Modelización con Diagramas de Casos de Uso

Maestrando: Ing. Javier Nader

1- Introducción

En la perspectiva Orientada a Objetos el principal bloque de todos los sistemas software es el objeto o clase. Por ejemplo, considérese una arquitectura sencilla de tres capas para un sistema de contabilidad, que involucre una interfaz de usuario, una capa intermedia y una base de datos. En la interfaz del usuario aparecen objetos tales como botones, menúes, y cuadros de diálogos. En la base de datos aparecen objetos concretos como tablas que representan entidades del dominio del problema. En la capa intermedia aparecen objetos como transacciones y reglas del negocio, así como vistas de más alto nivel de las entidades del problema.

Visualizar, especificar, construir y documentar sistemas desde la perspectiva orientada a objetos se puede realizar con el Lenguaje Unificado de Modelado (UML).

En este documento se presenta uno de los tantos diagramas del UML, *Diagramas de Casos de Uso.* Se introduce en la terminología de estos diagramas, como construirlos, sus características principales, ejemplos y se propone una herramienta CASE para automatizar su construcción. Los ejemplos mostrados son a efectos de ejemplificar en términos generales las características principales y la forma de los diagramas, ya que el presente documento no pretende mostrar una modelización completa sino introducir los conceptos principales de los diagramas de casos de uso.

Para presentar los conceptos principales se utilizaran ejemplos en donde se verá como se puede modelar un diagrama de contexto y un diagrama de requisitos utilizando los casos de uso. Estos modelos están basados en una descripción en donde se puede observar la necesidad de automatización de una biblioteca que pertenece a una universidad. En dicha automatización se destacarán ciertas necesidades como préstamos, devoluciones y mantenimiento de los libros entre otros.

2- Casos de Uso

Ningún sistema se encuentra aislado. Cualquier sistema interactúa con actores humanos o mecánicos que lo utilizan con algún objetivo y que esperan que el sistema funcione de forma predecible.

Los casos de uso bien estructurados denotan solo comportamientos esenciales del sistema o de un subsistema, y nunca deben ser excesivamente genéricos ni demasiado específicos.

2.1- Introducción a los Casos de Uso

Un factor clave al definir casos de uso es que no se debe especificar como implementarlos. Por ejemplo, se puede especificar como debería comportarse un Sistema de Gestión de una Biblioteca mediante casos de uso y como interactúan los usuarios con el sistema. Los casos de uso especifican un comportamiento deseado, no imponen como se llevara a cabo ese comportamiento. Lo más importante es que se permite que los usuarios finales y los expertos del dominio se comuniquen con lo desarrolladores sin entrar en detalles. Estos detalles llegaran, pero los casos de uso permiten centrarse en las cuestiones más importantes para el usuario final.

En UML, todos esos comportamientos se modelan como casos de uso que pueden especificarse independientemente de su realización. Un caso de uso es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable, de valor para un actor.

Un caso de uso describe un conjunto de secuencias, donde cada secuencia representa la interacción de los elementos externos al sistema (sus actores) con el propio sistema (y sus abstracciones clave). En realidad, estos comportamientos son funciones a nivel del sistema que se utilizan durante la captura de requisitos y el análisis para visualizar, especificar, construir y documentar el comportamiento del sistema. Por ejemplo, un caso de uso fundamental en el Sistema de Gestión de Bibliotecas es el procesamiento de Préstamos de libros.

Un caso de uso involucra la interacción de actores y el sistema. Un actor representa un conjunto coherente de roles que juegan los usuarios de los casos de uso al interactuar con estos. Los actores pueden ser personas o pueden ser sistemas mecánicos. Por ejemplo, en el modelado de la Biblioteca, el procesamiento de un préstamo (modelado como caso de uso) implica, entre otras cosas, la interacción con un responsable o encargado de la biblioteca (modelado como un actor).

Un caso de uso puede tener variantes. En cualquier sistema se pueden encontrar casos de uso que son versiones especializadas de otros casos de uso, casos de uso incluidos como parte de otros, y casos que extienden el comportamiento de otros casos de uso básicos. Se puede factorizar el comportamiento común y reutilizable de un conjunto de casos de uso organizándolos según estos tres tipos de relaciones. Por ejemplo, cuando se modela una biblioteca aparecen muchas variaciones del caso de uso básico de procesar un préstamo, tales como las diferencias entre procesar un préstamo de un libro frente a un préstamo que involucre gran cantidad de libros. En cada

opción, sin embargo, estos casos de uso comparten algo de comportamiento, como el caso de uso de aprobar el préstamo para muchos libros, un comportamiento que es parte del procesamiento de cualquier tipo de préstamo.

Los casos de uso se pueden aplicar al sistema completo. También se pueden aplicar a partes del sistema, incluyendo subsistemas e incluso clases e interfaces individuales. En cada caso, estos casos de uso no solo representan el comportamiento esperado de estos elementos, sino que también pueden utiizarse como la base para establecer casos de pruebas para estos elementos mientras evolucionan durante el desarrollo del sistema. Los casos de uso, aplicados al sistema completo son una fuente excelente de pruebas del sistema y de integración. UML proporciona una representación gráfica de un caso de uso y un actor, como se muestra en la figura 1. Esta notación permite visualizar un caso de uso independientemente de su implementación y en un contexto con otros caso de uso.

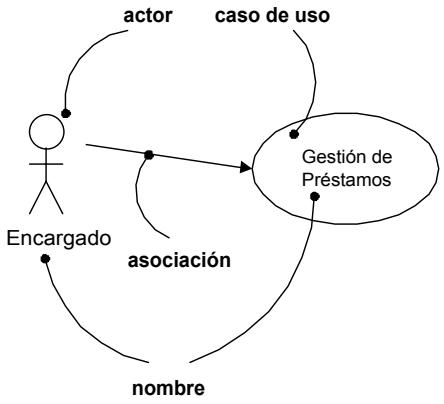

Figura 1. Actor, Asociación y Caso de Uso.

