Integración de Aplicaciones*

Rafael Z. Frantz (1), Rafael Corchuelo (2)

(1) Universidade Regional do Noroeste do Estado do Rio Grande do Sul
São Francisco, 501. Ijuí 98700-000 RS (Brasil)
rzfrantz@unijui.edu.br
(2) Universidad de Sevilla, ETSI Informática
Avda. de la Reina Mercedes, s/n. Sevilla 41.012 (Spain)
corchu@us.es

Resumen La integración de aplicaciones es actualmente uno de los grandes retos de la Ingeniería del Software. Según un reciente informe de IBM, los gastos de integración superan en una proporción de 5 a 20 dólares los de desarrollo de nueva funcionalidad. En este artículo esbozamos los fundamentos de una herramienta para el desarrollo de soluciones de integración. Nuestro objetivo es tan sólo presentar los conceptos fundamentales; a partir de esta propuesta continuaremos trabajando con el objetivo de formalizar cada uno de los bloques de construcción presentados de forma que se puedan implementar y utilizar para construir soluciones de integración. Creemos que este trabajo puede ser innovador puesto que hasta ahora las soluciones que conocemos tan sólo ofrecen patrones de diseño intuitivos.

1. Introducción

Hoy en día no es difícil encontrar empresas que estén ejecutando una cantidad muy grande y variada de aplicaciones en un entorno distribuido para llevar a cabo su negocio. Estas aplicaciones suelen ser paquetes de software comprados de terceros, hechas a medida para solucionar un problema especifico o aplicaciones heredadas. Tal heterogeneidad hace, muchas veces, con que unos procesos de negocio de la empresa tengan que utilizar dos o más aplicaciones. En nuestra experiencia, es habitual que estas aplicaciones no están preparadas para interactuar entre sí automáticamente. Así que conocer las diferentes aplicaciones, meter y llevar datos de una a otra y ejecutar funcionalidades en cada una en separado, es tarea de los usuarios, aunque tengan que hacer trabajo doble. Esto suele pasar cuando por lo menos una aplicación involucrada en un proceso no fue diseñada para trabajar integrada. En las empresas también es muy frecuente la necesidad de añadir funcionalidades nuevas a las aplicaciones ya existentes, lo que en muchos casos puede resultar prohibitivo. Así que, en este caso, hay dos posibilidades: desarrollar una nueva aplicación con todas las funciones actuales

* Proyecto IntegraWeb (CICYT TIN2007-64119, JA TIC-2602), Evangelischer Entwicklungsdienst (EED)

y añadir las nuevas deseadas o desarrollar otra solamente con las nuevas funcionalidades e integrarlas. La primera opción suele ser muy costosa, la segunda exigirá proyectar una solución de integración que ofrezca al usuario una visión de más alto nivel con la que interactuar.

Al hablar de de integración, hay que tener en cuenta algunas restricciones para que una solución de integración sea viable para las empresas. La primera restricción es que después de hacer la integración, las aplicaciones involucradas no deben cambiar. Un cambio en una de estas aplicaciones podrá afectar profundamente o incluso invalidar totalmente la solución de integración. De acuerdo con un estudio publicado reciente de IBM [1], por cada dólar gastado con el desarrollo de una aplicación, el coste para integrarla es del orden de 5 a 20 veces más. La siguiente restricción es que, después de integradas, las aplicaciones deben mantenerse desacopladas las una de las otras como antes de la integración. La solución de integración no debe cambiar las aplicaciones involucradas generando dependencias en ellas que antes no existían. Finalmente podemos añadir una tercera restricción según la cual la integración no debe ser hecha como parte del proceso de desarrollo de sistemas, sino conforme sea necesario.

La solución de integración puede estar fundada en una integración de los datos de las aplicaciones, a partir de un esquema de datos global, o en un flujo de llamadas funcionales entre las aplicaciones por medio de APIs. Así que esta solución de integración puede ser vista como una nube operativa y/o declarativa. La vista es operativa cuando representa una visión funcional, con un flujo de datos y una API de bajo nivel para la integración. A esta visión llamaremos Enterprise Application Integration (EAI). Si la visión es de un esquema central de datos ofreciendo una API de consulta a alto nivel, entonces la llamaremos Enterprise Information Integration (EII). La frontera que hay entre EAI y EII, no obstante, suele ser muy difusa.

