Métricas de Funcionalidad: una taxonomía para sistemas Web

Cristian Bravo Lillo, Luis A. Guerrero
{crbravo, luguerre}@dcc.uchile.cl
Departamento de Ciencias de la Computación
Universidad de Chile

Abstract

Muy poco de lo aprendido en ingeniería de software, se aplica en el desarrollo de sistemas Web. Esto se debe, entre otras cosas, porque hay algunas diferencias en el diseño, desarrollo y mantenimiento de estos sistemas. Una importante parte de un sistema Web tiene que ver con su contenido y su diseño gráfico, además de su funcionalidad. Debido a esto, las métricas de software tradicionales poco sirven en este tipo de sistemas. En el presente artículo se analiza la estructura de un sistema Web desde los archivos que lo componen, con el fin de establecer algunas métricas.

Palabras clave: Ingeniería Web, métricas para aplicaciones Web, aplicaciones/sitios Web.

1. Introducción

El desarrollo y crecimiento del Web ha sido vertiginoso y sin precedentes durante los últimos años, en cuanto a número de usuarios conectados, cantidad de sitios o portales Web y cantidad y tipo de herramientas que permiten construir páginas o sitios Web, con distintos grados de complejidad. Esto ha provocado un significativo impacto en áreas como: negocios, finanzas, entretenimiento, comunicación, educación, gobierno, industria, e incluso en nuestra vida personal y laboral. Existen numerosos lenguajes y tecnologías relacionadas con la programación de aplicaciones que permiten generar una página Web, no sólo del lado del servidor (*server-side*) sino también del lado del cliente (*client-side*). Esta enorme cantidad de recursos, y esta característica dual de las aplicaciones Web (que poseen un conjunto de funcionalidades independientes del lado del cliente y del lado del servidor), dificulta enormemente el análisis y diseño de esta clase de aplicaciones.

Las nuevas tendencias hacia el comercio electrónico, el trabajo en la casa, la transformación de aplicaciones tradicionales a aplicaciones con interfaces Web (*legacy systems*), y la expansión de Internet hacia nuevos servicios y hacia otras áreas como la televisión, los teléfonos celulares (tecnología WAP), e incluso algunos electrodomésticos, hace pensar que este gran auge va a continuar por mucho más tiempo, requiriendo cada vez aplicaciones más sofisticadas, pero con interfaces más claras y fáciles de usar. También es de esperar que se desarrollen más y mejores herramientas para el desarrollo de aplicaciones y páginas Web, que apoyen tanto su funcionalidad como su interfaz gráfica.

Podríamos decir que el actual desarrollo de sitios y aplicaciones Web está caracterizado por cuatro importantes factores: (1) las aplicaciones y sitios Web son cada vez más complejos (gráfica, contenido y funcionalidad), (2) cada vez hay más y mejores herramientas de desarrollo, (3) los tiempos de desarrollo requeridos por las empresas son cada vez más cortos, para estar mejor posicionados que la competencia, y (4) las aplicaciones y sitios Web requieren cambios periódicos de contenido y gráfica, para mantenerse atractivos a los usuarios (mucho mantenimiento).

Estos factores contribuyen a que el actual desarrollo de aplicaciones y sitios Web, sea hecho *ad hoc* para cada proyecto, requiriéndose un gran número de parches y cambios. Murugesan *et. al* sostienen que "*el desarrollo de sistemas basados en Web, carece de rigor, de un enfoque sistemático, de control de calidad y de aseguramiento de la calidad*" [Muru99]. La alta probabilidad de fallas de los sistemas Web construidos hasta ahora (por falta de metodologías más rigurosas), y el hecho de que, al hacerse los sistemas cada vez más complejos, una falla puede ser propagada a muchos lugares a la vez (un error puede ser ejecutado muchas veces en muchos sitios simultáneamente), pueden provocar un quiebre irreparable de la confianza de los usuarios en el Web, causando así lo que ya se ha definido como la *crisis del Web* [Zeln98].

