuenta serán los mismos que han sido tenidos en cuenta para la usabilidad de aplicaciones y páginas web.

Se han determinado una serie de directrices o guías de usabilidad para componentes web tomando como base los estudios llevados a cabo a lo largo de los años por expertos en usabilidad en entidades como la Nielsen Norman Group [36] y la W3C [37]. Esto se debe a que los componentes no son aplicaciones web por sí mismas, por lo que necesitan de una serie de directrices de usabilidad que no se corresponden enteramente con las directrices que siguen las aplicaciones web con el fin de ser usables, sino que son un subconjunto de éstas. Dichas directrices, las cuales se agrupan en un conjunto de categorías que tratan de cumplir con una serie de objetivos de usabilidad, vienen detalladas a continuación:

- Comunicación del propósito del componente. Un usuario final sin conocimientos de programación que utilice un componente debe ser capaz de deducir a primera vista cual es el objetivo para el que fue creado. Por lo tanto dicho componente deberá conseguir comunicar al usuario ese objetivo por medio de una serie de métricas de usabilidad:
  - Inclusión del nombre de la empresa, institución o desarrollador del componente web en un lugar visible.
  - Posicionamiento del logo de la compañía que ha creado el componente o en su defecto el título del componente web en un lugar visible y con un tamaño razonable.
  - Introducción de un párrafo en el que se indique cuál es el propósito y la funcionalidad del componente web, o en ausencia de éste un eslogan que determine la funcionalidad para la que dicho componente fue creado.
- Contenido web del componente. Escribir contenido en componentes web de manera eficiente es un aspecto muy importante. Los usuarios suelen escanear la información que aparece en las aplicaciones web (en este caso concretamente en los componentes web que en esencia también lo son) y no suelen leer la información que proveen de manera detallada, por lo que es importante llamar la atención de estos usuarios para conseguir que utilicen un determinado componente y no otro de la competencia. Ésto se consigue mediante un conjunto de métricas de calidad que se detallan a continuación:
  - El desarrollador de un componente debe seguir de manera consistente estándares de estilo y de uso de mayúsculas. De no seguir dichos estándares, el usuario puede percibir el componente como no fiable o no profesional.
  - Se debe evitar que en un componente exista un solo elemento agrupado como categoría o que aparezca aislado en una lista, lo que llevaría a un rediseño del contenido de dicho componente.
  - Si en un determinado componente aparecen siglas cuyo entendimiento y significado no esté extendido mundialmente, se debe explicar su significado para el entendimiento de ellas por parte de los usuarios que utilicen ese componente.
  - Las exclamaciones no pertenecen a un estilo de redacción profesional, por lo que no son susceptibles de uso dentro del contenido de un determinado componente.
  - Los enlaces a otras páginas de contenido que muestre un componente aparecen con un color diferente si han sido visitados por un usuario o si no lo han sido. Dicho color no debe ser gris ya que es más difícil de leer que otros para personas con problemas de visión y además es utilizado para mostrar que un enlace está deshabilitado.
  - Referente a los enlaces del componente es posible también medir el número de enlaces rotos de un sitio web (cuantificado en torno a la cantidad de enlaces que conllevan la aparición de errores HTML con código 404) con respecto al número de enlaces totales de dicho componente web.

- Las imágenes presentes en el componente deben presentar texto alternativo en caso de que dicha imagen no se pueda renderizar en un determinado navegador web o para que usuarios que tengan alguna discapacidad puedan utilizar alguna herramienta de lectura de pantalla para conocer cuál es el contenido que representa dicha imagen. Es posible medir el grado de imágenes con texto alternativo frente a las imágenes presentes en el componente que no dispongan de este atributo.

- Barra de búsqueda. La disposición en un determinado componente web de una barra de búsqueda es esencial para que usuarios que se sientan perdidos al navegar por el componente puedan encontrar la información que buscan de manera sencilla. En relación a este aspecto, se han determinado un conjunto de métricas de calidad, explicadas en detalle a continuación:
  - Se debe proveer a los usuarios de una barra de búsqueda en el componente, en contraposición al ofrecimiento de un enlace a un componente web que puede hacerse cargo de dicha búsqueda. Existen usuarios que demandan una barra de búsqueda en los componentes y páginas web, ya que si no asumen directamente que dicho componente no lleva consigo esta función implementada.
  - Dicha barra de búsqueda debe ser lo suficientemente grande (por lo menos 30 caracteres).
  - Si el componente web dispone de varias vistas web, la búsqueda en cada una de ellas debería buscar en todas las vistas web del componente.
- Diseño gráfico y animaciones. Al utilizar gráficos en un componente web, así como animaciones, dicho componente puede verse enormemente enriquecido. Por otro lado, los gráficos y las animaciones pueden llevar a tiempos muy altos de descarga del componente, por lo que es necesario que representen de manera gráfica contenido y no supongan un mero aditivo de estilo al componente. Se han determinado una serie de métricas de calidad relacionadas con este aspecto:
  - Como se ha comentado, los gráficos deben representar de manera más agradable contenido de un componente, pero no deben incluirse en un determinado componente web como una simple decoración.
  - Deben evitarse fondos de pantalla con marcas de agua, ya que disminuyen la visibilidad y no añaden en la mayoría de los casos ningún valor.
  - Elemento como el logo o eslogan del componente, críticos para el componente no deben ser animados, ya que muchos usuarios, al asociar las animaciones con anuncios, ignoran dichos elementos animados.
  - Se debe utilizar texto de alto contraste, al igual que colores de fondo para que sean lo más legibles posible.

- Un aspecto a tener en cuenta es el "scrolling" horizontal, muy molesto para muchos usuarios, los cuales además pueden perder de vista contenido importante del componente. Por lo tanto, debe evitarse su utilización en la medida de lo posible.

- Comunicación de fallos técnicos del componente y emergencias. En un determinado componente, a lo largo del tiempo, surgen fallos técnicos y emergencias que conllevan la inutilización de las páginas. Para indicar esta problemática, es posible utilizar diversas métricas de calidad para asegurarse de que los usuarios se encuentren informados de dichas incidencias:
  - Se debe incluir un mensaje explicativo del error que se ha producido, así como una fecha aproximada de resolución de la incidencia o un teléfono de atención al usuario.
  - Es posible también incluir una vista con HTML muy simplificado que indique el fallo técnico que se ha producido, con un enlace a la vista del componente para que, una vez solucionado el problema los usuarios del componente sean capaces de volver a esta vista de manera sencilla.

Para llevar a cabo la evaluación de la usabilidad, es posible la utilización de una métrica de calidad que evalúe el grado de cumplimiento de las directrices de usabilidad de cada una de las categorías descritas anteriormente por parte de los componentes web, a través de la siguiente fórmula:

metr_usabilidad (componente) = $\sum \frac{\textbf{directrices\_cumplidas}}{|\textbf{directrices}|}$ *100, donde

directrices_cumplidas define la cantidad de directrices de usabilidad, que son satisfechas por un componente determinado

directrices es el número total de las directrices de usabilidad definidas en el modelo

3.4.2. Accesibilidad.

A la hora de desarrollar un determinado componente web todos los usuarios [38], incluidos aquellos con discapacidades, deben ser tenidos en cuenta. Los usuarios, en especial aquellos con discapacidades y personas mayores, al acceder a una determinada página web deben ser capaces de percibir, entender, navegar e interactuar con su contenido. La accesibilidad web abarca todo tipo de discapacidades, tales como visuales, auditivas, motoras, físicas, cognitivas o neurológicas que afectan a millones de personas alrededor del mundo.

Las páginas y el software creado para la web tienen en su mayoría una gran cantidad de barreras de accesibilidad, que dificultan en gran medida o imposibilitan a usuarios con discapacidades o a personas ancianas el uso de la Web. Ésto ha ido cambiando a lo largo de los años y cada vez existe una mayor conciencia por la accesibilidad por parte de los desarrolladores web, lo que permite a las personas discapacitadas participar y contribuir de manera más activa para la web.

La accesibilidad no solamente afecta a las personas con discapacidades permanentes, también ayuda a personas con discapacidades temporales causadas por ejemplo por accidentes de tráfico o a personas mayores con dificultades para interactuar con la web debido a problemas de visión y dificultades motoras entre otras, así como a individuos con conexiones lentas a Internet.

Para llevar a cabo el desarrollo de aplicaciones web (en el caso que nos compete componentes web) accesibles para todo tipo de usuarios, es necesario que la accesibilidad sea tenida en mente a lo largo de todo ese proceso de desarrollo, para que sea más sencilla la evaluación de su usabilidad.

Las páginas y aplicaciones web, en concreto los componentes web que que están siendo objeto de este estudio, deberían desarrollarse siguiendo los estándares de guías de diseño de accesibilidad publicados por el W3C [37], en concreto dos de ellos proveídos por la WAI (W3C Web Accesibility Initiative) [39], el estándar WCAG (Web Content Accessibility Guidelines) 2.0 y el estándar WAI-ARIA (Web Accessibility Initiative for Accessible Rich Internet Applications), los cuales se explican a continuación:

- WCAG 2.0. [40] La WCAG 2.0 (Web Content Accessibility Guidelines) es un estándar desarrollado por la W3C en cooperación con organizaciones y personas alrededor del mundo con el objetivo de crear un estándar de accesibilidad para la Web que haga posible que el contenido de páginas y aplicaciones web sea accesible y cumpla con las necesidades expectativas de personas discapacitadas y organizaciones de forma internacional.

  El desarrollo de este estándar está concebido para desarrolladores de contenido web, herramientas de autorización web, herramientas de evaluación de la accesibilidad web, así como todo tipo de personas que quieran o necesiten un estándar de accesibilidad web de validez internacional.

  La versión 2.0 de este estándar es la sucesora de la versión 1.0 [41]. Se caracteriza por su estabilidad y por la utilización de 12 guías de diseño organizadas a lo largo de 4 principios (percepción, operabilidad, entendimiento y robustez). Para cada guía o pauta de accesibilidad existe una puntuación o un criterio de éxito, organizado en torno a tres niveles de menor a mayor puntuación: nivel A, nivel AA y nivel AAA.

  La W3C además ofrece referencias de diseño para que una determinada página o aplicación web pueda cumplir con las especificaciones de este estándar, así como técnicas para llevar a cabo la implementación de dicho estándar, así como tutoriales que explican más en detalle el estándar WCAG 2.0.

- WAI-ARIA 1.0. [42] La WAI-ARIA (Web Accessibility Initiative for Accessible Rich Internet Applications) es un estándar creado por la W3C que abarca como llevar a cabo el desarrollo de páginas y aplicaciones web cargadas de accesibilidad para su utilización por parte de personas discapacitadas, concretamente se encarga de hacer más accesible el contenido dinámico ofrecido por estas aplicaciones y desarrollado con tecnologías como AJAX (Asynchronous JavaScript And XML).

Este estándar aborda la accesibilidad de este contenido dinámico ofreciendo información acerca de éste a la tecnología asistida que utilizan las personas discapacitadas.

La WAI-ARIA 1.0 es un estándar que provee un framework que añade atributos a los elementos de las aplicaciones web relacionadas con su estado y con la interacción que ofrecen al usuario, así como nuevas técnicas de navegación (utilizando el teclado por ejemplo) a lo largo de estructuras o menús en lugar de componentes aislados organizados de manera jerárquica (como la estructura jerárquica ofrecida por el DOM del lenguaje HTML). En resumen, este estándar ofrece las siguientes características a los desarrolladores de aplicaciones web:

- Creación de roles de cada uno de los componentes de la aplicación como menús, árboles o pestañas entre otros.
- Roles para definir una estructura del contenido de la aplicación como cabeceras, regiones o tablas.
- Propiedades para describir el estado de los componentes de la aplicación web.
- Propiedades para definir el origen y el destino de eventos de drag-and-drop de componentes de la página web.

Para cuantificar si un determinado componente web (que se puede definir en esencia como una aplicación web en sí misma) es posible asignar una métrica de accesibilidad que tenga en cuenta el grado de cumplimiento en dicho componente de las directrices de accesibilidad pertenecientes a los dos estándares mencionados anteriormente.

Dicha métrica viene determinada por la siguiente fórmula:

metr_accesibilidad (componente) = nivel_WCAG + grado_WAI-ARIA , donde

nivel_WCAG define el nivel de accesibilidad de un componente según este estándar (niveles A, AA o AAA)

grado_WAI-ARIA es un valor que define el grado de cumplimiento de las directrices de accesibilidad especificadas para dicho estándar de la W3C por parte de un componente determinado

3.5. Calidad de las interacciones sociales de los usuarios finales con el componente.

La calidad de las interacciones sociales con los componentes web por parte de los usuarios se centra en las interacciones de éstos con los componentes web de la página a través de la reputación o valoración de estos últimos, las redes sociales más populares (Facebook, Google+ o Twitter), además de las interacciones con repositorios como Github, los cuales hospedan componentes web, en relación a los seguimientos y descargas de componentes a través de ellos por parte de estos usuarios.

Las características de grado más fino que se han tenido en cuenta para evaluar este aspecto de calidad son, como se ha comentado, la valoración de usuarios finales de los componentes de la aplicación, la popularidad en base a seguimientos realizados por los usuarios del repositorio del componente, cantidad de descargas, popularidad del componente en la red social de Facebook, en la red social de Google+ y en la de Twitter, las cuales se detallan a continuación:

3.5.1. Reputación.

La reputación mide en qué grado un componente web es concebido como confiable por parte de los usuarios finales que lo utilizan. En este aspecto, la credibilidad y la reputación obtenida por desarrolladores o instituciones que crean componentes web, ayuda en una gran parte a la difusión de dichos componentes.

Como podemos ver en el portal web de programmableweb.com, los componentes más difundidos y extendidos entre los usuarios de componentes web, son aquellos creados por instituciones o desarrolladores de una gran reputación entre dichos usuarios.

Desde el punto de vista del desarrollador del componente, tiene una gran importancia obtener un alto nivel de reputación, la cual puede ser conseguida gracias al éxito de los componentes que implementa.

Dicho éxito viene determinado por la funcionalidad del componente de cara al usuario final, pero también en gran parte a la documentación que se ofrece sobre información relevante de éste. Esta documentación puede ser distribuida a través de diferentes medios por Internet a través de blogs o foros entre otros y redactada siguiendo estándares de documentación, ayudando todo ésto a incrementar el nivel de reputación de un componente.

Un determinado componente web, en su especificación o en su ficha técnica por ejemplo, puede ofrecer un mecanismo de valoración, para que los usuarios puedan puntuar la calidad de un componente según unos determinados criterios. Esta valoración puede ser extrapolada a una métrica de calidad que se puede calcular a través de la media de las valoraciones de todos los usuarios finales, por medio de la siguiente fórmula:

$$\textbf{métrica\_valoracion (componente)} = \frac{\sum \textbf{valoracion\_usuarios}}{\textbf{| valoraciones |}}$$

valoración_usuarios determina la suma de valoraciones llevadas a cabo por usuarios finales del componente web

valoraciones se refiere al número total de valoraciones del componente realizadas por los usuarios finales de éste

3.5.2. Popularidad (seguimientos del repositorio).

Para cuantificar el número de seguimientos de un determinado componente web a través del repositorio que lo hospeda, obtendremos la información relacionada con dicha cantidad del repositorio con el objetivo de poder cuantificarla en relación a otros componentes que pueden encontrarse almacenados en dicho repositorio.

Mediante una clusterización de las cantidades de seguimientos de todos los componentes que estén siendo objeto de estudio, se clasificará un determinado componente en alguna de las clases resultantes de este proceso de clasificación.

Según esta clusterización, si un determinado componente se encuentra en una de las clases resultantes de dicho proceso de clusterización de mayor popularidad, este componente dispondrá de una buena calidad para esta métrica. En caso contrario, si un componente se dispone de una baja popularidad, dispondrá de una calidad para esta métrica calificada como "mala". De esta manera la métrica que clasifica la popularidad del repositorio de un componente quedará determinada de la siguiente manera:

metr_popularidad_repositorio (componente) = buena, regular, mala

3.5.3. Descargas (referentes al repositorio).

Con el objetivo de evaluar el número de descargas llevadas a cabo sobre un componente web a través del repositorio que lo hospeda, recopilaremos la información relacionada en este aspecto del repositorio con el objetivo de poder evaluarla en relación a otros componentes almacenados en dicho repositorio.

Al igual que en el caso de los seguimientos del repositorio de un componente web, mediante la creación de una clusterización del número de descargas de los componentes que estén siendo estudiados, se clasificará un determinado componente en alguna de las clases resultantes de dicho proceso de clasificación.

En relación a dicha clusterización, si un componente determinado se clasifica en una de las clases resultantes de dicho proceso de clusterización con una mayor cantidad de descargas, este componente dispondrá de una calidad para esta métrica evaluada como "buena".

Por el contrario, si un componente se dispone de un bajo número de descargas, su calidad asociada a esta métrica será "mala". La métrica que clasifica la calidad con respecto al número de descargas del componente desde su repositorio quedará definida de la siguiente manera:

metr_descargas_repositorio (componente) = buena, regular, mala

3.5.4. Popularidad en Facebook.

La cuantificación de la popularidad en la red social Facebook de un componente web podrá ser realizada a través de la medición de la cantidad de pulsaciones de un botón de "Me gusta" de ese componente, el cual permitirá que si un componente dispone de una página de Facebook con información sobre éste se sume un nuevo "like" por parte de un determinado usuario.

La cuantificación de esta métrica, al igual que las anteriores se realizará mediante un proceso de clustering análogo al de las métricas anteriores.

En relación a esta clusterización, para un componente que se clasifica en una de las clases resultantes de dicho proceso de clasificación con una mayor popularidad en la red social de Facebook, dicho componente tendrá asociada una calidad para esta métrica de "buena". En caso contrario, si un componente dispone de una baja popularidad en esta red social, su calidad será "mala" para esta métrica. La métrica que clasifica la calidad con respecto a la popularidad del componente en la red social de Facebook quedará definida de la siguiente manera:

metr_popularidad_fb (componente) = buena, regular, mala

3.5.5. Popularidad en Google+.

Al igual que en el caso anterior, si un componente web dispone de una página con documentación relevante sobre éste en la red social de Google+, recopilaremos información acerca de la cantidad de usuarios que han añadido a sus círculos/les ha gustado dicha página. La cuantificación de esta métrica de calidad se llevará a cabo mediante el mismo proceso de clusterización que en el caso de las métricas anteriores.

De manera análoga a la métrica de calificación de calidad de popularidad de la red social de Facebook, se ha determinado una métrica de calidad para la red social de Google+, que sigue el mismo procedimiento que la anterior:

metr_popularidad_g+ (componente) = buena, regular, mala

3.5.6. Popularidad en Twitter.

En un componente web, se evaluarán la cantidad de tweets que se hayan realizado en los que se haya mencionado a dicho componente web, recopilando la cantidad de usuarios que han publicado mensajes en relación a ese componente web.

Esta información se extrapolará a una métrica de calidad que, a través de un proceso de clusterización que incluirá a los componentes que estén siendo objeto de estudio (como puede ser el caso de los componentes que forman parte de un repositorio con información relevante acerca de las características de un conjunto de componentes web), podrá clasificar la calidad de dicho componente para esta métrica de calidad en cuestión.

Del mismo modo que la métrica de calificación de calidad de popularidad de las redes sociales de Facebook y Google+, se ha definido una métrica de calidad para la red social de Twitter, que sigue la misma metodología que las anteriores:

metr_popularidad_twitter (componente) = buena, regular, mala

3.6. Calidad del proveedor del componente y del hosting.

En lo referente a este aspecto de calidad, la calidad del proveedor y del hosting del componente se centra en la actuación del proveedor del componente y de los recursos y servicios web asociados a éste, los cuales son ofrecidos por el servicio de hosting que de soporte a dicho componente. Las características de grado fino que han sido consideradas para evaluar dicho aspecto de calidad son la mantenibilidad, disponibilidad, confiabilidad, rendimiento y seguridad, detalladas a continuación:

3.6.1. Mantenimiento.

La evaluación de la mantenibilidad de un determinado componente web hace referencia al grado de actualización y reparo de bugs por parte del desarrollador o desarrolladores del componente. Pueden utilizarse diversas métricas de calidad para la cuantificación de la mantenibilidad de un componente, entre las que se destacan dos de ellas:

- Número de bugs reparados del componente web en relación al número de bugs totales. Para llevar a cabo una medida del grado de mantenimiento de un componente web podremos evaluar el porcentaje de bugs que han sido reparados por parte del desarrollador en comparación al número de bugs totales del componente (a lo largo de todas las versiones de éste). Para ello, utilizaremos la siguiente fórmula:

$$\textbf{tasa\_resolucion\_bugs} = \frac{\textbf{bugs\_resueltos}}{\textbf{bugs\_totales}}$$

- Número de bugs resueltos por release del componente web con respecto al número de bugs totales de la release. Se puede realizar una medición del grado del porcentaje de bugs de un componente web que han sido reparados en comparación al número de bugs totales de una versión de un componente. Para ello, utilizaremos la siguiente fórmula:

$$\textbf{tasa\_resolucion\_bugs (por release)} = \frac{\textbf{bugs\_resueltos}}{\textbf{bugs\_totales}}$$

Como métrica de calidad de evaluación del mantenimiento del componente web podremos realizar una media de todas las tasas de resolución de bugs de cada release o versión de un determinado componente, a través de la siguiente fórmula:

$$\textbf{metr\_mantenibilidad (componente)} = \frac{\sum \textbf{tasa\_resolucion (release i)}}{|\textbf{releases}|}$$

tasa_resolucion (reléase i) define la cantidad de bugs resueltos en una reléase de un determinado componente

releases es el número total de versiones o releases del componente

3.6.2. Disponibilidad.

Esta característica de calidad aborda el grado de accesibilidad a los recursos y servicios web que ofrece el hosting que hospeda el componente. Los atributos tratados en este aspecto son la disponibilidad de los recursos y servicios web que ofrece el componente, además de la denegación de dichos servicios como consecuencia de la sobrecarga de tráfico del componente. En lo referente a la disponibilidad de recursos y servicios, ésta puede ser evaluada a través de las siguientes métricas:

- *Uptime* del servicio o período de tiempo de funcionamiento de los servicios que ofrece el componente.
- *Maintenance time* o ventana de tiempo que debe cumplirse entre una caída del funcionamiento del componente y la reparación de dicha incidencia.