En este artículo trataremos de integración de aplicaciones con sistemas de mensajería. En [2], los autores proponen varios patrones que ayudan a diseñar una solución de integración de este tipo. Muchos de estos patrones no son más que una idea abstracta para resolver, de una buena manera, un problema recurrente de software. No obstante, creemos que pueden convertirse en algo mucho más concreto, en bloques de construcción que pueden ser, directamente, utilizados/arrastrados al diseñar la solución de integración. Así que no los llamaremos patrones, si no Building Blocks.

El resto del artículo se organiza así: en la sección 2, presentamos los sistemas de mensajería para integración de aplicaciones, así como los posibles niveles y vistas para una solución de integración; en la sección 3 presentamos la jerarquía de los Building Blocks; en la sección 4 un ejemplo de integración utilizando Building Blocks y finalmente nuestras conclusiones.

2. Integración usando sistemas de mensajería

Para empezar esta discusión, definiremos brevemente lo que es un sistema de mensajería. Podemos decir que un sistema de mensajería es un sistema encargado

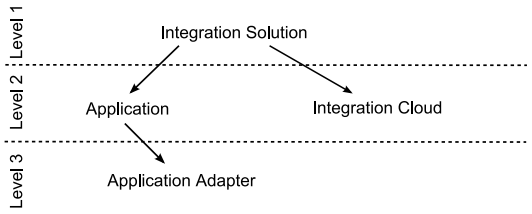

Figura 1. Niveles y vistas de la solución de integración

de administrar mensajes que tendrá que recibir y enviar a algún destino por medio de canales de comunicación. Tal como ocurre en las bases de datos, también debe haber para los sistemas de mensajería una persona encargada de administrarlos, por ejemplo, creando y configurando dichos canales de comunicación. Este estilo de integración de aplicaciones implica directamente que las aplicaciones involucradas en la solución de integración no conozcan explícitamente y no puedan acceder directamente a otra(s) aplicación(es), sino que deben hacerlo siempre por medio del sistema de mensajería.

Las principales ventajas de una solución de integración con sistemas de mensajería son: el bajo acoplamiento y la posibilidad de tener una comunicación asíncrona entre las aplicaciones integradas. En este caso entendemos por acoplamiento el conocimiento que una aplicación involucrada en la solución debe tener con respecto a las demás. Cuanto más conocimiento necesita una aplicación, decimos que más acoplada está, y por lo tanto más dependiente y vulnerable a los cambios de las demás. Cuando una aplicación es integrada por una solución diseñada con sistemas de mensajería, apenas suele conocer, o incluso puede ni conocer, la solución de integración. La aplicación conocerá la solución de integración si tiene una capa de software que le permita acceder el sistema de mensajería. No tendrá tal conocimiento si es una aplicación que fue desarrollada sin tener en cuenta la integración y ofrece como puerta de entrada solamente su interface GUI. La solución de integración no necesariamente debe ser implementada con un único sistema de mensajería y ejecutarse en un único ordenador, sino que puede estar compuesta de varios sistemas de mensajería distribuidos. La comunicación asíncrona permite que una aplicación pueda enviar un mensaje a otra sin que la aplicación destino tenga que estar lista para recibirlo. Esto significa que cuando tal aplicación termine, por ejemplo, de ejecutar lo que esté haciendo pueda acceder al canal y recibir el mensaje. Así que una no necesita aguardar la otra para ejecutar su tarea, el sistema de mensajería se encarga de recibir, transmitir y mantener el mensaje en un canal hasta que su receptor esté listo para recibir y procesarla.

