Patrones de Coordinación: Propuesta para la comprensión y utilización de Patrones de Diseño en el desarrollo de Sistemas Coordinados[1]

Pedro L. Pérez-Serrano
Departamento de Informática
Área de Lenguajes y Sistemas Informáticos
Universidad de Extremadura
QUERCUS Software Engineering Group
plperez@unex.es

Marisol Sánchez-Alonso
Departamento de Informática
Área de Lenguajes y Sistemas Informáticos
Universidad de Extremadura
QUERCUS Software Engineering Group
marisol@unex.es

Resumen

La necesidad de obtener soluciones software cada vez mas complejas, de mayor calidad y en un tiempo más reducido, ha propiciado el desarrollo de sistemas basados en componentes, en los cuales la complejidad reside en las necesidades de comunicación y coordinación entre ellos. Ello ha motivado el estudio de nuevas técnicas adecuadas a estas necesidades. Entre ellas destaca el uso de patrones, como una solución a un problema o parte de un problema probada y aceptada, y además diseñada para ser reutilizada, de manera que pueda ser aplicada cuantas veces se quiera en tantos sistemas como sea adecuada su utilización. Con ello además de dar solución al problema, se reduce el tiempo de desarrollo, y se mantiene la calidad.

En este artículo se propone el uso de patrones específicos para diseñar sistemas coordinados, basados en modelos de coordinación exógenos que faciliten la representación de las interacciones entre componentes a nivel de diseño, diferenciando los aspectos funcionales y de comportamiento. Los patrones de coordinación propuestos se definen en UML y pueden ser reutilizados en otros sistemas de coordinación, independientemente del lenguaje o modelo de coordinación que se aplique, de la plataforma de implementación y del entorno de desarrollo, siguiendo el enfoque MDA.

En particular, la propuesta integra el uso de los patrones de coordinación en el entorno formal de desarrollo para sistemas de cooperación definido por COFRE.

1. Introducción

El desarrollo estructurado [1], que ha tenido y sigue teniendo una amplia vigencia, permite hacer una clara separación entre el diseño y la implementación, y considera que la mayoría de los errores se producen durante el proceso de diseño. Se supone que los requisitos se definen de antemano y quedan inalterables a lo largo del desarrollo del proyecto. Pero en la realidad, en la mayoría de los proyectos, estos requisitos iniciales se modifican durante el ciclo de vida, casi siempre motivados por cambios realizados por el cliente, bien debidos a su falta de decisión o bien porque al inicio del ciclo de vida no se ha aportado toda la información necesaria, llegando finalmente a un sistema cuyos requisitos son muy diferentes de los que inicialmente fueron definidos.

Con objeto de solventar esta deficiencia, surge la ingeniería de requisitos [2] como una disciplina relevante dentro de la ingeniería del software, cuyo propósito es llevar a cabo las actividades de

[1] [1] Este trabajo está financiado por el proyecto CICYT con referencia TIC 02-04309-C02-01

adquisición, análisis y validación de los requisitos del usuario [3]. El resultado de estas actividades es la especificación de los requisitos del sistema o modelo conceptual, y que constituye, la definición del problema.

El modelo conceptual debe validarse, para determinar si las especificaciones del sistema han capturado de forma completa y correcta los requisitos establecidos por los usuarios. Es decir, si el sistema resuelve el problema para el cual se ha desarrollado.

Además de esta exigencia, la necesidad de construir sistemas cada vez más complejos, junto con la continua aparición de nuevos paradigmas de la programación y el gran desarrollo experimentado por las redes de comunicación ha potenciado el desarrollo de sistemas distribuidos. Todo ello unido a las demandas del mercado requiriendo sistemas de software de calidad y de desarrollo rápido, ha propiciado la construcción de sistemas basados en la reutilización de componentes [4]. Al mismo tiempo, en los sistemas distribuidos, la necesidad de cooperación y coordinación entre diferentes componentes ha aumentado considerablemente el desarrollo de sistemas coordinados.

Con objeto de facilitar la interacción entre los componentes del sistema, se han propuesto una variedad de lenguajes y modelos de coordinación [5]. Entre los modelos propuestos destacan los modelos exógenos [6], que promueven la separación de los aspectos funcionales y los aspectos de coordinación, de manera que se facilite la reutilización de componentes y se faciliten también los cambios tanto en los componentes funcionales del sistema, como en las interacciones de coordinación que existan entre ellos.

