Directrices de estructuración de modelos para Rational Software Modeler, Rational Systems Developer y Rational Software Architect ("Orientación RUP tradicional")

Documentación técnica

Bill Smith, Model Driven Development, IBM Rational Software

Versión 7.0
24 de marzo de 2006

Índice de contenido

1. Introducción

Usuarios a los que va dirigido

Este documento se ha diseñado para dar soporte a los usuarios de los productos Rational Software Architect (RSA), Rational Systems Developer (RSD) y Rational Software Modeler (RSM) (conocidos colectivamente como "**RS*x***"), en especial a los interesados en aplicar la guía de creación de modelos de Rational Unified Process (RUP®) para su uso con RS*x*. Si es usuario de Rational Software Modeler (RSM), le resultará útil este documento pero debe tener en cuenta que algunas secciones reflejan posibilidades que únicamente están disponibles en RSA y RSD.

El documento está centrado en la creación de un nuevo conjunto de modelos en RS*x*. Si es un nuevo usuario de RS*x* pero anteriormente ha utilizado Rational Rose o Rational XDE y piensa importar modelos desde dichos productos, es posible que este documento le sirva de guía para volver a estructurar los modelos importados.

El documento presupone que conoce RUP y UML. También presupone que está familiarizado con los conceptos fundamentales y la "teoría de las operaciones" de RS*x*, sobre los que puede obtener información en el artículo "**The New IBM Rational Design and Construction Products for Rose and XDE Users**"

Objetivo

RUP describe un conjunto de modelos como, por ejemplo, los modelos de guiones de uso, los modelos de análisis y los modelos de diseño que representan perspectivas bien definidas acerca de los dominios de problemas y soluciones de los sistemas. La utilidad de este conjunto de modelos ha sido demostrada en muchos proyectos en la vida real. Incluso si no sigue RUP, vale la pena tener en cuenta estas estructuras de modelos. Este documento describe cómo hacerlas realidad utilizando RS*x*.

También se pueden tener en cuenta otros métodos de creación de modelos, pero las directrices para estructurar modelos con los que dar soporte a los mismos está más allá del ámbito de este documento. Por ejemplo, se puede utilizar un método de "Desarrollo dirigido por negocio" para la creación de modelos para las arquitecturas orientadas a servicios. Puede comenzar por un modelo de proceso de negocio – expresado como, por ejemplo, un modelo de actividad UML 2 – y luego pasar directamente a especificar los contratos de un conjunto de servicios que automatizarán las tareas del proceso de negocio. Este método sugerirá estructuras de modelos diferentes a las descritas en este documento.

Es importante comprender que el proyecto y las estructuras de modelos descritos en este documento son directrices y no imposiciones. Si decide crear un modelo de un artefacto RUP determinado en RS*x* será una consideración personal de su propio proceso de desarrollo. Generalmente, se trata de una decisión específica del proyecto. También debe tener en cuenta que RUP no es un conjunto rígido de normas de proceso. Es una *infraestructura* de proceso en la que formular las definiciones de procesos. Dichas definiciones de procesos pueden estar dentro del rango de muy formales a informales.

El modo en que utilice la creación de modelos UML también puede estar dentro del rango de muy formal a muy informal. Puede optar por tratar los modelos como los planos arquitectónicos formales que se han de seguir estrictamente durante la construcción. O puede tratar los modelos como bocetos que sugieren líneas generales de un diseño pero que se pueden omitir cuando el proyecto pasa a la fase de implementación. RS*x* puede dar soporte a cualquiera de los extremos de estos ámbitos de creación de modelos y procesos. Las directrices contenidas en este documento no pretenden confundirle, sino ayudarle a comprender cómo se utilizan las características de RS*x* para facilitar el proceso que considere mejor.

Tenga en cuenta que RS*x* permite utilizar modelos no simplemente como planos sino como especificaciones, a partir de las cuales se pueden generar automáticamente partes importantes de una

implementación. Esto se realiza utilizando las transformaciones de modelo a modelo y de modelo a código. El uso de las transformaciones para realizar MDD (Model-Driven Development) introduce algunas de las cuestiones especiales relacionadas con las estructuras de modelos. Si va a utilizar los modelos y transformaciones para llevar a cabo el desarrollo dirigido por modelos, también debe buscar los recursos específicos de MDD de RS*x* en el área de diseño y construcción de Rational de developerWorks®.

Ámbito

Este documento describe cómo se representan los artefactos del modelo RUP en RS*x*. También ofrece directrices para las estructuras organizativas internas de dichos artefactos. No *intenta*

- Volver a formular los factores conceptuales subyacentes en los artefactos del modelo RUP ni proporcionar extensas descripciones de los mismos.

- Describir el proceso o las técnicas para especificar el contenido semántico o de diagramas detallado de los artefactos RUP asociados.

Para obtener información no específica de herramientas acerca de cómo definir, desarrollar y crear modelos del contenido de los artefactos RUP, consulte RUP.

Para obtener información acerca de las técnicas específicas de las herramientas para desarrollar el contenido de los modelos RS*x,* consulte

- La documentación del producto (guías de aprendizaje, ejemplos, ayuda en línea).
- Las guías de herramientas de las configuraciones de RUP de las que forma parte el documento.
- Recursos relacionados con RS*x* en developerWorks.

Convenios tipográficos

Las descripciones que pueden ser de interés para los usuarios de RS*x* que están realizando la migración desde IBM Rational Rose o XDE se presentan destacadas en un recuadro de texto con un borde de fondo gris:

> ***XDE/Rose***
> Descripción de interés para usuarios anteriores de XDE o Rose.

Organización del documento

Una sección de Conceptos básicos y terminología establece un vocabulario de trabajo y proporciona información general acerca de cómo se implementan los modelos en los productos RS*x*.

A continuación, en la sección Correlación de los modelos RUP con los modelos RS*x* describe cómo RS*x* da soporte a los tipos de modelos definidos por RUP.

Las siguientes son varias secciones que proporcionan directrices para estructurar modelos de varios tipos. Algunas de estas secciones describen los diferentes modos en que puede utilizar los modelos según el rigor que prefiera en relación con el control de la arquitectura, el método de creación de modelos y el proceso.

Por último, se describen los aspectos asociados al uso de modelos en equipos. Describe las estrategias para gestionar escalas y permitir que se puedan compartir modelos entre los miembros de los equipos para minimizar la contención de archivos y la fusión.

2. Terminología y conceptos básicos

Modelos

En RUP se define un modelo como “una especificación completa de un dominio de solución o problema desde una perspectiva concreta”. Un dominio de problema o un sistema puede especificarse mediante un número de modelos que representa diferentes perspectivas del dominio o del sistema. RUP propone un conjunto de modelos específico:

- Modelo de guión de uso empresarial
- Modelo de análisis empresarial
- Modelo de guión de uso
- Modelo de análisis (puede estar incluido en el modelo de diseño)
- Modelo de diseño
- Modelo de implementación
- Modelo de despliegue
- Modelo de datos

RUP es agnóstico en lo relacionado con las herramientas. Para RUP, un modelo puede ser un dibujo en una servilleta o en una pizarra, algo de una herramienta de creación de modelos o incluso una imagen mental. Desde la perspectiva RUP un modelo es un concepto lógico.

En el contexto de RS*x* se pueden describir modelos en términos lógicos pero también se pueden describir en términos físicos. Suponga que tiene equipos que trabajan en dos aplicaciones: un equipo de tres analistas que trabaja en una aplicación de gestión de horarios y un segundo equipo de cinco analistas que trabaja en una aplicación de un centro de atención telefónica. Actualmente los dos equipos trabajan para recopilar los requisitos y utilizan RS*x* para utilizar la creación de modelos de guiones de uso. En términos de RUP se puede decir que un equipo está creando “el modelo de guión de uso para la aplicación de horario de trabajo” y el otro está creando “el modelo de guión de uso para la aplicación de servicio de atención telefónica”. Pero si los equipos utilizan RS*x* es importante reconocer que los modelos tienen manifestaciones físicas. Este es el tema de la sección siguiente.

Archivos de creación de modelos

Los modelos RS*x* se mantienen en forma de archivos. (En la terminología de Eclipse un archivo se considera un recurso).[1] Por lo tanto, si encuentra el término "recurso de creación de modelos" en este documento o en otras fuentes, significa lo mismo que "archivo de creación de modelos". En el sentido más amplio RS*x* da soporte a dos clases de archivos de creación de modelos:

- **Los archivos de creación de modelos de preimplementación** se almacenan como archivos individuales en el sistema de archivos del sistema operativo del host. Tienen las extensiones de nombre de archivo “.emx”. Estos archivos contienen
    - Elementos de la semántica de UML (clases, actividades, relaciones, …)
    - Diagramas UML que muestran los elementos de la semántica UML y también *pueden* mostrar referencias visuales a cosas de otros dominios de semántica como, por ejemplo, Java, C++ o DDL.

[1] En Eclipse un recurso es un archivo pero también tiene propiedades adicionales y un comportamiento dentro del entorno de Eclipse. Los archivos de creación de modelos descritos aquí se tratan como recursos en Eclipse.

- **Los archivos de creación de modelos de implementaciones** se almacenan como proyectos de Eclipse *en un espacio* de trabajo de Eclipse. Los proyectos contienen
    - Artefactos de implementación (código fuente Java, páginas Web, archivos de metadatos basados en XML…)
    - Archivos de diagramas que muestran y reflejan directamente los artefactos de implementación.

  La semántica del modelo reside en los artefactos de implementación propiamente dichos. Por ejemplo, el modelo de semántica de una implementación Java se serializa y almacena como una colección de archivos de código fuente Java. Cada diagrama reside en su propio archivo físico dentro del proyecto. Los archivos de diagramas pueden tener diferentes extensiones. La más común es ".dnx". Los diagramas de creación de modelos de implementación suelen utilizar la notación UML pero también puede utilizar otra notación, por ejemplo, la notación IDEF1X o Information Engineering para la visualización de datos o la notación propiedad de IBM que se utiliza para diseñar las capas Web).

El objetivo documento es mostrar cómo se organizan las estructuras internas de los modelos de "preimplementación". **En el resto del documento el término "archivo de creación de modelos" se reserva para los archivos de creación de modelos de "preimplementación"** (los archivos con la extensión .emx). Puede encontrar las directrices para organizar el contenido de los proyectos de implementación en otras fuentes como, por ejemplo, la ayuda en línea para Rational Software Architect, Rational Application Developer y Rational Web Developer Community Edition.

Un archivo de creación de modelos no contiene necesariamente toda la información para un modelo (lógico). De hecho un archivo de creación de modelos generalmente contendrá únicamente un subconjunto de un modelo. En el ejemplo proporcionado anteriormente, había un equipo de tres trabajando en un modelo de guión de uso para una aplicación de horario. Dicho equipo puede optar por particionar físicamente su modelo de guión de uso en tres archivos de creación de modelos diferentes, de modo que cada miembro del equipo pueda trabajar en un subconjunto de guiones de uso diferentes sin tener que compartir el mismo archivo. La sección final de este documento describe los problemas asociados con la partición de modelos y la gestión de los archivos de creación de modelos.

Tipos de modelos

En RUP, los modelos tienen tipos específicos, por ejemplo, modelo de guión de uso, modelo de análisis, o modelo de datos. En RS*x*, los archivos de creación de modelos no tienen un tipo "definido fuertemente" sino que pueden seguir un convenio de uso de varios archivos de creación de modelos que tengan "un tipo definido ligeramente". Si desea seguir este convenio, puede establecer los tipos definidos ligeramente de cualquiera de los dos modos siguientes:

- Comience por un archivo de creación de modelos "en blanco" (consulte más adelante) y establezca su tipo según el nombre asignado y el tipo de contenido que colocará en el mismo (incluidos los perfiles UML que aplique).
- Cree un archivo de creación de modelos basado en un "modelo de plantilla" predefinido que represente un tipo de modelo determinado. Los productos RSx proporcionan un conjunto de modelos de plantilla para los tipos de modelos descritos en este documento. También puede crear sus propios modelos de plantilla (consulte la ayuda del producto, los foros y otros recursos en developerWorks).

De cualquier modo, el "tipo" de un archivo de creación de modelos en RSx es simplemente una cuestión de convenio relacionada con el nombre y el contenido del archivo. Por ejemplo, la herramienta no impedirá que un archivo que contiene un modelo de guión de uso contenga también las clases que llevan a cabo los guiones de uso (lo que en términos lógicos, RUP considera parte del modelo de diseño o análisis).

Estas directrices sugieren que trate los archivos de creación de modelos RSx como archivos con tipo.

Espacios de trabajo, proyectos y tipos de proyectos

Los lectores que estén familiarizados con los productos Eclipse, WebSphere Studio o Rational Application ya saben que los archivos residen en proyectos, que los proyectos pueden ser de diferentes tipos y que los proyectos se agrupan y gestionan en espacios de trabajo. Los archivos de creación de modelos RS*x* residen en los proyectos simplemente como otros archivos.

En este artículo, no se describirán detalladamente todos los tipos de proyectos disponibles en RSx y Rational Application Developer. Nuestro interés primordial está basado en las dos categorías de los proyectos:

- **Proyectos UML**
- **Proyectos de implementación**, que incluye los tipos de proyectos especializados como, por ejemplo, Proyecto empresarial, Proyecto EJB, Proyecto Web y Proyecto C++

Como se ha indicado anteriormente RS*x* da soporte a dos tipos de archivos de creación de modelos:

- Archivos con la extensión .emx que contienen modelos UML que se utilizan para la creación de modelos a niveles de abstracción por encima de la implementación (requisitos, análisis, diseño).
- Proyectos Eclipse que contienen la semántica de la implementación, generalmente serializados como elementos de archivos de código fuente, y diagramas que reflejan dicha semántica.

La norma para asignar modelos a los proyectos es sencilla:

a) Coloque los archivos de creación de modelos de “preimplementación” en proyectos UML.

b) Los modelos de implementación se ocupan de sí mismos porque esencialmente:

[modelo implementación] = [proyecto implementación]

Hay dos excepciones dentro de esta norma. Los siguientes archivos de creación de modelos UML son candidatos para su ubicación en un proyecto de implementación específico del lenguaje:

- "Modelos de sketchs" de diseño (que se describirán en una sección posterior).
- Modelos con diagramas de secuencia que describen las pruebas que se ejecutarán sobre el código en el proyecto

> ***XDE/Rose***
> Las teorías de funcionamiento en Rose y XDE incluyen como práctica refinar de modo iterativo el modelo de diseño hasta que se consigue un nivel de abstracción equivalente al código y luego se utiliza la tecnología de sincronización modelo-código para mantener la semántica de dicho modelo en concordancia con el propio código. Por lo tanto, en XDE, por ejemplo, existen modelos de implementación no simplemente como código y diagramas de proyectos sino también como archivos de "modelos de código" que persisten independientemente de los artefactos de implementación y en esencia representan una copia redundante de su semántica.
>
> La teoría de funcionamiento de RSx fomenta el uso de los modelos neutros en relación a las plataformas a niveles de abstracción superiores al nivel de código (por ejemplo, los modelos de diseño como los de un modelo de diseño de IT de empresa) y el uso de transformaciones para generar código a partir de dichos modelos. A continuación, en el nivel de código de la abstracción, RSA, RSD y RAD simplemente le permiten dibujar diagramas de semántica de código expresados en notación UML, sin tener en cuenta el método en que se utiliza un modelo semántico con persistencia diferente en el nivel de implementación de la abstracción.
>
> Tenga en cuenta que RSx no le *impide* definir modelos UML con una abstracción de nivel de código ni generar código a partir de los mismos ya que, de hecho, se espera este tipo de uso. Pero

Revisión de conceptos

Las ilustraciones siguientes resumen las descripciones anteriores. Las ilustraciones reflejan los escenarios descritos anteriormente, en los que existen dos equipos, uno que trabaja en una aplicación de servicio de atención telefónica y otro en una aplicación de gestión de horarios de trabajo.

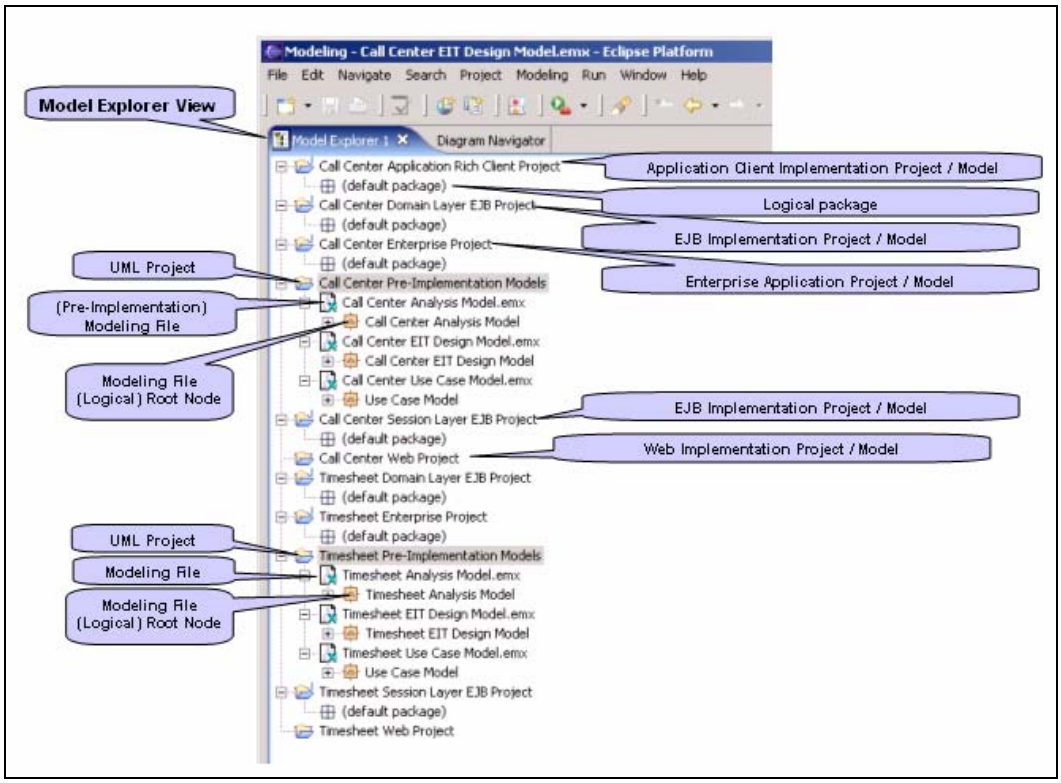


Figura 2-1

RSX proporciona una vista de explorador[2] que proporciona una vista física y lógica de los modelos. En el explorador, verá los proyectos del espacio de trabajo como nodos de nivel superior y dentro de cada proyecto verá los recursos que pertenecen a dicho proyecto. Por lo tanto, en la **Figura 2-1** vemos en el Explorador de modelos una colección de proyectos que corresponde a dos aplicaciones de nuestro escenario. Se observa que los proyectos UML se han utilizado para los modelos de preimplementación y se ha utilizado una colección de proyectos de tipos apropiados a las soluciones para los modelos de implementación.

