Proyectos de desarrollo corporativo con Software Libre

Gabriela Sasco[1] y Sergio Machuca[1],

[1] TELEMATICA SRL, Eduardo Acevedo 1622,
11200 Montevideo, Uruguay
{gsasco, smachuca}@telematica.com.uy

Resumen. El desarrollo de aplicaciones corporativas con software libre siempre ha sido considerado en entornos empresariales como un problema complejo.
Entre las causas se pueden mencionar, la poca información de experiencias, y herramientas, la necesidad de integrar las distintas herramientas manualmente, cuando el software propietario tiene todo integrado.
Con las herramientas de software libre, los desarrolladores pierden productividad al tener que dedicar tiempo en investigación y en muchos casos debiendo construir componentes de bajo nivel.
Uno de los puntos críticos en los entornos corporativos es la heterogeneidad e integración con sistemas legados. Existe muy poca documentación de experiencias de integración con entornos legados utilizando software libre, lo que muchas veces hace difícil evaluar la viabilidad de estos emprendimientos.
El presente trabajo resume experiencias de desarrollo e implantación de software en entornos corporativos utilizando herramientas de software libre con grandes componentes de integración con sistemas legados.

Keywords: Software libre, gestión proyectos, interoperabilidad, aplicaciones corporativas.

1 Introducción

La complejidad de la infraestructura tecnológica de TI en las grandes organizaciones ha crecido significativamente en los últimos años. La necesidad de aplicaciones móviles y de Internet requiere nuevos paradigmas de servicios y negocios. Una respuesta rápida a los requerimientos, redunda en una ventaja competitiva para las empresas.

El entorno de desarrollo debe ser capaz de interactuar con sistemas en diferentes plataformas, en particular de servidores de aplicaciones. Por ende, para desarrollar estos sistemas resulta imprescindible resolver la interoperabilidad entre sistemas basados en plataformas heterogéneas.

Los mecanismos de interoperabilidad entre plataformas pueden ser muy variados debido a que deben atender diferentes consideraciones como: variedad de plataformas; requerimientos de seguridad (por ejemplo encripción, puertos de comunicación); requerimientos de administración de los sistemas; diferentes formas de interacción (sincrónica o asíncrona, unidireccional o bidireccional); la existencia de requerimientos sobre la interacción (atomicidad transaccional, niveles de robustez

y performance, compatibilidad con lenguajes u otras plataformas); si se utilizan mecanismos estándares previstos para la interacción u opciones ad-hoc; disponibilidad (por ejemplo tolerancia fallas de componentes); nivel de programación (facilidad de uso, nivel de abstracción); etc.

En las plataformas de desarrollo empresarial todos los componentes, incluyendo los que resuelven los problemas anteriores, están integrados en la herramienta de forma nativa, por lo que los desarrolladores tienen una adecuada productividad.

En las herramientas de software libre, por su parte, los desarrolladores perderán productividad, por tener que dedicar tiempo en investigación y en muchos casos debiendo construir componentes de bajo nivel.

El desarrollo de aplicaciones corporativas con software libre siempre ha sido considerado en entornos empresariales como un problema complejo.

Entre las causas se pueden mencionar, la poca información de experiencias, y de las herramientas, la necesidad de integrar las distintas herramientas manualmente, cuando el software propietario tiene todo integrado.

Uno de los puntos críticos en el entorno corporativo es la heterogeneidad y la integración con sistemas legados. Existe muy poca documentación de experiencias al respecto utilizando software libre, lo que muchas veces hace difícil evaluar la viabilidad de estos emprendimientos.

El presente trabajo resume las experiencias realizadas en 15 años de desarrollo e implantación de Software en entorno corporativo siendo la mayoría en instituciones del estado. Las experiencias que exponemos, fueron realizadas en su totalidad utilizando Eclipse [1]/Java. En este trabajo trataremos de centrarnos en la parte de heterogeneidad e integración con sistemas legados y en aspectos de gestión de este tipo de proyectos.

2 Arquitectura utilizada

La mayoría de las experiencias se basan en el diseño de una plataforma de desarrollo y soporte de aplicaciones que permite la integración con diferentes entornos. Se plantea una infraestructura capaz de brindar todos los servicios de seguridad tales como autenticación, administración de usuarios, confidenciabilidad de los datos de los clientes y seguridad en las conexiones para todos los proyectos de gestión.

