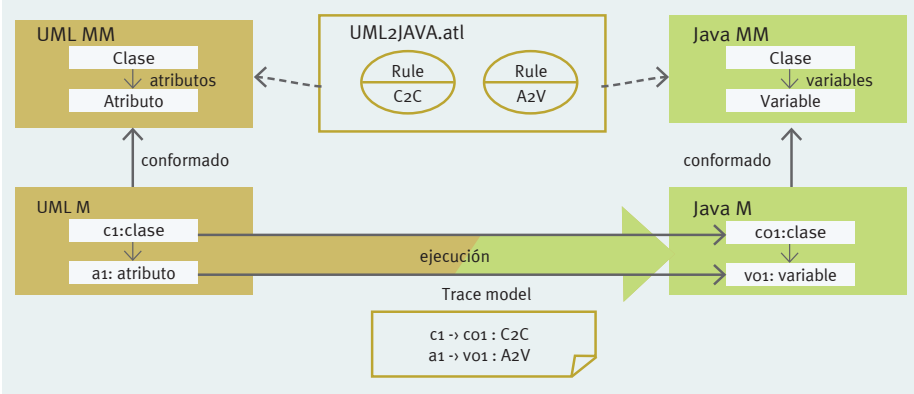
puede ser parcial, por lo cual necesitan menos gente y horas de trabajo para programar. Cambiar una parte que no funciona en el modelo es más rápido y menos costoso que hacerlo directamente en el código de la aplicación.

¿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.

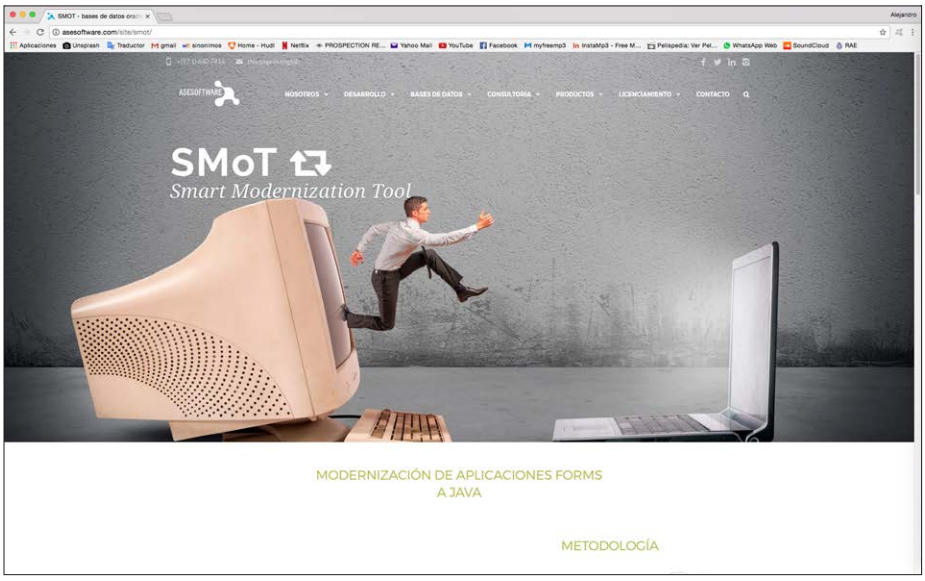
Aproximación de **'caja blanca' para migración** de aplicaciones heredadas

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



Asesoftware está utilizando SMoT, una herramienta que emplea MDE para la transformación código heredado de aplicaciones de Oracle Forms de sus clientes.

se encuentra embebido en los *triggers* (disparadores). Como tecnología destino se escogió JEE (*Java Plataform, Enterprise Edition*), pues así lo piden los clientes de Asesoftware.

