Integrando un modelo de reutilización en la producción de software: entorno distribuido para el desarrollo basado en reutilización

Yania Crespo[1], Miguel Ángel Laguna[1], and Francisco Javier Pérez[1]

Departamento de Informática, Universidad de Valladolid, ,
{yania, mlaguna, jperez}@infor.uva.es, http://giro.infor.uva.es

Resumen Este trabajo presenta la definición, diseño y puesta en marcha de un entorno distribuido para el desarrollo de software basado en reutilización que permite la integración tanto de herramientas establecidas como de nuevas herramientas. Para ello se ha definido una arquitectura genérica de colaboración entre entidades distribuidas, capaces de trabajar según el modelo de reutilización. Como parte de la arquitectura se ha definido el tipo de entidades (nodos) que pueden participar en la colaboración, un protocolo de comunicación que permite a dichas entidades solicitar servicios e intercambiar objetos pertenecientes al modelo de componente reutilizable, los servicios básicos que pueden ofrecer los nodos y un diseño basado en frameworks que permite la reutilización de mecanismos y servicios a la vez que la independencia de la implementación y la tecnología particular del nodo. El diseño del entorno se ha realizado teniendo en cuenta su escalabilidad y adaptabilidad. Se han definido mecanismos de extensión que permiten la definición de nuevos servicios, la inclusión de nueva funcionalidad en los nodos, la posible ampliación del modelo de componente reutilizable, la coexistencia de repositorios referenciales con repositorios de contención, etc.

1. Introducción

En la actualidad nadie duda que cuando se desea obtener mejoras significativas, en cuanto a productividad y calidad, en el proceso de desarrollo de software, una de las piezas imprescindibles es la reutilización sistemática del software. La introducción de la reutilización sistemática de software en los procesos de desarrollo requiere, en primer lugar, lo que podemos llamar un modelo de reutilización en el que son imprescindibles la definición de un modelo de componente reutilizable y de un proceso de desarrollo basado en reutilización. Por otra parte, se requiere contar con definiciones y técnicas de certificación de calidad, cambios organizacionales, formación en técnicas específicas para el desarrollo basado en reutilización (Ingeniería de Dominios, desarrollo basado en *frameworks*, etc.). Por último, y no por ello menos importante, se necesita un soporte operativo. Para implantar un modelo de reutilización en entornos reales de desarrollo se debe contar con la infraestructura y las técnicas apropiadas, y éstas deben permitir la integración de herramientas (tanto herramientas establecidas y altamente utilizadas, como nuevas herramientas). Por otra parte, los entornos de producción de software responden cada vez más a las características del mundo moderno en el que se impone la distribución de los recursos técnicos y humanos, la coexistencia de tecnologías heterogéneas,

etc., lo que introduce una complejidad a manejar a la hora de dotar de un soporte operativo a las organizaciones que opten por implantar la reutilización sistemática en sus procesos de desarrollo.

Existen numerosos trabajos relacionados con la utilización de frameworks tanto para el diseño de herramientas de desarrollo de software, como para elaborar entornos de integración para herramientas de este tipo ya existentes. En la mayoría de los casos, y en ambos tipos de trabajos, se incluyen herramientas que soportan el desarrollo basado en reutilización.

En cuanto al primer caso, por ejemplo en [1], se presenta un framework de herramientas CASE, en el que se definen y se desarrollan servicios genéricos propios de este tipo de herramientas. Los servicios son desarrollados separando las partes dependientes e independientes de la plataforma de ejecución y son ofrecidos por un servidor en forma de objetos CORBA. Este trabajo, es un buen punto de partida para desarrollar frameworks de herramientas de desarrollo. Así mismo, en [2], ya se define un framework más específico de herramientas CASE para desarrollo basado en componentes. En los dos casos anteriores, se proponen frameworks para desarrollar herramientas completas.

Enmarcado en el segundo caso y siendo un trabajo más similar al nuestro, [3] define un framework para integrar herramientas de desarrollo encapsulando, herramientas ya existentes bajo un interfaz único y dotándolas de servicios comunes para acceder a un repositorio integrado en dicho framework. Aparece también la idea de la distribución, ofreciéndose el acceso remoto al entorno a través de CORBA.

Nuestro trabajo se encuentra en este segundo campo y aborda el caso particular de la integración de herramientas de desarrollo de software basado en reutilización, en el que el elemento característico es la necesidad de soportar el intercambio de elementos y servicios relacionados con un modelo de reutilización, ya definido. A la vez que se ofrece un soporte para la utiliación de cualquier modelo de componente reutilizable, ésto, se hace de forma independiente a las herramientas utilizadas. Se incluye además, una clasificación de los diferentes tipos de herramientas que se pueden encontrar en un entorno de desarrollo basado en reutilización, para poder definir los servicios genéricos que deben ofrecer y para poder ofrecer instanciaciones predefinidas del framework para cada tipo. Para ello se ha definido una arquitectura genérica de colaboración, los tipos de entidades, capaces de trabajar con el modelo de reutilización, que componen esta arquitectura, el protocolo de intercambio que utilizan y los servicios básicos que pueden ofrecer.

