

# Integración en DENEb de componentes para la conectividad dinámica de los procesos Web. Aplicación a escenarios de gestión de emergencias basados en Sensor Web \*

J. Fabra, P. Álvarez, J.A. Bañares y J. Ezpeleta

Instituto de Investigación en Ingeniería de Aragón (I3A)  
Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza  
María de Luna 3, E-50018 Zaragoza (España)  
{jfabra,alvaper,banares,ezpeleta}@unizar.es

**Resumen** Las futuras plataformas de ejecución de procesos de negocio deberán proporcionar un alto grado de flexibilidad en la integración de sus procesos. Éstos deberían ser capaces de colaborar en base a protocolos de interacción consensuados y no consensuados e incluso de configurar en tiempo de ejecución el estilo de interacción, la tecnología de comunicación y el formato de codificación de los mensajes intercambiados como parte de sus interacciones. En este artículo se presenta una extensión de la plataforma DENEb para el desarrollo y ejecución de procesos de negocio. Esta extensión proporciona el soporte necesario para satisfacer los requisitos a nivel de integración previamente descritos, convirtiendo a DENEb en una plataforma adaptable, flexible y muy adecuada para entornos de procesos complejos y dinámicos. La aplicabilidad de las soluciones propuestas ha sido probada en un escenario de prevención de riesgos de incendio basado en las directrices de la iniciativa Sensor Web.

**Palabras clave:** Integración de procesos de negocio, Protocolos de interacción, Cooperación dinámica entre procesos, Redes de Petri.

## 1. Introducción

Actualmente, las organizaciones especifican *a priori* los protocolos de interacción que regulan la integración de sus procesos de negocio. Esta estrategia garantiza el éxito de la colaboración global si los procesos de cada organización individual han sido posteriormente construidos en base a las restricciones del protocolo. Por desgracia, en otros escenarios las organizaciones implementan sus procesos de negocio de manera individual. En este caso, cada proceso establece de manera local sus propias restricciones de interacción. Por tanto, para garantizar la integración de un conjunto de procesos que requieren colaborar es común

---

\* Este trabajo se ha financiado parcialmente gracias al proyecto de investigación del Ministerio de Educación y Ciencia TIN2006-13301.

tener que recurrir a la utilización de técnicas de mediación que garanticen la compatibilidad de sus restricciones.

Evidentemente, ambas estrategias tienen un claro carácter estático desde el punto de vista de las interacciones. Los procesos de negocio tienen programado como parte de su implementación, entre otras restricciones de interacción, qué mensajes intercambian y en qué orden envían o reciben los mensajes. Por tanto, están implementados para colaborar en base a un protocolo concreto. En [1,2] se esbozan los requisitos a nivel de integración de la siguiente generación de procesos de negocio (y aplicaciones Web, en general). Estos requisitos exigen formas mucho más flexibles de integración. Por ejemplo, los procesos deberían ser capaces de decidir en tiempo de ejecución con quién colaboran y cómo colaboran, adaptándose dinámicamente a las nuevas políticas y estrategias de negocio y respondiendo de manera eficiente a los cambios y eventos inesperados [3,4,5]. Este requisito choca frontalmente con las soluciones actuales, dado que los procesos tendrán que colaborar conforme a distintos protocolos durante su ejecución y, probablemente, conforme a protocolos que a priori desconozcan. Un ejemplo de dominio de problema donde esta flexibilidad es necesaria es la gestión de emergencias.

Las tecnologías semánticas y determinadas aproximaciones de diseño de procesos, por ejemplo, el enfoque conversacional [6,7], constituyen unos prometedores primeros pasos hacia la consecución de la flexibilidad requerida a nivel de integración. No obstante, nuevos modelos de procesos de negocio y, sobre todo, entornos operativos que faciliten el desarrollo y la ejecución de esta nueva generación de procesos son aún necesarios. Un ejemplo de esta siguiente generación de entornos es DENEb (*a platform for the Development and Execution of wEB processes*) [8]. DENEb está basada en el paradigma de las *redes-en-redes* y, en versiones anteriores, los procesos colaboraban en base a protocolos de interacción previamente consensuados. En este trabajo se presenta un mecanismo para ir un paso más allá en el incremento de la flexibilidad, permitiendo a los procesos DENEb colaborar utilizando protocolos de interacción no consensuados que son conocidos en tiempo de ejecución. Para conseguirlo, los procesos tienen la capacidad de enviar, recibir y ejecutar dinámicamente los protocolos de interacción. Esto supone un mayor nivel de flexibilidad con respecto las actuales plataformas de ejecución de procesos.

Este artículo está organizado de la siguiente manera. La Sección 2 describe el enfoque conversacional de una manera intuitiva. La implementación de la plataforma DENEb, así como los nuevos componentes que la dotan de capacidad y conectividad dinámica, se presentan en las Secciones 3 y 4, respectivamente. Las nuevas ideas presentadas se aplican a un caso de estudio para la integración de servicios Sensor Web en procesos de negocio en base a protocolos no consensuados en la Sección 5. Finalmente, se presentan las conclusiones.