En relación a la monitorización de la sobrecarga de tráfico en un componente éste, así como las APIs y servicios de terceros que utilice, podrán disponer de mecanismos para realizar un control de dicha sobrecarga, restringiendo el número de peticiones por IP (usualmente por día) a los servicios de un componente.

Dicha restricción, aunque puede ser vista como una gran limitación desde el punto de vista de los usuarios finales que utilicen el componente y de los desarrolladores de *mashups* que integrarán dicho componente en sus aplicaciones, ésta es necesaria para que el uso de los servicios del componente por parte de un número determinado de usuarios no sea abusivo y no prive al resto del acceso al componente y a su funcionalidad.

Es posible asociar una métrica de calidad a esta cuantificación de la disponibilidad de los servidores de un determinado componente, mediante el cálculo del grado en el que las peticiones de datos con éxito a dichos servidores se corresponden con el número de peticiones totales por día por parte de los usuarios. Todo ésto, quedará plasmado a través de la siguiente fórmula:

disponibilidad_srv (componente) = $\frac{\sum \textbf{petExito}}{\sum \textbf{petTotales}}$ * **disponSrv** **(por día)** , donde

petExito indica la cantidad de peticiones a servidores por día en las que el componente responde de manera exitosa

petTotales indica la cantidad total de peticiones a los servidores de las APIs que un componente utiliza por día

disponAPI hace referencia a la disponibilidad media que ofrezcan los servidores que ofrecen servicio al componente; si los proveedores de los servidores no ofrecen un valor de disponibilidad se asumirá un valor del 100% (un valor de 1 para el cálculo de esta métrica)

La evaluación de la disponibilidad de los servicios ofrecidos por un componente se llevará a cabo mediante una métrica que evalúa las características que se han comentado en esta sección, una vez que las características sean medidas según una misma escala uniforme, según los pesos definidos en la siguiente fórmula:

metr_disponibilidad_servidor (componente) = UP + MT + dispon_srv, donde

UP es el periodo de tiempo en el que el servidor del componente proporciona servicio

MT corresponde a la ventana de tiempo en la que se incluye el tiempo de caída por una incidencia hasta su reparación

dispon_srv mide el grado de disponibilidad de los servidores de un componente

3.6.3. Rendimiento.

Esta característica de calidad de rendimiento evalúa el grado en el que los tiempos de acceso a los servidores que dan soporte al componente y a los servicios y recursos que ofrece superan el tiempo mínimo de sesión de la RIA en la que participa el componente web. Para que un componente tenga una valoración de calidad en lo que respecta a esta métrica, el tiempo de acceso a los servidores debe ser menor al período de tiempo mínimo de sesión. Para ello, cuantificaremos esta métrica de calidad mediante el siguiente criterio:

- Si el tiempo de acceso a los servidores que dan servicio a un componente es menor que el tiempo mínimo de sesión el componente dispondrá de una buena calidad en lo que respecta a esta métrica.

- Si el tiempo mínimo de sesión y el tiempo de acceso a los servidores de un componente es equivalente o casi igual el componente obtendrá un valor "regular" de calidad en lo que a esta métrica se refiere.
- Si el componente tiene dispone de un tiempo medio de acceso a sus servidores mayor que el tiempo mínimo de sesión de la RIA en la que se integra el componente, este último dispondrá referentemente a esta métrica de un valor de mala calidad.

La métrica de calidad de rendimiento quedará definidad de la siguiente manera:

metr_rendimiento_srv (componente) = buena, regular, mala

3.6.4. Seguridad.

La seguridad de los servidores que ofrecen los recursos y servicios web que provee el componente, evalúa si dichos servidores disponen de certificados o firmas digitales válidos y de confianza para el cliente, siguiendo los estándares de seguridad en este ámbito.

Para llevar a cabo la evaluación de dicha seguridad se puede utilizar una métrica de calidad que cuantifique si esos certificados han sido emitidos por entidades públicas certificadoras o si las firmas digitales utilizan mecanismos de criptografía de clave asimétrica que implementan protocolos estándares o ampliamente utilizados. Esta métrica viene definida por la siguiente fórmula:

metr_seguridad_servidores = certificado_valido + protocolo_firma_digital , donde

certificado_valido es un valor booleano que indica si un componente utiliza un certificado digital emitido por una entidad pública certificadora

protocolo_firma_digital define mediante un valor booleano si un componente utiliza un protocolo de firma digital de confianza como RSA o DSA

La seguridad de los servidores que proveen servicio a un componente determinado depende de si dichos servidores disponen de certificados válidos y utilizan protocolos de firma digital de confianza, o si por el contrario no los usan. Esto queda definido mediante la siguiente métrica de calidad:

metr_seguridad_servidores (componente) = buena, mala

3.6.5. Confiabilidad.

El grado de confiabilidad del hosting que hospeda el código fuente de un determinado componente web y los servicios y recursos que dicho componente ofrece, evalúa que esos servicios y recursos web viene determinado por el concepto de no repudio, el cual determina que en una comunicación de datos tanto el emisor como el receptor no pueden negar su intervención en ese proceso de comunicación.

Este aspecto de calidad puede ofrecerse por parte del componente web, pero también el emisor participa en esta confiabilidad, y está relacionada con la métrica de seguridad explicada en detalle en la anterior sección.

Esto se debe a que la confiabilidad de los datos puede darse si el servicio de no repudio se encuentra implementado mediante la utilización de firmas digitales o testigos generados por terceras partes seguras tanto en el servidor proveedor de servicios de componente como en el emisor, el cual es un usuario final que utiliza dicho componente.

Para medir la confiabilidad de un determinado componente, es posible utilizar una métrica de calidad, la cual viene determinada por la siguiente fórmula:

metr_confiabilidad_servidores = firma_digital U testigo_terceros , donde

firma_digital define mediante un valor booleano si un componente utiliza en sus servidores un protocolo de firma digital de confianza como RSA o DSA

testigo_terceros es un valor booleano que indica si un componente utiliza en sus servidores un testigo de confianza que utiliza algoritmos simétricos como AES emitido por una tercera parte segura

La confiabilidad de los servidores que proveen servicio a un determinado componente depende de si dichos servidores utilizan protocolos de firma digital de confianza o testigos de terceros emitidos por una entidad de confianza, o si por el contrario no lo hacen. Todo esto queda determinado según la siguiente métrica de calidad:

metr_confiabilidad_servidores (componente) = buena, mala

Capítulo 4.

Propuesta de un marco de caracterización de métricas de calidad para *mashups* de componentes web.

Índice

Una vez se ha definido un modelo de caracterización de calidad para componentes web mediante técnicas de caja negra, es necesario abordar la calidad de dichos componentes como potenciales elementos en la composición de *mashups*. Los componentes web pueden ser widgets con una interfaz de usuario, operadores sin interfaz visual que interactúan con aquellos componentes que sí la tienen, fuentes de datos o servicios web de terceros.

Dichos componentes toman una serie de roles según la importancia que tomen en la orquestación e integración de éstos en una aplicación de *mashup* [4], los cuales pueden afectar de manera diferente a la percepción del usuario en lo que respecta a su calidad final. Al analizar un conjunto de *mashups* del repositorio colaborativo de programmableweb.com [14], han sido tenidos en cuenta los siguientes roles que toman los componentes al combinarse con otros en una aplicación de *mashup*:

- Maestro: un componente que toma el rol de maestro es aquel con el que los usuarios interactuar al utilizar un *mashup* y el cuál sincroniza mediante eventos con información a aquellos que toman el rol de esclavo en dicha aplicación.
- Esclavo: los componentes esclavos dependen de otro componente maestro y se sincronizan con los eventos generados por este último. La información que se presenta en estos componentes se sincroniza con aquella proveniente de la propagada por el componente que toma el rol de maestro, con el cual el usuario final de la aplicación interactúa.

- Filtro: los componentes que actúan como filtros en *mashups* permiten a los usuarios cribar y seleccionar los datos que éstos deseen para mostrarse en otros componentes, principalmente los componentes con rol de maestro.

Una vez quedan definidos los roles que toman los componentes en las aplicaciones web de *mashup*, se pueden identificar tres patrones para el desarrollo de *mashups*, el patrón esclavo-esclavo, el patrón maestro-esclavo y el patrón maestro-maestro, detallados a continuación [3]:

- En lo que respecta al patrón **esclavo-esclavo**, un usuario interactúa con componentes esclavos de manera independiente, sin que exista ninguna sincronización de información entre los componentes del *mashup*.
- El patrón **maestro-esclavo**, el cual es el patrón de desarrollo de *mashups* más utilizado, permite a los usuarios interactuar con un componente maestro. Una vez un usuario interactúe con dicho componente maestro y seleccione la información que desee mediante un componente de filtro, los otros componentes esclavos a través de la lógica de integración del *mashup*, se sincronizarán con dicha información.
- El patrón de desarrollo **maestro-maestro** es el más completo de los tres, ya que todos los componentes que se integran en un *mashup* que siga este patrón de desarrollo toman el rol tanto de maestro como de esclavo, ya que al interactuar y seleccionar datos en un determinado componente el resto se sincroniza con dicha información.

4.1. Diseño del modelo de calidad para mashups de componentes web.

En la sección anterior se ha definido la calidad de los *mashups* no como una simple agregación de la calidad individual de los componentes web que lo forman, sino como una composición de éstos en torno a la integración de datos, funcionalidades y experiencia de usuario de cada uno de ellos.

Las dimensiones y propiedades de los modelos de calidad web tradicionales se encargan de cuantificar la calidad en uso de las aplicaciones web (usabilidad y accesibilidad), pero no son adecuadas para evaluar esta calidad de composición de los *mashups*. Por este motivo se propone un modelo de calidad que se apoya en los modelo de calidad web tradicionales, con características de calidad refinadas para abordar la calidad de las aplicaciones web de mashups.

El modelo de calidad propuesto, dispone de una jerarquía organizada según unos ciertos niveles de importancia de calidad. Dichos niveles se clasifican, de mayor a menor importancia de calidad en este modelo, en calidad global del mashup, factores, criterios y métricas de calidad. La calidad global del componente se encuentra enfocada en torno a cuatro dimensiones de calidad principales, las cuales a su vez se dividen en subcaracterísticas de granularidad más fina y estas últimas serán cuantificadas en base a métricas de calidad donde sea posible: calidad de los datos, de composición del mashup, de presentación y de impacto social.

En el modelo de la siguiente figura quedan definidas estas cuatro características del modelo de calidad para aplicaciones web de *mashup*:

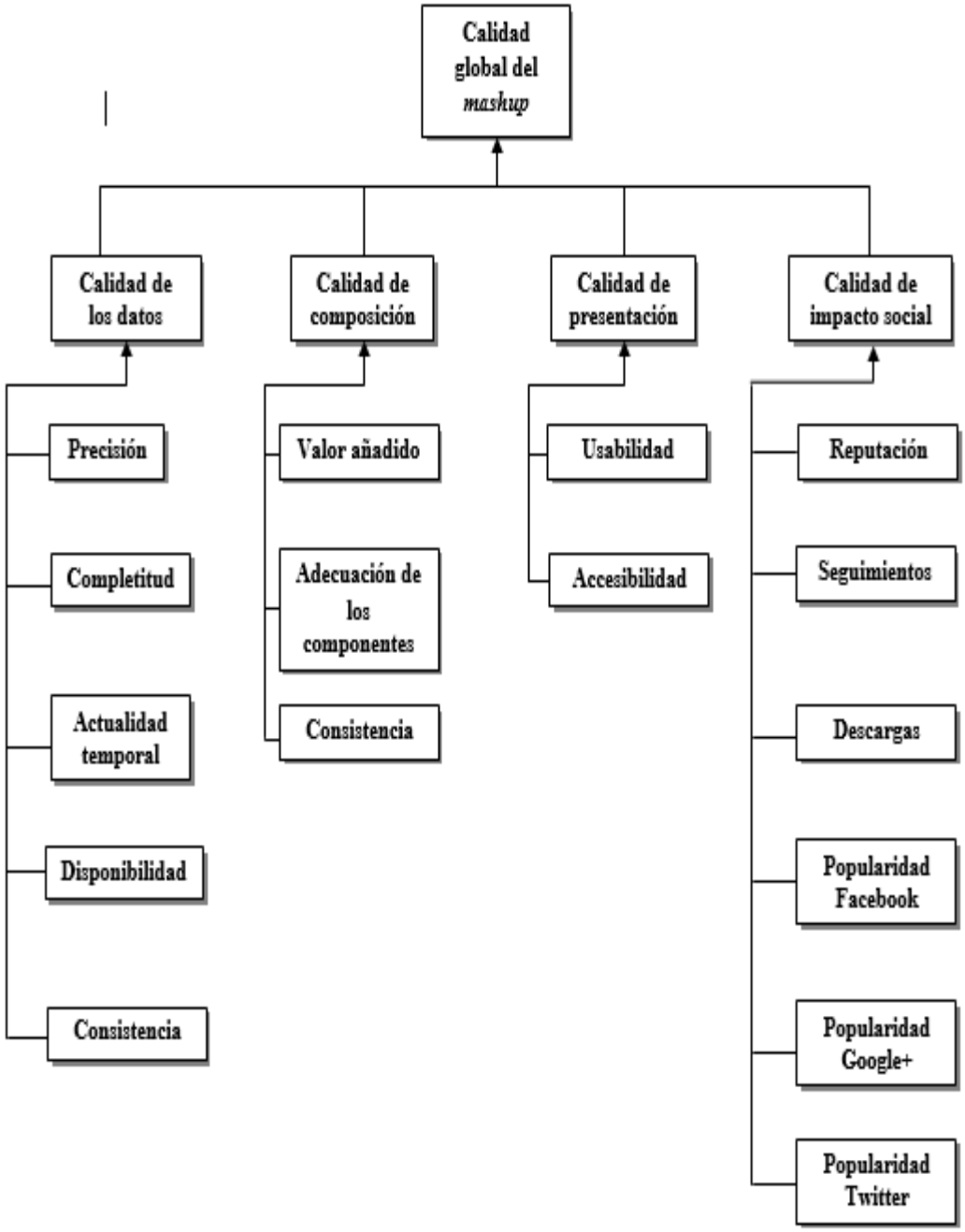

Figura 3: Modelo de calidad para *mashups* de componentes web

En referencia a los factores, criterios y métricas de calidad del modelo para componentes web, quedarán especificados en la siguiente tabla y explicados en las siguientes secciones de manera detallada:

Factores	**Criterios**	**Descripción**	**Métricas**
Calidad de los datos	Precisión	Probabilidad de que el funcionamiento con los datos que proveen los componentes de la composición es correcto.	metr_precision (*mashup*)$_{m\text{-}e}$ = 1 – (p(de_m) + p(de_s | dc_m)) metr_precision (*mashup*)$_{m\text{-}m}$ = 1 – [α(p(de_{m1}) + p(de_{m2}| dc_{m1})) + (1 – α) (p(de_{m2}) + p(de_{m1} | dc_{m2}))]
	Completitud	Grado de completitud en la implementación requerida para los componentes del *mashup*.	metr_completitud_situacional (*mashup*)$_{m\text{-}e}$ = |CDSm|+|CDSe^ CDSm| * 100 / CDI metr_completitud_situacional (*mashup*)$_{m\text{-}e}$ = |CDSm| * 100 / CDI
	Actualidad temporal	Grado de frescura de los datos.	metr_frescura_datos (*mashup*) = f_agregacion($fresc_1$ + $fresc_2$ + … + $fresc_n$)
	Disponibilidad	Probabilidad de que el *mashup* pueda proveer datos	metr_disponibilidad (mashup) = 1 – π_k (1 - $disp\_cw_k$)
	Consistencia	Grado de consistencia entre los datos que producen los componentes del *mashup*.	buena, mala
Calidad de la composición	Valor añadido	Consecución de funcionalidad adicional al integrar componentes	buena, neutra
	Adecuación de los componentes	Ajuste al propósito del *mashup* de los componentes web que lo forman	buena, neutra
	Consistencia	Grado de consistencia en la orquestación de componentes	buena, mala
Calidad de presentación	Usabilidad	Usabilidad de la interfaz visual de los componentes del *mashup* en términos HCI.	(∑directrices_cumplidas / |directrices|) *100

	Accesibilidad	Accesibilidad visual a personas discapacitadas	nivel_WCAG2.0 + grado_WAI-ARIA1.0
Calidad de impacto social	Reputación	Valoración del *mashup* en su conjunto por parte de usuarios finales	∑valoración_componentes/ |componentes|
	Seguimientos	Popularidad del *mashup* relativa a seguimientos de éste por parte de los usuarios	buena, regular, mala
	Descargas	Popularidad del *mashup* relativa a las descargas	buena, regular, mala
	Popularidad en Facebook	Popularidad entre los usuarios de Facebook	buena, regular, mala
	Popularidad en Google+	Popularidad entre los usuarios de Google+	buena, regular, mala
	Popularidad en Twitter	Popularidad entre los usuarios de Twitter	buena, regular, mala

Figura 4: Tabla de factores, criterios y métricas de los *mashups* de componentes web

4.2. Calidad de los datos.

La calidad de los datos en aplicaciones web de *mashup* está enfocada en la orquestación e integración de los datos de cada uno de los componentes que forman dichas aplicaciones. Para evaluar este factor de calidad es necesario centrarse en los componentes que los forman, así como los roles que toman en su composición.

Desglosaremos este factor de calidad en una serie de criterios, los cuales serán evaluados según los patrones de desarrollo maestro-esclavo y maestro-maestro, debido a que el patrón esclavo-esclavo supone que la calidad del *mashup* es una simple agregación de las calidades individuales de los componentes que lo forman, como consecuencia de la nula integración entre éstos.

Las características de grado fino que se tienen en cuenta para representar este aspecto de calidad son la precisión, la completitud, la actualidad temporal y la disponibilidad, al igual que en el modelo de calidad para componentes web, añadiendo la característica de consistencia de los datos, imprescindible para cuantificar las inconsistencias de datos que pueden surgir al integrar componentes web en aplicaciones de *mashup* de usuario final. Estas características se verán detalladas a continuación [3]:

4.2.1. Precisión.

Como se ha explicado en el modelo de calidad para componentes web, es posible realizar una evaluación de la precisión de un determinado componente mediante el cálculo la probabilidad de que los datos sean correctos mediante la siguiente fórmula:

metr_precision (componente) = p(dc) , con

p(dc) = 1 – p(de) , donde

p(dc) es la probabilidad de que los datos sean correctos

p(de) es la probabilidad de que haya un error en los datos

Los errores que se pueden producir en los datos pueden deberse, como se ha comentado en el capítulo anterior, a representaciones de datos erróneas, errores tipográficos o actualizaciones de datos inexistentes.

En lo que respecta a la medición de la precisión de los datos de una aplicación de *mashup* podemos utilizar, al igual que en el modelo de componentes web, una métrica para cuantificarla, la cual se calculará de manera diferente si dicho *mashup* sigue el patrón maestro-esclavo o maestro-maestro:

- Para el primer caso, el error puede producirse tanto en el componente maestro como en el componente esclavo. Como el componente que toma el rol de esclavo depende de los datos que le provea el componente maestro, la probabilidad del error en el componente esclavo irá determinada según la selección que se haya realizado en el componente maestro por un determinado usuario. Podremos definir una métrica de calidad de cálculo de la precisión del *mashup* para este patrón de desarrollo a través de la siguiente fórmula:

  $$\textbf{metr\_precision } (\textit{mashup})_{m\text{-}e} = 1 - (p(de_m) + p(de_s \mid dc_m))$$

- En lo referente al patrón maestro-maestro, éste puede verse como la unión de dos patrones maestro-esclavo y esclavo-maestro, debido al mecanismo de sincronización entre los componentes de un *mashup* desarrollado con este patrón. La métrica de calidad que mide la precisión de un *mashup* que sigue dicho patrón de desarrollo viene determinada por la siguiente fórmula:

  $$\textbf{metr\_precision } (\textit{mashup})_{m\text{-}m} = 1 - [\alpha(p(de_{m1}) + p(de_{m2} \mid dc_{m1})) + (1 - \alpha)(p(de_{m2}) + p(de_{m1} \mid dc_{m2}))]$$

4.2.2. Completitud.

Anteriormente, en el capítulo de especificación de las características del modelo de calidad para componentes web, se determinó que la completitud de los datos de un componente viene definida según el grado en el que dicho componente implementa la funcionalidad especificada en sus requisitos. Para cumplirlos, es necesario que dicho componente sea capaz de proveer un subconjunto del conjunto total de datos al usuario para que éste lo pueda utilizar.

