Reflexiones sobre el *Framework* de desarrollo del *Consejo Superior de Investigaciones Científicas*

Clara Cala Rivero
Directora
Centro Técnico de Informática
(CSIC)

Ángel L. Rodríguez Alcalde
Director de la OPCSIC
Centro Técnico de Informática
(CSIC)

José Ángel Barroso
Desarrollo
Centro Técnico de Informática
(CSIC)

Palabras clave

Framework, J2EE, Java, Arquitectura Orientada a Servicios, Spring, Arquitectura Tecnológica, Web 2.0.

Resumen

El Consejo Superior de Investigaciones Científicas (CSIC) ha definido un marco común para realizar sus desarrollos. Dicho marco representa una ventaja estratégica gracias a su enorme flexibilidad. Tal maleabilidad permite la integración constante de nuevas tecnologías o la incorporación de nuevas tendencias en el cambiante mundo del desarrollo de aplicaciones. La presente comunicación propone una discusión sobre los problemas y las soluciones ofertadas en la evolución de los sistemas de información en plataforma Web del CSIC hacia modelos más participativos, propios de la investigación científica, que confluyen con los postulados de la Web 2.0.

Retos y soluciones planteadas en el esquema general de desarrollo del CSIC

El reto de gestionar el desarrollo corporativo

La eterna discusión sobre la velocidad del cambio tecnológico y la evolución de los Sistemas Corporativos representa la base sobre la que se asientan tendencias contrapuestas en torno a quien representa el motor del cambio.

Tanto las demandas de los usuarios como la propia evolución tecnológica marcan un camino repleto de bifurcaciones que articulan el panorama de la creación de software en cualquier institución.

Desde una dimensión corporativa, la fijación de un marco de desarrollo, parece la única alternativa posible en aras de mantener un criterio coherente sobre el destino a medio y largo plazo de la creación de software. Paradójicamente parece más sencillo establecer criterios generales que especificaciones instrumentales.

Ante la presión de los ciudadanos, Internet y sus marcos de desarrollo se ven sometidos a cambios continuos y radicales, en un modelo autoorganizativo en el que surgen casi cada día especificaciones, desarrollos y tecnologías novedosas, algunas de las cuales se alinean con necesidades y modelos teóricos. Un ejemplo de esto es el paradigma Web 2.0 y su entorno tecnológico.

Definición de una *arquitectura de software*

Entendemos *framework* como una estructura de soporte claramente definida en la que cualquier otro proyecto de software que se emprenda pueda ser organizado y desarrollado. En general, un *framework* incluye soporte de programas, librerías y, entre otras cosas, un lenguaje de *scripting* que se comporta como cohesionador de alto nivel en el desarrollo y la integración de los diferentes componentes de una aplicación.

Desde nuestro criterio, un *framework* representa una *Arquitectura de Software* que modela las relaciones generales de las entidades del dominio. Al tiempo, proporciona una infraestructura y una "manera de hacer" que extiende y/o utilizan las aplicaciones del dominio.

Por qué un framework para el CSIC

El CSIC es una institución cuyo *modelo de negocio* es la Investigación Científica. Aunque su estructura responde a criterios históricos y funcionales, en la actualidad está implicada en cambios profundos cuyo objetivo principal es precisamente, dotarse de mayor flexibilidad para encarar la práctica científica en

la era de la madurez de Internet y en la que la participación (la de los ciudadanos, las de las empresas y la de las instituciones) es la clave del desarrollo y la difusión del conocimiento.

El CSIC es un organismo distribuido, conformado por centros e institutos de investigación repartidos por toda la geografía de España (con dos delegaciones fura de sus fronteras). Al tiempo, este modelo físico, se redefine en un modelo *lógico* de multitud de científicos y grupos de investigación. Esta *arquitectura* permite entrever la complejidad de su gestión (en todos los niveles), todo ello armonizado, como elemento infraestructural, por los diferentes Sistemas de Información Corporativos.

En la actualidad existen pocos teóricos que planteen objeciones a señalar como marco apropiado la indiscutible necesidad de combinar toda o parte de la información existente en los distintos sistemas EIS (Enterprise Information Systems) de la organización, máxime cuando los elementos que generan la información forman una malla distribuida. Las condiciones de sostenibilidad de tal modelo, habitual, por otro lado en cualquier otra gran institución, se centran en la alta disponibilidad, la seguridad, la fiabilidad y la escalabilidad.