2.2- Términos y Conceptos

2.2.1- Nombres

Gráficamente, un caso de uso se representa como una elipse. Cada caso de uso debe tener un nombre que lo distinga de otros casos de uso. Un nombre es una cadena de texto. Ese nombre solo se llama nombre simple; un nombre de camino consta del nombre del caso de uso precedido del nombre del paquete (se denomina paquete a un mecanismo de propósito general para organizar elementos en grupos) en el que se encuentra. Normalmente, un caso de uso se dibuja mostrando solo su nombre, como se muestra en la figura 2.

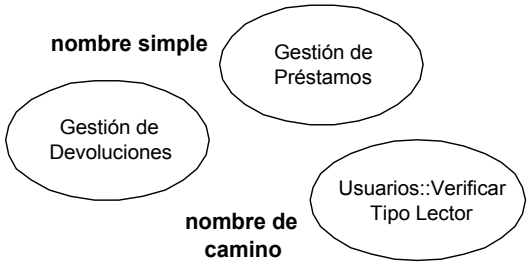

Figura 2. Nombres simples y de camino.

2.2.2- Casos de Uso y Actores

Un actor representa un conjunto coherente de roles que los usuarios de los casos de uso juegan al interactuar con estos. Normalmente, un actor representa un rol que es jugado por una persona, un dispositivo hardware o incluso otro sistema al interactuar con nuestro sistema.

Como se muestra en la figura 3, los actores se representan como monigotes. Se pueden definir categorías generales de actores (como Lector) y especializarlos (como Estudiante) a través de relaciones de generalización.

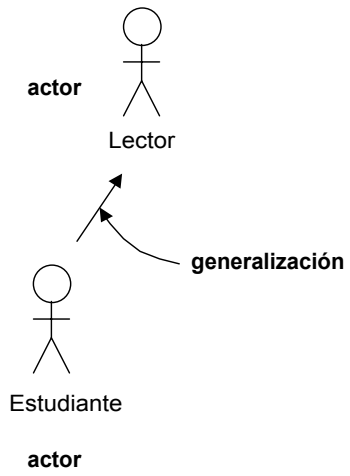

Figura 3. Actores

Los actores sólo pueden conectarse a los casos de uso a través de asociaciones (ver figura 1). Una asociación entre un actor y un caso de uso

indica que el actor y el caso de uso se comunican entre sí, y cada uno puede enviar y recibir mensajes.

2.2.3- Casos de Uso y Flujo de Eventos

El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos de forma textual, lo suficientemente claro para que alguien ajeno al sistema lo entienda fácilmente. Cuando se describe este flujo de eventos se debe incluir como y cuando empieza y acaba el caso de uso, cuando interactúa con los actores y que objetos se intercambian, el flujo básico y los flujos alternativos del comportamiento.
Por ejemplo, en el contexto de la biblioteca, se podría describir el caso de uso Gestión de Validación Tipo de Usuario de la siguiente forma:

Flujo de eventos principal: El caso de uso comienza cuando el sistema pide al usuario el Número de Identificación o Nombre de usuario. El usuario puede introducir su identificación a través del teclado. El usuario acepta la entrada pulsando el botón Enter. El sistema comprueba entonces este número o nombre para ver si es válido y a que tipo de lector pertenece. Si el número o nombre es válido, el sistema acepta la entrada y así finaliza el caso de uso.

También hay Flujos de Eventos Excepcionales como por ejemplo cuando el usuario puede borrar el número ingresado en cualquier momento antes de introducirlo y volver a teclear un nuevo número.

El flujo de eventos se puede especificar de muchas maneras, incluyendo texto informal, texto estructurado formal (pre y post condiciones) y pseudocódigo.

2.2.4- Casos de Uso y Escenarios

Normalmente, primero se describe el flujo de eventos de un caso de uso mediante texto. Sin embargo, conforme se mejora la comprensión de los requisitos del sistema estos flujos se pueden especificar gráficamente mediante diagramas de interacción. Normalmente, se usa un diagrama de secuencia para especificar el flujo principal de un caso de uso, y se usan variaciones de ese diagrama para especificar los flujos excepcionales del caso de uso.
Conviene separar el flujo principal de los flujos alternativos, porque un caso de uso describe un conjunto de secuencias, no una única secuencia, y sería imposible expresar todos los detalles de un caso de uso no trivial en una única secuencia. Cada secuencia se denomina escenario. Un escenario es una secuencia específica de acciones que ilustra un comportamiento. Los escenarios son a los casos de uso lo que las instancias a las clases, es decir, un escenario es básicamente una instancia de un caso de uso.

Se omitirán ejemplos de escenarios ya escapan al alcance de este documento. Para obtener más información de como modelar esta característica consultar la bibliografía especificada.

2.2.5- Casos de Uso y Colaboraciones

El caso de uso debe implementarse y esto se hace creando una sociedad de clases y otros elementos que colaboran para llevar a cabo el comportamiento del caso de uso. Esta sociedad de elementos, incluyendo tanto su estructura estática como dinámica, se modela en UML como una colaboración.
Como muestra la figura 4, la realización de un caso de uso puede especificarse explícitamente mediante una colaboración. Pero aunque la mayoría de las veces, un caso de uso es realizado exactamente por una colaboración, no será necesario mostrar explícitamente esta relación.

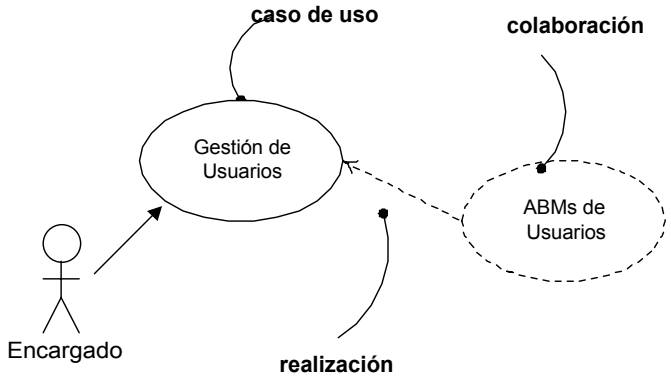

Figura 4. Casos de Uso y Colaboraciones

El objetivo de la arquitectura de un sistema es encontrar el conjunto mínimo de colaboraciones bien estructuradas que satisfagan el flujo de eventos especificado en todos los caso de uso del sistema.

