1. INTRODUCCIÓN AL UML

1.1. INTRODUCCIÓN

Como la guerra de métodos de modelado de objetos no hacía progresar la tecnología de objetos, Rumbaugh y Booch decidieron unificar sus trabajos en un método único: el método unificado (*The Unified Method*). La primera versión de la descripción del método unificado fue presentada en octubre de 1995 en un documento titulado *Unified Method 0.8*. Un año después se les unió Jacobson y el método unificado se transformó en UML (*The Unified Modeling Language*). Asimismo durante el año 1996 se crea un consorcio de colaboradores para trabajar en la definición de la versión 1.0 de UML, que agrupa a grandes empresas del sector. De esta colaboración nace la descripción del UML versión 1.0 enviada el 17 de enero de 1997 al OMG (*Object Management Group*), constituyéndose en un estándar en el mes de julio del mismo año.

Las principales etapas en la definición de UML aparecen reflejadas en la Figura 1.1 de forma cronológica.

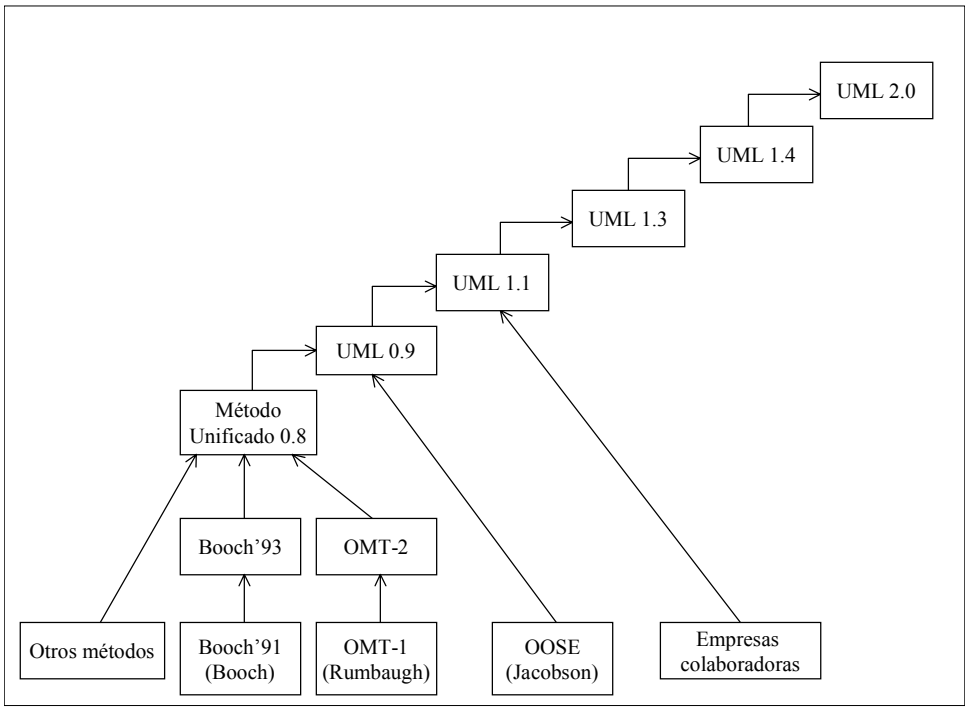

Figura 1.1. Historia de la definición del UML

Sus creadores insisten en el hecho de que UML es un lenguaje de modelado de objetos y no un método de objetos. La notación UML ha sido pensada para servir de lenguaje de modelado de objetos, independientemente del método implementado. Así pues, la notación UML puede subsistir a las notaciones de Booch, OMT (*Objetc Modeling Technique*) o incluso OOSE (*Object Oriented Software Engineering* también llamada *Objectory*).

1.2. MODELO CONCEPTUAL DEL UML

Para entender el UML, necesitamos constituir un modelo conceptual del lenguaje, y esto requiere aprender tres elementos importantes: los bloques de construcción básicos del

UML, las reglas que dictan cómo estos bloques se pueden poner juntos, y los mecanismos comunes que se aplican a lo largo del UML.