Un aspecto importante es lograr que la solución sea independiente de la plataforma en que se desarrollaran las aplicaciones de gestión. Esta plataforma tiene en cuenta la posible heterogeneidad de los sistemas corporativos y aplicaciones legadas.

2.1 Plataforma Base

La plataforma base está compuesta por la herramienta principal de desarrollo y los servidores de aplicaciones. Todos ellos basados en el estándar J2EE [2]. Las herramientas que componen esta plataforma son las siguientes:

JBOSS [3]
JBoss es el servidor de aplicaciones libre por excelencia. Su licencia es LGPL y está implementado al 100% en Java. Es probablemente el servidor de aplicaciones más descargado del mundo.

ECLIPSE [1]
Es la base de la plataforma de desarrollo. Es un framework extensible, open-source e independiente de la plataforma para desarrollar aplicaciones basadas en browser, especialmente las llamadas "aplicaciones rich-client" en oposición al "thin client"

Emplea plugins para proveer todas sus funcionalidades, lo que hace al framework menos pesado y le permite ser extendido.

AJAX (Asynchronous JavaScript And XML) [4]
Es una técnica de desarrollo Web. Permite que las aplicaciones se ejecuten en el navegador del usuario manteniendo en segundo plano comunicación asíncrona con el servidor. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla lo que aumenta la interactividad, velocidad y usabilidad en la misma.

MVC (Model View Controller) [5]
De acuerdo con este modelo el procesamiento se separa en tres secciones, llamadas el modelo, las vistas y el controlador.

Los componentes del modelo, corresponden a la lógica del negocio con la cual se comunica la aplicación Web.

Los componentes de control coordinan las actividades de la aplicación, que van desde la recepción de datos del usuario, las verificaciones de formatos y la selección de un componente del modelo a ser llamado.

Esta separación simplifica la escritura de componentes de vistas y de componentes de modelo: las páginas JSP no tienen que incluir manejo de errores, mientras que los elementos del control simplemente deciden sobre el paso siguiente.

STRUTS [6]
Es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el modelo MVC (Model View Controller) bajo la plataforma J2EE [2].

Posee una configuración del control centralizada y utiliza tablas XML para especificar las interrelaciones entre acciones y página u otras acciones, en lugar de codificarlas en los programas o páginas.

También contiene bibliotecas de entidades para facilitar la mayoría de las operaciones que generalmente realizan las páginas JSP y herramientas para validación de campos bajo varios esquemas que van desde validaciones locales en la página (en javaScript) hasta las validaciones de fondo hechas a nivel de las acciones.

HIBERNATE [7]
Es el puente entre nuestra aplicación y las BBDD, sus funciones van desde la ejecución de sentencias SQL a través de JDBC hasta la creación, modificación y eliminación de objetos persistentes.

La capa de persistencia permite que los desarrolladores no necesiten conocer nada acerca del esquema utilizado en la BD. Solo deberán conocer las interfaces proporcionadas por el motor de persistencia de Hibernate. De esta manera se logra

una separación clara y definida de la lógica de negocios de la aplicación con el diseño de las BBDD.

Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. Los entornos soportados van desde Oracle, DB2, MySql, etc.

JPA (Java Persistence API) [8]

Es una API de abstracción por encima de JDBC. Diseñada con el fin de no perder las ventajas de la orientación a objetos al interactuar con una base de datos relacional

Representa en forma estándar una API para ORM (Object-Relational Mapping)

Persiste en forma transparente un modelo de objetos, con herencia y relaciones complejas. Permite usar objetos regulares (conocidos como POJOs) que son manejados por una entidad nueva llamada EntityManager.

JSF (Java Server Faces) [9]

La tecnología JavaServer Faces establece un estándar para la construcción de interfases de usuario del lado del servidor

Posee un conjunto de APIs para representar componentes de interfaz de usuario y la gestión de su estado, manejo de eventos y validación de entrada, la definición de navegación de la página, y el apoyo a la internacionalización y la accesibilidad.

También incluye una biblioteca de etiquetas personalizadas para expresar una interfaz JavaServer Faces dentro de una página JSP.

RichFaces [10]

El proyecto RichFaces es un framework de interfaz de usuario de componentes avanzados para la fácil integración de capacidades AJAX en las aplicaciones de negocio utilizando JSF.