2.3- Organización de los Casos de Uso

Los casos de uso pueden organizarse agrupándolos en paquetes, de la misma manera que se organizan las clases. Los casos de uso también pueden organizarse especificando relaciones de generalización, inclusión y extensión entre ellos. Estas relaciones se utilizan para factorizar el comportamiento común (extrayendo ese comportamiento de los casos de uso en los que se incluye) y para factorizar variantes (poniendo ese comportamiento en otros casos de uso que lo extienden).

La generalización entre casos de uso es como la generalización entre clases. Aquí significa que el caso de uso de un hijo hereda el comportamiento y significado del caso de uso padre; el hijo puede añadir o redefinir el comportamiento del padre; el hijo puede ser colocado en cualquier lugar donde

aparezca el padre (tanto el padre como el hijo pueden tener instancias concretas). Por ejemplo, en el sistema de biblioteca puede tenerse el caso de uso Gestión de Préstamos, responsable del manejo de los préstamos de libros. Además, podría haber dos hijos especializados de este caso de uso (Gestión de Préstamos a Investigadores y Gestión de Préstamos a Estudiantes), los cuales se comportarían como Gestión de Préstamos, aunque ambos tengan su propio comportamiento. Como se muestra en la figura 5, la generalización entre casos de uso se representa con una línea continua con una punta de flecha.
Una relación de inclusión entre casos de uso significa que un caso de uso base incorpora explícitamente el comportamiento de otro caso de uso en el lugar especificado en el caso base. El caso de uso incluido nunca se encuentra aislado, sino que es instanciado sólo como parte de algún caso de uso más amplio que lo incluye. La inclusión puede verse como que el caso de uso base toma el comportamiento del caso de uso proveedor.

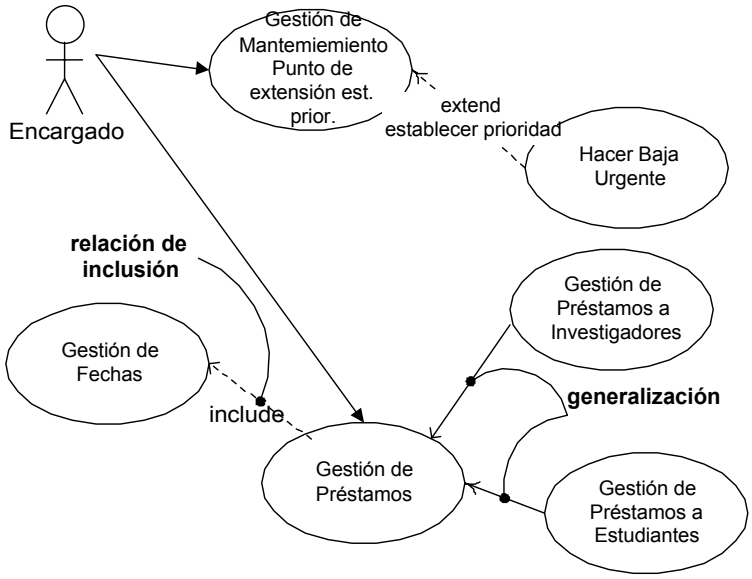

Figura 5. Generalización, inclusión y extensión

La relación de inclusión se usa para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte (que será incluido por un caso de uso base). La relación de inclusión es básicamente un ejemplo de delegación; se toma un conjunto de responsabilidades del sistema y se capturan en un sitio (el caso de uso a incluir en otros), a continuación se permite que otras partes del sistema (otros casos de uso) incluyan la nueva agregación de responsabilidades, siempre que se necesite usar esa funcionalidad.

Una relación de inclusión se representa como una dependencia, estereotipada con include. Para especificar la posición en un flujo de eventos en la cual el

caso de uso base incluye el comportamiento de otro caso de uso, simplemente se debe escribir include seguido del nombre del caso de uso que se quiere incluir, como en el siguiente flujo para Gestión de Préstamos.

Flujo de eventos Principal: Obtener y Verificar el libro pedido. include (Gestión de fechas*)*. Verificar que el libro pedido esté disponible.

Una relación de extensión entre casos de uso significa que un caso de uso base incorpora implícitamente el comportamiento de otro caso de uso en el lugar especificado por el caso de uso que extiende al base. El caso de uso base puede estar aislado pero, bajo ciertas condiciones, su comportamiento puede extenderse con el comportamiento del otro caso de uso. Este caso de uso base puede extenderse solo en ciertos puntos, llamados, puntos de extensión. La extensión se puede ver como que el caso de uso que extiende incorpora su comportamiento en el caso de uso base.

Una relación de extensión se utiliza para modelar la parte de un caso de uso que el usuario puede ver como comportamiento opcional del sistema. De esta forma, se separa el comportamiento opcional del obligatorio. También se puede utilizar una relación de extensión para modelar el subflujo separado que se ejecuta solo bajo ciertas condiciones. Por último, se puede utilizar una relación de extensión para modelar varios flujos que se pueden insertar en un punto dado, controlados por la interacción explícita con un actor.
Como ejemplo supongamos que existe un caso de uso que se denomina Hacer Baja Urgente y que realiza la baja de un libro en forma urgente aunque el libro lo tenga un investigador.

Una relación de extensión se representa como una dependencia, estereotipada como extend. Los puntos de extensión del caso de uso base se pueden listar en un comportamiento extra. Estos puntos de extensión solo son etiquetas que pueden aparecer en el flujo del caso de uso base.
Un caso de uso puede tener mas de un punto de extensión, y estos siempre se identifican por el nombre. Si hay varios puntos de extensión, el caso de uso que extiende al caso base simplemente mezclará sus flujos en orden.

3- Diagramas de Casos de Uso

Los diagramas de casos de uso son unos de los cinco tipos de diagramas de UML que se utilizan para el modelado de los aspectos dinámicos de un sistema (los otros cuatros tipos son los diagramas de actividades, de estados, de secuencia y de colaboración). Los diagramas de casos de uso son importantes para modelar el comportamiento de un sistema, subsistema o una clase. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones.