Otras herramientas de migración y sus desventajas

Para darles más flexibilidad a los clientes de la empresa de desarrollo, el proyecto escogió tecnologías *open source*. Si bien Oracle provee una herramienta que migra Oracle Forms a Java, por ejemplo, los beneficios de la solución MDE propuesta por el grupo TICSw radican en que se evitan las aproximaciones 'de caja negra'. La profesora Kelly Garcés señaló los perjuicios de esas aproximaciones disponibles en el mercado:

- Son aproximaciones de 'caja negra' cuando no se sabe qué se va a obtener en el destino. "Esto sucede cuando, por ejemplo, al faltar información en el código heredado se produce una nueva aplicación que no es funcional".
- Hay dificultad para el mantenimiento: las herramientas de 'caja negra' siguen sus propios estándares de convención de nombramiento, por lo que el código generado es difícil de mantener. Adicionalmente, si la aplicación legada tiene código muerto, este se reproducirá en la aplicación destino.
- En el proceso de migración no es posible saber exactamente cuánto se ha avanzado.
- Estas herramientas comerciales son costosas y es muy difícil desligarse de su proveedor tecnológico.

Aproximación de 'cajas blanca' para migración de aplicaciones heredadas

El alcance inicial de la migración fue la funcionalidad Crud del código heredado. Primero, se creó un modelo de nivel de abstracción bajo, de forma que se facilitaran las siguientes modificaciones. Después se hizo una transformación del modelo de bajo nivel a un modelo independiente de la tecnología, llamado "intermedio". En este se tomaron las decisiones arquitectónicas. Por último, se transformó el modelo a texto, lo que permite generar el código final.

A continuación, la profesora explicó que sobre el modelo intermedio se pueden poner editores gráficos, que otorgan una visión de la estructura del sistema y subsanan la falta de documentación. La potencialidad de esta aproximación de caja blanca radica en que se conoce lo que hay al interior del código heredado.

Además, con el editor, el arquitecto o el desarrollador que dirige el proyecto puede distribuir el trabajo, reorganizar los menús para mejorar la usabilidad, completar información faltante en el modelo. También establecer qué parte es código muerto o duplicado para no migrarlo: así se enfrenta el reto de la mantenibilidad. Y por último, se aprecia el progreso del proceso de migración.

Al comparar las ganancias en productividad y las mejoras en la calidad del código migrado, se encontró que la productividad aumentó el 40 %. Además, los desarrolladores señalaron la facilidad del editor gráfico para configurar la arquitectura y generar mucho código, lo cual resulta en menos trabajo. En términos de calidad, las mejoras fueron del 61 % en reducción de defectos.

Migración del PL/SQL

Cristo Rodríguez, de Asesoftware, explicó los pasos del proceso de migración del código PL/SQL: primero se obtuvo un modelo del código PL/SQL a través de un *parser* (compilador). Posteriormente, ese modelo se transformó en uno de más alto nivel de abstracción en el cual se identificaron patrones de código típicos, repetitivos. Este se mezcló con el modelo intermedio. Por último, a partir de los dos modelos, se generó el código de una aplicación JEE con la funcionalidad Crud, con pruebas unitarias y con reglas propias del PL/SQL.

Cuando este proceso es manual, alrededor del 37 % del esfuerzo se dedica a inspeccionar el código PL/SQL y a transformarlo en la tecnología destino. Con la herramienta automática, esta etapa se reduce a la mitad. "¿Por qué no hubo un ahorro del 100 %? Porque no todo el código es capturado por el catálogo de patrones que tenemos", dijo Cristo Rodríguez.

Lecciones aprendidas

- Todas las personas involucradas en el proyecto recibieron entrenamiento, cursos formales e ingresaron a la Maestría en Ingeniería de *Software* de la Universidad de los Andes, donde el modelaje es una asignatura. Además, el compromiso de la gerencia fue clave.
- Los patrones de código son fuertemente dependientes de los estándares y convenciones de código de las organizaciones, así como de las decisiones

arquitectónicas tomadas cuando se construyó la aplicación heredada. El propósito es extender los patrones a las particularidades de cada cliente para ampliar la cobertura de descubrimiento de código automático.

