Análisis de Herramientas CASE para uso didáctico en Diseño de Bases de Datos

Cecilia Belletti, Regina Motz

cecibell@adinet.com.uy, motz@athenea.ort.edu.uy

Facultad de Ingeniería, Universidad ORT Uruguay

RESUMEN

En este trabajo se plantea un marco evaluatorio que resume las características elementales que debería presentar una herramienta CASE para el apoyo a los cursos de Bases de Datos, contemplando aspectos técnicos y didácticos. Se realiza una evaluación de herramientas CASE existentes relevadas y se plantea cuán lejos se está de satisfacer el ideal planteado y posibles alternativas para lograrlo.

1. Introducción

El objetivo final de nuestro trabajo es encontrar una herramienta CASE que sirva de apoyo a los cursos de Bases de Datos contemplando aspectos tanto técnicos como didácticos para mejorar las instancias de formación y de información al estudiante fuera del salón de clase y aprovechar las características de las mismas para consolidar prácticas de ingeniería de software, elevando la productividad y calidad en el desarrollo de sistemas de información. Con este fin se presenta un marco evaluatorio donde se consideran los aspectos antes mencionados y dentro del mismo, se realiza una evaluación de distintas herramientas CASE.

La organización de este artículo es la siguiente. En la sección 2 se resumen las distintas clasificaciones que encontramos sobre las herramientas CASE. Más adelante, en las sección 3 nos introducimos en los aspectos didácticos que pueden tener relevancia al adoptar una herramienta de apoyo de estas características, y en la sección 4 se plantea un marco de evaluación contemplando aspectos técnicos como didácticos. En la sección 5 se resume la evaluación realizada de los dos productos más relevantes con respecto al marco planteado en las secciones anteriores. En la última sección se detallan las conclusiones y las propuestas de los pasos a seguir a partir de este trabajo.

2. Herramientas CASE

El término CASE (computer-aided software engineering) significa “ingeniería de software asistida por computadora”, y abarca el uso de un método asistido por computadora para organizar y controlar el desarrollo de software, especialmente en proyectos grandes o complejos, que involucran muchos componentes de software y recursos humanos. [1]

Actualmente existe una variada gama de herramientas CASE para distintos propósitos específicos, y son ampliamente utilizadas, ya sea como apoyo a algunas actividades o incluso como herramienta de gestión del proceso de desarrollo de un proyecto de software.

2.1. Clasificación de herramientas

Antes de incursionar en la descripción de las distintas herramientas evaluadas, distinguiremos los distintos tipos de herramientas que existen, así como los parámetros a tomar en cuenta al momento de evaluarlas.

Dependiendo de la fase del ciclo de vida que soportan, podemos distinguir entre herramientas CASE ***front-end*** o ***Upper CASE***, que abarcan las primeras fases de análisis y diseño; y ***back-end*** o ***Lower CASE***, cuyo objetivo suele ser el diseño detallado y la generación de código. Se denomina ***ICASE*** (Integrated CASE) a las herramientas que comprenden ambos tipos, e ***IPSE*** (Integrated Programming Support Environment) a las que incluyen componentes para la gestión de proyectos y de configuración [2].

Otra clasificación posible de las herramientas CASE es la presentada por Esperanza Marcos, Adoración de Miguel y Mario Piattini en [2]:

1) **Herramientas de análisis y diseño**
   Permiten crear y modificar diagramas Entidad-Relación, de flujo de datos, de clases, etc.
   Son importantes también las herramientas de prototipado. Éstas incluyen diseñadores de formularios, de menúes, de informes, y lenguajes de especificación ejecutables. Un aspecto a destacar es la capacidad de análisis y verificación de especificaciones soportado por la herramienta (sintáctica y semánticamente), como por ejemplo, la capacidad de normalizar hasta la tercera forma normal.
2) **Generación de código y documentación**
   Generan código a partir de las especificaciones de diseño. Además, soportan la generación automatizada de documentación a partir de la información almacenada.
3) **Herramientas de prueba**
4) **Herramientas de gestión de la configuración**
5) **Herramientas de ingeniería inversa**
   - ***Ingeniería inversa de datos***, extraen información de código fuente y construye diagramas orientados a objetos o Entidad-Relación.
   - ***Ingeniería inversa de procesos***, permiten aislar la lógica de las entidades y las reglas del negocio a partir del código.
   - ***Reestructuración de código fuente,*** modifican el formato o implantan un formato estándar.
   - ***Redocumentación***, permiten generar diagramas para mejorar la comprensión del código.
   - ***Análisis de código***, generan, por ejemplo, la indentación automática.

Para el campo de las Bases de Datos específicamente, podemos definir las siguientes clases de herramientas de diseño [1]:

- **Sistemas de prototipos de investigación**
- **Herramientas CASE comerciales específicas**
- **Herramientas CASE generales.**

Los primeros son los más completos pero suelen ser difíciles de encontrar y no cuentan con soporte adecuado. Las herramientas específicas son adecuadas para quienes desarrollen en entornos donde el componente de datos tenga un peso fundamental. Y las últimas constituyen entornos muy completos, integrándose con otras herramientas, lenguajes 4GL, generadores de código, etc.

3. Aspectos didácticos

3.1. Agentes involucrados y entorno

En todo proceso de "enseñar-aprender" intervienen distintos agentes en un entorno específico. En este sección presentamos en qué entorno y con características de agentes es que pensamos debe desarrollarse un curso asistido por herramientas CASE.

Como se muestra gráficamente en la Figura 1, el sistema didáctico es una relación ternaria que involucra los siguientes elementos: el enseñante, el enseñado (aprendiz) y el saber (conocimiento). Para que la enseñanza de un determinado elemento del saber sea posible, éste deberá haber sufrido ciertas deformaciones que permitan que sea enseñado. El "saber a enseñar" es necesariamente distinto al saber inicialmente diseñado como el que debe ser enseñado.

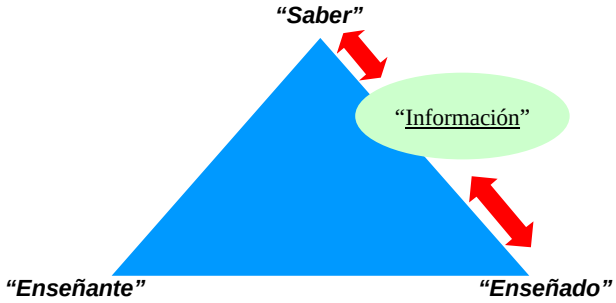

Figura 1 Elementos del sistema didáctico

El sistema didáctico [3] es un sistema abierto y su supervivencia supone una compatibilización con su entorno. Le impone responder a las exigencias que acompañan y justifican el proyecto social a cuya actualización debe responder. La compatibilización antes mencionada pasa por la disminución de la conciencia del entorno por parte de los agentes del sistema. Comúnmente, el saber enseñado vive encerrado sobre sí mismo, protegido por el distanciamiento del resto del mundo. En algunos casos, se revela una verdadera capacidad de producción de saber a los efectos del autoconsumo.

En general, los sistemas didácticos son formaciones que surgen con cada comienzo de cursos. Se forma un conjunto didáctico alrededor de un "saber" (ordinariamente designado por un programa o curricula), y se une a docentes y alumnos. El entorno inmediato está inicialmente formado por el "sistema de enseñanza" que reúne el conjunto de sistemas didácticos y tiene a su lado un conjunto diversificado de dispositivos estructurales que permiten el funcionamiento didáctico. Como se muestra en la Figura 2, el sistema de enseñanza posee a su vez un entorno, que está constituido básicamente por el entorno familiar del alumno, los académicos y la instancia política, decisional y ejecutiva (órgano de gobierno) [3].

