INTRODUCCION AL MODELADO

A menudo, los nuevos inventos nacen sin necesidad y se documentan sobre servilletas mucho antes de que se proporcione una definición autorizada y formal. Todo gira en torno de una visión. Un sistema complejo toma forma cuando alguien tiene la visión de cómo la tecnología puede mejorar las cosas. Los desarrolladores tienen que entender completamente la idea y mantenerla en mente mientras crean el sistema que le de forma.

En la actualidad, pensar en un desarrollo de software sin pasar por una etapa de modelado es prácticamente imposible. La utilización de modelos es una metodología aceptada y recomendad no solo académicamente, sino también dentro del ambiente profesional privado. De hecho desde las más altas jerarquías en la pirámide empresaria determinan las políticas que se aplicaran a para el empleo de modelos en el desarrollo de productos de software. El éxito de los proyectos de desarrollo de aplicaciones o sistemas se debe a que sirven como enlace entre quien tiene la idea y el desarrollador.

Los modelos actuales que representan sistemas de software son creados a través de lenguajes de modelado. Como los lenguajes de programación, estos también tienen una sintaxis (la forma de los elementos del lenguaje) y una semántica (el significado de esos elementos) definida.

Existen muchos lenguajes de modelado, cada uno con diferentes propósitos. Algunos son específicos para un área en particular, como base de datos o sistemas de tiempo real, y otros son de propósito general, para todo tipo de aplicaciones.

El UML es considerado como un estándar dentro de la comunidad científica, ya que le ayuda a capturar la idea de un sistema para comunicarla posteriormente a quien esté involucrado en su proceso de desarrollo, esto se lleva a cabo mediante un conjunto de símbolos y diagramas. Cada diagrama tiene fines distintos dentro del proceso de desarrollo.

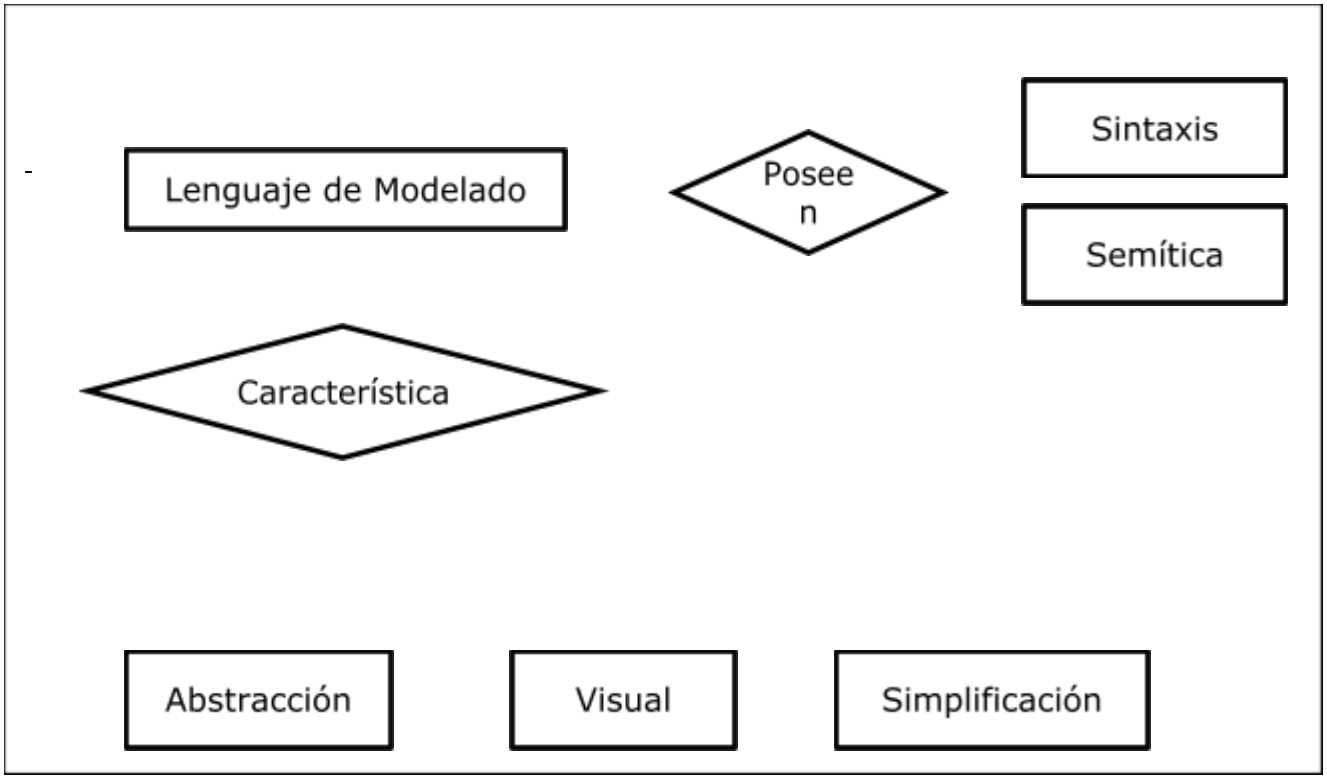
¿Qué son los Lenguajes de Modelado?

