Capítulo 35

SOBRE EL DIBUJO DE DIAGRAMAS Y LAS HERRAMIENTAS

Las burbujas no explotan.
Bertrand Meyer

Objetivos

- Aprender consejos para dibujar los diagramas UML en un proyecto.
- Ilustrar algunas funciones comunes de las herramientas CASE para UML.

Introducción

En un proyecto real, la realización del análisis y diseño mientras se dibujan los diagramas UML no ocurre ordenadamente como en las páginas de un libro. Tiene lugar en el contexto de un equipo de desarrollo de software ocupado, que trabaja en oficinas o habitaciones, hacen garabatos en pizarras y quizás utilizando una herramienta, y a menudo con tendencia a querer comenzar a programar en lugar de trabajar con detalle algunos detalles mediante la realización de diagramas. Si la herramienta para UML, o el proceso para dibujar los diagramas, resultan molestos y engorrosos, o parece que tiene menos valor que la programación, se evitarán.

Este Capítulo propone mantener un equilibrio entre la programación y el dibujo de diagramas, y fomentar un entorno de soporte para hacer que la tarea de dibujar los diagramas sea cómoda y útil en lugar de difícil.

35.1. Diseño especulativo y razonamiento visual

Los diseños que se ilustran mediante diagramas UML serán incompletos, y sólo servirán como "trampolín" para la programación. Demasiados diagramas antes de programar nos lleva a una pérdida de tiempo en direcciones de diseño especulativas, o una pérdida de

rr

tiempo obsesionados con herramientas para UML. No hay nada como el código real para decirte qué funciona. Bertrand Meyer lo dice mejor: "Las burbujas no explotan".

No obstante, promuevo enérgicamente que se anticipe *alguna* idea mediante la realización de diagramas antes de programar, y se que puede ser útil, especialmente para explorar las estrategias de diseño más importantes. La pregunta interesante es "¿cuánto tiempo se tiene que dedicar a dibujar los diagramas antes de programar?" En parte la respuesta está en función de la experiencia y estilo cognitivo de los diseñadores.

Algunas personas son buenas para el razonamiento espacial/visual y expresar sus pensamientos acerca del diseño del software en un lenguaje visual complementa su naturaleza; otros no. Un gran porcentaje del cerebro está dedicado al razonamiento y procesamiento visual o simbólico, en lugar de al procesamiento textual (código). Los lenguajes visuales como UML juegan con una capacidad natural de la mente en la mayoría de la gente. A aquellos a los que se les ha enseñado UML obviamente les resultará más sencillo que a los que no. Y, en general, los diseñadores de objetos con más experiencia pueden diseñar de manera efectiva dibujando sin perderse en especulaciones no realistas, debido a su experiencia y juicio. Aplicados por expertos, los diagramas pueden ayudar a un grupo a moverse más rápidamente hacia un diseño apropiado, debido a la capacidad de ignorar los detalles y centrarse en los verdaderos problemas.

Una excepción a esta sugerencia de dibujar los diagramas "ligeros" son los sistemas que se modelan de manera natural como máquinas de estados. Hay algunas herramientas CASE que pueden hacer un trabajo impresionante generando código completo basado en máquinas de estados UML detalladas para todas las clases. Pero no todos los dominios encajan de manera natural en un enfoque fuertemente centrado en las máquinas de estados; por ejemplo, las máquinas de control y telecomunicaciones a menudo se ajustan bien, los sistemas de información de gestión normalmente no.

35.2. Sugerencias para dibujar los diagramas de UML en el proceso de desarrollo

Nivel de esfuerzo

Como guía, considere realizar los diagramas en parejas durante los siguientes periodos, antes de programar de manera seria en la iteración.

Iteración de 2-semanas	De medio día a un día casi al comienzo de la iteración (ej. lunes o martes)
Iteración de 4-semanas	Uno o dos días cerca del comienzo

En ambos casos, la realización de los diagramas no tiene que parar después de este esfuerzo inicial enfocado a este fin, los desarrolladores podrían dirigirse —idealmente en parejas— "a la pizarra" durante sesiones cortas para esbozar ideas antes de programar más. Y podrían realizar otra sesión más larga de medio día a mitad de la iteración, cuando tropiecen con un problema complejo en el ámbito de su tarea inicial, o terminen su primera tarea y pasen a la segunda.

Otras sugerencias

® Dibuje en parejas, no a solas. Lo más importante, la sinergia nos conduce a diseños mejores. En segundo lugar, la pareja aprende rápidamente técnicas de diseño el uno del otro y, por tanto, ambos llegan a ser mejores diseñadores. Es difícil mejorar como diseñador de software cuando uno diseña individualmente. Cambie de manera regular el compañero de dibujo/diseño para estar periodos de tiempo amplios expuestos al conocimiento de los demás.

