INGENIERÍA INVERSA. REUTILIZACIÓN DEL *SOFTWARE*

Javier AZNAR ALMAZÁN

Nunca pienso en el futuro. Llega enseguida.
Albert Einstein.

Por la calle del ya voy se va a la casa del nunca.
Miguel de Cervantes Saavedra.

Introducción

A reutilización es una estrategia a largo plazo en la que una organización construye una biblioteca de componentes usados con frecuencia, permitiendo montar rápidamente nuevos programas a partir de componentes ya existentes. Cuando está respaldada por un compromiso a largo plazo, la reutilización puede producir mayores ahorros de planificación y esfuerzo que cualquier otra técnica de desarrollo rápido. Además, puede usarse en prácticamente cualquier tipo de organización y para cualquier tipo de *software*. La reutilización también puede implementarse de forma oportunista como estrategia a corto plazo, recuperando código para un nuevo programa a partir de aplicaciones existentes. Este enfoque a corto plazo también puede producir ahorros significativos de planificación y esfuerzo, pero el ahorro posible es bastante menos importante, con demasiada frecuencia, que el obtenido con una reutilización planificada.

Podríamos pensar que la reutilización sólo se aplica al código, pero puede implicar la reutilización de cualquier elemento obtenido de un esfuerzo previo de desarrollo: código, diseños, datos, documentación, materiales de prueba, especificaciones y planes. En aplicaciones de sistemas de información, la planificación de la reutilización de los datos puede ser tan importante o más que la planificación para la reutilización del código.

Por otra parte, y aunque comúnmente no se vea como reutilización utilizar personal que ya ha trabajado en proyectos similares, es una de las formas más fáciles y potentes de reutilizar.

Dentro de este artículo se van a considerar dos tipos básicos de reutilización:

— Reutilización planificada.
— Reutilización oportunista.

La reutilización planificada es generalmente la mencionada por los expertos cuando hablan de reutilización; pero la reutilización oportunista también puede ofrecer reducciones de la planificación. Esta última contempla dos partes:

— Reutilización de componentes de origen interno.
— Reutilización de componentes de origen externo.

Generalmente, el uso de componentes externos no se ve como reutilización; suele verse como una decisión entre comprar y construir. Pero las cuestiones implicadas son similares, por lo que en este artículo se tratarán varios aspectos de este tema.

La reutilización, por último, acorta la planificación por la razón obvia de que normalmente es más rápido, fácil y fiable reutilizar algo que ya ha sido creado que crear algo nuevo partiendo de cero. Puede emplearse en proyectos de prácticamente cualquier tamaño, y resulta apropiada tanto para sistemas de gestión internos como para *software* de sistemas o *prêt-a-porter*.

Sobre la reutilización oportunista

Se utiliza cuando se descubre que un sistema existente tiene cosas en común con un sistema que se está a punto de construir. Entonces ahorra tiempo recuperando piezas del sistema existente, y utilizándolas en el nuevo sistema.

¿Adaptar o recuperar?

Si se desea hacer una reutilización oportunista, existen dos opciones: adaptar el viejo sistema para su nuevo uso, o diseñar el nuevo sistema desde el principio y recuperar componentes del antiguo sistema.

La experiencia demuestra que es mejor crear un nuevo diseño para el nuevo sistema y luego recuperar partes de sistemas existentes. Este enfoque funciona mejor porque sólo requiere que se comprendan pequeñas partes aisladas del antiguo sistema. Adaptar un antiguo programa para un nuevo uso requiere que se comprenda todo el programa, lo cual es una tarea bastante compleja.