Desde un punto de vista teórico, la intención del desarrollador de un determinado componente es la de proveer un conjunto de datos completo o ideal (CDI), pero desde una perspectiva práctica solamente un subconjunto de este CDI es necesario para cumplir con los requisitos de un componente (denominado conjunto de datos situacional o CDS). Debido a ésto, la completitud de los datos de un componente se mide a través de una completitud situacional de los datos, a través de la siguiente fórmula:

metr_completitud_componente_situacional (componente) = CDS *100 , donde
CDI

CDS es el conjunto de datos situacional del componente

CDI es el conjunto de datos ideal del componente

Como en el caso de la precisión, se tendrán en cuenta los patrones de desarrollo maestro-esclavo y maestro-maestro para definir la completitud de *mashup* que sigan dichos patrones:

- En el caso del patrón maestro-esclavo se tomarán el conjunto de cardinalidades CDR (conjunto de datos real) formado por la razón entre la suma de las cardinalidades del conjunto situacional del componente maestro y la unión de las cardinalidades de los CDS de todos los componentes que forman la aplicación y el conjunto de datos ideal del *mashup*. Esto se puede cuantificar mediante la siguiente fórmula:

$$\text{metr\_completitud\_situacional } (mashup)_{m\text{-}e} = \frac{|CDSm| + |CDSe \wedge CDSm|}{CDI} * 100$$

- En lo referente al patrón de desarrollo maestro-maestro, la cardinalidad del CDR surgirá de la razón de la suma de las cardinalidades de dos conjuntos de datos situacionales de componentes maestros y el conjunto de datos ideal de la aplicación de *mashup*. Esto se llevará cabo a través de la siguiente fórmula:

$$\text{metr\_completitud\_situacional } (mashup)_{m\text{-}e} = \frac{|CDSm|}{CDI} * 100$$

Existe un caso no cubierto por la completitud situacional, y es aquel en el que no existan enlaces entre componentes maestros en el patrón maestro-maestro o enlaces de un componente esclavo a uno maestro en el patrón maestro-esclavo. La completitud composicional (CC) es un concepto que cubre dicha problemática y se define de la siguiente manera para los dos patrones de desarrollo anteriormente mencionados:

- Para el patrón maestro-esclavo, se mide como la suma de las cardinalidades de los componentes de un *mashup* en relación a la cardinalidad del conjunto de datos situacional del componente que toma el rol de maestro, a través de una métrica de calidad que utiliza la siguiente fórmula:

$$\text{metr\_completitud\_composicional } (mashup)_{m\text{-}e} = \frac{|CDSm \wedge CDSe|}{|CDSm|}$$

- En el caso del patrón de desarrollo maestro-maestro, se utiliza una métrica de calidad para la evaluación de la completitud composicional de los *mashups* que siguen este patrón de desarrollo, a través de la siguiente fórmula:

$$\text{metr\_completitud\_composicional } (mashup)_{m\text{-}e} = \alpha \frac{|CDSm1 \wedge CDSm2|}{|CDSm1|} + (1-\alpha) \frac{|CDSm2 \wedge CDSm1|}{|CDSm2}$$

4.2.3. Actualidad temporal.

La actualidad temporal de los datos de un determinado componente ha sido definida en el modelo de calidad para componentes web por su grado de actualización temporal de cara a los usuarios. Para llevar a cabo una evaluación de la frescura o actualidad temporal de estos datos se ha determinado el uso de una métrica de calidad cuantificada a través de la siguiente fórmula:

metr_frescura_datos (componente) = max (0,1 – (actualidad_datos / volatilidad_datos))^{s}, donde s es el exponente que controla la sensibilidad de esta actualidad de los datos.

En relación a la actualidad temporal de los datos de un determinado *mashup*, ésta puede ser evaluada mediante una métrica de calidad expresada como una agregación de los valores de frescura de cada uno de los componentes que lo forman, de la siguiente manera:

metr_frescura_datos (*mashup*) = f_agregacion($fresc_1$ + $fresc_2$ + … + $fresc_n$) , donde

f_agregacion puede ser el mínimo, la media o el máximo de las actualidades temporales de los componentes que forman parte de un *mashup*

Al añadir esta función de agregación, la actualidad de los datos de un determinado *mashup* deja de ser dependiente del patrón de desarrollo de *mashups* y pasa a depender directamente del rol que toma el tiempo en el dominio de aplicación.

De esta manera, si el tiempo adquiere mucha importancia en el dominio en el que se crea una aplicación de *mashup* se utilizará una función de agregación del mínimo, por ejemplo para el caso de un *mashup* relacionado por ejemplo con tiempo atmosférico; en caso contrario, se utilizará un función de agregación del máximo si nos encontramos con un portal de noticias de diferentes periódicos deportivos en formato electrónico.

Por otro lado, si el tiempo no es un factor determinante para un determinado *mashup*, como por ejemplo una aplicación que permita localizar restaurantes en una determinada ciudad, podremos utilizar la función de agregación de la media.

4.2.4. Disponibilidad.

La disponibilidad de los datos de un componente web hace referencia a la probabilidad de que dicho componente pueda proveer un conjunto de datos, es decir, hace referencia a la probabilidad de que un determinado componente se encuentre disponible. La métrica de calidad asociada a la disponibilidad de un componente se determina mediante la siguiente fórmula:

metr_disponibilidad (componente) = $\frac{\sum \text{petExito}}{\sum \text{petTotales}}$ * disponAPI (por día) , donde

petExito indica la cantidad de peticiones de datos por día en las que un determinado componente responde con los datos que espera el usuario

petTotales indica la cantidad total de peticiones de datos diarias realizados por los usuarios que utilicen un componente

disponAPI hace referencia a la disponibilidad media que ofrezcan las APIs que provean datos al *mashup*

En relación a la disponibilidad de las aplicaciones web de *mashup*, ésta hace referencia a la probabilidad de que dicha aplicación sea capaz de proveer datos a los usuarios que la utilicen. Para ello, solamente es estrictamente necesario que uno sólo de los componentes que forman la aplicación de *mashup* se encuentre disponible para que dicha aplicación lo esté.

Para determinar la disponibilidad de un determinado *mashup* es posible utilizar una métrica de calidad, la cual viene determinada por la siguiente fórmula:

metr_disponibilidad (mashup) = $1 - \pi_k (1 - disp\_cw_k)$, donde

$disp\_cw_k$ es la disponibilidad de un determinado componente k que participa en la integración del *mashup*

4.2.5. Consistencia.

Este criterio de calidad surge de la integración de los datos de componentes web en aplicaciones de *mashup*. El modelo de calidad para *mashups* de componentes web supone que cada componente, de manera independiente, ofrece datos consistentes que no se contradicen entre ellos. Sin embargo, al integrarse dichos componentes en aplicaciones web de *mashup*, los conjuntos de datos pueden entrar en conflicto unos con otros y llevar a inconsistencias relacionadas de datos en la aplicación web final. En este tipo de aplicaciones, los desarrolladores de *mashups* no son expertos en el dominio en el que se mueve la aplicación, lo que lleva a inconsistencias que surgen en el *mashup* en tiempo de ejecución.

Se ha determinado una métrica de calidad que evalúa si existen inconsistencias en una aplicación de *mashup*. La calidad de un determinado *mashup* en relación a inconsistencias entre los componentes que lo forman será "buena" si no existen dichas inconsistencias y "mala" si éstas aparecen en el *mashup* objeto de evaluación. Todo esto queda definido según la siguiente métrica de calidad:

metr_consistencia = buena, mala

4.3. Calidad de composición.

La característica de calidad de composición de un determinado *mashup* evalúa cómo se produce la orquestación e integración de los componentes que lo forman y su adecuación a las funcionalidades para las que dicha aplicación fue creada.

Para la evaluación de cada uno de los criterios de calidad en los que se desgrana el factor de calidad de composición de un *mashup*, se tomará *feedback* de los usuarios finales que utilicen dicha aplicación mediante preguntas una vez cargado en el navegador dicho *mashup* o cuando los usuarios completen una tarea determinada, cuestiones relacionadas con dichos criterios de calidad, los cuales son tres: el valor añadido, la adecuación de los componentes a ese *mashup* y la consistencia de la orquestación de los componentes, los cuales se explican a continuación:

4.3.1. Valor añadido.

El atributo de calidad de valor añadido está relacionado con la cantidad de características y funcionalidades que ofrece un *mashup* al integrar en su desarrollo un conjunto de componentes. Dicha aplicación es capaz de proveer un valor añadido si la cantidad de funcionalidades que provee es mayor que las ofrecidas por los componentes de forma independiente.

El valor añadido de un *mashup* puede ser cuantificado en base al grado de calidad de integración de los componentes que lo forman. Existen *mashups* que ofrecen un conjunto de componentes aislados y unidos en una aplicación, pero con una pobre integración entre ellos, por lo que el valor añadido que se deducirá de estas aplicaciones será muy bajo. Por otro lado, existe un mayor valor añadido en aplicaciones en las que los componentes están integrados y sincronizados.

Las cuestiones que se les preguntarán a los usuarios con respecto al valor añadido que ofrece el *mashup* objeto de estudio, serán acerca de las funcionalidades que provee un *mashup*. Si dichas funcionalidades que nos han indicado los usuarios finales que utilizan el *mashup* sobrepasan aquellas propias de los componentes que se integran en dicho *mashup*, la aplicación ofrecerá un valor añadido al orquestar un conjunto de componentes y por tanto una buena calidad en lo que a valor añadido se refiere. Si la funcionalidad ofrecida por el *mashup* es la suma de las funcionalidades de los componentes que la integran, la calidad referente al valor añadido será neutra. Esto quedará reflejado con la siguiente métrica de calidad:

metr_valor_añadido (*mashup*) = buena, neutra

4.3.2. Adecuación de los componentes.

Este criterio de calidad mide el grado en el que las características de los componentes de un *mashup* determinado se ajustan al propósito para el cual fue creada dicha aplicación.

Las preguntas que se les realizarán a los usuarios acerca de la adecuación de los componentes que forman un *mashup* a éste, serán acerca de los objetivos conseguidos al realizar tareas específicas. Si dichos objetivos conseguidos se corresponden con aquellos para los que los componentes del *mashup* fueron concebidos y desarrollados, la aplicación ofrecerá una buena calidad en lo que a adecuación de componentes concierne. Si los objetivos para los que fue creado el *mashup* no se cumplen, la calidad referente a la adecuación de componentes será mala. Esto quedará reflejado con la siguiente métrica de calidad:

metr_adecuacion (*mashup*) = buena, neutra

4.3.3. Consistencia.

La baja calidad de orquestación en algunas aplicaciones de *mashup* puede llevar a inconsistencias entre los componentes que los forman. Estas inconsistencias pueden darse de la siguiente manera: en el ámbito de la interoperabilidad de los componentes de un *mashup*, dos componentes integrados pueden tener una salida y una entrada respectivamente compatibles en lo que a tipo de datos se refiere, pero puede darse el caso en el que la salida de un componente provea una información errónea en un formato compatible con la entrada del otro componente, lo que llevaría a una inconsistencia de orquestación de componentes para una aplicación de *mashup*.

Con respecto a la medición de la consistencia de los componentes que forman un determinado *mashup* se realizarán preguntas a los usuarios finales referentes a este criterio de calidad, acerca de que si al completar las tareas llevadas a cabo en dicha aplicación, han obtenido los resultados esperados.

Si es así, la aplicación ofrecerá una buena calidad en lo que a consistencia se refiere. En caso contrario, la calidad referente al criterio de consistencia será mala. Esto quedará reflejado mediante la siguiente métrica de calidad:

metr_consistencia (*mashup*) = buena, mala

4.4. Calidad de presentación.

Para esta característica de calidad se tiene en cuenta, al igual que para los componentes web, la usabilidad en términos de HCI y la accesibilidad, que vuelven a ser las características de granularidad más fina a evaluar en relación a la calidad de presentación:

4.4.1. Usabilidad.

La usabilidad de los *mashups* debe ser evaluada en torno a varios aspectos como el aprendizaje, la predecibilidad, la eficiencia, la memorabilidad, la consistencia del layout y la satisfacción de los usuarios que utilizan dicha aplicación.

La usabilidad para componentes web, como se ha comentado en el modelo de calidad para componentes especificado en el anterior capítulo, debe ser tenida en cuenta también para las aplicaciones de *mashup*.

Como dichos componentes son en esencia aplicaciones web, las métricas de usabilidad a tener en cuenta serán las mismas que se utilizan para cuantificar la usabilidad de las aplicaciones y páginas web tradicionales.

Como consecuencia de la integración de componentes web *mashups*, otras dimensiones de calidad diferentes a las evaluadas para componentes deben tomarse en consideración a la hora de cuantificar la usabilidad de *mashups*, como por ejemplo la predecibilidad o la anteriormente mencionada consistencia del layout de la aplicación. En relación a este atributo de calidad de la consistencia del layout del *mashup*, este provee una capa que hace de pegamento en la integración del conjunto de componentes que lo forman.

Otra dimensión de calidad a tener en cuenta en la medición de la usabilidad es el aprendizaje, ya que a primera vista, las características que ofrece un determinado *mashup* deben ser lo suficientemente visibles y autoexplicativos para que todo tipo de usuarios, incluidos aquellos sin apenas conocimientos técnicos de programación puedan ser capaces de interactuar con una determinada aplicación de *mashup.*

Si tenemos en cuenta algunas dimensiones de medición de la usabilidad de las páginas y aplicaciones web tradicionales, éstas no son adecuadas para la cuantificación de la usabilidad de los *mashups* de integración de componentes web. Criterios como la navegabilidad, las características de los enlaces no tienen un gran impacto en la calidad final del *mashup*, al igual que la legibilidad y coherencia de los textos, debido a que en las aplicaciones de *mashup* no contienen una gran cantidad de texto dentro de los componentes que integra.

Para llevar a cabo la evaluación de la usabilidad de un determinado *mashup* se seguirán las directrices o guías de usabilidad utilizadas por las páginas y aplicaciones web a lo largo de los años, acuñadas por expertos en usabilidad web de entidades como la Nielsen Norman Group [36] y la W3C [35].

Es posible la utilización de una métrica de calidad que evalúe el grado de cumplimiento de las directrices de usabilidad mencionadas anteriormente mediante la siguiente fórmula:

$$\textbf{metr\_usabilidad (mashup)} = \frac{\sum \textbf{directrices\_cumplidas} * 100}{|\ \textbf{directrices}\ |} \quad \text{, donde}$$

directrices_cumplidas define la cantidad de directrices de usabilidad que son satisfechas por un *mashup* determinado

directrices es el número total de las directrices de usabilidad que se tienen en cuenta para este modelo de métricas de calidad para *mashups*

4.4.2. Accesibilidad.

La accesibilidad en *mashups* debe ser abordada de manera similar a la evaluación de la usabilidad de las páginas web tradicionales, por lo que para llevar a cabo el desarrollo de aplicaciones web de *mashup* accesibles para todo tipo de usuarios y especialmente para aquellos que sufren algún tipo de discapacidad, es necesario seguir los estándares de guías de diseño de accesibilidad publicados por la WAI (W3C Web Accesibility Initiative) [36], los cuales son el estándar WCAG (Web Content Accessibility Guidelines) 2.0 [38] y el estándar WAI-ARIA (Web Accessibility Initiative for Accessible Rich Internet Applications) 1.0 [39].

Además del seguimiento de estas pautas de diseño estándares, es necesario tener en cuenta la accesibilidad a la hora de desarrollar el layout de la aplicación de *mashup* y en la integración de los componentes en el *mashup*, ya que pueden corregirse problemas de accesibilidad propios de alguno de los componentes web que componen un *mashup* o detectarse nuevos problemas de accesibilidad en la interfaz de usuario la aplicación resultante de su integración.

Para cuantificar si una aplicación web de *mashup* es accesible, se puede asignar una métrica que evalúe la accesibilidad de dicho *mashup* como aplicación web, cuantificando el grado de cumplimiento de las directrices de accesibilidad pertenecientes a los dos estándares mencionados anteriormente. Dicha métrica viene determinada por la siguiente fórmula:

metr_accesibilidad (*mashup*) = nivel_WCAG + grado_WAI-ARIA , donde

nivel_WCAG define el nivel de accesibilidad obtenido por la aplicación de *mashup* según este estándar (niveles A, AA o AAA)

grado_WAI-ARIA es un valor que define el grado en el que un *mashup* cumple con las directrices de accesibilidad especificadas en dicho estándar de la W3C

4.5. Calidad de las interacciones sociales de usuarios finales con el *mashup*.

Esta característica de calidad se centra en el impacto social de las aplicaciones de *mashup* de usuario final, es decir, la reputación de *mashups* por parte de usuarios finales, las interacciones con las redes sociales más populares (Facebook, Google+ o Twitter) y con repositorios web de *mashups* colaborativos (programmableWeb, Github) relacionados con estas aplicaciones por parte de los usuarios que las utilizan.

Las características de grano más fino que se han tenido en cuenta para evaluar este aspecto de calidad de *mashups* son la popularidad en base a seguimientos realizados por los usuarios del *mashup* en un determinado repositorio colaborativo, popularidad del componente en la red social de Facebook, en la red social de Google+ y en la de Twitter, las cuales se detallan a continuación:

4.5.1. Reputación.

La reputación de un determinado *mashup* mide la confiabilidad de los usuarios finales que lo utilizan. Relacionado con este aspecto, la credibilidad y la reputación obtenida por desarrolladores de los llamados *web mashups* ayuda en gran parte a su difusión entre los usuarios.

Como podemos observar en el portal web de programmableweb.com, los *mashups* más extendidos son aquellos creados por instituciones o desarrolladores con gran reputación entre dichos usuarios. Este éxito de difusión viene determinado tanto por la funcionalidad del componente de cara al usuario final como por la calidad final de éste

Es posible ofrecer un mecanismo de valoración de un *mashup* a los usuarios que lo utilicen para que éstos puedan puntuar la calidad de dicha aplicación según unos determinados criterios. Esta valoración puede ser extrapolada a una métrica de calidad que se puede calcular a través de la media de las valoraciones de todos los componentes web que integran una aplicación de *mashup*, por medio de la siguiente fórmula:

$$\textbf{metr\_valoracion\_mashup} = \frac{\sum \textbf{valoracion\_componente\_k}}{|\textbf{componentes\_mashup}|}$$

valoración_componente_k determina la suma de las medias de las valoraciones de los componentes web que forman parte de un *mashup*, llevadas a cabo por usuarios finales

componentes_mashup se refiere a la cantidad de componentes que componen dicho *mashup*

4.5.2. Popularidad (seguimientos del repositorio).

Llevar a cabo la evaluación de la cantidad de seguimientos de un determinado *mashup* a través de un repositorio colaborativo con información técnica y social de éste, obtendremos la información relacionada con dicha cantidad del repositorio llevar a cabo una evaluación de esta popularidad en relación a otras aplicaciones de *mashup* cuyas características pueden encontrarse almacenadas en dicho repositorio.

Mediante una clusterización de las cantidades de seguimientos de todos los *mashups* que estén siendo objeto de estudio, se clasificará una determinada aplicación en alguna de las clases resultantes de este proceso de clasificación.

Según esta clusterización, si un *mashup* se clasifica en una de las clases resultantes de dicho proceso de clasificación de mayor popularidad en un determinado repositorio, este componente dispondrá de una buena calidad para esta métrica. Por el contrario, si un componente lleva asociada una baja popularidad, dispondrá de una calidad "mala" para esta métrica. De esta manera, la métrica que clasifica la popularidad en un repositorio de un *mashup* quedará determinada de la siguiente manera:

metr_popularidad_repositorio (*mashup*) = buena, regular, mala

4.5.3. Descargas.

Con el objetivo de cuantificar el número de descargas llevadas a cabo sobre un *mashup* web determinado a través de un repositorio que lo hospede, recopilaremos la información relacionada en este aspecto de dicho repositorio con el objetivo de poder evaluarla en relación a otros *mashup* almacenados en el repositorio.

Al igual que en el caso de los seguimientos de una aplicación web de *mashup*, mediante la creación de una clusterización del número de descargas de dichas aplicaciones que estén siendo objeto de estudio se clasificará ese *mashup* en alguna de las clases resultantes de dicho proceso de clasificación.

Acorde a este proceso de clasificación, en el caso en el que un *mashup* se clasifique en una de las clases resultantes de dicho proceso de clusterización con un gran número de descargas en un repositorio determinado, este componente llevará asociado una calidad calificada como "buena" en relación a esta métrica. Si un componente lleva asociado un bajo número de descargas de un repositorio que lo aloja, dispondrá de una mala calidad para esta métrica de calidad. De esta manera, la métrica que clasifica la cantidad de descargas desde un repositorio determinado para un *mashup* quedará determinado de la siguiente manera:

metr_descargas_repositorio (*mashup*) = buena, regular, mala

4.5.4. Popularidad en Facebook.

La evaluación de la popularidad de una aplicación de *mashup* en la red social Facebook podrá ser llevada a cabo a través de la medición de la cantidad de pulsaciones de un botón de "Me gusta" en la página de dicha aplicación en esta red social, medición que será cuantificada y extrapolada a una métrica de calidad, al igual que en el caso anterior de la cuantificación de los seguimientos de un *mashup*. Esta métrica se evaluará según un proceso de clusterización análogo al proceso utilizado en el caso anterior.

