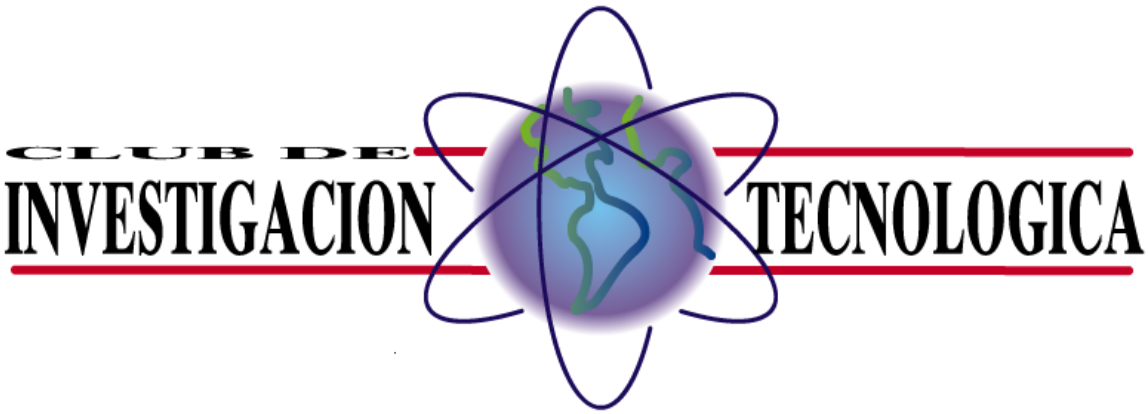


Transformación de Aplicaciones Legacy

Preparado por: Declan Good
Agosto 2002

Editado y Publicado por
Club de Investigación Tecnológica S.A.

Copias adicionales gratuitas con
leda@cit.co.cr o en www.artinsoft.com
San José, Costa Rica

Acerca del Autor

Declan Good cuenta con muchos años de experiencia en la maximización del valor por medio de la inversión en tecnología. Fue Jefe de Planeamiento e Investigación de Informática en el "Canadian Customs and Excise" (Aduana Canadiense) , luego su carrera se ha concentrado en proyectos de inversión en tecnología y de arquitectura técnica en Canadá y el Reino Unido.

Trabajó con firmas de consultoría como Woods Gordon en Ottawa (actualmente Ernst & Young) después de dejar el Gobierno Canadiense, y luego trabajó con consultores en estrategia de Tecnología de la Información Butler Cox en el Reino Unido. Ha laborado como Consultor Independiente desde 1992. Declan tiene un título en ingeniería de Carleton University, Ottawa, y el University College, Dublin. Puede ser contactado en la siguiente dirección: declan_good@compuserve.com

Reconocimientos

Este proyecto de investigación fue desarrollado para el Club de Investigación Tecnológica en el Reino Unido con el patrocinio de ArtinSoft.

Quisiera agradecer a Carlos Arraya y a Roberto Sasso de ArtinSoft por su ayuda y apoyo en el desarrollo de esta investigación. Donal Daly de Oracle, Rod McKenzie de ArtinSoft, y Martin Langham de Charteris hicieron considerables aportes a la misma. Me gustaría agradecer también a todas aquellas personas y compañías que me brindaron información y asesoría. Esta es una traducción de Dylia Sasso

Índice

Sinopsis

Sinopsis

Una aplicación legacy es una aplicación basada en tecnologías y hardware más viejos, tales como mainframes, la cual continua brindando servicios esenciales para una organización. Las aplicaciones Legacy muchas veces son grandes, monolíticas y difíciles de modificar, y desecharlas o reemplazarlas, en general, implica aplicar reingeniería a los procesos de la organización. La transformación de aplicaciones legacy se trata de la retención y extensión del valor de la inversión en Legacy por medio de la migración a nuevas plataformas.

La re-implementación de aplicaciones en nuevas plataformas de esta manera puede reducir costos operativos y las capacidades adicionales de las nuevas tecnologías pueden brindar acceso a valiosas funciones tales como Servicios Web y Ambientes Integrados de Desarrollo. Una vez finalizada la transformación, las aplicaciones pueden adecuarse más a las necesidades futuras de las empresas agregando nueva funcionalidad a la aplicación transformada.