> ***XDE/Rose***
> A diferencia del explorador de modelos de RSA, los exploradores de modelos de Rose y XDE sólo proporcionan una vista lógica de los modelos. Tenga en cuenta que la vista de los recursos proporcionados por el explorador de modelos RSA no es una vista física "pura" proporcionada por la vista del navegador de Eclipse. Aunque pueden verse algunos recursos físicos en el Explorador de modelos, principalmente están representados por iconos que indican vistas *lógicas* de los recursos.

[2] En las versiones 6.x, esta vista era el “Explorador de modelos”. A partir de la versión 7.0, los modelos aparecen en el explorador de finalidad general. La ilustración de este documento está basada en la versión 6.0 y muestra el “Explorador de modelos”.

En la Figura 1-2, se muestra cómo el modelo de guión de uso de Timesheet puede organizarse internamente en paquetes que representan subconjuntos del dominio del problema cuyas funciones es posible agrupar.

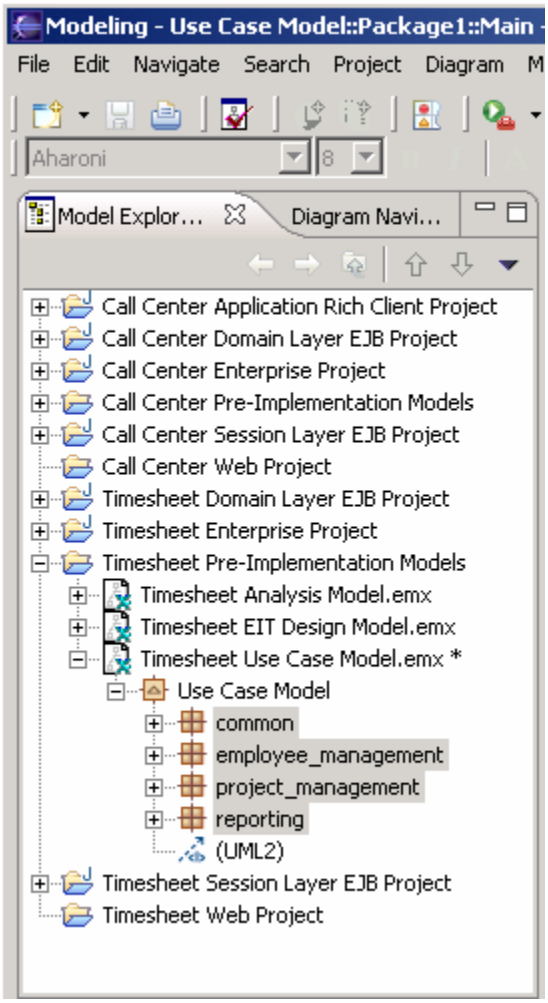


Figura 2-2

En la **Figura 2-1 y 1-2** cada modelo de preimplementación reside en un solo archivo de creación de modelos. En la Figura 1-3, se muestra cómo en el modelo de guión de uso Timesheet se puede descomponer en varios archivos de creación de modelos que se corresponden con los mismos subconjuntos del dominio del problema. Observe cómo la raíz de cada archivo de creación de modelos tiene asignado un nombre para mantener un espacio de nombres coherente entre todos los archivos de creación de modelos de que consta el modelo de guión de uso completo.

3. Correlación de los modelos RUP con los modelos RS*x*

La tabla siguiente muestra cómo los modelos RUP utilizados más frecuentemente se pueden correlacionar, por convenio, con los tipos de archivos de creación de modelos“RSx”. Generalmente la correlación es directa, pero resulta clave para utilizar este documento como guía para realizar prácticas de RUP con RSx. En la tabla siguiente se describen los tipos de archivos de creación de modelos RSx que se tratan a continuación. Las directrices para la organización interna de los diferentes tipos de archivos de creación de modelos y de los tipos de proyectos en que se han de incluir se proporcionan en las secciones posteriores. Estas descripciones posteriores se presentan en relación con los tipos de archivos de creación de modelos RSx que se listan aquí.

Modelo RUP	***Tipo de archivo de creación de modelos RSx***
Modelo de guión de uso	Archivo de creación de modelos basado en la plantilla de “Modelo de guión de uso” (alternativamente: comience por el archivo de creación de modelos en blanco, limite el contenido en base a la guía del modelo de guión de uso RUP)
Modelo de análisis	Modelo de análisis (alternativamente: comience por un archivo de creación de modelos en blanco, limite el contenido en base a la guía del modelo de análisis RUP) (alternativamente, utilice paquetes de «análisis» del Modelo de diseño)
Modelo de diseño	Para las aplicaciones de n niveles : el archivo de creación de modelos basado en la plantilla de “Modelo de diseño de IT de empresa” (alternativamente: comience por un archivo de creación de modelos en blanco, limite el contenido en base a la guía del modelo de diseño RUP) Para otros tipos de aplicaciones: comience por el archivo de creación de modelos y limite el contenido en base a la guía del modelo de diseño RUP Para el diseño de "sketch": archivo de creación de modelos en blanco Suplemento opcional: archivo de creación de modelos en blanco utilizado como modelo de vista general de implementación
Modelo de implementación	Los proyectos Eclipse que contienen artefactos de implementación y los archivos de diagrama
Modelo de despliegue	Comience por el archivo de creación de modelos en blanco, limite el contenido en base a la guía del modelo de despliegue RUP

Tipos de *archivos de creación de modelos RSx*

Archivo de creación de modelos en blanco

RSx proporciona la opción de crear un “Modelo en blanco” (Archivo →Nuevo→Modelo UML →Modelo en blanco). Un “Modelo en blanco”es un archivo de creación de modelos que no está basado en una plantilla de modelo. No tiene aplicados perfiles especiales y ningún contenido predeterminado que no sea un diagrama “Principal” (formato libre). **Puede utilizar un archivo de creación de modelos como punto de partida para cualquier tipo de modelo**. Para seleccionar el nombre, el contenido que desea definir y los perfiles que desea aplicar, puede utilizar un archivo de creación de modelos en blanco, para crear un modelo de guión de uso, un modelo de análisis, un modelo de diseño, un modelo de despliegue o cualquier otro tipo de modelo RUP.

Archivo de creación de modelos de guión de uso

RSx permite crear un archivo de “Modelo de guión de uso” basado en una plantilla de modelo. La plantilla añade contenido predeterminado como se muestra en la ***Figura 3-1***. Queda excluido del ámbito de este documento describir cómo se utilizan las series de búsqueda y el contenido de los “bloques de creación”. Las plantillas contienen instrucciones que en su mayor parte son autodescriptivas.

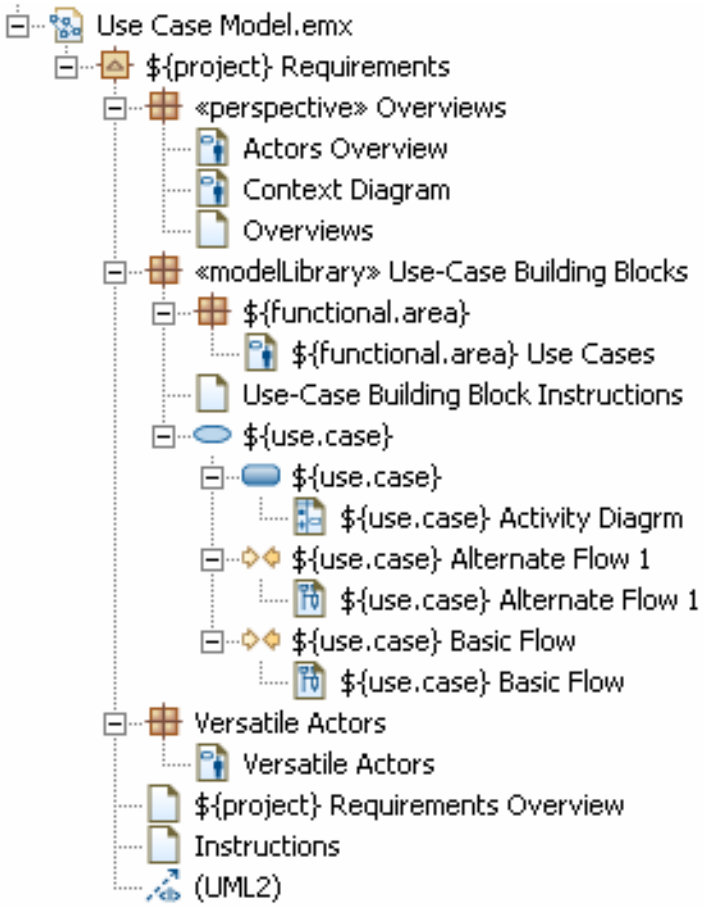


Figura 3-1

Archivo de creación de modelos de análisis

RSx proporciona la opción de crear un archivo de “Modelo de análisis” basado en una plantilla de modelo. Las plantillas añaden contenido predeterminado como se muestra en la ***Figura 3-2***. Asimismo se aplica un perfil de “Análisis” a los archivos de modelos creados a partir de esta plantilla:

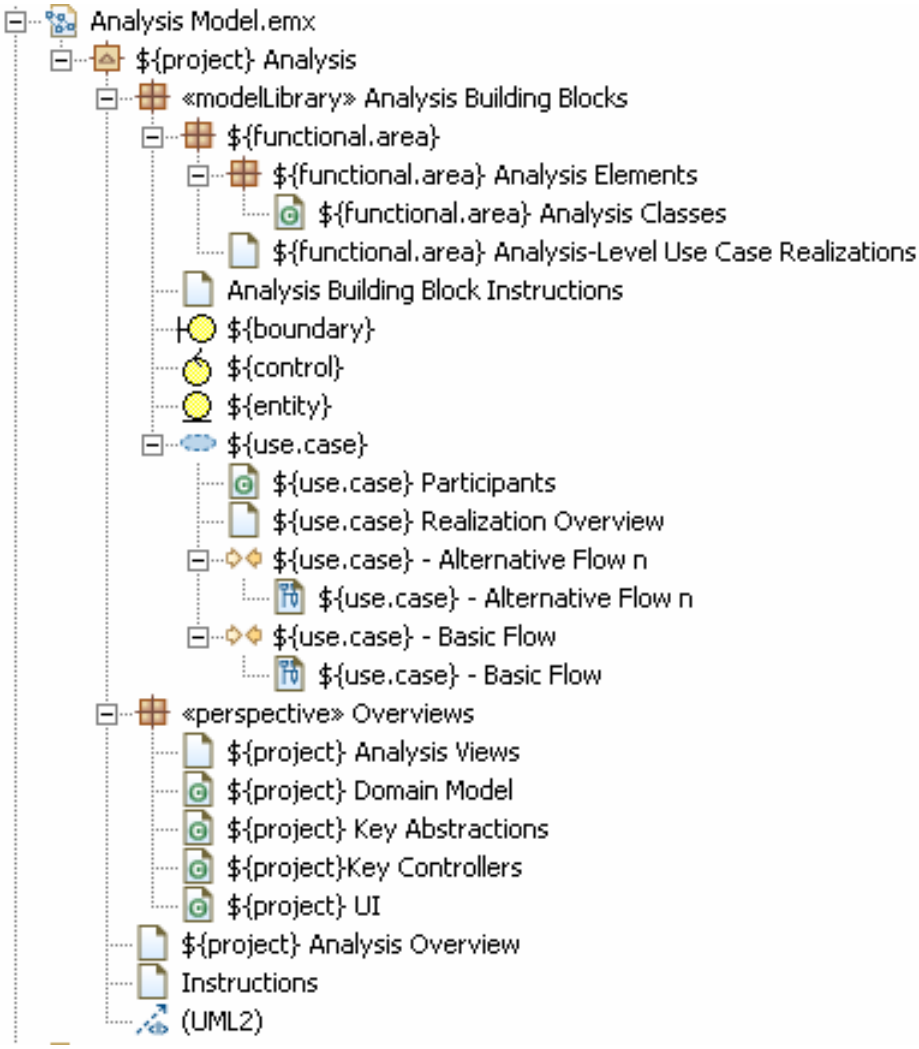


Figura 3-2

Archivo de creación de modelos de diseño de TI de empresa

RSx proporciona la opción de crear un archivo de “Modelo de diseño de TI de empresa” (EITDM) basado en una plantilla de modelo. La plantilla proporciona el contenido predeterminado como se muestra en la **Figura 3-3**. Asimismo se aplicará un perfil de “Transformación de EJB” [3] a los archivos de creación de modelos creados a partir de esta plantilla. Esta es la plantilla adecuada para diseño (y opcionalmente para análisis) cuando se trata de aplicaciones de empresa y de utilizar las transformaciones que generan código *RSX* para dar soporte a la creación de este tipo de aplicaciones.

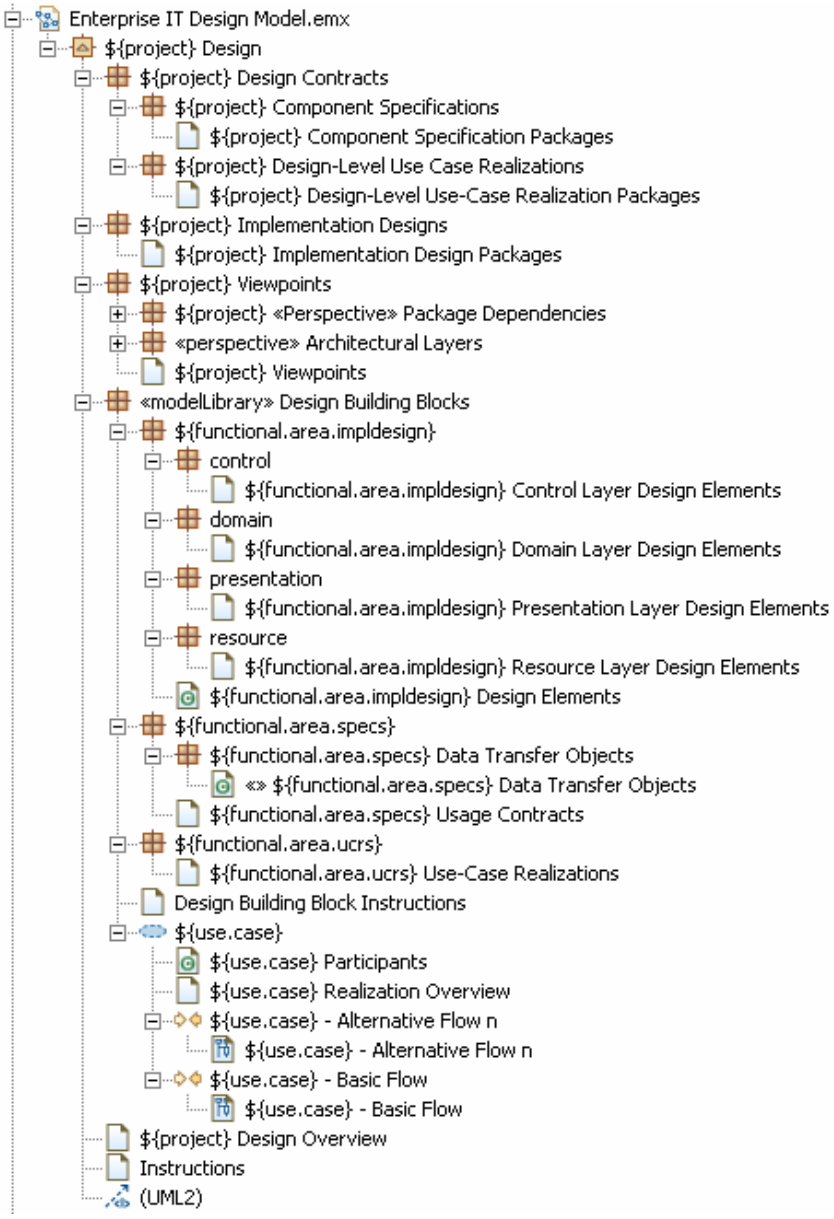


Figura 3-3

[3] Probablemente, el conjunto de perfiles que habilitan la transformación y que se suministra como parte de la plantilla del modelo de diseño de TI de empresa irá evolucionando a medida que se creen actualizaciones del producto.

Archivo de creación de modelos de visión general de la implementación

Asimismo, como parte del modelo de diseño, es posible que le resulte útil definir un “Modelo de visión general de la implementación” que capture una vista de alto nivel sobre cómo se ha de organizar la implementación. El “Modelo de visión general de la implementación” se utiliza en las primeras fases de diseño – antes de generar o escribir el código para representar los proyectos y carpetas o paquetes reales de RSx o Rational Application Developer en los que espera que resida el código y los archivos relacionados (metadatos, descriptores de despliegue, etc.). También puede utilizar este modelo para mostrar las dependencias previstas entre dichos proyectos y paquetes, lo cual puede resultarle útil para identificar los requisitos de compilación del sistema. El modelo de visión general de la implementación también puede ser un lugar que contenga los diagramas conceptuales e informales de la arquitectura de la solución.

Modelo de implementación

Como se ha indicado anteriormente, en RS*x* un modelo de implementación consta de un proyecto que contiene artefactos de implementación y opcionalmente diagramas que muestran dichos elementos [4]

“Modelos de sketch”

Como se ha indicado en la sección “Conceptos básicos y terminología”, puede tratar los modelos de diseño como planos arquitectónicos que se mantienen durante la vida útil del sistema y se utilizan para dar soporte y aplicar el control arquitectónico. O puede tratarlos como sketches que sugieren un diseño y le ayudan a clarificarlo y comunicarlo, pero que pueden desecharse una vez iniciada la implementación. RSx da soporte a ambos métodos. Sus características generalmente no van destinadas a un método u otro, pero las opciones acerca de cómo utilizar los modelos de diseño le ayuda, sin lugar a dudas, a determinar qué características de RS*x* utilizará y cómo las utilizará. Donde resulte importante la distinción en el contexto de las directrices presentadas en este documento, se utilizará el término “modelo de sketch” para indicar que se está utilizando un modelo de un modo más "desechable".

[4] Para crear estos diagramas, en lugar de utilizar Archivo→Nuevo→Modelo UML para crear un modelo, utilice Archivo→Nuevo→Diagrama de clase para crear un diagrama en el que puede crear "vistas" de código en notación UML (o en otra). Cada diagrama individual se mantiene como un archivo individual cuya extensión es .dnx, y cuya versión se puede controlar prácticamente del mismo modo que el archivo de código. Estos diagramas no contienen información de semántica, simplemente notación. Toda la información semántica reside en el código propiamente dicho. Cuando cambie alguna cosa como, por ejemplo, el nombre de clase o la firma de operación en uno de estos diagramas, realmente, está cambiando el código subyacente. Cuando realiza este tipo de cambios en el código (mediante un editor de texto) los diagramas en los que se muestra el código modificado se actualizan automáticamente.

4. Directrices generales y técnicas para organizar las estructuras internas de los modelos

La herramienta principal para organizar el contenido de los modelos UML es el paquete. Los paquetes UML tienen dos finalidades principales:

- Partición, organización y etiquetado de la información de los modelos.
    - Agrupación de elementos que corresponden a un tema específico del dominio del problema o de la solución.
    - Separación de tipos de información de modelos diferentes como, por ejemplo, interfaces, implementaciones, diagramas, etc.
    - Agrupación de elementos para definir y controlar las dependencias de otros elementos.
    - Agrupación de diagramas que proporcionan vistas alternativas del mismo modelo.
- Establecimiento de espacios de nombres.
    - Para elementos del modelo.
    - Para artefactos de implementación generados a partir de elementos de modelos (esto puede requerir correlaciones entre los espacios de nombres del lenguaje de implementación y el modelo).
    - Para una unidad de reutilización.