En relación a esta clusterización, para un *mashup* que se clasifica en una de las clases resultantes de dicho proceso de clasificación con una mayor popularidad en la red social de Facebook, este *mashup* llevará asociada una buena calidad para esta métrica de calidad.

Por el contrario, si un *mashup* dispone de una baja popularidad en esta red social, su calidad será "mala" para esta métrica que está siendo objeto de estudio. La métrica que eval la calidad con respecto a la popularidad del *mashup* en la red social de Facebook quedará definida de la siguiente manera:

metr_popularidad_fb (*mashup*) = buena, regular, mala

4.5.5. Popularidad en Google+.

Al igual que para el caso anterior, si un *mashup* dispone de una página con documentación de características relevantes sobre éste en la red social de Google+, se recopilará información acerca de la cantidad de usuarios que han añadido a sus círculos/les ha gustado dicha página. La evaluación de esta métrica de calidad se realizará a través del mismo proceso de clusterización que en el caso de las anteriores métricas.

Del mismo modo que la métrica de evaluación de calidad referente a la popularidad del *mashup* en la red social de Facebook, se ha determinado una métrica de calidad para la evaluación de la popularidad del *mashup* para la red social de Google+, la cual sigue el mismo procedimiento que la anterior:

metr_popularidad_g+ (*mashup*) = buena, regular, mala

4.5.6. Popularidad en Twitter.

En una aplicación web de *mashup*, se cuantificarán la cantidad de tweets que hayan posteado usuarios en los que se haya mencionado a dicha aplicación web. Para ello, se recogerá la información en relación a la cantidad de usuarios que han publicado mensajes acerca de dicha aplicación.

Esta información se extrapolará a una métrica de calidad que, mediante un proceso de clusterización que incluirá a los web *mashups* que estén siendo objeto de estudio (como puede ser el caso de aquellos que forman parte de un repositorio con información relevante como programmableWeb acerca de las características de un enorme conjunto de aplicaciones web de *mashup*), podrá clasificar su calidad para este atributo de calidad en cuestión.

De manera análoga a las métricas de calificación de calidad para la popularidad referente a las redes sociales de Facebook y Google+, se ha definido una métrica de calidad para la red social de Twitter, que sigue el mismo procedimiento que las anteriores:

metr_popularidad_twitter (*mashup*) = buena, regular, mala

Capítulo 5.

Evaluación de las métricas y framework de anotado de medición de calidad.

Índice

El principal propósito del desarrollo de los modelos de medición de calidad para componentes web y aplicaciones de *mashup* es el de asignar un resumen de dicha calidad a cada uno de dichos componentes y aplicaciones, tanto en un ámbito global como en un alto nivel según las características definidas para cada modelo y en un bajo nivel según las subcaracterísticas en las que se desgrane cada categoría de métricas general.

Los datos de cada uno de estos componentes o aplicaciones deben ser recolectados para medir su calidad. Una vez se hayan recolectado estos datos, serán procesados para su posterior extrapolación a dichas métricas de calidad en una escala común y uniforme.

En dicha escala, se utilizará un criterio de medición matemático, el cual será el mismo para todas y cada una de las métricas de un determinado modelo según la característica de uniformidad anteriormente mencionada, por lo que se realizará una cuantificación de dichas métricas en una misma escala que permitirá asignar un resumen de la calidad a un componente o mashup web en todos los niveles de la jerarquía del modelo de calidad.

Por último, en este capítulo se expondrá el diseño de un framework de anotado de calidad, el cual a partir del análisis de la importancia relativa de las métricas de calidad en cada uno de los niveles de la jerarquía del modelo de calidad que corresponda podrá clasificar los pesos de importancia de cada uno de los elementos de dicho modelo con respecto a los elementos que se encuentran en su mismo nivel en la jerarquía.

Se detallará el uso para la determinación de los pesos de importancia de dichas métricas en los niveles alto y bajo de la jerarquía de técnicas de decisión multicriterio como el AHP (Analytic Hierarchy Process), muy utilizadas para la resolución de estos problemas en los que intervienen múltiples criterios de decisión, como es el caso de los modelos de calidad definidos.

Utilizando este framework de anotado de calidad podremos obtener un resumen de calidad de un determinado componente o aplicación de mashup de usuario final para asignárselo a éstos, tanto a un nivel global que caracterizará a un componente o *mashup* en una primera aproximación, como en un ámbito en el que se tiene en cuenta cada una de las características de dichos componentes y aplicaciones, debido a que una calidad por sí sola no conlleva mucha información acerca de la calidad de cada una de las características que están en un nivel más bajo que ésta.

5.1. Recolección de información de componentes y *mashups* web.

La recogida de datos acerca de los componentes web y *mashups* es imprescindible para la obtención de información relevante sobre ellos, información que se utilizará para la evaluación de las métricas de calidad definidas en los modelos de calidad definidos anteriormente.

Para llevar a cabo esta recolección de información pueden usarse diversas tecnologías. Una de ellas puede ser una base de datos tradicional que almacene y procese información relativa a componentes y *mashups*.

Por otro lado, otra alternativa que puede efectuar eficientemente este cometido es la utilización de servicios web de instituciones o empresas, como es el caso de Google Analytics, la cual genera estadísticas acerca de una enorme variedad de características, como tráfico de un sitio web o información de la interacción con éste de los usuarios que lo visitan, entre muchas otras.

Mediante estas tecnologías, es posible recolectar información acerca de un gran número de características relacionadas con los componentes web y las aplicaciones que surgen de la composición de dichos elementos, información que podremos encontrar en repositorios de información de componentes y *mashups* web colaborativos (como es el caso de programmableWeb) o en repositorios que hospeden el código fuente utilizado para implementarlos (como es el caso de la plataforma Github). Algunas de las características de las que podremos recoger información son las siguientes:

- Información acerca de los protocolos, lenguajes y formatos de datos utilizados en las comunicaciones entre componentes.
- Seguridad ofrecida por los servidores proveedores del servicio que ofrecen estos componentes.
- Información acerca de los desarrolladores que contribuyen a la creación de un componente o *mashup*.
- Información acerca del número de versiones de un componente y el tiempo transcurrido entre la subida de éstas.
- Analíticas del tiempo de ejecución requerido para hacer funcionar un componente web o un *mashup*.

- Evaluación de la usabilidad y accesibilidad ofrecidas por los componentes en relación a los estándares definidos en relación a estos aspectos.
- Número de seguimientos de un componente por parte de los usuarios en el repositorio en el que éste se encuentre hospedado.
- Interacción social de los usuarios con las funciones relacionadas con las redes sociales integradas en un determinado componente o *mashup*.
- Valoración de un componente por parte de un usuario final
- Bugs surgidos en un determinado componente por cada versión.

Es perfectamente factible utilizar en conjunto una base de datos y un servicio como Google Analytics. Para que esto fuera posible, en una página o aplicación web se debería desarrollar la implementación de la conexión en el lado servidor con una base de datos determinada, mientras que en el lado cliente (aunque también es posible su utilización en el lado servidor) se implementaría la conexión con la plataforma Analytics.

En referencia al servicio de Google Analytics, es posible la utilización de las APIs que ofrece para dos propósitos, el de recolección y posterior envío de datos sobre componentes web o *mashups* a la plataforma y el de consulta de los datos que existen en la plataforma para su procesamiento y extrapolación a métricas de calidad de componentes web para la evaluación de su calidad según los modelos definidos en los capítulos anteriores:

- Con respecto a la recogida y envío de datos a la plataforma Analytics es posible utilizar la Collection API que ofrece el servicio. Esta API puede utilizarse tanto en el lado cliente o front-end de una página o aplicación web como en el lado back-end o servidor de ésta.

  Es necesario que los componentes web se muestren en una aplicación web para poder recolectar información sobre éstos y almacenarla en Google Analytics. Para ello, como se ha explicado anteriormente, será necesario incluir en dicha aplicación web un código de seguimiento que permita generar información y estadísticas desde el lado cliente o el lado servidor para poder entregárselas a este servicio.
- En relación a la consulta de datos para el procesamiento de éstos y su posterior extrapolación a métricas de calidad de componentes web, se utilizará la API de Reporting para obtener datos de Google Analytics en formato de dimensiones y métricas, las cuales nos darán información relevante, relativa a componentes web y *mashups*.

  Esta información podrá ser procesada y extrapolada a métricas de calidad, cuyos valores podrán ser integrados en la medición de los modelos de calidad definidos, y aprovechados posteriormente por el *framework* de anotado para asignar el resumen de calidad a un determinado componente web o *mashup*.

En relación a la utilización de una base de datos, es factible la utilización de ésta para el almacenamiento de la información que hace referencia a las métricas de calidad, así como a todos los datos estadísticos relacionados con ellas.

Estos datos pueden hacer referencia a la información producida por su procesamiento siguiendo metodologías estadísticas directas como promedios o clusterizaciones de valores obtenidos al evaluar características de calidad de *mashups* y componentes web.

5.2. Evaluación de las métricas de calidad.

Una vez han quedado definidos los modelos de calidad y toda su estructura jerárquica de características, es necesario concertar una serie de métodos matemáticos de medición, ponderación y calificación de estas características y métricas de calidad de manera automática y no intrusiva.

Se llevará a cabo la definición una escala de medición de calidad de los componentes web y *mashups* uniforme que permitirá cuantificar la calidad de dichos elementos de manera más objetiva. Además, se expondrán diversos métodos y procedimientos de calificación de dichas métricas, como es el caso de la utilización de promedios y clusterizaciones para dicha medición.

5.2.1. Escala de medición de la calidad.

Una vez obtenidos los datos de un determinado componente o *mashup*, enviados, procesados y extrapolados, es necesaria la asignación de un valor numérico, siguiendo un criterio determinado. Esto apoyará la objetividad en la medición de la calidad de un determinado componente. La evaluación de los componentes web y *mashups* mediante técnicas de caja negra de manera objetiva, automática y no intrusiva conlleva a la definición de una escala numérica de medición de calidad.

En referencia a esta escala de medición, la característica principal de ésta es su uniformidad. Esta propiedad caracteriza a dicha escala y permite que todas y cada una de las métricas de calidad pertenecientes a un determinado modelo de calidad software puedan ser cuantificadas según un criterio común a cada una de ellas.

La consecuencia de este tipo de medición es la homogeneidad a través de la cual se califican las métricas de un determinado modelo, que aportará objetividad a este último, pudiendo centrarse la evaluación de un componente o de un *mashup* en los pesos de importancia relativa que tienen las métricas que se encuentran en un mismo nivel de jerarquía de un modelo determinado, para todos los niveles de dicha jerarquía.

Esta medición por pesos de las métricas de calidad se llevará a cabo como se ha comentado mediante la utilización del *framework* de anotado de calidad. Se utilizará una escala de medición simétrica, la cual permitirá asignar valores de calidad positivos y negativos en torno a un valor de calidad central o neutro. Este valor neutro podrá utilizarse, en un ámbito relacionado con las métricas de calidad como es nuestro caso, con la ausencia de calidad o de información de calidad relacionada con dichas métricas.

Para esta escala se ha dispuesto de un intervalo de medición del 0 al 10, con 0 como valor mínimo (y peor valor de calidad), 10 como valor máximo (y mejor valor de calidad) y 5 como valor de calidad neutra. Por tanto, la correspondencia de los valores de esta escala de medición con la escala de medición definida queda reflejada en la siguiente tabla:

Valores del intervalo de medición de la escala	Significado de calidad asociado
0	Extremadamente mala calidad
1	Pésima mala calidad
2	Muy mala calidad
3	Bastante mala calidad
4	Mala calidad
5	Ausencia de calidad
6	Buena calidad
7	Bastante buena calidad
8	Muy buena calidad
9	Estupenda calidad
10	Extremadamente buena calidad

Figura 5: Valores de la escala de medición en relación al grado de calidad que ofrecen

Para el caso de la medición de una métrica en el que ésta se puntúe según un criterio probabilístico (como un cuantificación de una métrica según una probabilidad entre 0 y 1), no numérico (clasificación entre una lista de valores) o de otro tipo, el cual no sea medible en la escala numérica anteriormente definida, será necesaria una transformación (previa extrapolación de ese criterio a un valor numérico si fuera necesario) del valor de calificación de esa métrica en cuestión, con el objetivo de clasificarla en esa escala de valores numérica del 0 al 10. La calificación de las métricas de calidad de los modelos de calidad para componentes web y para *mashups* creados a partir de la composición de estos componentes podrá realizarse según el criterio descrito en esta escala de medición.

5.2.2. Métodos de evaluación de las métricas.

En lo que respecta a los métodos matemáticos y estadísticos que se utilizarán en la medición de las métricas de los modelos de calidad se utilizarán, como se ha indicado en las especificaciones de métricas para ambos modelos explicadas en los capítulos anteriores, la media aritmética de valores y las clusterizaciones de los datos relacionados con dichas métricas sobre componentes web y *mashups*:

- Media arimética. La media aritmética es uno de los métodos matemáticas usados en el cálculo del valor numérico de métricas de calidad de determinados componentes o *mashups*. Es utilizada para hallar promedio aritméticos sobre conjuntos de valores que representan la medición de una característica que se distribuye por igual entre todos los componentes o *mashups*.

- Clusterización. Para la medición de algunas métricas de calidad cuyo criterio de medición no sigue un criterio numérico es posible llevar a cabo una clusterización de los valores de dichas métricas. Este proceso consiste en la utilización de un sistema basado en clases en las que se clasificarán los componentes web o los *mashups* según corresponda según un determinado criterio.El criterio que se seguirá será el de la utilización de cuantiles propios de la estadística descriptiva, los cuales dividen un conjunto de datos en partes iguales. Debido a que existen diversos tipos de cuantiles, utilizaremos para este caso de estudio según la escala de medición descrita anteriormente (con un intervalo de medición del 0 al 10), concretamente los deciles, nueve valores que dividen un conjunto de datos que se quiera procesar en diez grupos o *clústers* de igual tamaño, representados mediante diez intervalos del siguiente modo:

  [0-D1), [D1-D2), [D2-D3), [D3-D4), [D4-D5), [D5-D6), [D6-D7), [D7-D8), [D8-D9), [D9-MÁX), MÁX, siendo MÁX el valor máximo del conjunto de datos.

  Una vez los intervalos han sido definidos se asignará un correspondencia entre éstos intervalos y los valores de la escala de medición definida anteriormente, asignándose a cada intervalo un valor de calidad de la escala. El valor de máxima calidad en la escala de calificación se definirá según el valor máximo del conjunto de datos a estudiar para una característica.

Intervalos de los c*lústers*	Valores de la escala de medición
[0-D1)	0
[D1-D2),	1
[D2-D3),	2
[D3-D4),	3
[D4-D5),	4
[D5-D6),	5
[D6-D7),	6
[D7-D8),	7
[D8-D9),	8
[D9-MÁX),	9
MÁX	10

Figura 6: Correspondencia de los intervalos con la escala de medición de calidad

Cada vez que sea necesario cuantificar una métrica de un componente o *mashup* (debido a la inclusión de un nuevo componente o *mashup* web o a la modificación del valor de dicha métrica) se clasificará el valor numérico de ésta en el intervalo al que pertenezca según esta clusterización, asociándosele la correspondencia del valor de nuestra escala con dicho intervalo y, por ende, asignándole un valor entre 0 y 10 a dicha métrica en cuestión.

Si el valor numérico de la métrica en cuestión supera el valor máximo del conjunto de datos se le asociará el valor máximo de calidad a esa métrica de calidad, convitiéndose posteriormente este valor en el nuevo valor máximo (MÁX) del conjunto de datos correspondiente a dicha métrica.

Mediante este procedimiento, se podrá medir y cuantificar en la escala de medición definida la calidad de los componentes web y la calidad de los *mashups* de componentes en muy diversos aspectos y métricas de calidad.

5.3. Creación de un *framework* de anotado de resumen de calidad.

Se llevará a cabo el desarrollo de un *framework* de anotado cuyo propósito es el de realizar una evaluación de los modelos de calidad para componentes web y *mashups* descritos en capítulos anteriores, para llevar a cabo una asignación de un resumen de calidad para determinados componentes y *mashups*.

5.3.1. Agregación de valores a lo largo del árbol jerárquico de un modelo de calidad.

La estructura jerárquica de estos modelos de calidad tiene un papel de gran importancia en la creación de este *framework*. Dicha estructura se divide en torno a niveles de calidad por debajo de la calidad global que asocian estos modelos a componentes y *mashups*.

Estos tres niveles se categorizan en orden decreciente de nivel de calidad en calidad de alto nivel o factores, calidad de bajo nivel o criterios y atributos de medición de la calidad de bajo nivel, o métricas. Según estos modelos de calidad, las métricas se agrupan en torno a diferentes criterios de calidad, clasificándose éstos en factores, y éstos a su vez se congregan en torno a la calidad global de dichos modelos.

La medición de la calidad de estos modelos se llevará a cabo en el nivel más bajo de la jerarquía de dichos modelos, las métricas de calidad. Al encontrarse en el más bajo nivel de esta jerarquía, las métricas de calidad permiten medir directamente características software de un componente o *mashup* determinado, mientras que los criterios y factores de calidad son características medibles de manera indirecta al encontrarse en niveles más altos de la jerarquía, por lo que sus valores dependen de los valores asignados a las métricas de calidad y de los pesos computables que se les asignen.

Los valores de las métricas serán cuantificados según las metodologías descritas a lo largo de este capítulo. En las agrupaciones de métricas en torno a criterios de calidad, los pesos de importancia de éstas con respecto a las demás métricas de su agrupación deben ser computados. Una vez calculados los valores de cuantificación de cada métrica y evaluado su peso de importancia en el criterio de calidad en el que cada una se clasifique, podremos obtener el valor numérico correspondiente a cada uno de dichos criterios a través de la siguiente fórmula:

$C = \sum_i \omega_i^C \cdot metr^i$, donde

C representa al criterio en torno al que se agrupan un conjunto de métricas

$metr^i$ es el valor numérico de la métrica *i* perteneciente al criterio *C*

ω_i^C se refiere al peso (entre 0 y 1) de la métrica *i* en el criterio *C*

En relación a la fórmula anterior, todas y cada una de las métricas deben ser tenidas en consideración, por lo que se cumple la siguiente fórmula en la que el sumatorio de todos los pesos de las métricas de un determinado criterio es igual a 1:

$$\sum_i \omega_i^C = 1$$

Como se ha comentado anteriormente, los criterios se agrupan en torno a factores de calidad de manera similar a cómo las métricas lo hacen respecto a los criterios. Una vez que se han calculado los valores de estos criterios, se evaluarán sus pesos para obtener los valores de los factores, siguiendo la siguiente fórmula:

$F = \sum_j \omega_j^F \cdot C^j$, donde

F representa al factor de calidad en torno al que se agrupan un conjunto de criterios

C^j es el valor numérico del conjunto j perteneciente al factor *F*

ω_j^F se refiere al peso (entre 0 y 1) del criterio *j* en el factor *F*

Por último, la calidad global de un determinado componente web o *mashup* se calculará por medio de la ponderación de los pesos de los factores de calidad, quedando definida por la siguiente fórmula:

$Q = \sum_k \omega_k^Q \cdot F^k$, donde

Q representa la calidad final de un componente web o *mashup*, que será el valor de salida que producen los modelos de calidad

F^k es el valor numérico del factor de calidad *k*

ω_k^Q se refiere al peso (entre 0 y 1) del factor k respecto a la calidad global *Q*

Para mostrar un ejemplo de este proceso de ponderación de los pesos de métricas, criterios y factores de calidad siguiendo la estructura jerárquica de los modelos de calidad de componentes y *mashups* web, se presenta la siguiente tabla:

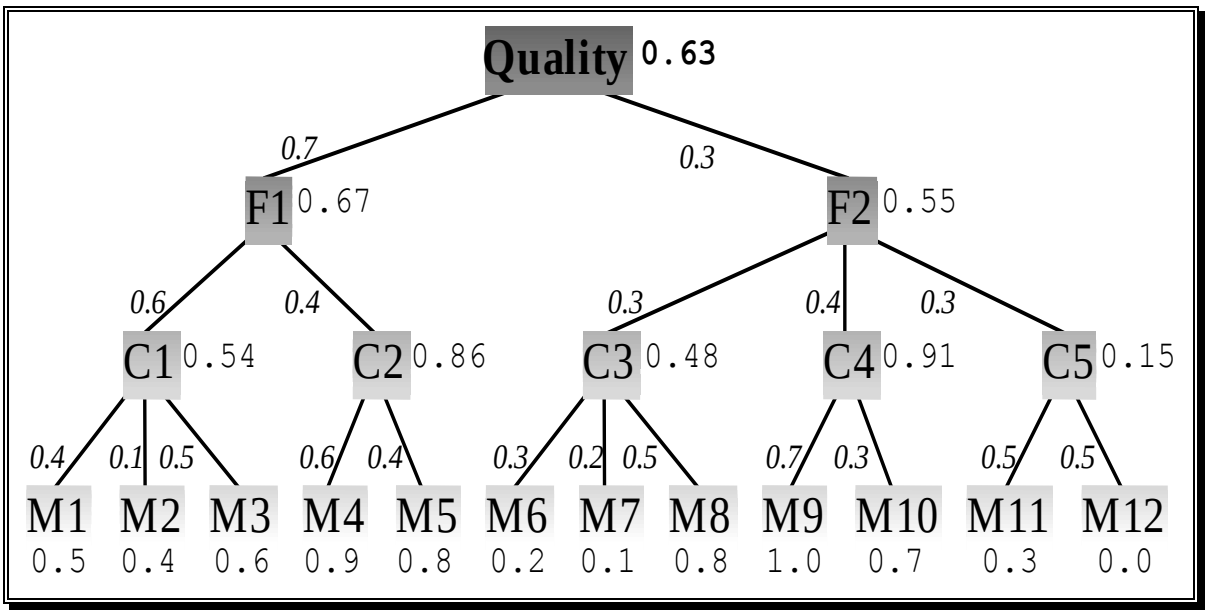

Figura 7: Espécimen del árbol del modelo de calidad con los valores de calidad y sus pesos agregados

En esta figura se muestra un árbol representativo de un modelo de calidad, con dos factores, cinco criterios y doce métricas de calidad, con sus valores y pesos asociados. Las ramas del árbol han sido etiquetadas con dichos pesos, mostrando el valor e importancia de cada nodo del árbol con respecto al nivel superior de la jerarquía del modelo al que pertenezca. Dichos pesos, y los valores de los niveles de la jerarquía superiores al nivel de las métricas de calidad, se calcularán siguiendo la evaluación que se muestra en la siguiente tabla:

C1 = 0.4·0.5 + 0.1·0.4 + 0.5·0.6 = 0.54 C2 = 0.6·0.9 + 0.4·0.8 = 0.86	C3 = 0.3·0.2 + 0.2·0.1 + 0.5·0.8 = 0.48 C4 = 0.7·1.0 + 0.3·0.7 = 0.91 C5 = 0.5·0.3 + 0.5·0.0 = 0.15
F1 = 0.6·0.54 + 0.4·0.86 = 0.67	F2 = 0.3·0.48 + 0.4·0.91 + 0.3·0.15 = 0.55
Quality = 0.7·0.67 + 0.3·0.55 = 0.63	

Figura 8: Cálculo de los valores de calidad a lo largo del árbol de un modelo de calidad

Los valores de los pesos del árbol jerárquico de los modelos de calidad de componentes web y *mashups* deben ser calculados por ingenieros de software, que seguirán un determinado criterio según su experiencia en el entorno de trabajo y según técnicas de decisión multicriterio sobre modelos cuyos atributos se agrupen en torno a una estructura jerárquica, como el Proceso Jerárquico Analítico (en inglés AHP o Analytic Hierarchy Process), el cual se detallará en la siguiente sección.