Para organizar la presentación, los contenidos del trabajo se estructuran como sigue. En la Sección 2 se presenta la arquitectura propuesta, la definición de los tipos de entidades (nodos) que participan y los servicios que éstas pueden ofrecer. En la Sección 3 se introduce la parte genérica del protocolo de comunicación entre las entidades que participan y se presenta su definición particular para un modelo de componente reutilizable específico. En la Sección 4 se explica la forma en que se que permite la reutilización de mecanismos y servicios a la vez que la independencia de la implementación y la tecnología particular del nodo y la solución a la coexistencia de repositorios referenciales con repositorios de contención. En la Sección 5 se describe la puesta en marcha de un ejemplo de entorno distribuido. Finalmente en la Sección 6 se concluye y se describen las líneas de acción inmediatas.

2. Arquitectura del entorno

La integración de herramientas y aplicaciones requiere una arquitectura lo suficientemente completa y simultáneamente abstracta como para poder incluir las herramientas actuales junto con la posibilidad de poder incorporar las que aparezcan en un futuro.

El entorno de desarrollo distribuído basado en reutilización, se compone de un conjunto de entidades genéricas, capaces de enviar y recibir mensajes mediante el protocolo particular del entorno, y de realizar y solicitar servicios sobre objetos pertenecientes al modelo de componente reutilizable empleado. A estas entidades se las ha denominado **Nodos**.

2.1. Tipos de Nodos

Los nodos se comunican entre ellos para intercambiar objetos pertenecientes al modelo de componente reutilizable y para solicitar y ejecutar servicios sobre dichos objetos. A partir de las funciones y las tareas que necesitan realizarse en un entorno de desarrollo *para* y *con* reutilización [4,5], y de las herramientas necesarias para ello, se han establecido inicialmente cuatro tipos de nodos que desempeñarán papeles bien diferenciados.

Tool Herramientas interactivas, utilizadas directamente por los usuarios del entorno. En la mayoría de los casos se tratará de heramientas CASE con funcionalidades específicas para reutilización y para trabajar con un modelo de componente reutilizable concreto.

Broker Nodos encaminadores, actúan como distribuidores de mensajes entre los demás nodos del entorno de desarrollo. Realizan diversas funciones de gestión del protocolo. Entre las tareas más importantes de las que llevan a cabo, está la de validar los mensajes intercambiados para garantizar que sólo contienen objetos y modelos[1] pertenecientes al modelo de componente reutilizable empleado.

Processor Nodos específicos para realizar un servicio concreto sobre objetos y/o modelos y devolver el resultado. Interaccionan exclusivamente con otros nodos.

Repository Nodos del entorno de desarrollo que ofrecen servicios de repositorio. Realizan tareas de almacenamiento y consulta para objetos pertenecientes al modelo de componente reutilizable empleado.

Los repositorios de componentes reutilizables pueden ser referenciales o de contención. Los repositorios de contención incluyen los componentes reutilizables en su propio esquema de almacenamiento, mientras que los referenciales actúan como catálogos de información, conteniendo las direcciones donde se deben solicitar los componentes deseados. La forma de enviar componentes reutilizables cambia en cada caso, aunque el cambio no es significativo (en un caso se envía sólo la información del modelo necesaria para catalogar el componente y su referencia y en el otro caso, en lugar de la referencia se envía el componente). Lo que sí resulta significativo es la forma de recuperación de los componentes.

[1] Se dice objetos y modelos para referirse a objetos que representan un asset o conjuntos de objetos –assets– conectados según las relaciones del modelo de reutilización al que pertenecen

Para poder integrar en este entorno el soporte necesario para dotar de transparencia a la recuperación en el caso de estar en presencia de repositorios referenciales, y para dar soporte al almacenamiento controlado de ficheros de una organización que sean referenciados por un repositorio, se ha definido un quinto tipo de nodo:

FileBroker Nodos que ofrecen servicios de repositorio de contención local y de recuperación transparente de componentes reutilizables, ubicados en el repositorio del proveedor o en el del propio nodo. Se describe más extensamente en el apartado 4.3.

2.2. Servicios de los Nodos

La comunicación entre los nodos, se realiza siempre a través de un nodo Broker. En principio, todos los nodos tienen la capacidad de realizar una petición a cualquier otro nodo, y de responder a estas peticiones siempre y cuando se realicen mediante el protocolo propio del entorno.

Los servicios que ofrece un nodo estarán dados, en general, por los servicios comunes a todos los tipos de nodos (están predefinidos), los servicios predefinidos para el tipo de nodo genérico al que corresponde, los servicios adicionales que se definan para ser ofrecidos por un tipo especializado de uno de los tipos genéricos, más los servicios particulares que añada la implementación específica de un nodo.

Para indicar a la hora de realizar una implementación de un nodo, qué servicios debe proporcionar y en qué consisten estos, se ha diseñado una plantilla de especificación de servicios. Siguiendo dicha plantilla, se han especificado los servicios comunes y los servicios predefinidos para cada tipo de nodo genérico. Lo que sigue es un resumen de estos servicios:

Servicios Comunes Todos los nodos son capaces de identificar al usuario y de registrar sus accesos mediante el establecimiento de sesiones, además son capaces de proporcionar información acerca de su estado, los servicios que ofrecen y las funciones que pueden ejecutar.

Servicios de los nodos Tool Los nodos Tool generalmente no ofrecerán servicios propios al resto de los nodos del entorno con excepción de los servicios comunes. Sólo se ha definido un servicio inicialmente, que permite preguntar a un nodo Tool por información relativa a un usuario, como datos personales, de acceso, o acerca de la intención de uso de un componente reutilizable, etc. Este servicio será especialmente útil de cara a integrar de forma transparente en el entorno, repositorios que respondan al modelo referencial a través de nodos FileBroker, como se verá más adelante en el apartado 4.3.