Tradicionalmente, RUP ha propuesto estrategias de empaquetado específicas para diferentes tipos de modelos. Estas estrategias se reflejan en las secciones específicas del tipo de modelo de este documento. RSx también introduce algunas herramientas organizativas adicionales que se describen aquí:

Representación de puntos de vista mediante paquetes de «perspectivas»

En algunos casos en los que resulte más adecuado ver los elementos organizados de más de un modo puede crear paquetes adicionales con diagramas que muestren los esquemas organizativos alternativos. Esta misma técnica puede servir siempre que haya una necesidad de representar una vista determinada del contenido del modelo que entre dentro del esquema de empaquetado del modelo. RSx da soporte a esta técnica proporcionando un estereotipo de paquete de «perspectiva» como parte del "perfil base" de UML. Puede considerar un paquete de «perspectiva» como el equivalente de un RUP para el “Punto de vista” de sistemas de ingeniería o IEEE 1471- 2000.

No coloque elementos semánticos (clases, paquetes, asociaciones, etc.) en los paquetes de «perspectivas». Simplemente coloque en su interior diagramas que muestren las vistas basadas en el punto de vista de la aplicación o en el punto de vista organizativo alternativo. Al aplicar el estereotipo de «perspectiva» a un paquete ocurren varias cosas. Visualmente se identifica que el paquete representa un punto de vista determinado. También da soporte a una norma de validación de modelos que le avisa cuando se colocan elementos semánticos en un paquete de «perspectiva». También sirve como una designación de los paquetes que deben ser ignorados por las transformaciones RSx.

Creación de descripciones de actualización automática de temas específicos relacionados con el uso de diagramas de temas

A diferencia de los diagramas "normales" en los que se colocan manualmente los elementos que se desean mostrar, el contenido de un diagrama de tema queda determinado por una consulta que se ejecuta sobre el contenido de un modelo existente. Para crear un diagrama de tema seleccione un elemento de modelo de "tema", a continuación defina los otros elementos que desea que aparezcan en el diagrama basados en los tipos de relaciones que tienen para el elemento de tema. A medida que el contenido semántico del modelo cambie, los diagramas de temas se ajustan en consecuencia.

Examen de modelos mediante los diagramas de exploración

Los diagramas de exploración no son *específicamente* una herramienta para la organización de modelos. Su finalidad es facilitar el descubrimiento y comprensión del contenido del modelo sin tener que componer manualmente los diagramas. Pero en el contexto de la organización de modelos se recomienda tenerlos en cuenta ya que pueden disminuir la necesidad de componer los diagramas a guardar. Lo que a su vez, puede disminuir el tamaño y la complejidad de los modelos, con lo cual resultan más fáciles de organizar.

Los diagramas de exploración se parecen un poco a los diagramas de temas, pero la diferencia clave es que los diagramas de exploración no tienen persistencia ya que siempre se generan en el momento. Para generar un diagrama de exploración, seleccione un elemento de modelo (de un diagrama o del explorador de modelos) y utilice el menú de contexto para “Explorar en el diagrama de exploración”. Esto generará un diagrama que muestra el elemento seleccionado como el "punto focal" con elementos relacionados presentados en un diseño radial alrededor del punto focal. Por supuesto, puede seleccionar uno de los elementos relacionados en dicho diagrama de exploración y convertirlo en el punto focal de otro diagrama de exploración y continuar de esta forma durante el tiempo que desee.

Navegación entre diagramas

En RSx hay dos mecanismos para navegar entre diagramas:

- Se puede arrastrar un nodo de diagrama del explorador de modelos a otros diagrama de contenedor. A continuación, puede pulsar dos veces el icono resultante del diagrama de contenedor para abrir el diagrama referenciado.
- Siempre que crea un paquete UML nuevo en un modelo, se crea automáticamente un diagrama “Principal” (un diagrama de formato libre). De forma predeterminada, este diagrama “Principal” se crea como el diagrama predeterminado del paquete. Puede cambiar el nombre del diagrama por otro que no sea “Principal” y continuará tratándose como el diagrama predeterminado. También puede seleccionar un diagrama diferente del paquete y convertirlo en el diagrama "predeterminado" de dicho paquete. La finalidad del diagrama predeterminado es esta: si coloca el paquete en algún otro diagrama de contenedor puede pulsar dos veces el paquete y se abrirá su diagrama predeterminado.

Estos mecanismos dan soporte a las **directrices** de organización siguientes, que se pueden aplicar a los modelos de cualquier tipo:

1. Composición del diagrama Principal (o de cualquier otro diagrama predeterminado) de cada archivo de creación de modelos que desee mostrar
    a. Cada paquete de nivel superior del archivo de creación de modelos.
    b. Los iconos de diagramas de cualquier otro diagrama que residen en el paquete raíz del archivo de creación de modelos (en otras palabras no mostrar el icono para el diagrama predeterminado propiamente dicho).
2. Composición del diagrama Principal (o de cualquier otro diagrama predeterminado) para cada paquete de nivel superior que desee mostrar
    a. Los paquetes que contiene directamente.
    b. Los iconos de diagramas de otros diagramas que contiene directamente.
3. Repetición de este patrón para cada nivel de paquete inferior en orden sucesivo

5. Directrices para la organización interna del modelo de guión de uso

Organización de nivel superior del modelo de guión de uso

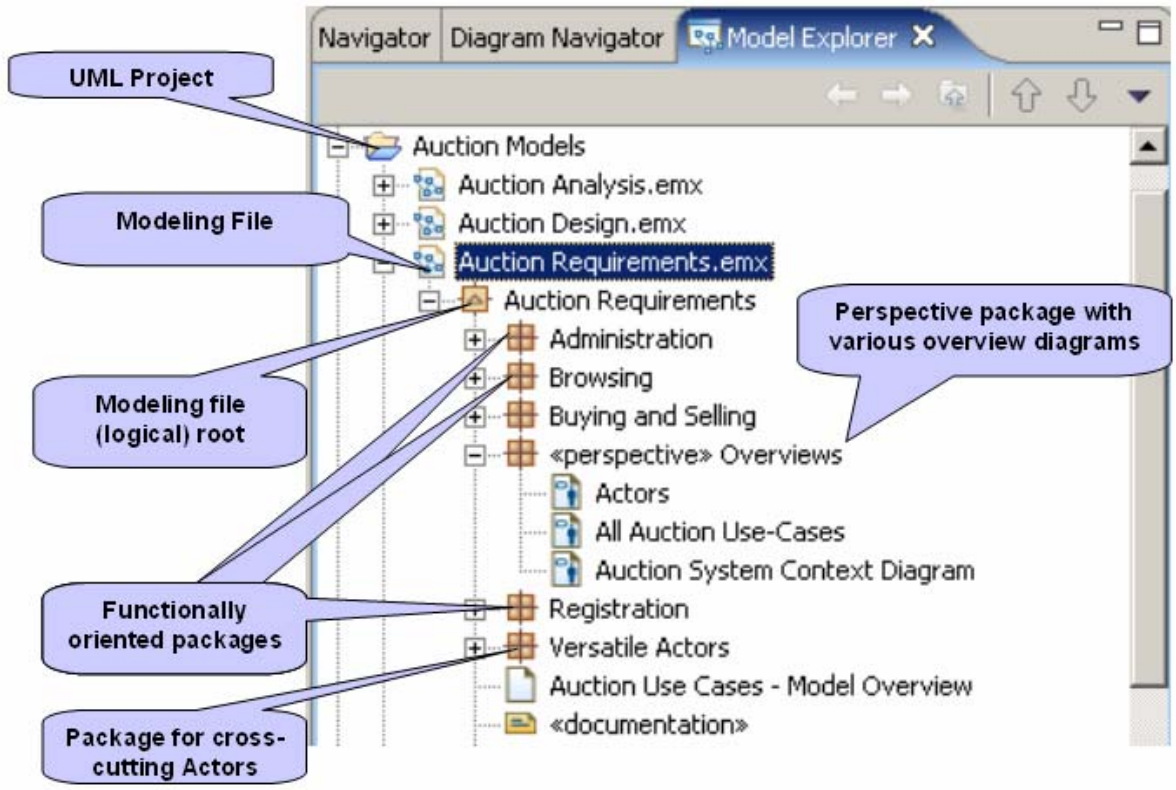


Figura 5-1

La **Figura 5-1** muestra las directrices siguientes para estructurar los modelos de guión de uso:

1. Utilice los paquetes de nivel superior para establecer agrupaciones orientadas funcionalmente:

    - Generalmente, esto se adecua bien a los aspectos de división de trabajo cuando un equipo de personas trabajan en el modelo de guión de uso. Y deja las bases sentadas por si posteriormente decide dividir el modelo de guión de uso en varios archivos de creación de modelos debido a que la compartición de archivos se ha convertido en un problema (simplemente puede crear un archivo de creación de modelos diferente por paquete de nivel superior).

    - Comparado con otros métodos organizativos, generalmente se correlacionará mejor con la organización de la implementación eventual. Esto es importante si utiliza las transformación para sembrar cada nivel inferior de abstracción sucesivo. Específicamente, si va a generar contenido con el que sembrar en un modelo de análisis basado en el modelo de guión, es recomendable que la estructura del paquete del modelo de guión de uso se correlacione bien con la estructura del paquete del modelo de análisis de destino. Y a su vez, deseará que la estructura del paquete del modelo de análisis se correlacione bien con el modelo de diseño y

la estructura del paquete del modelo de diseño se correlacione bien con el conjunto de proyectos de que consta la implementación. Cuanto más sencillas sean estas correlaciones, menor será el trabajo necesario para configurar las transformaciones de un nivel de abstracción al siguiente.

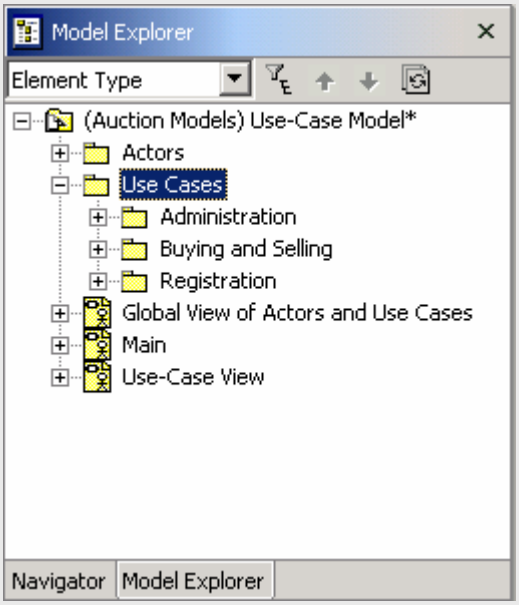
2. Utilice otro paquete de nivel superior para capturar actores "versátiles" o de "gran espectro".

3. Utilice los diagramas de los paquetes de "perspectiva" para capturar vistas cruzadas de alto nivel de la lógica de los guiones de uso:

    o Proporcionar vistas cruzadas y vistas de guiones de uso importantes para la arquitectura manteniendo los elementos semánticos del modelo organizado en agrupaciones orientadas a funciones.

XDE/Rose

Las directrices RSX revisan de algún modo las directrices tradicionales para la organización de alto nivel del modelo de guión de uso, que era crear un paquete para los actores y otro para los guiones de uso. Después, a medida que el tamaño y la complejidad del modelo lo exigían, se utilizaban los paquetes de nivel inferior para establecer agrupaciones orientadas funcionalmente, como se muestra en el ejemplo basado en XDE:



Contenido del modelo de guión de uso

Queda fuera del ámbito de este documento ser una guía de aprendizaje detallada acerca de cómo escribir buenos guiones de uso o las ventajas o desventajas de la creación de modelos de guiones de uso. No obstante, la siguiente es una breve descripción de lo que se incluye en un modelo de guión de uso, además de los actores y los guiones de uso.

- **Recomendación:** cree un diagrama "principal" en la raíz del modelo, que describa los demás paquetes del modelo y permita desplegar dichos paquetes y sus respectivos diagramas "principales".

- **Recomendación:** en cada paquete de guión de uso, incluya un diagrama que muestre los guiones de uso del paquete, cualquier relación entre los mismos y los actores que participan en ellos. Si el número de guiones es muy elevado, lo apropiado será más de un diagrama.

- **Recomendación:** describa el flujo principal y alternativo de cada guión de uso en su campo Documentación[5] (consulte la **Figura 5-2**).

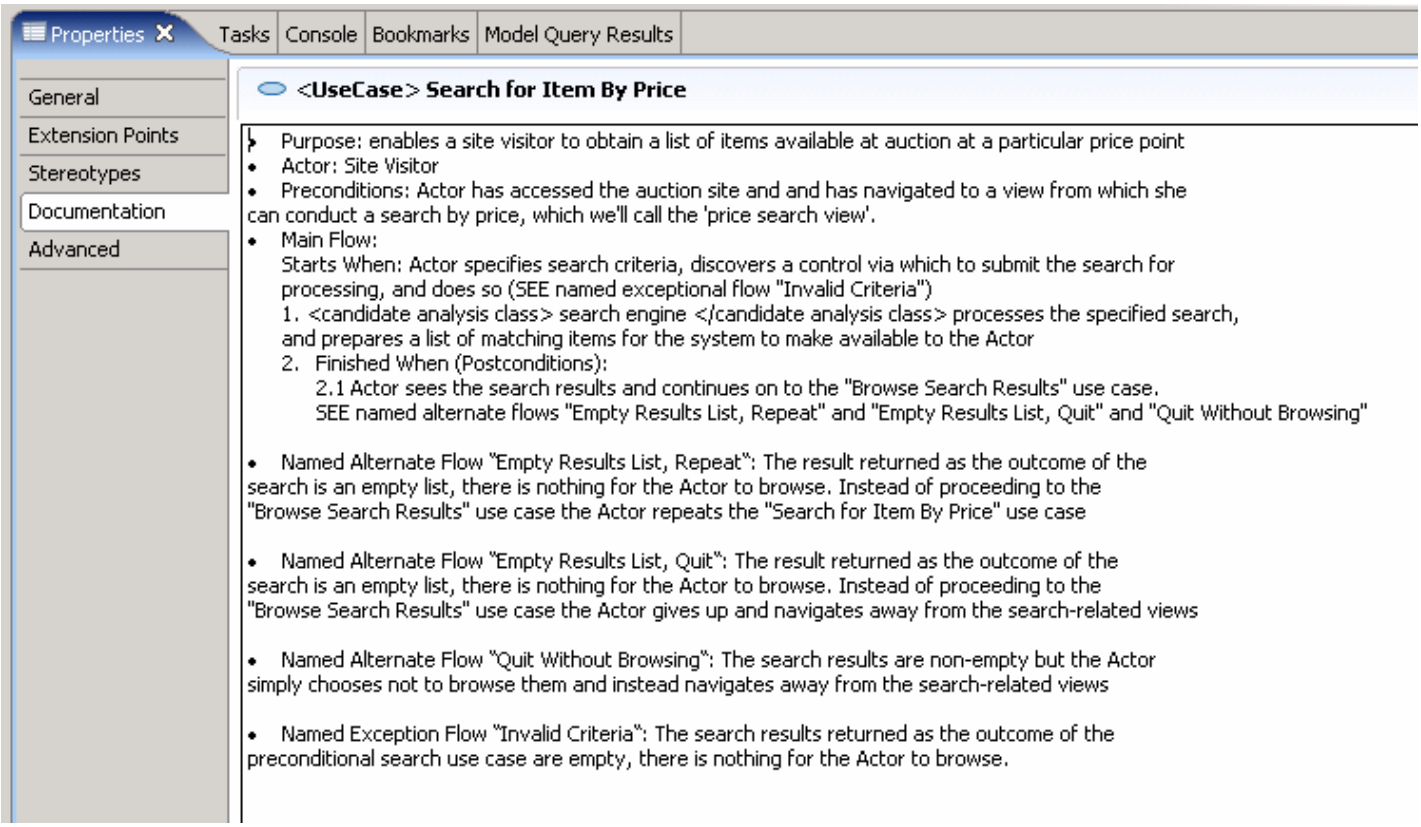


Figura 5-2

- **Opcional:** cuando la complejidad de un guión de uso lo requiera, añada un diagrama de actividad y compóngalo de modo que refleje el flujo de actividad general del guión de uso (consulte la **Figura 5-3**). **Lógica:** ayuda a mostrar las condiciones que corresponden a cada uno de los flujos (principal y alternativo/excepcional) y ayuda a garantizar que todos diferentes flujos vuelvan a converger finalmente. Si se añade un diagrama de actividad en RS*x* se añadirá automáticamente una actividad al guión de uso, con el diagrama debajo de la actividad).

- **Opcional:** cree un modelo de realización de tipo "caja negra" de cada uno de los flujos nombrados (principal, alternativo y excepcional) del guión de uso: añada una aparición de colaboración al guión de uso; añádalo a una instancia de interacción correspondiente al flujo principal del guión de uso más una instancia de interacción a cada uno de los flujos nombrados alternativo y excepcional; componga un diagrama de secuencias (o, de forma alternativa, un diagrama de comunicación) para cada instancia de interacción. Estas instancias de colaboración de guión de uso no se deben confundir con las realizaciones de guiones de uso de nivel de análisis (como se describe en el Modelo de análisis) o con las realizaciones de guiones de uso de nivel de diseño (como se describe

[5] El formato descrito en el ejemplo de guión de uso se ha obtenido creando la plantilla textual para una descripción del guión de uso utilizando un editor compatible con RTF y luego copiando y pegando la plantilla en el campo de descripción del guión de uso.

en el Modelo de diseño). Estas son realizaciones de guiones de uso de tipo “caja blanca” y describen las interacciones entre los elementos internos de una solución. Las apariciones de colaboración propuestas aquí para el Modelo de guión de uso son estrictamente interacciones de tipo “caja negra” entre los actores y el sistema. (Consulte la **Figura 5-3**). **Lógica:** esto proporciona a los usuarios que no son técnicos una imagen de alto nivel de cómo los usuarios del sistema interactuarán con el sistema. También puede ayudarle a identificar las diferentes vistas (pantallas o páginas) que serán necesarias como parte de la implementación. También establece de modo formal la denominación de los diferentes flujos del guión de uso (escenarios) asignando estos nombres a elementos semánticos del modelo (por ejemplo, a las apariciones de colaboración).

> ***XDE/Rose***
> En UML 1.x debe utilizar la “instancia de colaboración”, en lugar de la “aparición de colaboración” para este fin.