2.1. Niveles de la solución de integración

De acuerdo con nuestra visión sobre integración de aplicaciones, una solución de integración se puede dividir en, por lo menos, tres niveles y cuatro vistas. Sobre los niveles distribuimos las siguientes vistas de la solución: Solución de

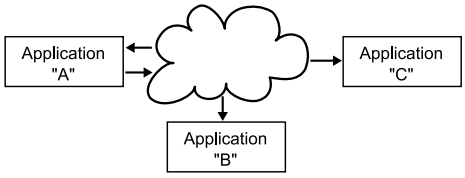

Figura 2. Vista de la solución de integración

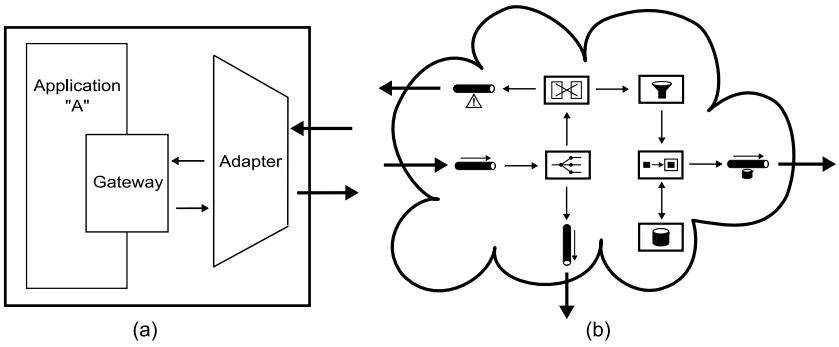

Figura 3. Ejemplo de aplicación y de la nube de integración

Integración, Nube de Integración, Aplicación y Adaptador de Aplicación. Estos niveles y vistas son lógicos, y los proponemos para ayudar a entender y diseñar una solución de integración. En la figura 1 localizamos las vistas en sus posibles niveles de la solución y a continuación describimos con más detalles cada uno de ellos.

Solución de Integración: El nivel 1 es el más alto y por lo tanto el más abstracto de los tres. En este nivel proponemos, tan solo, representar las aplicaciones involucradas en la solución de integración así como los flujos de entrada y/o salida entre la aplicación y dicha solución. La solución de integración se representa por una nube, que llamamos "Nube de Integración". En la figura 2 se puede ver la solución de integración de tres aplicaciones: application "A", application "B", y application "C". La aplicación "A" tiene tanto un flujo de entrada como otro de salida con la nube de integración y las aplicaciones "B" y "C" tienen tan sólo un flujo de entrada a partir de la solución de integración.

Nube de Integración: Esta vista presenta los Building Blocks que ejecutan gran parte de las tareas necesarias para la integración, además de ser la vista más destacada de una solución de integración. En la figura 3 (b) vemos dichos Building Blocks que existen por debajo de la nube de integración. Tenga en cuenta que para recibir o enviar datos a las aplicaciones involucradas hay canales de comunicación. Dichos canales conectan los Building Blocks internos y específicos

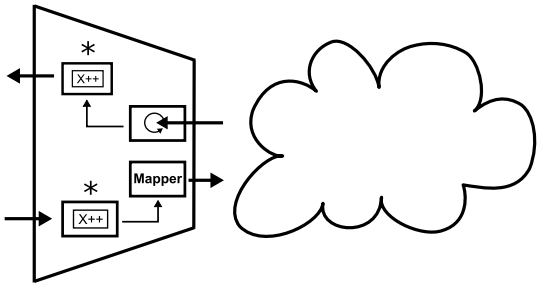

Figura 4. Ejemplo de adaptador de aplicación

de la nube de integración con las aplicaciones. Más adelante presentaremos con detalles estos Building Blocks.