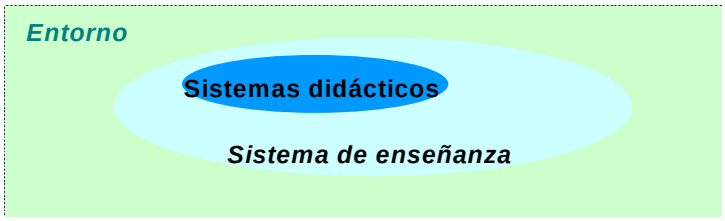

Figura 2: Entorno del sistema didáctico

3.2. Otros elementos considerados

Es importante diferenciar las actividades orientadas principalmente a la creación de capacidades del estudiante (formativas) de las actividades orientadas a la incorporación de conocimientos. Ambas instancias no son separadas en la enseñanza, sino que se sirven mutuamente.

También hay que considerar que se mantenga un balance entre la profundidad y la extensión de lo que se desea cubrir a través del apoyo de la herramienta. No hay que descuidar tampoco, el balance entre el "aprendizaje receptivo" y el "aprendizaje autodidáctico" según los objetivos planteados al organizarse los cursos.

La situación de los estudiantes se debe tomar en cuenta, así como las características y objetivos de cada carrera en particular.

Más que una sustitución (u oposición) se desea buscar una complementación entre lo que el estudiante recibe y lo que busca por si mismo. El rol de enseñante sigue estando vigente y es imprescindible para que se produzca la transposición didáctica.

Como se puede ver en la Figura 3, ante un problema planteado, y partiendo de la base de que existe por lo menos una solución válida a dicho problema, en todos los casos, vamos a aspirar a que el estudiante proponga una solución por lo menos (premisa pedagógica), con ayuda de la herramienta.

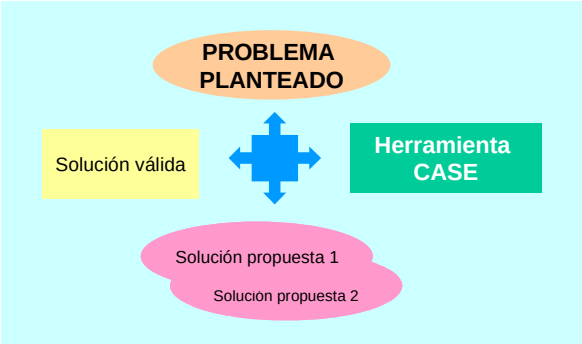

Figura 3: Elementos involucrados

Dado un problema propuesto, el estudiante propone una solución **1**:

a) La herramienta es capaz de proveer una solución **H** (la "correcta"). Entonces, existe una interfaz que compara la solución **H** con la **1** y produce como resultado un listado con las diferencias y sugerencias. Además, también puede mostrar la solución correcta con las aclaraciones o bien, de primera, o se obliga a que se propongan nuevas soluciones hasta llegar a la correcta o hasta que se considere que se intentó lo suficiente.

b) La herramienta es capaz de verificar la solución **1** y brindar la lista de problemas encontrados en el mismo. De este modo, el estudiante podría ingresar soluciones mejoradas (2, 3, etc.) hasta que en determinado momento se presenta la correcta (**H**). Pero en este caso, como la herramienta no puede realizar la comparación, no estaría validando (podría decir que la solución producida por el alumno no tiene errores, pero no era lo pedido, por lo que tampoco estaría correcta).