Servicios de los nodos Broker Las funciones principales de los nodos Broker incluyen el encaminamiento y distribución de los mensajes en el entorno, y la verificación de los mismos.

Para distribuir los mensajes, mantienen un registro de los nodos activos en un entorno, con información, envíada por los propios nodos, acerca de la ubicación y el estado de cada uno. A través del Broker un nodo puede conocer el estado de cualquier otro nodo.

En cuanto a la comprobación de los mensajes, el Broker se encarga de validar que los objetos intercambiados responden al modelo de componente reutilizable empleado. Se garantiza por lo tanto, que los nodos sólo recibirán objetos pertenecientes al modelo con el que son capaces de trabajar.

Bajo petición de un nodo, un Broker puede almacenar temporalmente los objetos intercambiados en una comunicación. La principal utilidad de este servicio es que el Broker actúa como una caché de objetos, reduciéndose considerablemente el tráfico de red.

Otro servicio importante del Broker es el control de acceso a los servicios del entorno según el rol del usuario solicitante. Se ha definido una política general de accesos, según los roles que marca el desarrollo basado en reutilización. Dicha política puede extenderse en cada nodo añadiendo nuevas restricciones, resultando el control de acceso en una conjunción entre las restricciones predefinidas y las de la extensión.

Servicios de los nodos Processor Los nodos Processor pueden ofrecer cualquier tipo de servicio no interactivo sobre un objeto o conjunto de objetos. Se instanciarán para ofrecer servicios específicos, que no están predefinidos inicialmente. Posibles servicios pueden ser comprobar un conjunto de objetos para verificar que pertenecen al modelo de componente reutilizable, realizar cálculos de métricas, etc.

Servicios de los nodos Repository Los nodos Repository ofrecen servicios de repositorio referencial de componentes reutilizables, pero diseñados para trabajar directamente con el modelo de componente reutilizable empleado. En un principio, se han definido sólo los servicios básicos: consulta, recuperación, modificación y almacenamiento.

El servicio de consulta es la base de otros, y permite realizar consultas construídas mediante un modelo de objetos. En estas consultas, se realizan las búsquedas, no sólo a partir de un objeto determinado, sino también por la existencia de relaciones con otros objetos. Tampoco se limitan los resultados a objetos de un único tipo, sino que se pueden obtener objetos de diferentes tipos, relaciones y objetos conectados con las relaciones existentes entre ellos. Este servicio puede depender del esquema de clasificación, pero si esté se incluye en el metamodelo de componente reutilizable, que puede expresarse en el protocolo del entorno, el servicio de consulta se puede definir independientemente del esquema de clasifcación.

El servicio de recuperación permite obtener componentes reutilizables siguiendo las reglas del modelo de reutilización empleado, es decir extrayendo además de los componentes estrictamente solicitados, todos aquellos relacionados con ellos que indica el modelo, normalmente según el tipo o la semántica de las relaciones que los unen.

3. Protocolo de intercambio de objetos y servicios

La coexistencia de herramientas y entornos heterogéneos trabajando sobre un mismo modelo de componente reutilizable y la necesidad de comunicación entre ellas, principalmente la comunicación con repositorios, hace necesario el desarrollo de un protocolo común que permita esta comunicación. La definición del protocolo (formato

de mensajes, conexiones, etc.) debe ser lo suficientemente general, para que pueda ser utilizado por el mayor conjunto posible de herramientas y aplicaciones. Por otra parte, ciertos aspectos del protocolo deberán ser específicos para poder recoger el modelo de componente reutilizable, de forma que, por ejemplo, se puedan realizar las comprobaciones sobre la corrección respecto al modelo de los elementos intercambiados.

El protocolo desarrollado define los objetos y servicios que pueden intercambiarse entre los nodos del entorno, así como el formato de los mensajes que se utilizan para intercambiarlos.

Para permitir que los mensajes puedan procesarse fácilmente con independencia de la tecnología utilizada en la implementación del protocolo, se ha optado por utilizar XML como lenguaje de codificación para los mismos.

Además, un protocolo para un entorno de estas características debe permitir reducir lo máximo posible el tráfico de red y el uso de recursos, dedicados a la comunicación. Para ello, se ha intentado que se pueda envíar el máximo volumen de objetos y se puedan realizar el máximo número de servicios con el mínimo número de mensajes.

3.1. Formato de los mensajes

El formato de los mensajes responde principalmente a estas necesidades:

- Permitir el intercambio de cualquier conjunto de objetos pertenecientes al modelo de componente reutilizable junto con las relaciones entre ellos.
- Permitir que mediante una única petición y a partir de un mismo conjunto de datos de entrada, se puedan realizar varias operaciones simples (servicios), que constituyan conjuntamente una operación más compleja.
- Delimitar y separar los elementos dependientes del modelo de reutilización.

El formato general puede verse en la figura 1. Puede distinguirse:

la parte independiente del modelo de componente reutilizable: constituída por las cabeceras del mensaje (MIPHeader), las cabeceras de los bloques de petición (Request, Result y Dialog) y los datos de gestión del protocolo (XMD)

la parte dependiente del modelo de componente reutilizable (X-REM): consistente en el conjunto de objetos y relaciones pertenecientes al modelo

El conjunto principal de datos de trabajo de los servicios es el modelo de componente reutilizable. Además de estos datos, los servicios pueden tomar como argumentos y generar como resultados, datos pertenecientes a tipos simples, concretamente a los tipos básicos definidos en el lenguaje Java [6].