Los lenguajes de modelado son la herramienta que utilizamos para construir nuestros modelos de sistemas. Los lenguajes de modelado son, estructuralmente, similares a los de programación.

Una de las primeras cosas que hacemos cuando llegamos al destino de nuestras vacaciones es pasar por la oficina de turismo a buscar un folleto con un mapa dónde estén las referencias de los lugares para comer, para alojarse, de entretenimiento y las principales atracciones turísticas del lugar. Ese mapa, que nos acompañará durante toda nuestra estadía, no era más que un modelo de nuestro destino de vacaciones. Otra característica importante es que es visualmente atractivo y fácil de comprender, de manera que en parte de atrás, tenemos más información de cada lugar, por ejemplo, los teléfonos de los hoteles disponibles en la región. De esta manera tenemos un panorama general, pero al mismo tiempo podemos obtener información más específica fácilmente.

Entonces, podemos decir que un modelo es una **abstracción** de una entidad del mundo real, que nos ayuda a entenderla y a comprenderla en todas sus características y funcionalidades. Es fundamental que sea simple, capaz de adaptarse a las necesidades de mayor o menor abstracción (esto es poder ver los elementos con distintos niveles de detalle), y con una **notación gráfica**. Este último aspecto es trascendental y vital. De nada sirve un modelo si visualmente resulta extraño de comprender.

Nota: Así como un lenguaje de programación nos da errores de compilación por no respetar sus sintaxis, lo mismo pasará en el modelado si no respetamos las formas establecidas. La mayoría de los lenguajes de modelado no se ejecutan, pero es posible la generación de código a partir de modelos.



Figursa 1. Esquema con las características de los lenguajes de modelado.

Surgimiento de los modelos

El surgimiento de los modelos de software está íntimamente relacionado con la evolución de las Ciencias de la Computación y, de la Ingeniería de Software. Los primeros sistemas de software eran puramente aplicaciones científicas, llevadas a cargo generalmente por una sola persona. A medida que los cambios tecnológicos, científicos o de hardware fueron sucediendo, la situación cambió drásticamente. Podemos enumerar esta evolución desde cuatro puntos de vista: según los lenguajes de programación, según las aplicaciones , según el perfil del programador y según la confiabilidad esperada,. Analicemos cada uno de ellos.

Evolución desde los lenguajes de programación.

Podemos nombrar la evolución en los lenguajes de programación de la siguiente manera, inicialmente, los primeros programas se codificaban a través de **tarjetas perforadas**. Una tarjeta perforada es un pedazo de papel que representa información digital a través de la presencia o ausencia de agujeros en posición fijas. Luego los lenguajes fueron mutando de programar con tarjetas perforadas a emplear ceros y unos, accediendo

directamente a registros de memoria con directivas de lenguaje **Assembler**. Luego pasaron a instrucciones de alto nivel de Java, como por ejemplo, **GetConnection**, para conectarse a una base de datos o a los constructores para recorrer estructuras de datos como el **foreach**.

La utilización de estos modernos **frameworks** complejos dio lugar a lo que se conoce como **middleware**. En estos framework es bastante común encontrar funcionalidad prefinida para manejar conceptos como seguridad, transacciones o persistencia. Asimismo, proveen capacidad para manejar trabajo en equipos distribuidos. En este sentido, una de las mayores dificultades en los ambientes distribuidos es el **debugging** de código. Encontrar un error en ambientes de ejecución paralela o concurrente es extremadamente difícil.

Evolución desde las aplicaciones.

Las primeras aplicaciones consistían en complejos cálculos matemáticos sobre enormes computadoras que ocupaban habitaciones enteras. Hoy tenemos prácticamente cualquier tipo de computación sobre pequeñísimos chips para teléfonos celulares u hornos microondas, por misionar dos casos. La computación interactúa con nosotros constantemente al estar presente en cajeros, sistemas de videoclub, manejo de información, telefonía, comunicación, internet, etc. La presencia y la posibilidad de computo actuales han influido y potenciado otras ares con la medicina, el desarrollo de medicamentos, la robótica, el reconocimiento de huellas digitales o de retina, la biotécnica, etc.

Evolución desde el perfil del desarrollador.