1.2.1. Bloques de construcción del UML

El vocabulario del UML engloba tres tipos de bloques de construcción:

1. Cosas
2. Relaciones
3. Diagramas

Las cosas son abstracciones del mundo real; las relaciones conectan estas cosas; los diagramas agrupan colecciones de cosas.

1.2.1.1. Cosas

Hay cuatro tipos de cosas en el UML:

1. Cosas de estructura

Las cosas estructurales son los nombres de los modelos UML. Éstas son mayormente las partes estáticas de un modelo, representando elementos que son bien conceptuales o bien físicos.

Los siguientes elementos – clases, interfaces, colaboraciones, casos de uso, componentes y nodos – son las cosas estructurales básicas que nos pueden ayudar en un modelo de UML. Hay algunas variaciones en estos elementos, tales como actores, señales y utilidades (tipos de clases).

2. Cosas de comportamiento

Las cosas de comportamiento son las partes dinámicas de los modelos del UML. Éstas son los verbos de un modelo, representando el comportamiento en el tiempo y en el espacio.

Los dos siguientes elementos – interacciones y máquinas de estado – son las cosas de comportamiento básicas que nos pueden ayudar en un modelo de UML. Semánticamente, estos elementos están conectados a varios elementos estructurales, principalmente clases, colaboraciones y objetos.

3. Cosas de agrupamiento

Las cosas de agrupamiento son las partes de organización de los modelos del UML. Éstas son las cajas en las que se puede descomponer un modelo.

Los paquetes son las cosas de agrupamiento básicas con que podemos organizar un modelo de UML. Hay algunas variaciones, tales como subsistemas (tipos de paquetes).

4. Cosas de anotación

Las cosas de anotación son las partes de explicación de los modelos del UML. Éstas son los comentarios que podemos aplicar para describir cualquier elemento en un modelo.

Las notas son las cosas de anotación básicas que podemos incluir en un modelo de UML. Normalmente utilizamos notas para adornar nuestros diagramas con restricciones o comentarios que son expresadas en texto formal o informal.

1.2.1.2. Relaciones

Hay tres tipos de relaciones en el UML:

1. Relaciones de asociación

Una asociación es una relación de estructura que describe un conjunto de enlaces, siendo un enlace una conexión entre objetos. La agregación es un tipo especial de asociación, representando una relación de estructura entre un todo y sus partes.

2. Relaciones de dependencia

Una dependencia es una relación semántica entre dos cosas en las que un cambio en una cosa (la cosa independiente) puede afectar a las semánticas de la otra cosa (la cosa dependiente). También hay variaciones, tales como *«includes»* y *«extends»*.

3. Relaciones de generalización

Una generalización es una relación de especialización/generalización en la que los objetos del elemento especializado (el hijo) son sustituibles por objetos del elemento generalizado (el padre). De esta forma, el hijo comparte la estructura y el comportamiento del padre.

1.2.1.3. Diagramas

Un diagrama es la presentación gráfica de un conjunto de elementos, la mayoría de las veces como si se tratara de un grafo conectado de vértices (cosas) y de arcos (relaciones). Nosotros dibujamos diagramas para visualizar un sistema desde diferentes perspectivas. En teoría, un diagrama puede contener cualquier combinación de cosas y relaciones. Sin embargo, en la práctica sólo aparece un número pequeño de combinaciones que sean consistentes con las cinco vistas comprendidas en la arquitectura de un sistema de software. La notación UML incluye nueve tipos de diagramas:

1. Diagrama de clase

Un diagrama de clase muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones. Este diagrama es el que más se encuentra en los sistemas de modelado orientado a objetos. Los diagramas de clase dirigen la visión de diseño estática de un sistema.

2. Diagrama de objeto

Un diagrama de objeto muestra un conjunto de objetos y sus relaciones. Este diagrama representa una fotografía estática de instancias de las cosas que se encuentran en un diagrama de clase. Los diagramas de objeto dirigen la visión de diseño estática o la visión de proceso estática de un sistema, al igual que los diagramas de clase, pero desde la perspectiva del mundo real.

3. Diagrama de caso de uso

Un diagrama de caso de uso muestra un conjunto de casos de uso y actores (un tipo especial de clase) y sus relaciones. Los diagramas de casos de uso dirigen la visión de caso de uso estática de un sistema. Estos diagramas son importantes a la hora de organizar y modelar los comportamientos de un sistema.