A pesar de estos avances propuestos en el tratamiento del problema de la coordinación, los modelos de coordinación se centran en proporcionar plataformas de implementación, descuidando los aspectos metodológicos. Esto platea nuevos problemas, entre los cuales se encuentra garantizar en todo momento la coherencia entre las distintas etapas dentro del proceso de desarrollo, garantizar dicha coherencia al modificar o añadir nuevas restricciones de coordinación, así como garantizar que el comportamiento de todos los componentes coordinados que constituyen el sistema sea el deseado.

Así mismo, cabe mencionar la complejidad de estos sistemas y la dificultad de comprender por parte de los usuarios cualquier especto de los mismos durante la etapa de diseño.

Aparte de la necesidad de desarrollar sistemas cada vez más complejos, estos son requeridos con un menor tiempo de entrega, con unos requisitos mínimos de calidad, y sujetos a continuos cambios para adaptarse a las nuevas necesidades de su entorno. Para cubrir estas exigencias y agilizar el proceso de desarrollo, se han propuesto técnicas basadas en la reutilización y construcción de sistemas mediante componentes ya desarrollados y probados [7]. En [21] se estudia el aspecto de coordinación en sistemas en los cuales se requieren cambios por motivos de añadir o modificar los mecanismos de coordinación.

En ese sentido, el uso de técnicas que permitan separar los aspectos funcionales de los componentes, de los patrones de iteración entre ellos, puede resultar un campo de estudio prometedor en tanto en cuanto favorecen la reutilización de los componentes y las modificaciones del sistema, de manera que los cambios sólo afectarán a los elementos con los que estén directamente relacionados.

Con estos objetivos se propone una definición de patrones de coordinación para diseñar sistemas coordinados, reflejando a nivel de diseño tanto los aspectos funcionales específicos de cada componente, como los aspectos de coordinación que reflejen las interacciones entre ellos.

La propuesta del uso de patrones de coordinación definidos haciendo uso de los diagramas de UML, se realiza con el ánimo de proporcionar una herramienta independiente que pueda ser integrada en diversos métodos y plataformas de desarrollo e implementada en distintos lenguajes de programación. En particular, en este trabajo nos centraremos en la integración de los patrones de coordinación con el entorno formal de desarrollo definido en COFRE [8].

La estructura del resto de apartados de este artículo es la siguiente: en la sección 2 se comenta el problema de la coordinación así como las motivaciones que han dado origen al estudio de los patrones de diseño en los sistemas coordinados. Seguidamente en la sección 3 se realiza una breve definición de patrones y se tratan de forma específica los beneficios que puede aporta su uso para describir el aspecto de

coordinación. En la sección 4 se presenta nuestra propuesta para la definición de patrones de coordinación en UML. La sección 5 propone una integración en una serie de pasos de la propuesta con el entorno de desarrollo definido en COFRE y en la sección 6 se presentan las conclusiones.

2. Motivaciones

El aspecto de coordinación de un sistema constituido por un conjunto de elementos que cooperan y se comunican para la realización de tareas en común, ha estado ligado en sus inicios a la implementación interna de cada objeto. Esto hacía que la implementación de los componentes fuera cada vez más compleja a medida que el sistema iba creciendo, pues requería incorporar el código asociado con la aparición de nuevas reglas de coordinación o modificación de las ya existentes. Esto además de dificultar la modificación del sistema y la comprensión de los desarrolladores acerca de la funcionalidad proporcionada por cada uno de los componentes, impedía su reusabilidad, ya que en ellos se mezclaba la funcionalidad que debían proporcionar, con las reglas necesarias para su coordinación con el resto de componentes del sistema . Por otra parte, las reglas de coordinación que determinaban el comportamiento coordinado quedaban dispersas entre los componentes del sistema y dificultaba la aplicación de soluciones comunes a problemas de coordinación en otros sistemas, entornos o dominios.

La separación de los aspectos funcionales y de coordinación de un sistema en componentes separados [9] solventa en parte estos problemas ya que la reusabilidad de los componentes funcionales no se ve afectada por la aplicación, modificación o eliminación de reglas de coordinación en el sistema ni por la modificación de los componentes con los que ha de coordinarse, y por otra parte, las reglas de coordinación se aglutinan en componentes específicos de coordinación, que pueden ser a su vez aplicados a otros sistemas o dominios que se rijan por el mismo patrón de coordinación. Además la comprensión de la funcionalidad de los componentes y en definitiva, del sistema en sí, se ve incrementada, al no enmascararse esta con la parte de código necesaria para la coordinación.