® Para aclarar un punto al que se ha hecho alusión varias veces, en los procesos iterativos (como el UP), los programadores son también los diseñadores; no hay un equipo separado que dibuja los diseños y se los entrega a los programadores. Los desarrolladores se ponen su sombrero de UML, y dibujan un poco. Entonces se ponen su sombrero de programador e implementan, y continúan diseñando mientras programan.

[9] Si hay diez desarrolladores, suponga que hay cinco equipos de dibujo trabajando durante un día en pizarras diferentes. Si el arquitecto dedica tiempo a rotar por los cinco equipos, llegará a ver puntos de dependencia, conflictos e ideas de un equipo que sirven para otro. El arquitecto entonces puede actuar de enlace para llegar a una armonía entre los diseños y clarificar las dependencias.

[9] Contrate un escritor técnico para el proyecto y enséñele algo de la notación UML y los conceptos básicos del A/DOO (de manera que pueda entender el contexto). Disponga de la ayuda del escritor en la realización del "trabajo engorroso" con las herramientas CASE para UML, en los procesos de ingeniería inversa generando los diagramas a partir del código, en la impresión y presentación de grandes diagramas impresos con plotter, etcétera. Los desarrolladores dedicarán su tiempo (más caro) a hacer lo que hacen mejor: idear diseños y programar. Un escritor técnico les ayuda realizando la gestión de los diagramas, además de las verdaderas responsabilidades de la escritura técnica tales como el trabajo en el documento para el usuario final. Esto se conoce como el patrón Analista Mercenario [Coplien95a],

[9] Organice el área de desarrollo con muchas pizarras amplias distribuidas muy cerca unas de otras.

[9] Generalizando, maximice el entorno de trabajo para dibujar cómodamente en las paredes. Cree un entorno "amigable" para dibujar y colgar los diagramas. No puede esperar que se consiga una cultura de modelado visual con éxito en un entorno donde los desarrolladores se están peleando por dibujar en pizarras pequeñas de 60x90 cm, monitores de ordenador de tamaño corriente o trozos de papel. Para dibujar de manera cómoda hacen falta espacios muy amplios y abiertos —físicos o virtuales—. [9]

[9] Como accesorio de las pizarras, utilice finas hojas blancas de plástico con "adherencia estática" (vienen en paquetes de 20 o más) que se pueden colocar en las paredes; éstas se encuentran disponibles en muchas papelerías. Las hojas permanecen unidas a la pared mediante adherencia estática, y se puede utilizar como una pizarra con un rotulador que se pueda borrar. Se puede empapelar una pared con estas hojas para crear "pizarras" temporales, masivas. He dirigido grupos donde

hemos empapelado todas las paredes —de arriba abajo— de la sala del proyecto con estas hojas, y lo: encontramos de gran ayuda para la comunicación.

Si utiliza una pizarra para los dibujos de UML, utilice un dispositivo (existe al menos uno en el mercado) que capture los dibujos hechos a mano y los transmita a un ordenador como un fichero gráfico. Un diseño involucra una parte receptora en la esquina de la pizarra que captura la imagen para enviarla al ordenador y unas fundas especiales transmisoras en las que se insertan los rotuladores.