3.2. Modelo de componente reutilizable en XML

Para permitir el intercambio de objetos pertenecientes al modelo de reutilización, se ha trasladado un modelo de componente reutilizable completo a XML, partiendo de la descripción del modelo en la forma de un diagrama de clases UML. La manera en que se ha definido en XML el modelo de reutilización, hace posible la ampliación posterior del modelo, ya que la traducción del modelo UML al modelo XML es inmediata, y los

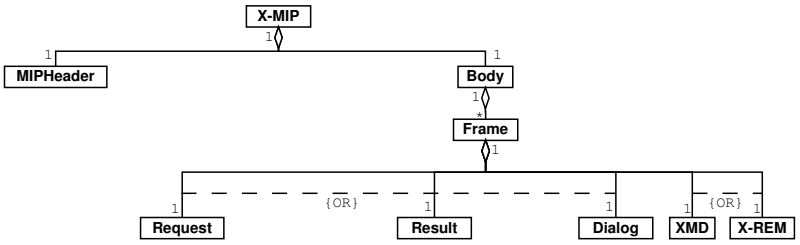

Figura 1. Estructura general de los mensajes

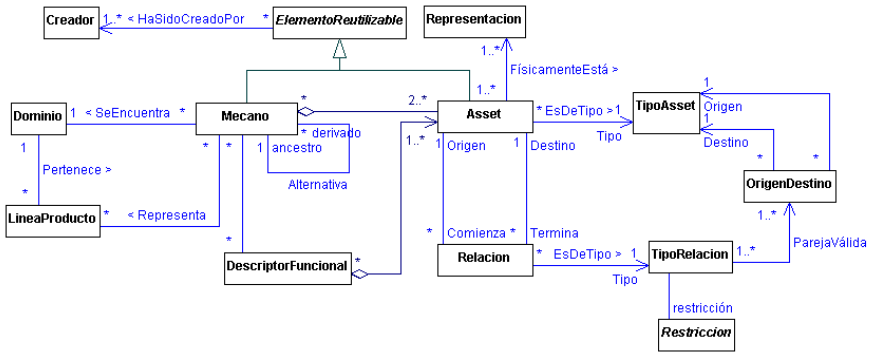

Figura 2. Modelo de MECANO

cambios o ampliaciones en el modelo original pueden trasladarse al modelo XML. El formato, permite intercambiar conjuntos de objetos y las relaciones entre ellos.

El modelo de componente reutilizable que se ha seleccionado, es el modelo de reutilización de MECANO (ver figura 2, del grupo GIRO de la Universidad de Valladolid [7],por ser el modelo para el que se dispone en nuestro grupo de bases teóricas y herramientas. El modelo de MECANO presenta una estructura de reutilización compleja que da soporte a un elemento software reutilizable definido en diferentes niveles de abstracción y que permite que tanto el desarrollador para reutilización como el desarrollador con reutilización tengan una visión más cercana a la reutilización de subsistemas a la hora de ponerla en práctica, cada uno desde la perspectiva propia del rol que desempeña.

A continuación se describe cómo se ha trasladado dicho modelo, patiendo de su definición semiformal en un diagrama de clases UML, a un formato de documento XML.

Se ha declarado en primer lugar, un nodo raíz que contiene los objetos y las relaciones del modelo, un nodo Object que contiene todos los objetos pertenecientes a las clases del modelo y un nodo Association que contiene todas las relaciones.

```
<!ELEMENT X–REM (Object*, Association*)>
<!ELEMENT Object (Asset| Mecano| Author| Representation| Domain| ... ) >
<!ELEMENT Association (assocFrom_Asset| assocFrom_Mecano| ... ) >
```

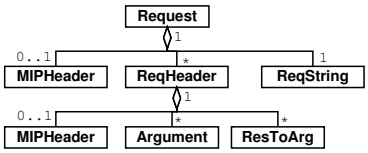

Figura 3. Estructura del bloque de petición *Request*.

Cada clase del modelo, se representará en XML como un elemento del mismo nombre que la clase. Los atributos de cada clase, se incorporarán como elementos y todos se declararán como opcionales. Se muestra sólo el ejemplo de la clase Asset:

```
<!ELEMENT Asset (Name?, Abstract?, AbstractionLevel?, ...)>
```

Por último, los objetos del documento XML (elementos Object) contienen información necesaria para la gestión del protocolo, como un identificador específico para poder establecer referencias a ellos. Las asociaciones (elementos Association), contienen las referencias a los objetos origen y destino de las mismas. Para poder incluir las relaciones en el documento, manteniendo la semántica del modelo, se declaran bajo Association todos los posibles orígenes de asociación como: asocFrom_ClassName, que actuará como contenedor de las relaciones. En cada uno de estos elementos se incluyen las clases de destino de la relación y se les añade la cardinalidad que establece el modelo de reutilización.

```
<!ATTLIST Object
  objID                ID                          #REQUIRED
>
<!ATTLIST Association
  sObjLink             CDATA                       #REQUIRED
  tObjLink             CDATA                       #REQUIRED
>
<!ELEMENT asocFrom_Asset  (asocTo_Author*, asocTo_Representation*, ...)>
<!ELEMENT asocTo_Author    EMPTY>
```

3.3. Solicitud de servicios

La solicitud de servicios se realiza mediante el bloque de petición *Request*, cuya estructura se muestra en la figura 3.