Mejora áreas de JSF 2, incluyendo el ajuste de performance, la facilidad de uso, los recursos dinámicos y el desarrollo de los componentes. Esto permite a los usuarios sacar el máximo provecho de todas las mejoras de productividad de JSF.

Usabilidad

Desde hace un tiempo, los desarrolladores de aplicaciones Web se han tenido que ocupar de la experiencia del usuario ante el sitio o aplicación Web. Surgen entonces, técnicas y métodos de evaluación y diseño de usabilidad.

No basta con tener aplicativos y sitios que le permitan al usuario cumplir un objetivo puntual de manera eficiente y fácil de usar y navegar. Es necesario proporcionarle una buena experiencia, generarle la necesidad de volver a visitar nuestro sitio

En una paradigma basado en request–response entre navegadores y servidores, lograr esta fluidez y amigabilidad de las aplicaciones es muy costoso sin la ayuda de herramientas como JSF, RichFaces, separación entre capas lograda a través de Struts, etc.

En nuestro caso se trabajo para el establecimiento de estándares y políticas que definían los componentes y comportamientos comunes de las interfases de usuario y como debían ser implementados.

2.2 Servicios de Seguridad

Las aplicaciones, en particular las corporativas y de Internet, que requieren acceder a información sensible de clientes o usuarios, requieren establecer esquemas de seguridad adecuados. Se debe diseñar una infraestructura que garantice el acceso a la información de las personas autorizadas y de procedimientos que aseguren que las aplicaciones no tienen defectos o vulnerabilidades que permiten evadir los controles.

Gestión de usuarios.
Los sistemas que se orientan al uso desde Internet exigen la identificación del usuario, por lo cual es necesario armar una solución de autenticación. Para resolver este punto se optó por usar OpenLDAP [11]:

- Cumple con el protocolo LDAP, el cual se ha convertido en un estándar para manejar usuarios en Internet
- Permite resolver los problemas de heterogeneidad de ambientes e independizarse de la plataforma de desarrollo.
- Es de distribución libre y no tuvo costos asociados.

Repositorio de usuarios.
En algunos casos fue necesario definir una estructura que almacena los usuarios de los distintos servicios ofrecidos y que garantiza la existencia de un usuario único (común) para todos los servicios. Si un cliente ya es un usuario registrado en algún servicio, al solicitar uno adicional, puede usar el usuario mencionado para acceder a las facilidades del nuevo servicio. Así se evita que el usuario deba recordar distintos usuarios y contraseñas para los diversos servicios.

Autenticación.
En los servidores de aplicación, la autenticación se resuelve configurando el servidor apache para el acceso a LDAP. Sin embargo, para algunas aplicaciones que necesitaban de servidores Windows NT, la autenticación se realiza desde la aplicación mediante una DLL que accede a la estructura LDAP.

Delegación.
Es posible implementar una delegación de la gestión de los usuarios de forma que clientes corporativos gestionen las cuentas de sus usuarios.

Para resolver este punto se definió una estructura específica donde se definen usuarios supervisores que pueden delegar la administración de sus cuentas a otros usuarios.

El servicio es brindado por componentes JAVA especialmente diseñados y su ejecución está ligada a los permisos que tenga el mismo.

SSO (Single Sign On).
Además de la necesidad de un único usuario para todos los servicios expuestos, era un requisito que el usuario pudiese presentar las credenciales una única vez y que el ambiente las recuerde. Es por esto que se implementó una solución de SSO basado en CAS (Central Authentication Service) [12].

Es una solución empresarial de SSO. La idea en que se basa es que cada aplicación que interviene, en vez de requerir que el usuario final se autentique a ella, confía en

una autoridad común en donde el usuario inicia sesión, la que se mantiene durante toda la sesión del contexto.

Certificación.
Permiten lograr niveles de seguridad muy altos en las aplicaciones. Esto era un requerimiento importante en los proyectos, pues existen aplicaciones altamente sensibles. El uso de certificados nos permite realizar la autenticación garantizando con los mismos la identidad del usuario (esta facilidad no se está usando actualmente). También se garantiza la confidencialidad de la información y el no repudio de la misma.

Se trabajó con OpenSSL. Es una herramienta de dominio público que permite generar los certificados.