El enfoque de recuperar partes es oportunista porque se necesita algo más que suerte para poder reutilizar diseño y codificación de un sistema anterior

sin haberlo planificado previamente. Puede tener éxito si el antiguo sistema estaba bien diseñado e implementado. Así, es de gran ayuda que el antiguo sistema utilice técnicas de modularidad y ocultación de la información. También puede tener éxito si hay solapamiento entre las plantillas de desarrollo del antiguo y del nuevo sistema. Sin algún solapamiento, aunque sea mínimo, intentar recuperar partes de un antiguo sistema puede resultar más bien un ejercicio de criptografía que de reutilización.

Sobreestimación del ahorro

El mayor problema de la reutilización oportunista consiste en que es fácil sobrestimar los posibles ahorros de esfuerzo y planificación. Aunque el antiguo sistema sea muy parecido al nuevo (digamos que podría ser reutilizado en un 80 por 100 del código) se tienen que considerar el análisis de requisitos, el diseño, la construcción, la prueba, la documentación y otras actividades en la planificación del esfuerzo.

Dependiendo de la situación específica se podría de utilizar el análisis de requisitos y el diseño o ninguno de ellos; si el antiguo proyecto no fue creado para ser reutilizado, no se debería contar con reutilizar ninguna de estas cosas. Si lo único que se puede reutilizar es el código, esta coincidencia del 80 por 100 del código puede reducirse a un 20 por 100 de disminución en el esfuerzo, y una disminución incluso menor en la planificación, aunque los sistemas sean similares en un 80 por 100.

Parte del trabajo de reutilización del 80 por 100 del código consistirá en saber cuál es ese 80 por 100 del código a reutilizar. Esto suele resultar decepcionalmente costoso, y puede anular rápidamente este posible ahorro del 20 por 100.

Otro problema común ocurre cuando se reutilizan partes del antiguo sistema de forma no prevista en el diseño e implementación de dicho antiguo sistema (nuevos errores emergen en el código del sistema antiguo). Cuando esto ocurre, si el equipo de desarrollo no está familiarizado con el antiguo sistema, que es lo más usual, se encontrará, de repente, depurando y arreglando código extraño, de forma que se anulará automáticamente el ahorro del 20 por 100.

Experiencia con la reutilización oportunista

Algunos proyectos que se han hecho con la reutilización oportunista han tenido éxito. Un proyecto realizado por el ejército francés tuvo una mejora del 37 por 100 de la productividad al recuperar código de un sistema similar existente (Henry y Faller, 1995). Los responsables del proyecto indicaron que el éxito del mismo se debía al uso de la modularidad y ocultación de informa-

ción en el primer sistema, a los ámbitos similares nuevo y antiguo y a tener aproximadamente la mitad de plantilla común en ambos sistemas.

El laboratorio de ingeniería del *software* de la NASA hizo un estudio de 10 proyectos que buscaban una reutilización agresiva (McGarry, Waligora y McDermott, 1989). Los proyectos iniciales no pudieron sacar mucho partido del código de proyectos anteriores, debido a que estos últimos no habían establecido una base de código suficiente. Sin embargo, en los proyectos subsiguientes, los que utilizaron diseño funcional pudieron aprovechar el 35 por 100 de su código de proyectos anteriores. Los que usaron diseño basado en objetos pudieron recuperar más del 70 por 100 de su código anterior.

Reutilización externa

No existe ninguna razón técnica para hacer el desarrollo propio de un componente externo ya preparado, pero sí puede haber una razón institucional. Tal como desear controlar una tecnología que es crítica para la institución. Desde un punto de vista meramente técnico es posible que un vendedor externo pueda poner más recursos en un componente prefabricado de los que se puedan disponer internamente y a un menor coste. Generalmente, los módulos de código reutilizables se ofrecen con un precio que oscila del 1 al 20 por 100 del esfuerzo que requeriría su desarrollo para el cliente.

Para muchas personas el mercado libre se está orientando rápidamente hacia la reutilización. Ha sido objeto de investigación industrial y académica durante años, y los vendedores llevan ofreciendo desde hace tiempo bibliotecas comerciales de código para dominios específicos de las aplicaciones, tales como interfaces de usuario, bases de datos, funciones matemáticas y demás.