Los diagramas de caso de uso son importantes para visualizar, especificar y documentar el comportamiento de un elemento. Estos diagramas facilitan que los sistemas, subsistemas o clases sean abordables y comprensibles, al

presentar una vista externa de cómo pueden utilizarse estos elementos en un contexto dado. Los diagramas de casos de uso también son importantes, para probar sistemas ejecutables a través de ingeniería directa y para comprender sistemas ejecutables a través de ingeniería inversa.

Con UML, los diagramas de casos de uso se emplean de forma que los usuarios puedan comprender como utilizar ese elemento y de forma que los desarrolladores puedan implementarlo.

Los diagramas de casos de uso pueden contener notas y restricciones. También pueden contener paquetes, que se emplean para agrupar elementos del modelo en partes mayores. Ocasionalmente, se pueden incluir instancias de casos de uso en los diagramas, especialmente cuando se quiera visualizar un sistema especifico en ejecución.

3.1- Usos Comunes

3.1.1- Modelado del Contexto del Sistema

En UML, se puede modelar el contexto de un sistema con un diagrama de casos de uso, destacando los actores en torno al sistema. La decisión sobre qué incluir como un actor es importante, porque al hacer eso se especifica un tipo de cosas que interactúan con el sistema. La decisión sobre qué no incluir es igualmente importante, si no más, porque restringe el entorno para que sólo incluya a aquellos actores necesarios en la vida del sistema.

Para modelar el contexto de un sistema:

- Hay que identificar los actores en torno al sistema, considerando qué grupos requieren ayuda del sistema para llevar a cabo sus tareas; que grupos son necesarios para ejecutar las funciones del sistema; que grupos interactúan con el hardware externo o con otros sistema software; y que grupos realizan funciones secundarias de administración y mantenimiento.

- Hay que organizar los actores similares en jerarquías de generalización/especialización.

- Hay que proporcionar un estereotipo para cada uno de esos actores, si así se ayuda a entender el sistema.

- Hay que introducir esos actores en un diagrama de casos de uso y especificar las vías de comunicación de cada actor con los casos de uso del sistema.

Como ejemplo, se muestra un diagrama de contexto de un Sistema de Gestión de Biblioteca, cuyo modelado se basa en el ejercicio de autocomprobación de la Unidad Nro. 8. del Módulo II de la Maestría en Ingeniería de Software. La descripción del problema es la siguiente:

La Universidad Z necesita automatizar gran parte de la gestión de su Biblioteca debido al gran número de libros que posee y de lectores que utilizan sus servicios. Además en los últimos tiempos los lectores se han quejado de las dificultades que tienen a la hora de saber que libros de una materia están disponibles, o por el contrario, están prestados.

La biblioteca clasifica los libros como libros de texto, de referencia, y especializados. Los especializados a su vez pueden pertenecer a un departamento de la Universidad o a la Biblioteca. Los lectores de la Biblioteca, por otra parte, se clasifican en dos categorías: estudiantes e investigadores. Los estudiantes sólo pueden acceder a los libros de texto y de referencia. Los investigadores pueden acceder a todos los libros. El préstamo de los libros puede ser de sala o para sacarlos de la Biblioteca. Los libros de referencia no se pueden sacar de la biblioteca. Los libros de los departamentos tienen un plazo de devolución fijado por el departamento al que pertenece el libro. Para el resto el plazo es el mismo dentro de cada categoría, y está fijado por los responsables de la biblioteca. En cualquier caso el encargado de la Biblioteca es quien se ocupa del préstamo y devolución de los libros.

Se pretende que la aplicación funcione en un ordenador central al que se puedan conectar los lectores desde otros ordenadores para hacer consultas. Las consultas permitirán acceder a la información sobre los libros que estén disponibles y que posean una determinada palabra en su título, o que pertenezcan a un cierto autor. Por otra parte el encargado de la biblioteca se puede conectar también desde el terminal que posee en su lugar de trabajo para gestionar el préstamo y devolución de libros. Es de especial interés que la aplicación controle las fechas de préstamo y devolución. Además el encargado podrá dar de alta o de baja los libros que existan en la biblioteca.

Adicionalmente se ha decidido desarrollar una interfaz para la aplicación que sea fácil de reutilizar y modificar. La interfaz debe estar formada por los siguientes elementos: menúes, ventanas de diálogos y ventanas de salida. El menú permitirá al usuario elegir entre varios submenúes. Los submenúes a su vez estarán formados por opciones o ítems. Las diversas opciones permitirán ejecutar las funciones o acciones de la aplicación que utilice la interfaz. Las ventanas de diálogos servirán para que el usuario introduzca una información de entrada en la aplicación. Un diálogo tendrá una o más líneas de entrada para introducir información.
Las ventanas de salida permitirán mostrar resultados y mensajes a los usuarios. Se pueden tener abiertas varias ventanas, cada una de ellas identificada por un título.

La figura 6 muestra el diagrama de contexto del sistema de Gestión de Biblioteca, destacando los actores en torno al sistema. Se puede ver que existen Lectores y Encargado. Estos actores representan los roles que juegan las personas que interactúan con el sistema.