Se creo un conjunto de bibliotecas que permiten las interfaces con openSSL para la generación automática de certificados para el uso en algunas aplicaciones que requerían un nivel elevado de seguridad.

Cumplimiento de seguridad.
Otro aspecto importante que se debió atender es el test de las aplicaciones. Se elaboro un documento de estándares de aplicaciones seguras basado en OWASP (Open Web Applicaction Security Project) [13].

Se trabajó con un grupo experto en tests de penetración y análisis de vulnerabilidades, a fin de detectar y prevenir los diferentes ataques que puede sufrir un sitio (escaneo de SQL Injection, ejecución maliciosa de código, fuga de información a través del mal manejo de errores, robo de sesiones, etc.).

Se estableció un procedimiento que incluye la realización de esos escaneos, previo a la liberación de servicios, que tienen componentes muy sensibles (ej. transacciones que involucran dinero, información de tarjetas de crédito) y la realización de auditorias en forma periódica a las aplicaciones en producción.

3 Integración

En algunas organizaciones existen diversas necesidades de interoperabilidad. La interoperabilidad entre entornos .NET [14] y J2EE es conocida y extensamente estudiada. La forma normal de integración es a través de Web Services [15], [16].

Sin embargo, en grandes organizaciones se observan sistemas IBM zSeries (390) [17], iSeries (AS/400) [18], aplicaciones legadas 3270 y 5250, aplicaciones CICS (Transaction Server for OS/390) o aplicaciones cuyo mecanismo de comunicación previsto es IBM MQSeries [19]. También existen aplicaciones en ambientes UNIX propietarios que tienen características similares y no es simple interactuar con ellas.

La gran heterogeneidad y las restricciones impuestas por esas tecnologías y aplicaciones legadas implican gran complejidad para resolver la interconexión con tales sistemas. Esto hace que la interconexión sea un factor central en los proyectos.

Desde hace algunos años surgió el concepto de SOA [20] que facilita esta integración y de los ESB (Enterprise Service Bus). En la mayoría de las organizaciones, aún no se dispone de infraestructuras basadas en este modelo, por lo que los proyectos deben establecer su integración en forma independiente. Aún en el

caso que existiera un ESB, los proyectos deben encargarse al menos de exponer el nuevo servicio en el mismo y realizar las tareas de integración con los sistemas legados.

A continuación, presentamos una breve reseña de las tecnologías que hemos analizado y trabajado:

Web Services: conjunto de estándares basados en XML (SOAP, WSDL, UDDI), que permite la interoperabilidad entre sistemas. Se caracterizan por permitir conexiones tanto sincrónicas request/reply como asíncronas, conectando aplicaciones no necesariamente J2EE. Utilizan HTTP como protocolo de transporte, lo cual lo hace muy apropiado para conexiones en Internet.

Es el método de integración más común. Actualmente es posible implantarlo en casi todas las plataformas. Sin embargo, en ciertos entornos, se considera muy costoso montar la infraestructura necesaria para darle soporte.

En otros casos, la estrategia que se sigue es la de implementar un "front end" en otra plataforma que brinde los servicios, pero utilice las herramientas que mencionaremos a continuación.

IBM MQSeries [19]: es un transporte genérico de mensajes con un API de alto nivel. Habilita la comunicación entre aplicaciones usando mensajes y colas. Provee una infraestructura middleware de proceso independiente del tiempo y asincrónica. Las aplicaciones integradas a través de "MQSeries" pueden usar una interfase de programación común, "Message Queue Interface (MQI)", la cual es consistente en todas las plataformas soportadas.

Se utilizaron para interactuar con aplicaciones en Windows y 390. Se armó un conjunto de bibliotecas java que facilitaron los desarrollos y brindaron al equipo de desarrollo una interfaz similar a la estándar definida.

EHLL API, HATS, HACL (o productos equivalentes) [21]**:** Son APIs para la comunicación con entornos OS/390. Las dos primeras funcionan en un entorno SNA (OS/390, AS/400) donde la comunicación se realiza por medio de pantallas. Si se desea acceder a los datos de una aplicación, es necesario conectarse con el mainframe, reconocer las pantallas, navegar entre ellas y tratar con los mensajes de error que aparecen en ellas. El contenido de las pantallas es manejado en un array de 24x80 caracteres y se proveen algunas funciones de búsqueda de caracteres y de reconocimiento de campos.