- De manera alternativa, si utiliza una pizarra para los dibujos de UML, utilice una cámara digital para capturar las imágenes, normalmente en dos o tres secciones. Ésta es una práctica para dibujar los diagramas, bastante común y efectiva.
- Otra tecnología de pizarra es una pizarra "que imprime", que normalmente es una pizarra de dos lados con un escáner y una impresora conectada. También son útiles.
- Imprima las imágenes UML dibujadas a mano (capturadas mediante una cámara o un dispositivo de pizarra) y cuélguelas de manera visible *muy* cerca de las estaciones de trabajo para la programación. Lo importante de los diagramas es inspirar la dirección de la programación, de manera que los programadores puedan echarles un vistazo mientras están programando. Si se dibujan pero "se entierran", no tenía mucho sentido dibujarlos.
- Si dibuja los diagramas UML a mano, utilice una notación simple elegida para acelerar y facilitar la realización de los diagramas.
- Incluso si lleva a cabo el diseño creativo en una pizarra, utilice una herramienta CASE para UML para generar los diagramas de paquetes y de clases mediante un proceso de ingeniería inversa a partir del código fuente (de la última iteración) por lo menos al comienzo de cada iteración posterior. Entonces, utilice estos diagramas generados mediante un proceso de ingeniería inversa como punto de partida para el siguiente diseño creativo.
- Imprima periódicamente los diagramas de paquetes y de clases interesantes/inestables/difíciles, generados recientemente mediante un proceso de ingeniería inversa, en un tamaño ampliado (para facilitar que se vea) con un plotter que pueda imprimir en papel continuo de de 90 ó 120 cm de ancho. Cuélguelos en las paredes muy cerca de los desarrolladores como ayuda visual. El escritor técnico, si está presente, puede hacer este trabajo. Anime a los desarrolladores a que dibujen y hagan garabatos sobre los diagramas durante el trabajo de diseño creativo.
- Con respecto a la ingeniería inversa, pocas herramientas para UML soportan el proceso de ingeniería inversa para generar diagramas de secuencia —no únicamente diagramas de clases— a partir del código fuente. Si está disponible, utilice una para generar los diagramas de secuencia para los escenarios significativos para la arquitectura, imprímalos a tamaño grande con el plotter, y cuélguelos para facilitar que se vean.
- Si utiliza una herramienta CASE para UML (de hecho, hágalo para todo el trabajo de programación), utilice una estación de trabajo con un monitor dual (dos monitores de pantalla plana de tamaño ordinario son más baratas que un único monitor

de pantalla plana de tamaño grande). L|s sistemas operativos modernos soportan (al menos) tarjetas de vídeo duales y del esta manera dos monitores. Organice sus ventanas dentro de la herramienta para UML entre los dos monitores. ¿Por qué? Un - monitor pequeño reprime **psicológicamente le y** creativament**emte** que refiere **a** - - los dibujos y lenguajes visuales porque el espacio del área visual es demasiado pequeño y estrecho. Un desarrollador puede caer en la aptitud desmotivada de "el diseño ha terminado porque la ventana está llena, y parece demasiado desordenado".

• Cuando utilice una herramienta CASE para UML y realice el diseño creativo por parejas o pequeños grupos, conecte dos proyectores de ordenador a las dos tarjetas de vídeo del ordenador y alinee las proyecciones en la pared de manera que el equipo pueda ver y trabajar con un espacio amplio para el área visual. Un área pequeña y unos diagramas difíciles de ver son impedimentos psicológicos y sociales para el diseño visual en colaboración de los pequeños grupos.

35=3» Herramientas y características de ejemplo

Este libro es imparcial respecto a la herramienta

Sería algo extraño no mencionar ninguna herramienta CASE *(Computer Aid Software Engineering,* ingeniería del software asistida por ordenador) para UML, porque este libro trata en parte de la realización de dibujos en UML, que tiene lugar con una herramienta CASE, o en una pizarra. Al mismo tiempo, no se pueden cubrir por igual todas las herramientas, y las evaluaciones apropiadas quedan fuera del alcance de este libro. Para ser imparcial:

Este libro no avala ninguna herramienta CASE para UML. Los siguientes ejemplos sólo sirven para ilustrar algunas de las características típicas y claves que podemos encontrar en las herramientas CASE para UML.

Las herramientas se ajustan de manera inconsistente a UML

Pocas herramientas dibujan toda la notación de UML correctamente, y conforme con la versión actual de la especificación de UML —realmente con cualquier versión—. Aunque esto estaría bien, no debería ser un factor en la elección de una herramienta, puesto que es mucho más importante su funcionalidad y facilidad de uso.

Ejemplo uno

En las Figuras 35.1 y 35.2 se utiliza Together de TogetherSoft para ilustrar y definir dos funciones claves de una herramienta CASE para UML: **ingeniería directa e inversa.** Estas funciones son lo que distinguen esencialmente una herramienta CASE para UML de una herramienta para dibujar.