Sin embargo la reutilización de componentes de coordinación es de difícil aplicación si no existe un mecanismo que permita describir las reglas y condiciones de coordinación que llevan a cabo, desde el punto de vista de la arquitectura del sistema. En este sentido, la utilización de patrones podría resolver el problema si se hace uso de una notación estandarizada para su descripción, teniendo en cuenta además que el beneficio del uso de patrones en el desarrollo y comprensión de sistemas [10]. El uso de patrones no sólo va a facilitar la reutilización de componentes y la construcción de sistemas basados en componentes [11], sino que además facilita la descripción de la arquitectura de un sistema coordinado independientemente del lenguaje o modelo concreto de coordinación que se aplique e independiente de la plataforma sobre la que se implemente el sistema, siguiendo el enfoque MDA [12]. Ello facilita así mismo la reutilización de dicha arquitectura o parte de ella, independientemente del entorno, plataforma e incluso dominio sobre el que se desarrolle.

3. Patrones de diseño aplicados al problema de la coordinación

Un patrón de diseño es una solución comúnmente aceptada como correcta a un problema de diseño concreto, a la que se ha dado un nombre y que puede ser aplicada en otros contextos ó problemas. La idea de patrón es que éste se adapte perfectamente al nuevo sistema, nunca es el sistema el que debe adaptarse al patrón.

En la Ingeniería del software, los patrones de diseño intentan describir soluciones que han resuelto de forma exitosa problemas comunes del software y cuyas soluciones se han aceptado como correctas [13]. Los patrones de diseño reflejan las estructuras conceptuales comunes a dichas soluciones, que pueden aplicarse de forma repetida cuando se está desarrollando software (analizando, diseñando, e implementando) en un contexto particular. Representan el conocimiento y la experiencia adquirida a lo largo del tiempo por los desarrolladores de sistemas. Con el uso de patrones vamos a poder reutilizar software [14].

Los patrones además nos van a ayudar a analizar y entender el problema. Un desarrollador con conocimiento extenso de un conjunto de

patrones, será capaz de identificarlos cuando se encuentre analizando un problema.

Por lo tanto, permiten a los desarrolladores aprovechar soluciones previas y adaptarlas a sus problemas específicos, nunca al revés, es decir, adaptar nuestros problemas a los patrones existentes.

Un patrón que se describa mediante UML [16] es independiente de la plataforma de desarrollo y del lenguaje, y puede ser reutilizado para adaptarse a otros problemas en los que los componentes del sistema se relacionan siguiendo el esquema que el patrón proporciona. Mediante la definición y utilización de patrones en UML a nivel de diseño, se facilita la comprensión del problema y de la solución a desarrollar.

3.1. Uso de patrones de diseño para describir la coordinación

Una vez entendido y estudiado el concepto de patrones y su funcionamiento, el objetivo inicial de nuestro trabajo se va a centrar en proponer un conjunto de patrones de diseño adecuados a la representación de la coordinación. Estos deben reflejar tanto las circunstancias o eventos bajo las cuales se debe realizar la coordinación, como las acciones sujetas a coordinación, independientemente del lenguaje o modelo de coordinación con el cual vaya a implementarse posteriormente el sistema. Un ejemplo del uso de patrones utilizando UML lo podemos encontrar en [20], donde define un patrón de colaboración para resolver un problema concreto, la coordinación de llegadas y salidas de trenes en una estación.

El uso de una notación estándar como UML para expresar tanto la estructura como el comportamiento de estos patrones, contribuirá a mantener esta independencia y a facilitar su reutilización, al mismo tiempo que contribuye a la comprensión del sistema.

La estructura de los patrones puede representarse mediante los diagramas de clases y su comportamiento en los diagramas de secuencia. Si bien se podrá realizar los diagramas de colaboración, para entender mejor o aclarar los distintos tipos de mensajes que se van a intercambiar entre los elementos coordinados

Mediante esta herramienta de diseño, se intenta reflejar todos los aspectos de coordinación (envío de mensajes, recepción de mensaje, sincronización), así como el comportamiento coordinado del sistema, pudiéndose integrar con entornos de desarrollo que permitan validar, verificar y chequear el sistema así descrito.

Además de poder dar solución al diseño de sistemas de coordinación, mediante el uso delos distintos diagramas de UML se puede generar documentación que facilite tanto la comprensión como la aplicación de la solución expresada en el patrón.

Una vez definidos los patrones, su estructura y comportamiento, podrán ser almacenados en un repositorio de patrones de coordinación. De esta forma se facilitará la reutilización tanto de los patrones de coordinación como de la documentación generada.

4. Patrones de Coordinación

El primer paso a realizar para la definición de patrones de coordinación es determinar su estructura, teniendo en cuenta que nuestro objetivo es tratar la coordinación de forma separada de la funcionalidad de los componentes, siguiendo la propuesta del desarrollo orientado a aspectos.