HACL corresponde a una implementación particular mediante clases Java y las terceras corresponden a herramientas que facilitan el desarrollo. A modo de ejemplo con HATS es posible mediante asistentes generar una clase que implemente una consulta determinada, depurarla de errores y luego integrarla a una aplicación.

En nuestro caso, existían en la organización aplicaciones legadas orientadas a terminales IBM 3270, centrales en la gestión de la organización. Debido a estas características no era razonable, en término de tiempos ni costos, cambiar y/o agregar una interfaz que se pudiese consumir de aplicaciones orientadas a la Web. Lo opción que mejor se adaptaba a esta situación, fue utilizar HATS.

Con la ayuda de las herramientas que provee el propio software, se crearon macros que emulaban la navegación por las pantallas de las terminales y la entrada de la información requerida. Luego, los resultados se presentan en aplicaciones Web

realizando un análisis sintáctico de los resultados, traduciendo códigos de retorno y mensajes en textos (de error y de ayuda).

APPC (Advanced Program to Program Communications) y CPI-C (Common Programming Interface - Communications) [22]: protocolos y APIs para comunicación de programas entre diferentes equipos en un entorno SNA (IBM Systems Network Architecture). Son interfaces de programación de un relativo bajo nivel y están disponibles en todas las plataformas (IBM S/390, IBM AS/400, UNIX, MS NT, Linux) para comunicaciones bidireccionales. Definen un conjunto de operaciones que los programas transaccionales pueden usar para comunicarse entre si.

APPC puede ser usado tanto con los lenguajes llamados "viejos" (COBOL, FORTRAN), lenguajes simples (REXX), como de los "nuevos" lenguajes (C++). APPC no requiere un ambiente event-driver ni uno orientado a objetos.

En nuestro caso hemos tenido que interactuar con una aplicación que aceptaba requerimientos con este mecanismo. En una primera instancia se utilizó un componente que implementaba la comunicación con APPC/CPI-C.

CICS [23]: monitor de transacciones difundido en entornos mainframes 390.

CTG (CICS Transaction Gateway) [24]**:** es una implementación de un cliente CICS que puede realizar invocaciones a un servidor CICS remoto. Provee APIS para poder invocarlo desde entornos JAVA y .NET.

Se construyó un WebService que permite acceder a funcionalidades de los sistemas legados en CICS. Este WebService es consumido desde las aplicaciones Web.

En la actualidad, se está analizando este mecanismo para independizarnos de HLLAPI, tendiendo a migrar las aplicaciones que utilizan HATS al mismo.

La implementación de estas interfases, requirió de un grupo de técnicos que incluyo diferentes especialistas, desarrolladores 390, especialistas en CTG y desarrolladores de servicios Web quienes permitieron que la interfaz desarrollada fuese reutilizable.

RSH (remote shell)/SSH (secure shell): Existen situaciones donde se implantan sistemas altamente especializados donde no es posible instalar software adicional y únicamente es posible interactuar con el sistema mediante comandos. En algunos casos es la única opción para la realización de operaciones BATCH.

Estas herramientas permiten el loging y la ejecución remota de comandos. Son apropiados para la ejecución batch de scripts.

Las experiencias incluyeron la ejecución de comandos mediante un encapsulado java en entornos UNIX HP e IBM (AIX pSeries). Este mecanismo fue utilizado en aplicaciones que requerían comunicarse con sistemas de tiempo real y de aprovisionamiento de servicios. La puerta de entrada en estos, eran scripts Unix que en la mayoría culminaban en el acceso a la base de datos de esos servicios. Uno de los problemas con el que nos tuvimos que enfrentar en esta clase de soluciones es la de mantener una operación atómica entre aplicaciones transaccionales pero que debían incluir entre ellas este tipo de acceso. Surge así la necesidad de crear mecanismos por fuera de los sistemas transaccionales. Dependiendo del caso se implementaron diversas soluciones como por ejemplo por cada una de las operaciones crear las operación inversas, colas de operaciones donde se pospone en el tiempo pero se asegura que se ejecuten, etc.

4 Gestión de los proyectos

Queremos hacer un análisis de las consideraciones que deben realizarse al encarar proyectos de desarrollo corporativo utilizando herramientas de software libre. En nuestro caso, los proyectos fueron gestionados según el estándar PMBOK® [25].

