Modelos y Sistemas.

Los modelos representan una abstracción de nuestro sistema que nos permite concentrarnos en los detalles que más nos interesan, dejando de lado otros aspectos secundarios. En este sentido es importante reconocer que los modelos no son el sistema. Los modelos deberán seguir un proceso hasta convertirse, en el futuro, en el sistema implementado. Una pregunta razonable que nos podemos haces en este momento es cómo podemos estar seguros de que la implementación que nos da como salida la etapa de codificación respecta los modelos planteados en la etapa de análisis y diseño. Podemos abarcar esta cuestión desde tres lugares.

Como primera medida, es fundamental el total entendimiento del leguaje de modelado y de los modelos desarrollados por estos lenguajes, en tanto por parte de los programadores como de los líderes de proyectos. De esta manera, nos aseguramos de que los programadores codifiquen correctamente el comportamiento expresado en nuestros modelos. Por otro lado, también es importante tener en cuenta que si hay una situación en las que por razones de implementación (por ejemplo, la ubicación física de un servidor) es necesario introducir cambios en los modelos del sistema, estos sean correctamente documentados para así poder mantener siempre una relación estrecha y directa entre los modelos y el código. Otra alternativa es brindada directamente por los entornos de los lenguajes de modelado.

Por último, existen procesos de desarrollo que intentan maximizar los procesos de automatización dentro del desarrollo de software, es decir , partir de modelos que automáticamente se transforman en el producto final. Esta técnica se conoce como **Model-Driven Development** o **Desarrollo Basado en Modelos**. Otras herramientas nos permiten, a partir de modelos, crear esqueletos o armazones de nuestro sistema, a través de los cuales podemos imaginarnos cómo será el producto final. Resulta claro que introducir cambios en estos esqueletos es mucho más sencillo que hacerlo en el sistema final, de la misma manera que llevar a cabo modificaciones en el plano de una habitación es mucho más sencillo que efectuarlos en la habitación en sí. Imaginemos que realizar una nueva división en la habitación consiste en trazar una línea en el plano, mientras que levantar realmente la pared en la habitación es una tarea mucho más compleja.

El rol de los modelos en la Ingeniería del Software.

La utilización de modelos es el caballito de batalla de la Ingeniería de Software. Pero ¿Qué es la Ingeniería de Software? Y aun antes, ¿Qué entendemos por software? La definición tradicional sugiere que un **sistema de software** es una colección de **componentes interrelacionados** que trabajan conjuntamente para cumplir algún objetivo. Sin embargo, una definición más adecuada sería; *un conjunto de entidades cuto comportamiento dará solución a un problema inicial, y será ejecutado automáticamente (por computadoras).*Esta definición es más adecuada por las siguientes razones:

- **Entidad** es un concepto mas abstracto que el concepto de componente. Además, hablar de componentes puede orientar hacia la Programación Basada en Componentes y generar, así, una confusión.
- Hace explicita la palabra **comportamiento**, que resulta más precisa y adecuada que la palabra interacción.
- Hace referencia explicita también a un **problema inicial**, el cual se quiere resolver.

Es fundamental comprender que el sistema de software no es únicamente el CD o DVD que contiene el programa, sino que el concepto de entidad abarca también, además del código, toda la documentación del sistema, incluyendo sus modelos. Además. Se desprende de la definición que las entidades colaboraran entre sí para lograra su objetivo, y que el comportamiento individual afectara e impactará en las demás. Por lo tanto, es fundamental que la interacción de las entidades estén correctamente ilustrada y reflejada en nuestros modelos de sistema.

Finalmente, la IIEE define al software como la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo.

Errores del software versus errores del diseño

En algunas ocasiones, se confunden los errores propio es del software con errores del diseño. El software es intrínsecamente flexible y debemos estar preparados para ello. Supongamos que hemos desarrollado un sistema para capturar mensajes de transmisión por radio. Es implementado correctamente, es instalado y su funcionamiento es el esperado. Sin embargo, por razones operativas, un mes después, el servidor donde funciona nuestro software se cambia de piso en el edificio donde funciona la empresa cliente. En este nuevo lugar, se producen interferencias con otros aparatos tecnológicos y nuestro programa comienza a fallar. La solución consiste en mejorar la recepción de la señal, pero estos cambios degradan notablemente la performance y vuelve demasiado lenta la ejecución normal de nuestro sistema. ¿Estamos frente a una falla de software o frente a una falla en el diseño? La segunda opción es la correcta. Debió estar mejor contemplada la interacción entre la performance y el impacto de cada entidad en ella.

Ingeniería en Software.

Ahora que tenemos una definición más certera de los sistemas de software, podemos encarar la definición de Ingeniería de Software. Algunas definiciones encontramos son las siguientes:

- La Ingeniería de Software es la rama de la Ingeniería que aplica los principios de la Ciencias de la Computacion y las Matemáticas para lograr **soluciones costo-efectivas** (eficaces en costo o económicas) a los problemas de desarrollo de software, es decir, permite elaborar consistentemente productos correctos, utilizables y costo-efectivo.
- La Ingeniería de Software involucra construir un producto de software de alta calidad lidiando con las múltiples restricciones (tiempo, presupuesto y demás).
- La disciplina de Ingenia de Software involucra conocimientos, herramientas y métodos para definir y capturar los requerimientos, realizar el diseño del sistema y sus correspondiente codificación, validación y mantenimiento. La Ingeniería de Software se nutre de conocimientos de otras aéreas como Ingeniería Computacional, Ciencias de la Computacion, Administración de Empresas, Matemática,

Administración de Proyectos, Administración de calidad e Ingeniería de Sistemas.