Dentro de este bloque, se utiliza un lenguaje sencillo que permite solicitar la ejecución combinada de varios de ellos (elemento Reqstring). Una petición de servicios realizada mediante este lenguaje, está constituída por los identificadores de cada servicio, enlazados mediante operadores que expresan ejecución en secuencia, encadenamiento (composición de servicios) y operaciones condicionales.

En una misma petición, y mediante un único mensaje, se puede solicitar, por ejemplo, la apertura de una sesión, el almacenamiento de un conjunto de elementos reutilizables, la consulta de esos componentes para obtener sus identificadores en el repositorio y el cierre de la sesión:

```
<reqString>
  OPENSESSION;STORE.QUERY;CLOSESESSION;
</reqString>
```

El protocolo permite también indicar el nodo en el que se desea ejecutar cada servicio, lo que ofrece la posibilidad por ejemplo, de encadenar servicios entre varios nodos. Para ello se incluirán cabeceras de mensaje en el bloque de petición (elementos MIPHeader.

En la cebecera del bloque de petición se incluye una subcabecera para cada servicio solicitado, dónde se añaden los datos de entrada no pertenecientes al modelo de componente reutilizable (elementos Argument), información acerca de como debe realizarse la composición de servicios (elemento ResToArg), y se puede incluir también la cabecera MIPHeader que indica qué nodo que debe ejecutar ese servicio.

3.4. Extensión y adaptación del protocolo

Las posibilidades de adaptación y extensión del protocolo son varias. La más importante es, quizá, la posibilidad de extender el modelo de componente reutilizable o incluso de utilizar otros modelos. También existe la posibilidad de utilizar el protocolo a través de diferentes medios de transmisión (conexiones) y la posibilidad de aplicar filtros de compresión y encriptación para mejorar el rendimiento y la seguridad del mismo.

4. Diseño y solución basado en frameworks

Para permitir la reutilización de mecanismos y servicios a la vez que la independencia del modelo de reutilización respecto a la implementación y la tecnología particular de cada nodo, se han diseñado estos utilizado una arquitectura basada en frameworks [8].

La arquitectura ofrece, por un lado, la escalabilidad de funciones y servicios en los nodos, permitiendo que un nodo incorpore nuevos servicios, y por otro lado, la adaptabilidad de los servicios, permitiendo que la funcionalidad independiente del modelo de reutilización pueda recaer en diferentes soluciones tecnológicas.

El framework implementa las funcionalidades básicas para trabajar con objetos del modelo de componente reutilizable y las funciones necesarias para la ejecución de los diferentes servicios, que son independientes de la implementación incluyendo puntos de variabilidad como parte de métodos plantilla.

En esta sección se presenta una visión general del framework y se detallan algunos aspectos de éste a modo de ilustración. Como parte de esta presentación también se introducen detalles acerca de los nodos FileBroker, como se mencionó en el apartado 2.1. En la figura 4 aparece una visión general del framework mediante su descomposición modular.

4.1. Framework genérico para los nodos del entorno de desarrollo

La figura 4 representa el framework genérico que mantiene la funcionalidad de los nodos. Cada nodo dispone de la funcionalidad básica para enviar y recibir mensajes empleando el protocolo particular del entorno, y de realizar y solicitar servicios sobre objetos pertenecientes al modelo de componente reutilizable empleado. Para ello se han diseñado de forma independiente:

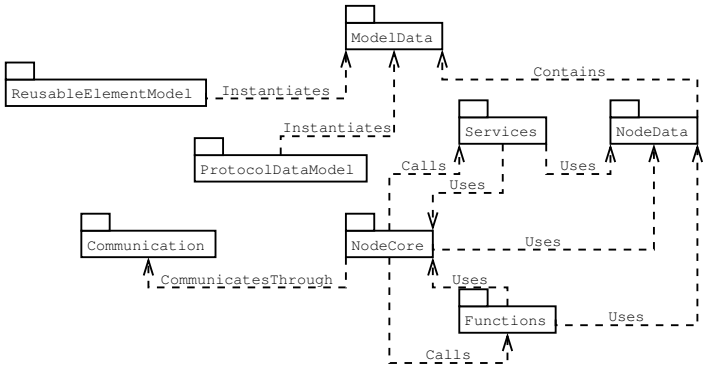

Figura 4. Visión general del framework

- las funciones relacionadas con la comunicación (paquete *Communication*)
- las estructuras de datos que representan el formato de los mensajes (paquete *NodeData*)
- las estructuras de datos que representan el modelo de componente reutilizable (paquete *ReusableElementModel*)
- el núcleo de los nodos que gestiona toda la funcionalidad del nodo
- la definición abstracta de las funciones que es capaz de ejecutar el nodo

La funcionalidad que es capaz de ejecutar un nodo sobre modelos de componentes reutilizables, se divide en *funciones* y *servicios*. Las funciones engloban a cualquier funcionalidad del nodo y son ejecutadas desde otras funciones del propio nodo, por ejemplo, la validación de mensajes que realiza un nodo Broker. Los servicios son funciones con la capacidad añadida de que pueden ser solicitados desde otros nodos y una vez finalizados devuelven los resultados generados al nodo que los solicitó.

La adaptabilidad del nodo, permite configurar qué funciones y servicios será capaz de ejecutar y dependerá tanto del tipo del nodo como de las configuraciones adicionales que realice el desarrollador que lo instancie. La escalabilidad de un nodo, permite que un desarrollador, incluya nuevas funciones y servicios en los nodos. Inicialmente, se han elaborado las pertinentes configuraciones por defecto, para instanciar los diferentes tipos de nodos con los servicios predefinidos, comunes y específicos, correspondientes.