La adopción de la filosofía de los modelos de coordinación exógenos [6] (en [19] podemos ver el estudio de los aspectos de coordinación para modelos endógenos utilizando los casos de uso en UML) facilita la descripción de la coordinación de forma separada a la funcionalidad de los componentes funcionales del sistema, por ello se ha tomado como referencia el modelo *Coordinated Roles*(CR) que a su vez es el modelo que ha inspirado COFRE, conjunto de herramientas con el que se explica en la siguiente sección la integración de los patrones de coordinación que aquí se describen.

CR se basa en el Protocolo de Notificación de Eventos y en la definición de unos componentes denominados coordinadores, donde se especifica el aspecto de coordinación del sistema. En los coordinadores se definen un conjunto de roles que permiten expresar de forma transparente las reglas de coordinación entre los componentes del sistema. Mediante el protocolo de Notificación de Eventos, los coordinadores tienen conocimiento sobre los eventos que se producen en los componentes u objetos que coordina y en

definitiva controlan a los componentes coordinados.

Los eventos (cuya notificación puede realizarse de forma síncrona o asíncrona) sobre los cuales un coordinador puede tener conocimiento son:

- **RM**: recepción de un mensaje invocando una determinada acción.
- **EoP**: fin del procesamiento de un mensaje invocando una determinada acción.
- **BoP**: comienzo del procesamiento de un mensaje.
- **SR**: el estado de un componente ha alcanzado un determinado valor.

Los patrones de coordinación deben describir tanto la relación de eventos que desencadenan la relación de coordinación como el conjunto de acciones sobre los componentes sujetos a la coordinación, todo ello transparente tanto a los componentes que han de coordinarse, como al propio patrón de coordinación. Para ello, los patrones de coordinación se definen distinguiendo dos partes:

1. La descripción del evento o eventos que han de producirse en uno ó más objetos del sistema y que condicionan la ejecución de una ó más métodos en los objetos con los que han de coordinarse.
2. La descripción de las acciones (métodos) sujetas a restricciones de coordinación, y que pueden variar entre un método simple de un objeto a un conjunto de métodos pertenecientes a diversos objetos, que deben ejecutarse en secuencia, todos ó ninguno, sólo aquellos que hayan recibido la invocación, de forma exclusiva y/o con prioridad, etc.

Así pues, un patrón puede definirse como composición de descripciones de eventos simples (de los tipos anteriormente definidos) y de un conjunto de condiciones ligadas a la ocurrencia de los eventos, que determinan las acciones sujetas a coordinación.

La estructura estática de los patrones de coordinación, se describe mediante diagramas de clase UML, que permiten definir de forma clara las responsabilidades de cada uno de los eventos, así como sus acciones (operaciones) y atributos.

Hay que considerar que un patrón de coordinación no sólo va a permitir describir el aspecto de coordinación entre componentes funcionales, sino que a su vez puede describir la coordinación entre reglas de coordinación definidas mediante otros patrones de coordinación.

El aspecto dinámico de la coordinación se describirá haciendo uso de los diagramas de secuencia y colaboración, que permitirán mostrar tanto secuencial como estructuralmente los mensajes que son necesarios para establecer la coordinación entre los componentes del sistema, dependiendo de los eventos desencadenantes, del modo de comunicación y de las condiciones que rigen la coordinación.

Como objeto de clarificar estas ideas, en las figuras 1 y 2 que se muestran a continuación, se describen los diagramas de clases y de secuencia de un patrón de coordinación que representa la coordinación entre dos acciones de objetos activos.

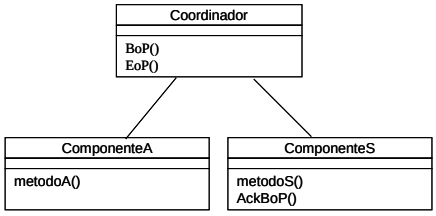

Figura 1. Diagrama de clases del patrón.

El patrón de coordinación describe la secuencia de dos acciones en dos objetos activos distintos, de manera que el *metodoS* en el objeto *Y* pueda llevarse a cabo una vez invocada siempre y cuando el *metodoA* en el objeto *X* haya finalizado. El evento que impone una restricción sobre la ejecución del *metodoS* en *Y* es un evento *EoP* asíncrono. Simplemente se trata de notificar al coordinador cuando el método α ha finalizado su procesamiento. Las acciones sujetas a restricción (*metodoS* en este caso) se representan siempre con un evento *BoP* síncrono es decir, antes de la ejecución ha de notificarse que ha sido invocada. Las notificaciones son gestionadas por un objeto coordinador que en este caso responderá a dicha notificación permitiendo la ejecución de *metodoS*, si previamente ha recibido una notificación *EoP* de la acción del *metodoA*.

