Más abstracción, **menos código:** ventaja de modelar

Cuáles son los beneficios de la ingeniería de software dirigida por modelos (*Model-Driven Engineering*, MDE, por sus siglas en inglés), qué hay detrás y consejos para implementarla: de eso se trató la conferencia "Versión para incrédulos" del español Jordi Cabot, reconocido internacionalmente como experto en MDE.

Foto: Natalia Fernanda Madrid Vidales

Jordi Cabot durante su conferencia titulada "Versión para incrédulos".

Para sacarle el máximo beneficio con el mínimo esfuerzo a la metodología MDE hay que saber cuáles son los modelos útiles para un proyecto de desarrollo de *software*. Si los modelos son adecuados y tienen propiedades relevantes, entonces pueden ser el principal artefacto que manipule el ingeniero en lugar del código. Además, hay que plantearse de qué tamaño es el proyecto, cuál es la tecnología destino y qué equipo lo va a trabajar. En función de estos parámetros se puede pensar en la mejor estrategia de modelaje.

Así lo afirmó Jordi Cabot en el 2.º Foro de Ingeniería de *Software*: "Tendencias para automatizar el desarrollo de *software*. Casos reales", que se llevó a cabo el 19 de octubre del 2016. MDE es una metodología de desarrollo que emplea modelos, con la cual es posible semiautomatizar la producción de código y con ello reducir costos y mejorar los tiempos de desarrollo.

El Departamento de Ingeniería de Sistemas y Computación (DISC) organizó este Foro ISIS con el fin de dar a conocer experiencias de implantación exitosas de MDE en la industria de *software* colombiana. Además de Cabot, profesor de la Universitat Oberta de Catalunya (España), asistieron Kelly Garcés, profesora del DISC, quien con Cristo Rodríguez, de Asesoftware, habló del proyecto de migración de aplicaciones Oracle Forms usando MDE; Catalina Acero, directora del área de Soluciones de Ingeniería de Heinsohn Business Technology, explicó cómo el uso de MDE ha servido para reutilizar componentes desarrollados por su compañía y así aumentar calidad y productividad; y Andrés Yie, director de tecnología del Grupo GHL, relató cómo esta empresa que gerencia hoteles ha logrado desarrollar *software* de manera eficiente con un pequeño grupo de desarrolladores. Encontrará testimonios de estas experiencias a continuación del siguiente artículo sobre la intervención de Jordi Cabot.

Escribir menos código

El experto afirmó que si bien los modelos están presentes en cualquier ingeniería, la de *software* se ha rezagado en la implantación de esta metodología, perdiendo una ventaja competitiva. "Son artefactos centrales que están en toda actividad de desarrollo: la documentación, la transformación, la generación de código, los prototipos y el análisis estático" (el que se realiza sin ejecutar el programa), aseguró Cabot.

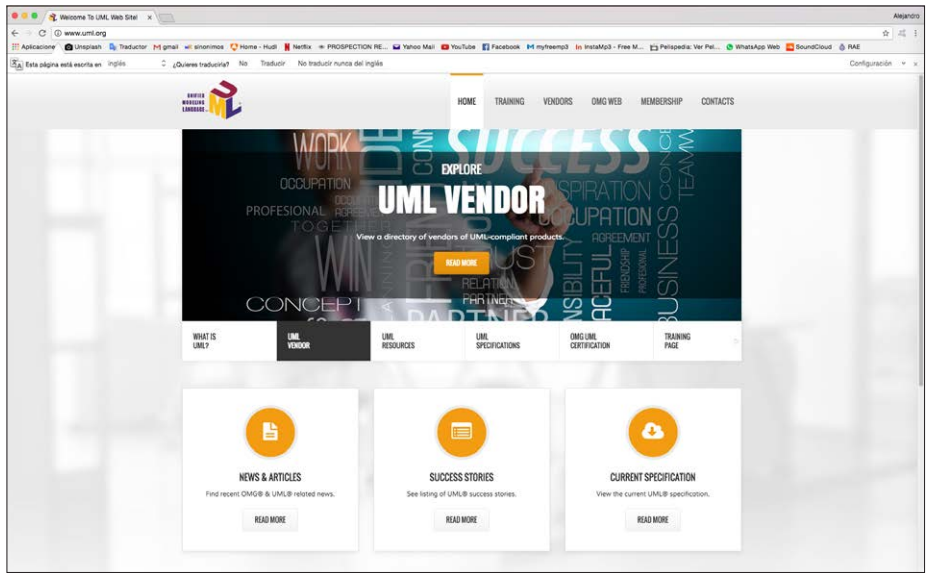
Dijo que el objetivo de la metodología siempre ha sido escribir menos código y hacerlo de manera más fácil, de tal forma que aumentar el nivel de abstracción —eso es lo que hacen los modelos— era el camino natural. Esta evolución, que lleva 30 años, se ha traducido en diversos *frameworks* como Symfony, Django, Yeoman, Rails, Meteor, Grails y Java Script que simplifican la generación de código, al definir un modelo de datos en un lenguaje textual, y a partir de allí se ejecuta un proceso que tiene como salida la interfaz gráfica, la lógica y el