## 2. El enfoque conversacional

Internamente, los *procesos de negocio* están compuestos de un conjunto de tareas que deben ser ejecutadas de acuerdo a un conjunto de restricciones de orden. Este flujo de tareas recibe el nombre de *lógica de negocio* del proceso. Desde el punto de vista de la implementación, la tecnología de *workflows* puede ser reutilizada para describir y ejecutar la lógica de negocio de un proceso.

Normalmente, una tarea concreta corresponde con la ejecución de una acción interna (por ejemplo, el acceso a un repositorio de datos de la organización propietaria del proceso de negocio o un cómputo utilizando datos locales al propio proceso) o la interacción con una entidad software externa (otro proceso de negocio, un servicio Web, un agente software, etc.). Estas interacciones externas pueden ser tan simples como la invocación de una operación de un servicio o tan complejas como la ejecución de un protocolo de negociación para la subasta de un producto. En cualquier caso, estas interacciones están descritas por medio de *protocolos de interacción* o *políticas de conversación* [6]. Un protocolo describe fundamentalmente qué mensajes son intercambiados en el marco de una interacción, cuál es la estructura y el contenido de los mensajes intercambiados, en qué orden se envían los mensajes, etc. Además, un protocolo está organizado como una colección de *roles*, donde cada rol describe la parte del protocolo que debería ser ejecutada por cada proceso participante en la interacción (por ejemplo, en un sencillo protocolo de compra-venta habrá dos roles: el rol comprador y el rol vendedor). No obstante, en un protocolo pueden existir varias secuencias válidas de intercambio de mensajes, recibiendo cada una de ellas el nombre de *conversación*.

Desde el punto de vista de un proceso concreto, las acciones necesarias para ejecutar las distintas conversaciones que puede mantener con otros procesos durante su ejecución (por ejemplo, iniciar y finalizar una conversación, crear un mensaje, extraer información de un mensaje recibido, enviar o recibir un mensaje, tomar una decisión que condicione el flujo de la conversación, etc.) reciben el nombre de *lógica de interacción*. Tradicionalmente, en la implementación de los procesos de negocio están mezcladas la lógica de negocio y la lógica de interacción. Este es el caso de los procesos implementados en el estándar BPEL4WS. Evidentemente, estas soluciones son poco flexibles desde el punto de vista de la integración de procesos. Dado un conjunto de procesos que están colaborando, cualquier modificación en su protocolo de interacción implica detener la ejecución de los procesos involucrados, reprogramar la lógica de interacción de su implementación y, finalmente, volver a ponerlos en ejecución para que colaboren en base al nuevo protocolo.

El *enfoque conversacional* propone la separación explícita entre los aspectos de la lógica de negocio (los *workflows*) y los aspectos de interacción (los protocolos). El objetivo de esta propuesta de diseño es permitir la ejecución de distintos protocolos de interacción reutilizando la misma lógica de negocio. De esta manera, un proceso será capaz en tiempo de ejecución de seleccionar un protocolo de un conjunto predeterminado y ejecutar su correspondiente rol sin necesidad de detener su ejecución para modificar su implementación. No obstante, el enfoque

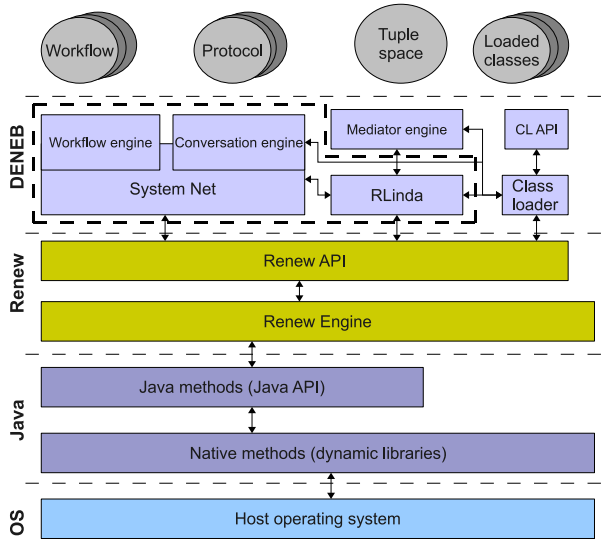
conversacional permite alcanzar una mayor flexibilidad en la integración de los procesos. Por ejemplo, los procesos podrían en tiempo de ejecución determinar (e incluso negociar) el protocolo en base al cual van a colaborar y, una vez consensuado, ejecutar el protocolo para completar la interacción. Para conseguir esta flexibilidad es necesario previamente que los procesos compartan una manera común de describir los protocolos y que los procesos dispongan de mecanismos a nivel de infraestructura para el envío, recepción y ejecución de los protocolos (roles). En el caso más simple, un proceso dedicado a la venta de productos por Internet podría establecer unilateralmente el protocolo de compra-venta a ejecutar para colaborar con él. Este proceso enviaría el rol comprador a cualquier proceso que muestre interés en la compra de alguno de sus productos. Una vez recibido el rol, cada proceso comprador lo pondría en ejecución para llevar a cabo el proceso de compra.

### 3. Arquitectura e implementación de DENEb

Las redes de Petri [9] han sido ampliamente utilizadas para la especificación, análisis e implementación tanto de *workflows* como de protocolos de comunicación [10,11]. Este formalismo resulta de gran interés, dado que aporta una semántica formal precisa y clara, una notación gráfica muy intuitiva y numerosas técnicas y herramientas para el análisis, simulación y ejecución.