Aplicación: Si fuera posible hacer clic en la aplicación "A", de la vista de la solución de integración, veríamos los detalles presentados en la figura 3 (a). En esta vista ya podemos ver que además de la aplicación hay dos otros componentes: el Gateway y el Adapter. Definimos el Gateway como una capa de software que pertenece exclusivamente a la aplicación y que la permite enviar o recibir datos a/de un sistema de mensajería por medio de un Adapter. El Gateway es para el Adapter la interfaz de comunicación con la aplicación y se representa por una API bajo nivel o por la propia GUI de la aplicación. El Adapter es la capa de software que permite hacer toda la comunicación con el sistema de mensajería. Lo consideramos parte de la solución de integración, al contrario que el Gateway. Por lo tanto, al diseñar una solución de integración se suele diseñar, un Adapter, para cada aplicación. Puede haber situaciones en que la aplicación ya tiene un Gateway desarrollado especialmente para acceder sistemas de mensajería directamente, así que en este caso específico no tendremos que diseñar para ella el Adapter. Pero hay que tener en cuenta que esto no es lo que suele pasar cuando hablamos de integración de aplicaciones.

Adaptador de Aplicación: En el tercer nivel está la vista del Adaptador de Aplicación y es aquí donde se puede ver de que está compuesto el Adapter. La figura 4 presenta sus detalles; llamamos a sus bloques internos Adapter Blocks. Son ellos realmente los responsables por la funcionalidad del Adapter. Estos bloques no son más que un conjunto especial de Building Blocks utilizados en el Adapter, donde unos de ellos, además de ejecutar procesos internos, permiten comunicarse con los demás bloques de la nube de integración o con el Gateway de la aplicación. En [2] estos bloques se llaman de Endpoint. El Adapter suele tener también, además de los Adapter Blocks propuestos en [2], código ad-hoc escrito en un lenguage/tecnología específica. La razón es que para comunicarse con el Gateway de la aplicación o ejecutar algún proceso interno y específico del Adapter, tendremos que escribir código. Así que los bloques de la figura 4

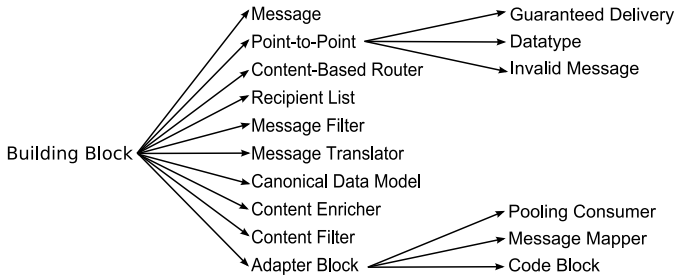

Figura 5. Jerarquía de los Building Blocks

marcados con asterisco, representan un tipo específico de Adapter Block del que no se habla en [2]. Son bloques de muy bajo nivel y los llamamos Code Block.

3. Clasificación de los bloques de construcción

Clasificamos los Building Blocks en dos grupos: aquéllos utilizados para diseñar el flujo de la solución de integración, representado en la vista de la Nube de Integración y aquéllos utilizados en el Adaptador de Aplicación. Hay un Building Block básico y esencial para cualquier solución de integración por medio de Messaging, que llamase Message. Así que antes de presentar los bloques específicos de cada grupo vemos la diferencia entre Messaging y Message. En la figura 5 presentamos la jerarquía completa de los Building Blocks que describimos más adelante.

Mientras Messaging es la tecnología (por ejemplo, Java Message Service (JMS), Microsoft MSMQ o WebSphere MQ) que permite a dos o más aplicaciones comunicarse de forma asíncrona teniendo en cuenta un bajo acoplamiento y una transmisión fiable (store-and-forward), Message no es más que una estructura de datos pasiva que viaja de una aplicación a otra por medio de otros Building Blocks. Message puede representar un comando, una descripción de un evento o sencillamente una información, y está compuesto de dos partes: cabecera y cuerpo. La cabecera contiene meta-información que suele ser utilizada por el sistema de mensajería para saber, por ejemplo, de quién es y a quién debe entregarse el mensaje. El cuerpo contiene la información (dato) transmitido y es ignorado por el sistema de mensajería [2].