5.3.2. Utilización del AHP como técnica de decisión.

El AHP (*Analytic Hierarchy Process* [43]) es una técnica de decisión multicriterio basada en las matemáticas y la psicología. Se utiliza en situaciones en las que los atributos sobre los que se decide se organizan siguiendo una estructura jerárquica, descomponiendo la consecución de un objetivo global de decisión en factores, criterios y alternativas, en ese orden.

Este método permite establecer los pesos representantes de la importancia relativa de dichos factores con respecto a la calidad global, de los criterios en relación a estos factores y de las métricas de calidad con respecto a los criterios.

Es por esta razón por la que es adecuado el uso de este procedimiento para la medición de los modelos de calidad de componentes web y mashups. En dichos modelos, su estructura jerárquica se organiza en torno a un valor de calidad global, dividiéndose la jerarquía en factores, criterios y métricas de calidad, de manera análoga al proceso AHP.

La medición de la importancia relativa conlleva una gran ventaja con respecto a otros criterios de decisión, la objetividad que ofrece en la evaluación de la calidad de los modelos. Esta importancia relativa de los elementos evaluables dentro de un modelo determinado depende del contexto y el dominio en el que se sitúen los componentes y mashups web que se cuantifiquen.

Los atributos del modelo de más bajo nivel, en este caso las métricas de calidad, deben ser calificados según la importancia relativa, de un modo acorde a la opinión de expertos del dominio, ingenieros de software o usuarios entre otros.

Los elementos en el mismo nivel de la jerarquía deben ser comparados entre ellos en parejas para establecer el nivel de importancia relativa de cada uno de ellos con respecto al resto. Es posible utilizar una escala del 1 al 9 para determinar la importancia relativa de cada par de elementos del mismo nivel en la jerarquía del modelo.

Dicha escala se basa en los estudios experimentales llevados a cabo por el psicólogo George Miller, el cual afirmó que una persona es capaz de manejar información acerca de 7 ± 2 conceptos simultáneamente [44]. Una interpretación de los valores que esta escala asigna a cada par de elementos que se comparan en el mismo nivel de la jerarquía, en un determinado contexto, es la siguiente:

- El valor uno de la escala se asigna a un elemento A de la jerarquía cuando, al compararlo con otro elemento B, se evalúa que tienen la misma importancia relativa.
- El valor tres se asigna cuando, al comparar dos elementos A y B, se determina que A tiene más importancia relativa que B.
- Se asigna el valor cinco de la escala cuando un elemento A tiene bastante más importancia que B al compararlo con éste.
- Siete es el valor que se obtiene en la escala de importancia relativa para un elemento A que dispone de mucha más importancia relativa que un elemento B, cuando éstos se comparan.
- El valor nueve en la escala de medición se le asigna a un elemento A que, al compararlo con otro elemento B, tiene una importancia relativa extremadamente mayor que este último.

Una vez definida la escala, según el criterio de medición que ésta sigue podremos cuantificar la importancia relativa de los factores, criterios y métricas de los modelos de calidad para componentes web y *mashups* mediante la creación de matrices de importancia.

Dichas matrices permiten, ofreciendo una perspectiva similar a una tabla, definir para un elemento de un árbol jerárquico de un modelo (en este caso el valor de calidad global o un criterio o factor determinado) las importancias relativas de los elementos que se encuentran en un nivel en la jerarquía por debajo de él con respecto a éste.

La matriz de importancia relativa queda definida de la siguiente manera:

$A = (a_{ij}), a_{ij} = w_i / w_j, i, j = 1,..., n$, donde

A es la matriz de importancia relativa

a_{ij} representa cada uno de los elementos de la matriz

ω_i / ω_j es el cociente entre el peso de la importancia relativa del elemento *i* con respecto al elemento *j*, a partir del cual se obtiene el valor de la comparación entre estos dos elementos, comparación que representa a los elementos de la matriz *A*

Esta matriz dispone de elementos positivos debido a la utilización de los valores de la escala comparativa anterior y cumple la propiedad recíproca $a_{ji} = 1/a_{ij}, i, j = 1,..., n$, por lo que además de ser una matriz simétrica es considerada también recíproca, debido al cumplimiento de esta propiedad de reciprocidad en todos sus elementos.

Cada matriz genera un autovector, cuyo valor puede calcularse normalizando los valores de los elementos de cada columna de dicha matriz (dividiendo cada uno de ellos por la suma de todos los elementos de dicha columna) y computando la media por fila, lo que permite obtener un ránking de importancia relativa en todos los niveles de un modelo jerárquico.

Cada uno de los elementos de este autovector supone el peso de la importancia relativa, con respecto a un elemento de un nivel superior en la jerarquía, de una agrupación de elementos en un mismo nivel de jerarquía para un determinado modelo.

5.4. Asociación de una especificación de calidad a un componente o *mashup* web.

Una vez se hayan calculado los valores de los elementos del árbol jerárquico de un modelo de calidad para un componente o *mashup* determinado (según la asignación de calidad llevada a cabo por el *framework* de anotado), se asociará una especificación de calidad a ese componente o *mashup* para que los usuarios que lo utilicen puedan componer una mejor RIA y de mayor calidad que ante la ausencia de dicha especificación.

Esta especificación compondrá una descripción de la calidad de un componente o mashup, la cual podrá ser consultada por los usuarios que utilicen dichos componentes y mashups, para que éstos al desarrollar una aplicación web de mashup puedan elegir componentes que además de adecuarse a la funcionalidad que estos usuarios están esperando, sean componentes de calidad en diversos aspectos.

Cabe decir que esta asociación de calidad proviene de una descripción informativa, la cual no es un contrato de adecuación de calidad, debido a que los componentes son piezas de software desarrolladas por terceros asegura un nivel de calidad, por lo que no se puede garantizar la calidad de un componente de cara a una especificación formal de calidad.

El valor de calidad final que se asigna mediante un framework a un componente o aplicación de mashup web según un modelo determinado no es suficiente para evaluar su calidad. Por ello, se ha realizado una medición de calidad según una descomposición en diversos niveles de calidad agrupados en torno a una estructura jerárquica.

Esta calidad global, así como su descomposición en niveles de calidad de más bajo nivel, quedarán reflejados en esta descripción de calidad, presentándose la calidad de componentes y mashups en diversos aspectos, así como la interpretación que se deriva de dicha calidad.

Esta especificación de calidad consistirá en un documento en el que se indique a nivel informativo la calidad de un determinado componente web o mashup, tanto para la calidad global como para aquella en los diferentes niveles de los modelos de calidad definidos. Dicho documento dispondrá de tres secciones bien diferencias, las cuales son las siguientes:

- Una descripción en la que se presenta la calidad global de un componente web o mashup determinado, la cual llevará consigo un resumen presentando dicha calidad.
- Una sección de definiciones de la calidad del componente o *mashup* ofrecido por el componente o *mashup,* organizadas en torno a los aspectos de calidad de alto nivel correspondientes a los factores de alto nivel de los modelos de calidad definidos en capítulos anteriores, los cuales se desgranarán en los criterios de calidad de más bajo nivel, pertenecientes a cada una de las características de alto nivel mencionadas anteriormente.
- Un apartado con las mediciones de calidad correspondientes a las métricas de calidad de más bajo nivel.

Cada uno de los aspectos de calidad, tanto de alto como de bajo nivel, irán acompañados de la anotación de calidad obtenida a través de la aplicación del *framework* de anotado al componente o *mashup* al que vaya referida dicha SLA.

La generación de una especificación de calidad para un componente o aplicación de *mashup* determinada permitirá a un usuario final que desee crear una RIA la posibilidad de utilizar estos elementos para la composición de una aplicación enriquecida con calidad.

Estos componentes serán elegidos por la funcionalidad que éstos ofrezcan al usuario, además de su calidad, es decir, no se tendrán en cuenta únicamente como se adecúa un componente a un *mashup* según las necesidades de este desarrollador de aplicaciones.

Dicha calidad no es un mero resumen global, sino que consiste en un conjunto de anotaciones de calidad estructuradas jerárquicamente según una serie de factores y criterios.

Esto permitirá a estos desarrolladores de *mashups* buscar y elegir componentes que dispongan de una cierta calidad en áreas de interés con respecto al propósito que pretenden alcanzar con la creación de dicho *mashup*.

Capítulo 6.

Caso de uso de aplicación del marco de calidad.

Índice

Una vez se han definido los modelos de calidad para componentes web y *mashups*, además del *framework* de anotado de resumen de calidad, se ha determinado la aplicación del marco de caracterización a un caso de uso que sirva como escenario de ejemplo de aplicación de estas tecnologías para la creación de RIAs por parte de usuarios finales de una mayor riqueza que ante la ausencia de dichas tecnologías.

Para ello, se partirá de una aplicación relacionada con el dominio de los componentes web, cuyo principal propósito es el de ofrecer un repositorio de componentes web Polymer, además de un *sandbox* de prueba de estos componentes con otros del repositorio.

En dicha aplicación, se implementará un subconjunto de las métricas del modelo de calidad de componentes web y se aplicará este subconjunto a los componentes de la página. Seguidamente, se empleará el *framework* de anotado para establecer la calidad global de dichos componentes y su calidad en relación a los factores y criterios de calidad del modelo.

Por último, cuando los componentes web hayan sido calificados y su calidad asociada a una especificación, en un ámbito informativo, los usuarios podrán consultar dichas especificaciones y aprovecharse de éstas para la consecución de RIAs de una calidad mayor que ante la ausencia de dichas descripciones de calidad.

Actualmente la aplicación se encuentra en la fase de implementación de métricas del modelo para componentes web y, en un futuro, se aplicará el *framework* de anotado de calidad a dicho modelo para cada componente y especificaciones de calidad a los componentes de la aplicación web.

6.1. Aplicación web repositorio y *sandbox* de componentes web Polymer.

La aplicación web que está siendo objeto de estudio de este escenario de ejemplo es *Polymer Bricks*, un site que funciona como un repositorio de componentes web Polymer hospedados en la plataforma Github.

La aplicación no ofrece los componentes web como recurso, sino que actúa como un recubrimiento de los componentes que se encuentran almacenados en Github. Dicha página, en un futuro, dispondrá de un *sandbox* cuyo propósito consistirá en la prueba de componentes web y su funcionamiento de manera aislada, y en el testeo de la interoperabilidad de dichos componentes con el resto de componentes del site. Con respecto a la arquitectura del sitio web, ésta se describe mediante el siguiente diagrama de capas:

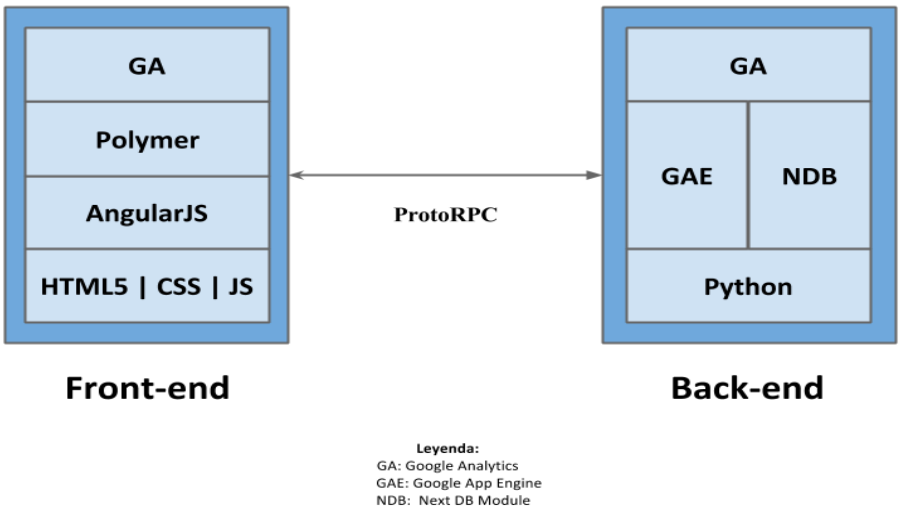

Figura 9: Diagrama de arquitectura de *Polymer Bricks*

Como se puede observar en la figura, la estructura del site se organiza en torno a dos módulos principales, un módulo front-end y otro back-end, los cuales a su vez se dividen en una serie de capas software, como se indica a continuación:

- Módulo front-end: dicho modulo contiene la parte cliente de la aplicación, con la que el usuario interactuará para buscar componentes que se adecúen a las funcionalidades que buscan para la construcción de sus RIAs y para la prueba de dichos componentes en el *sandbox*. Se agrupa en torno a las siguientes capas:
    - *Capa de HTML5, CSS y JavaScript*. Dicha capa, la cual contiene a los tres lenguajes de programación pilares de la programación web (contenido, estilo y comportamiento respectivamente), contiene la base sobre la que se apoyan las demás capas que se encuentran en niveles superiores del módulo.
    - *Capa de AngularJS*: contiene el framework web de JavaScript que constituye el núcleo de la parte front-end del site, con soporte para el modelo MVC. Porporciona soporte para la utilización de componentes web visuales, como por ejemplo aquellos de la librería Polymer.

- *Capa de Polymer*: esta capa abarca la utilización de la librería web de Polymer, la cual lleva consigo el uso de componentes web visuales y no visuales. La capa de Polymer se construye por encima de la del núcleo de Angular, ya que esta última ofrece soporte para la utilización de componentes web del lado front-end. Como decisión de diseño, se optó por la creación de un site con esta tecnología para demostrar el potencial de los componentes web en su utilización en aplicaciones web en fase de producción, y no únicamente como opción para crear prototipos de aplicaciones web.
- *Capa de GA (Google Analytics)*: la plataforma de Google Analytics se usa tanto en el módulo front-end como en el módulo back-end del site. Dicha plataforma se utilizará, en el lado front-end, para la recolección, almacenamiento y consulta de información de tráfico generada por la interacción de los usuarios con la aplicación.

- Módulo back-end: este módulo contiene la parte servidora de la aplicación que dará todo el soporte necesario a la parte cliente o front-end. Este módulo está agrupado en torno a las siguientes capas:
  - *Capa de Python*: supone la base del módulo back-end, el cual estará escrito totalmente en el lenguaje de programación Python. Se determinó su utilización debido a que es un lenguaje de alto nivel, orientado a objetos, cuya filosofía de diseño es la legibilidad de su código.
  - Capa de GAE (Google App Engine) y NDB (Next DB Module): dicha capa se divide en dos subcapas, GAE y NDB, las cuales se encuentran al mismo nivel. La primera constituye el núcleo del back-end de la aplicación, el cual consiste en una plataforma como servicio (PaaS) que permite desarrollar y hospedar aplicaciones en la infraestrucutra de Google en la nube. La segunda tecnología (NDB) es una API que provee persistencia al site, la cual se comunica a través de dicha API con el servicio de base de datos no relacional de Google, Google Cloud Datastore.
  - *Capa de GA (Google Analytics)*: al igual que en el lado front-end, se usará la plataforma de Google Analytics para la recolección y obtención de información de métricas de calidad en el módulo back-end del site.

En referencia a las tecnologías utilizadas en el site, las cuales son usadas para implementar las capas software de los módulos de la arquitectura de la aplicación, éstas se detallan a continuación:

- HTML5, CSS y JavaScript. Son los pilares sobre los que se construyen las aplicaciones web. Estas tecnologías definen el contenido, el estilo y el comportamiento respectivamente de dichas aplicaciones.

  Sobre estos lenguajes se han creado a lo largo del teimpo multitud de *frameworks* de desarrollo web, algunos de ellos utilizados en el desarrollo de *Polymer* Bricks, como por ejemplo AngularJS (implementado totalmente en JavaScript) y Polymer (librería que implementa los nuevos estándares de la W3C acerca de los componentes web con una sintaxis simple y orientada a usuarios finales).

- AngularJS. Es un framework de desarrollo web para la parte cliente o front-end de páginas o aplicaciones web, implementado en su totalidad mediante el lenguaje JavaScript y de código abierto. Se usa normalmente en aplicaciones SPA (Single Page Applications), que utilizan el patrón de arquitectura de MVC (Modelo Vista Controlador). Este framework constituye el núcleo de la parte front-end del site y sobre él se construyen otras capas software (Polymer y Google Analytics).
- Polymer. Polymer es una librería front-end para el desarrollo de componentes web HTML que sigue las especificaciones sobre componentes establecidas por la W3C. Permite tanto la creación de componentes web como elementos HTML autónomos como la construcción de aplicaciones web desarrolladas completamente con esta librería. La capa de Polymer de la aplicación se construye sobre la capa de AngularJS y, gracias al enorme potencial de dicho *framework*, es posible la integración entre AngularJS y la librería de Polymer.
- Python. Este lenguaje de programación ha sido la opción elegida para llevar a cabo la implementación del núcleo del back-end del site, ya que dicho lenguaje está orientado a objetos y es multiplataforma. Python es usado por Google App Engine y el módulo NDB.
- Google App Engine. Es un servicio ofrecido en la nube por Google cuyo funcionamiento es el de una plataforma como servicio (PaaS) para desarrollar y hospedar aplicaciones (en el lado servidor de una página web) en las que la escalabilidad de usuarios que acceden a la aplicación y la gestión de los servidores es realizada de manera automática por Google.
- NDB. NDB o Next DB Module es un módulo escrito en Python consistente en una API de acceso a bases de datos no relacionales Google Datastore. Dicho módulo se encargará de la persistencia de los datos del site e interactuará con la plataforma de Google App Engine almacenando los datos del site, así como toda la información relacionada con las métricas de calidad asociadas a los componentes web que éste alberga.
- Google Analytics. Es un servicio ofrecido por Google para la generación de todo tipo de estadísticas acerca del tráfico web generado en páginas o aplicaciones web y de interacción de los usuarios con ellas. Esta plataforma es utilizada por el site para la recolección de información de los componentes y de las métricas de calidad asociadas a los componentes web hospedados en esta página.

En cuanto a los principios de diseño contemplados para la comunicación de datos entre el módulo front-end y el módulo back-end del site son los siguientes:

- HTTPS REST. Esta alternativa consiste en la comunicación de datos a través del el estilo arquitectónico REST y el protocolo HTTPS, ofreciendo cifrado en ambos extremos mediante el protocolo SSL.

  Es la alternativa con el menor *overhead* posible en los mensajes que se transmiten entre los módulos del site, pero por el agujero de seguridad descubierto recientemente en el protocolo SSL, ésta no es una alternativa factible para su utilización en *Polymer Bricks*.