Se debió analizar como integrar metodológicamente la formalización dada por el PMBOK del PMI, junto a la experiencia de desarrollar aplicaciones con Software Libre.

Destacamos algunos conceptos de gestión basados en la experiencia personal sobre el tema, en proyectos concretos, donde seleccionamos en cada caso los procesos adecuados para lograr los diferentes objetivos.

Como se mencionó anteriormente existen diferencias entre el desarrollo utilizando software propietario y software libre. En las plataformas de desarrollo empresarial, los componentes se integran en la herramienta de forma nativa, por lo que los desarrolladores se desentienden de varios temas. Con las herramientas de software libre, por su parte, nuestros desarrolladores tendrán que dedicar tiempo en investigación de componentes y en muchos casos debiendo agregar componentes de bajo nivel.

Esta diferencia se traslada al proyecto de tal forma que en vez de generar el proceso de adquisición de la herramienta de desarrollo pasamos a construir la misma.

Optar por una u otra solución, tiene un fuerte impacto en todas las restricciones del proyecto (Alcance, Tiempo, Costo, Recurso, Calidad y Riesgo) y la forma de ser gestionados.

Cuando optamos por Software Libre, surge la necesidad de incluir tareas y entregables para el desarrollo del entorno de trabajo (de investigación y adquisición de componentes, de integración, de adaptación de código existente, etc.).

Como consecuencia, la adquisición del equipo del proyecto, debió contemplar capacidades y habilidades no requeridas en el desarrollo de software propietario.

Entre las capacidades que se requieren se encuentra la de arquitecto, con la habilidad de investigar y diseñar una arquitectura capaz de integrar ambientes heterogéneo e ínter operar con sistemas ya existentes en sistemas 390, as/400, AIX, Linux y Windows.

Esta variedad en los sistemas a integrar, necesita la inclusión en el equipo, de expertos en diferentes tecnologías (experto LDAP, experto MQ, experto en CICS, experto en EHLLAP, programadores JAVA con experiencia en componentización, programadores Java Web, etc.)

Por otro lado, aunque este es un tema general, debimos incluir el perfil de experto en seguridad, debido a que la mayoría de las aplicaciones que analizamos fueron desarrolladas para Internet

Para tratar el tema de usabilidad, necesitamos incluir diseñadores Web, que diseñaran la comunicación con el usuario final y midieran los resultados. Se agregaron a su vez, expertos funcionales, con conocimientos tanto de los objetivos del proyecto como del funcionamiento de los sistemas existentes que bebían se integrados para alcanzar dicho objetivo.

Con el ritmo que cambian las tecnologías, sobre todo en lo que se refiere a software libre, con constante actualizaciones y cambios de tendencia provocadas por la comunidad, es de esperar que los proyectos sufran mas cambios a lo largo del ciclo

de su vida. Es por esto que los procesos de gestión del alcance y gestión de la configuración, necesiten una dedicación mayor.

Estos cambios de alcance o de configuraciones, deberán ser comunicados en forma ágil a los miembros del equipo quien constantemente tendrán que estar enterados del estado de arte, versionado, etc. de los componentes de la solución. Un buen plan de comunicación deberá ser diseñado para este fin.

Otra área que se ve afectada al utilizar software libre es la gestión del riesgo. Existen riesgos intrínsicos al utilizar tecnologías muchas veces inmaduras que obligan a reformular el objetivo de utilizar total o parcialmente herramientas de software libre. Riesgo de tiempo de armado del ambiente de desarrollo.

El solo hecho de integrar sistemas muy heterogéneos, que son el corazón mismo de la organización, introduce un alto riesgo debido a la complejidad que esto tiene.

5 Resumen estadístico

Entre las experiencias que se han resumido, se pueden mencionar las siguientes estadísticas:

- Más de 120.000 usuarios en LDAP
- Más de 70 aplicaciones Web
- Más de 50 aplicaciones que acceden a equipos 390 y AS/400.
- Aplicaciones distribuidas en más de 20 servidores heterogéneos.

6 Conclusiones

Hemos visto que las grandes organizaciones tienen la necesidad de interconectar aplicaciones que en muchos casos residen en sistemas 390, as/400 y Unix. En la mayoría de los casos no es posible utilizar Web Services, que comúnmente se encuentran en .NET y J2EE, por lo que es necesario utilizar diferentes mecanismos y herramientas para realizar la comunicación.