Los elementos clave de la solución de la ecuación de esta arquitectura, en tanto hablamos de creación de software y desde una orientación exclusivamente funcional, son tres:

- escalabilidad,
- disponibilidad y
- participación.

La *escalabilidad* es una variable definitiva en un entorno tan distribuido; los desarrollos, generalmente se deben redimensionar en tiempo de producción, precisamente es la estructura del organismo quien demanda esta capacidad. En general, la multiplicidad de Sistemas de Información y la necesidad de gestionar adecuadamente la información hacen que la creación de software deba ser extremadamente flexible. La definición de un framework ayuda a que la integración sea sencilla y por ende su capacidad para crecer.

Por definición, la actividad científica no está sujeta a márgenes temporales rígidos. Si a esto sumamos que la dispersión del personal en colaboración con otros centros de investigación en husos horarios distintos al nuestro es relativamente habitual; la institución debe proporcionar la información de sus sistemas corporativos con *disponibilidad* total. La clave de este punto es la tecnología Web, de modo que un framework orientado en este sentido resulta imprescindible.

Finalmente, el desarrollo de la actividad Científica implica, por definición, un modelo *participativo* de gestión de la información. En la mayoría de los casos, la fuente de los datos se encuentra dispersa por todas las instancias de cualquier institución científica de un cierto tamaño. Para incidir aún más en la problemática los equipos de trabajo son multidisciplinares y en muchas ocasiones se implican

varias instituciones.[1]

Un framework que sea capaz de proporcionar un marco tecnológico dirigido a la participación, o lo que es lo mismo, que permita una orientación Web 2.0 de los desarrollos es inherente al *modelo de negocio* del CSIC.

Desde una óptica tecnológica, las aplicaciones orientadas a la gestión de la información generan una problemática específica del mundo del desarrollo y de la arquitectura de aplicaciones (objetos distribuidos, transacciones, acceso a BBDD, generación dinámica de presentaciones, mensajes, contenedores, seguridad, etc.) que dificultan el desarrollo. En este sentido, la definición de una plataforma que sea capaz de ocultar las enormes complejidades que implica la creación de tales sistemas corporativos adquiere un papel protagonista en el conjunto de la creación de software.

Los elementos tecnológicos esenciales para la selección del framework

En el sentido más amplio posible, en el mercado arquitectónico conviven dos grandes tecnologías J2EE y .NET. El CSIC se ha decantado por la propuesta J2EE.

En la elección de esta tecnología ha contado, primero con el hecho de que Java EE (que es como se conoce la nueva plataforma) es también considerada informalmente como un estándar, puesto que cualquier proveedor debe cumplir una serie de requerimientos de conformidad para poder publicar que un producto es *conforme* con Java EE. Es cierto que no es un estándar ni de ISO ni ECMA, por citar dos de los organismos más importantes.

Además de utilizar un estándar *de facto*, el framework del CSIC cuenta con una serie de ventajas operativas:

- **Orientación a Servicios (SOA)**, La mayor parte de la funcionalidad que proporciona el framework se expone como servicios. Esta visión logra simplificar la integración de las operativas desarrolladas y lo hace de forma independiente a la tecnología empleada para resolverla.
- **Integración de Sistemas**, el framework proporciona las herramientas y componentes necesarios para integrar sistemas, permitiendo una reutilización sencilla de elementos desarrollados por terceros.
- **Extensibilidad** para nuevos servicios y tecnologías. Permite la adaptación de la plataforma a las necesidades de la organización y proporciona un amplio margen para su crecimiento futuro.
- **Adopción de estándares** que aseguren el soporte de mercado presente y futuro.
- **Desarrollo homogéneo y productivo**, gracias a una serie de modelos de aplicación definidos, una metodología de desarrollo y una arquitectura común.

[1] Una parte importante de los centros e institutos de investigación se forman mediante convenios entre diferentes instituciones y administraciones. En el CSIC se habla, en este sentido, de Centros Mixtos.