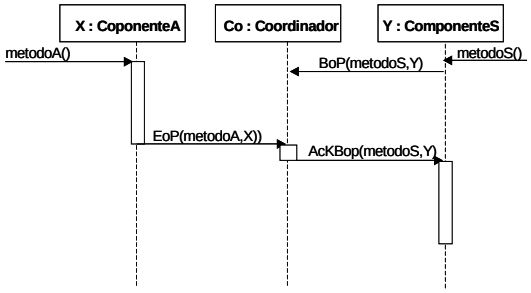

Figura 2. Diagrama de secuencia del patrón.

Este patrón de coordinador simple, podrá aplicarse en aquellos casos en que métodos de dos objetos activos necesiten coordinarse de manera que la ejecución de uno de los métodos esté supeditada a la ejecución previa del otro.

La descripción estática y dinámica del sistema, haciendo uso de los diagramas UML descritos, junto con documentación que describa y facilite su aplicación puede ser almacenada en un repositorio de patrones de coordinación para su posterior reutilización.

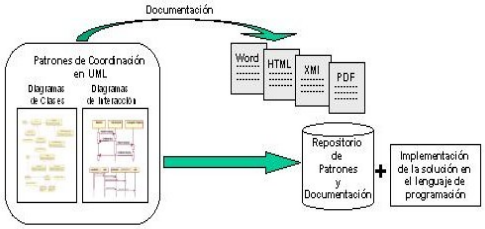

Figura 3. Descripción de Patrones de Coordinación.

La figura 3 resume las actividades principales de nuestra propuesta:

- Definición de los aspectos funcionales mediante diagramas de clase donde se describen tanto los eventos que determinan la relación de coordinación, como las asociaciones con los componentes sujetos a dicha coordinación. De esta forma se define la parte estática de los patrones de coordinación. Los eventos que determinan la ocurrencia de una relación de coordinación entre componentes se pueden definir a partir de los cuatro tipos de eventos anteriormente definidos, determinando las relaciones que existen entre ellos, si más de un evento debe producirse al mismo tiempo para que se lleve a cabo la coordinación, o si más de un evento de forma individual puede disparar la coordinación. Esta combinación de eventos en función de los eventos simples se denominan eventos compuestos. Así mismo, las acciones en los componentes sujetas a coordinación pueden ser más de una y tener que poder producirse bien de forma independiente, bien todas simultáneamente, bien siguiendo un determinado orden o bien de forma exclusiva. La descripción de las acciones que requieren coordinación, podrá por tanto definirse de forma simple, cuando afecta a una sóla acción o componente, o como combinación de más de una de ellas, mediante alguna de las relaciones descritas anteriormente.
- Definición los aspectos dinámicos, donde se refleje el comportamiento coordinado del sistema. Pare ello definiremos utilizaremos los diagramas de interacción de UML. Los diagramas de secuencia, mostrarán la ordenación temporal de los mensajes entre los distintos componentes del sistema, y los diagramas de colaboración facilitarán la comprensión, mostrando en todo momento la organización estructural de los componentes que se envían y reciben mensajes. La cantidad de mensajes necesarios para establecer la comunicación y el orden en que tienen que producirse dependerá del evento producido, del modo de comunicación (síncrona o asíncrona) y de la relación existente entre las acciones de los componentes sujetas a coordinación.
- Una vez generados los aspectos funcionales y de coordinación se generará documentación en diferentes formatos estandarizados que faciliten su comprensión y aplicación, y que podrán ser almacenados junto con los diagramas anteriores, en formato XMI, en un repositorio de patrones de coordinación.

La generación de código a partir de los patrones generados y almacenados en el repositorio dependerá de la plataforma y del modelo o lenguaje concreto de coordinación elegido para la implementación.

5. Integración con COFRE

El uso de patrones de coordinación permite describir a nivel arquitectónico las relaciones de coordinación entre los componentes del sistema. Sin embargo, consideramos interesante la integración de los patrones de coordinación en entornos de desarrollo que nos permitan describir los sistemas coordinados desde la fase de especificación de requisitos.

En este sentido, proponemos la integración de los patrones de coordinación con COFRE, que está constituido por un conjunto de herramientas que permiten especificar los requisitos de un sistema coordinado gráfica y formalmente, al tiempo que permite validar mediante la simulación, el comportamiento coordinado del sistema.