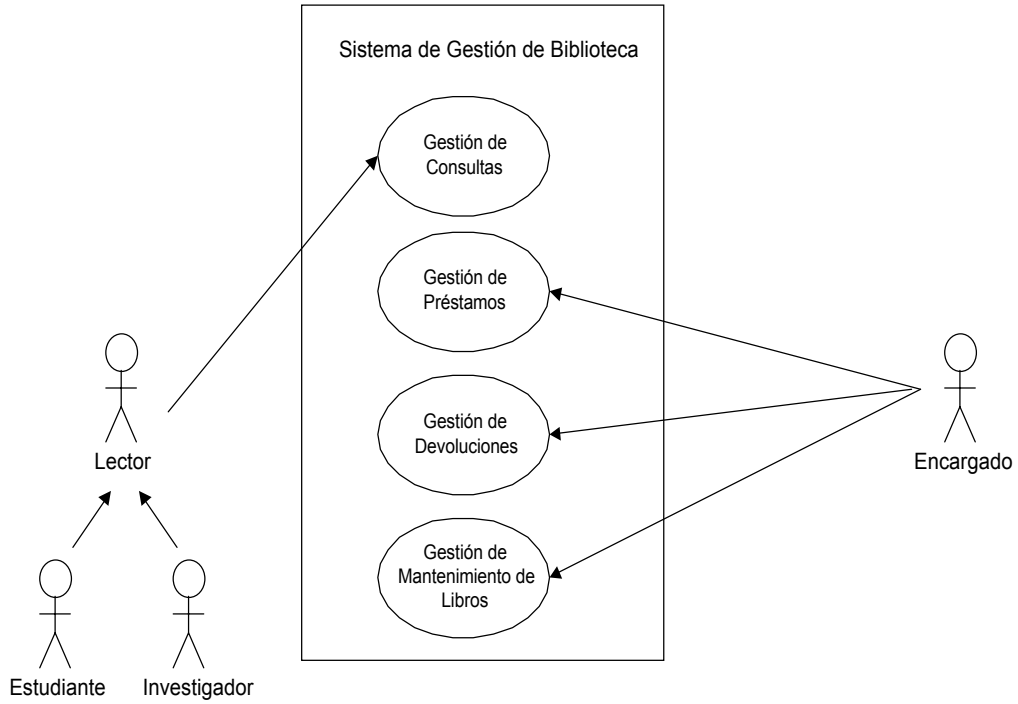

Figura 6. Diagrama de Modelado de contexto de un sistema.

Esta misma técnica se utiliza para el modelado del contexto de un subsistema. Un sistema de nivel dado de abstracción es a menudo un subsistema de un sistema mayor a un nivel superior de abstracción. Por lo tanto, el modelado del contexto de un subsistema es útil al construir sistemas de sistemas interconectados.

3.1.2- Modelado de los Requisitos de un Sistema

Los requisitos se pueden expresar de varias formas, desde texto sin estructura hasta expresiones en un lenguaje formal, y en cualquier otra forma intermedia. La mayoría de los requisitos funcionales de un sistema, si no todos, se pueden expresar con casos de uso, y los diagramas de casos de uso de UML son fundamentales para manejar esos requisitos.

Para modelar los requisitos de un sistema:

- Hay que identificar el contexto del sistema, identificando los actores a su alrededor (según Booch, G., Jacobson, I.y Rumbaugh, J.)

- Hay que considerar el comportamiento que cada actor espera del sistema o requiere que éste le proporcione.

- Hay que nombrar esos comportamientos comunes como casos de uso.

- Hay que factorizar el comportamiento común en nuevos casos de uso que puedan ser utilizados por otros; hay que factorizar el comportamiento variante en nuevos casos de uso que extiendan los flujos principales.

- Hay que modelar esos casos de uso, actores y relaciones en un diagrama de casos de uso.

- Hay que adornar esos casos de usos con notas que enuncien los requisitos no funcionales. Puede que haya que asociar varias de estás notas al sistema global.

La figura 7 muestra un diagrama de casos de uso. Aunque omite algunas de las relaciones entre los actores y los casos de uso como así también entre casos de uso debido a las limitaciones de tamaño y espacio de este documento, añade casos de usos adicionales que son invisibles para un usuario normal, aunque son comportamientos fundamentales del sistema, como por ejemplo, Gestión de Usuarios y Gestión de Validación de tipo de Usuario. Con la utilización de una Herramienta CASE para realizar el modelado se puede evitar las omisiones de relaciones entre objetos gracias a las facilidades de automatización de gráficos.

Como se explicó anteriormente comportamiento de un caso de uso se puede especificar describiéndolo de forma textual, se muestra algunas descripciones de los casos de uso más importantes para poder relacionar el enunciado del problema con el diagrama de requisitos. Las especificaciones de los casos de usos se pueden completar con un flujo de eventos detallado para una mejor descripción, comprensión y refinamiento.

Especificación de Gestión de Consultas: El sistema debe permitir a los lectores acceder a la información sobre los libros que estén disponibles y que posean una determinada palabra en su título, o que pertenezcan a un cierto autor.

Especificación de Gestión de Mantenimiento de Libros: El sistema debe permitir al encargado de la biblioteca poder introducir información para gestionar el mantenimiento de los libros. Dicha información debe tener en cuenta que biblioteca clasifica los libros como libros de texto, de referencia, y especializados. Los especializados a su vez pueden pertenecer a un departamento de la Universidad o a la Biblioteca. El préstamo de los libros puede ser de sala o para sacarlos de la Biblioteca. Los libros de referencia no se pueden sacar de la biblioteca. Los libros de los departamentos tienen un plazo de devolución fijado por el departamento al que pertenece el libro. Para el resto el plazo es el mismo dentro de cada categoría.
El usuario encargado debe poder dar de altas y bajas de libros.

Especificación de Gestión de Interfaz de Usuario: El sistema debe proveer una interfaz de usuario que debe ser reutilizable y fácil de usar. Para esto, la interfaz debe estar formada por los siguientes elementos: menúes, ventanas de diálogos y ventanas de salida. El menú permitirá al usuario elegir entre varios submenúes. Los submenúes a su vez estarán formados por opciones o ítems. Las diversas opciones permitirán ejecutar las funciones o acciones de la aplicación que utilice la interfaz. Las ventanas de diálogos servirán para que el usuario introduzca una información de entrada en la aplicación. Un diálogo tendrá una o más líneas de entrada para introducir información.
Las ventanas de salida permitirán mostrar resultados y mensajes a los usuarios. Se pueden tener abiertas varias ventanas, cada una de ellas identificada por un título.

Especificación de Gestión de Validación Tipo de Usuario: El sistema debe poder validar e identificar el tipo de usuario que se conecta al sistema. Los tipos de usuarios son: Lectores y Encargados de biblioteca. Los lectores de la biblioteca, por otra parte, se clasifican en dos categorías: estudiantes e investigadores.

Especificación de Gestión de Usuarios: El sistema debe poder gestionar las altas, bajas y modificaciones de usuarios teniendo en cuenta las características de estos. La gestión de usuarios la realiza el usuario encargado de la biblioteca.