- **Unificación en el proceso de desarrollo** gracias a una serie de modelos de patrones y productos documentados, herramientas de desarrollo y una arquitectura común.
- **Unificación de plataformas**, permitiendo reducir la heterogeneidad tecnológica, que conlleva aumento de costes y mayor complejidad.
- **Elevada modularidad.**
- **Flexibilidad ante cambios y/o nuevos requerimientos.** Ya hemos visto que esto era una demanda funcional y que tiene un referente técnico que lo hace posible.
- **Minimización del desarrollo**, debido a la existencia de servicios comunes que convierten, en muchos casos, problemas de desarrollo en tareas de parametrización y configuración de servicios ya existentes (acceso a bases de datos, acceso a servicios, gestión XML, correos...).
- **Ocultación de la complejidad tecnológica** por la plataforma, aislando al desarrollador de aplicaciones de la complejidad asociada a la tecnología base; aspecto, este último, que proporciona una mayor productividad.

Los problemas tecnológicos y sus posibles soluciones: experiencias con el framework de desarrollo del CSIC

Cuestiones de desarrollo

En general, podemos decir que la definición y uso de un marco de Infraestructura de Aplicaciones (como hemos definido framework) se diseña y optimiza para resolver la mayor parte de la complejidad de los desarrollos, aprovechando las mejores prácticas y facilitando la construcción de nuevas aplicaciones.

Los puntales sobre los que se realiza todo el diseño de aplicaciones en el CSIC son tres:

- Spring,
- Hibernate y
- JSF

No vamos a comentar los dos primeros ya que el objetivo principal de esta comunicación es destacar los problemas y las soluciones que se han dado a estos en el CSIC. Spring es un framework de código abierto bastante popular. La arquitectura de Spring está basada en el *patrón de inyección de dependencias* (DI) y puede trabajar de manera independiente o con servidores de aplicaciones existentes. Por otro lado hace uso de sus potentes archivos de configuración en XML.

Algunas de sus características que han propiciado esta decisión se basan en la enorme flexibilidad, frente a otras plataformas. Mientras que la integración del

código de servicio en Spring se muestra como parte del interfaz de programación, los desarrolladores de la aplicación tienen la flexibilidad de ensamblar los servicios que necesiten. Esta característica permite crear servidores de aplicaciones personalizados de poco peso. Finalmente el uso más habitual que el CSIC le da a Spring es combinado con Hibernate para proveer servicios de transacción y proporcionar servicios de persistencia.

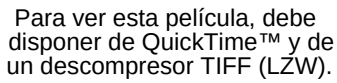

figura 1. Framework separado en capas

Desde el punto de vista más tecnológico, la elección de JSF (más adelante veremos el gran problema que esto puede representar) se planteó a partir de un análisis comparativo con Struts y, aunque pudiera parecer lo contrario, ya en aquel momento (2005), se manifestaron algunas grandes ventajas de JSF.

Debemos decir que JSF puede ser entendido como una evolución de Struts, tanto es así que el propio creador de Struts (C. R. McClanahan) está implicado en el desarrollo de JSF. Es muy importante destacar que JSF es una *especificación*[2] puesto que entendemos como una ventaja el hecho de que cada fabricante pueda diseñar su propia implementación.

JSF tiene como objetivo la normalización y estandarización del desarrollo de las aplicaciones Web. Estas tareas las realiza de forma más eficiente que Struts, en principio porque JSF ha obtenido toda la experiencia de Struts. De aquí ha tomado los dos elementos que nos hicieron decidirnos por una tecnología tan novedosa[3]:

1. *Flexibilidad.* JSF permite la creación de componentes propios y, más importante, es posible *dibujar* los componentes según interese

[2] Las especificaciones que definen JSF son JSR 127, JSR 252 y JSR 276

[3] Una de las primeras comparaciones que se hicieron entre JSF y Struts se puede consultar en el enlace [http://websphere.sys-con.com/read/46516.htm?CFID=61124&CFTOKEN=FD559D82-11F9-B3B2-738E901F37DDB4DE]

creando nuestros propios renderizadores.

2. *Sencillez.* En esto supera de forma muy clara a Struts.

No obstante esto, las propias novedades brillantes, engendran un punto peligroso: JSF no puede competir en grado de madurez con Struts y esto conlleva que la presencia en el mercado de personal suficientemente formado es casi nula, de modo que los proyectos pasan por un plan de formación que genera un *retraso* inevitable en el desarrollo de los proyectos.