5.1. COFRE

COFRE es un entorno formal para la especificación y validación de sistemas coordinados. Está constituido por un conjunto de herramientas que permiten la representación gráfica y formal del sistema en distintos niveles de abstracción, comenzando con la descripción de los requisitos de coordinación del sistema.

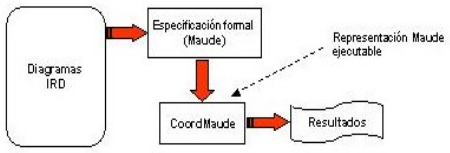

Figura 4. Esquema de COFRE.

Para la descripción de las relaciones de coordinación del sistema COFRE se inspira en el modelo de coordinación exógeno CR que permite describir de forma separada y transparente la coordinación a los componentes funcionales que se coordinan.

Para obtener las distintas representaciones formales que propone el entorno y construir las herramientas de transformación entre ellas, hace uso del lenguaje de especificación algebraica Maude [17]. Maude es un lenguaje reflexivo, lo cual falicita la ampliación de su sintaxis y la definición de herramientas de transformación de representaciones. Al mismo tiempo, el lenguaje está basado en la lógica ecuacional y la lógica de reescritura, lo que permite la ejecución de especificaciones mediante la reducción de términos ecuacionales y la aplicación de reglas de reescritura. Estas propiedades han permitido la utilización del lenguaje no sólo para representar formalmente las especificaciones del sistema en diferentes niveles de abstracción, sino que también es el lenguaje en el que se han realizado las herramientas de conversión y chequeo de la concordancia entre diferentes representaciones.

La figura 4 muestra un esquema resumido de la propuesta realizada por COFRE, cuyas herramientas permiten:

- La especificación de forma gráfica de las relaciones de coordinación entre los componentes funcionales de un sistema coordinado, mediante el uso de los Diagramas de Requisitos Interelemento (IRD).
- La especificación formal de diagrama (IRD), describiendo las relaciones de coordinación de un sistema, en el lenguaje Maude. Esta representación, es independiente del modelo de coordinación que se adopte posteriormente, y corresponde a la especificación formal de los requisitos de coordinación del sistema, de manera que va a ser utilizado como representación de referencia para comprobar la concordancia de representaciones posteriores con respecto a los requisitos iniciales de coordinación.
- La generación a partir de la especificación formal, de la representación del sistema haciendo uso de la sintaxis de CR. Esta representación es convertida, mediante la herramienta *CoordMaude* [19] definida en el propio lenguaje Maude, a una representación detallada, donde se describe el comportamiento coordinado del sistema y que puede ser ejecutada. La ejecución de las especificaciones permite simular el comportamiento del sistema, tomando como entrada distintos estados en los que puede encontrarse el sistema y reproduciendo distintas secuencias de eventos.
- El entorno cuenta además con un chequeador de concordancia, también realizado en Maude, que permite comprobar la consistencia

entre las representaciones formales obtenidas a partir del IRD y las especificaciones ejecutables, debido a que el entorno permite añadir detalles sobre la funcionalidad de los componentes del sistema a partir de cualquier representación, y debe comprobarse que estos no interfieren en el comportamiento coordinado.

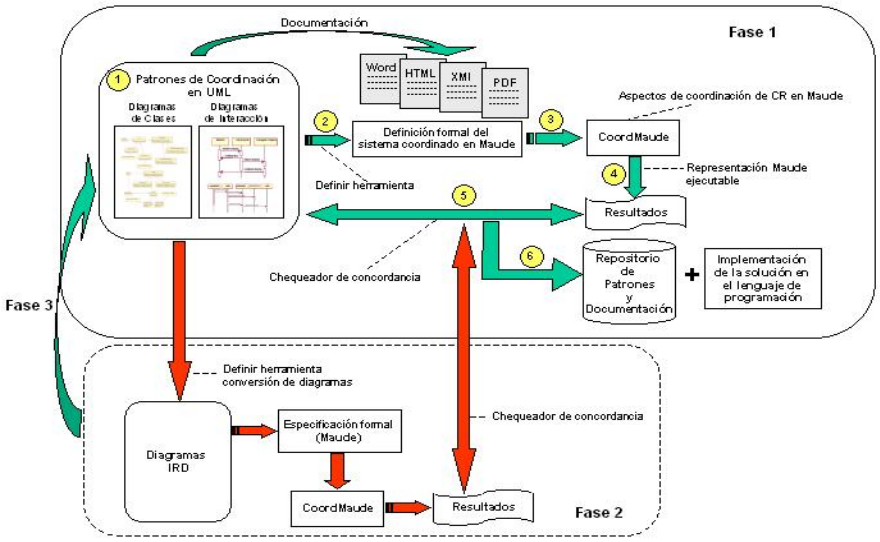

Figura 5. Integración con COFRE.