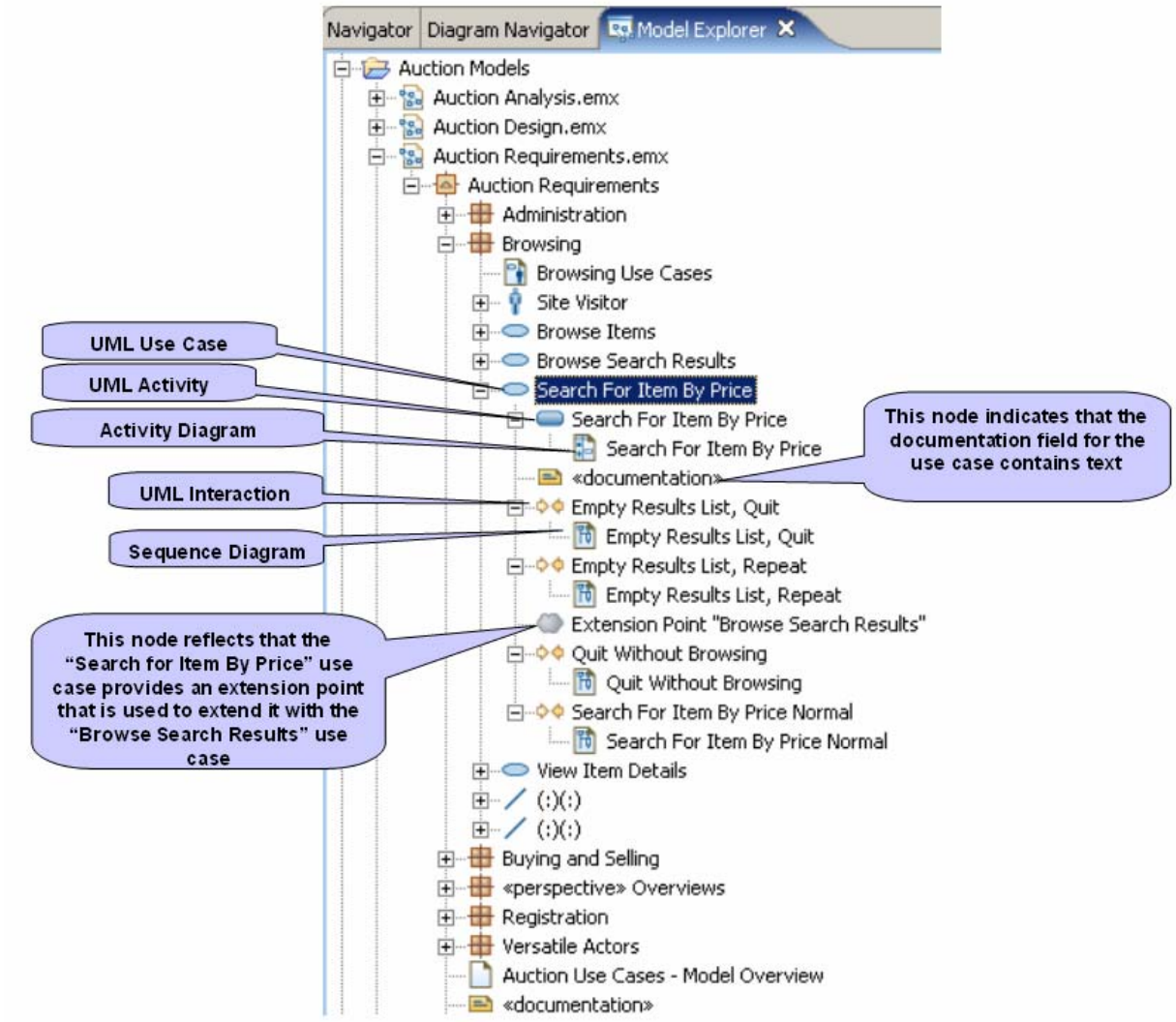


Figura 5-3

- **Opcional:** si está siguiendo las directrices de RUP para identificar las vistas importantes de la ‘arquitectura” y, en especial si va a mantener un documento de la arquitectura de software, añada un

paquete de «perspectiva» de nivel superior que contenga los diagramas de guiones de uso que muestren los guiones de uso importantes a nivel de arquitectura. Es posible que desee asignar al paquete el nombre “Vista de guiones de uso de la arquitectura”.

6. Directrices para la organización interna del Modelo de análisis

El Modelo de análisis representa un primer boceto de una solución. Es el primer paso desde la obtención de los requisitos hasta el diseño final, dedicado a la captura de información acerca del dominio del negocio y a mostrar elementos de solución posibles en un nivel de abstracción superior que se acerque al negocio. Es donde residen las clases de análisis y las realizaciones de los guiones de uso de nivel de análisis. Es a través del proceso de las realizaciones de guiones de uso de creación de modelos (primordialmente los diagramas de secuencias) que se comienzan a *descubrir* las clases necesarias para la solución – y estas serán, en especial, las clases correspondientes a las líneas básicas que descubra que necesita en los diagramas de secuencias. Hay algunas normas generales que se pueden aplicar para sugerir contenido del modelo de análisis basado en el contenido del modelo de guión de uso. Estas se tratarán posteriormente en esta sección.

En RUP, la decisión de si se ha de mantener o no un Modelo de análisis independientemente del Modelo de diseño es una decisión específica del proyecto, una decisión basada en si cree que mantener el modelo de análisis por separado garantizará el tiempo invertido. Si se crea un Modelo de análisis diferente**,** pero no se mantiene, entonces las clases de análisis se pasarán al Modelo de diseño y se refinarán**.** O es posible que el modelo de análisis evolucione gradualmente para convertirse en el modelo de diseño [6]. En términos específicos del producto, estas son algunas de las opciones que debe tener en cuenta:

1. Cree un modelo de análisis que resida en el archivo de creación de modelos (o en un conjunto de archivos) basado en la plantilla del modelo de análisis. A continuación, utilice un proceso manual o las transformaciones automatizadas, para crear versiones refinadas de los elementos del análisis en un segundo archivo de modelo (o conjunto de archivos) basado en la plantilla de modelo de diseño de IT de empresa y luego deseche los archivos de creación de modelos de análisis. Esto le deja la opción de mantener la continuidad del modelo de análisis separado o de descartarlo.
2. Efectúe la creación de modelos a nivel de análisis en un archivo de creación de modelos (o conjunto de archivos) basándose en la plantilla de modelo de diseño de IT de empresa, a la que aplica el perfil de análisis. De este modo, puede iniciar la creación de modelos de realizaciones de guiones de uso utilizando las clases de análisis, a continuación, con el tiempo puede refinarlas de modo que las interfaces de diseños adopten los roles de los comportamientos.
3. Un híbrido de la segunda y tercera opción es mantener un modelo de análisis de clasificaciones, dentro del o de los mismos archivos de creación de modelos que el modelo de diseño. Para hacerlo debe segregar el contenido del análisis en paquetes a los que aplicará la palabra clave «analysis». Esto le brinda la oportunidad de retener elementos de nivel de análisis dentro de los mismos archivos de creación de modelos que sus homólogos de nivel de diseño más refinados.

Un tema que se debe tener en cuenta cuando se utilizan las transformaciones RSx para generar implementaciones es que, en muchos casos, dichas transformaciones pueden aceptar elementos de

[6] De hecho, RUP invoca la opción de crear clases de análisis y serializaciones de guiones de uso de nivel de análisis en el modelo de diseño que posteriormente evolucionan directamente y desde ese punto a sus formatos de diseño. Bajo este método, a medida que se “detecta” el modelo de diseño puede ir creando paquetes que conserven algunas de las perspectivas de “análisis puro”.

nivel de análisis como entrada, lo que evita realizar algunos de los pasos para refinar manualmente estos elementos en elementos de diseño. Cuando se utiliza RSx de este modo, es más recomendable utilizar la segunda o tercera opción mencionada anteriormente. Las transformaciones que generan código estándar y que se empaquetan como parte de RSx ignoran los paquetes de modelos que tienen la palabra clave «analysis».

Organización de alto nivel del modelo de análisis

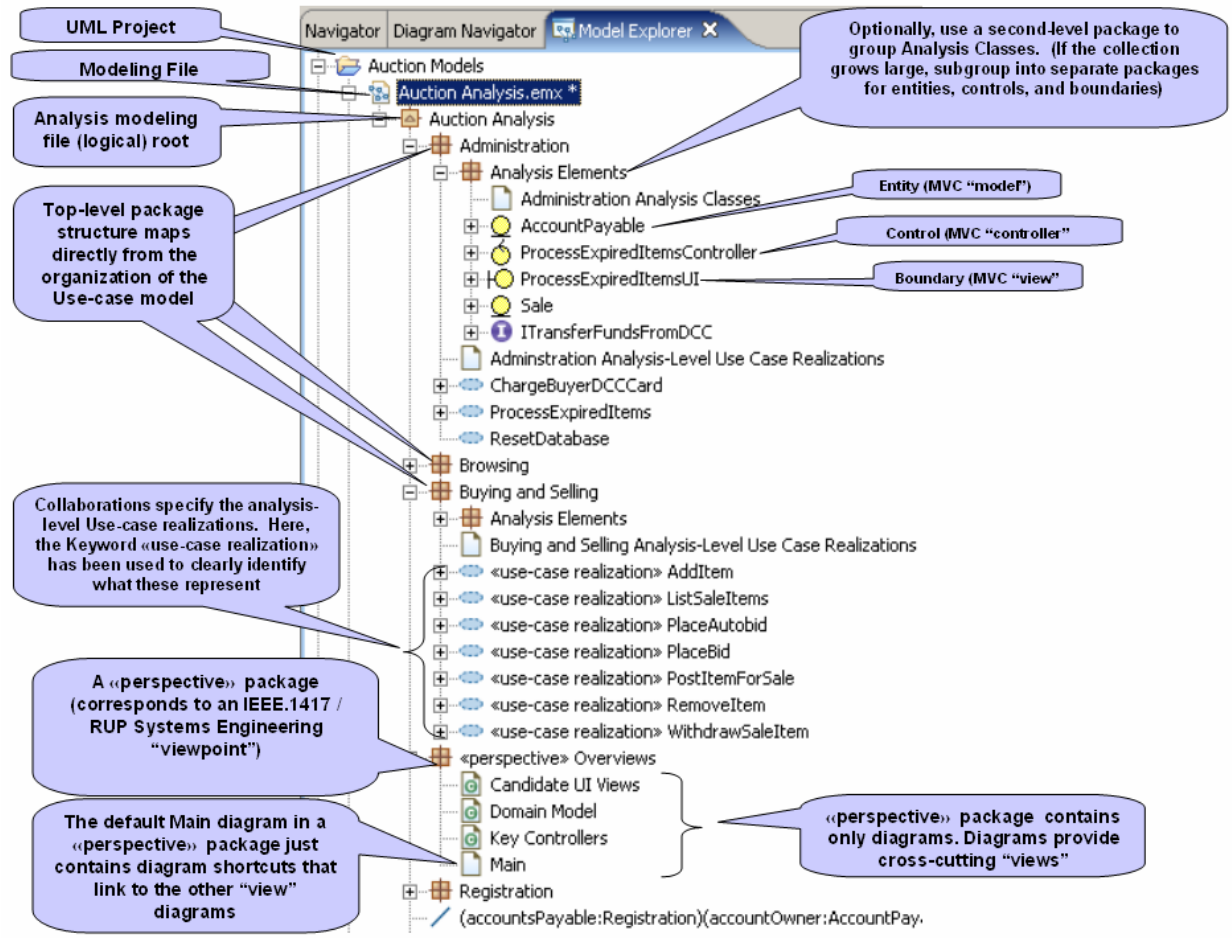


Figura 6-1

La ***Figura 6-1*** ilustra las siguientes directrices para estructurar los modelos de análisis:

1. Utilice los paquetes de nivel superior para establecer agrupaciones orientadas funcionalmente para las clases de análisis: Lógica: la misma lógica que se utiliza para el modelo de guión de uso.

2. Opcionalmente, en los paquetes de nivel superior utilice subpaquetes para recopilar y organizar las clases de análisis.

3. Utilice diagramas en los paquetes de "perspectivas" para capturar vistas alternativas de alto nivel o cruzadas de los elementos de análisis. Lógica: proporcionar diferentes perspectivas a los diferentes interesados manteniendo los elementos semánticos del modelo organizados en agrupaciones orientadas funcionalmente.

Puede encontrar una descripción de una ligera variación de este método en la **Figura 6-2** donde se puede ver el uso de un paquete de alto nivel para segregar las realizaciones de guiones de uso a partir de las clases de análisis. En dicho paquete de nivel superior se observa un conjunto de subpaquetes orientados funcionalmente que coincide con el conjunto de paquetes de nivel superior. Al aislar las realizaciones de guiones de uso de este modo se puede volver a modificar la estructura del paquete que contiene las clases de análisis, sin que ello afecte *necesariamente* la organización de las realizaciones de los guiones de uso. (En especial, si el modelo de análisis va a evolucionar al diseño *in situ*, es probable que la organización de paquetes para las clases evolucione de tal modo que ya no coincida con el uso original de los guiones de uso).

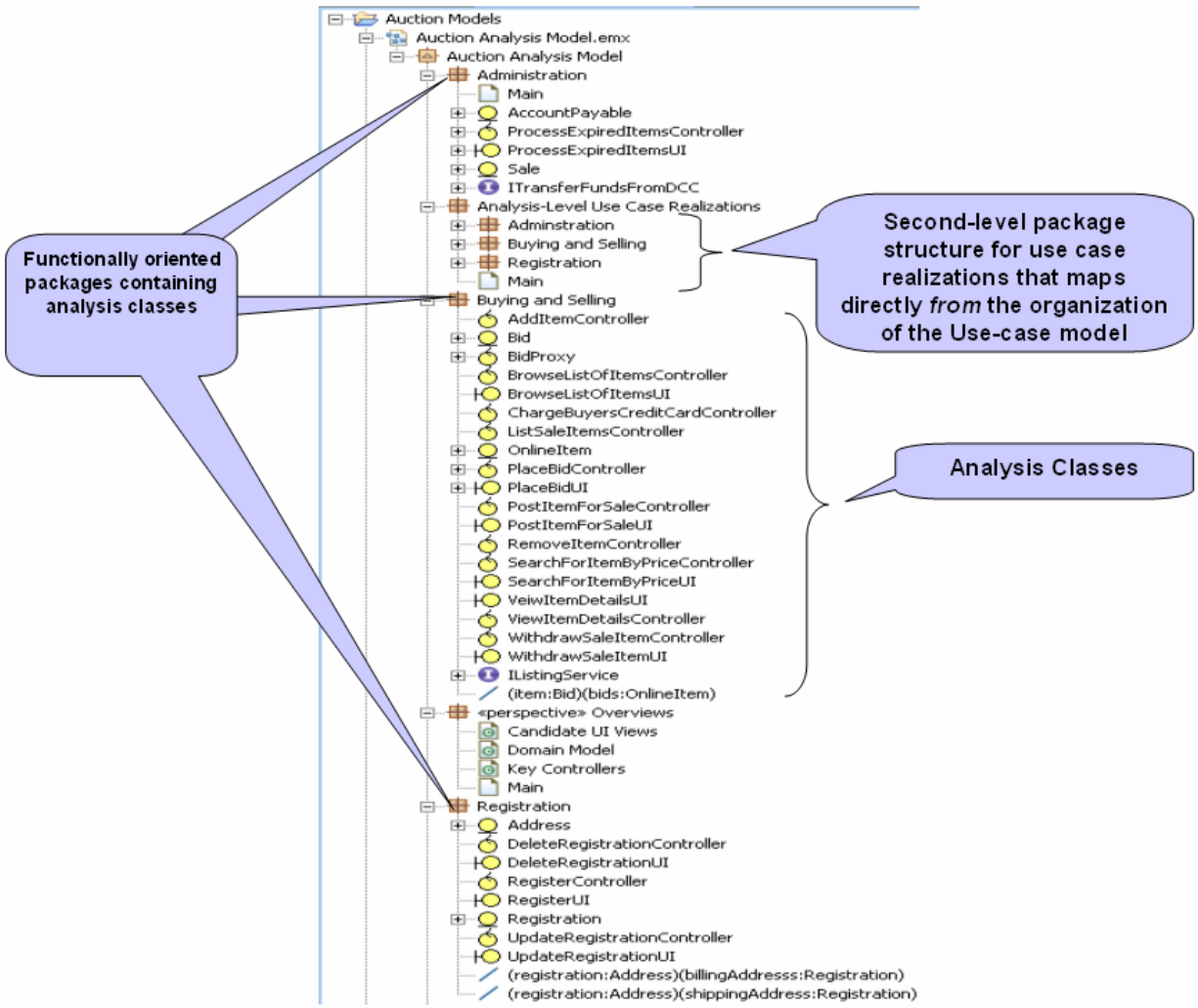


Figura 6-2

Según su situación, puede haber motivos para introducir el uso de un convenio de nombres que prevea la fusión y reutilización del contenido del modelo creado mediante varios grupos independientes, incluidos grupos de empresas diferentes (business partners). Si este es un problema, puede utilizar un convenio de espacio de nombres de dominio de Internet invertido como se muestra a continuación. **Figura 6-3**. Tenga en cuenta que probablemente este no sea un gran problema *en sí* para la creación de modelos de análisis, pero si adopta el método que permite que el modelo de análisis evolucione en un modelo de diseño *in situ* y tiene previsto volver a utilizarlo o realizar la integración de empresas a nivel de diseño, es posible que deba tenerlo en cuenta. Otra

ventaja potencial de adoptar este método es que puede correlacionar bien la organización del código generado a partir del análisis/diseño y puede simplificar la configuración posterior de las transformaciones que generan código.