acceso a datos básicos de una aplicación. "El modelaje es la combinación y la formalización de ese proceso y quien lo diseña define y especifica las funciones del programa. Con esta información, el desarrollador produce el *software*", explicó.

Al hablar del patrón que sigue dijo que "el *software* no es más que un conjunto de modelos al que se suman unas transformaciones que los manipulan hasta llegar a la aplicación final". Para que ello se dé se deben cumplir estas etapas:

- Expresar los requisitos en lenguaje natural.
- Formalizarlos en diagramas y casos uso.
- Elaborar un diagrama de clases con los datos que se manipularán en el *software* para dar respuesta a esos requisitos.
- Transformar los diagramas a código Java o SQL (*Structured Query Language*).

De otra parte, para que el modelo y el programa sean válidos siguen una serie de reglas sintácticas y estructurales. Eso se refiere a que obedecen a un metamodelo, de manera que es interpretable. Para ello se emplea un lenguaje. ¿Cuál? UML (*Unified Modeling Language*) es el más conocido, es estándar y da las opciones que se necesitan para construir una aplicación. También es la base de los demás lenguajes.



http://www.uml.org/

UML (*Unified Modeling Language*) proporciona las reglas sintácticas y estructurales para que el modelo y el programa sean válidos y es la base de otros lenguajes de modelaje.

Empezar con proyectos pequeños

Según Jordi Cabot, entre los beneficios que provee MDE están los siguientes:

- Reduce hasta cuatro veces el número de errores en la aplicación, lo que redunda en una mejora de hasta un 80 % en el costo del mantenimiento. "Esto es particularmente cierto cuando llega un nuevo desarrollador y encuentra los modelos que explican una arquitectura, sus funciones, los esquemas de la base de datos".
- En temas de modernización, es útil para visualizar la arquitectura, las clases, los diagramas, junto con otras funciones del código de un sistema existente.
- Sirve para emplear algún tipo de métrica o de análisis.

Por último, el experto mencionó algunos consejos para cuando se desarrolla con MDE:

- Ser ágil. No hay que modelarlo todo, solo lo que se necesita en una iteración determinada, lo cual obedece a una filosofía ágil. "Sin embargo, no se puede seguir un proceso estricto de metodología de desarrollo ágil y modelar al mismo tiempo, aunque hay gente estudiando este tema".
- El tamaño sí importa. Al usar MDE conviene empezar con un proyecto piloto pequeño, con gente entrenada, con un experto externo que guíe el equipo para minimizar riesgos.
- Ser pacientes. Cualquier nueva tecnología, al principio, hace perder tiempo y tiene un costo alto, pero luego, al automatizar la producción eso se invierte.
- Comprometer a la gerencia. Si los directivos no se involucran, el proyecto no funcionará porque hay muchas decisiones que requieren aprobación e implican nuevos costos.
- "Otra cosa: si está trabajando en MDE y falla, no diga que no sirve para nada sin antes verificar que no es usted el que ha fallado", dijo.

> "El modelaje es la combinación y la formalización de un proceso y quien lo diseña define y especifica las funciones del programa. Con esta información, el desarrollador produce el *software*".
>
> **Jordi Cabot**

La implantación, el primer desafío

Al finalizar su conferencia, el profesor Cabot respondió unas preguntas de la revista Foros ISIS.

¿Quién modela?

El modelador. Es alguien capaz de hacer un ejercicio de abstracción matemática. Pero, además, tiene habilidades sociales. Si tienes una visión más global del negocio y si en lugar de programar te dedicas a modelar, tu función será de más impacto.