La utilización de las redes de Petri para la especificación de la lógica de negocio y la lógica de interacción de los procesos Web facilitaría la integración de ambos aspectos (recuérdese que uno de los principales problemas en torno a los servicios Web es la dificultad para integrar las especificaciones resultantes de los estándares de composición y coreografía). No obstante, las redes de Petri ordinarias (también llamadas redes lugar/transición o redes de Petri generalizadas) se caracterizan por una estructura estática. Esta limitación choca frontalmente con la flexibilidad (dinamismo) requerida en el mundo de los procesos Web. El paradigma de las *redes-en-redes* [12], pertenecientes a la familia de los formalismos de las redes de Petri objeto, se presenta como una alternativa válida para conseguir la flexibilidad requerida. Las *redes-en-redes* se componen de una parte estática (llamada *red sistema*) y de una parte dinámica, compuesta de instancias de *redes objeto* que se mueven dentro de la red sistema. La red sistema y las redes objeto pueden interactuar utilizando *canales* en los que pueden utilizarse parámetros para intercambiar información.

En [8] se describe DENEb (*a platform for the Development and Execution of wEB processes*), una plataforma para el desarrollo y la ejecución de procesos Web basada en el paradigma de *redes-en-redes*. La idea intuitiva de DENEb es sencilla. De acuerdo al enfoque conversacional, la lógica de negocio de un proceso (*workflow*) se implementa como una red objeto y cada uno de los roles que ejecuta para conversar con otros procesos también se implementan como redes objeto. Estas redes objeto colaboran e intercambian información entre ellas a través de canales. Por otro lado, el entorno operativo que ejecuta los procesos (los *workflows* y sus conversaciones) se implementa como la red sistema. Cabe



**Figura 1.** Arquitectura de alto nivel de DENEb

destacar que también podrían utilizarse los estándares actuales de servicios Web, tanto en la descripción de los workflows (BPEL4WS o BPMN, por ejemplo), como en la de los roles (WSCDL, WSCI), ya que actualmente DENEb soporta la traducción de varios de estos estándares a redes objeto.

La Figura 1 muestra la arquitectura de alto nivel de DENEb. La plataforma se ejecuta sobre la herramienta Renew (<http://www.renew.de/>) y la máquina virtual de Java. Desde un punto de vista arquitectural, DENEb consta de tres componentes clave: el *componente de composición*, el *componente de conversación* y el *broker de mensajes*. El componente de composición gestiona el ciclo de vida de los procesos e ejecuta su lógica de negocio por medio de un intérprete de *workflows*. Las conversaciones que tienen lugar entre los procesos son ejecutadas por el componente de conversación. Por tanto, este componente es responsable de la gestión de los roles en ejecución. Por último, el broker de mensajes separa la lógica del intercambio de los mensajes (expresado en los roles) de los protocolos de transporte (HTTP, SMTP, RPC, etc.), formatos de codificación y del estilo de interacción (REST, SOAP, etc.) utilizados para el envío de los mensajes. Este broker consta de dos componentes: un repositorio asíncrono de mensajes, basado en RLinda o DRLinda [13], y un conjunto de mediadores que son ejecutados por su correspondiente motor.

Los mediadores son los componentes (implementados como redes o como clases Java) responsables de conectar a DENEb con el mundo exterior. Gracias a ellos, los procesos DENEb pueden interactuar con procesos (o aplicaciones, en general) ejecutados en otras plataformas y/o entornos heterogéneos. Internamente, las conversaciones envían y reciben mensajes expresados en un formato

interno dependiente de la plataforma, más concretamente como tuplas FIPA SC00026H. Estos mensajes son temporalmente almacenados en el repositorio de mensajes. Los mediadores leen del repositorio los mensajes pendientes de enviar a otros procesos y, utilizando protocolos de comunicación, formatos de codificación y estilos de interacción concretos, los envían a su correspondiente destinatario. Del mismo modo, mensajes enviados desde el exterior a los procesos ejecutados en DENEb, son recibidos por determinados mediadores, transformados a tuplas FIPA y escritos en el repositorio para que la conversación destinataria los recupere. Funcionalmente, estos mediadores podrían realizar tareas más complejas, por ejemplo, la mediación semántica de los datos intercambiados en los mensajes o el filtrado de información.