Los servicios predefinidos, se implementan con puntos de variabilidad que permiten añadir la parte dependiente de la implementación. En el siguiente apartado se presenta un ejemplo.

4.2. Un ejemplo de adaptabilidad de un servicio: sección del framework para el servicio de consulta

La ejecución de un servicio comprende diversas tareas que al ser analizadas y clasificadas pueden agruparse en las que dependen exclusivamente del protocolo y el modelo de reutilización (en particular del modelo de componente reutilizable), y las que dependen de determinadas soluciones tecnológicas que realizan las funciones de más

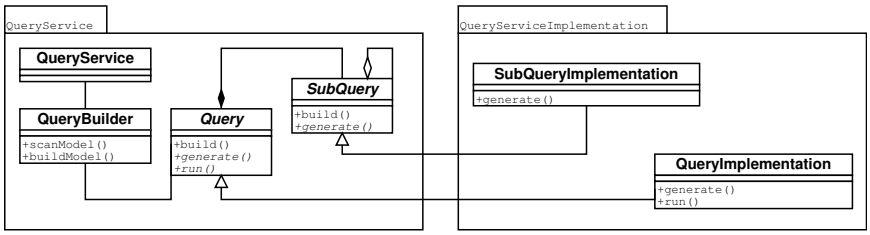

Figura 5. Fragmento del framework relativo a los servicios, en particular al servicio de consulta

bajo nivel del servicio. A la hora de abordar la solución basada en frameworks, se ha analizado cómo se puede describir el servicio separando completamente las tareas de un tipo y de otro, dejando implementadas las primeras y como métodos abstractos las segundas, e introduciendo la lógica que las une como métodos plantilla.

Por ejemplo, en el caso de un nodo Repository, las funciones de más bajo nivel son implementadas por un almacen de datos, que ofrecerá servicios de almacenamiento, consulta, modificación, etc. Este almacen de datos, puede ser cualquier solución comercial de bases de datos orientadas a objetos o relacionales, y la forma en que se soliciten las operaciones y se obtengan los resultados puede variar considerablemente de unas a otras. Incluso para almacenar componentes pertecientes a un mismo modelo de componente reutilizable, pueden existir bases de datos con diferentes esquemas.

Por el contrario, existen una serie de funciones independientes del almacén de datos utilizado. En el servicio de consulta, estas funciones incluirán:

- convertir el modelo de entrada en un grafo dirigido
- recorrer el grafo drigido generando subconcultas y consultas
- ejecutar la consulta utilizando una tecnología particular
- construir el grafo de resultados
- convertir el grafo de resultados en un modelo de salida

La separación entre estas funciones puede verse en el fragmento del framework relativo al servicio de consulta en la figura 5.

En el paquete *QueryService* se definen las funciones propias del servicio y dependientes del modelo que realizan las tareas mencionadas anteriormente para analizar el modelo, construir a partir de este una consulta genérica, ejecutarla y formatear los resultados. Para la construcción de esta consulta genérica se utiliza un lenguaje al estilo de sql.

Finalmente, la implementación de las operaciones de más bajo nivel, dependientes de la solución particular de almacenamiento de datos empleada, es realizada por las clases que heredan, al instanciar el framework, de aquellas que construyen las consultas genéricas. Estas operaciones de bajo nivel, toman como entrada las consultas genéricas citadas anteriormente y las traducen a la sintaxis y el esquema particular del almacén de datos empleado.

4.3. Repositorios de Contención

Un repositorio referencial no mantiene los ficheros de los componentes reutilizables, sino que gestiona un catálogo de información, conteniendo las direcciones desde dónde el desarrollador o proveedor ofrece dichos componentes a sus clientes. Esto pasa generalmente por utilizar una URL como referencia para recuperar el fichero de un componente reutilizable, lo que supone en la mayoría de los casos que el fichero requerido se puede obtener directamente desde esta dirección, y que de algún modo es accesible publicamente. En ocasiones puede ser que la URL no sea suficiente para obtener un fichero. Puede ser que el proveedor requiera el envío de un formulario para recopilar información acerca del uso que se le va a dar al componente, la autentificación de un usuario autorizado, o la utilización de una aplicación cliente específica para recuperar el fichero en cuestión.

En una URL se puede añadir información diversa, como un login y un password de un usuario, y puede ir incluso dirigida a una aplicación web que resuelva la petición, incluyendo una 'querystring' que contenga la información requerida por el proveedor. Pero en ambos casos, se requiere del envío de información variable y dependiente del usuario. Esta información no se puede incluir estáticamente en la referencia asociada al componente reutilizable que aparece en el catálogo.

Para mantener la posibilidad de acceder al fichero del componente a partir de su URL y permitir, de la manera más transparente posible al usuario, el envío de la información adicional requerida por el proveedor, se propone la siguiente solución.

La idea que se plantea consiste en que la URL que figure como referencia del componente reutilizable no tenga que identificar directamente el recurso, sino que pueda apuntar a una ubicación intermedia dónde se puedan alojar herramientas que permitan resolver la ubicación definitiva y realizar las tareas pertinentes (envío de un formulario, solicitud de datos al usuario de forma interactiva, etc.) para obtener el fichero que almacena el proveedor.