Especificación de Gestión de Préstamos: El sistema debe poder gestionar los préstamos de los libros teniendo en cuenta las características de los libros y de los lectores. Además, es de especial interés que la aplicación controle las fechas del préstamo. La gestión de préstamos debe ser realizada por el usuario encargado de la biblioteca.

Especificación de Gestión de Devoluciones: El sistema debe poder gestionar las devoluciones teniendo en cuenta controlar las fechas del préstamo y devolución. La gestión de devoluciones debe ser realizada por el usuario encargado de la biblioteca.

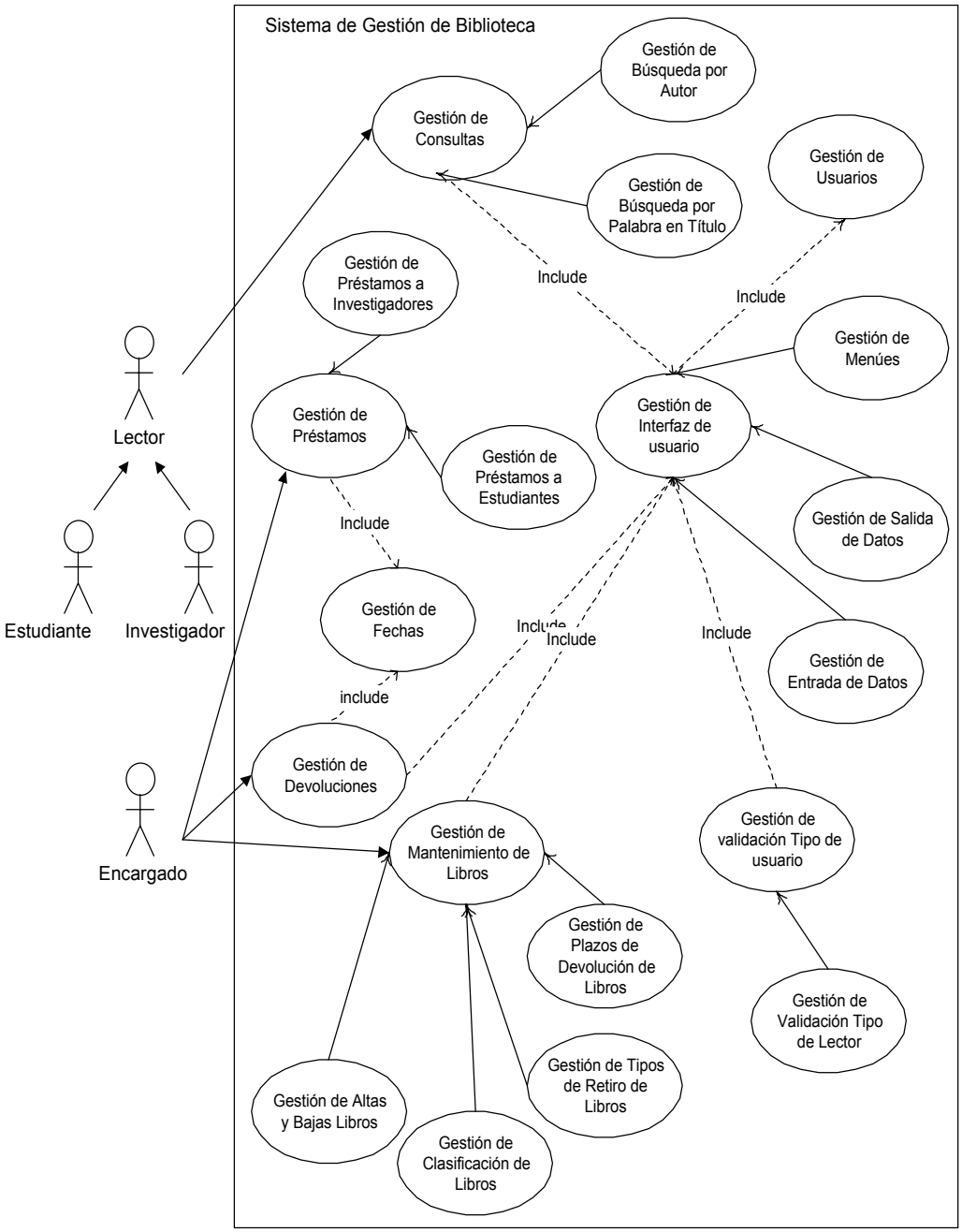

Figura 7. Diagrama de Modelado de Requisitos

Las figuras 6 y 7 son modelos preliminares, a los cuales se le pueden aplicar todos los conceptos enunciados en este documento para completarlos y refinarlos.

3.1.3 Herramienta de Modelado Visual Orientado a Objetos: Rational Rose.

La modelización de estos diagramas es conveniente que sea soportada por una herramienta CASE, como por ejemplo, ROSE de RATIONAL. Esta herramienta soporta el enfoque y perspectiva de UML. Con RATIONAL ROSE se pueden utilizar los diagramas para visualizar el sistema desde diferentes perspectivas.

Estos diagramas pueden tener cualquier combinación de elementos y relaciones. Hay que tener en cuenta que en la práctica solo surge un pequeño número de combinaciones, las cuales son consistentes con las vistas más útiles que comprenden la arquitectura de un sistema con gran cantidad de software. RATIONAL ROSE incluye, entre otros, los siguientes diagramas:

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Casos de Usos
- Diagrama de Secuencia
- Diagrama de Colaboración
- Diagrama de Estados
- Diagrama de Actividades
- Diagrama de Componentes
- Diagrama de Despliegue

Además de prestar soporte a la notación de la metodología UML, la herramienta ofrece también modelar con la notación de OMT y BOOCH .

Rational Rose es la herramienta visual en análisis orientado a objetos, modelización, diseño y construcción de aplicaciones. Usando una herramienta del tipo Rational Rose el equipo de desarrollo puede comunicar efectivamente la arquitectura del software por representación gráfica, documentar y generar el código de la aplicación.

Rational Rose es una herramienta de modelado visual que es usada en la automatización del desarrollo basado en componentes, la figura 8 muestra la ubicación de la herramienta (aparece sombreada en la figura) en la arquitectura de trabajo de Rational propuesta por Booch, Jacobson y Rumbaugh:

Figura 8. Ubicación de la Herramienta.