- ProtoRPC. Es una librería RPC implementada y mantenida por Google para la comunicación de datos utilizando este protocolo a través de HTTP. Utiliza para la comunicación de datos implementaciones para estándares como JSON. Ésta ha sido la opción elegida para la comunicación entre el front-end y el back-end, debido a su utilización del protocolo RPC sobre HTTP, que minimiza el *overhead* de los mensajes que se intercambian entre las partes de la comunicación, y en el caso concreto que compete al site, la comunicación entre los módulos front-end y back-end.
- Cloud Endpoints. Esta alternativa permite crear una API en el lado back-end de una aplicación web y permite acceder a ella mediante una API REST. Está implementada sobre ProtoRPC y dispone un proxy HTTP, el cual ralentiza la comunicación entre los módulos, por lo que su uso ha sido descartado en el site.
- ZeroRPC. Librería RPC ligera y agnóstica en lo referente al lenguaje (soporta tanto JavaScript como Python en los lados cliente y servidor de las aplicaciones web), es utilizada para comunicación distribuida. Dicha librería se basa en el protocolo RPC y utiliza ZeroMQ para la comunicación de datos y MessagePack para la transmisión de mensajes. Su inconveniente es la utilización de un proxy en su implementación, que haría más lenta la comunicación entre el front-end y el back-end del site, por lo que su uso no es adecuado para el site.

6.2. Implementación y aplicación del marco de métricas de calidad.

El caso de uso que está siendo objeto de estudio contempla la aplicación de un subconjunto de métricas de calidad, las cuales pertenecen al modelo para componentes web definido en capítulos previos, a la aplicación web de *Polymer Bricks*. Este subconjunto de métricas no será aplicado directamente sobre la página, sino que se realizará una medición de métricas sobre los componentes pertenecientes alojados en su repositorio.

Para llevar a cabo la implementación y aplicación de las métricas a dichos componentes, será necesario primeramente la recogida y procesamiento de datos relativos a éstos. Seguidamente, se deberán calcular las métricas de calidad de más bajo nivel del y, una vez obtenidos los valores numéricos de dichas métricas, asignaremos mediante el *framework* de anotado un resumen de calidad, tanto global como en diferentes niveles del modelo de calidad a los componentes.

Por último, asociaremos una especificación de calidad a dichos componentes como se ha comentado anteriormente para que en el desarrollo de una aplicación de *mashup* por parte de un usuario final éste pueda aprovecharse de las ventajas ofrecidas por estas especificaciones, de tal manera que se llegue a la consecución de una RIA de una mayor calidad que ante la ausencia de este marco de caracterización de calidad.

6.2.1. Evaluación de las métricas de calidad.

Para efectuar la evaluación de las métricas de calidad en este escenario de ejemplo en primer lugar se recolectará y gestionará la información relacionada con los componentes de la aplicación *Polymer Bricks*. Una vez esta información ha sido obtenida y procesada, se llevará a cabo la medición automática de las métricas correspondientes al modelo de calidad para componentes web, según los criterios seguidos por dicho modelo y los procedimientos de evaluación definidos anteriormente.

La realización, tanto del proceso de recolección y tratamiento de información relativa a los componentes, como de la calificación de las métricas del modelo de calidad para dichos componentes, se describe en las siguientes secciones.

6.2.1.1. Recolección, gestión y consulta de información.

Para llevar a cabo la recogida y tratamiento de la información relacionada con los componentes de la aplicación, se ha determinado la utilización del servicio de Google Analytics, una plataforma que genera estadísticas de tráfico y de interacción de usuarios con aplicaciones y páginas web.

En relación a la plataforma Analytics y su integración en el site, es necesaria la definición en su módulo front-end de un código de seguimiento implementado en el lenguaje JavaScript, el cual se encargará de cargar de forma asíncrona un objeto de seguimiento global a dicho site de los servidores de Google.

Este código de seguimiento, el cual debe incluirse en todas las páginas o vistas de la aplicación con el objetivo de que se generen estadísticas en cada una de ellas, se expone a continuación:

```
<script>
  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,
  m)
  })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
  ga('create', 'UA-XXXX-Y', 'auto');
  ga('send', 'pageview');
</script>
```

Figura 10: Código "*analytics.js*" y objeto global de seguimiento "*ga*" []

Para el envío de eventos a Analytics en el lado front-end de una aplicación web se podrá utilizar el método "send" del objeto global "ga" cargado asíncronamente mediante este código de seguimiento, siguiendo la siguiente estructura:

```
ga('send', 'event', 'categoría', 'acción', 'etiqueta', valor);
```

send – Método del objeto global "ga"

event – Evento de envío de información a Google Analytics

categoría – Categoría del evento que se envía

acción – Tipo de interacción con la aplicación

etiqueta – Clasificación del evento o de algún objeto relacionado con él

valor – Valor numérico positivo del evento

Figura 11: Estructura de un evento de envío a Google Analytics

Existe la posibilidad de la recopilación y envío de datos, así como de consulta de éstos desde el lado servidor o back-end de una aplicación (así como también desde la parte cliente en referencia a la consulta de datos a la plataforma).

Para el primer caso, en el back-end se puede utilizar una de las APIs que trae consigo Google Analytics, la Management API, la cual permite enviar peticiones HTTP a la plataforma con la información que se desee recopilar. En referencia al segundo caso, es posible utilizar la Core Reporting API tanto en el lado servidor como en el lado front-end de la aplicación, incluida también en la *suite* de APIs que ofrece el servicio de Analytics.

Teniendo en cuenta que el site actúa como un recubrimiento de los componentes hospedados en Github, es necesario obtener a través de la API que ofrece dicha plataforma información relevante a estos componentes. Dicha información hace referencia a características propias de los repositorios de dichos componentes, como seguimientos o descargas de un componente determinado o versionado de los repositorios de éstos entre muchas otras.

6.2.1.2. Persistencia de la información.

Para el almacenamiento y persistencia de los datos recolectados, se ha determinado el uso de la plataforma Analytics para el almacenamiento de datos de forma transitoria, previa a la persistencia de datos definitiva en la base de datos no relacional conectada al módulo back-end de la aplicación.

Dicha base de datos no relacional ha sido utilizada para la persistencia definitiva de esta información, hospedada en un módulo conectado al lado back-end del site como se ha detallado anteriormente en la especificación de la arquitectura del site.

6.2.1.3. Cuantificación de las métricas de calidad.

En referencia a las métricas del modelo de calidad para componentes web, éstas son las características medibles de más bajo nivel para cada uno de dichos componentes, las cuales necesitarán de un proceso de evaluación sobre los componentes relativos a este caso de uso.

En primer lugar será necesario obtener, en el lado front-end, información de cada una de estas métricas a través de las interacciones con la aplicación por parte de los usuarios y enviarla a la plataforma de Google Analytics.

De manera simultánea, dicha información será enviada al lado back-end mediante la llamada a métodos de una API implementada en este módulo del site y privada, visible únicamente al front-end. El back-end se encargará de procesar esa información y almacenarla en la base de datos no relacional.

El back-end del site será el encargado de procesar esta información que recibe del front-end, extrapolándola a métricas de calidad de los componentes, cuyos valores se almacenarán también en dicha base de datos.

En lo que respecta a las mediciones de las métricas, actualmente en el site se está llevando a cabo la implementación de cinco métricas de calidad, cada una correspondiente a un factor de calidad de alto nivel, se llevarán a cabo según los criterios definidos en el modelo de calidad para componentes.

Cada una de las métricas, así como el proceso de implementación llevado a cabo en la aplicación, se detalla a continuación:

- Factor de calidad de los datos.

  En lo que respecta a la calidad de los datos, como se ha explicado en capítulos anteriores, disponemos de cuatro criterios de calidad de más bajo nivel: precisión, completitud, actualidad temporal y disponibilidad de los datos. Para ilustrar cómo se realizaría la medición de estos criterios mediante métricas de calidad, en el contexto del site, se ha determinado la utilización de la métrica de actualidad temporal de los datos para su cuantificación en dicho site, a partir de la siguiente fórmula:

  ActTemporalDatos = max (0,1 – (actualidad_datos / volatilidad_datos))^{s} , donde

  actualidad_datos es la edad que tienen los datos proveídos por un componente desde que fueron creados o actualizados por última vez

  volatilidad_datos se refiere a una dimensión estática que representa el período de tiempo de validez de los datos según un contexto determinado

  s es el exponente que controla la sensibilidad de dicha actualidad de los datos, cuyo valor se estima de manera subjetiva por parte del ente que analice los datos, dependiendo del contexto en el que se encuentren éstos

  Con el fin de implementar esta métrica de frescura de los datos, la cual mide cómo de actualizados están los datos que provee un componente, pueden hacerse pruebas desde el back-end del site de obtención de datos para los distintos servicios o APIs externas que ofrezca el componente, mediante la utilización de datos de invocación de prueba.

  De esta manera, podremos comprobar la frescura de dichos datos en términos de actualidad o edad con respecto a su validez, la cual será determinada en la aplicación de manera estática, establecida según un determinado contexto o escenario, al igual que para constante *s* de sensibilidad.

Para comprobar la frescura de los datos solamente será necesaria una entrada de prueba para cada API o servicio llamado de esta manera.

El valor de esta métrica será almacenado por el back-end en la base de datos al convertir el porcentaje resultante para cada métrica en un valor medible en la escala numérica definida para el site.

- Factor de calidad de la API del componente.

  La calidad de la API del componente se divide en cuatro criterios de calidad: interoperabilidad, granularidad, confiabilidad y usabilidad de la API.

  Con el fin de presentar la calificación de estas métricas para el site, se ha determinado la utilización de la métrica de interoperabilidad que ofrece un componente web determinado, la cual viene determinada por la siguiente fórmula:

  metr_interoperabilidad = $\frac{|\ \textbf{entradasYsalidas\_cw}\ |}{\textbf{attr\_parametrizables}}$, donde

  entradasYsalidas_cw se refiere a la cantidad de entradas y salidas que ofrece un componente para su interconexión con otros

  attr_parametrizables son los atributos publicados por el proveedor del componente y customizables por parte de los usuarios que los utilicen

  Para llevar a cabo esta medición de la interoperabilidad, no es posible inspeccionar el código fuente de un componente en busca de entradas, salidas o atributos parametrizables de dicho componente debido a la aproximación de calidad realizada según un modelo de caja negra.

  Los componentes web, de manera acorde al marco de caracterización definido, son evaluados en referencia a sus características externas y de calidad en uso, según un modelo de caja negra, con motivo de la característica de componetización de éstos. Esta aproximación (y no una de caja blanca de cuantificación de la calidad interna del código fuente) se lleva a cabo debido a la naturaleza de las aplicaciones *mashup* de la Web 2.0

  Como alternativa a la inspección del código fuente de los componentes web del repositorio, en el site se ha implementado un sistema de anotado de metadatos para los componentes web del repositorios acorde a un *template* o plantilla en el lenguaje XML, validado por un fichero de *schema* XSD o un fichero del estándar RDF.

  Estos metadatos contendrán todo tipo de información sobre el componente como su autor, versión, entradas y salidas, atributos parametrizables definidos en éste o los elementos que importa para su utilización dentro de él entre muchos otros. Un ejemplo de esta plantilla XML utilizada para un determinado componente aparecerá representado de la siguiente manera:

```xml
<?xml version="1.0" ?>
<Template xmlns="https://centauro.ls.fi.upm.es/ns/template#">
  <!-- Widget MetaData -->
  <Repo.ComponentDescription>
    <Vendor>Polymer Group</Vendor>
    <Name>wrapper-core-input</Name>
    <Author>David Herranz</Author>
    <Version>0.6</Version>
    <DisplayName>__MSG_displayName__</DisplayName>
    <Description>__MSG_description__</Description>
    <WikiURI>doc/index.html</WikiURI>
    <ContactMail>dherranz@conwet.com</ContactMail>
    <License>Apache License 2.0</License>
    <LicenseURL>http://www.apache.org/licenses/LICENSE-2.0.html</LicenseURL>
  </Repo.ComponentDescription>

  <Platform.IO>
    <InputEndpoint label="__MSG_slotLabel__" name="slotText" type="text"/>
    <OutputEndpoint label="__MSG_eventLabel__" name="eventValue" type="text"/>
  </Platform.IO>

  <Platform.Attributes>
    <Attribute name="placeholder" type="string" default="Insertar texto" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
    </Attribute>
    <Attribute name="value" type="string" default="Insertar texto" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
    </Attribute>
    <Attribute name="color" type="string" default="white" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
    </Attribute>
    <Attribute name="width" type="string" default="70px" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
    </Attribute>
    <Attribute name="height" type="string" default="32px" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
    </Attribute>
    <Attribute name="handleOutputsAs" type="string" default="json" description="__MSG_attrDescription__" label="__MSG_prefLabel__">
      <Option label="__MSG_optionName__" value="json"/>
      <Option label="__MSG_optionName__" value="xml"/>
      <Option label="__MSG_optionName__" value="text"/>
      <Option label="__MSG_optionName__" value="arraybuffer"/>
      <Option label="__MSG_optionName__" value="blob"/>
      <Option label="__MSG_optionName__" value="document"/>
    </Attribute>
  </Platform.Attributes>

  <Platform.Demo>
    <XHTML href="demo.html"/>
  </Platform.Demo>

  <Translations default="en">
    <Translation lang="en">
      <msg name="displayName">Wrapper Widget of the core-input component</msg>
      <msg name="description">Wrapper widget of the core-input Polymer element</msg>
      <msg name="attrDescription">Attribute description</msg>
      <msg name="attrLabel">Attribute label</msg>
      <msg name="optionName">Attribute option</msg>
      <msg name="slotLabel">Slot label</msg>
      <msg name="eventLabel">Event label</msg>
    </Translation>
    <Translation lang="es">
      <msg name="displayName">Widget wrapper del componente core-input</msg>
      <msg name="description">Widget de recubrimiento del elemento Polymer core-input</msg>
      <msg name="attrDescription">Descripción del atributo</msg>
      <msg name="attrLabel">Etiqueta del atributo</msg>
      <msg name="optionName">Opción del atributo</msg>
      <msg name="slotLabel">Etiqueta del endpoint de entrada</msg>
      <msg name="eventLabel">Etiqueta del endpoint de salida</msg>
    </Translation>
  </Translations>
</Template>
```

Figura 12: Template XML para el elemento del site *wrapper-core-input*

Mediante esta plantilla, el back-end del site podrá consultar para cada componente las entradas y salidas de éste, así como los atributos parametrizables definidos en el componente a fin de llevar a cabo la medición de la interoperabilidad de cada componente en el site. Una vez realizada dicha medición y obtenido el valor de interoperabilidad, éste deberá ser extrapolado a un valor dentro de la escala de medición del uno al diez en el site.

- Factor de calidad de presentación:

  En referencia al factor de calidad de presentación de los componentes, este se desgrana en los criterios de usabilidad y accesibilidad. Se ha decidido usar la métrica de accesibilidad de los componentes para su evaluación en el site, la cual evalúa el nivel de accesibilidad de un componente acorde al nivel de conformidad con los estándares publicados por la W3C Web Accessibility Initiative (WAI), mediante la siguiente fórmula:

  accesibilidad_componente = 0.5*nivelWCAG + 0.5*gradoWAI-ARIA

  Esta métrica será la encargada de extrapolar la conformidad de un componente determinado con estos dos estándares de la WAI (tanto el nivel obtenido para el estándar WCAG 2.0 como el porcentaje de cumplimiento del estándar ARIA 1.0) a un valor evaluable en la escala de medición del site que representará la calificación de este criterio para dicho componente.

Para llevar a cabo esta cuantificación existen diversas páginas de validación de código fuente HTML y CSS de páginas web, que ofrecen una idea de la accesibilidad de una determinada página web con respecto a estas dos tecnologías, como por ejemplo el validador oficial de la W3C para HTML [45] y para CSS [46], además de la web Site Analyzer [47], utilizadas para la evaluación de directrices de accesibilidad establecidas por la W3C. Estas herramientas disponen de dos inconvenientes:

- Uno de ellos consiste en que ofrecen una validación de HTML y CSS para páginas web, con una indicación de si éstas son o no válidas contra un subconjunto de reglas de accesibilidad, sin ofrecer una validación que conlleve una certificación de nivel contra los estándares WCAG 2.0 y WAI-ARIA 1.0. Por este motivo, será necesaria la comprobación de cada una de las directrices de ambos estándares.
- El otro inconveniente es que dichas herramientas web no ofrecen una API pública que pueda ser accedida por el back-end del site para la medición contra estándares de accesibilidad de los componentes web del repositorio.

Por estos dos motivos, será necesaria la implementación en la aplicación de cada una de las directrices de accesibilidad referentes a los dos estándares de medición de accesibilidad de la W3C, los estándares WCAG 2.0 y WAI-ARIA 1.0, los cuales han sido tenidos en cuenta para el modelo de calidad para componentes web. Dichas directrices serán en un futuro implementadas en el site *Polymer Bricks*. Los componentes web del repositorio serán evaluados según estas directrices en relación a los dos estándares, analizándose las tecnologías HTML y CSS de éstos.

Cada componente será evaluado y dispondrá de un nivel de accesibilidad que surge de la media entre la puntuación del nivel de accesibilidad que éste obtenga según el estándar WCAG 2.0 y el porcentaje de cumplimiento para dicho componente de las directrices del estándar WAI-ARIA 1.0.

- Factor de calidad de impacto social:

En cuanto a la calidad de impacto social, disponemos de seis criterios de calidad de más bajo nivel: reputación, seguimientos del componente o de su repositorio, descargas y popularidad en las redes sociales de Facebook, Google+ y Twitter. En relación al caso de uso del site se medirá la métrica de reputación de usuarios finales, la cual se evaluará para cada componente web del site realizando una ponderación media de todas las valoraciones de dichos componentes individualmente, mediante la siguiente fórmula:

$$\textbf{metr\_valoracion} = \frac{\Sigma \textbf{valoracionesUsuarios}}{|\ \textbf{valoraciones}\ |}$$

El site dispondrá para cada componente de un sistema de valoración que se ofrecerá a los usuarios finales que visiten y utilicen la aplicación, el cual permitirá puntuar la calidad de un componente mediante un *rating* basado en estrellas en una escala de 0,5 a 5. Podrán considerarse valores intermedios, los cuales posteriormente serán redondeados en el back-end. La aplicación presentará también la valoración media del componente por parte de los usuarios para que éstos la visualicen.

Esta valoración se almacenará tanto en Google Analytics como en la base de datos no relacional del back-end de *Polymer Bricks*.

- Para el primer caso, se enviará desde el front-end, una vez se lleve a cabo una valoración por un usuario determinado, un evento a Google Analytics en el código JavaScript de la siguiente manera:

```
$scope.valorar = function() {

$scope.valoracion=(document.querySelector("#rateComponent").value*10)/5;
 ga('send', 'event', 'repo', 'rating', 'idComponent', $scope.valorado);
});
```

  Mediante este código en el framework web de AngularJS, tecnología que constituye el núcleo en el que está implementado el front-end de la aplicación (como se ha explicado previamente en la descripción de la arquitectura), se enviará un evento a Google Analytics en el que se indicará la categoría del evento que se envía (perteneciente al repositorio y no al sandbox en este caso), la acción del evento (rating o valoración), etiqueta del evento con el id del componente y valor con la puntuación de la valoración, la cual será transformada posteriormente a la escala de medición del 0 al 10 definida para la medición de métricas según el modelo de calidad para componentes web.

  En la plataforma Analytics quedarán almacenadas las valoraciones correspondientes a todos los componentes del repositorio, las cuales se podrán consultar posteriormente a través de la API del servicio y analizar mediante estadísticas, como por ejemplo la media de las valoraciones de un componente que constituye la métrica que está siendo objeto de estudio.

- En el segundo caso, se almacenará información con respecto a los componentes y sus repositorios en Github, además de información de usuarios de la aplicación y las valoraciones que han realizado de éstos (con sus valores correspondientes en la escala de medición del site).

  Por ello, se almacenarán instancias de los repositorios de los componentes que incluirán, entre otros, información de la valoración media por parte de los usuarios, el sumatorio de las reputaciones y el número de ellas que se han llevado a cabo.

  Por otra parte, en la información almacenada en las instancias de la base de datos referente a los usuarios de la aplicación, se incluirá información de las valoraciones que han realizado para cada componente.

  En el back-end se llevará a cabo el procesamiento de toda esta información así como su redondeo en caso de que sea necesario, además de su almacenamiento en la base de datos para su persistencia en la infraestructura de la aplicación.

- Factor de calidad del proveedor y del hosting:

  Para la calidad del proveedor del componente y el hosting, se dispone de cinco criterios de calidad: mantenimiento del proveedor y disponibilidad, rendimiento, seguridad y confiabilidad del hosting. Con el fin de mostrar cómo realizar la medición de estas métricas, se utiliza la métrica de mantenimiento para su evaluación en el site *Polymer Bricks*. Dicha métrica se calificará según la siguiente fórmula:

  $$\textbf{metr\_mantenimiento} = \frac{\sum \textbf{tasaResolucionBugsRelease}}{|\ \textbf{releases}\ |}$$, donde

  $$\textbf{tasaResolucionBugsRelease (mantenimiento\_release)} = \frac{\textbf{bugsResueltos} * 100}{\textbf{bugsTotales}}$$

  Para cada componente se mostrará al usuario el grado de mantenimiento global (ponderación media entre todas las *releases*), así como el valor de mantenimiento de la versión que se esté visualizando, la cual será la última del repositorio de dicho componente.