Un repositorio de contención integrado en el entorno de desarrollo del modelo referencial: el nodo FileBroker. La parte central de la propuesta para integrar un repositorio de contención es incluir en el entorno de desarrollo del modelo referencial, un nodo que ofrece servicios de contención local y de recuperación de componentes reutilizables de forma transparente.

Este nodo, denominado FileBroker, actuará como interfaz entre la URL de un objeto, en el modelo referencial, y el fichero del componente, ubicado en el repositorio de contención local o en el del proveedor. Al realizar la recuperación, se añadirá de forma transparente toda la información adicional necesaria para o se realizarán todas la tareas que se requieran para obtener el fichero del proveedor.

Un nodo FileBroker almacenará las referencias reales a los ficheros y generará URL's relativas consistentes en su propia dirección y algún identificador adicional. Estas URL's relativas serán las que se almacenen en el repositorio referencial,

El FileBroker ofrecerá también un servicio de repositorio de contención local para los ficheros de componentes, orientado a proveedores que no puedan o no quieran mantener sus propios repositorios.

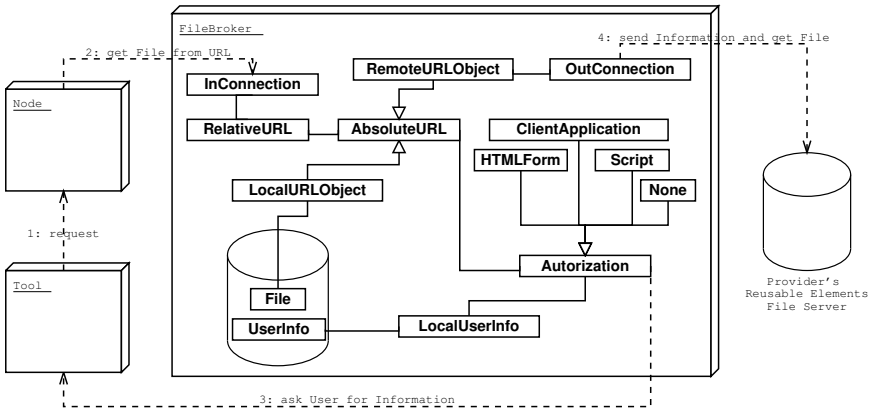

Figura 6. Arquitectura propuesta para desarrollar nodos de tipo FileBroker

Además, ofrecerá a los usuarios un registro en el que puedan almacenar sus datos o los relativos a las cuentas que posean con los proveedores de componentes, de esta manera el nodo FileBroker podrá extraer sus datos automáticamente de este registro local en tiempo de recuperación, sin tener que preguntar al usuario. El único servicio predefinido en los nodos Tool (ver apartado 2.2) se introdujo en este sentido, para permitir que el nodo FileBroker pudiera solicitar información a los usuarios.

La figura 6 muestra la propuesta de arquitectura que recoge la introducción de nodos del tipo FileBroker, y que servirá tanto explicar gráficamente sus funciones como para orientar su desarrollo posterior.

5. Puesta en marcha de un entorno distribuido para el desarrollo basado en reutilización

A lo largo de los últimos años en nuestro grupo hemos venido desarrollando trabajos que han pasado por la definición del Modelo de Reutilización GIRO (MRG) basado en la definición de un modelo de componente reutilizable denominado Mecano [7,9,10] de un proceso de desarrollo basado en reutilización [11], así como en complementar éstos con trabajos específicos en las fases de Ingeniería de Requisitos [12], diseño e implementación basado en frameworks [13] y en certificación de la calidad de componentes reutilizables [14]. Por otra parte, se había trabajado en el desarrollo y puesta en marcha de la biblioteca de reutilización GIRO [15].

Una vez definidos todos los aspectos de la propuesta de entorno genérico para el desarrollo basado en reutilización, se ha obtenido una primera versión de implementación de éste. A partir de esta versión, se realiza la puesta en marcha de una instancia específica de entorno distribuido para el desarrollo basado en reutilización según el modelo MRG. Esto tiene un doble propósito, hacer una primera validación del entorno genérico propuesto, a la vez que dotar al modelo MRG de un soporte operativo útil, que

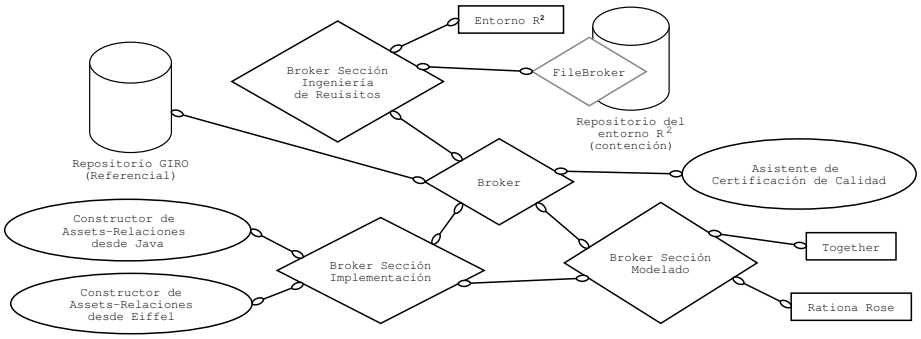

Figura 7. Puesta en marcha de un entorno real de colaboración para el desarrollo basado en reutilización

permitiera la integración de herramientas establecidas de desarrollo de software con las herramientas específicas que hemos desarrollado.