Para evitar una posible crisis del Web, y lograr el éxito en el desarrollo de aplicaciones Web cada vez más complejas, hay una imperiosa necesidad por enfoques disciplinados, y nuevos métodos y herramientas de desarrollo y evaluación de sistemas basados en Web [Muru99].

El enfoque del presente artículo está orientado a la definición de una serie de métricas y de su aplicación a los sistemas Web. Con esto pretendemos establecer una diferencia cuantitativa entre algunos conceptos fundamentales, lo que nos permitirá no sólo estudiar objetivamente la funcionalidad presente en estos sistemas, sino separar claramente qué características son importantes y cuáles no a la hora de diseñarlos y mantenerlos. En la sección 2 se discuten las principales características de un sistema Web, y se define una taxonomía. En la sección 3 se analiza la funcionalidad de un sitio Web, tanto desde el lado del cliente como del lado del servidor. En la sección 4 se clasifican los archivos que componen el sitio Web, según su funcionalidad dentro del sistema. La sección 5 presenta las métricas para sistemas Web. Finalmente, en la sección 6 se presentan algunas conclusiones.

2. Naturaleza de un sistema Web

Términos como *aplicación Web* y *sitio Web* tienes distintos significados para distinta gente. Jim Conallen hace la diferencia entre ellos desde el punto de vista de la modificación de la lógica del negocio. Según este autor, una *aplicación Web* es un *sistema Web* (servidor Web, red, protocolo HTTP, y browser) en el cual el usuario, a través de navegación y entrada de datos, afecta el estado del negocio [Cona99]. De este modo, un *sitio Web* es un sistema Web donde la navegación del usuario no modifica lógica de negocio, o un sistema Web donde no hay lógica del negocio.

Una buena parte de los sistemas Web existentes extraen parte de la información que presentan a los usuarios, desde bases de datos, y ocasionalmente modifican esta información, dependiendo de las acciones del usuario del sistema. A pesar de que existen muchos motores

de bases de datos diferentes, existen cuatro operaciones básicas que son utilizadas para manejar información: crear (*insert*), recuperar (*select*), modificar (*update*) y borrar (*delete*). De estas cuatro operaciones, es común que los sistemas Web utilicen la segunda para desplegar información. Sin embargo, son las otras tres las que modifican el estado de una base de datos (el estado de la lógica del negocio), y es la presencia de estas operaciones, por tanto, la que marca, según la definición de Conallen, la funcionalidad de un sistema: en qué proporción la interacción del usuario con el sistema permite modificar el estado de los datos del sistema.

Sin embargo, esta clasificación de Conallen no contempla la *complejidad* de los sistemas Web, lo cual es de vital importancia en la ingeniería Web, para la construcción de las aplicaciones y para su futuro mentenimiento. Por ejemplo, según la clasificación de Conallen, los motores de búsqueda (*searh engines*) serían catalogados como sitios Web, ya que la navegación e interacción de los usuarios con el sistema no afecta el estado de la lógica del negocio, pues solamente son ejecutadas instrucciones de tipo *select* para obtener la información presentada. Y claramente un *sitio Web* de este tipo es mucho más *complejo* que un portal donde se muestre, por ejemplo, la cartelera cinematográfica de la semana. No obstante, estos dos *sitios Web*, tienen algo en común, y es que la información que despliegan a sus usuarios cambia periódicamente, es decir, el *contenido* del sitio cambia.

Podríamos decir, basados en nuestra experiencia en el desarrollo de sitios y aplicaciones Web, que un sistema Web tiene tres componentes fundamentales: el **diseño gráfico**, el **contenido** y la **funcionalidad** (o código ejecutable). Y es importante hacer esta diferenciación, pues el perfil de la persona encargada de implementar cada componente, es distinto.

De acuerdo con estos tres nuevos criterios, podemos definir una taxonomía para sistemas Web. Los sistemas Web se clasificarían en esta taxonomía, dependiendo de la cantidad y tipo de código que posean, según la cantidad de información que contengan, y según la calidad del diseño. La figura 1 muestra esta taxonomía, compuesta por un sistema con tres dimensiones: contenido, diseño y funcionalidad.