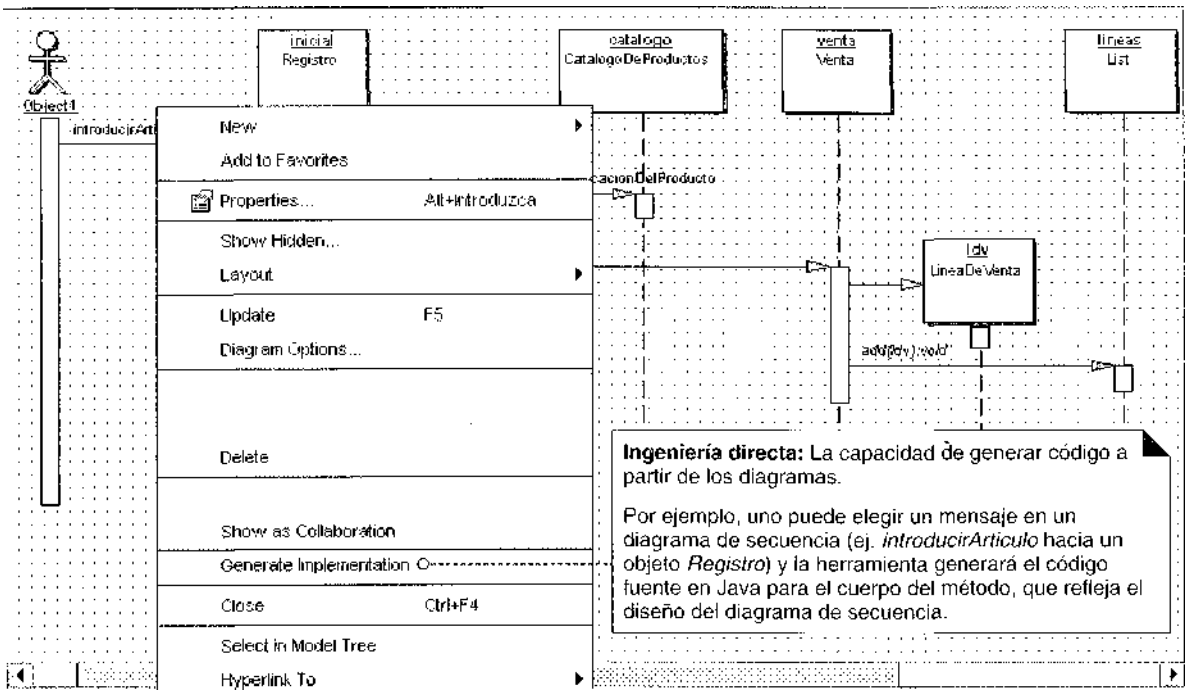

Figura 35.1. Ingeniería directa.

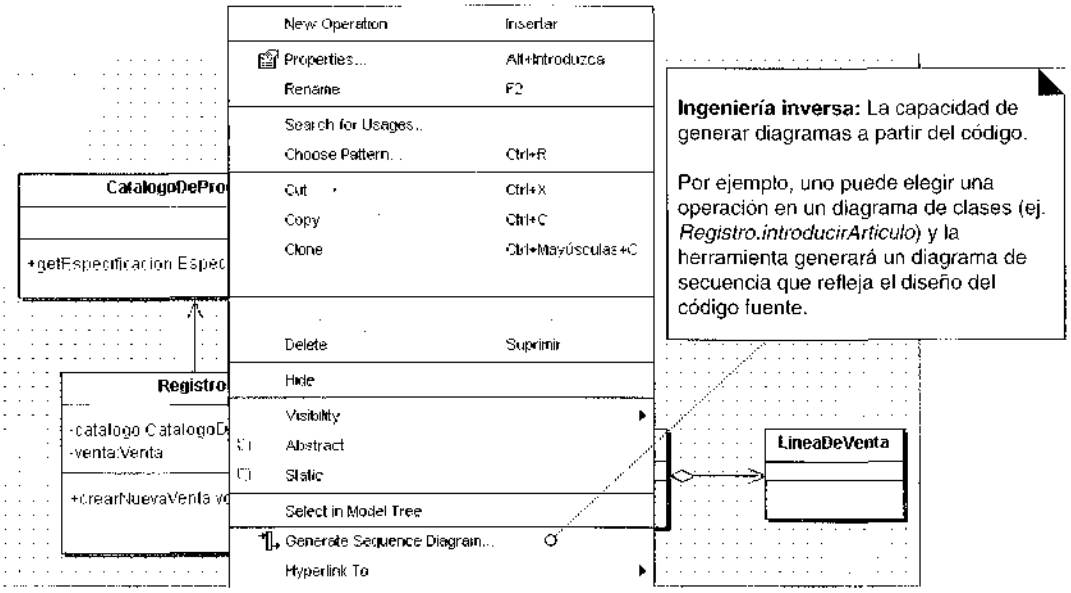

Figura 35.2. Ingeniería inversa.

35.4. Ejemplo dos

En las Figuras 35.3 y 35.4 se utiliza Rational Rose para ilustrar algunas otras funciones básicas de una herramienta CASE para UML.