¿El modelaje es costoso?

Al final, el producto será más barato en vista de que la generación de código

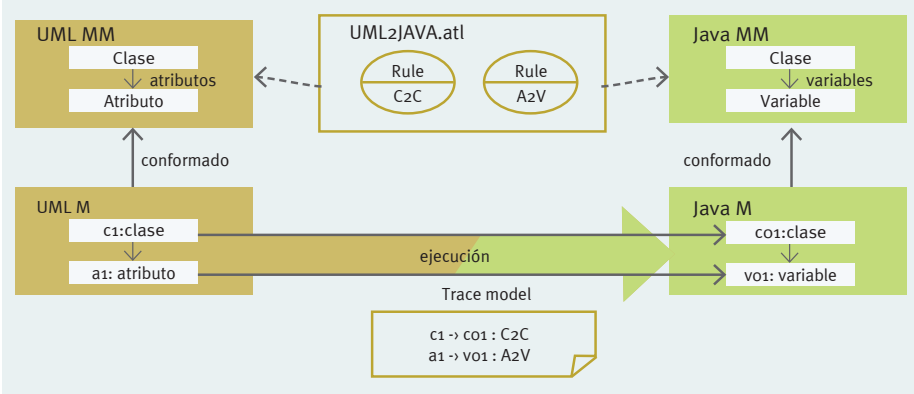
puede ser parcial, por lo cual necesitan menos gente y horas de trabajo para programar. Cambiar una parte que no funciona en el modelo es más rápido y menos costoso que hacerlo directamente en el código de la aplicación.

¿Qué representa para las empresas trabajar con MDE?

Les facilita evolucionar su *software* de manera más expedita, a medida que aparecen nuevas tecnologías. Es decir, si hay una tecnología mejor que la anterior, en lugar de reescribir el código, bastará con coger el compilador para esa nueva plataforma y regenerar el código a partir de los modelos existentes.

¿Cuáles son los desafíos para MDE?

A nivel industrial, la adopción. Una vez esto suceda, la escalabilidad. Hace un tiempo una empresa empezó a usar ATL (*Active Template Library*), pero luego de unos años lo dejó porque era mucho más rápido usar Java. Para escalabilidad, utilizar la nube, transformaciones distribuidas, usar base de datos en SQL (*Structured Query Language*) para guardar modelos grandes, un tema muy importante. El otro son las herramientas de verificación y testeo de modelos. Hay algunas, pero sigue siendo un trabajo en proceso. ■



Ejemplo de transformación de modelo a modelo.

Aproximación de **'caja blanca' para migración** de aplicaciones heredadas

Ahorro en esfuerzo y calidad en las aplicaciones migradas son los resultados de una solución MDE (*Model-driven Development Engineering*) a la que se llegó luego de evaluar los requerimientos de los clientes y las alternativas del mercado. Se migró la funcionalidad Crud y el código PL/SQL.

SMoT es una herramienta que emplea MDE para transformar código heredado de aplicaciones de Oracle Forms a nuevas tecnologías, conservando la funcionalidad original. Resultó de la alianza entre el grupo TICSw (Tecnologías de Información y Construcción de Software) del Departamento de Ingeniería de Sistemas y Computación (DISC) y la empresa Asesoftware, que ya la está utilizando con sus clientes. Este fue uno de los casos de éxito presentado en el foro por la profesora Kelly Garcés, del DISC, y por Cristo Rodríguez, de Asesoftware.

La profesora habló de cómo llegó al DISC este requerimiento y de cómo se escogieron las tecnologías que soportarían la solución. Asesoftware es una empresa con 25 años en el mercado colombiano y comenzó desarrollando y manteniendo aplicaciones de Oracle Forms para sus clientes. Hoy su gran desafío es atender las peticiones de migración de sus usuarios.

El grupo TICSw les propuso una solución basada en MDE. La primera etapa se destinó a migrar la funcionalidad Crud —*Create, Read, Update and Delete*— y a generar las interfaces gráficas y, un año después, parte del código PL/SQL (*Procedural Language/Structured Query Language*), un lenguaje de programación incrustado en Oracle, que es una mezcla entre código imperativo y SQL (*Structured Query Language*). En particular, se migró el que