4. Marco evaluatorio

El criterio utilizado para la evaluación de las distintas herramientas se basa inicialmente en el propuesto por S. RAM (1994) [4]. Consiste en evaluar según dieciocho parámetros que detallamos a continuación:

1) *Origen:* Permite distinguir herramientas desarrolladas en entornos académicos de las creadas en entornos comerciales.
2) *Fases de desarrollo soportadas*
3) *Modelo / conceptos subyacentes:* Pueden ser semánticos, orientados a objetos, basados en formas, lenguaje natural, relacional, etc.
4) *Metodología / algoritmos utilizados:* Examina reglas o heurísticos empleados para producir un diseño.
5) *Entradas*
6) *Salidas*
7) *Medios de representación / Interfaces:* Gráficas o de texto.
8) *Repositorio de información*
9) *Documentación*
10) *Análisis de alternativas:* Si la herramienta presenta alternativas y ayuda al usuario a tomar decisiones.
11) *Verificación y validación del diseño*
12) *Público:* Usuarios finales, desarrolladores, etc.
13) *Validación / Difusión de uso*
14) *Características operativas:* Entorno de hardware, lenguaje de desarrollo.
15) *Facilidad de modificación*
16) *Facilidad de reutilización*
17) *Extensiones futuras*
18) *Comentarios generales*

De estos dieciocho puntos, a continuación presentamos una lista más detallada de las características y atributos de calidad a evaluar:

Atributos de calidad

- Usabilidad: Facilidad de aprendizaje, Facilidad de comprensión (interfaz intuitiva), Facilidad de operación.
- Portabilidad: Del modelo, Facilidad de instalación.

Funcionalidad

- Validación del modelo: Alcance de la validación, propuesta de alternativas / soluciones.
- Generación de diagramas a partir de bases de datos (ingeniería reversa): Bases de datos soportadas, Generación asistida / supervisada, Características de los diagramas generados, Restricciones de integridad.
- Generación de código y definición de datos a partir de los diagramas: Lenguajes soportados, Soporte para restricciones no estructurales, Bases de datos soportadas, Manejo de restricciones de integridad.

Simbología *(qué puede representarse y qué no)*

- Atributos clave
- Cardinalidad
- Relaciones múltiples
- Generalización
- Agregación
- Autorrelación
- Representación de entidades débiles

Capacidad de interacción con una base de datos

- Bases de datos soportadas
- Lenguajes de consulta

Dada una solución propuesta*:*

- ¿La herramienta es capaz de verificarla?
- ¿Cuál es la salida de dicha verificación?
    i. Listado de problemas
    ii. Listado de sugerencias
    iii. Listado de alternativas

Contando sólo con una solución, la herramienta sólo puede verificarla, no puede "validarla" ya que para ello debería ser "inteligente" y conocer el problema.

Dada una solución "correcta" y una solución propuesta:

- ¿La herramienta es capaz de verificar la solución propuesta?
- ¿Cuál es la salida de dicha verificación?
    i. Listado de problemas
    ii. Listado de sugerencias
    iii. Listado de alternativas
- ¿Es capaz de comparar ambas soluciones (validación)?
- ¿Cuál es la salida de dicha comparación?

Cuando existe una solución que se toma como la correcta, entonces puede validar, ya que lo haría por comparación.

5. Evaluación de Herramientas

A continuación se muestra un resumen de la evaluación realizada a las dos herramientas más importantes. Esta importancia radica en que de las herramientas relevadas, son las que poseen validación del modelo, y ofrecen prestaciones más allá de las de una aplicación de dibujo.

ER/Studio (Embarcadero) [5]

Esta herramienta se destaca por ser sencilla de aprender y operar. Además, ofrece una interfaz intuitiva y ayuda en línea completa y ordenada. Permite tanto la validación del modelo lógico y físico diseñados (validez de dependencias, relaciones, claves y límites soportados por la base de datos física).

Otras características incluyen la posibilidad de definición de restricciones no estructurales y el soporte de especificación de cardinalidades y existencia. Las notaciones utilizadas son la IDEF1X e IE (no implementa Codd) y puede representar los siguientes tipos de atributos:

- Clave primaria no heredada
- Clave primaria heredada
- Atributo no clave heredado
- Atributo no clave no heredado
- Claves foráneas

Como características avanzadas, encontramos la generación del modelo físico a partir del modelo lógico y viceversa, el soporte de diversos DBMS y la ingeniería reversa del modelo a partir de la base de datos. Permite generación de código JAVA para manejo a través de JDBC (genera el código necesario para realizar el mantenimiento básico de datos). Además, interactúa directamente con el DBMS a través de ODBC.