En la taxonomía de la figura 1, la zona *A* determina los sistemas que no contienen código, es decir, que no poseen funcionalidad. Estos sistemas podrían ser llamadas *portales* o *sitios Web*. La zona *B*, por el contrario, define los sistemas que contienen mucha funcionalidad, independientemente del contenido y del diseño. Estos sistemas podrían llamarse *aplicaciones Web*. Pero, ¿qué pasa con un sistema Web que no cae en ninguna de las dos regiones señaladas?, ¿sería considerado un sitio o una aplicación Web?. Retomaremos este punto más adelante.

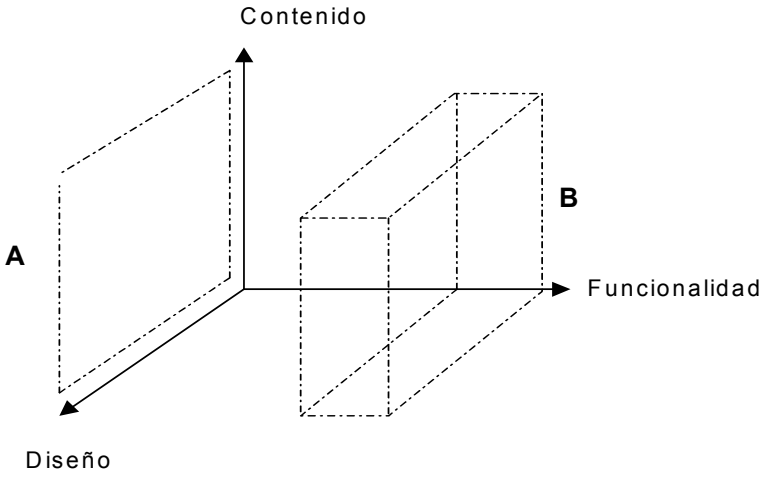

Figura 1. Taxonomía de sistemas Web según diseño, contenido y funcionalidad

A partir de la anterior taxonomía, es posible clasificar los sistemas Web de acuerdo con el tipo de código que poseen. A continuación se exponen las características de estos tipos de funcionalidad y se presentan algunas métricas para medirlos.

3. Tipos de funcionalidad (*server-side*, *client-side* y *embedded*)

A pesar de que, debido a la multitud de recursos existentes, podemos encontrar numerosas variantes de un sistema Web, su arquitectura básica es sencilla. Todo sistema Web está compuesto por un servidor Web y uno o más clientes o browsers. Estos dos componentes *conversan* en un protocolo no orientado a la conexión, llamado HTTP, a través del cual un cliente solicita una página Web al servidor, que busca y envía la página a través de la red hacia el cliente. El cliente interpreta esta página, escrita básicamente en HTML, y la despliega gráficamente. Este esquema, usado en los primeros sitios Web desarrollados, a pesar de ser sencillo, es estático y no permite, en la práctica, presentar funcionalidad importante.

Posteriormente, la funcionalidad de las páginas Web se vio incrementada con el uso de CGI's. Bajo este esquema, la página Web solicitada al servidor era construida en el momento de su solicitud y luego enviada al cliente. El servidor delega esta construcción en otras aplicaciones, las que pueden efectuar numerosas tareas (incluyendo, por ejemplo, consultas en línea a bases de datos) antes de devolver la página construida al servidor para ser finalmente enviada de vuelta al cliente. A los programas que son invocados por el servidor Web para generar dinámicamente estas páginas, se les conoce como scripts de servidor (*server-side scripts*).

Paralelamente, los browsers, o programas encargados de interpretar el HTML y de desplegarlo gráficamente del lado del cliente, fueron evolucionando y haciéndose cada vez más complejos. De esta tendencia surgieron los scripts de cliente (*client-side scripts*), que son ejecutados una vez que la página ha sido recibida completamente por el browser. Actualmente

existen dos tecnologías de este tipo: JavaScript (basado en el estilo de Java) y VBScript (basado en el estilo de Microsoft Visual Basic).