En la actualidad, los equipos de trabajo dedicados al desarrollo de software son multipersonales e interdisciplinarios, con jerarquías desde arquitectos, líderes de proyecto, analistas funcionales, testers, programadores, etc. Estos estructurados equipos actuales de desarrollo son fruto de investigaciones en la teoría de las interrelaciones y manejo de grupos. Por el contrario, en los primeros programas de computación el equipo de desarrollo consistía, en general de una única persona que llevaba toda la responsabilidad del proyecto.

Evolución desde la confiabilidad esperada.

Como dijimos, las primaras computadoras eran enormes aparatos que ocupaban habitaciones enteras. Programarlas para llevar a cabo una tarea no era para nada trivial, y su preparación llegaba a ocupar días enteros. Bajo este contexto, los errores eran esperables y se tardaban días o semanas en corregirlos. La anécdota cuenta que por esos tiempos nación el concepto de bug (bicho en inglés) para nombrar el error de un programa ya que la presencia de insectos dentro de las computadoras causaban fácilmente problemas de hardware.

En la actualidad, en cambio, se busca intensamente y cada vez más la ausencia de errores en software. Para esto, se han desarrollado sofisticadas estrategias formales para la detección de errores, manejo de riesgo, mantenimiento y evolución.

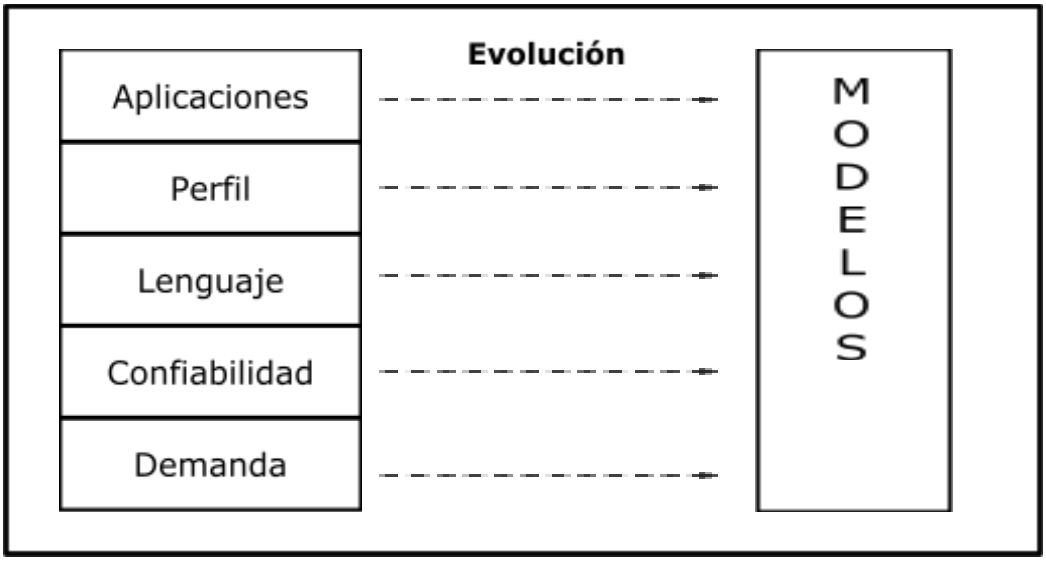


Figura 2. La evolución del software transformó la utilización de modelos. De una etapa considerada como poco productiva o inútil, paso a ser una etapa imprescindible en cualquier proceso de desarrollo.

Cambios en el proceso de desarrollo.

Como vimos, mucho ha cambiado en lo que se refiere al empleo y a la utilización de computadoras, mostrándose un gran avance y maduración. Todos estos cambios influyeron en la manera y en los pasos que se siguen durante el proceso de crear y desarrollar software específico. Como mencionamos, los primeros programas eran implementados por una sola persona sin que existiera la noción de análisis o modelo más allá de la idea en la mente del único programado a cargo. A medida que los proyectos fueron creciendo en magnitud y se fie acrecentando la injerencia de la computación en el desenvolvimiento de la sociedad, este prototipo de equipo quedo obsoleto. Más personas fueron asignadas a cada proyecto y nació la necesidad de primero poner en papel los conceptos que luego serían implementados. En este periodo es conocido como la Crisis del Software (año 1968), momento en el que se reunieron expertos de la comunidad para enfrentar los siguientes problemas en el desarrollo del software:

- ✔ Los proyectos no terminaban en los plazos estipulados.
- ✔ No se cumplían los presupuestos planeados.
- ✔ Se obtendría un código final de baja calidad que no cumplía las especificaciones planteadas inicialmente.