3.1. Bloques de la Nube de Integración

Los bloques de la nube de integración están agrupados en: Channels, Routers y Message Transformations. El primer grupo contiene varios tipos de canales que permiten a los productores de mensajes (aplicaciones o los demás bloques) escribir y a los consumidores leer los mensajes del canal. Aquí presentaremos los canales Point-to-Point, Guaranteed Delivery, Datatype y Invalid Message. Ya en el grupo de los Routers están los bloques de construcción con los cuales es

posible cambiar/decidir la ruta de un mensaje, así que presentaremos el Content-Based Router, el Recipient List y el Message Filter. Finalmente hablaremos de los Message Transformations, los cuales pueden cambiar el contenido de un mensaje: Message Translator, Canonical Data Model, Content Enricher y Content Filter.

Point-to-Point Channel: Es un canal muy sencillo que recibe mensajes de uno o más productores y puede contener uno (lo más normal) o más consumidores. La característica más importante de este canal es que garantiza que solamente uno de los consumidores recibirá el mensaje.

Guaranteed Delivery Channel: Podemos decir que este y los siguientes canales son una especialización del Point-to-Point. La diferencia entre el Point-to-Point y el Guaranteed Delivery es que este último garantiza la entrega del mensaje aunque el sistema de mensajería tenga problemas. Es este canal el que permite hacer lo que antes llamamos "store-and-forward".

Datatype Channel: Este canal es útil en situaciones en que el productor debe enviar un determinado tipo de mensaje al canal con la garantía de que los consumidores conocen tal tipo y podrán procesarlo. La diferencia con el Point-to-Point es que aquí hay solamente un tipo de mensaje en el canal.

Invalid Message Channel: Hay situaciones en las cuales se recibe un mensaje pero no se puede procesar por alguna razón. Por ejemplo, llega un mensaje en formato texto pero en realidad se aguardaba un mensaje binario. Entonces lo que el receptor debe hacer, en estos casos, es mover el mensaje incorrecto a un Invalid Message Channel. Así que dichos canales tendrán mensajes que representan problemas en la solución de integración, siendo por lo tanto una especie de "log" de la solución.

Content-Based Router: Este Router tiene la capacidad de recibir un mensaje, examinar su contenido y hacer el debido encaminamiento del mensaje a solamente uno de sus consumidores conocidos. Es un Building Block que ayuda disminuir la cantidad de canales de la solución, aunque puede requerir un mantenimiento frecuente.

Recipient List: Tiene una funcionalidad semejante al anterior, pero es diferente por permitir encaminar una copia del mensaje recibido a todos los consumidores interesados en ella. Una forma de conseguir esto es haciendo que los mensajes incorporen de forma explícita la lista de destinatarios; otra es que este Building Block la calcule usando reglas o accediendo a alguna base de datos.

Message Filter: Lo utilizamos siempre delante de una aplicación o de otro Building Block con el objetivo de evitar mensajes indeseados. Así que basado en un cierto criterio configurado en el Message Filter, un mensaje puede ser fácilmente rechazado.

Message Translator: Las aplicaciones involucradas en una solución de integración suelen utilizar internamente formatos diferentes de datos. Así que cuando tengan que comunicarse hay que hacer la traducción de un formato para otro. Esta traducción puede ser hecha en la solución de integración por un Message Translator.

Canonical Data Model: La idea de haber un traductor entre dos aplicaciones es buena, pero en soluciones de integración con varias aplicaciones en que una tiene que comunicarse con varias otras, si seguimos esta idea tendremos demasiados traductores. Además, si una de estas aplicaciones sufre un cambio en su formato de datos, tendremos que cambiar todos traductores. Diseñando un Canonical Data Model para la solución de integración, tendremos solamente un Message Translator para cada aplicación. Este hará la traducción del formato de datos de la aplicación para el modelo canónico de la solución de integración. Así que cada mensaje ahora es transformado dos veces (formato de la aplicación/modelo canónico/formato de la aplicación), y no más solamente una, pero la cantidad de transformadores será mucho menor en dichas soluciones.

Content Enricher: El objetivo de este traductor es añadir datos al mensaje. Puede que el mensaje no contenga toda la información necesaria, entonces, por ejemplo, con el Content Enricher es posible buscar tal información en alguna fuente de datos externa, en el ambiente de ejecución de la solución, o aún computar dicha información desde el mensaje original.