Pero la reutilización comercial comenzó a imponerse en la plataforma PC con Microsoft Visual Basic y VBX (controles de Visual Basic). Reconociendo su popularidad, Microsoft ha reforzado el soporte de la reutilización con OCX (controles OLE). Las empresas de lenguajes de programación, como Borland, Gupta, Microsoft y Powersoft, soportan tanto VBX como OCX en los lenguajes que ofrecen.

Los problemas citados tradicionalmente como obstáculos para la reutilización también están siendo controlados por el mercado libre. Por ejemplo, las instituciones o las empresas no necesitan crear sus propios repositorios de componentes reutilizables porque los vendedores de VBX lo hacen por ellos. El éxito de los vendedores depende parcialmente de lo bien que organicen, cataloguen y den publicidad a sus productos para reutilización, de modo que los clientes puedan encontrar los componentes que necesiten y comprarlos.

Sobre la reutilización planificada

La reutilización planificada, como su nombre indica, es una estrategia (junto con unas acciones de mantenimiento *software)* a largo plazo que no ayudará en el primer proyecto que se utilice. A pesar de esto, no hay ningún otro método tan eficaz en la reducción de planificaciones de desarrollos a largo plazo.

Para comenzar un programa de reutilización, primero debe investigarse el *software* existente en la organización e identificar aquellos que pueden ser reutilizados.

Consideraciones de gestión

La reutilización vincula proyectos que anteriormente podrían haberse realizado de forma aislada, lo que amplía el ámbito de las decisiones sobre los proyectos. Implica que diferentes proyectos tendrán que estandarizar sus procesos *software*, lenguajes y herramientas. La reutilización efectiva requiere una inversión a largo plazo en formación y un plan multiproyecto para la construcción y el mantenimiento de componentes reusables. Estas cuestiones plantean todo un desafío, y prácticamente todos los estudios sobre programas de reutilización o informes sobre un programa específico dicen que el compromiso de la institución es mucho más importante para el éxito que la capacidad técnica.

Éstas son algunas de las tareas de gestión implicadas en la dirección de un programa de reutilización:

— Designar un responsable de la institución (pero entre los altos directivos) para el programa de reutilización. Los directivos de primer nivel no suelen tener incentivos para el desarrollo de componentes reutilizables en un único programa, ya que esto repercute sobre la mala imagen del proyecto, y no hay ningún incentivo para que el siguiente proyecto salga mejor.
— Asegurar un compromiso a largo plazo con el programa. Es importante que la reutilización no sea tratada como algo pasajero.
— Hacer que la reutilización sea una parte integral y explícita del proceso de desarrollo. La reutilización no aparecerá por sí sola y tampoco aparecerá como subproducto de otros buenos procesos. Para dejar clara esta prioridad, el soporte de la reutilización se debe convertir en una actividad más de los usuarios.
— Transformar los programas de medida de productividad del *software* de la organización para que en lugar de medir el *software* desarrollado midan el *software* entregado. Esto ayudará a asegurar que los desarrolladores son recompensados por el uso de componentes reutilizados.

— Establecer un grupo de reutilización separado para encargarse de las tareas relativas a los componentes reusables.
— Proporcionar formación al grupo de reutilización y a los posibles clientes del grupo.
— Difundir en la organización la existencia de la iniciativa de la reutilización mediante una activa campaña de difusión.
— Crear y mantener una lista formal de las personas que están reutilizando componentes para que puedan ser notificadas en el caso de que surjan problemas con los componentes que están usando.
— Preparar un sistema de control de coste o compensación para el uso de componentes de la biblioteca que sean componentes reusables.

Uno de los aspectos más difíciles de la preparación de un programa de reutilización consiste en que es difícil hacer que ésta tenga éxito sin el apoyo de otras técnicas claves de desarrollo, tales como el control de calidad y la gestión de configuración.