- Gran parte del código PL/SQL implementa lógica de interfaz de usuario que es muy particular del ambiente gráfico de Oracle Forms (por ejemplo, navegar los registros de la base de datos con accesos directos de teclado) y, por lo tanto, es casi imposible migrarla a otras tecnologías modernas como Web.

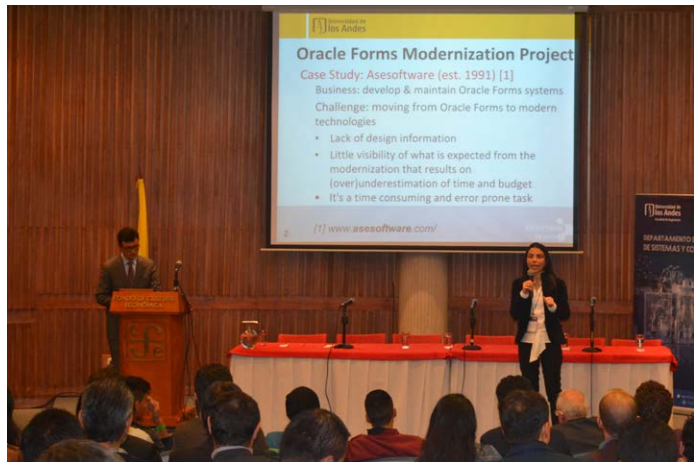


Foto: Natalia Fernanda Madrid Vidales

Kelly Garcés, profesora del Departamento de Ingeniería de Sistemas y Computación, y Cristo Rodríguez, durante su presentación del trabajo que el DISC desarrolló en alianza con Asesoftware.

Dos conclusiones

El modelo intermedio impide los altos costos de retrabajo. No solo se transforman las funcionalidades Crud, sino también la lógica escrita en PL/SQL. El código destino es claro y entendible, lo cual facilitará su mantenimiento. Adicionalmente, en PL/SQL la traducción del código no es, por ejemplo, de instrucción a instrucción, sino que cada patrón catalogado tiene un propósito funcional y es implementado siguiendo las mejores prácticas en la arquitectura destino.

La segunda conclusión es que, a través de modelos, se facilita independizar la solución de las tecnologías origen y destino. Por ejemplo, si Asesoftware decide migrar Oracle Forms hacia .NET, el proceso de modernización es el mismo, sólo bastaría construir una nueva transformación para generar el código en esa tecnología específica. ■

Calidad y eficiencia, beneficios de los generadores

Con el fin desarrollar proyectos de *software* con alto nivel de calidad y de productividad, Heinsohn escogió una solución basada en MDE (*Model Driven Engineering*). Por cada 4,6 líneas de código desarrolladas a mano, se generan automáticamente hasta 8794.

La reutilización de componentes existentes al momento de construir nuevas aplicaciones se ha constituido en una ventaja competitiva para Heinsohn, empresa de desarrollo de *software*. Así lo afirmó Catalina Acero, socia y directora del área de Soluciones de Ingeniería, quien presentó la experiencia, los resultados y los desafíos del uso de MDE en Heinsohn, durante su conferencia "Ingeniería dirigida por modelos en el campo de batalla".

Relató cómo enfrentaron el reto de semiautomatizar la construcción de componentes particulares de un negocio que se integran con los componentes transversales, tales como módulo de seguridad, notificaciones de correo, manejo de históricos. Además, ¿cómo minimizar el tiempo que los ingenieros invierten en entender los microdetalles de las nuevas tecnologías y en solucionar errores de integración, sean de tecnología o por incomprensión del negocio?

Foto: Natalia Fernanda Madrid Vidales

Catalina Acero es directora del área de Soluciones de Ingeniería de Heinsohn Business Technology, una compañía con 700 colaboradores y presencia en Colombia, Estados Unidos, El Salvador y Ecuador.