Content Filter: Algunas veces es necesario quitar de un mensaje ciertos datos, así que esto es lo que hace el Content Filter. El objetivo es quitar datos de un mensaje para simplificarlo ya que no son de interés para los próximos bloques del flujo de integración, reducir el tráfico de la red o incluso por cuestiones de seguridad.

3.2. Bloques del Adaptador de Aplicación

El Adaptador de Aplicación contiene en su interior un tipo especial de Building Block, que llamamos Adapter Block. El principal objetivo del Adaptador de Aplicación, ya descrito anteriormente con más detalles, es ejecutar tareas que permitan la conexión de una aplicación al sistema de mensajería. En [2] este adaptador es clasificado como un tipo de Message Channel, llamado Channel Adapter. En este artículo lo llamamos Adaptador de Aplicación y proponemos la creación de un grupo específico para él debido a su gran importancia en la integración de aplicaciones, además de ser un contenedor de Adapter Blocks. A continuación presentamos tres Adapter Blocks de este grupo: Polling Consumer, Messaging Mapper y Code Block.

Polling Consumer: Lo que permite este Adapter Block es consumir mensajes de un canal tan sólo cuando la aplicación que tiene que procesarlos esté lista para hacerlo. Los mensajes se pueden consumir tanto de forma síncrona como no síncrona y, por supuesto, es posible consultar si existe algún mensaje disponible.

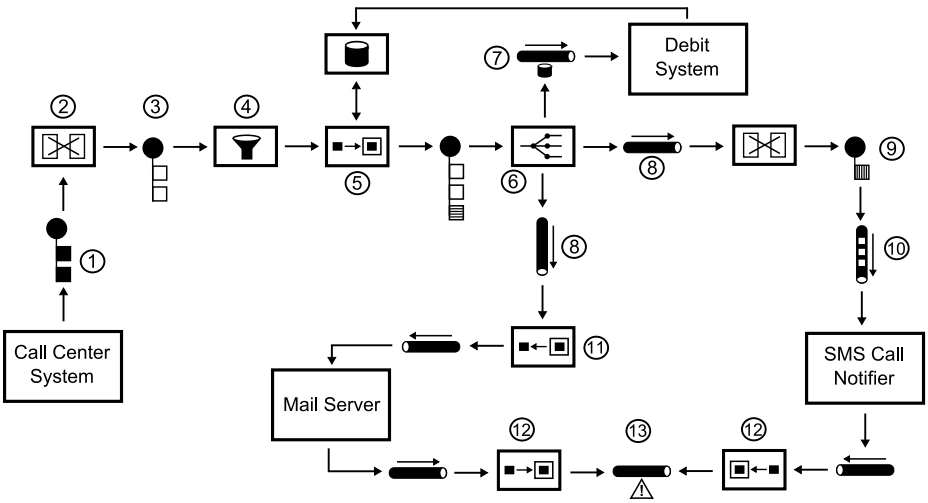

Figura 6. Ejemplo de una solución de integración

Messaging Mapper: Messaging Mapper permite transformar los objetos de una aplicación en mensajes del sistema de mensajería, con una gran independencia entre ellos. Este Adapter Block debe contener todas las reglas para hacer tal mapeo, así que ni los objetos ni el sistema de mensajería deben conocer el Messaging Mapper.

Code Block: Este tipo de Adapter Block es, quizás, el más sencillo de todos. Lo proponemos aquí como un bloque que pueda contener código escrito en un lenguaje/tecnología específica, y que, junto con los demás Adapter Blocks, permitirá la conexión de una aplicación al sistema de mensajería.