Actualmente, JSF empieza a ser una tecnología aceptada en los equipos de desarrollo y es posible encontrar técnicos formados. En cualquier caso, si los desarrolladores conocen cualquier otro MVC, los retrasos se reducen bastante. Pero no puede ocultarse que es después de una cierta experiencia cuando se ven las bondades de JSF.

Desde nuestro punto de vista, si se realiza un análisis de los recursos, quizá no sea una buena decisión para evolucionar un sistema que no ha entrado en la fase de obsolescencia, pero si es la opción recomendada para emprender nuevos desarrollos, máxime si, el equipo de trabajo no tiene experiencia en Struts.

La decisión tomada por el CSIC se orientó hacia la implementación del grupo Apache: MyFaces. Hemos dicho que una de las ventajas es que JSF es una especificación y existen implementaciones de distintos fabricantes. En nuestro caso, se optó por una implementación de OSS ya que la elegida forma parte del Proyecto de Apache. Esta visión sale favorecida del estudio que tome en cuenta el número de componentes que posee, su rendimiento, la información que existe en la red, el coste de adquisición de licencias, etc.

Además de las razones técnicas, la decisión puede parecer aún más arriesgada, pero hemos de insistir en que la Institución es un Organismo Público de Investigación en la que la formación, el intercambio de información y el riesgo son características intrínsecas a su modelo de negocio.

Desde el punto de vista funcional, JSF ofrece algunos de los elementos que nos permiten acercarnos al objetivo de formalizar aplicaciones completamente Web 2.0: la normalización y estandarización de los desarrollos. Como veremos a continucación, en este camino se ha optado por un uso importante de AJAX.

AJAX (***A**synchronous **J**avaScript **A**nd **X**ML)* permite desarrollar aplicaciones web más dinámicas, ricas en interfaz y con una mayor interactividad por parte del usuario.

Mediante esta tecnología se pueden establecer conexiones asíncronas con el servidor, es decir, sin necesidad de recargar la página, rompiendo de esta forma las limitaciones de las aplicaciones web tradicionales. Este es precisamente el punto en que nos encontramos en el CSIC. La demanda de cambios orientados a una mayor participación por parte de las unidades, centros y usuarios ha propiciado un cambio radical y (como se verá en la comunicación *"Intranet del CSIC. Un portal Web 2.0"*).

La popularidad de AJAX tiene su origen en el éxito del modelo de Google: Gmail, Google maps, Google Calentar, etc. En nuestro entorno es utilizada en aplicaciones web de cualquier tipo.

Cuando hablamos de AJAX lo estamos haciendo de un conjunto de tecnologías:

- HTML/XHTML, para estructurar textos y elaborar las páginas web.
- CSS, hojas de estilo
- JavaScript, lenguaje de scripting que en general se ejecuta en el navegador del usuario.

Es en este último lenguaje donde reside la base de AJAX: el objeto *XMLHttpRequest* está encargado de realizar las llamadas asíncronas al servidor.

La elección de estas tecnologías en las aplicaciones corporativas del CSIC, ha permitido aumentar la interactividad del usuario y esto ha redundado en una mejora de la usabilidad. Aunque este era el principal objetivo, hay un elemento técnico no poco importante, ya que redunda en la calidad interactiva de las aplicaciones, esto es, la utilización de AJAX ha aumentando notablemente la velocidad de respuesta de las aplicaciónes.

Veamos brevemente como se ha materializado en una de las aplicaciones que implica a una parte importante de usuarios de toda la estructura organizativa del CSIC: *Gestión de Entidades y personas (GEP)*. En esta aplicación se ha integrado esta tecnología para mejorar el interfaz de forma que el usuario tenga que realizar menos pasos (clicks) por operación.

GEP está desarrollado siguiendo las directrices del framework de desarrollo de aplicaciones del CSIC: Hibernate/Spring/JSF. Se están usando, también los componentes de Tomahawk/Sandbox. Estos se ven apmpliados por una gran variedad de componentes ya que como hemos dicho, el framework del CSIC es un entorno muy rico. Una parte significativa de los componentes están basados en AJAX.

Los componentes de Tomahawk/Sandbox que usan AJAX se basan en la integración con el framework DOJO Toolkit. DOJO es un framework de JavaScript que nos ofrece entre otros muchos instrumentos (controles del interfaz de usuario, efectos, utilidades comunes, etc.) una API para gestionar el acceso asíncrono al servidor.