Con Rational Rose se pueden realizar y utilizar todos los beneficios de la modelización visual, incluyendo:

- Ciclos de desarrollos cortos por medio de un desarrollo interactivo controlado.
- Incremento de la productividad por medio del desarrollo " conducido por el modelo " y eliminación del riesgo.
- Soporte para proyectos de gran tamaño y organizaciones con múltiples proyectos por medio de un ambiente escalable.
- Significante reutilización del software a través de software basado en arquitectura y componentes.
- Mejora la comunicación del equipo de trabajo por medio del lenguaje de modelización.
- Integración con sistema heredados por medio de las capacidades de ingeniería reversa y extensibilidad de interfaces.
- Soporte de generación de código a lenguajes como: Java, C++, Visual Basic, Power Builder, Smalltalk.
- Soporte para definición de interfaces de Base de Datos.

La figura 9 muestra un diagrama de caso de uso realizado con la herramienta Rational Rose. Donde se puede observar la ventana principal de la herramienta, que se divide en dos vistas: uno muestra la lista de objetos creados, a la izquierda y la otra, a la derecha muestra el objeto en que se está trabajando, en este caso se muestra el diagrama de caso de uso Gestión de Biblioteca.
Se esta trabajando sobre la vista de casos de uso, en la cual se han definido en el diagrama, cuyo nombre es Gestión de Biblioteca, cuatro actores: Lector, Estudiante e Investigador, estos dos últimos heredan de Lector y el restante es Encargado. Los casos de uso definidos son los identificados como Gestión de Consultas, Gestión de Préstamos, Gestión de Devoluciones y por último Gestión de Mantenimiento de Libros. Los actores se comunican con los casos de usos por medio de las asociaciones.

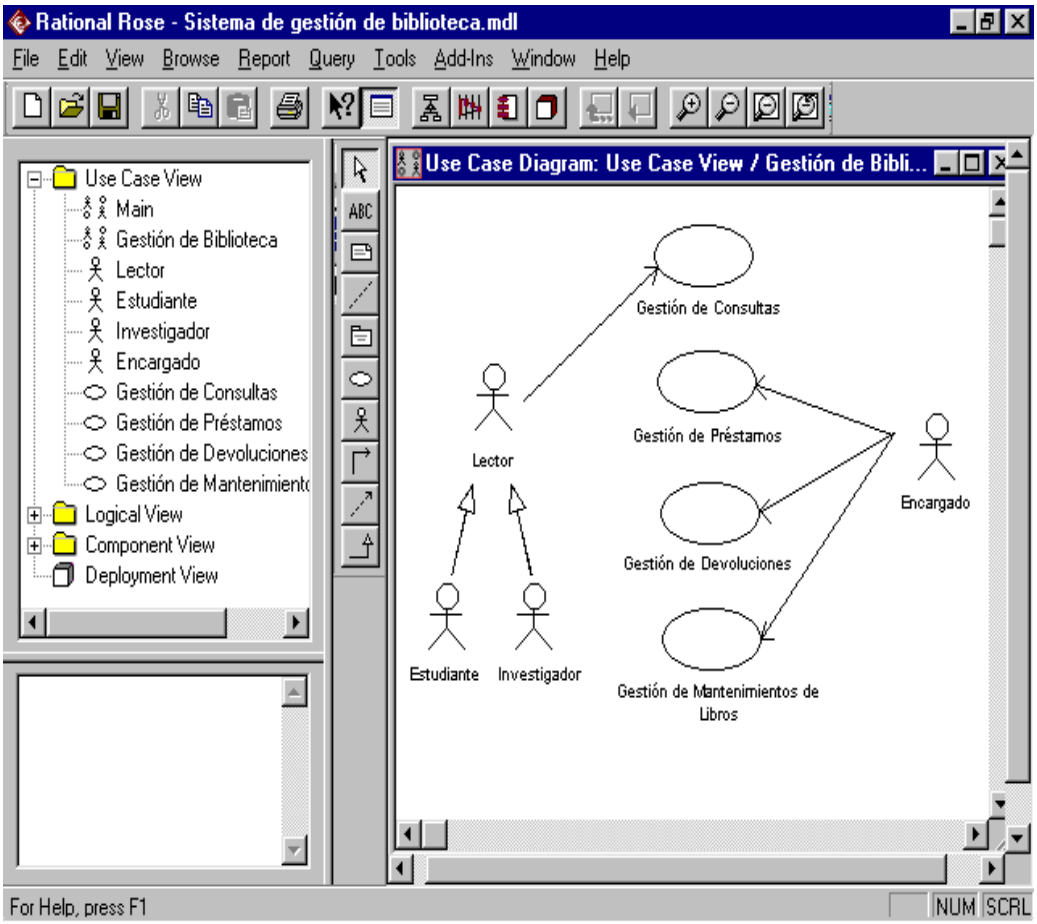

Figura 9. Rational Rose.

La figura 10 muestra una de las formas para agregar información a un caso de uso, para completar las especificaciones de actores y asociaciones la herramienta dispone de ventanas similares.