ER-Win (Computer Associates) [6]

ER-Win también es sencillo de aprender y operar, y ofrece una interfaz intuitiva y ergonómica al usuario. Como la herramienta anterior, también permite la validación del modelo lógico y físico (validez de dependencias, relaciones, claves y límites de la base de datos física). De la misma forma, permite la definición de restricciones no estructurales, la herencia de entidades, y soporta la especificación de cardinalidades y existencia. Soporta la misma notación y permite representar los mismos tipos de atributos que ER/Studio.

Entre otras características destacables, permite la organización automática de componentes del modelo (auto layout). Permite la generación del modelo físico a partir del modelo lógico y viceversa. También soporta diversos DBMS y ofrece la posibilidad de realizar ingeniería reversa del modelo a partir de la base de datos. Al igual que la herramienta anterior, interactúa con el DBMS a través de ODBC.

Comparando ambas herramientas, ER-Win ofrece más facilidades a la hora de diseñar y se destaca que soporta subtipos. ER/Studio es más complicado para distribuir asociaciones entre entidades y relaciones. Es una herramienta potente para la generación de aplicaciones y bases de datos e ingeniería reversa, y ofrece una ayuda más intuitiva.

La salida de la verificación de ambas herramientas mencionadas, no siempre es fácil de interpretar, y no contiene una "explicación" ni listado de sugerencias o alternativas. Tampoco permiten comparar distintas soluciones (distintos modelos que supuestamente deberían ser equivalentes dado que serían soluciones a un mismo problema).

6. Conclusiones

Las herramientas existentes relevadas no satisfacen completamente las aspiraciones expresadas en secciones anteriores. Si bien son herramientas potentes, por si solas no hacen de su uso una experiencia didáctica porque están orientadas al uso comercial.

Al evaluar cuán lejos estamos de la herramienta ideal, se desprende que es importante aprovechar las prestaciones de las herramientas existentes (para no reinventar la rueda), pero para que las mismas sean un apoyo a la enseñanza y no sólo una herramienta de "dibujo", es necesario potenciarlas y / o complementarlas.

A continuación se plantea una alternativa de complemento, como se muestra en el diagrama de la Figura 5:

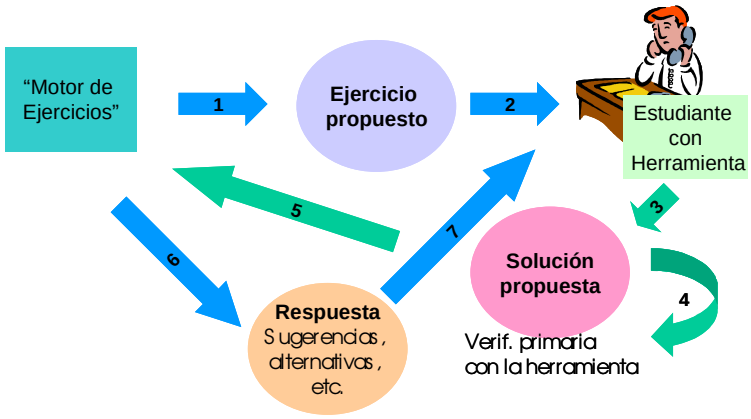

Figura 5: Alternativa para complementar las herramientas CASE relevadas

Partiendo de la base que existe un "proveedor de ejercicios", ante un ejercicio propuesto, el estudiante lo resuelve con la ayuda de la herramienta CASE. Con la misma realiza una primera verificación y si ésta es satisfactoria, envía la solución. El "proveedor" procesa la solución y devuelve un resultado (listado, alternativa, etc.). El estudiante, en base a esta información, mejora la solución inicial y la vuelve a enviar. Se itera hasta que la solución enviada por el estudiante es considerada satisfactoria o si se supera cierta cantidad de intentos (lo que motivaría que se le enviara una posible solución, por ejemplo).