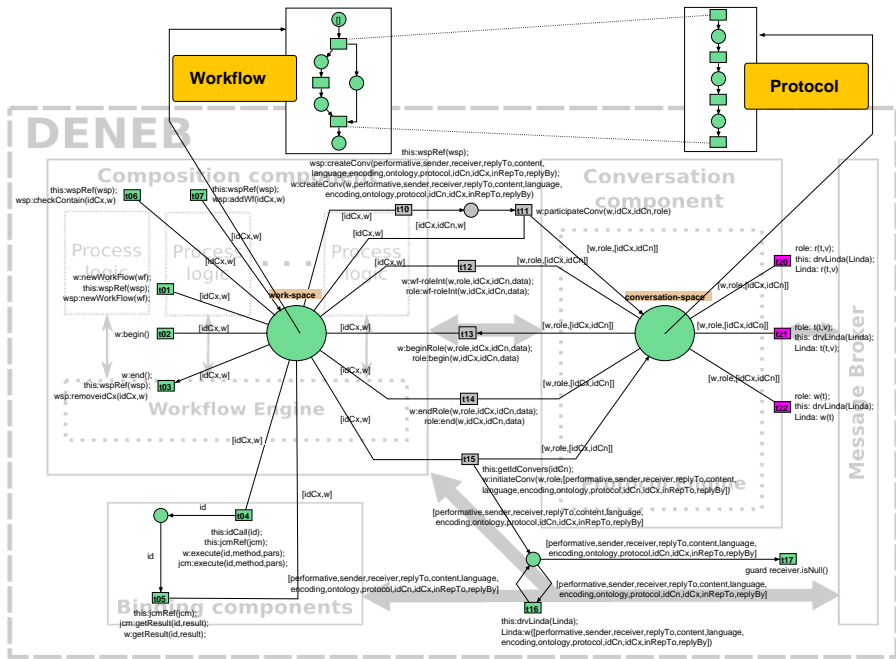


Figura 2. Implementación de DENEb: red sistema.

A partir del diseño arquitectural previo, la Figura 2 muestra una descripción detallada de la implementación de DENEb. Desde el punto de vista del paradigma de *redes-en-redes* corresponde con la red sistema (en el fondo se han representado los componentes arquitecturales previamente descritos para aclarar la relación entre esos componentes y los elementos de la red implementada). Una descripción detallada sobre esta implementación y el ciclo de vida de ejecución de las redes objeto que representan los *workflows* y los roles fue presentada el año anterior en esta conferencia [14].

## 4. Diseño e integración del componente de conectividad dinámica

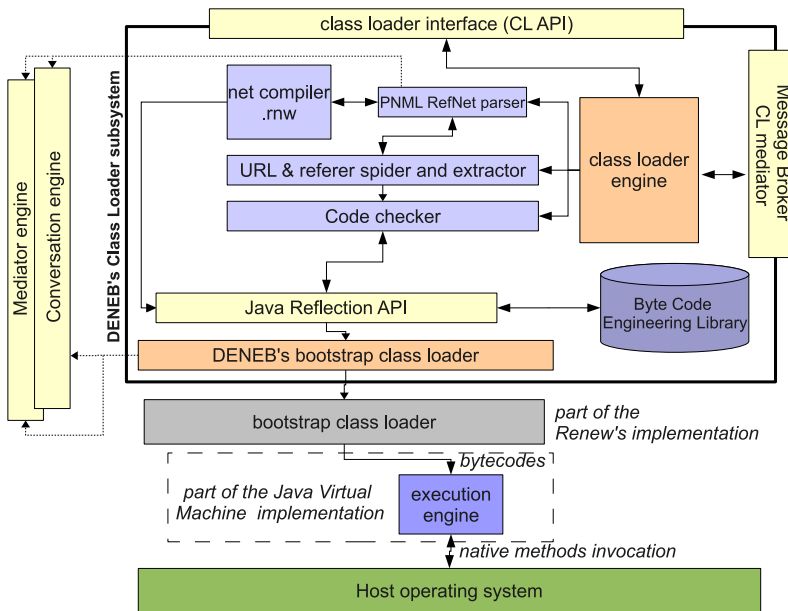
Se ha integrado en DENEb un nuevo componente capaz de recibir en tiempo de ejecución nuevos protocolos y mediadores. Utilizando la funcionalidad de este componente, un proceso no tiene que conocer de antemano todos los protocolos que regulan sus conversaciones, sino que podrá intercambiar protocolos con otros procesos en tiempo de ejecución. Esto permite, por ejemplo, que un proceso pueda decidir durante su ejecución con quién colabora o incluso negociar las condiciones de regulan cómo llevar a cabo la colaboración en función de su historial previo de ejecución. Por otro lado, la posibilidad de incorporar nuevos mediadores permite enriquecer las tecnologías de comunicación, los formatos y los mecanismos de mediación de datos (incluidos los semánticos) y los estilos de interacción soportados por la plataforma. Estas capacidades que potencian la conectividad de los procesos Web a nivel de protocolo y a nivel de mensajes intercambiados, respectivamente, han sido declaradas como requisitos clave en las futuras plataformas de desarrollo y ejecución de servicios y procesos Web [1,2].

La parte superior derecha de la Figura 1 muestra el *cargador de componentes* (*component loader*, CL), responsable de la funcionalidad previamente descrita. La interacción desde el exterior (por ejemplo, desde una herramienta de administración) con el componente cargador se realiza a través de su interfaz (CL API). Por otro lado, la interacción entre el componente y el núcleo de la plataforma DENEb (por ejemplo, para registrar los nuevos protocolos o mediadores) tiene lugar a través de la API de la herramienta Renew.