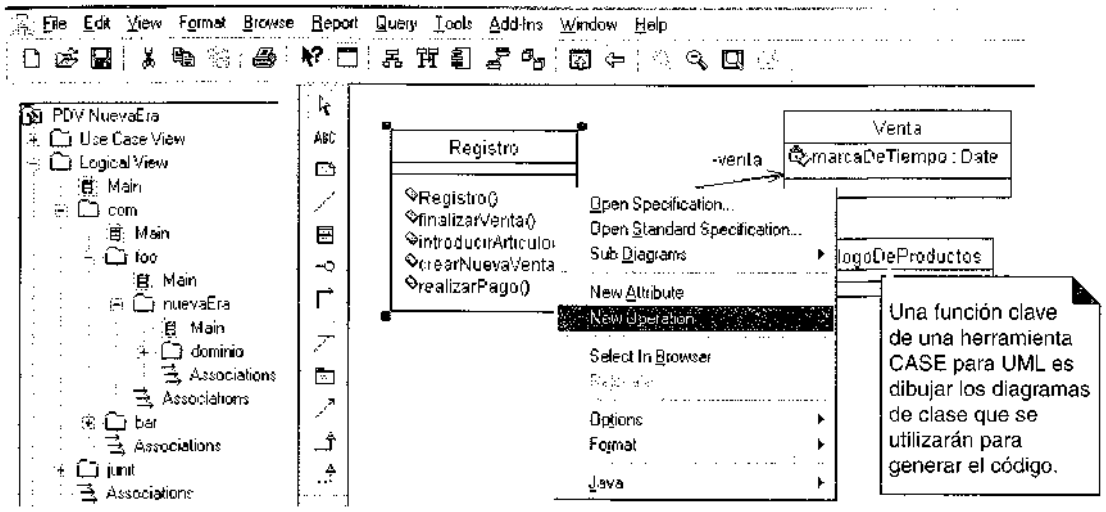

Figura 35.3. Creación de diagramas de clase.

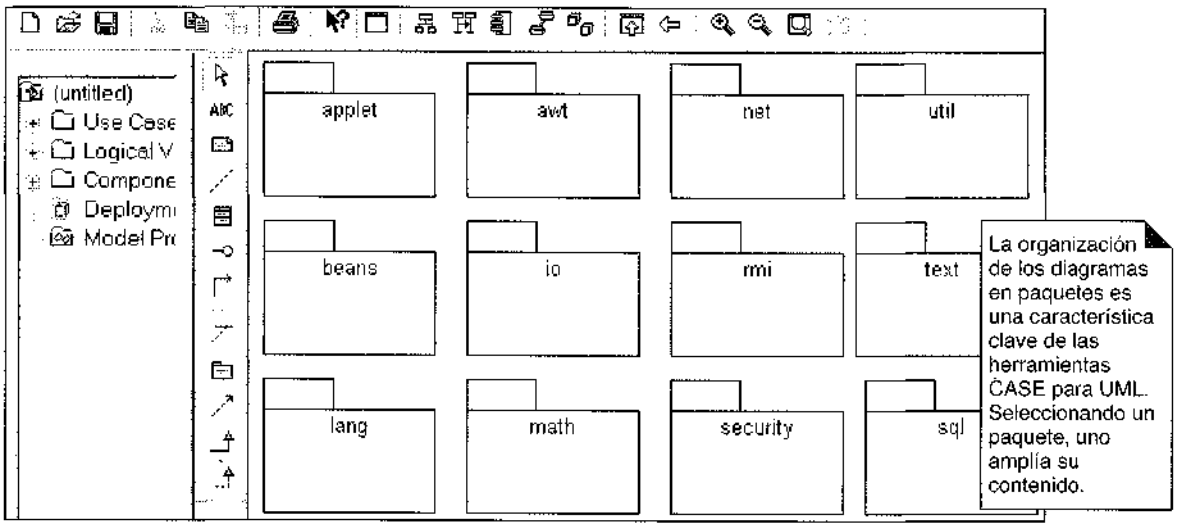

Figura 35.4. Gestión de paquetes.

Peticiones a los vendedores de herramientas CASE para UML

Sugiero que los consumidores hagan cuatro peticiones a los vendedores de herramientas CASE para UML:

1. Implementación correcta de la notación actual de UML en la herramienta.
2. Que el propio equipo de desarrollo de la herramienta CASE haya dibujado, leído y revisado seriamente diagramas UML (incluyendo el proceso de ingeniería inversa de los diagramas) en el proceso de construcción de la propia herramienta para UML.

1. Utilizar la versión N de la herramienta para UML para crear la versión N+l.
2. Proporcionar el soporte para los procesos de ingeniería directa e inversa de los diagramas de secuencia; la mayoría de las herramientas sólo lo soportan para el diagrama de clases.

Microsoft aboga por que los creadores de las herramientas “coman su propia comida para perros”. Buen consejo.