  Se ha determinado el uso de un mecanismo en el site para que el usuario pueda notificar un error o bug y asociarlo a ese componente en cuestión al visualizar el componente en el repositorio o una vez probado éste en el *sandbox*.

  Estas notificaciones se enviarán a Google Analytics, debido a que dicha información surge de la interacción del usuario con la aplicación, con el propósito de almacenar dicha información para que después sea consultada y analizada por el back-end, información que penalizará la métrica de mantenimiento de los componentes web.

  Se utilizará la capacidad de procesamiento del back-end para almacenar y procesar información sobre cada una de las releases de un componente determinado acerca de muchas de sus características, que se obtendrán del repositorio del componente en la plataforma Github.

  La característica que más nos interesa en este ámbito del mantenimiento es el número de bugs que han sido detectados desde la fecha de lanzamiento de una *release* hasta el lanzamiento de la siguiente, ya que éstos pertenecerán a dicha versión (para el caso de la última versión del componente se contarán los bugs que se han detectado desde su lanzamiento hasta la actualidad).

  Una vez se disponga de la cantidad de errores o bugs totales para un *release* determinada, es necesario el análisis de cuántos de dichos bugs han sido resueltos por el desarrollador del componente para esa *release*, con el fin de comprobar el ratio de resolución de errores por *release* con respecto a los errores totales para dicha versión. La fórmula de cálculo de dicho ratio se corresponde con aquella de la tasa de resolución de bugs por *release* mostrada anteriormente.

  Por último se realizará una ponderación media de los ratios de todas las releases para hallar la métrica de calidad de mantenimiento total del componente. El valor de esta métrica será convertido a la escala de medición del site para almacenarlo en la base de datos, incluyéndose también en la instancia de la base de datos referente al componente el grado de mantenimiento de la última versión o *release* de éste.

En un futuro, a medida que avance el proyecto de *Polymer Bricks*, se implementarán cada una de las métricas del modelo de calidad para componentes web, para poder aplicar el *framework* de anotado de calidad a cada uno de los componentes del repositorio de la aplicación, así como posteriormente asignar una especificación de la calidad a cada uno de ellos.

6.2.2. Clasificación de la calidad mediante el *framework* de anotado.

Una vez dispongamos de algoritmos para implementar en la aplicación *Polymer Bricks* las métricas de calidad de bajo nivel correspondientes al modelo de calidad para componentes web, se podrá implementar en la página el framework de anotado de calidad correspondiente a este modelo, con el objetivo de asignar un resumen de calidad a los componentes web de la aplicación.

Como se ha descrito anteriormente, al disponer de un modelo de calidad con una estructura jerárquica dividida en calidad global dividida en factores, criterios y métricas de calidad, será necesario realizar un análisis de la importancia relativa de los niveles del modelo, con respecto al elemento del nivel inmediatamente superior al que pertenezcan siguiendo un procedimiento *bottom-up*, para que en el momento en el que se disponga de la medición de los elementos del nivel más bajo del modelo (las métricas) se puedan obtener los valores de los elementos de los niveles superiores de dicho modelo.

Por esto, se llevará a cabo la creación de matrices de importancia relativa que medirán la importancia de las métricas de calidad con respecto a los criterios, de los criterios en relación a los factores de calidad y de los factores con respecto a la calidad global del modelo.

Estas matrices de importancia relativa quedarán definidas a continuación, según el modelo de calidad para componentes web descrito en capítulos anteriores. En lo que respecta a los criterios de calidad del modelo, la mayoría de éstos, a excepción del criterio de funcionalidad del factor de calidad de los datos y del criterio de disponibilidad de los servidores del componente perteneciente al factor de calidad del proveedor del componente y su hosting, disponen de una única métrica en un nivel más bajo que define la calidad de dicho criterio.

Por este motivo, en estos casos se asumirá una importancia relativa de la única métrica asociada a ese criterio de uno en una escala de importancia de cero a uno, por lo que no será necesaria la creación de una matriz de importancia para dichos criterios.

De esta manera, las matrices de importancia para los distintos niveles del modelo de calidad para componentes web quedarán especificadas de la siguiente manera:

Matrices de importancia relativa de las métricas con respecto a los criterios de calidad. Aparecerán seguidamente definidas las matrices correspondientes a los criterios de calidad de funcionalidad, de accesibilidad y de disponibilidad del servicio, debido a que los demás criterios solamente tienen asociada una métrica de calidad (con un peso de importancia de uno) y no es necesaria la creación de una matriz de importancia relativa para dichos criterios:

- Criterio de funcionalidad. En lo que respecta a este criterio, destacamos tres atributos o métricas: la cantidad de RIAs en las que participa el componente, la conformidad con estándares de la API del componente y la seguridad en el acceso a las funcionalidades del componente. En relación al ámbito de funcionalidad del componente, el número de aplicaciones en las que participe éste se considera de mayor importancia en comparación a la seguridad de acceso y conformidad de éste. Desde el punto de vista de la funcionalidad, la conformidad con estándares tiene una importancia ligeramente superior a la de la seguridad de acceso. La matriz quedará descrita del siguiente modo:

	Cantidad RIAs	Conformidad	Seguridad Acceso	**Autovector**
Cantidad RIAs	1	5	7	**0.73**
Conformidad	1/5	1	3	**0.20**
Seguridad Acceso	1/7	1/3	1	**0.07**
Normalizado por:	47/35	19/3	10	

Figura 13: Matriz de importancia para el criterio de funcionalidad

Una vez dispongamos de la matriz de importancia, hallaremos los valores del autovector, dividiendo cada uno de los elementos de la matriz por el valor de normalización de su respectiva columna, computando la media por cada fila. De este modo se obtendrán los valores del autovector especificado en la columna de la derecha de la matriz, los cuales indicarán el peso de importancia de cada métrica en el criterio de funcionalidad:

crit_funcionalidad = 0.73*metrCantRIAs + 0.20*metrConform + 0.07*metrSeg

- Criterio de accesibilidad. Referentemente a este criterio de calidad, destacamos dos atributos o métricas del mismo: el nivel de accesibilidad según el estándar WCAG 2.0 y el grado de cumplimiento del estándar WAI-ARIA 1.0. En relación al ámbito de accesibilidad del componente, los dos estándares disponen de una importancia similar. La matriz quedará definida del siguiente modo:

	WCAG 2.0	WAI-ARIA 1.0	**Autovector**
WCAG 2.0	1	1	**0.50**
WAI-ARIA 1.0	1	1	**0.50**
Normalizado por:	2	2	

Figura 14: Matriz de importancia para el criterio de funcionalidad

Una vez dispongamos de la matriz de importancia, hallaremos los valores del autovector, realizando la división de cada uno de los elementos de la matriz por el valor de normalización de su respectiva columna, computando la media por cada fila de la matriz. De este modo se obtendrán los valores del autovector especificado en la columna de la derecha de la matriz, los cuales indicarán el peso de importancia de cada métrica en el criterio de funcionalidad:

crit_accesibilidad = 0.50*metrWCAG + 0.50*metrWAI-ARIA

- Criterio de disponibilidad. En relación a dicho criterio de calidad, éste se divide en tres métricas: el período de funcionamiento o *uptime* del servidor, la ventana de mantenimiento entre reparación de incidencias y la disponibilidad de los servidores. Desde el punto de vista de la disponibilidad del hosting, la disponibilidad de los servidores se considera de mayor importancia que su *uptime* y de una importancia bastante mayor que su ventana de mantenimiento o *maintenance time*. Por otro lado, el *uptime* dispone de una importancia superior a la ventana de mantenimiento. La matriz de importancia relativa se definirá del siguiente modo:

	Uptime	Ventana Mantenimiento	Disponibilidad Servidores	**Autovector**
Uptime	1	5	1/3	**0.24**
Ventana Mantenimiento	1/5	1	1/7	**0.12**
Disponibilidad Servidores	3	7	1	**0.64**
Normalizado por:	21/5	13	31/21	

Figura 15: Matriz de importancia para el criterio de disponibilidad

Cuando se cree la matriz de importancia, hallaremos los valores de su autovector asociado, los cuales indicarán el peso de importancia relativa de cada métrica respecto al criterio de disponibilidad que está siendo objeto de estudio:

crit_dispon_srv = 0.24*metrUP + 0.12*metrMT + 0.64*metrDisponSrv

Matrices de importancia relativa de los criterios con respecto a los factores de calidad. A continuación se definirán las matrices de importancia relativa correspondientes a los factores del modelo de calidad para componentes web de calidad de los datos, de la API del componente, de presentación, de impacto social y del proveedor y hosting:

- Factor de calidad de datos. En este factor de calidad se engloban cuatro atributos o criterios: precisión, completitud, actualidad temporal y disponibilidad de los datos del componente. En relación al ámbito de la calidad de los datos, su precisión se considera de mayor importancia que la completitud, frescura o actualidad y disponibilidad. Desde este punto de vista, la disponibilidad tiene menor importancia que la actualidad y mucha menos importancia que la completitud. Por último cabe decir que en comparación con la frescura, la completitud tiene una importancia ligeramente superior a la primera. La matriz quedará descrita del siguiente modo:

	Precisión	Completitud	Actualidad temporal	Disponibilidad	**Autovector**
Precisión	1	3	5	7	**0.56**
Completitud	1/3	1	3	5	**0.26**
Actualidad temporal	1/5	1/3	1	3	**0.12**
Disponibilidad	1/7	1/5	1/3	1	**0.06**
Normalizado por:	176/105	68/15	28/3	16	

Figura 16: Matriz de importancia para el factor de calidad de los datos

Al construir la matriz de importancia, hallaremos los valores del autovector, según el mismo procedimiento seguido anteriormente. De este modo se obtendrán los valores del autovector que aparece especificado en la columna de la derecha, que indicarán el peso de importancia de cada criterio en función del factor de calidad de los datos:

fact_datos = 0.56*critPrec + 0.26*critCompl + 0.12*critActual + 0.06*critDispon

- Factor de calidad de la API. En relación a la calidad de la API, se tienen en cuenta cinco atributos o criterios: funcionalidad, interoperabilidad, granularidad, confiabilidad y usabilidad de la API del componente. En relación a este ámbito, la funcionalidad dispone de una mayor importancia que la interoperabilidad, granularidad, usabilidad y confiabilidad.

  Por otro lado, la interoperabilidad tiene mayor importancia que la granularidad, usabilidad y confiabilidad, y la granularidad es más importante que las dos últimas en ese orden. Por último en comparación con la confiabilidad, la usabilidad de la API dispone de una importancia bastante inferior a la confiabilidad. La matriz se describirá de la siguiente manera:

	Funcionalidad	Interoperabilidad	Granularidad	Usabilidad	Confiabilidad	**Autovector**
Funcionalidad	1	3	5	7	9	**0.50**
Interoperabilidad	1/3	1	3	5	9	**0.27**
Granularidad	1/5	1/3	1	3	3	**0.12**
Usabilidad	1/7	1/5	1/3	1	5	**0.08**
Confiabilidad	1/9	1/9	1/3	1/5	1	**0.03**
Normalizado por:	563/315	209/45	29/3	81/5	27	

Figura 17: Matriz de importancia para el factor de calidad de la API

Hallaremos los valores del autovector al construir la matriz de importancia relativa, según el mismo procedimiento seguido previamente, obteniéndose los valores del autovector que aparece en la columna de la derecha de la matriz, que indicarán el peso de importancia de cada criterio en función del factor de calidad de la API del componente:

fact_API = 0.50*critFunc + 0.27*critInterop + 0.12*critGr + 0.08*critUsab + 0.03*critConf

- Factor de presentación. Este factor se divide en dos criterios de calidad: la usabilidad y la accesibilidad. Desde el ámbito de la presentación del componente, se considera que el criterio de la usabilidad tiene una importancia bastante mayor en un componente cuando se compara con la accesibilidad ya que, aunque la accesibilidad del componente para personas discapacitadas es un asunto vital para que todo tipo de usuarios puedan utilizarlo, si un componente no es usable difícilmente podrá ser adaptado para que sea accesible. La matriz de importancia relativa para este factor de calidad se definirá del siguiente modo:

	Usabilidad	Accesibilidad	**Autovector**
Usabilidad	1	3	**0.75**
Accesibilidad	1/3	1	**0.25**
Normalizado por:	4/3	4	

Figura 18: Matriz de importancia para el factor de presentación

Cuando se cree la matriz de importancia, hallaremos los valores de su autovector asociado, los cuales indicarán el peso de importancia relativa de cada criterio respecto al factor de presentación que está siendo analizado:

fact_presentación = 0.75*critUsabilidad + 0.25*critAccesibilidad

- Factor de impacto social del componente. Es posible distinguir seis criterios para este factor de calidad: valoración de usuario final, seguimientos, descargas y popularidad en las redes sociales de Facebook, Google+ y Twitter. En relación a este factor de calidad, la valoración

de usuario final dispone de una mayor importancia que todos los criterios en el orden en el que se han descrito previamente. Con respecto a los demás criterios, los que disponen de mayor importancia son los seguimientos y las descargas de los repositorios de un componente (en ese orden), seguidos de la popularidad en las tres redes sociales que tienen una importancia relativa similar entre ellas. La matriz de importancia quedará definida de la siguiente manera:

	Valoración	Seguimientos	Descargas	Popularidad Facebook	Popularidad Google+	Popularidad Twitter	**Autovector**
Valoración	1	3	5	7	7	7	**0.32**
Seguimientos	1/3	1	3	5	5	5	**0.25**
Descargas	1/5	1/3	1	3	3	3	**0.13**
Popularidad Facebook	1/7	1/5	1/3	1	1	5	**0.10**
Popularidad Google+	1/7	1/5	1/3	1	1	1	**0.10**
Popularidad Twitter	1/7	1/5	1/3	1	1	1	**0.10**
Normalizado por:	206/105	74/15	30/3	18	18	22	

Figura 19: Matriz de importancia para el factor de impacto social

Hallaremos los valores del autovector que aparecen en la columna de la derecha de la matriz, al construir dicha matriz de importancia relativa, según el mismo procedimiento seguido en casos anteriores, que representan los pesos de importancia relativa de los criterios del factor de impacto social del componente:

fact_impacto_social = 0.32*critValoracion + 0.25*critSeguim + 0.13*critDesc + 0.10*critPopFB + 0.10*critPopG + 0.10critPopTwitter

- Factor de calidad del proveedor y hosting. En lo que respecta a este factor de calidad, se incluyen cinco atributos o criterios: mantenimiento, disponibilidad, rendimiento, seguridad y confiabilidad. En relación a este ámbito, el mantenimiento dispone de una mayor importancia que los otros criterios de calidad.

Por otro lado, la disponibilidad tiene mayor importancia que el rendimiento, la seguridad y la confiabilidad y el rendimiento tiene mayor importancia que los dos últimos criterios. Por último, cabe decir que la seguridad y la confiabilidad referida al no repudio disponen de una importancia similar. La matriz se describirá de la siguiente manera:

	Mantenimiento	Disponibilidad	Rendimiento	Seguridad	Confiabilidad	**Autovector**
Mantenimiento	1	3	5	7	7	**0.50**
Disponibilidad	1/3	1	3	5	5	**0.27**
Rendimiento	1/5	1/3	1	3	3	**0.12**
Seguridad	1/7	1/5	1/3	1	1	**0.08**
Confiabilidad	1/7	1/5	1/3	1	1	**0.03**
Normalizado por:	191/105	71/15	29/3	17	17	

Figura 20: Matriz de importancia para el factor de calidad de la API

Se hallarán los valores del autovector al construir la matriz de importancia relativa, según el procedimiento seguido anteriormente, obteniéndose los valores del autovector que aparece en la columna de la derecha de la matriz, los cuales indicarán el peso de importancia relativo de cada criterio en función del factor de calidad del proveedor y el hosting del componente:

fact_proveedor_hosting = 0.51*critMant + 0.26*critDisp + 0.13*critRend + 0.05*critSeg + 0.05*critConf

Matrices de importancia relativa de los factores con respecto a la calidad global del modelo. Una vez definidas las matrices de importancia relativa de las métricas (respecto a los criterios) y de los criterios (con respecto a los factores de calidad) se deberá determinar la matriz de importancia relativa correspondiente a los factores en relación a la calidad global del modelo de calidad para componentes web. Con respecto a la calidad global del modelo se establece una división de ésta en cinco atributos o factores de calidad: calidad de los datos, de la API, de presentación, de impacto social y del proveedor y el hosting del componente.

En relación al ámbito de la calidad global del componente, la calidad de su API se considera el factor de mayor importancia de los cinco en los que se desgrana dicha calidad global. El segundo factor de más importancia con respecto a los demás es el de la calidad de los datos, seguido del factor de calidad de presentación (con los criterios de usabilidad y accesibilidad, de menor importancia que los criterios de calidad de los datos).

Seguidamente en orden de importancia se encuentra el factor de calidad del proveedor y hosting y por último, se ha determinado que la calidad de impacto social dispone de la menor importancia relativa de todos los factores, debido a la subjetividad introducida por los usuarios finales, principalmente en los criterios relacionados con valoraciones de usuario final y popularidad en las redes sociales más populares. La matriz quedará descrita de la siguiente manera:

	API	Datos	Presentación	Proveedor y hosting	Impacto social	**Autovector**
API	1	3	5	7	9	**0.49**
Datos	1/3	1	3	5	7	**0.26**
Presentación	1/5	1/3	1	3	7	**0.14**
Proveedor y hosting	1/7	1/5	1/3	1	3	**0.07**
Impacto social	1/9	1/7	1/7	1/3	1	**0.04**
Normalizado por:	563/315	491/105	199/21	49/3	27	

Figura 21: Matriz de importancia de los factores respecto a la calidad global

Según el procedimiento de cálculo del autovector de la matriz de importancia relativa, se lleva a cabo una normalización de los valores de cada columna (cada columna de la matriz se divide por la suma de ésta) y se computa la ponderación media por fila para hallar los valores de dicho autovector.

Se ha construido la matriz de importancia relativa y, aplicándose el procedimiento descrito, se obtienen los valores del autovector el cual aparece en la columna de la derecha de la matriz. Los valores del autovector indican el peso de importancia relativa de cada factor de calidad en función de la calidad global del modelo de calidad para componentes web:

calidad_global = 0.49*factDatos+ 0.26*factAPI + 0.14*factPresentacion + 0.07*factProveedorHosting + 0.04*factImpSocial

Una vez finalizado el proceso de asignación del resumen de calidad a un componente por medio del *framework* de anotado podrán aplicarse dichas anotaciones al árbol del modelo de calidad para componentes, de la siguiente manera:

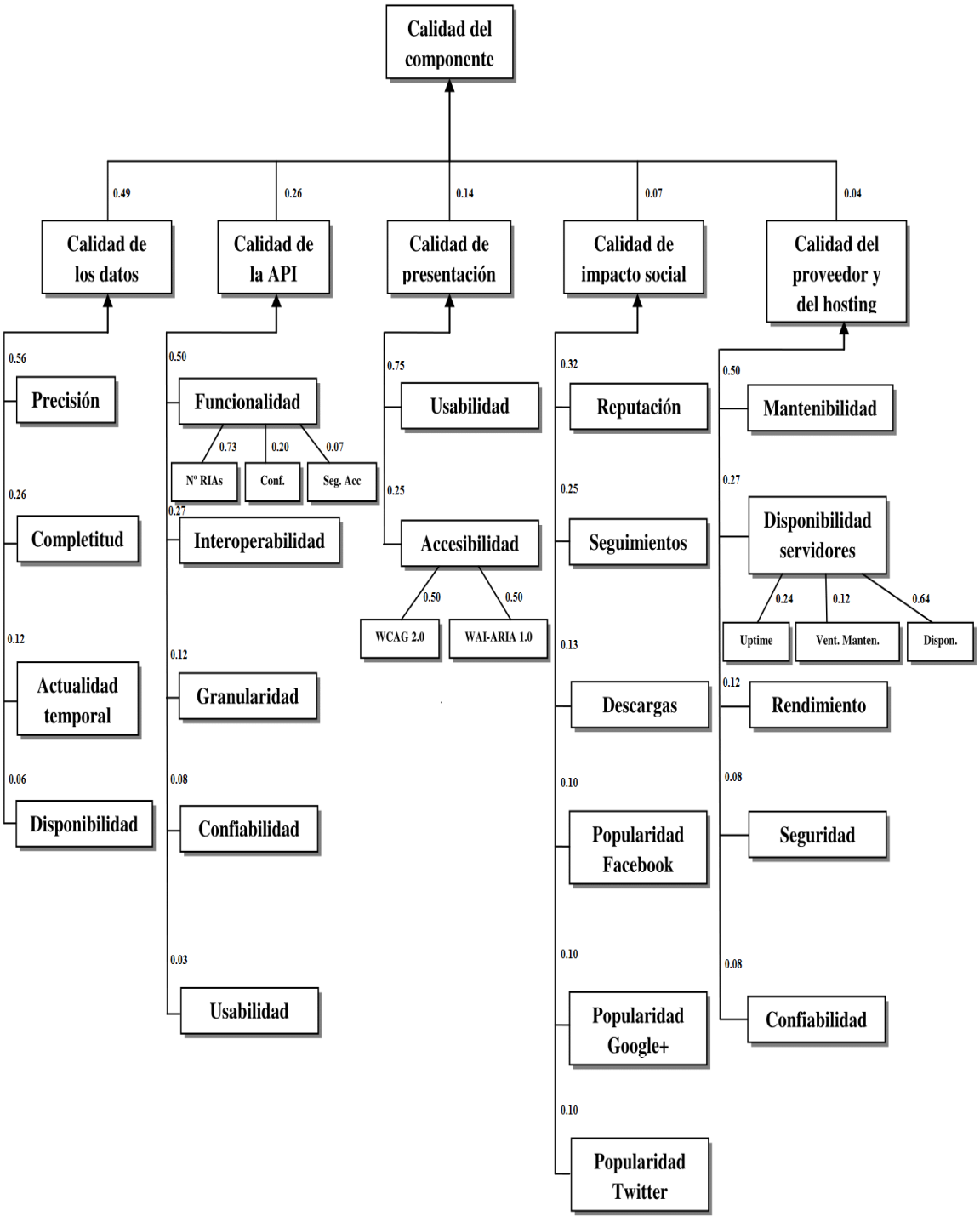

Figura 22: Árbol del modelo de calidad con las anotaciones de calidad agregadas

Al disponer de las anotaciones de calidad en los distintos niveles del modelo, podremos asignar un resumen de calidad a los componentes web que se analicen y se encuentren hospedados en *Polymer Bricks*, además de una descripción de calidad con dicho resumen de calidad detallado en diferentes ámbitos de calidad. Este resumen de calidad dispondrá de una especificación de calidad global para cada componente, así como una descripción de la calidad para cada de los aspectos especificados en el modelo de calidad para componentes web.