La Figura 3 presenta el diseño detallado del cargador de componentes. Desde el punto de vista de los protocolos, un proceso externo puede enviar directamente un protocolo descrito en el lenguaje PNML<sup>1</sup> o puede enviar la URL donde está accesible el nuevo protocolo. La utilización del estándar PNML para el intercambio de roles entre procesos resuelve los aspectos de interoperabilidad relacionados con la descripción de los protocolos. En cualquiera de los dos casos, el proceso externo envía un *mensaje de control* conteniendo el protocolo o su URL que es almacenado en el repositorio de mensajes de DENEb. El cargador de componentes está monitorizando la escritura de este tipo de mensajes de control. Cuando detecta un nuevo mensaje, lo recupera del repositorio, extrae el protocolo (o lo descarga de la URL correspondiente) y, finalmente, ejecuta una serie de acciones internas para registrar el correspondiente protocolo en tiempo de ejecución. A continuación se describen brevemente estas acciones. En primer lugar, el *parser* de PNML comprueba la corrección de la descripción suministrada y, si es posible, genera a partir de ella una red ejecutable por Renew equivalente. Esta red está implementada como una clase Java. Por tanto, utilizando la API de reflexión de Java (*Java API Reflection*) y la biblioteca *Byte Code Engineering Library* (BCEL), los *bytecodes* de Java son optimizados y posteriormente cargados en el núcleo de DENEb utilizando la API de Renew. Nótese que la

---

<sup>1</sup> *Petri Net Markup Language*, PNML ISO/IEC 15909, un lenguaje de descripción de redes de Petri basado en XML.



**Figura 3.** Arquitectura del componente de carga dinámica en DENEb.

implementación descrita es un prototipo que se asume se ejecutará en un entorno de confianza y, por tanto, no se darán comportamientos maliciosos. Un desarrollo futuro contará con los mecanismos de seguridad adecuados.

Por otro lado, la carga de un mediador es similar a la de un rol de un protocolo. Un mediador puede implementarse directamente como una clase Java o puede implementarse como una red. En este último caso, la red será traducida a la clase Java que la implementa utilizando el mismo procedimiento que para los protocolos. En cualquier caso, la clase resultante que implementa el mediador será traducida a sus *bytcodes* optimizados y registrada a través de la API de Renew en el motor de mediadores. También resaltar que se ha desarrollado un verificador de código básico para un análisis preliminar del comportamiento de los mediadores implementados directamente en código Java.

## 5. Caso de estudio: integración de servicios Sensor Web en procesos de negocio en base a protocolos no consensuados

El Centro de Investigación e Información Ambiental de Galicia (Consellería de Medio Ambiente y Desenvolvemento Sostible de la Xunta de Galicia) tiene una red de estaciones meteorológicas <sup>2</sup> que monitorizan permanentemente una

<sup>2</sup> <http://www.meteogalicia.es/galego/observacion/estacions/aRede/todaarede.asp>

colección de parámetros atmosféricos y ambientales en distintos puntos del territorio de la comunidad autónoma. La información obtenida por estas estaciones puede ser de interés para predecir ciertos riesgos naturales, por ejemplo, el riesgo de incendios. Si la vegetación de una zona es densa y una estación próxima determina que la temperatura es elevada, la humedad es baja y el viento fuerte, el riesgo de incendio en esa zona y la velocidad de propagación en caso de suceder serían potencialmente altos (este ejemplo es una simplificación de los modelos de análisis que son utilizados para detectar los niveles de riesgo).

El objetivo es construir una aplicación para el centro de gestión de emergencias que notifique a un operador las zonas con un riesgo de incendio potencialmente alto. Desde el punto de vista del operador, para simplificar su labor, se desea que las alarmas sean representadas sobre un mapa digitalizado de la comunidad autónoma. También es deseable que el operador pueda interactuar con el mapa y acceder a la información disponible sobre el posible riesgo notificado. Desde el punto de vista de la implementación de la aplicación, se desea que ésta sea accesible vía Web y que su lógica de negocio interna sea implementada en base a procesos de negocio. Estos procesos para realizar su labor necesitarán poder cooperar de forma remota con las distintas estaciones, por ejemplo, para configurar sus planes de monitorización y acceder a las observaciones del territorio obtenidas. Toda la infraestructura intermedia entre los procesos de negocio de la aplicación y las estaciones físicas deberá estar basada en el paradigma SOA, los estándares y tecnologías de servicios Web y los estándares y recomendaciones de la iniciativa *Sensor Web* de OGC<sup>3</sup>.

### 5.1. Diseño arquitectural de la solución

La Figura 4 muestra la arquitectura basada en Sensor Web que ha sido diseñada en el marco de este trabajo para el desarrollo de aplicaciones dedicadas a la gestión de emergencias y la prevención de riesgos naturales. Esta arquitectura consta de tres niveles: nivel de servicios, nivel de procesos de negocio y nivel de aplicación.

El nivel de servicios está formado por el núcleo de servicios Sensor Web y un conjunto de servicios externos accesibles a través de Internet. Los servicios Sensor Web permiten conocer qué sensores están disponibles (más concretamente, el servicio *Sensor Type Catalog*, STC), configurar planes de monitorización en las estaciones remotas (*Sensor Planning Services*, SPS) y acceder a las observaciones de las estaciones (*Sensor Collection Service*, SCS). Un servicio de notificación es el responsable de la coordinación entre estos servicios. Por otro lado, servicios externos de tipo GIS (*Geographic Information Systems*) también integran este nivel inferior. Entre estos servicios caben destacar servidores de mapas (*Google Maps*, por ejemplo) y distintos servidores especializados en proveer información medioambiental en diferentes formatos (*Web Feature Server*, WFS, y *Web Coverage Server*, WCS). Las interfaces de todos los servicios de este nivel están basadas en las especificaciones estándar propuestas por el consorcio

---

<sup>3</sup> Open Geospatial Consortium, <http://www.opengeospatial.org/>

OGC y son accesibles a través de los protocolos de transporte (HTTP, SOAP, etc.) y formatos de codificación de datos (XML) típicos de las soluciones Web.