Aquí radica uno de los problemas a los que ha habido que enfrentarse en los equipos de desarrollo del CSIC: *muchos de estos componentes son muy novedosos e, incluso, algunos de ellos se sitúan aún en la fase de experimentación y desarrollo*. No obstante, en este caso no se ha resentido ni el tiempo de desarrollo ni la dificultad del mismo. La razón principal es que permiten que la integración con AJAX de una aplicación Web JSF sea transparente para el desarrollador e incorporan toda la potencia de DOJO en la aplicación. Todo lo cual favorece el aprendizaje (transparencia) y la calidad (potencia) en los desarrollos.

A continuación presentamos brevemente algunos de los componentes de Tomahawk/Sandbox que se han utilizado en GEP y usan AJAX:

- *InputSuggestAJAX*: Incorpora a una caja de texto (input) de un formulario Web la funcionalidad de mostrar sugerencias de valores según el usuario va introduciendo texto. Las aplicaciones para las que se ha utilizado este tipo de componente son entre otras, las siguientes:
    - Sugerencia en búsquedas: En GEP al enviar un correo electrónico cuando se definen los destinatarios del mensaje el sistema sugiere los correos electrónicos que coinciden con el nombre que está introduciendo el usuario.
    - Cuando existen listas demasiado largas para mostrarlas con un combo seleccionable, al usuario se le presenta la opción de buscar un elemento en la lista de selección en vez de tener que buscar el valor realizando *scroll*.

- *PPRPanelGroup*: permite definir zonas en la página que se recargan dinámicamente. Este componente tiene muchas aplicaciones, en nuestros desarrollos se utiliza, por ejemplo en:
    - La recarga de elementos que dependen los unos de los otros en un formulario.
      Por ejemplo: País ->Autonomía -> Provincia ->Ciudad.
      Otro ejemplo: Aplicación -> Roles por aplicación -> Tipos de roles por aplicación.
      El usuario define el elemento padre y automáticamente y sin salir de la página se van recargando los elementos que tienen dependencia.
    - Mostrar detalles de una lista de resultados sin abandonar la página. Cuando se realiza una búsqueda de personas se presentan los detalles de uno de los elementos (una persona) en la lista de resultados sin salir de la página a partir de un bloque de datos que aparece encima de la lista.

Efectivamente, el uso de estos componentes basados en AJAX también plantea problemas. En primer lugar, en el desarrollo de las aplicaciones se hace necesario un incremento en la *atención* y en la *limpieza* del código por parte del desarrollador, en general por el uso excesivo de JavaScript en las páginas. Así pues, la *sobrecarga* de JavaScript puede representar un problema y necesita una cierta profilaxis en la creación de código en páginas Web.

Por otra parte, la excesiva dependencia de las aplicaciones respecto de JavaScript no es despreciable y puede enrarecer los desarrollos.

Finalmente, entre los problemas más importantes y a los que más atención hay que prestar destacan las diferencias de visibilidad de los resultados entre los distintos navegadores. Es necesario recordar aquí que, el CSIC considera como

corporativos los navegadores IE y Firefox sobre plataformas Windows así como Safari y Firefox sobre plataforma MacOS X.

Es precisamente la diversidad en la plataforma de usuario final la mayor de las demandas para los equipos de desarrollo corporativo. El CSIC es una institución realmente multiplataforma (donde conviven equipos con Linux, Windows XP y 2003 y MacOS X Tiger) por lo que la arquitectura Web es la piedra angular de la realización de los desarrollos.

Cuestiones de integración de sistemas

La base de la arquitectura de servicios del CSIC se formaliza a partir de los siguientes elementos:

- **Oracle 10g** como motor de bases de datos,
- **Liferay Portal**, como portal Web
- **JBOSS** como servidor de aplicaciones,
- **jBPM** para gestionar los flujos y los procesos
- **Acegi** (CAS) para gestionar la autenticación y la autorización y
- **Alfresco** como gestor documental.

Vamos a comentar de forma muy breve, puesto que la base de esta comunicación es la arquitectura de desarrollo, algunos de los aspectos más interesantes surgidos en la fase de despliegue e integración de las aplicaciones desarrolladas en la arquitectura de servicios. Esta fase es la que ha generado la mayor parte de la problemática.