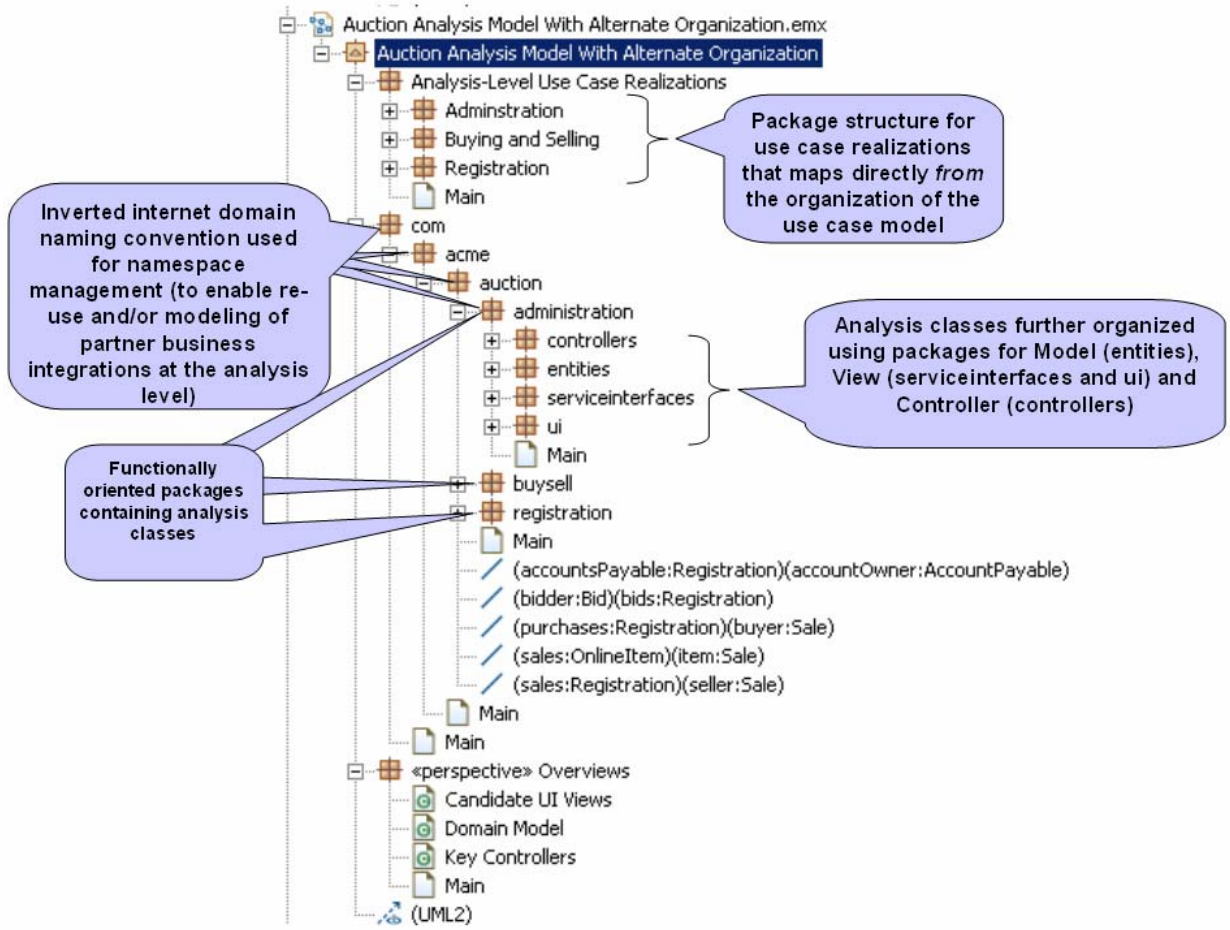


Figura 6-3

Contenido del modelo de análisis

Hay varios modos de detectar cuáles son las clases de análisis. Un modo es comenzar a dibujar los diagramas de secuencias que sugieren realizaciones de guiones de uso. A medida que lo haga, detectará las líneas básicas que necesita y generalmente cada una de estas líneas se corresponderá a una clase de análisis probable. Cuando descubra clases de este modo, puede crearlas en los paquetes de realización de guiones de uso del modelo de análisis pero no puede dejarlas ahí. Debe volver a modificar el modelo para trasladar las clases de análisis a paquetes orientados funcionalmente, como se ha descrito anteriormente en las directrices para la organización de alto nivel del modelo de análisis (consulte la ***Figura 6-1***).

Otro método práctico para descubrir las clases de análisis: "siembre" el modelo de análisis con clases basadas en estas normas generales:

- Para cada guión de uso, (del modelo de guiones de uso), añada una clase de "control" al modelo de

análisis. Las clases de "control" representan la lógica empresarial asociada al guión de uso. Posteriormente, durante el diseño, se correlacionarán también con cuestiones como, por ejemplo, la gestión de sesiones).

- Para cada relación entre actor y guión de uso (en el modelo de guiones de uso) añada una clase "límite" al modelo de análisis. Las clases "límite" representan interfaces entre la solución y un actor humano o entre la solución y algún sistema externo. Las clases "límite" que se corresponden con un actor humano finalmente se correlacionarán con uno o varios elementos de interfaz de usuario durante el diseño y la implementación. Las clases "límite" que se corresponden con un sistema externo finalmente se correlacionarán con alguna capa de adaptador en la fase de diseño e implementación.
- Durante un proceso como, por ejemplo, el análisis de tarjetas CRC, o un análisis de texto de las descripciones de los guiones de uso, identifique las clases de "control" (verbos) y las clases de "entidad" (sustantivos)

Cuando utilice este método de siembra para identificar las clases de análisis, puede colocar las clases directamente en los paquetes orientados funcionalmente como se ha descrito anteriormente en las directrices para la organización de alto nivel del modelo de análisis (consulte la ***Figura 6-1***).

No obstante, a medida que avance en la detección de las clases de análisis con toda seguridad reconocerá que es necesario realizar cambios en la organización del paquete funcional original.

Opcional: utilice los paquetes de segundo nivel en los paquetes de clases de análisis para organizar en mayor medida el contenido de dichos paquetes (consulte la **Figura 6-4**).

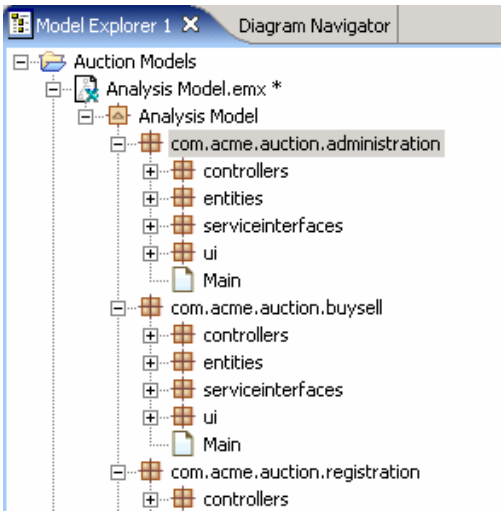


Figura 6-4

Recomendado: el modelo de análisis debe contener las realizaciones de guiones de uso a nivel de análisis que describen cómo se efectúan los guiones de uso en términos de las clases de análisis. Cada análisis de las realizaciones de guiones de uso (representadas mediante una colaboración UML) lleva a cabo un guión de uso del modelo de guiones de uso y tiene el mismo nombre que el guión de uso. Consulte la **Figura 6-5**. Para cada flujo de guión de uso con nombre [7] que cree que debe diseñarse como una realización de nivel de análisis, añada un diagrama de secuencia (que añadirá automáticamente una interacción propia). La **Figura 6-6** muestra los tipos de contenido semántico que se añadirán al modelo a medida que crea los diagramas de secuencias. Tenga en cuenta que puede filtrar los tipos de elementos UML en la vista del explorador de modelo y también ocultar la "confusión"

[7] Como se ha establecido anteriormente en el modelo de guiones de uso

que se muestra en la **Figura 6-6**).

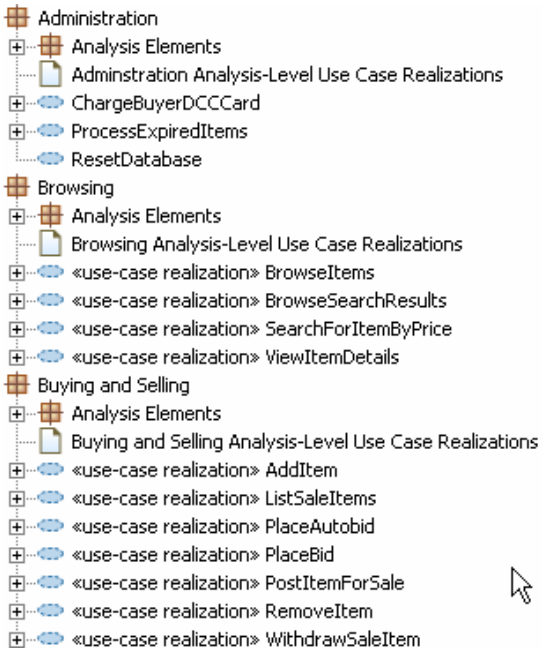


Figura 6-5

Opcional: cuando haya creado el diagrama de secuencias para un flujo de guión de uso, puede seleccionar su interacción UML propia en el explorador de modelos y añadirle un diagrama de comunicación. El nuevo diagrama de comunicación se rellenará automáticamente con las instancias de las clases de análisis que han participado en el diagrama de secuencias.

Recomendado: cree una relación de dependencia de realización a partir de la realización del guión de uso (Colaboración UML) y el guión de uso correspondiente del modelo de guiones de uso (consulte la **Figura 6-6**). Dado que puede utilizar características como los diagramas de temas y los análisis de rastreo para comprender las relaciones de rastreo del modelo, no necesita realmente retener los diagramas permanentes para mostrar las relaciones de rastreo, de modo que se le recomienda que cree las relaciones utilizando algún tipo de diagrama "desechable", por ejemplo:

- Añada un diagrama de formato libre a la colaboración.
- Arrastre la colaboración al mismo.
- Arrastre el guión del uso al mismo.
- Arrastre la relación de dependencia.
- Finalmente, en el explorador de modelos, suprima el diagrama de la colaboración.

Recomendado: incluya un diagrama de “participantes” para cada realización de guión de uso, para mostrar las clases de análisis que participan en la realización (esto es, las clases de análisis cuyas instancias aparecen en los diagramas de interacción que describen la realización del guión de uso) y las relaciones entre dichas clases que dan soporte a la colaboración descrita en los diagramas de interacción. Consulte la **Figura 6-6**.

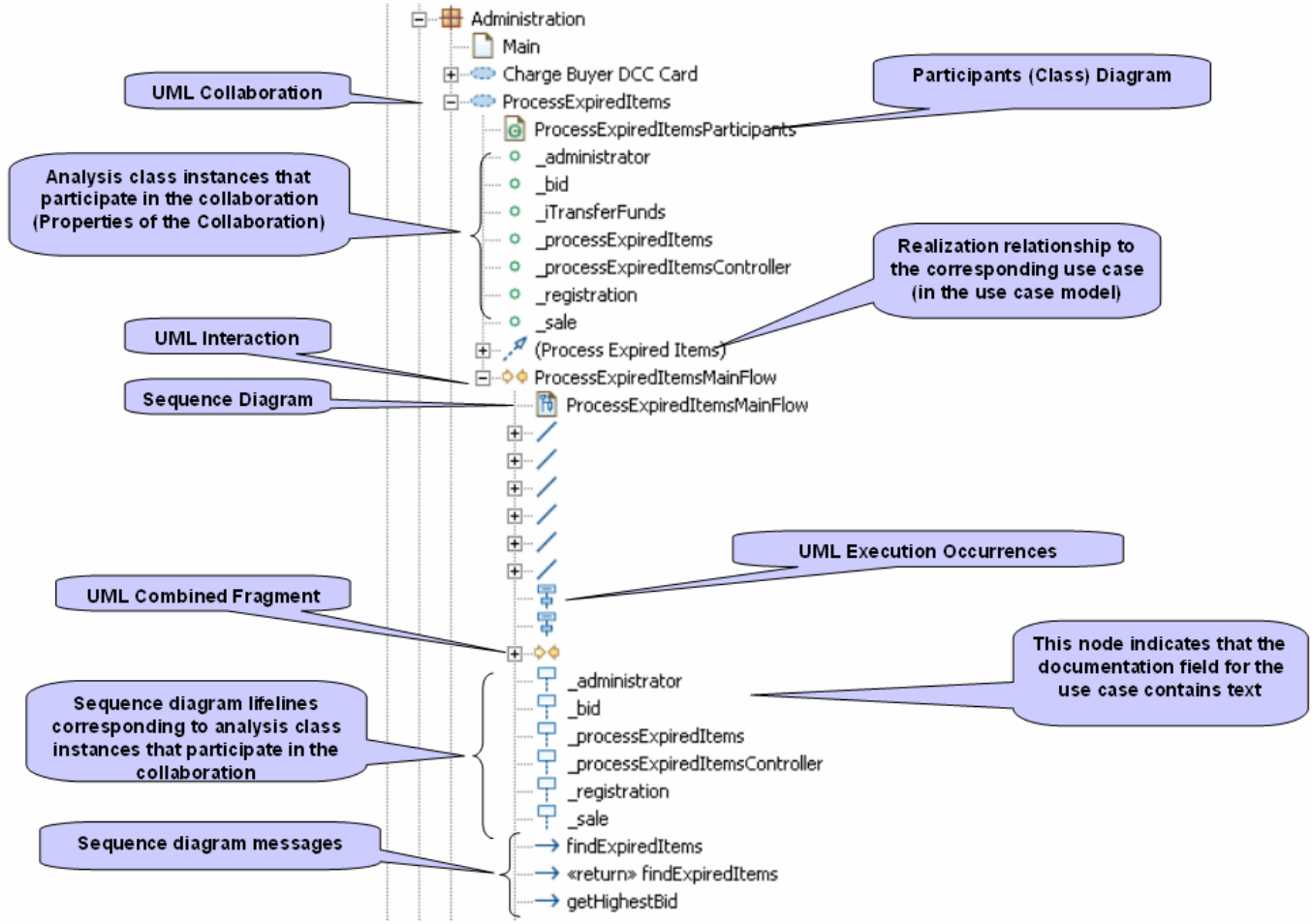
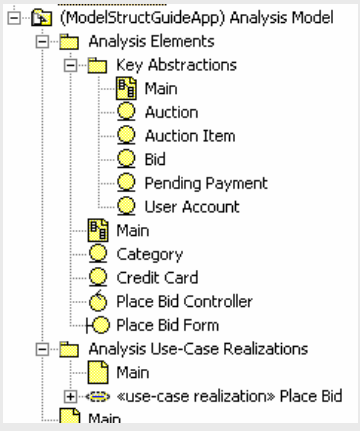


Figura 6-6

XDE/Rose

La estructura recomendada tradicionalmente para el **Modelo de análisis** que se muestra a continuación, se ha modificado en RSx para enfatizar una organización de paquetes orientados funcionalmente para las clases de análisis. Tenga en cuenta también que el uso de un paquete de abstracciones clave, Key Abstractions, (que puede comprometer un método de empaquetado orientado funcionalmente) se ha sustituido por el uso de un diagrama de abstracciones de claves (o diagramas) en una paquete de "perspectiva".



7. Directrices para la organización interna del Modelo de diseño

Organización de alto nivel del Modelo de diseño

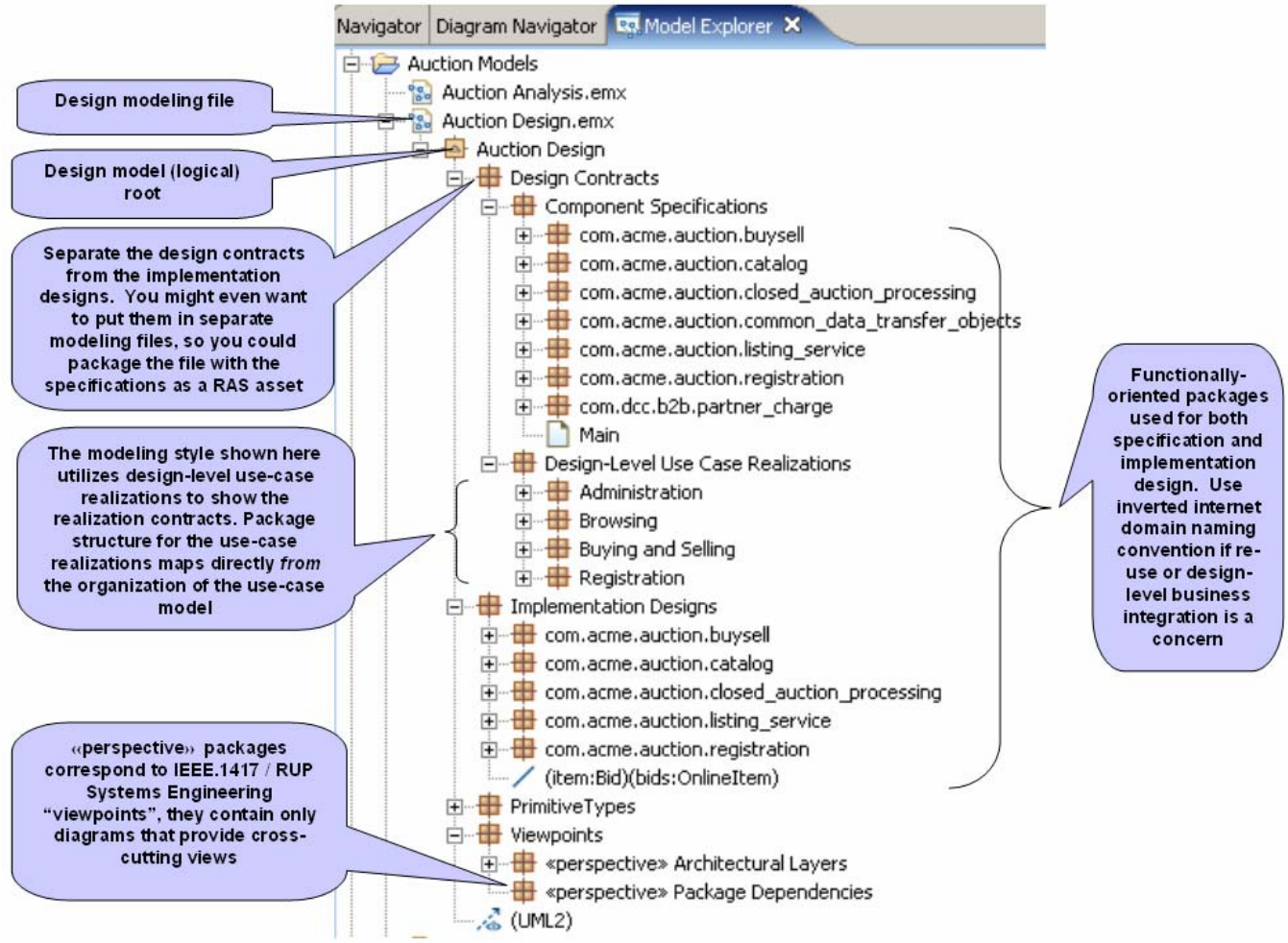


Figura 7-1

Figura 7-1 anterior ilustra las directrices siguientes para estructurar los modelos de diseño:

1. Especificaciones separadas de los diseños de implementación. La ilustración muestra el uso de paquetes de "Design Contracts" e "Implementation Designs" de nivel superior para conseguirlo.

2. Utilice los paquetes de nivel inferior para establecer agrupaciones orientadas funcionalmente. Por ejemplo, puede comenzar por la organización que ha utilizado durante el análisis y permitir su evolución a medida que toma decisiones relacionadas con el modo en que se correlacionan las clases de análisis con las clases de diseño reales, los componentes y los servicios. Es probable que cualquier esquema organizativo inicial evolucione durante el diseño, consulte la descripción que se incluye a continuación.

   Es posible que llegado este punto sea necesario hablar de los subsistemas. En las versiones de

UML anteriores a la versión 2 un subsistema es un tipo de paquete especializado. En UML 2 un subsistema es un tipo de componente especializado y un componente puede contener paquetes. Por lo tanto, en UML 2 los componentes de "subsistema" son alternativas viables de organización y espacio de nombres para paquetes, no obstante, UML 2 no es muy claro acerca del uso de subsistemas comparado con el uso de paquetes. Sugerencia: utilice paquetes con niveles de granularidad como, por ejemplo, el diseño de subsistemas de una aplicación determinada, y reserve los subsistemas para representar las aplicaciones completas, por ejemplo, CRM o SCM, en las vistas de la arquitectura de toda la empresa.

> ***XDE/Rose***
> *En el momento de la redacción de este documento estaba previsto que las herramientas de importación de modelos Rose y XDE ofrecieran la opción de correlacionar subsistemas UML 1.x con subsistemas UML2 o paquetes con la palabra clave* «subsystem» *aplicada.*

3. Es probable que la organización de los elementos de diseño evolucione más allá de cómo se organizan los guiones de uso del sistema (en el modelo de guiones de uso y, tal vez, en el modelo de análisis, si se mantiene un modelo de análisis separado). Utilice los paquetes para subdividir adicionalmente los contratos de diseño en especificaciones de elementos de diseño (contratos de uso) y en las realizaciones de guiones de uso de nivel de diseño (los contratos de realización) y mantenga una subestructura de paquetes para las realizaciones de guiones de uso que continúe duplicando la organización de los propios guiones de uso.