Tómese en cuenta que hasta ahora no se ha mencionado quién es el "proveedor" ni el responsable de procesar las soluciones enviadas. Esta respuesta podría ser tanto "manual" (un docente "virtual") como automatizada.

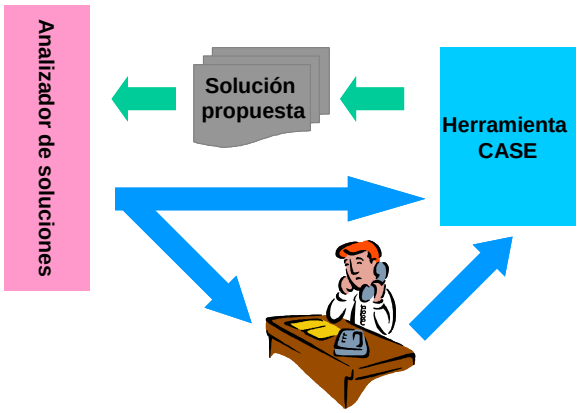

Figura 6: Rol del "analizador de soluciones"

Si el "analizador" es una aplicación, ésta puede estar asociada a la herramienta CASE (ejecutándose localmente en el equipo del estudiante) o puede ejecutarse en el servidor ("motor" o "proveedor").

Para ello es necesario que la herramienta CASE produzca una *salida* que pueda ser tomada como *entrada* por la aplicación. A su vez, la misma deberá generar una *salida* que pueda ser o no importada por la herramienta CASE (ver Figura 6). En el primer aspecto, se destaca ER/Studio dado que permite generar una "export" de la estructura en XML. Tanto ER-Win como ER/Studio tienen formatos propios que podrían ser analizados. Por otro lado, la aplicación podría generar un archivo que fuera importado por la herramienta CASE y a su vez, un archivo de texto con el listado de problemas, aclaraciones y sugerencias para el estudiante.

La complejidad de este "analizador", planteado como una caja negra, dependería de la profundidad y extensión que se quiera dar a los ejercicios. Resulta evidente que no es viable que contenga toda la inteligencia, pero si se toma desde el punto de vista de un manejador de contenidos, quedaría en manos de los docentes ingresar los mismos, y la lógica se limitaría a una comparación de contenidos (previamente procesados).

Agradecimientos

Agradecemos especialmente al Coordinador de Tecnología Educativa de Universidad ORT Uruguay, Dr. Alejandro Armellini, su aporte sobre el aspecto didáctico y a Agustín Cuervo, quien colaboró en el relevamiento primario de herramientas CASE.

Referencias

[1] URL: http://whatis.techtarget.com/definition/0,,sid9_gci213838,00.html
[2] DE MIGUEL, Adoración. PIATTINI, Mario. MARCOS, Esperanza. 2000. *Diseño de Bases de Datos Relacionales.* Madrid. Editorial Alfaomega. p. 420-427.
[3] CHEVALLARD, Yves. 1991. *La transposición didáctica. Del Saber Sabio al Saber Enseñado.* Buenos Aires. Editorial AIQUE.
[4] RAM, S. *Automated Tools for Database Design: State of the Art.* CMI Working Paper, Department of MIS, University of Arizona, 1994.URL: http://vishnu.bpa.arizona.edu/ram
[5] Embarcadero Technologies. *ER/Studio.*
URL: http://www.embarcadero.com/products/erstudio/index.asp
[6] Computer Associates. *AllFusion ERWin Data Modeler.*
URL: http://www3.ca.com/Solutions/Product.asp?ID=260