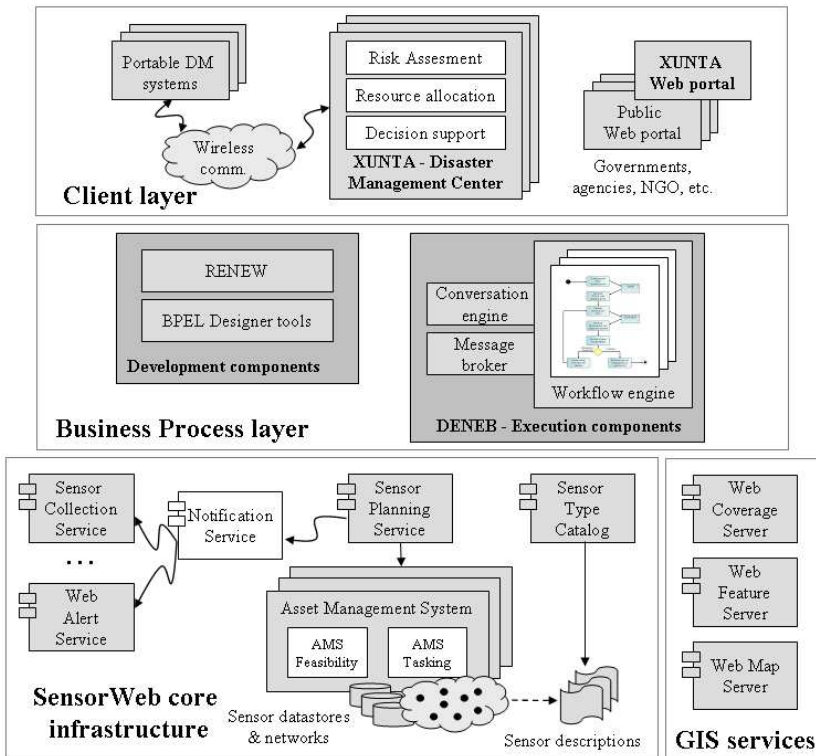


Figura 4. Arquitectura de tres niveles basada en Sensor Web.

En el nivel intermedio de la arquitectura están los procesos de negocio que implementan parte de la lógica de las aplicaciones finales. En nuestro caso concreto, uno de los procesos de negocio implementados es responsable de acceder a las observaciones de las estaciones, analizar si existen posible alarmas en base a un modelo de riesgos y, finalmente, informar al operador a través de la aplicación de notificación en caso de que fuese necesario. Este proceso coopera con los servicios del nivel inferior (reutilizables por cualquier proceso de negocio que requiera de su funcionalidad) en base a los protocolos de interacción publicados como parte de su interfaz. Estos protocolos son conocidos en tiempo de diseño, excepto los protocolos que facilitan a los procesos la interacción con las estaciones para la configuración de sus parámetros de observación. Además, estos protocolos pueden ser distintos para cada estación concreta. El servicio SPS es responsable de facilitar a los procesos el protocolo concreto de la estación con la que desean colaborar.

El nivel de procesos de negocio podría haber sido implementado utilizando el estándar BPEL y alguna de las herramientas e intérpretes basados en este estándar. No obstante, la naturaleza estática de BPEL y la necesidad de integrar en tiempo de ejecución protocolos desconocidos en tiempo de diseño (los protocolos para interactuar con las estaciones) hizo aconsejable la búsqueda de alternativas. En este caso concreto, DENEb ha sido utilizado como entorno de ejecución de procesos y las redes de Petri como lenguaje de implementación de estos procesos.

Por último, el nivel de aplicación está formado por las aplicaciones finales integradas en el centro de emergencias (por ejemplo, sistemas de información complejos, aplicaciones Web ejecutadas en navegadores, aplicaciones móviles, etc.). En general, estas aplicaciones integran en su lógica interna la funcionalidad ofrecida por los procesos del nivel inferior. Esta integración también se realiza vía Web.

## 5.2. Implementación del proceso de negocio

Dentro del marco arquitectural descrito anteriormente, en esta sección se presenta un proceso DENEb que explota la conectividad dinámica de la plataforma para la integración de servicios en base a protocolos conocidos en tiempo de ejecución. Más concretamente, se describe la parte correspondiente al acceso a la información de los sensores de una estación dentro de un proceso encargado de la monitorización de las observaciones. Este proceso se encuentra ejecutándose en el nivel de negocio, y persigue la integración de los servicios del nivel inferior.