Finalmente, como consecuencia de la notable integración de distintas tecnologías de medios a través de Internet, surgió un tercer tipo de funcionalidad: aquella en que el servidor Web invoca una aplicación externa, en la cual delega incluso el despliegue gráfico de la funcionalidad que la aplicación ofrece. A esta clase de aplicaciones (que podrían considerarse un caso especial de scripts del lado del cliente), usualmente se les llama objetos incrustados o embebidos. Este es el caso de los Applets de Java, y una gran variedad de otros objetos como controles ActiveX y productos de MacroMedia (Flash, Shockwave, Director, etc.), que requieren a veces módulos especiales (o *plug-ins*) para ser ejecutados.

En general, los scripts de servidor se encuentran físicamente contenidos en archivos donde también existe HTML. De la misma manera, un script de cliente puede estar contenido dentro de una página en HTML o estar contenido en un archivo separado, que es referenciado por una página. Incluso es común que, en desarrollos medianamente complejos, sea un script de servidor el que genere un script de cliente. Estas posibilidades entregan mucha flexibilidad, pero hacen muy complejo el diseño y mantención de un sistema Web.

4. Scripts de servidor (server-side scripts)

Normalmente, un servidor Web está asociado a diversas aplicaciones que pueden ejecutar o interpretar scripts del lado del servidor. Estos scripts generan las páginas que finalmente son enviadas al cliente, y pueden ser de tres tipos:

1. **Interpretados**: para los cuales es necesario invocar un *parser* que permita compilar y ejecutar una aplicación de este tipo. Ejemplos de esta clase de aplicación son las tecnologías *PHP* (HyperText Processor), *Microsoft Active Server Pages* (ASP) y *Cold Fusion*.
2. **Compilados**: para los cuales un binario es invocado directamente para generar la página final. Ejemplos de este tipo son las tecnologías Microsoft ISAPI y Netscape NSAPI.
3. **Híbridos**: que son un punto medio entre los dos tipos anteriores. Un ejemplo de esta categoría lo constituyen las *Java Server Pages* (JSP), de Sun MicroSystems. Una página JSP es un script que, la primera vez que es solicitado, es compilado y almacenado en el lado del servidor. Las siguientes peticiones de la página generada por este script son respondidas con la invocación del binario que resultó de la primera compilación.

5. Funcionalidad según tipos de archivos

A pesar de la forma en que pueden estar mezclados los distintos tipos de funcionalidad mencionados anteriormente, es posible determinar con exactitud la forma de un sistema Web, estudiando los tipos de archivos que lo componen. La tabla 1 presenta una clasificación de los tipos de archivos que normalmente se encuentran presentes en un sistema Web.

Tabla 1.Tipos de archivos que componen un sistema Web

Tipo de archivo	Extensiones de archivo	Descripción
Estructurales	Html, ihtml	Archivos que contienen la descripción lógica de las interfaces de los objetos que componen un sistema Web.
Funcionales (servidor)	Php, jsp, asp, pl, cgi	Archivos que implementan funcionalidad de un sistema Web, del lado del servidor.
Funcionales (cliente)	Js, css, vs	Archivos que implementan funcionalidad o apariencia de un sistema Web, del lado del cliente.
Funcionales (incrustados)	Class, swf, dir	Archivos de funcionalidad externa, que pueden ser visualizados a través de una funcionalidad añadida al browser, como applets, plug-in's, etc.
Imágenes	Gif, jpg, bmp	Archivos binarios de imágenes, en varios formatos. A pesar de que existen muchos más formatos de imágenes, sólo estos son actualmente desplegables a través de un browser.
Documentos	Pdf, ps, doc, xls, ppt, rtf	Documentos varios que pueden ser desplegados o *bajados* (*download*).
Otros		Archivos que no encajan en los tipos anteriores.