4. Diagrama de secuencia
5. Diagrama de colaboración

Estos diagramas son tipos de diagramas de interacción. Un diagrama de interacción muestra una interacción, que consiste de un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden enviarse entre ellos. Los diagramas de interacción dirigen la visión dinámica de un sistema.

Un diagrama de secuencia es un diagrama de interacción que enfatiza el orden de los mensajes en el tiempo. Un diagrama de colaboración es un diagrama de interacción que enfatiza la organización estructural de los objetos que envían y reciben mensajes. Los diagramas de secuencia y los diagramas de colaboración son isomórficos, es decir, se pueden transformar el uno en el otro.

6. Diagrama de estado

Un diagrama de estado muestra una máquina de estado, que consta de estados, transiciones, eventos, acciones y actividades. Los diagramas de estado dirigen la visión dinámica de un sistema. Estos diagramas son importantes a la hora de modelar el comportamiento de una interfaz, clase o colaboración, y enfatizan el comportamiento de un objeto ordenado por los eventos que se suceden, lo cual es especialmente útil en los sistemas de tiempo real.

7. Diagrama de actividad

Un diagrama de actividad es un tipo especial de diagrama de estado que muestra el flujo de actividad a actividad dentro de un sistema. Los diagramas de actividad dirigen la visión dinámica de un sistema. Estos diagramas son importantes a la hora de modelar la función de un sistema y enfatizan el flujo de control entre los objetos.

8. Diagrama de componente

Un diagrama de componente muestra las organizaciones y dependencias entre un conjunto de componentes. Los diagramas de componente dirigen la visión de implementación estática de un sistema. Estos diagramas se relacionan con los diagramas

de clase en el sentido de que un componente, normalmente, engloba a una o varias clases, interfaces o colaboraciones.

9. Diagrama de despliegue

Un diagrama de despliegue muestra la configuración de los nodos que se procesan en tiempo de ejecución y los componentes que están dentro de ellos. Los diagramas de despliegue dirigen la visión de despliegue estática de una arquitectura. Estos diagramas se relacionan con los diagramas de componente en el sentido de que un nodo encierra, normalmente, uno o más componentes.

1.2.2. Reglas del UML

Los bloques de construcción del UML no se pueden mostrar todos ellos juntos de una forma aleatoria. Al igual que cualquier otro lenguaje, el UML tiene una serie de reglas que especifican lo que un modelo bien formado debería contemplar. Un modelo bien formado es un modelo semánticamente consistente consigo mismo y en armonía con el resto de los modelos con los que se relaciona.

El UML posee reglas semánticas para:

- Nombres: ¿Cómo podemos llamar a las cosas, relaciones y diagramas?
- Ámbito: ¿Cuál es el contexto que da un significado específico a un nombre?
- Visibilidad: ¿Cuántos de esos nombres van a ser vistos y usados por otros?
- Integridad: ¿Cuántas cosas se relacionan unas con otras de manera apropiada y consistente?
- Ejecución: ¿Qué significa realmente ejecutar o simular un modelo dinámico?

1.2.3. Mecanismos comunes del UML

La construcción de los bloques del UML resulta más sencilla y más armoniosa, si se realiza de acuerdo a un patrón de características comunes.

En UML se aplican de forma consistente estos cuatro mecanismos comunes:

1. Especificaciones

Detrás de cada parte de la notación gráfica del UML hay una especificación que proporciona una declaración textual de la sintaxis y semánticas de dicho bloque de construcción. Por ejemplo, detrás de un icono de clase hay una especificación que proporciona el conjunto completo de atributos, de operaciones (incluyendo sus formatos completos) y de comportamientos que engloba dicha clase; visualmente, dicho icono de clase podría mostrar únicamente una pequeña parte de esta especificación.