Tal y como se describió previamente, los protocolos que facilitan a los procesos la interacción con las estaciones para la configuración de los parámetros de las mismas no son conocidos por los procesos, sino únicamente por el SPS. Estos procesos pueden ser heterogéneos, por lo que nuestro proceso de monitorización desconoce *a priori* los detalles del protocolo requerido para la interacción. La Figura 5 muestra cómo el acceso a una estación se realizará en dos pasos. Primero, el proceso de monitorización solicita el envío por parte del SPS del protocolo de interacción de la estación  $i$ . El SPS envía dicho protocolo, que es procesado por el cargador de componentes según la forma de envío y el tipo de implementación (Figura 3), tras lo cual el proceso tiene acceso a la red correspondiente a dicho protocolo. Así pues, sólo resta poner en ejecución el protocolo de interacción con los datos de configuración del proceso de monitorización. Estos datos van descritos en el estándar propuesto por el SWE, y la interacción con la estación se realiza mediante invocaciones REST. Sin embargo, tanto la especificación de estas llamadas como el encapsulamiento de los datos de configuración en dichas llamadas son propios de cada sensor y, por tanto, desconocidos *a priori* para el proceso de monitorización.

Desde el punto de vista de la implementación, tanto el *workflow* del proceso de monitorización como el protocolo de interacción (una vez procesado por el CL) se implementan utilizando redes de Petri, lo que produce un modelo ejecutable directamente mediante la utilización de la plataforma DENEb. La Figura 6 muestra los fragmentos de las redes correspondientes a la carga y ejecución del

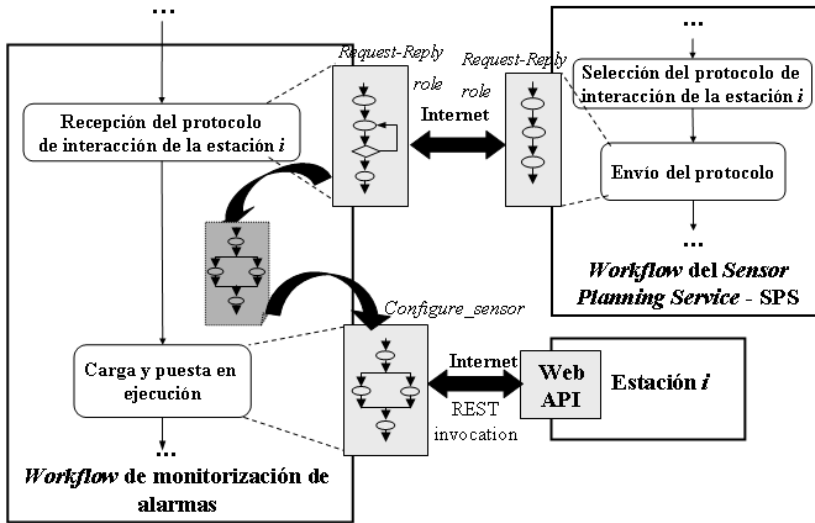
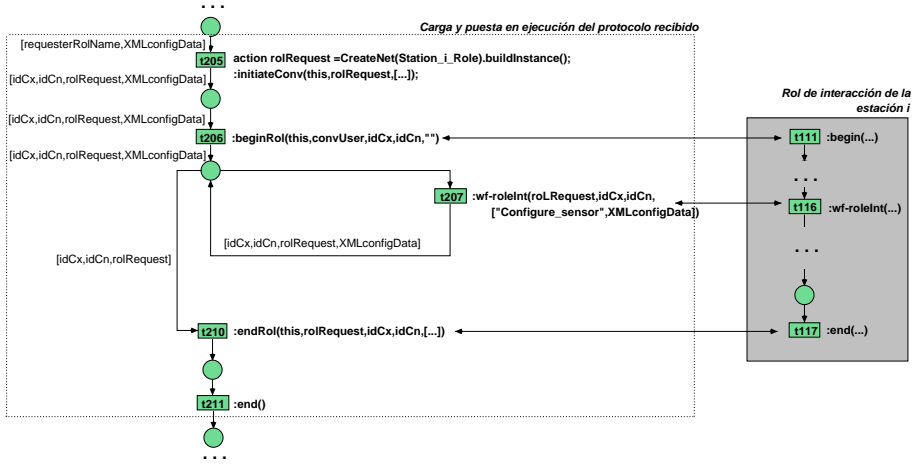


Figura 5. Interacciones en la adquisición del protocolo de acceso a la estación  $i$ .

protocolo recibido, cuyo esquema se muestra en la parte derecha de la figura. La parte izquierda de la figura representa la implementación del proceso de puesta en marcha del protocolo recibido en el *workflow* del proceso de monitorización. Obsérvese que, tras el proceso de adquisición dinámica llevado a cabo con el SPS, el nombre del rol recibido se almacena en la variable *Station.i.Role*, que se utiliza para instanciar el protocolo recibido mediante el disparo de la transición  $t_{205}$  (que corresponde con el disparo sincronizado con la red sistema mostrada en la Figura 2). El disparo sincronizado de las transiciones  $t_{206}$  y  $t_{111}$  a través de la red sistema pone en ejecución la instancia del protocolo recibido para el acceso a la estación  $i$ . Una vez en marcha, el *workflow* y el protocolo intercambian información, como los datos de configuración y la operación a realizar en base al estándar SWE, mediante el disparo de la transición  $t_{207}$ , que implementa el canal `:wf-roleInt(...)`. En ciertas situaciones, este intercambio de información podría llevarse a cabo más de una vez en función de la existencia de subestaciones asociadas a la estación principal, hasta que la transición  $t_{210}$  se sensibilice. Una vez enviada la información de configuración, el proceso de monitorización espera a que el protocolo de interacción termine mediante el disparo sincronizado de las transiciones  $t_{210}$  y  $t_{117}$ , respectivamente. De esta forma, el proceso de monitorización ha conseguido acceder a la estación utilizando su protocolo de interacción específico y desconocido a priori.