Como consecuencia, los usuarios finales podrán consultar esta calidad, tanto global como local a cada uno de los niveles en los que se desgrana el modelo, y decidir qué componentes del repositorio del site son adecuados, según su calidad en distintos ámbitos como la calidad de datos, usabilidad, API o mantenimiento entre otros, para el desarrollo de aplicaciones de *mashups* por parte de dichos usuarios.

Dichos componentes dispondrán de una mayor calidad que ante la ausencia de este marco de caracterización de calidad, por lo que las RIAs creadas por usuarios finales a partir de esos componentes dispondrán de una mayor calidad que otras aplicaciones creadas a partir de la elección de componentes basada en aspectos puramente funcionales sin considerar su calidad.

6.3. Escenario de ejemplo.

Con el fin de demostrar la utilidad del marco de caracterización de métricas de calidad en el desarrollo de RIAs de usuario final, se ha elegido un escenario de ejemplo hipotético cuya implementación se llevará a cabo a futuro en la aplicación de *Polymer Bricks*. Este escenario, relacionado con el ámbito del ocio, consiste en la creación por parte de un desarrollador de *mashups* de una RIA de compra de entradas de cine.

Dicho mashup ayudará a usuarios a comprar entradas de cine en cualquier empresa de exhibición cinematográfica (que provea la compra online de entradas) y en cualquier lugar. Para ello, el mashup necesitará de componentes que variarán desde un componente que incluya las películas que están en cartelera, a otros que incluyan un mapa que sitúe las salas de cine donde se proyecten dichas películas o una pasarela de compra de entradas vía PayPal.

El desarrollador de este mashup debe buscar por la red componentes que se adecúen a sus necesidades de composición, por lo que debe investigar qué componentes son útiles para él en términos funcionales, pero sobretodo también en términos no funcionales de calidad.

Estos términos o características no funcionales, los cuales deben ser maximizados por los desarrolladores de componentes web, se refieren a la interoperabilidad, la usabilidad, la adecuación de sus datos o la conformidad con protocolos de comunicación y formatos de datos estándares entre muchos otros.

La persona encargada de crear este *mashup* deberá llevar a cabo una búsqueda exhaustiva por la red, en busca de repositorios de componentes que no solamente le ofrezcan información sobre características funcionales de los componentes, sino que también dispongan de información acerca de características de calidad del componente en diversos ámbitos.

Es por esto que el repositorio de componentes web del site *Polymer Bricks* es una buena alternativa para la búsqueda de componentes por parte de este desarrollador de mashup.

Dicho site ofrecerá componentes con distintas características funcionales y de calidad en el ámbito de los datos o de la presentación entre otros. Cada uno de estos componentes dispondrá de una valoración de calidad tanto global como en términos de calidad de los datos que ofrece, de interoperabilidad con otros componentes o de usabilidad y accesibilidad, especificada en un descripción de calidad a nivel informativo.

El desarrollador del *mashup* de compra de entradas de cine podrá consultar los componentes del repositorio y sus especificaciones asociadas y estudiar si dichos componentes se adecúan a las necesidades de su composición. Además de consultar las especificaciones de calidad de dichos componentes, cuando el *sandbox* de la aplicación se desarrolle en un futuro, podrá testear dichos componentes antes de integrarlos directamente en el *mashup* que está construyendo.

Para este mashup de compra de entradas de cine, al basarse en la composición de componentes web, publicados muchos de ellos a través de APIs de terceros (como por ejemplo la API de Google Maps), el desarrollador necesita de componentes que maximicen sus cualidades y características externas, es decir, aquellas que radican en la API de cada uno de dichos componentes, ya que éstos se integrarán e interactuarán unos con otros en un ambiente totalmente heterogéneo.

Atributos como la funcionalidad que ofrece un componente, su interoperabilidad con otros componentes dentro de una misma aplicación, o la facilidad de uso de su API son criterios clave en la creación de aplicaciones de *mashup* basadas en la integración de componentes web.

Es necesario que estos componentes dispongan de cierta calidad en los datos que estos proveen a los usuarios, ya que los datos referentes a los cines o a las películas que éstos tengan en cartelera deben ser precisos, actuales y deben disponer de un grado de disponibilidad aceptable.

Por otro lado, los componentes que formen parte de dicho mashup necesitan ser usables para ofrecer una gran experiencia a los usuarios y accesibles, para que aquellas personas con discapacidades de un cierto tipo puedan usar los servicios ofrecidos por esta aplicación.

Con el fin de que los componentes del *mashup* sean susceptibles a un uso correcto por parte de los usuarios que utilizan la aplicación, dichos componentes deben de llevar un cierto grado de mantenimiento por parte de los proveedores de dichos componentes.

Además de este mantenimiento, los servidores que ofrecen el soporte necesario para el funcionamiento del componente precisan de un alto grado de rendimiento, seguridad y disponibilidad para que los componentes a los que dan servicio puedan ser escogidos por el desarrollador del *mashup* objeto de estudio para dicha aplicación de composición.

Por último, cabe decir que, aunque con un grado de menor importancia que los anteriores factores de decisión, las medidas acerca de interacciones sociales de usuarios finales con los componentes web como la valoración por parte de usuarios o desarrolladores que hayan utilizado los componentes, o la popularidad de éstos en redes sociales son un factor a tener en cuenta en la elección de componentes por parte de desarrolladores de aplicaciones de *mashup*.

Teniendo en cuenta todos estos factores de calidad, el desarrollador de la aplicación de *mashup*, con respecto a este escenario de ejemplo, podrá realizar una búsqueda exhaustiva de todos los componentes en el repositorio de la aplicación *Polymer Bricks*, eligiendo aquellos que se adecúen a las necesidades tanto del desarrollador como de la aplicación en cuestión.

Dicha elección estará basada en las especificaciones informativas de calidad especificadas para los componentes del site, siendo estas especificaciones no sólo un resumen global de la calidad de dichos componentes, sino una síntesis de los diferentes ámbitos o factores de calidad de estos elementos.

Por último cabe decir que, gracias a la aplicación del marco de caracterización de métricas de calidad para componentes web de *mashup* a este escenario, el desarrollador de la aplicación de *mashup* de compra de entradas de cine podrá construir una Rich Internet Application (RIA) dotada de una considerable mayor calidad que ante la ausencia de este marco de calidad.

Capítulo 7.

Conclusiones.

Índice

En este proyecto de fin de grado se ha llevado a cabo la realización de un marco de caracterización para componentes web utilizados en desarrollos de usuario final. Se ha realizado un análisis exhaustivo acerca del estado de la cuestión y trabajos previos acerca de las tecnologías y herramientas que existen en el mercado actualmente y se encuentran relacionadas con los componentes web y los desarrollos de usuario final, así como de los repositorios colaborativos con información acerca de dichos componentes y *mashups*. Además también se ha investigado sobre el estado actual de los acuerdos de nivel de servicio o SLAs, debido a que es interesante analizar como los términos de servicio pueden afectar a la calidad que se percibe de los componentes web.

A continuación se han definido dos modelos de calidad, uno para componentes web y otro para aplicaciones de *mashup*, evaluando las características de calidad externas de los componentes web como elementos autónomos y como potenciales participantes de composición de aplicaciones web de *mashup*. Además de esto, se ha desarrollado un *framework* de anotado de resumen de calidad para dichos modelos para la asignación a un componente de una especificación de calidad.

Se ha desarrollado un caso de uso de implementación y aplicación de este marco de caracterización de métricas de calidad para componentes web en una aplicación relacionada con el desarrollo de *mashups* de usuario final.

Por último, este capítulo cierra el trabajo con una serie de conclusiones técnicas y personales, además de un conjunto de posibles líneas futuras de desarrollo del proyecto que puedan dotar a este marco de calidad de una mayor utilidad en el ámbito de los componentes web utilizados en desarrollos de usuario final.

7.1. Conclusiones técnicas.

Al comenzar este proyecto se analizaron las tecnologías relacionadas con los componentes web y las aplicaciones de *mashup* de usuario final. En base a esto, se establecieron los objetivos a cumplir para la creación de un marco de caracterización de calidad para esta tecnología emergente. En este punto de finalización del proyecto, se puede decir que los objetivos se han satisfecho adecuadamente.

Se ha desarrollado un marco de caracterización de calidad completo, compuesto de dos modelos de calidad, uno desde el punto de vista de los componentes como elementos autónomos y otro desde la perspectiva de éstos como potenciales elementos de composiciones de *mashup*, y de un *framework* de aplicación de un resumen de anotado a dichos componentes.

La especificación de los modelos de calidad ha ayudado a desgranar la calidad de los componentes web en niveles de manera jerárquica, para la evaluación de dichos componentes en ámbitos bien diferenciados. Una serie de subdivisiones de la calidad global en factores, criterios y métricas de calidad fueron realizadas para cubrir todas las características de los componentes susceptibles de ser medidas por los modelos.

A continuación, se implementó el *framework* de anotado, encargado de la asignación de un resumen de calidad a los componentes web según las características definidas en los modelos de calidad. Este *framework* es el encargado de llevar a cabo anotaciones de calidad en los diferentes niveles del modelo para asociar, según estas anotaciones, una especificación de calidad a nivel informativo en diferentes ámbitos de calidad para los componentes.

Uno de los problemas encontrados en la realización de este trabajo ha sido la búsqueda de documentación acerca de los modelos de métricas de calidad enfocados en componentes web y usuarios finales, debido a la falta de propuestas de modelos de calidad para componentes web, las cuales son casi inexistentes.

Sin embargo, al tener que realizar un estudio para el establecimiento de métricas de calidad sin ningún trabajo previo para algunos ámbitos o factores de calidad, se han determinado las métricas para estos modelos a través de una investigación exhaustiva de bajo nivel de cada una de las características externas de calidad relevantes acerca de los componentes web y su papel en la composición de éstos para formar aplicaciones de *mashup*.

Todo esto ha llevado a que el estudio de estos ámbitos se haya realizado de una manera completa, siguiendo una metodología *bottom-up*, sin dejar de lado ninguno de los factores, criterios y métricas asociados con las características de calidad externas a los componentes web.

Como conclusión final a este apartado, cabe decir que la realización, tanto de los modelos de calidad como del *framework* de anotado de calidad, tiene una utilidad bastante considerable para escenarios en los que intervengan desarrolladores de *mashups* en la creación de aplicaciones web.

Dichos desarrolladores serán los encargados de llevar a cabo composiciones de componentes en aplicaciones web de *mashup*, analizando y discerniendo qué componentes son los adecuados para dicha composición.

El establecimiento de una especificación de calidad para estos componentes ayudará en gran medida a dicho desarrollador a escoger qué componentes son de mayor calidad para la creación de una aplicación, la cual dispondrá de una mayor calidad que ante la ausencia de dichas especificaciones.

7.2. Conclusiones personales.

Una vez concluido el trabajo de fin de grado, realizado durante cuatro meses de trabajo, mi valoración del trabajo realizado en el CoNWeT Lab es muy positiva en todos los ámbitos.

En un nivel académico, la realización de este proyecto me ha permitido afianzar mis conocimientos acerca de las tecnologías relacionadas con la programación web y adquirir nuevos conocimientos acerca de tecnologías emergentes en la Web 2.0, como es el caso de los componentes web y los desarrollos de *mashups* de usuarios finales. La aplicación de conocimientos teóricos adquiridos en la carrera a un caso de uso de una aplicación web real me ha servido para afianzar dichos conocimientos.

Desde una perspectiva laboral, el ambiente de trabajo ha sido el adecuado. Las reuniones semanales de seguimiento del trabajo con el tutor han sido claves para mantener una comunicación constante con el tutor y desempeñar este proyecto siguiendo los objetivos y plazos establecidos.

La existencia de entregas en el proyecto, tanto intermedias como finales, ha sido vital para llevar a cabo un seguimiento del trabajo según una planificación especificada, con el fin de terminar un trabajo determinado según los plazos establecidos en la planificación de dicho proyecto.

Desde un punto de vista personal, mi valoración del trabajo ha sido enormemente satisfactoria, teniendo la oportunidad de realizar un trabajo con una gran utilidad en un caso de uso real, en un ambiente profesional, afable y colaborativo que desearía encontrar cuando me incorpore al mundo laboral.

7.3. Líneas futuras.

Este proyecto de fin de grado supone una propuesta de un marco de caracterización de calidad para componentes web utilizados en desarrollos de usuario final, mediante la presentación de dos propuestas de modelos de calidad. En relación al primero de estos modelos, éste se presenta desde la perspectiva de los componentes web como elementos autónomos e independientes. Con respecto a la segunda propuesta de modelo de calidad, los componentes son vistos desde un ámbito de composición de elementos para una aplicación web de *mashup*, la cual debe ofrecer características como un valor añadido que no sea la mera adición de los valores que ofrecen los componentes por separado o consistencia en la integración de sus componentes, entre otros.

Con respecto a los modelos de calidad definidos, algunas de las métricas incluidas en ellos son difíciles de transformar en métricas de calidad medibles de manera objetiva, y también automática.

Por ello, algunas de las métricas se evalúan a través de *feedback* proporcionado por los usuarios o mediante un procedimiento en el que se califique como buena o mala una métrica según un procedimiento no del todo objetivo. Una de las líneas de trabajo a realizar en un futuro será el establecimiento de procedimientos formales automatizables y objetivos de medición de estas métricas de calidad mencionadas para dichos modelos.

Otra línea de trabajo muy importante a llevar cabo es la de la implantación de todas las métricas de calidad del modelo de calidad para componentes web en la aplicación explicada en el capítulo relacionado con la implantación del marco de calidad a un caso de uso real, relacionado con el mundo de los componentes web y los desarrollos de usuario final. A través de este marco de caracterización de calidad un usuario final desarrollador de *mashups* podrá construir RIAs de una calidad considerablemente mayor que ante la ausencia de dicho marco.

Este trabajo se centra en el estudio de los componentes web utilizados en desarrollos finales y en una proposición de modelos de calidad para éstos en dicho contexto de uso. Esta propuesta consiste en una aproximación de calidad para componentes en una primera iteración de trabajo, por lo que este marco de caracterización de calidad puede ser sometido en un futuro a modificaciones. Debido a esto, un estudio más profundo puede ser realizado para la definición de nuevos factores, criterios y métricas de calidad o para el perfeccionamiento de las establecidas en dichos modelos de calidad. Se deberá también adaptar la implementación del *framework* de anotado a las nuevas especificaciones de métricas de calidad que se definan en un estudio más profundo en este ámbito o a las modificaciones de éstas que se lleven a cabo.

Con estas últimas líneas concluyo el trabajo realizado durante estos cuatro meses, encontrándome muy satisfecho del trabajo realizado y de mi aportación al mismo, siendo un trabajo de gran utilidad para el establecimiento de calidad para las tecnologías de la Web 2.0, un ámbito en constante evolución.

Bibliografía.

[1] Cinzia Cappiello, Florian Daniel, Maristella Matera. "A Quality Model for Mashup Components" (ICWE 2009, LNCS 5648, pp. 236–250, 2009. Springer-Verlag Berlin Heidelberg 2009)

[2] Cinzia Cappiello Florian Daniel, Agnes Koschminder, Maristella Matera and Mateo Picozzi. "A Quality Model for Mahups" (ICWE 2010 Workshops, LNCS 6757, pp. 137-151–371, 2010. Springer-Verlag Berlin Heidelberg 2011)

[3] Cinzia Cappiello, Florian Daniel, Maristella Matera, Cesare Pautasso, "Information Quality in Mashups" (IEEE Computer Science, 2010)

[4] Mateo Picozzi, Marta Rodolfi, Cinzia Cappiello, Maristella Matera. "Quality-Based Recommendations for Mashup Composition" (ICWE 2010 Workshops, LNCS 6385, pp. 360–371, 2010. Springer-Verlag Berlin Heidelberg 2010)

[5] "Introduction to Web Components"; http://www.w3.org/TR/2013/WD-components-intro-20130606/

[6] "Polymer"; https://www.polymer-project.org/

[7] "Material Design"; http://www.google.com/design/spec/material-design/introduction.html

[8] "Angular JS"; https://angularjs.org/

[9] "Angular UI"; https://angular-ui.github.io/

[10] "x-tags"; http://x-tags.org/

[11] "Bosonic"; http://bosonic.github.io/

[12] "End-User Development"; http://www.interaction-design.org/encyclopedia/end-user_development.html

[13] "Categorías de *mashup*"; http://es.wikipedia.org/wiki/Mashup_%28aplicaci%C3%B3n_web_h%C3%ADbrida%29#Tipos_de_mashups

[14] "Polymer designer"; https://www.polymer-project.org/tools/designer/

[15] "Yahoo Pipes"; https://pipes.yahoo.com/pipes/

[16] "Wirecloud"; http://conwet.fi.upm.es/wirecloud/

[17] "Programmable Web"; http://www.programmableweb.com/

[18] "Github"; https://github.com/

[19] "Bower"; http://bower.io/

[20] “Custom Elements”; http://customelements.io/

[21] “Component Kitchen”; http://component.kitchen/

[22] “Built with Polymer”; http://builtwithpolymer.org/

[23] International Organization for standarization. *Software Engineering — Product Quality — Part 1: Quality Model*. ISO 9126-1:2001 Ginebra: ISO, 1991, 25 p.

[24] International Organization for standarization. *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*. ISO 25010:2011 Ginebra: ISO, 2001, 25 p.

[25] “Google Cloud Platform”; https://cloud.google.com/docs/

[26] “Google App Engine”; https://cloud.google.com/appengine/docs/

[27] “Google Compute Engine”; https://cloud.google.com/compute/docs/

[28] “Google Cloud Storage”; https://cloud.google.com/storage/docs/overview/

[29] “Google Cloud SQL”; https://cloud.google.com/sql/docs/introduction/

[30] “Google Cloud Datastore”; https://cloud.google.com/datastore/docs/concepts/overview/

[31] “Google Big Query”; https://cloud.google.com/bigquery/what-is-bigquery

[32] “Google Cloud Endpoints”; https://cloud.google.com/appengine/docs/java/endpoints/

[33] “Google Cloud DNS”; https://cloud.google.com/dns/docs

[34] “Google Analytics”; https://developers.google.com/analytics/

[35] Nielsen, J.: Web Usability. New Riders (2000)

[36] “Nielsen Norman Group”; http://www.nngroup.com/

[37] “World Wide Web Consortium (W3C)”; http://www.w3.org/

[38] “Introduction to Web Accessibility”; http://www.w3.org/WAI/intro/accessibility.php

[39] “W3C Web Accessibility Initiative (WAI)”; http://www.w3.org/WAI/

[40] “W3C Web Content Accessibility Guidelines”; http://www.w3.org/WAI/intro/wcag

[41] “Web Content Accessibility Guidelines 2.0”; http://www.w3.org/TR/WCAG20/

[42] “W3C Web Accessibility Initiative for Accessible Rich Internet Applications”; http://www.w3.org/WAI/intro/aria.php

[43] T. L. Saaty, “How to Make a Decision: The Analytic Hierarchy Process”, *European Journal of Operational Research*, 48, pp. 9-26, 1990.

[44] “The Magical Number Seven, Plus or Minus Two”; Miller, G. A. (1956). "The magical number seven, plus or minus two: Some limits on our capacity for processing information". *Psychological Review* 63 (2): 81–97. Doi_10.1037/h0043158. PMID 13310704.

[45] “W3C Accessibility HTML Validator”; http://validator.w3.org/

[46] “W3C Accessibility CSS Validator”; http://jigsaw.w3.org/css-validator/

[47] “Site Analyzer”; http://www.goingup.com/analyzer/

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Wed Jan 07 16:56:12 CET 2015
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)