De acuerdo con la taxonomía anteriormente presentada, podemos decir que el **contenido** de un sistema está relacionado con los archivos estructurales y de documentos que posee, el **diseño** está relacionado con los archivos de imágenes referenciados por un sitio, y la funcionalidad está relacionada con los archivos de los tres tipos antes mencionados. Así, basados en las extensiones de los archivos, es posible determinar su papel dentro del sistema Web y el perfil más adecuado de las personas que deberían construirlos. Por ejemplo, los archivos con extensiones php, asp, jsp, entre otros, son archivos que determinan el grado de funcionalidad del sistema, y deberían haber sido generados por un ingeniero de sistemas o por un programador. Los archivos con extensiones gif, jpeg y bmp (este último sólo en el caso del browser Microsoft Internet Explorer), forman parte del diseño del sistema, y posiblemente fueron generados por un diseñador gráfico. Finalmente, los archivos con extensión html, ps, pdf, doc, entre otros, forman parte del contenido del sistema, y posiblemente fueron generados por una persona con un perfil de diseñador de contenido.

Si se realiza un análisis sintáctico básico sobre los archivos con extensión HTML para separar los diversos tipos de funcionalidad, y si se calculan las sumatorias de los pesos de cada tipo de archivo, según funcionalidad, diseño y contenido, es posible ubicar un sistema Web en nuestra taxonomía. De este modo, si el sistema se ubica en la región A del la taxonomía (ver figura 1), podríamos decir que es un portal o sitio Web. Si el sistema se ubica en la zona B, podríamos decir que estamos ante una aplicación Web.

Sin embargo, muchas aplicaciones no caen en ninguna de estas dos zonas, a pesar de que poseen algún grado de funcionalidad. ¿Deberían catalogarse estos sistemas como aplicaciones Web? Consideramos que, independientemente del "nombre" que le demos al sistema, y desde

el punto de vista de la ingeniería de sistemas aplicada al Web, lo más útil sería determinar el grado de funcionalidad que cada aplicación tiene, así como el de diseño y contenido. Esto sería particularmente útil para dos cosas: (1) determinar con mayor precisión el costo de desarrollo del sistema, desde el punto de vista de los perfiles del equipo de desarrollo, y (2) proyectar con mayor precisión la cantidad y tipo de mantenimiento que le deberíamos dar al sistema. Con estos dos objetivos en mente, nuestro siguiente paso será definir algunas métricas para sistemas Web.

6. Métricas para sistemas Web

Una *métrica de software* es cualquier tipo de medida relacionada con un sistema de software, proceso o documentación relacionada [Somm96]. El típico ejemplo de una métrica de software, es el número de líneas de código.

Las métricas de software se pueden dividir en dos categorías: métricas de control y métricas de predicción. Las *métricas de control* son usadas para controlar el proceso de software. Ejemplos de este tipo de métricas son el tiempo y esfuerzo invertido. La estimación y medida de estas métricas pueden ser usados para mejorar el proceso de desarrollo del software, con lo que se pueden lograr productos finales de mejor calidad. Por otra parte, las *métricas de predicción* son medidas de atributos de un producto, que sirven para predecir cosas sobre el producto. Por ejemplo, se puede predecir la facilidad de mantenimiento de un componente de software midiendo su complejidad ciclomática [McCa76].

Con el fin de poder estimar con mayor precisión el tipo y cantidad de mantenimiento que requiere un sistema Web, hemos desarrollado algunas métricas de predicción basadas en los tipos de archivos que componen el sistema completo. Estas métricas también son útiles para determinar el esfuerzo requerido para desarrollar, a futuro, sistemas con características similares. Las dos métricas se refieren a la funcionalidad del sistema Web, y al diseño gráfico del mismo.