La utilización de la plataforma DENEb en el diseño e implementación del SWE abre nuevos retos, como la posibilidad de añadir al sistema nuevas redes de sensores heterogéneas de forma *plug-and-play*, la utilización de protocolos de interacción diferentes con la misma estación o, incluso, el mismo protocolo con varias estaciones diferentes, en el caso de que lo permitiesen.



**Figura 6.** Workflow del proceso de monitorización correspondiente a la carga y puesta en ejecución del protocolo de interacción con la estación *i*.

## 6. Conclusiones

En este trabajo se ha mostrado un enfoque que mejora los aspectos de interoperabilidad entre procesos y satisface los requisitos de integración de los entornos interorganizacionales dinámicos y evolutivos. Esta propuesta se basa en la descripción adecuada de los servicios y de los protocolos de interacción, separando la lógica de negocio de la lógica de interacción.

La plataforma DENEb aporta a los desarrolladores y usuarios finales una infraestructura ejecutable y una serie de abstracciones para implementar procesos Web y los aspectos relacionados de composición y orquestación dinámicos. DENEb implementa los *workflows* y los protocolos desde la perspectiva del paradigma de las Redes-en-Redes y facilita un mayor nivel de flexibilidad para la integración de servicios.

La flexibilidad de la plataforma DENEb se ha mejorado y extendido mediante el desarrollo de una componente de carga dinámica que no sólo facilita el trabajo con protocolos preestablecidos, sino que también permite que los procesos interacciones utilizando protocolos acordados en tiempo de ejecución. Obviamente, esta flexibilidad abre nuevos retos, por ejemplo, el estudio de la compatibilidad entre la lógica de negocio de un proceso y la lógica de interacción representada por los roles que recibe en tiempo de ejecución. En el caso de DENEb, esta cuestión se está estudiando utilizando la capacidad de análisis de las redes de Petri, de manera que los problemas de compatibilidad se plantean y verifican en términos del modelo correspondiente a la ejecución sincronizada del *workflow* y el rol dados.

Esta última característica abre nuevas posibilidades a DENEb, fusionando los aspectos relacionados con la integración de servicios que aporta el enfoque

conversacional y la habilidad para llevar a cabo la composición dinámica *ad-hoc* de servicios, tanto para procesos complejos como para protocolos de interacción.

## Referencias

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. *Computer* **40** (2007) 38–45
2. Commission of the European Communities: 3S Green Paper on Software and Service Architectures, Infrastructures and Engineering. Technical report (2007)
3. Mecella, M., Pernici, B., Craca, P.: Compatibility of e-Services in a Cooperative Multi-platform Environment. In: Second International Workshop on Technologies for E-Services – TES 2001, London, UK, Springer-Verlag (2001) 44–57
4. Schmidt, R.: Web services based architectures to support dynamic inter-organizational business processes. In: International Conference on Web Services - Europe 2003 – ICWS-Europe’03. Volume 2853 of LNCS., Springer (2003) 123–136
5. Meng, J., Su, S.Y.W., Lam, H., Helal, A.: Achieving dynamic inter-organizational workflow management by integrating business processes, events and rules. In: 35th Annual Hawaii International Conference on System Sciences – HICSS’02. (2002)
6. Hanson, J.E., Nandi, P., Levine, D.W.: Conversation-enabled web services for agents and e-business. In: International Conference on Internet Computing. (2002) 791–796
7. Ardissono, L., Petrone, G., Segnan, M.: Enabling flexible interaction with web services. In: Extending Web Service Technologies. The use of Multi-Agent approaches. Springer Verlag (2004) 187–208
8. Fabra, J., Álvarez, P., Bañares, J.A., Ezpeleta, J.: Runtime Protocol Binding: Flexible Service Integration by Means of Flexible Service Interactions. In: 2008 IEEE International Conference on Services Computing – SCC’08. (2008) 291–298
9. Murata, T.: Petri nets: Properties, analysis and applications. In: Proceedings of IEEE. Volume 77. (1989) 541–580
10. van der Aalst, W., Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge, MA, USA (2004)
11. Girault, C., Valk, R.: Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2001)
12. Valk, R.: Petri Nets as Token Objects - An Introduction to Elementary Object Nets. In: 19th Int. Conf. on Application and Theory of Petri Nets. Volume 1420 of LNCS., Springer-Verlag (1998) 1–25
13. Fabra, J., Alvarez, P., Ezpeleta, J.: DRLinda: A Distributed Message Broker For Collaborative Interactions Among Business Processes. In: 8th International Conference on Electronic Commerce and Web Technologies – EC-Web’07. Volume 4655 of LNCS., Springer Verlag (2007) 212–221
14. Fabra, J., Alvarez, P., Bañares, J.A., Ezpeleta, J.: DENEb: Una plataforma para el desarrollo y ejecución de procesos Web dinámicos. In: III Jornadas Científico-Técnicas en Servicios Web y SOA – JSWeb’07. (2007) 11–18