5.2. Fases de integración

La integración entre COFRE y los patrones de coordinación proporciona un entorno de desarrollo para sistemas coordinados que engloba las fases de especificación y diseño, y que permite la independencia con la fase de implementación del sistema. Dicha integración, aparte de proporcionar las herramientas que facilitan la especificación y validación de los requisitos de coordinación de un sistema, proporciona una descripción de la arquitectura del sistema estandarizada, mediante el uso de patrones de coordinación descritos en UML. Esta descripción facilita el desarrollo de sistemas basados en componentes reutilizables, donde ola reutilización es aplicable tanto a los componentes funcionales como a los componentes que describen el comportamiento coordinado del sistema, y que son descritos mediante los patrones de coordinación.

Para la integración de COFRE y de los patrones de coordinación se proponen tres fases, tal y como se muestra en la figura 5:

Fase 1:

1. Descripción UML de los patrones de coordinación.
2. Descripción formal del patrón mediante el lenguaje Maude.
3. Generación de la representación CoordMaude, a partir de la especificación formal.
4. Ejecución y simulación del comportamiento coordinado del patrón.
5. Verificación de la concordancia con el diseño inicial del patrón. Dicha verificación significará que el modelo resultante de la simulación es una representación fiel del patrón de coordinación descrito, y que durante las transformaciones y la generación de casos

concretos para la simulación del sistema no se han incorporado restricciones ni condiciones que afecten a la coordinación descrita en el patrón.

Fase 2:

Correspondería con la generación a partir de la descripción de un sistema mediante patrones de coordinación, del diagrama IRD equivalente. A partir del diagrama IRD se seguirán los pasos de COFRE y se podrá simular el comportamiento de la solución obtenida del sistema.

La generación detallada del sistema así obtenida debe ser consistente con las especificaciones generadas para cada patrón, llevadas a cabo en la fase 1. De esta forma se garantiza que la composición de soluciones a partir de patrones de coordinación es consistente con el comportamiento coordinado definido por cada uno de ellos, al tiempo que se proporcionan dos niveles de abstracción del sistema:

- Especificación de los requisitos del sistema proporcionado por el IRD y su especificación formal, y
- Diseño del sistema proporcionado por los patrones de coordinación.

Fase 3:

Una vez realizada la correspondencia y la traducción automática de diagramas UML a diagramas IRD, se podrá invertir el proceso de manera que a partir del diagrama IRD de un sistema se generen los correspondientes diagramas UML que describan el sistema, identificando los patrones de coordinación a utilizar, y verificando que el comportamiento de ambas soluciones es perfectamente consistente.

6. Conclusiones

Cada vez los sistemas son más complejos, con mayores requisitos de calidad y exigencias de desarrollo en menor tiempo. En el caso concreto de entornos de cooperativos el comportamiento global del sistema viene determinado no sólo por el comportamiento individual de los componentes que lo constituyen sino también por las reglas de coordinación existentes entre ellos. Para facilitar el desarrollo de este tipo de sistemas y cumplir con los requisitos de calidad y tiempo de desarrollo, se promueve la separación del aspecto de coordinación, de la funcionalidad de cada componente, facilitándose así tanto la reutilización de los componentes funcionales, como de las reglas de coordinación entre ellos.

Pero para que la reutilización de los componentes sea efectiva debe poder producirse también a nivel arquitectónico. En este sentido interesa que la arquitectura del sistema pueda expresarse de forma independiente a la plataforma de implementación y haciendo uso de una notación ampliamente utilizada. Con este objetivo se definen un conjunto de patrones de coordinación en UML, con el objetivo de definir soluciones que nos permitan mejorar la calidad y reusabilidad del diseño de un sistema coordinado.

El uso de los patrones aporta entre otros los siguientes beneficios:

- Los patrones permiten reutilizar una solución correcta a un problema, ya que ha sido resuelta con anterioridad de forma satisfactoria, ello contribuye a minimizar el tiempo de desarrollo, y a garantizar la calidad de la solución.
- El uso de patrones facilita la comprensión del sistema a diseñadores y programadores.