4. Ejemplo

La figura 6 presenta una solución de integración de cuatro aplicaciones, que al principio no fueran diseñadas teniendo en cuenta la integración (y que está inspirado en un sistema real que se usa en UNIJUI). Son aplicaciones muy distintas y desarrolladas con diferentes tecnologías. El objetivo de esta solución es hacer que todas las llamadas telefónicas registradas por el "Call Center System" (CCS) en su base de datos y que tengan algún coste para la empresa, sean también, registradas en su "Debit System" (DS). Además de registrar dichas llamadas en el DS, algunas informaciones de la llamada (por ejemplo: coste, hora de la llamada, ciudad y numero de destino) son enviadas por SMS y/o correo electrónico al usuario que la hizo. En esta empresa los empleados que tienen una clave pueden acceder cualquier terminal telefónico, en cualquiera de las ciudades donde está la empresa y hacer una llamada. Todas las llamadas son registradas y al fin del mes el empleado tiene que decir cuáles fueron hechas por razones de trabajo

y cuáles fueron llamadas privadas, ya que las privadas tendrán que ser pagas por el empleado.

La dirección del flujo de mensajes de la solución está indicada con las flechas entre los Building Blocks de la figura 6. Así que todo empieza con un mensaje en formato privado (1) desde el CCS que es traducido por un Message Translator (2) a un mensaje en el formato canónico (Canonical Data Model) (3). Después de esta traducción hay un Message Filter (4) que rechazará todos los mensajes que no sean de pago y enviará los demás a un Content Enricher (5) que debe añadir al mensaje datos, como por ejemplo: nombre, teléfono, correo, etc. Después el mensaje llegará en un Recipient List (6) que enviará una copia del mensaje a los posibles canales de salida. El canal hacia el DS es del tipo Guaranteed Delivery (7) y siempre recibirá una copia y los canales hacia el "Mail Server" (MS) y el "SMS Call Notifier" (SMS-CN) son de tipo Point-to-Point (8). Estos canales solamente recibirán un mensaje si tiene una dirección de correo y/o un numero de móvil. El mensaje hacia el SMS-CN será ahora traducido a un formato privado de mensajes SMS (9) y enviado a un Datatype Channel (10), desde el cual la aplicación SMS-CN lo leerá. El otro flujo hacia el MS quitará del mensaje con un Content Filter (11) informaciones que no sean de interés para enviar por correo. En caso de que no consigan enviar el mensaje, tanto el MS como el SMS-CN le añadirán una descripción del problema (12) y, a su vez, lo enviarán a un Invalid Message Channel (13).

Considerando las topologías de integración de aplicaciones expuestas en [2], clasificamos este ejemplo de la figura 6 como Hub-and-Spoke. Aquí el "hub" sería el Recipient List (6) y los "spokes" cada una de sus salidas (7 y 8) hacia las aplicaciones DS, MS y SMS-CN.

5. Conclusiones

La integración es una tarea cada vez más frecuente en las empresas, y que además de tener un alto coste suele consumir muchos recursos. Para tales empresas la necesidad de integración es, en gran parte, consecuencia de la evolución natural de su negocio. Enterprise Integration Information (EII) es una metáfora declarativa en la que el objetivo es ver todo el sistema como un gran modelo de datos; por el contrario Enterprise Application Integration (EAI) es una metáfora operativa en la que el objetivo es ver todo el sistema como un gran flujo de información. Una solución de integración usando sistemas de mensajería permite tener, entre otras ventajas, bajo acoplamiento y comunicación asíncrona, las cuales son muy importantes a la hora de escoger un estilo de integración y diseñar dicha solución. Los patrones presentados en [2] para el estilo de integración por medio de sistemas de mensajería, aunque muy importantes, son demasiados abstractos para permitir su uso más directo a la hora de diseñar la solución de integración.

Así que propusimos, en este artículo, una división de la solución de integración en tres niveles y cuatro vistas para ayudarnos a diseñar y entender mejor dicha solución. Con esto también empezamos un proceso de investigación, clasificación

y formalización de todos los patrones para transformarlos en Building Blocks, y así permitir diseñar la solución de integración arrastrando y conectando tales bloques. Creemos que de momento debemos seguir trabajando hasta lograr este reto.

Referencias

1. Weiss, J.: Aligning relationships: Optimizing the value of strategic outsourcing. Global services report, IBM (2005)
2. Hohpe, G., Woolf, B.: Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions. The Addison Wesley Signature Series. Addison-Wesley, Boston (10 October 2003)