La gran mayoría de los incidentes se producen, en general y para cualquier entorno tecnológico en el momento en que se definen los entornos de preproducción (o producción), es decir, en la puesta a disposición de los usuarios de los desarrollos. En caso que nos ocupa, el CSIC ha necesitado definir un framework de ejecución tan complejo como el de desarrollo.

El primer elemento que genera esta complejidad viene de la necesaria integración entre entornos de carácter comercial y otros provenientes del ámbito del software de fuente abierta.

El software de fuente abierta (OSS)[4], en general, se somete de forma estricta a los estándares, estén estos consolidados o en proceso de definición. Esto tiene un aspecto positivo en que, en la mayor parte de las ocasiones no hemos de preocuparnos de la compatibilidad entre proyectos de OSS y por tanto, tampoco en la integración de los mismos. En el lado contrario, en muchas ocasiones, los estándares abiertos no han sido asumidos por la industria o el hecho de que estén en proceso de definición hacen que, realmente no podamos hablar de estandarización.

En el primero de los grupos encontramos a Acegi, que es quien se ocupa de proporcionar los servicios de seguridad desde Spring. Acegi utiliza el servicio

[4] El CSIC ha tomado una decisión de carácter estratégico que prioriza, en igualdad tecnológica, la opción de OSS.

de autenticación CAS (Central Authentication Service) desarrollado por la Universidad de Yale, con el que una aplicación utilizando Acegi Security participa en un entorno *single sign on* a nivel general de las aplicaciones del CSIC. En este caso la integración entre OpenLDAP, CAS y el portal ha sido compleja, aunque no exenta de dificultad, mediante un portlet de sincronización que gestiona las ACLs y define el lugar en el que se trata la gestión de los usuarios para las aplicaciones.

En el panorama opuesto y también relacionado con el problema de la gestión de usuarios, grupos y permisos, se encuentra la necesaria integración de el gestor documental elegido, Alfresco[5] y la herramienta de portal, Liferay. En este caso, aún en desarrollo, se han localizado los mayores problemas de integración puesto que ambas plataformas utilizan sus propios esquemas de gestión de usuarios, en este caso se ha optado por realizar una integración basada en un desarrollo propio a la espera de la necesaria integración de los dos sistemas[6].

En algunos casos, la integración ha llevado (gracias a la total disponibilidad del código y a su calidad) a la reescritura de componentes enteros de aplicaciones como el caso de jBPM. Esto puede resultar costoso e ineficiente en algunos entornos donde la productividad inmediata es prioritaria. En nuestro caso ha ayudado a mantener un sistema muy flexible que se despliega de forma sutil y eficiente en la arquitectura de ejecución definida.

Conclusión

El marco tecnológico de referencia definido por el CSIC demuestra una gran solidez y cohesión entre sus componentes. Se trata de una arquitectura que soluciona de manera eficiente la integración entre sus capas y demuestra solvencia a la hora de resolver los supuestos comunes de cada capa. Se basa en un modelo que puede proporcionar una mejora en la sencillez de creación de aplicaciones y facilidad de mantenimiento, tanto para aplicaciones departamentales como para aplicaciones Web corporativas. Su objetivo se centra en dotar a los usuarios de herramientas más participativas en la orbita de lo que se ha dado en llamar Web 2.0.

La definición y puesta en marcha de una marco de referencia ayuda a resolver los problemas más graves que aparecen en la realización de aplicaciones en entornos con elementos de riesgo entre sus elementos constitutivos. Cuestiones como el aprendizaje de las nuevos enfoques, la captación de técnicos con formación suficiente o la integración de herramientas en los entornos de ejecución obtienen una más rápida respuesta si bien se consideran puntos a tener en cuenta para tratar de minimizar los riesgos.

---

[5] Después de dos años de utilización, se ha sustituido la herramienta de EMC Documentum, por la más flexible, moderna y sencilla de usar de Alfresco.

[6] Recientemente se ha realizado una conferencia entre los responsables de Liferay y Alfresco (Ontario (CA), 18 de julio de 2006) en la que han estado representados algunos de los proyectos del CSIC así como la propuesta técnica de integración que se ha diseñado para el CSIC.