6.1. Índice de funcionalidad

De los archivos incluidos en la clasificación de la tabla 1, para medir funcionalidad nos interesan principalmente los archivos estructurales y funcionales. Si, dado un conjunto de archivos, consideramos sus pesos en kilobytes, podemos definir la siguiente métrica, llamada *Índice de Funcionalidad* (IF):

$$IF_{TIPO} = \frac{\sum peso(archivo\ \ funcional_{TIPO})}{\sum peso(archivo\ estructural) + \sum peso(archivo\ \ funcional_{TIPO})}$$

donde TIPO puede corresponder a uno de los tres tipos de archivos funcionales en un sistema: de servidor, de cliente o incrustada. Dado esto, se define un Índice de Funcionalidad Neto (IFN) como una combinación lineal de los valores anteriores:

$$IFN = \alpha\ IF_{servidor} + \beta\ IF_{cliente} + \delta\ IF_{embebido}$$
$$con \quad \alpha + \beta + \delta = 1$$

Esta última métrica pretende medir (con un valor entre 0 y 1) la funcionalidad presente en un sistema Web y provee mayor precisión para poder hablar de sitios o aplicaciones Web. Si aplicamos la métrica a un sistema Web, según el valor del índice podemos decir lo siguiente:

- Si IFN < 0,2, hablamos de un *sitio Web no funcional*.
- Si 0,2 ≤ IFN < 0,5, hablamos de un *sitio Web funcional*.
- Si 0,5 ≤ IFN, hablamos de *software en Web* o *aplicación Web*.

6.2. Índice visual

Si consideramos los archivos de imágenes y los estructurales de la clasificación de la tabla 1, podemos definir la siguiente métrica, llamada *Índice Visual* (IV):

$$IV = \frac{\sum peso(archivo\ imagen)}{\varepsilon \sum peso(archivo\ estructural) + \sum peso(archivo\ imagen)}$$

Esta métrica permite estimar la cantidad de imágenes que posee un sitio, comparado con la cantidad de estructura que posee. El factor de peso (‘ε’) sirve para equilibrar los pesos de los archivos de imagen y los de estructura, pues los archivos de imágenes son notablemente más “grandes” (en peso) que los archivos HTML que sirven de estructura.

Los valores de los parámetros dentro de estas métricas no han sido determinados aún, y es parte del trabajo futuro a realizar en esta investigación.

7. Conclusiones

En el presente artículo se hizo una clasificación de los distintos archivos que componen un sistema Web. Basados en el tipo de cada archivo, fue posible determinar a qué concepto contribuye cada uno de ellos: **funcionalidad**, **contenido** o **diseño**. Usando esta misma clasificación de los tipos de archivos se definió un índice de funcionalidad y un índice visual. Estos dos índices pueden ser usados como métricas de predicción para hacer estimaciones sobre el mantenimiento de los sistemas, así como para estimar con mayor precisión el costo de nuevos proyectos.

A pesar de haber definido un índice de funcionalidad y un índice visual, no hemos definido aún un índice de contenido. Esto por cuanto no hemos definido aún la forma exacta de medir la información contenida en las bases de datos que pudiesen generar dinámicamente el contenido de un sitio.

Como trabajo futuro debemos precisar los valores de algunos parámetros de los índices, así como el valor del peso de las bases de datos para el índice de contenido. También hace falta realizar un mayor número de mediciones de sistemas Web para ajustar con mejor precisión los parámetros de los índices.

Como parte del trabajo futuro, debemos colocar estas métricas en relación con los actuales sistemas de medición de tamaño de software, siendo nuestra principal preocupación el modelo de puntos por función en sus dos versiones, MKI y MKII.

8. Referencias

[Cona99] Conallen, J. "Modeling Web Application Architectures with UML". Communications of the ACM, Vol.42, No.10, October, 1999, pp. 63-70.

[McCa76] McCabe, T.J. "A Complexity measure". IEEE Transactions on Software Engineering, SE-2 (4), 1976, pp. 308-320.

[Muru99] Murugesan, S., Deshpande, Y., Hansen S., and Ginige, A. "Web Engineering: A New Discipline for Development of Web-based Systems". Proceedings of the International Conference on Software Engineering, ICSE'99, May, 1999, Los Angeles, USA.

[Somm96] Sommerville, I. Software Engineering. Fifth Edition, Addison-Wesley, 1996.

[Zeln98] Zelnick, N., "Nifty Technology and Nonconformance: The Web in Crisis". IEEE Computer, October, 1998, pp. 115-116 and 119.