A los beneficios anteriores hay que añadir los beneficios específicos de los patrones de coordinación de nuestra propuesta:

- Facilita la especificación y descripción arquitectónica de un sistema coordinado.
- Permite la definición de los aspectos funcionales y de comportamiento de los componentes de un sistema coordinado de forma separada lo cual facilita a su vez las modificaciones del sistema en cuanto a la aparición eliminación o modificación tanto de componentes funcionales como de las reglas de coordinación que se establecen entre ellos.
- Aborda el desarrollo de un sistema coordinado a nivel arquitectónico, independientemente de la plataforma de implementación.
- Permite documentar de la solución diseñada, facilitando así su comprensión.
- Dicha documentación puede ser almacenada y reutilizada, al igual que los patrones de coordinación y/o acompañando a estos.

La integración del uso de los patrones de coordinación de la propuesta con un entorno para la especificación y simulación del comportamiento de sistemas coordinados como es COFRE, proporcionará además un entorno donde

se podrán simular y ejecutar las soluciones desarrolladas a partir de los patrones, chequeando al mismo tiempo la concordancia con los requerimientos iniciales.

Referencias

[1] W. W. Royce. "Managing the Development of Large Software System Concepts and Techniques". In Proc. IEEE WESCOM, 1970.

[2] R. H. Thayer and M. Dorfman (Eds.) "Software Requirements Engineering". IEEE Computer Society Press, 1997.

[3] C. Rolland and N. Prakash. "From Conceptual Modelling to Requirements Engineering". Annals of Software Engineering, Volume 10, pags 151-176, 2000.

[4] Ian Sommerville. Ingeniería del software. Ed. Addison Wesley, 6ª edición 2002.

[5] Fr∅lund. "Coordinating Distributed Objects. An Actor-Based Approach to Synchronization". The MIT Press. 1996.

[6] F. Arbab. "The IWIM Model for Coordination of Concurrent Activities". P. Ciancarini, C. Hankin (Eds.). First International conference Coordination'96. LNCS 1061. 1996.

[7] Leach, Ronald J. *Software Reuse*. McGraw-Hill, New York, NY, 1997.

[8] Sánchez-Alonso, M. Murillo, J.M. and Hernández J. COFRE: Environment for Specifying Coordination Requirements using Formal and Graphical Techniques. Journal of Research and Practice in Information Technology, Vol. (36) pp: 231-246, Australian Computer Society Inc. 2004.

[9] J.M. Murillo, J. Hernández, F. Sánchez, L.A. Alvarez. "Coordinated Roles: Promoting Re-usability of Coordinated Active Objects using Event Notification Protocols". Third International conference Coordination´99. LNCS 1594. Springer-Verlag 1999.

[10] Gamma, E. et al, Design Patterns. Elements of Reusable Object Oriented Software, Addison-Wesley, 1995.

[11] J.Cheesman, J. Daniels. UML Components. A Simple Process for Specifying Component-Based Software. Addeson-Wesley, 2001.

[12] OMG. Overview and Guide to OMG's architecture, MDA Guide Version 1.0. Object Management Group, 2001. http://www.omg.org/docs/omg/03-05-01.pdf .

[13] J. Vlissides. Pattern Hatching. Design Patterns Applied. Addison-Wesley, 1998.

[14] Wilhelm Schafer, Rubén Prieto-Díaz. Software Reusability. Ed: Ellis Horwood Limited,1994.

[15] Erich Gamma, Ralph Johnson, Richard Helm y John Vlissides. Patrones de Diseño. Addison-Wesley Iberoamericana España, 2002.

[16] Graig Lauman. UML y patrones. Prentice may, 2º edición, 2003.

[17] Clavel, M. Durán, F. Eker, S. Lincoln, P. Martí-Oliet N. Meseguer, J. and Quesada, J. Maude: Specification and Programming in Rewriting Logic. Computer Science Laboratory. SRI International. March'99.

[18] Sánchez-Alonso, M. Clemente, P.J. Murillo, J.M. and Hernández, J. CoordMaude: Simplifying Formal Coordination Specifications of Cooperation Environments. *2nd Workshop on Languages Description Tools and Applications (LDTA'03)*. ENTCS nº 82.

[19] H. Mucini, F. Manchinelli. Eliciting Coordination Policies from Requirements. Techical Report, University of L'Aquila,2002.

[20] D.M. Beder, A. Romanovsky, C.R. Snow, R.J. Stround. An Application of Fault Tolerance Patterns and Coordinated Atomic Actions to a Problem in Railway Sheduling.

[21] L.F. Andrade, J.L. Fiadeiro, J. Gouveia, G. Koutsoukos, A. Lopes, M. Wermelinger. Coordination Pattern for Component-Based Systems. *COORDINATION´00*, G. Catalin-Roman & A. Porto (eds), LNCS 1906, Springer-Velag 2000.