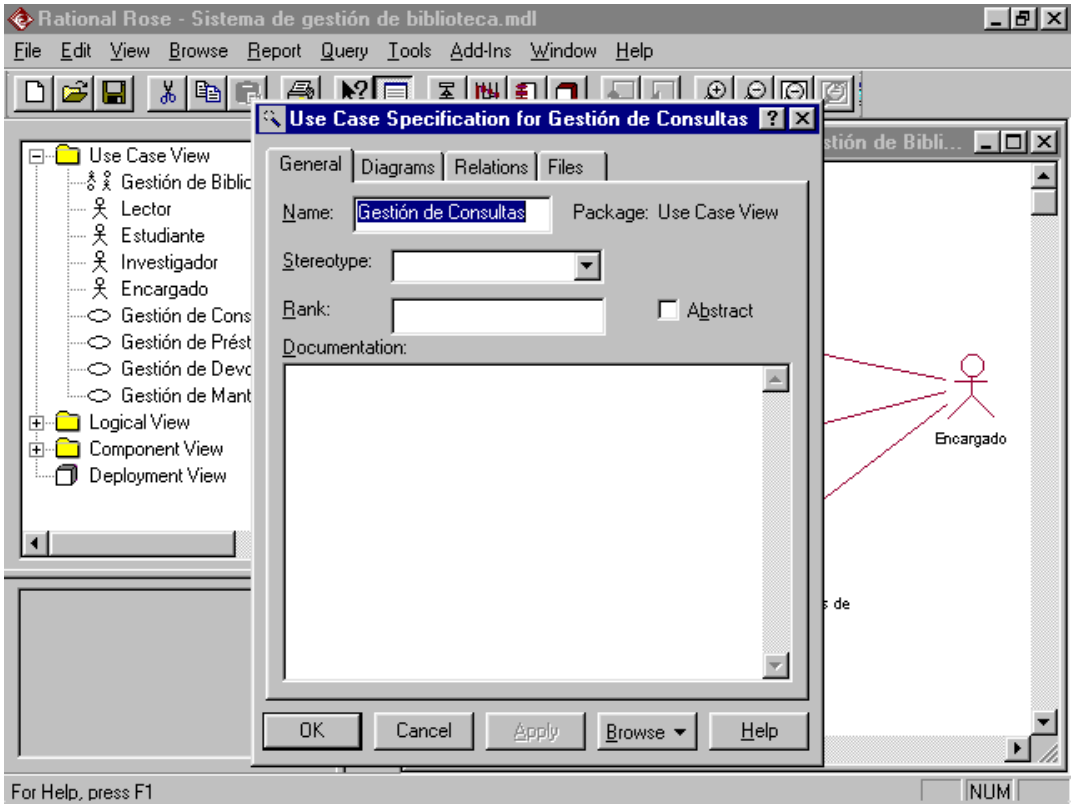

Figura 10. Ventana para completar una especificación.

Para obtener más detalle de la herramienta CASE RATIONAL ROSE consultar www.rational.com/ rose/.

4- Conclusiones

El presente documento solo muestra de forma introductoria uno de los diagramas de UML, *Diagramas de Casos de Uso*, se puede observar que los diagramas son fundamentales para el desarrollo de software con la metodología y perspectiva de UML, y aunque siendo esta independiente del proceso de construcción, para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. Así, los casos de uso se pueden utilizar para unificar los flujos de trabajo como se muestra en la figura 11:

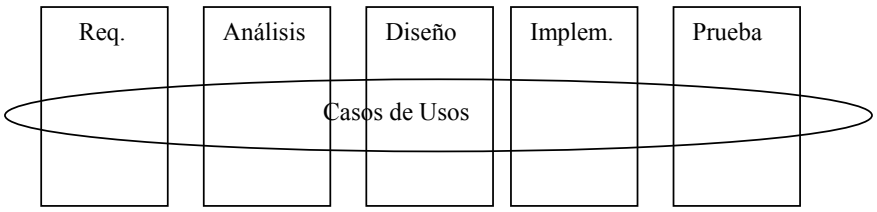

Figura 11. Los casos de uso vinculan el flujo de trabajo.

La visualización, especificación, construcción y documentación de un sistema con gran cantidad de software requiere que el sistema sea visto desde varias perspectivas. Diferentes usuarios (usuarios finales, analistas, desarrolladores, integradores, testeadores, documentadores y líderes de proyecto) siguen diferentes agendas en relación al proyecto, y cada uno mira a ese sistema de formas diferentes en diversos momentos a lo largo de la vida del proyecto. La arquitectura de un sistema es quizás el concepto más importante que puede emplearse para manejar estos diferentes puntos de vista y controlar el desarrollo iterativo e incremental de un sistema a lo largo de su ciclo de vida.

La arquitectura no tiene que ver solamente con la estructura y el comportamiento, sino también con el uso, la funcionalidad, el rendimiento, la capacidad de adaptación, la reutilización, la capacidad de ser comprendido, las restricciones económicas y de tecnología y los compromisos entre alternativas, así como los aspectos estéticos.

En los sistemas Orientados a Objetos, se intenta conectar todas las vistas casi independientes de un sistema en un todo semántico. Con los casos de uso se comienza a construir un modelo que conectará el flujo de trabajo de todo el ciclo de vida del proyecto, ya que es posible conectar las diferentes fases del ciclo de vida.

Si bien los conceptos presentados no presentan mayor complejidad, el uso de estos diagramas no debe ser aislado sino deben ser utilizados como un todo en el proceso de construcción del software conectados a los diferentes diagramas como los de Clases, de Objetos, de Secuencia,
de Colaboración, de Estados, etc. con lo que se logrará lo mencionado en el párrafo anterior.

Aprender a aplicar de forma eficaz y correcta UML y los casos de uso requiere aprender el modelo conceptual del lenguaje, que agrupa tres elementos principales: los bloques básicos de construcción UML, las reglas que dictan como pueden combinarse esos bloques y algunos mecanismos comunes que se aplican a lo largo del lenguaje.

Como podemos ver, el ejemplo presentado es básico, pero su construcción completa y correcta es compleja sin el uso de una herramienta CASE, que es conveniente para darnos soporte y automatización en visualizar, especificar, construir y documentar de manera correcta los sistemas, ya que ella nos guía en la metodología.

Bibliografía

- Booch, G., Jacobson, I., Rumbaugh, J., "The Unified Modeling Languaje User Guide". Addison-Wesley Publishing Company, 1999.

- Booch, G., "Object-Oriented Analisis and Design with Applications, 2 ed.", Addison-Wesley Publishing Company, Redwood City, California, 1993.

- Jacobson, I.; Chirsterton, M., P. y Overgaard, G., "Object Oriented Software Engineering: A Use Case Driven Aproach", Wokingham, England, Addison-Wesley Publishing Company, 1992.

- Rumbaugh, J., Blaba, M., Premerlani, W., Eddy, F., and Lorensen, W. "Object Oriented Modeling and Design", Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

- www.rational.com/rose/

- Herramienta CASE Rational Rose 98. Rational Software Corporation.