4. *Considere* las capas de la arquitectura como la base para el esquema organizativo de segundo nivel de los elementos que componen las especificaciones y los diseños de implementación de las áreas funcionales (consulte las descripciones que se muestran a continuación).

5. En los componentes y paquetes UML que agrupan elementos del modelo semántico, coloque diagramas que proporcionen vistas específicas de dicha agrupación. Esta directriz se aplica tanto si la agrupación está basada en subconjuntos orientados funcionalmente del dominio de empresa, en una capa arquitectónica o en otra cosa. Asigne al diagrama predeterminado el mismo nombre que al propio paquete o componente, y créelo de modo que muestre una vista general del contenido del paquete. De este modo, algunos diagramas se acercan a lo que muestran, lo que facilita la comprensión del modelo y su navegación.

6. Es posible que desee introducir el uso de un espacio de nombres del dominio Internet invertido en el modelo de diseño. Lógica:
   - Básicamente por las mismas razones que hacerlo es importante con respecto a las implementaciones específicas del lenguaje:
     a. Los escenarios que requieren trabajo de integración en los que participan varias aplicaciones dirigidas por modelos (especialmente con empresas asociadas).
     b. Escenarios de reutilización.
   - Esto simplificará la configuración posterior de las transformaciones en la implementación (la ubicación de origen a destino y la correlación de nombres).

7. *Considere* nombres de paquetes que sean válidos en las plataformas de implementación de destino, para evitar los inconvenientes y posible confusión de la correlación de espacios de nombres. (En gran parte, esto significa simplemente “no utilice en los nombres espacios ni signos de puntuación que no sea el signo de subrayado”).

8. Utilice las minúsculas en los nombres de paquetes para poder diferenciarlos fácilmente de los nombres de clase de un paquete.

9. *Utilice* nombres diferentes para las interfaces y para los componentes o clases que las realizan. Utilice ILoan y Loan o Loan y LoanImpl para los nombres de interfaz e implementación. Esto realmente no es necesario en el modelo, pero suele ser una buena idea cuando se genera código, de modo, que esta es otra área en la que puede ahorrarse el trabajo posterior de configurar las transformaciones.

10. En el escenario siguiente, cualquier contenido de nivel de análisis del que no se tenga que generar código se debe segregar en paquetes con el estereotipo "analysis"[8].
    - A) Ha seleccionado omitir la utilización de un modelo de análisis diferente y completar el modelo de diseño con contenido de nivel de análisis y mantener dicho contenido en el nivel de análisis de abstracción creando al mismo tiempo contenido de nivel de diseño en el mismo modelo, y
    - B) Dirigirá la transformación de modelo a código a partir del modelo de diseño de TI de empresa.

11. Utilice diagramas en los paquetes de "perspectiva" para capturar vistas cruzadas y de alto nivel de los elementos de diseño. Lógica: proporciona vistas cruzadas, vistas de contenido importante para la arquitectura y vistas dirigidas a los distintos usuarios interesados y al mismo tiempo mantiene los elementos semánticos del modelo organizados en grupos orientados funcionalmente.

Es importante reconocer que las estructuras de los paquetes de los modelos de diseño evolucionarán con el tiempo. Por último, la organización debe corresponder con el modo en que estructura la arquitectura en componentes y servicios. En general, este método de organización *final* del diseño permitirá obtener los mejores paquetes posibles de elementos reutilizables y la correlación más directa del diseño con el conjunto de proyectos y carpetas que contendrán los artefactos de implementación (código, metadatos, documentación) generado a partir del diseño.

No obstante, la organización *inicial* se debe corresponder más o menos directamente con el método de organización utilizado para el modelo de guión de uso y que luego se revisa durante el análisis[9]. De hecho, como se ha descrito en la sección anterior “Directrices para la organización interna del modelo de análisis”) puede optar por dejar que el modelo de análisis evolucione a diseño en el mismo sitio. En otras palabras, la organización inicial del diseño tenderá a agrupar conjuntos débilmente acoplados y cohesivos de aspectos empresariales y a aislar los elementos cruzados o reutilizables. Este método para la organización inicial ha demostrado ser eficaz ya que

- Si piensa utilizar las transformaciones que generan contenido del modelo de diseño a partir del contenido del modelo de guión de uso o análisis, las correlaciones de los paquetes de origen con los paquetes de destino serán sencillas y directas.
- Un método de organización inicial basado en el acoplamiento débil y cohesión funcional de los paquetes, será sin lugar a dudas el más adecuado para obtener la mejor correlación con la organización final orientada a componentes, lo que significa que reducirá la cantidad de modificaciones que deberá volver a realizar como parte del proceso de diseño.
- El acoplamiento débil de paquetes tiene el potencial de mejorar los flujos de trabajo del equipo y facilitar la reutilización en los casos en los que el diseño se descompone en varios archivos de creación de modelos.

Existen métodos alternativos y en algunos casos son recomendables como una organización *final*:

- Si su objetivo son las aplicaciones Web basadas en J2EE, incluidos EJB, la organización del diseño puede prever los convenios de RSA y Rational Application Developer relacionados con

[8] Las transformaciones ignorarán estos paquetes.

[9] A medida que se van detectando, los paquetes de las clases de análisis se vuelven a modificar generalmente de forma importante a medida que se va descubriendo, para poder dar soporte mejor a la reutilización de los requisitos funcionales no previstos.

proyectos J2EE.[10] En especial, puede optar por definir paquetes de diseño de nivel superior que se correspondan con las capas de la arquitectura (presentación y empresa, con la capa empresarial subdividida en las capas de sesión y dominio). Obviamente, este no es un método neutral para las plataformas y, por lo tanto, sólo se recomienda si sabe que la solución que está diseñando no se implementará en una plataforma que no sea J2EE.

- De un modo más general, suele darse el caso en el que las aplicaciones de varios niveles las crea un desarrollador con experiencia y la división del trabajo corresponde a las capas de presentación y empresarial, por lo tanto, una vez más puede optar por utilizar paquetes de nivel superior que se correspondan con estas capas de la arquitectura. Debe prestar atención con la organización de clases diseñadas a dar soporte a determinadas *funciones empresariales* para dar soporte a una arquitectura *determinada*. Resulta más difícil su modificación.
- Si encuentra un motivo para utilizar un método de organización no orientado a servicios/subsistemas/componentes, todavía podrá correlacionar la organización del diseño con un conjunto de proyectos y carpetas de destino realizando la tarea adicional de configurar las transformaciones de generación de código. Se puede utilizar un tipo especial de modelo auxiliar, al que se hace referencia como "modelo de correlación" para definir correlaciones especialmente complejas.

Contenido del modelo de diseño

No existen reglas estrictas ni rápidas sobre lo que deben residir en el modelo de diseño pero es posible que las recomendaciones siguientes le resulten útiles.

[10] En términos generales: un proyecto de empresa por sistema o aplicación o subsistema grande y para cada proyecto de empresa, un proyecto Web para la capa de presentación y varios proyectos EJB. Donde los proyectos EJB se corresponden generalmente con los componentes o con subsistemas menores y en los que generalmente se utilizan proyectos EJB diferentes para la capa de sesión (EJB de sesión) y la capa de dominio (EJB de entidad) por componente o subsistema. Consulte la sección 9 de este documento para obtener más información.

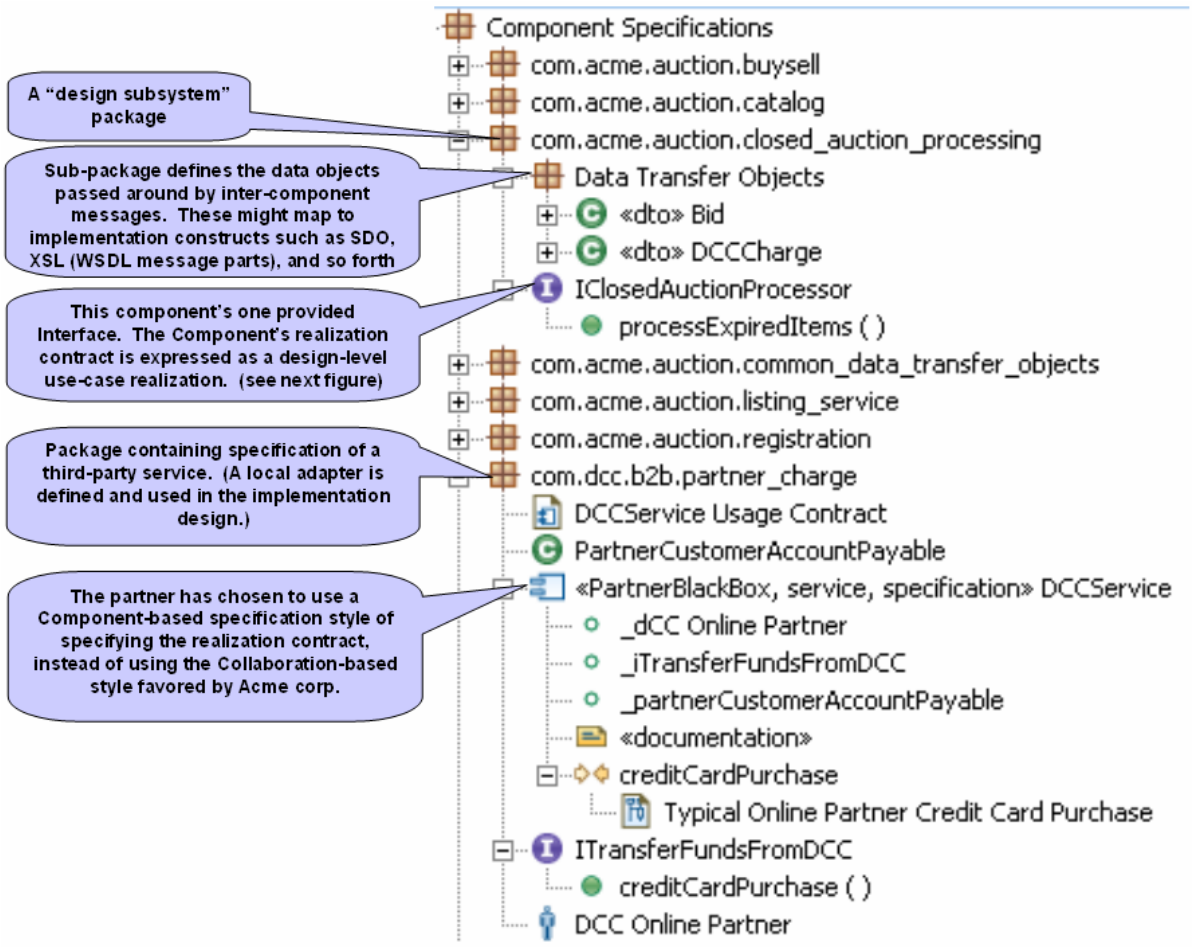

A "design subsystem" package
Sub-package defines the data objects passed around by inter-component messages. These might map to implementation constructs such as SDO, XSL (WSDL message parts), and so forth
This component's one provided Interface. The Component's realization contract is expressed as a design-level use-case realization. (see next figure)
Package containing specification of a third-party service. (A local adapter is defined and used in the implementation design.)
The partner has chosen to use a Component-based specification style of specifying the realization contract, instead of using the Collaboration-based style favored by Acme corp.
Component Specifications
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
Data Transfer Objects
«dto» Bid
«dto» DCCCharge
IClosedAuctionProcessor
processExpiredItems ( )
com.acme.auction.common_data_transfer_objects
com.acme.auction.listing_service
com.acme.auction.registration
com.dcc.b2b.partner_charge
DCCService Usage Contract
PartnerCustomerAccountPayable
«PartnerBlackBox, service, specification» DCCService
_dCC Online Partner
_iTransferFundsFromDCC
_partnerCustomerAccountPayable
«documentation»
creditCardPurchase
Typical Online Partner Credit Card Purchase
ITransferFundsFromDCC
creditCardPurchase ( )
DCC Online Partner

Figura 7-2

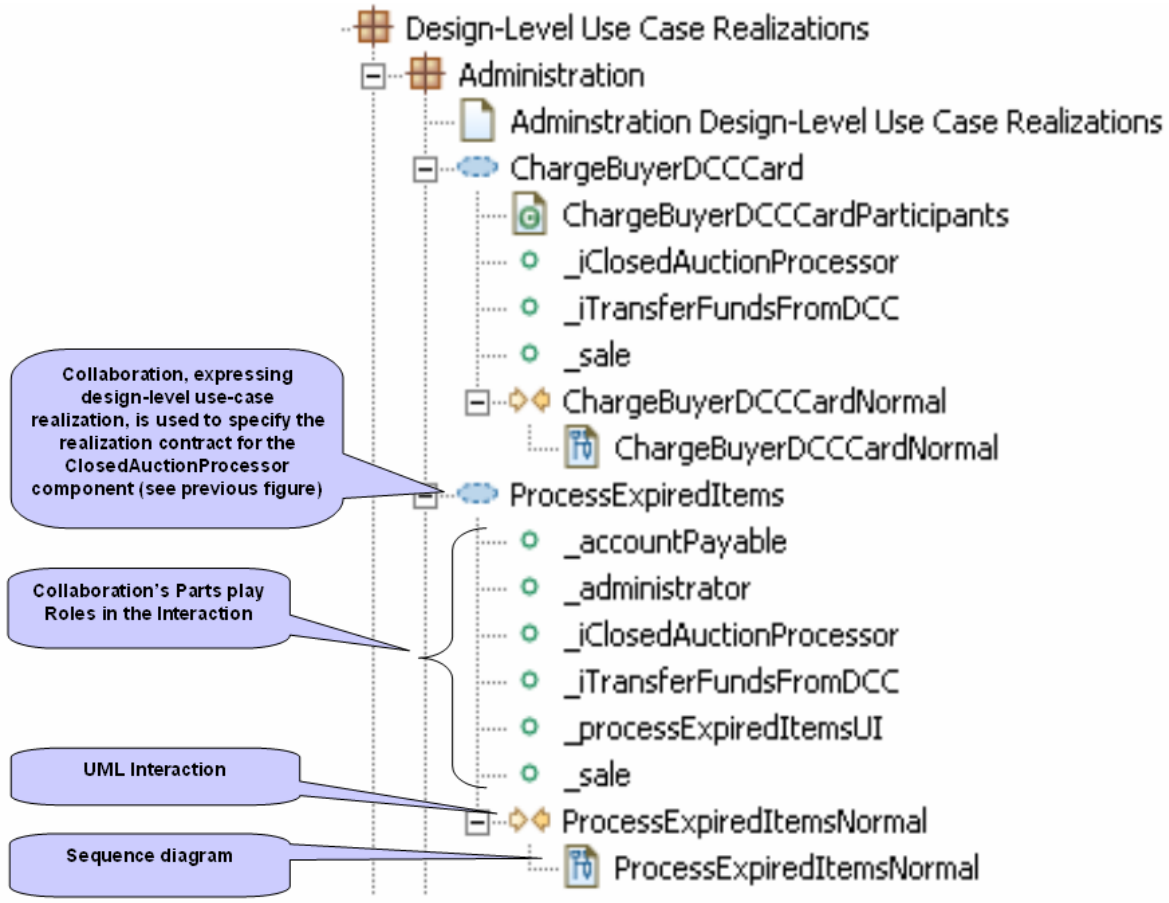


Figura 7-3

Figura 7-2 y **Figura 7-3** siguen la estructura organizativa que se muestra en **Figura 7-1** y muestran cómo se pueden especificar los contratos de diseño.

- El contrato de uso para un componente "ClosedAuctionProcessor" se expresa mediante una única interfaz[11] (**Figura 7-2**). El contrato de realización correspondiente se especifica mediante una única realización de guión de uso a nivel de diseño expresada como una colaboración[12] (**Figura 7-3**). Tenga en cuenta que aunque las realizaciones de guión de uso a nivel de análisis muestran las colaboraciones entre las clases de análisis, las realizaciones a nivel de diseño muestran las colaboraciones entre elementos de diseño menos abstractos [13]. Si se desea empaquetar el subconjunto de especificaciones de un modelo de diseño de forma independiente al subconjunto del diseño de implementación, resulta importante que las realizaciones de guiones de uso a nivel de diseño utilicen únicamente en sus roles elementos de especificación

[11] Por supuesto, se puede suministrar a los componentes varias interfaces, simplemente en este ejemplo sólo se proporciona una.

[12] Otros componentes pueden participar en varios guiones de uso, por lo tanto, sus contratos de realización pueden residir en varias realizaciones de guiones de uso. En dichos casos, también puede incluir, en el mismo paquete que la interfaz del componente, un diagrama denominado "{nombre de componente} dónde se utiliza" en el que puede colocar varios enlaces a los diferentes diagramas que componen las realizaciones de guiones de uso para dichos guiones de uso.

[13] Otra diferencia probable es que algunos diagramas que "participan" en las realizaciones de nivel de diseño pueden ser diagramas de componentes que muestran las relaciones entre componentes, en lugar de (o además de) los diagramas de clase participantes que se sugieren para las realizaciones de guiones de uso a nivel de análisis.

de análisis o de diseño –pero nunca elementos de diseño de implementación.

- Los contratos de uso y realización para el “DCCService” de terceros se incluyen en un solo paquete[14]. Una vez más, vemos que el contrato de uso consta de una sola interfaz pero que en este caso el contrato de realización se expresa utilizando un componente de "especificación" (**Figura 7-2**). (De lo contrario, la especificación del contrato de realización es prácticamente igual y se expresa utilizando comportamientos – en este caso una interacción denominada “creditCardPurchase”).Puede encontrar otro ejemplo de uso de componentes en lugar de colaboraciones en la **Figura 7-4**.

- Las operaciones se definen en interfaces que se pueden realizar mediante los componentes de "especificación" (si se utilizan) o mediante clasificadores en el diseño de la implementación que implementan las interfaces.

- Como parte del contrato de uso también se puede incluir la especificación de los objetos de transferencia de datos (que servirán como los tipos de parámetros de las operaciones proporcionadas y pueden correlacionarse con las construcciones de implementaciones como, por ejemplo, el esquema XML o SDO). Para los componentes que no se han diseñado de modo que se puedan distribuir, puede optar por especificar o no los objetos de transferencia de datos como especificaciones de los tipos utilizados como parámetros de operación. Para los servicios distribuibles, por ejemplo, los servicios Web, es obligatorio que las operaciones del servicio no hagan referencia a objetos de un espacio de direcciones local, por lo tanto, se deben utilizar DTO.

XDE/Rose
En las versiones anteriores de UML, la directriz para las realizaciones de guiones de uso era utilizar una instancia de colaboración por guión de uso y un diagrama de interacción y secuencia para cada uno de los flujos importantes de la realización.

En general, en RSx puede utilizar simplemente una interacción y un diagrama ya que los diagramas de secuencia UML2 dan soporte a anotaciones para rutas de ejecución alternativas.