La gran heterogeneidad y las restricciones impuestas por tecnologías y aplicaciones legadas implican una gran complejidad para resolver la interconexión con tales sistemas.

Hemos presentado el resumen de un conjunto de experiencias desarrolladas durante los últimos 15 años que confirman la posibilidad de resolver necesidades corporativas utilizando software libre.

También se presentan algunas de las complejidades que se agregan sobre todo en las etapas iniciales del armado de la solución, cuando preparamos el equipo de trabajo y su ambiente y la complejidad de adaptar e integrar las herramientas de desarrollo.

Se mencionan los diferentes roles y capacidades que deben ser agregados en la ejecución de un proyecto de este tipo (arquitecto investigador, expertos en LDAP, MQ, etc.)

Se da una mirada a como se ven afectadas las áreas y procesos de la gestión de proyecto cuando hablamos de uno que incluye software libre El impacto en la gestión

del alcance y de la configuración, debiendo reflejar en un menor tiempo los cambios en la tecnología utilizada.

Los planes de comunicación, deberán incluir la puesta al día del equipo del proyecto del estado de arte de los componentes utilizados, reflejando no solo los cambios provocados por el proyecto en sí, sino por una comunidad dedicada a enriquecer las bibliotecas que utilizamos de base.

Deben afrontarse riesgos intrínsicos a utilizar tecnologías muchas veces inmaduras que obligan a reformular el objetivo de utilizar total o parcialmente herramientas de software libre. Riesgo en el armado del ambiente de desarrollo, no solo debido a la complejidad en sí de la tarea sino en el alto impacto que esto tiene, por ser la base de nuestro desarrollo.

Referencias

1. Eclipse. http://www.eclipse.org/. Ult. Acc. 2012.
2. J2EE. Sun Web Site http://java.sun.com/j2ee. Ult. Acc. 2012.
3. JBOSS. http://www.jboss.org. Ult. Acc. 2012.
4. AJAX. http://es.wikipedia.org/wiki/AJAX. Ult. Acc. 2012.
5. MVC. http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/app-arch/app-arch2.html. Ult. Acc. 2012.
6. STRUTS. http://struts.apache.org/. Ult. Acc. 2012.
7. HIBERNATE. http://www.hibernate.org/. Ult. Acc. 2012.
8. JPA. http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html. Ult. Acc. 2012.
9. JSF. http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. Ult. Acc. 2012.
10. RichFaces. http://www.jboss.org/richfaces. Ult. Acc. 2012.
11. OpenLDAP. http://www.openldap.org/. Ult. Acc. 2012.
12. CAS - (Central Authentication Service). http://www.cas.org/. Ult. Acc. 2012.
13. OWASP. https://www.owasp.org/index.php/Main_Page. Ult. Acc. 2012.
14. Microsoft .NET. Microsoft Web Site http://www.microsoft.com/net/. Ult. Acc. 2012.
15. Web Services. W3C Web Site http://www.w3.org/2002/ws/. Ult. Acc. 2012.
16. Ruggia, Besil, Pais, Sande; Interoperabilidad entre Servidores de Aplicaciones Heterogéneos, CITA (Congreso Iberoamericano de Telemática 2003)
17. IBM zSeries. IBM Web Site. http://www-1.ibm.com/servers/eserver/zseries/
18. IBM iSeries (AS/400). IBM Web Site http://www-1.ibm.com/servers/eserver/iseries/software/. Ult. Acc. 2012.
19. IBM ®MQSeries. IBM Web Site http://www-306.ibm.com/software/integration/wmq/. Ult. Acc. 2012.
20. SOA. http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios. Ult. Acc. 2012.
21. HATS. http://www-01.ibm.com/software/Webservers/hats/. Ult. Acc. 2012.
22. Guía del programador para APPC. IBM Communications Server para Linux.
23. CICS® Transaction Server for OS/390®. CICS External Interfaces Guide, Release 3. Document Number SC33-1944-32
24. CTG (CICS Transaction Gateway): http://www-01.ibm.com/software/htp/cics/ctg. Ult. Acc. 2012.
25. PMI, PMBOK®. http://www.pmi.org/. Ult. Acc. 2012.