Consideraciones técnicas

El grupo encargado del cuidado y recogida de componentes reutilizables tendrá mucho trabajo. Éstas son algunas de sus tareas:

— Evaluar si las arquitecturas actuales del *software* de la organización soportan la reutilización, y desarrollar arquitecturas que la soporten si es necesario.
— Evaluar si los estándares actuales de codificación soportan ésta y recomendar nuevos estándares si es apropiado.
— Definir estándares de lenguajes de programación e interfaces que soporten la reutilización. Es prácticamente imposible reutilizar componentes si están escritos en lenguajes distintos, usan interfaces de llamada a funciones completamente distintas y usan diferentes convenios de codificación.
— Definir procesos que soporten la reutilización.
— Crear una biblioteca formal de componentes reutilizables y ofrecer algún medio para examinar esta biblioteca buscando información de reutilización.

Además de estas tareas generales, el grupo necesitará realizar trabajos de diseño, implementación y control de calidad en los propios componentes reusables.

Centrarse en componentes específicos del dominio

Los programas de reutilización que tienen más éxito se centran en la construcción de componentes reutilizables para dominios específicos de la aplicación. Se debe intentar centrar la reutilización en el *dominio de la aplicación* o *componente de gestión.*

Crear componentes pequeños y simples

Es mejor centrar el trabajo de reutilización en la creación de componentes especializados, pequeños y simples, que en componentes grandes y generales. Los desarrolladores que intentan crear *software* reutilizable, creando componentes generales, rara vez se anticipan adecuadamente a las necesidades futuras de los usuarios. Los futuros usuarios verán el componente grande y complicado, comprobarán que no cumple todas sus necesidades y decidirán no utilizarlo en absoluto. *Grande y complicado* implica *demasiado difícil de comprender,* y esto se traduce en *propenso a errores en su uso.* Los datos obtenidos por el laboratorio de ingeniería del *software* de la NASA sugieren que, si un componente necesita ser modificado en más de un 25 por 100, cuesta lo mismo desarrollar uno nuevo que reutilizar el existente (NASA, 1990).

Es mejor dividir el componente grande en componentes pequeños y centrar los esfuerzos de reutilización sobre estos últimos. Utilizando un lenguaje de diseño estructurado se debe *factorizar completamente* cualquier componente que intente reutilizar. No se deben presentar los grandes componentes como entidades individuales y monolíticas. Con componentes más pequeños y precisos se proporcionará la esperanza de resolver el problema de los futuros usuarios de una vez, aunque de todos modos esta esperanza es generalmente poco realista. Al menos, con componentes más pequeños se mejora la posibilidad de entregar algo útil.

Centrarse en la ocultación de la información y la encapsulación

Otra de las claves para tener éxito es centrarse en la ocultación de la información y la encapsulación, el núcleo del diseño basado en objetos.

El diseño orientado a objetos añade las ideas de herencia y polimorfismo al diseño basado en objetos, pero la incorporación de estas características no parece facilitar el logro de la reutilización. La clave real para obtener el éxito es la ocultación de la información entre módulos y la encapsulación.

Crear una buena documentación

La reutilización requiere también una buena documentación. Cuando se crea un componente reutilizable no sólo se está creando un programa.

Un tipo de documentación que resulta especialmente valiosa para la reutilización es una lista de las limitaciones conocidas de cada componente. A menudo, el *software* se entrega con defectos conocidos, debido básicamente a que algunos errores tienen una prioridad muy baja como para ser corregidos. Pero un defecto de baja prioridad en un contexto puede convertirse en uno de alta prioridad cuando el componente es reutilizado en otro programa. No se debe permitir que los usuarios descubran los defectos por sí mismos cuando ya se conocen de antemano.