Asimismo, en UML 2 ya no existe una "instancia de colaboración". En lugar de ello, existe un "uso de colaboración", que requiere como tipo una colaboración. Por lo tanto, en RSx utilice las colaboraciones para las realizaciones de guiones de uso

- [14] Tenga en cuenta que la situación hipotética es que la empresa DCC suministraba a la empresa Acme la especificación UML, y Acme se incorporaba después a su modelo de diseño. Este es el tipo de escenario en el que pueden resultar útiles los espacios de dominios de Internet invertidos.
-

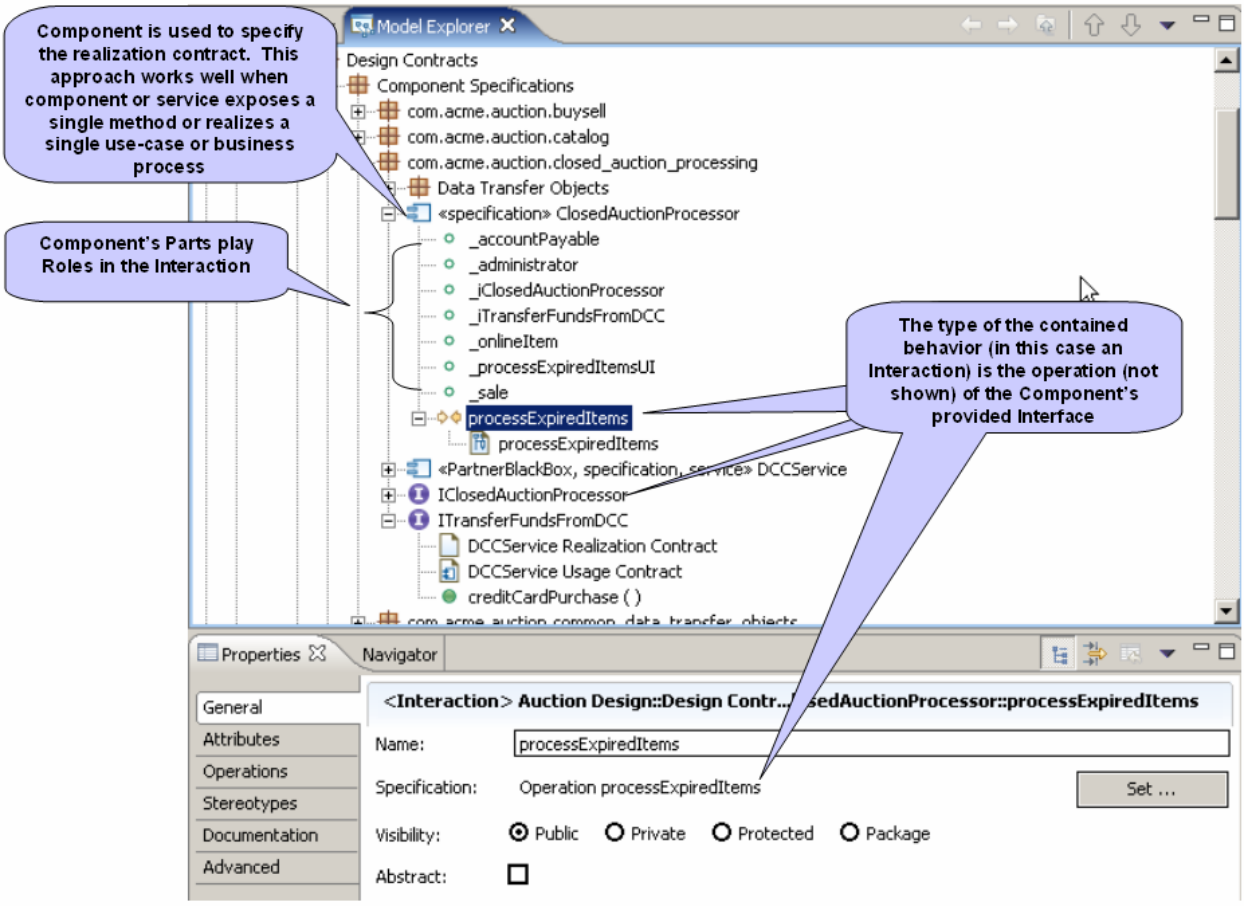


Figura 7-4

- Puede encontrar un posible método para especificar los diseños de implementación en la **Figura 7-5**. La estructura de la implementación se define utilizando clases simples que contienen operaciones. Este método es típico de los modelos de diseño creados utilizando UML 1.x. Encontrará un segundo método que puede resultar mejor para mantener los objetivos de UML2 en la **Figura 7-6**. Aquí, en lugar de las clases vemos que se utilizan componentes y que los componentes no poseen operaciones sino que poseen comportamientos (en este caso una interacción).

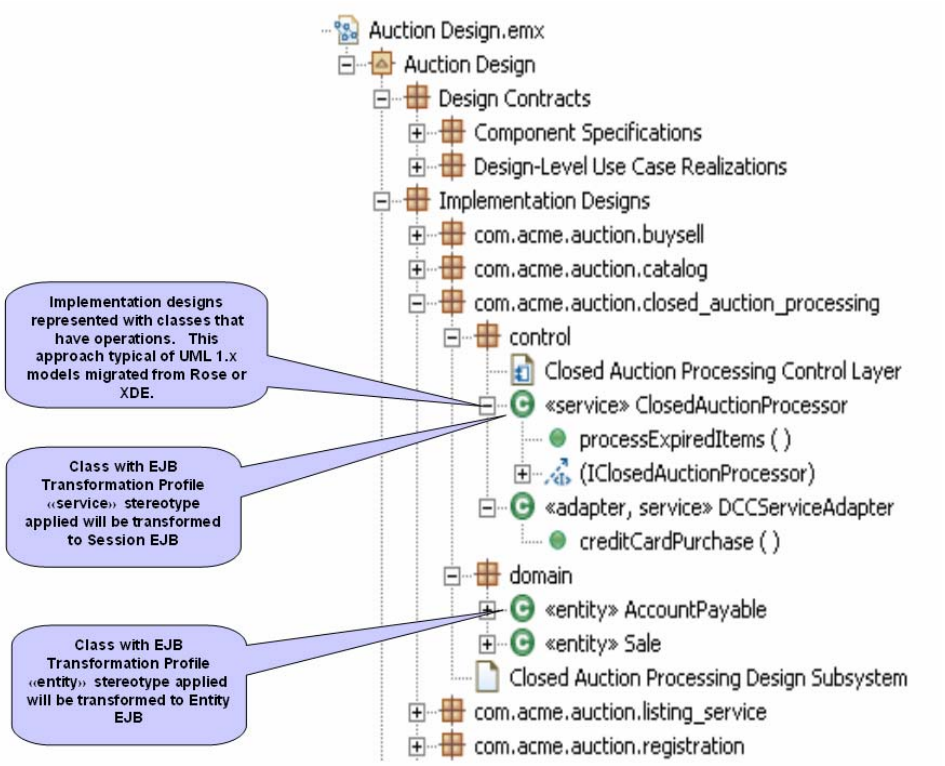


Figura 7-5

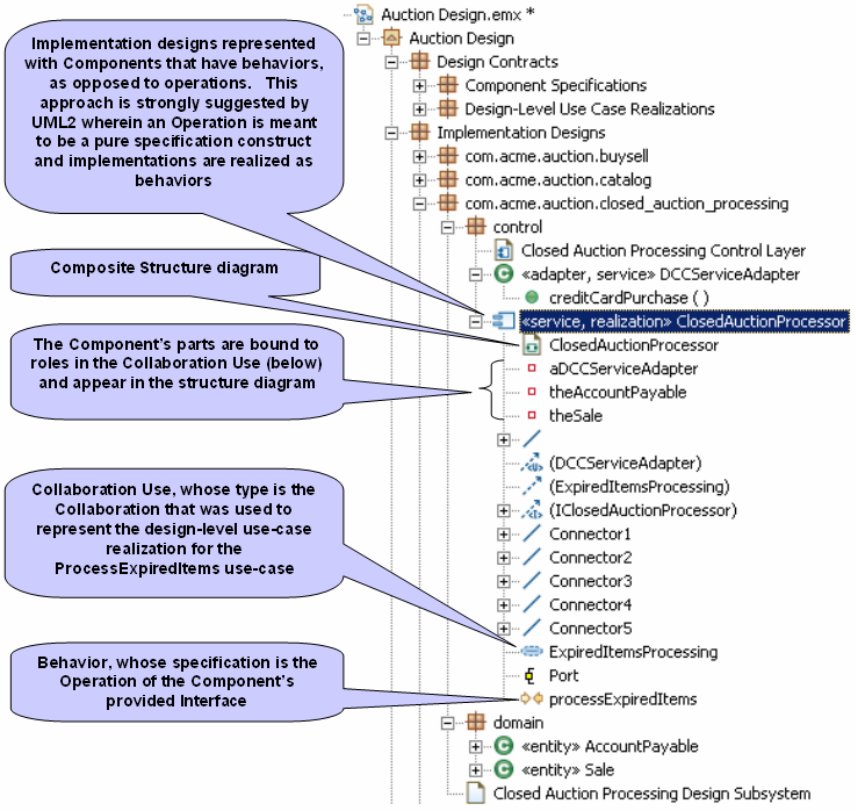


Figura 7-6

8. Directrices para la organización interna del Modelo de visión general de la implementación

> ***XDE/Rose***
> En las directrices de la estructura de modelos de XDE, se recomendaba un modelo de vista general como un dispositivo para proporcionar una visión general del nivel de subsistemas de la implementación. Los detalles de cada subsistema se especificaban en el modelo de código del proyecto que implementaba el subsistema.

Estrictamente hablando, no deberá ser necesario utilizar un modelo de vista general de la implementación en RSx. Si se siguen las directrices organizativas del modelo de diseño, entonces la organización (final) del modelo de diseño debe formarse de modo natural alrededor de los componentes (incluidas las variedades más sólidas de "subsistema" y las más distribuibles de "servicio"). A continuación, mediante las transformaciones, los paquetes de diseño se pueden correlacionar con los proyectos. Por ejemplo, en el caso de una implementación J2EE se correlacionarán con los diferentes proyectos Java, EJB, Web, aplicación J2EE y otros en los que se ha desarrollador la implementación. Y, de hecho, dichos proyectos representan el modelo de implementación para la solución, como se indica en la sección de Conceptos básicos y terminología de este documento.

Si lo analiza ascendentemente, indica que debe considerar el modo en que se organizará la implementación en relación con los proyectos y carpetas y que debe considerarlo en relación con la organización del modelo de diseño, de modo que las correlaciones de las transformaciones acaben siendo directas. Pero si lo analiza descendentemente, indica que la organización del modelo de diseño, tal y como ha evolucionado durante el trabajo de diseño, debe determinar el conjunto de modelos de implementación (proyectos) necesarios. De cualquier modo, la organización final del modelo de diseño debe cubrir de forma natural la necesidad de una visión general de los proyectos y subproyectos, esto es, un diagrama que muestre los paquetes del modelo de diseño debe ser equivalente a una visión general de los proyectos y sus subcarpetas.

No obstante, es posible que prefiera realizar un borrador de la estructura del proyecto en la primera fase o que simplemente prefiera ver una descripción de las estructuras del proyecto que resulte más explícita visualmente – por ejemplo, una en la que los elementos que representan proyectos y carpetas se indican de forma explícita con palabras claves como "project" y "folder" o incluso como "EJB Project" y "Web Project". Otra consideración es que mostrar los artefactos de implementación más refinados, por ejemplo, los JAR, puede no resultar adecuado para el modelo de diseño (que en la teoría de funcionamiento de Rational Software Architect está diseñado para que sea neutro para las plataformas). Pero dichos elementos pueden resultar perfectamente aceptables para incluirlos en el modelo de vista general de la implementación. Por lo tanto, existen algunos motivos por los que es posible que desee utilizar un modelo de vista general de la implementación. **Figura 8-1**. A continuación, se muestra un modelo de vista general de la implementación de ejemplo.

Una consideración final es que un modelo de vista general de la implementación puede ser un buen lugar para capturar diagramas informativos de los diferentes aspectos de la solución. La **Figura 8-2** siguiente muestra un diagrama informal conceptual del sistema de subastas en el que están basados la mayor parte de los ejemplos de este documento.

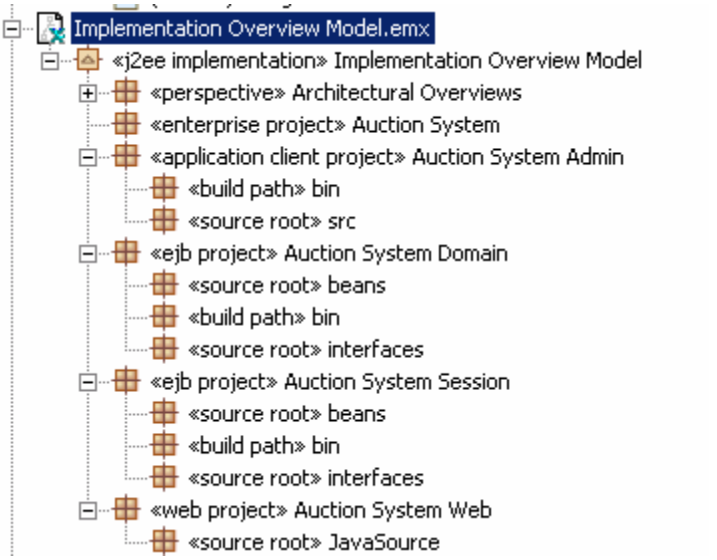


Figura 8-1

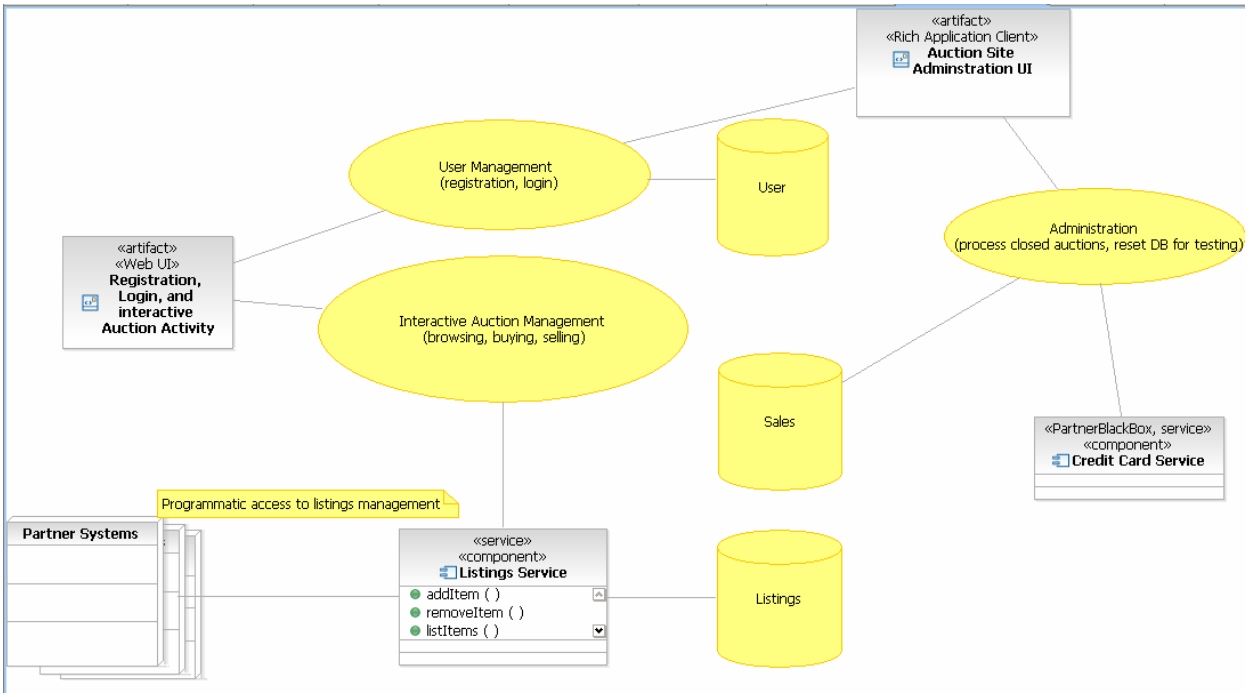


Figura 8-2

9. Directrices para la organización interna del modelo de despliegue

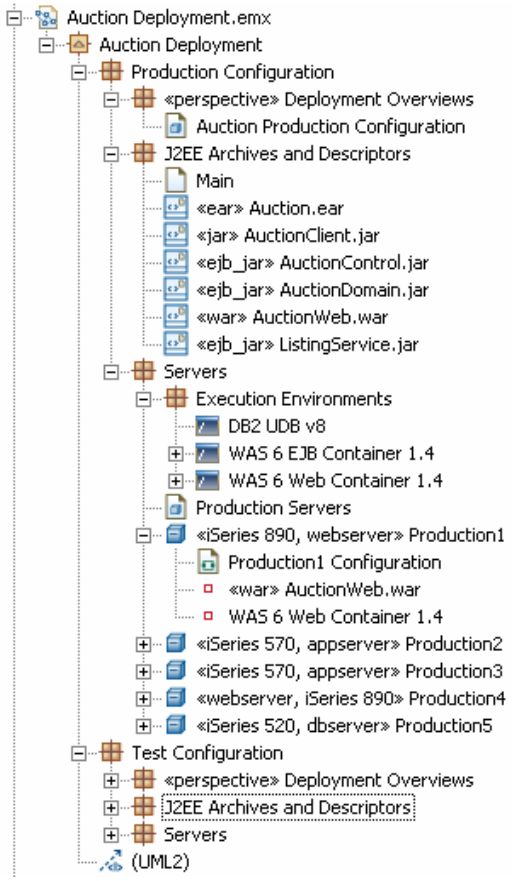


Figura 9-1

Probablemente, se deba hablar menos del modelo de despliegue que de cualquier otro modelo descrito en este documento. En general, habrán menos implicaciones en sentido descendente en la organización de la creación de modelos de despliegue y en las opciones de contenido, por lo tanto, lleve a cabo las acciones que tengan sentido. Sin embargo, para que le ayude a pensar, puede encontrar una estrategia posible y un poco de contenido representativo en la **Figura 9-1**. Tan sólo un par de notas relacionadas con este ejemplo:

1. Las especificaciones de las configuraciones de producción se han separado de las de las configuraciones de prueba.

2. las vistas generales (por ejemplo, de clústeres, centros de datos o empresas) se mantienen en paquetes de "perspectivas".

3. Se ha adoptado un método simple en relación con la clasificación y tipificación de los nodos y artefactos: una combinación de paquetes y de uso de palabras clave. Un método más sofisticado podría ser desarrollar un perfil UML especializado que defina los estereotipos especializados y las propiedades adecuados para describir y documentar los tipos de recursos utilizados en su propio entorno.

10. Utilización de un archivo de creación de modelos para representar el documento de la arquitectura de software

Con las herramientas para organizar modelos que contiene como, por ejemplo, enlaces de diagramas y soporte para varios archivos de modelos con referencias cruzadas de modelos, se convierte casi en un asunto trivial crear un modelo que, de hecho, represente el documento de la arquitectura de software RUP y las vistas de arquitectura “4+1”.