En la Figura 7 se muestra la configuración específica que hemos propuesto, con ella cubrimos el soporte a las diferentes vistas del proceso de desarrollo. Se ha realizado la instanciación del framework para herramientas propias, el entorno R^2, una herramienta para asistir en la obtención de certificación de calidad, y herramientas que controlan el código implementado en Java y Eiffel desde el punto de vista de las relaciones definidas en el modelo de Mecano. Por otra parte, se destacan las instanciaciones realizadas para las herramientas CASE externas (altamente utilizadas) Together y Rational Rose, mediante sus mecanismos de extensibilidad.

Se han realizado pruebas internas, aunque aun no he han podido realizar pruebas de puesta en marcha en un entorno real de desarrollo.

6. Conclusiones y trabajo futuro

En este trabajo se ha presentado una propuesta de entorno distribuido como soporte al desarrollo basado en reutilización. Dicha propuesta consta de la definición de una arquitectura genérica, un protocolo y el diseño e implementación de un framework que permite, mediante su instanciación, la reutilización de mecanismos y servicios a la vez que la independencia de la implementación y la tecnología particular de los nodos específicos que se conecten. El modelo de componente reutilizable empleado es un modelo definido en nuestro grupo.

Se ha realizado la puesta en marcha de un entorno de pruebas mediante la adaptación de las herramientes ya existentes para integrarlas mediante el entorno de desarrollo, algunas herramientas son propias y específicas para trabajar con el modelo de reutilización de nuestro grupo y otras son herramientas CASE establecidas que se han adaptado a través de sus mecanismos de extensibilidad.

Es deseable, y se plantea a llevar a cabo en el futuro, realizar pruebas de implantación real, dónde entre otros, se puedan realizar análisis de rendimiento y eficiencia, lo

cual requeriría también de estudios empíricos y elaboración de experimentos sobre el impacto de su utilización en entornos reales de desarrollo.

Queda abierta también la posible evolución del entorno de desarrollo y explorar cómo se adapta la arquitectura y el protocolo a diferentes modelos de componente reutilizable.

Referencias

1. Kewley, J.M., Prodan, R.: A Distributed Object-Oriented Framework for Tool Development. En Li, Q., Firesmith, D., Riehle, R., Pour, G., Mayer, B., eds.: Proceedings of the 34th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS 34), IEEE Computer Society Press (2000) 353–62
2. Ler, C., Rosenblum, D.: Wren—an environment for component-based development (2001)
3. Karlsen, E.W.: The UniForM WorkBench — a Higher Order Tool Integration Framework. En: AFM'98 — International Workshop on Current Trends in Applied Formal Methods, Boppard, Germany, October 7–9 (1998)
4. Girardi, M.R.: Main approaches to software classification and retrieval. En: Actas del Curso de Ingeniería del Software y Reutilización: aspectos dinámicos y generación automática. (1998)
5. Karlsson, E., ed.: Software reuse. A holistic approach. John Wiley & Sons (1995)
6. Arnold, K., Gosling, J.: The Java Programming Language. Java Series. Sun Microsystems (1996)
7. García, F.: Modelo de reutilización soportado por estructuras complejas de reutilización denominadas mecanos. PhD thesis, Departamento de Informática y Automática, Universidad de Salamanca (2000)
8. Fayad, M.E., Schmidt, D.C.: Building Application Frameworks: Object-Oriented Foundations of Framework Design (1999) John Wiley & Sons.
9. García, F.J., Marqués, J.M., Laguna, M.Á., Maudes, J.M.: Influencia de las relaciones entre elementos software reutilizables en la generación de mecanos. En Toval, A., Nicolás, J., eds.: Actas de Las III Jornadas En Ingenería Del Software (JIS'98), Murcia, 11-13 de Noviembre de 1998 (1998) 155–166
10. García, F.J., Marqués, J.M., Maudes, J.M.: Mecano: Una propuesta de componente software reutilizable. En Díaz, O., Lopistéguy, P., eds.: Actas de Las II Jornadas de Ingeniería Del Software, Donostia-San Sebastián, España, 3-5 septiembre de 1997 (1997) 232–244
11. Laguna, M., González-Baixauli, B., López, O., García, F.: Introducing Systematic Reuse in Mainstream Software Process. En: Euromicro 2003 (accepted), Antalya, Turkey (2003)
12. López Villegas, O.: Reutilización de Requisitos para Incremento de la Calidad y Productividad en el Desarrollo de Especificaciones. PhD thesis, Departamento de Informática, Universidad de Valladolid (2003)
13. Prieto, F., Crespo, Y., Marqués, J., Laguna, M.: Mecanos y análisis de conceptos formales como soporte para la construcción de frameworks. En: Actas de las V Jornadas de Ingeniería de Software y Bases de Datos (JISBD'2000), 8-10 Noviembre de 2000 - Valladolid (España) (2000) 163–175
14. Manso, M.E., García, F.J., Romay, M.P., Marqués, J.M.: Modelo de cualificación y auditorías de elementos reutilizables de un repositorio. En Botella, P., Hernández, J., Saltor, F., eds.: Actas de Las IV Jornadas de Ingeniería Del Software Y Bases de Datos (JISDB'99), 24-26 Noviembre de 1999 - Cáceres (España) (1999) 355–366
15. Hernández, C., García, F., Laguna, M.: La biblioteca de reutilización giro. En: I Jornadas de Trabajo DOLMEN, 12-13 Junio de 2001- Sevilla (España) (2001) 102–112