Estos cambios introdujeron nuevas etapas en los proceso de desarrollo. Básicamente, los procesos de desarrollo cuentan con las siguientes etapas. La primera de ellas consiste en la **recolección de requerimientos**, es decir, establecer las responsabilidades y funcionabilidades del sistema. La segunda etapa se concentra en el **análisis y el diseño**. En ésta, los actores protagónicos son los lenguajes de modelado y los modelos que son creados a partir de ellos. El principal objetivo de esta etapa es obtener una visión completa del sistema, a través de modelos que correctamente abstraigan las principales características y el comportamiento esperado del sistema. Luego, se pasa la etapa de **implementación** del producto. Esto significa su codificación y programación. Finalmente, se lanza el sistema a producción, pasando previamente por una etapa **testing**, en la que se valida el sistema. Una vez que el sistema entra en funcionamiento los cambios en el los requerimientos y la corrección de errores involucran las tareas principales en la actividad que se conoce como **mantenimiento de sistema.**

Sin embargo, lo más novedosos procesos de desarrollo incorporan dos etapas más. La primera de ellas, conocida como **arquitectura del sistema**, se sitúa entre la captura de requerimientos y el análisis de diseño. En objetivo es modelar los principales **componentes** del sistema junto con su interacción básica. Algunos esquemas clásicos de arquitectura son la **cliente/servidor** o la **Pipe and Filter**. La segunda nueva etapa es de **validación** y **verificación formal**, la cual se sitúa generalmente antes del nacimiento del software en pos de asegurarnos el correcto funcionamiento del sistema, a través de técnicas matemáticas formales.

Existen varios procesos de desarrollo de software, pasando por el clásico **modelo de cascada** hasta los más modernos, como los **procesos agiles de desarrollo**. Si bien son bastante diferentes entre sí, con distintos objetivos y áreas de aplicación, la etapas de mencionamos anteriormente se ven reflejadas en cualquier proceso de desarrollo, de una manera u otra.

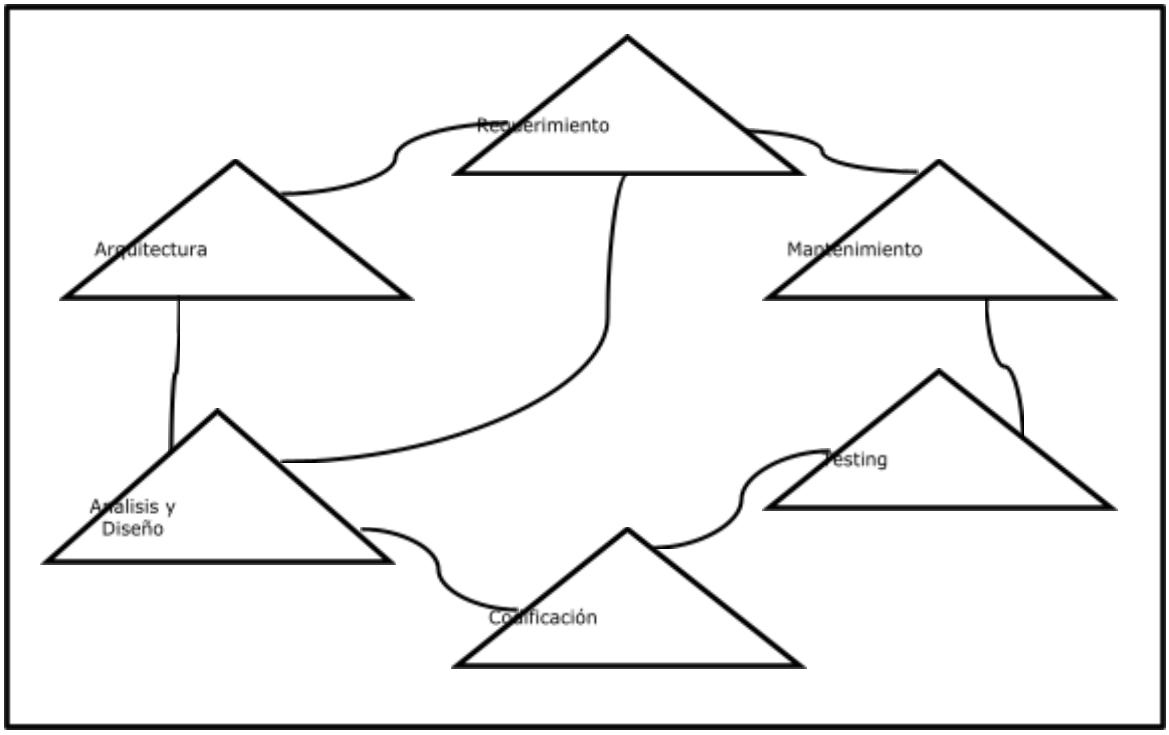


Figura 3. Esquema de los procesos de desarrollos modernos.