2. Adornos

La mayoría de los elementos en el UML tienen una notación gráfica y directa que proporciona una representación visual de los aspectos más importantes del elemento. Por ejemplo, la notación de clase también expone los aspectos más importantes de una

clase, principalmente su nombre, sus atributos y sus operaciones. Pero una especificación de clase puede incluir otros detalles, tales como si es abstracta o la visibilidad de sus atributos y de sus operaciones. Muchos de estos detalles se pueden añadir como adornos textuales o gráficos dentro del rectángulo básico de la clase.

3. Divisiones comunes

A la hora de modelar sistemas orientados a objetos, el mundo aparece dividido como mínimo en un par de formas. Por ejemplo, está la división de clase y de objeto. Una clase es una abstracción, mientras que un objeto es una manifestación concreta de dicha abstracción en el UML. Podemos modelar tanto las clases como los objetos. Gráficamente, el UML distingue un objeto de una clase, ya que el objeto aparece con un formato algo diferente y además subrayado.

4. Mecanismos de extensión

El UML proporciona un lenguaje estándar para escribir modelos de software, pero no es posible para un lenguaje cerrado expresar todos los detalles de todos los modelos en todos los dominios a lo largo de todo el tiempo. Por esta razón, el UML es un lenguaje abierto, que se puede extender de forma controlada. Los mecanismos de extensión del UML incluyen:

- Estereotipos

Un estereotipo extiende el vocabulario del UML, permitiendo que podamos crear nuevos tipos de bloques de construcción que son derivados a partir de los ya existentes y que son específicos a nuestro problema. Se distinguen claramente ya que van encerrados entre los siguientes delimitadores: « ».

- Valores añadidos (*tagged values*)

Un valor añadido, en el sentido de una "coletilla", extiende las propiedades de un bloque de construcción del UML, permitiendo que podamos crear nueva información en la especificación de dicho elemento. En la práctica se utilizan muy poco.

- Restricciones

Una restricción extiende las semánticas de un bloque de construcción del UML, permitiendo que podamos añadir nuevas reglas o modificar las ya existentes. Se distinguen claramente ya que van encerrados entre los siguientes delimitadores: { }.

1.3. ARQUITECTURA

La visualización, especificación, construcción y documentación de un sistema de software demanda que dicho sistema pueda ser visto desde diferentes perspectivas. Las diferentes personas implicadas en el desarrollo de una aplicación, tales como usuarios finales, analistas, desarrolladores, directores de gestión del proyecto, etc., ven el sistema de diferentes formas en diferentes momentos a lo largo de la vida del proyecto. La arquitectura de un sistema es quizá la parte más importante que puede utilizarse para

gestionar los diferentes puntos de vista y controlar el desarrollo iterativo e incremental de un sistema a lo largo de su ciclo de vida.

La arquitectura de un sistema de software se describe mediante cinco vistas, las cuales se muestran en la Figura 1.2.

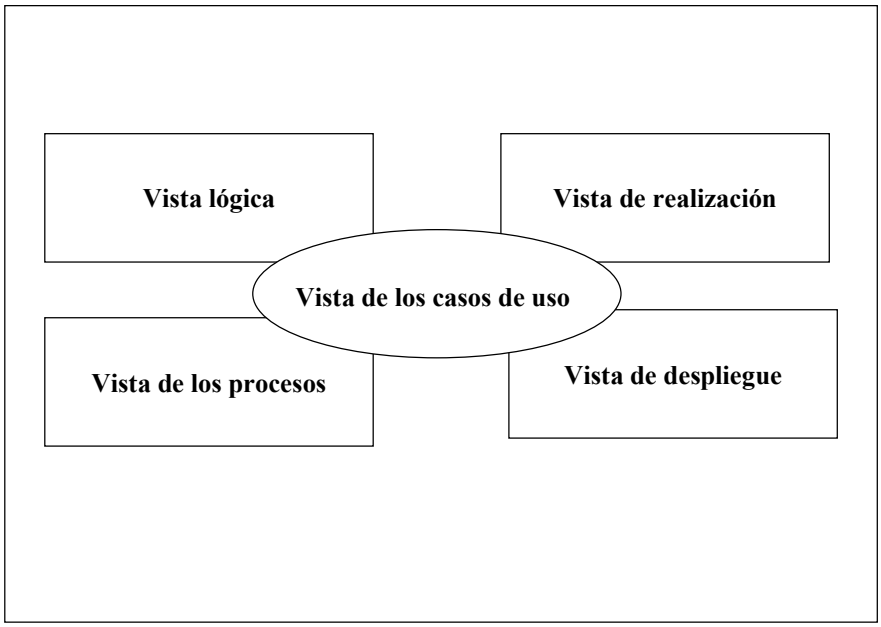

Figura 1.2. Representación del modelo de arquitectura de Philippe Krutchen

Esta representación de la arquitectura está basada en el modelo de Philippe Krutchen, denominado modelo de las 4 + 1 vistas, que son:

1. La vista lógica

La vista lógica describe los aspectos estáticos y dinámicos de un sistema en términos de clases y objetos y se concentra en la abstracción, el encapsulamiento y la uniformidad. El sistema se descompone en un juego de abstracciones, surgidas del ámbito del problema. Más allá de la satisfacción de las necesidades funcionales del usuario, la vista lógica permite identificar y generalizar los elementos y los mecanismos que constituyen las diferentes partes del sistema. La vista lógica trata los siguientes elementos de modelado:

- los objetos,
- las clases,
- las colaboraciones,
- las interacciones.

2. La vista de realización

La vista de realización se preocupa de la organización de los módulos en el entorno de desarrollo. Muestra la asignación de las clases en los módulos y la asignación de los módulos en los subsistemas. Los subsistemas se organizan en niveles jerárquicos con

las interfaces bien definidas. La vista de realización trata los siguientes elementos de modelado:
- los módulos,
- los subprogramas,
- las tareas,
- los subsistemas (paquetes estereotipados).

3. La vista de los procesos

La vista de los procesos representa la descomposición en flujos de ejecución (proceso, *threads*, tareas, etc.), la sincronización entre flujos y la asignación de los objetos y las clases dentro de los diferentes flujos. La vista de los procesos también se preocupa de la disponibilidad del sistema, de la fiabilidad de las aplicaciones y del rendimiento. La vista de los procesos manipula los siguientes elementos de modelado:
- las tareas, los *threads*, los procesos;
- las interacciones.

4. La vista de despliegue

La vista de despliegue describe los diferentes recursos de hardware y la implantación del programa en dichos recursos. La vista de despliegue cobra toda su importancia cuando el sistema es distribuido. La vista de despliegue manipula los siguientes elementos de modelado:
- los nodos,
- los módulos,
- los programas principales.

5. La vista de los casos de uso

La vista de los casos de uso forma el adhesivo que unifica las cuatro vistas anteriores. Los casos de uso motivan y justifican las opciones de arquitectura. Permiten identificar las interfaces críticas, fuerzan a los diseñadores a concentrarse sobre los problemas concretos, demuestran y validan las otras vistas de la arquitectura. La vista de los casos de uso se concentra en los siguientes elementos de modelado:
- los actores,
- los casos de uso,
- las clases,
- las colaboraciones.

1.4. CICLO DE VIDA DE DESARROLLO DE SOFTWARE

El UML es independiente del proceso, lo que significa que no está restringido a ningún ciclo de vida de desarrollo de software en particular. Sin embargo, para conseguir los mayores beneficios del UML, deberíamos considerar que un proceso sea:

- controlado por los casos de uso

Controlado por los casos de uso significa que los casos de uso se utilizan como mecanismos para crear el comportamiento deseado del sistema, para verificar y validar

la arquitectura del sistema, para realizar pruebas y para establecer la comunicación entre las personas implicadas en el proyecto.

- centrado sobre la arquitectura

Centrado sobre la arquitectura significa que la arquitectura de un sistema se utiliza como un artefacto para la concepción, construcción, gestión y evolución del sistema que se está desarrollando.

- iterativo e incremental

Un proceso iterativo es aquel que implica gestionar una sucesión continua de versiones ejecutables. Un proceso incremental es aquel que implica la integración continua de la arquitectura del sistema para producir estas versiones, con cada nueva versión se engloban mejoras incrementales con respecto a la otra. Además, un proceso iterativo e incremental es controlado por el riesgo, lo que significa que cada nueva versión se centra en acometer y reducir los riesgos más importantes para alcanzar el éxito del proyecto.