Construir componentes reutilizables libres de errores

Una reutilización con éxito requiere la creación de componentes que estén virtualmente libres de errores. Si un desarrollador que intenta emplear un componente reutilizable encuentra que contiene defectos, el programa de reutilización perderá rápidamente su brillo. Un programa de reutilización basado en programas de baja calidad puede incrementar el coste global del desarrollo *software*. En vez de pagarle al desarrollador original para que depure completamente un componente, se le pagará a un desarrollador que no está familiarizado con el mismo para depurarlo, probablemente de un modo menos eficiente.

Gestión de los riesgos de la reutilización

Generalmente, los programas de reutilización mejoran la calidad y la productividad con un menor coste. A nivel de proyecto, la reutilización tiende a reducir el riesgo porque hay que codificar a mano menos partes del sistema y la calidad de los componentes reutilizados generalmente es más alta que la de los componentes desarrollados en el acto. Sin embargo, a nivel de la organización, aparece un conjunto de riesgos alrededor de la dificultad de predecir cuáles son los componentes que se necesitará reutilizar.

Esfuerzo desperdiciado

La creación de componentes reutilizables cuesta dos o tres veces más que la creación de un componente normal. Una vez desarrollado el componente se ha de usar como mínimo dos o tres veces para quedar a la par. Esto se conoce

como la regla del tres: antes de obtener ningún beneficio por la reutilización de un componente hay que reutilizarlo tres veces. Si no se está seguro de que el componente va a ser reutilizado tres veces, podría seguir teniendo sentido tener componentes preconstruidos disponibles, desde el punto de vista de un desarrollo futuro, pero esta velocidad futura requerirá su precio.

Esfuerzos mal dirigidos

Si se define un grupo separado para desarrollar componentes reutilizables existe la posibilidad de que este grupo desarrolle componentes que nunca se han utilizado. Si el grupo de reutilización realiza hipótesis incorrectas y uno de cada cuatro de sus componentes nunca es utilizado, tendrá que planificar usar cada componente reutilizado como mínimo cinco veces para equilibrar el coste. Dentro de una organización que no desarrolla una gran cantidad de sistemas realmente parecidos, este punto de equilibrio puede resultar difícil de alcanzar.

Cambios en la tecnología

Uno de los riesgos de la reutilización planificada se debe al hecho de que es una estrategia a largo plazo. No sólo se necesita reutilizar un componente varias veces para recuperar la inversión, sino que se necesita usarlo antes de que la tecnología en la que está basado quede obsoleta.

Sobreestimación del ahorro

No se debe suponer que la reutilización de código ahorrará mucho tiempo. Si solo se reutiliza código, se ahorrará tiempo de codificación. Si se reutilizan otros elementos del proyecto, se podrá ahorrar tiempo en otras facetas.

Se debe tener en cuenta que reutilizar un componente tiene un coste debido al tiempo necesario para localizarlo y aprender a usarlo. Se puede planificar emplear para la reutilización de una línea de código alrededor de una quinta parte del esfuerzo necesario para desarrollarla desde cero.

Conclusiones

Sobre la reutilización oportunista

Para asegurar el éxito en su utilización es necesario:

— Sacar partido de la continuidad de personal entre antiguos y nuevos programas.
— No sobreestimar el ahorro que se va a alcanzar.

Sobre la reutilización planificada

Para asegurar el éxito en su utilización es necesario:

— Un compromiso seguro y a largo plazo de los altos directivos con el programa de reutilización.
— Hacer de la reutilización una parte integral del proceso de desarrollo.
— Establecer un grupo de reutilización separado, cuyo trabajo sea identificar candidatos a componentes reutilizables, crear estándares que soporten la reutilización y diseminar información sobre componentes reutilizables a los posibles usuarios.
— Centrarse en componentes pequeños, sencillos y específicos del dominio.
— Centrar los esfuerzos de diseño en la ocultación de la información y la encapsulación.
— Llevar la calidad de los componentes hasta el nivel de productos para generar una buena documentación y asegurar que estén libres de errores.