En su forma más simple, puede llevar a cabo algo similar a lo que contiene la **Figura 10-1**. Cree un archivo de creación de modelos y rellénelo con un conjunto de paquetes sencillos correspondiente a las vistas 4+1. (El ejemplo se muestra sin un paquete para la vista de procesos, ya que el sistema de este ejemplo no presenta demasiados aspectos en relación a la concurrencia).

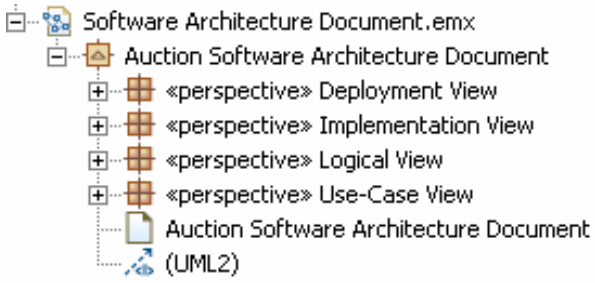


Figura 10-1

A continuación, puede componer el diagrama predeterminado siguiendo las líneas que se sugieren en la **Figura 10-2**. También puede añadir notas o texto adicional a este diagrama

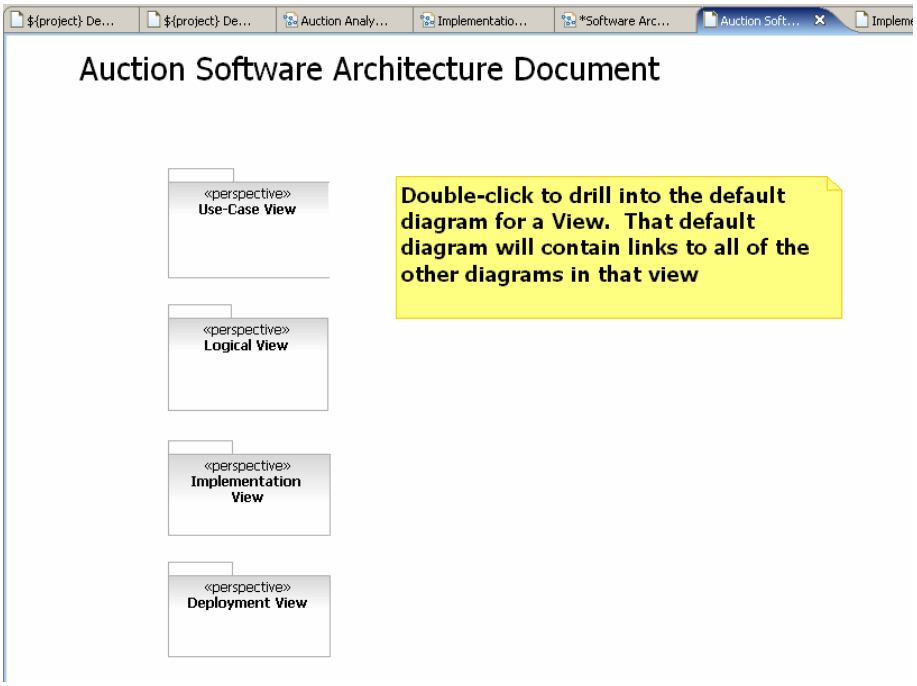


Figura 10-2

A continuación, simplemente cree diagramas en el archivo de creación de modelos del documento de la arquitectura de software utilizando estos métodos:

- Cree los diagramas compuestos mediante elementos semánticos UML de otros archivos de creación de modelos de modo que muestren nuevas vistas que no estaban en los otros archivos de creación de modelos pero que se necesitan como parte del documento de arquitectura.

- Cree los diagramas compuestos de formas geométricas y/o de elementos UML “ad-hoc” que residen en el archivo de creación de modelos del documento de la arquitectura de software. (Dichos elementos UML simplemente deben ser para fines de documentación o clarificación y no deben tener importancia semántica con la implementación real de la solución descrita).

- Cree diagramas que simplemente contengan enlaces con los diagramas existentes en otros archivos de creación de modelos. (Esta técnica funciona bien si el archivo de creación de modelos del documento de arquitectura se ha de distribuir junto con otros archivos de creación de modelos para que lo utilicen los lectores. Si el documento de arquitectura se va a publicar en la Web, siga uno de los otros métodos).

11. Consideraciones acerca del desarrollo en los equipos y la gestión de modelos

En esta sección se describen algunas de las consideraciones que se han de tener en cuenta en relación con cuándo y porqué opta por realizar una partición de un modelo en varios archivos de creación de modelos. Puede encontrar un tratamiento más extenso de estos temas en la ayuda en línea de RSx. Aquí se presupone que el lector ya está familiarizado con los conceptos del desarrollo en paralelo y con la idea de fusión de los cambios realizados en paralelo con las diferentes copias de un elemento.

En primer lugar, ofrecemos una revisión rápida: en la sección “Conceptos básicos y terminología” se han descrito los diferentes tipos de modelos como, por ejemplo, de guión de uso, de análisis y de diseño, como los reconoce RUP. Se han ofrecido ejemplos para ilustrar estos conceptos en RSx…

- Si está creando varias aplicaciones, es posible que tenga varios modelos de cada tipo (por ejemplo, varios modelos de guión de uso, varios modelos de análisis, etc.).

- Un modelo (en el sentido lógico) se puede guardar como uno o varios archivos de creación de modelos, por ejemplo, «el modelo de diseño para la aplicación "X"») se puede mantener como un archivo de creación de modelos individual o como una colección de varios archivos de creación de modelos.

Partición de modelos

Las técnicas para la partición de modelos en varios archivos de creación de modelos se describen en la ayuda en línea de RSx y no se tratan aquí. En este documento nos interesaremos con el cuándo y el porqué de una partición. Hay dos circunstancias bajo las cuales puede optar por mantener un modelo determinado como varios archivos de creación de modelos:

1. El modelo ha adquirido un tamaño difícil de manejar o su estructura de paquetes ha adquirido una profundidad difícil de manejar.[15]

2. Comienza a recibir demasiados cambios simultáneos en un archivo de creación de modelos, dando lugar a la necesidad de realizar fusiones con más dos contribuyentes >de cambios [16]. Si esta descripción no le resulta clara, siga leyendo.

Creación de modelos en equipos

Cuando la política de gestión de la configuración permite el desarrollo en paralelo de modelos [17], se

[15] La partición es necesaria principalmente cuando el tamaño de los archivos aumenta demasiado para las máquinas que se utilizan en la comunidad de usuarios. Por ejemplo, resulta difícil trabajar cotidianamente con un modelo que alcanza los 30 MB de disco en una máquina de 1 GB de RAM. En dicha máquina, es posible que se desee particionar el modelo a fin de mantener de 5 a 10 MB de los modelos en la RAM en cualquier momento. Un método alternativo es actualizar la RAM (una solución relativamente barata, pero que funciona muy bien). Una máquina con 2 GB de RAM sin un archivo de intercambio tiene un rendimiento más rápido en prácticamente cada operación de Eclipse, lo que hace que los modelos muy grandes vuelvan a funcionar bien.

[16] Las herramientas de fusión del modelo RSx dan soporte a las fusiones de un máximo de 3 contribuyentes: dos contribuyentes de "cambios" y un contribuyente "base" (esto es, el ancestro común). Cuando hay más de dos contribuyentes de "cambios" que manejar, las fusiones se realizarán en cascada y, a partir de ese punto, uno de los dos contribuyentes de "cambios" será el conjunto de resultados de las fusiones completadas previamente en la sesión.

[17] Ejemplos de políticas de desarrollo en “paralelo”:

- Se puede extraer un archivo de creación de modelos para un acceso no exclusivo.
- En los procesos de desarrollo de distintos miembros se trabaja con un archivo de creación de modelos en paralelo.

realizan cambios sin coordinar en el archivo y, de he hecho, también en el modelo lógico o en el subconjunto de modelos que representa el archivo. En algún punto se deben fusionar estos cambios. Si resulta que algunos de los cambios entran en conflicto con otros, la fusión se considera "**no trivial**" debido a que alguien debe tomar una decisión en relación con qué cambios conflictivos se deben "mantener". Una fusión trivial es aquella en la que los cambios no coordinados no entran en conflicto y el motor de fusión del modelo puede realizar las fusiones sin la intervención manual del usuario.
Las fusiones no triviales pueden resultar arduas. ¿Cómo se pueden minimizar?

Tiene dos herramientas básicas para evitar conflictos y las fusiones no triviales resultantes de los mismos. Una es una "arquitectura sólida". La otra es una "propiedad sólida". Las dos van mano a mano, la arquitectura sólida *da pie* a una sólida propiedad.

Herramienta número 1: arquitectura sólida

La "arquitectura sólida", en este contexto, hace referencia primordialmente a la descomposición. Los ***principios*** de la descomposición de la arquitectura que se aplican aquí son los mismos que conducen el desarrollo orientado a objetos, el diseño basado en objetos y las arquitecturas orientadas a servicio:

- Permiten un desacoplamiento máximo de las funciones empresariales.

- Agrupan los elementos que deben permanecer fuertemente acoplados. Aíslan estas agrupaciones entre sí.

- Si la descomposición resultante tiene un gran número "partes", entonces dependiendo del modelo de personal (recuerde que la arquitectura sólida y la propiedad sólida van mano a mano), es posible que desee agrupar estas partes en agregados de alta afinidad (lo que en términos de creación de modelos significa paquetes UML).

- Es posible que ***siempre*** haya cosas que deben manipularlas muchas (y en algunos casos todas) de las unidades de descomposición. Agrupe estas cosas en un paquete "común" y planifique cada iteración del desarrollo de modo que incluya una pequeña "cascada" al principio de la iteración con el objetivo de estabilizar los elementos "comunes".

- También existe el elemento de tiempo. A medida que pasa de una comprensión más abstracta a una comprensión más concreta de la a solución, aprenderá a organizar mejor la arquitectura (y el modelo). Debe planificar una actividad para volver a modificar el modelo (volver a organizar) durante la transición de cada fase a la siguiente (análisis de negocio, requisitos, análisis de aplicaciones, diseño de alto nivel...).

Si analiza la solución y todo le parece altamente interdependiente y fuertemente acoplado, es posible que se deba trabajar más la arquitectura o bien que hay algo en la naturaleza de su dominio de problemas que implica que realmente no puede descomponer el problema. En cualquier de estos casos, tiene dos opciones:

- Asignar el proyecto a un equipo muy pequeño que comparta un espacio físico y se comunique activamente entre sí en relación con cualquier cambio que realice que afecte de algún modo a otros elementos.
- Prepararse para realizar muchas fusiones no triviales.

XDE/Rose

Si es usuario de Rose o XDE, quizás haya tenido que realizar fusiones de modelos no triviales. La buena noticia es que en RSx las fusiones de modelos no triviales son menos problemáticas. Una diferencia importante es que RS***x*** opera en una ***combinación*** de archivos de creación de modelos relacionados y no en una ***jerarquía*** de archivos hoja o subunidades. Esta diferencia importante significa que se da soporte a la descomposición lógica de alto nivel mejor que en el nivel físico de RS***x***.

También se da soporte a la comparación de fusiones con mejores herramientas – de hecho, las operaciones de fusión resultan **mucho mejores**.

- El motor de fusiones de proceso de fondo en RS***x*** es 10.000 (sí, diez mil) veces más rápido que en el proceso de fondo de XDE.
- También implementa diferentes tecnologías como, por ejemplo, los "deltas compuestos" (un mecanismo de agrupación que clasifica los cambios por diagramas y agrupa las tendencias de diagramas de gran tamaño de modo que puedan realizar operaciones de grupo u operaciones atómicas), la "protección de integridad del modelo", la "atomicidad" y el "proceso de delta en cascada". Así se disminuye la posibilidad de dañar un archivo fusionándolo al mismo nivel, de un modo tan simple como editando el archivo.

Herramienta 2: propiedad sólida

Una vez establecida la descomposición de la arquitectura sólida, la correlación de la propiedad "sólida" de los componentes de la arquitectura con los técnicos individuales o equipos pequeños se realizará de forma directa (dejando a un lado los conocimientos especializados). Cuando un técnico puede trabajar de forma exclusiva en cada paquete lógico (o rama) de un modelo, las fusiones que se realicen en dicho modelo serán en gran parte triviales (independientemente de si el modelo se almacena como un archivo de creación de modelos individual o como varios archivos de creación de modelos). Lo mismo se puede decir si en cada rama puede trabajar de forma exclusiva un pequeño equipo cuyos miembros suelen realizar muchas comunicaciones informales acerca de lo que están llevando a cabo.

¿Se pueden evitar las fusiones no triviales dividiendo los modelos en varios archivos de creación de modelos? En resumidas cuentas: "no". **Las interdependencias de la arquitectura son un fenómeno lógico, no un fenómeno físico**. Cuando efectúa una partición del modelo en varios archivos de creación de modelos, las representaciones de las interdependencias de los elementos simplemente se convierten en referencias entre archivos y no en referencias en los archivos. No facilita la resolución de conflictos (de hecho, la vuelve más complicada). Y cuando introduce las referencias entre archivos, introduce puntos de problemas potenciales (vea el recuadro).

Referencias cruzadas de archivos de creación de modelos

Cuando dos elementos de creación de modelos residen en archivos de creación de modelos diferentes y crea una relación entre los mismos, crea una “referencia cruzada de archivos de creación de modelos”. Dado que los archivos de creación de modelos (archivos .exm) están expuestos en el sistema de archivos del sistema operativo y se pueden mover, renombrar o modificar de cualquier otro modo fuera del entorno de Eclipse, estas referencias representan puntos de problemas potenciales. No obstante, siempre que los archivos de creación de modelos se modifique o que sus cambios se gestionen, en el entorno Eclipse y respete las directrices siguientes no sufrirá estos puntos de división.

Cuando trabaje con (edite) un "alojamiento" de los archivos de creación de modelos, esto es, un grupo de archivos de creación de modelos que se hacen referencia entre sí, debe tener todos los archivos de creación de modelos presentes en el espacio de trabajo. Esto no implica estrictamente que todos los archivos de creación de modelos de un "alojamiento" deban residir en el mismo proyecto sino que al utilizar un solo proyecto se garantiza, generalmente, que todos los modelos estarán presentes, ya que en los flujos de trabajo típicos del gestor de configuración todos los archivos de modelos van juntos en el mismo proyecto.

Resumen:

- Si su arquitectura no es sólida o si tiene una arquitectura pero la propiedad no es sólida, tendrá que realizar frecuentemente fusiones no triviales que no podrá evitar aunque efectúe muchas particiones de los modelos.

- Si tiene una arquitectura sólida y una propiedad sólida, disminuirá de forma significativa la frecuencia de las fusiones no triviales aunque no las eliminará. No las eliminará porque siempre habrán interdependencias de componentes. Los elementos "comunes" mencionados anteriormente son un ejemplo, aunque no pueden ser el único ejemplo.

- La partición de los modelos en varios archivos no es ni mucho menos tan importante como la estructuración lógica de los modelos para permitir que diferentes técnicos trabajen en un archivo de creación de modelos de forma paralela sin introducir cambios conflictivos.

- La buena noticia es que RS*x* maneja la fusión de modelos con mucha más rapidez y eficacia que otras herramientas de creación de modelos disponibles.

¿Cuándo se debe realizar una partición de modelos? Intente evitar la partición, excepto en aquellos casos en los que el tamaño de un modelo ha comenzando a poner a prueba los límites del hardware y cuando habitualmente sea posible trabajar con los archivos de creación de modelos de forma exclusiva, esto es, *un miembro del equipo* tiene reservado un archivo en un momento dado, **y** *de forma aislada*, esto es, la mayor parte de los cambios se pueden realizar en el archivo sin que sea necesario acceder a otros archivos que contienen elementos del modelo relacionados.

Por último: Cambios en las versiones 6.x y 7.x de RSx

En el primer release de RSx (las versiones 6.x), no se daba soporte al concepto de "subunidad" como se da soporte en Rose y XDE (una subunidad es un archivo de creación de modelos que contiene un subconjunto de un modelo y es "transparente" en el sentido de que no aparece en la vista lógica del modelo que ve en la vista del explorador de modelos).En su lugar, la partición de modelos se limitaba a definir varios modelos de nivel superior, “nivel superior” en el sentido que cada archivo de creación de modelos debe aparecer como una entrada de nivel superior independiente en la vista del explorador de modelos.

RSx también permite seleccionar un paquete UML de un modelo existente y “crear un modelo” basándose en dicho paquete. Puede considerarlo como una “poda” del paquete para crear un archivo de creación de modelos nuevo. El resultado es que en la vista del explorador de modelos, el paquete aparece como un nuevo modelo lógico de nivel superior. La visibilidad de la estructura que contiene la jerarquía original se pierde, pero siempre que abre un archivo de creación de modelos a partir del cual se han “podado” de este modo los paquetes, se abrirán también todos los modelos “podados”. Como resultado de esto pueden producirse extracciones innecesarias y se puede congestionar la vista del explorador de modelos con la aparición de un gran número de pestañas en el panel del editor que dificultan la navegación.

Además del soporte de subunidades de las versiones 7.0 de RSx, también puede particionar modelos en varios archivos sin perder la visibilidad de la estructura jerárquica y junto con la desaparición de las pestañas del “editor de modelos” en el panel de editor, se solucionan otros problemas de navegación.

No obstante, la experiencia de las versiones 6.x anteriores proporciona una lección en relación con la estructura de modelos que continúa siendo aplicable. Como se ha indicado anteriormente, al abrir un archivo de creación de modelos del que se han podado los paquetes se abren todos los modelos podados. Esto se puede evitar planificando una estrategia de partición previa, en lugar de utilizar la característica de “creación de un modelo”.

Los que siguen la característica de “creación de un modelo” generalmente comienzan con un único archivo de creación de modelos y crean paquetes para representar la organización de la arquitectura de la solución. Por ejemplo, es posible que exista un paquete para cada componente principal o subsistema de la solución (donde "componente" y "subsistema" reflejan el uso informal de estos términos que se daba en los primeros tiempos de la informática de sistemas y no sus definiciones semánticas formales en UML). Es posible que también haya paquetes para componentes “comunes” o de “programa de utilidad” o de “infraestructura”. A medida que crece dicho modelo, los paquetes correspondientes a los componentes específicos de la aplicación se pueden “podar” como archivos de creación de modelos individuales, de modo que los equipos que creen esos componentes puedan(teóricamente) trabajar con estos archivos de modelos de forma aislada. Pero en la práctica, si se abre uno de estos archivos de modelo específicos de un componente se abre el modelo "maestro" original (en el que residen las piezas "comunes") y, a su vez, se abren todos los otros modelos específicos de los componentes de la aplicación. Como resultados se producen las extracciones innecesarias mencionadas anteriormente y se dificulta la navegación.

Un método mejor es planificar con antelación la definición de un modelo diferente para cada componente específico de la aplicación, junto con un modelo para los componentes "comunes", o quizás, varios modelos que definan capas sucesivas de componentes comunes/reutilizables que, por ejemplo, reflejen capas de abstracción de la implementación de la arquitectura. De este modo, el equipo propietario podrá abrir cualquier modelo de componente específico de la aplicación y sólo se tendrán que abrir también los modelos "comunes" de los que depende el modelo específico de la aplicación.