- Por último, la definición de IEEE dice que la Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantifícale al desarrollo, operación y mantenimiento de software. En términos prácticos, es una área o disciplina de la Informática que ofrece métodos técnicas para desarrollar y mantener software de calidad.

De todas estas definiciones podemos obtener en claro los siguientes aspectos: la Ingeniería de Software debe lidiar con la **escalabilidad** y la **complejidad** de sistemas de software, identificando claramente aquello que se pretende por software de alta calidad. A su vez, requiere, como todas las ingenierías, rigo, creatividad, documentación y gestión. Por ultimo, es una actividad multidisciplinaria.

Modelos del Software.

Con lo que hemos visto hasta ahora sabemos que los modelos son una simplificación del sistema que queremos construir. La principal razón para modela es comprender más profundamente el sistema que vamos a desarrollar. A través del modelado obtenemos los siguientes beneficios:

- Visualizar en etapas tempranas del desarrollo el comportamiento del sistema.
- Especificar tanto el comportamiento como la estructura del sistema.
- Guiar el desarrollo del sistema.

Los modelos no solo son útiles en grandes proyectos, sino que también lo son en proyectos pequeños y medianos. Aun las golosinas para niños que vienen con pequeños juguetes para armar tienen un folleto de instrucciones y un dibujo de cómo se ve el juguete armado. Es decir, la golosina viene acompañada no solo con el juguete, sino con su modelo. También es cierto que aunque la construcción de modelos en sistemas pequeños es sumamente recomendable, quizás con mayor estructura y comportamiento, es directamente imposible.

Un sistema pueden ser tomado desde muchos puntos de vista: a partir de su **estructura**, su **jerarquía**, su **comportamiento**, sus características, de manera interna, de manera externa desde alguna **funcionalidad** en especial (por ejemplo, el sistema la seguridad, etc). En ocasiones, será necesario concentrarse en detalle en un componente en especial, conociendo sus atributos, sus métodos y la funcionalidad que provee. En otras, quizá queramos apreciar cómo interactúa con otros componentes, por lo que nos alcanzara con ver únicamente el nombre del componente y obviar su comportamiento interno. Existe muchos modelos para nuestro sistema y cada uno muestra un punto de vista en particular, o provee menor o mayor nivel de detalle sobre los componentes.

Así como un sistema se divide en **subsistemas** menores de manera tal que la colaboración de todos ellos permite solucionar el problema planteado, un modelo también se divide en distintos **submodelos**, y cada uno de ellos muestra y se enfoca en un área en particular planteando en términos de operaciones de conjunto, la intersección de todos ellos dará como resultado el modelo de nuestros sistema.

Características del modelado.

Así como mencionamos los principios de la Ingeniería de Software, también existen características distintivas para la especificación de propiedades y el comportamiento a partir de modelos de software. A continuación, las encontraremos descriptas:

- **Los modelos son metamórficos**: los modelos pueden tomar múltiples formas. El estilo particular de cada modelo está íntimamente relacionado con el tipo de proyecto en marcha. Si necesitamos simular el comportamiento de las mareas en una determinada laguna, quizás un modelo matemático a partir de integrales de Gauss y software de simulación sea el modelos adecuado. Si, en cambio, necesitamos estudiar el trayecto optimo para que un viajante pueda recorrer todas las ciudades que necesita en el menor tiempo posible, entonces el modo matemático de grafos nos viene a la perfección. Lo mismo ocurre con nuestros sistemas: existen modelos enfocados en la arquitectura, en la estructura estática, basados en el comportamiento, etc. Ante cada sistema, debemos pensar en los tipos de modelos que vamos a utilizar.

- **Los modelos deben manejar distintos niveles de abstracción**: supongamos que tenemos el modelo finalizado de nuestro sistema y lo queremos presentar a los gerentes del proyecto. SI llevamos todos y cada uno de los diferentes modelos, la reunión será un fracaso ya que habrá una cantidad enorme de modelos. Quizá, lo más conveniente sea llevar un modelo de la arquitectura donde se vea la interacción de los principales componente. En caso de querer profundizar en el comportamiento de alguno de ellos, tendremos listo el modelo desde ese componente en particular de esta manera, logramos el efecto zoom in-zoom out sobre las propiedades salientes del sistema.
- **Los modelos deben ser coherentes**: de poco sirve uin modelo sobre el comportamiento de un tren si no tiene en cuenta la capacidad de peso que soportan las vías. Es decir, debemos incorporar a nuestro modelo la mayor cantidad de información posible del entorno sobre el cual nuestro sistema se ejecutara. Cualquier tipo de información que sea relevante y no sea incorporada a nuestro modelo terminara repercutiendo negativamente en la calidad del software desarrollado.
- **Todos los modelos cuentan**: hemos visto como diferentes modelos capturan distintos puntos de vista del sistema. No es aconsejable tomar decisiones basándonos en un solo, ignorando el resto, ya que hay variables y parámetros que están siendo excluidos y que son importantes para la decisión en cuestión. Por ejemplo, si seguimos únicamente el punto de vista de eficiencia, obtendremos un sistema que se ejecute muy rápido, pero seguramente muy poco amigable. Todos los distintos criterios y puntos de vista deben balancearse.