En resumen, el proceso de transformación de aplicaciones legacy puede ser efectivo en cuanto a costos y puede ser una manera precisa de preservar las inversiones legacy y por tanto evitar los costos e impactos comerciales de la migración hacia un software totalmente nuevo. Este estudio explica cómo funciona la transformación y propone una estrategia para evaluar la idoneidad de las aplicaciones existentes para la migración a plataformas modernas tales como J2EE y .NET.

El objetivo de la transformación de aplicaciones legacy es mantener el valor del activo legacy en la nueva plataforma. En la práctica esta transformación puede tomar diversas formas. Por ejemplo, puede involucrar la traducción de código fuente, o cierto grado de reutilización del código existente más una capacidad de conectar el sistema legacy a la web para brindarle al cliente el acceso requerido por la empresa. Si fuese necesario re-escribir, entonces se podría extraer las reglas existentes de la empresa para que formen parte de la declaración de requisitos para una re-escritura.

El estudio adopta la posición de que el J2EE o .NET son plataformas meta adecuadas para la transformación. Los argumentos a favor se basan en factores técnicos y de costos, en el hecho de que la mayoría de los productos de traducción automática se dirigen hacia estas plataformas, en una creciente oferta de expertos en J2EE y .NET, facilitando así el reclutamiento de personal, y en la disponibilidad de protocolos estándar basados en XML para ser utilizados con otras aplicaciones, lo que facilita la publicación de la función de aplicación a una red (por lo general denominada “Servicios Web”).

Hoy día es factible automatizar una parte considerable de este proceso de transformación, convirtiendo la transformación en una propuesta económica atractiva en comparación con re-escritura o reemplazo la aplicación legacy. Las herramientas disponibles cubren todos los aspectos del proceso aunque se requiere cierta intervención manual. La utilización de herramientas en la práctica dependerá de la escala de la tarea y de si la automatización es necesaria o económica en todos los casos. Se asume por supuesto que las aplicaciones existentes poseen una calidad suficiente y satisfacen las necesidades de las empresas de manera adecuada como para que valga la pena su transformación.

Las herramientas disponibles en el mercado son soluciones específicas – son aplicables a escenarios específicos y manejan únicamente parte del proceso de transformación. Por consiguiente, existirá una necesidad de comprar servicios para ayudar a diseñar y ejecutar la transformación ya que hay demasiadas incógnitas que superar sin la ayuda de expertos con experiencia previa. Los recursos internos serán necesarios para impartir los conocimientos disponibles acerca de las aplicaciones legacy, y para crear la base futura de conocimiento para mantener las aplicaciones transformadas y alineadas con las necesidades de las empresas.

La transformación de aplicaciones legacy es una tarea que implica tanto riesgo como recompensa. Es fácil caer en la trampa de utilizar aplicaciones que parecen ser estables y tener la esperanza de que sean adecuadas para seguir adelante con la empresa al menos a mediano plazo. Pero estas aplicaciones legacy se encuentran en el centro de las operaciones de hoy en día y si se alejan mucho de las necesidades de las empresas, el impacto será considerable, y posiblemente catastrófico. El reto para el Gerente de Sistemas es presentar los argumentos para la inversión en legacy de la mejor forma posible, pero a la vez darle a la administración el panorama completo de estos riesgos y recompenses de manera que estos puedan tomar una decisión con una amplia disposición de los hechos. Por último, la transformación de aplicaciones legacy es un proyecto 'habilitador', que permite que sucedan otras cosas pero a su vez tiene sus propios beneficios directos.

Al escoger un proveedor o proveedores para un proyecto de transformación, es mejor alcanzar un balance entre los participantes orientados hacia el proyecto (quienes se harán cargo de la transformación en sí), y los proveedores de infraestructura y el personal interno quienes deben lograr que el resultado final funcione cada día. Con los últimos adelantos, los conocimientos especializados en transformación deben ser el centro de la solución. Esto puede lograrse asignando un gerente de proyecto independiente (interno o contratista) y manteniendo los cambios funcionales y el trabajo de integración separados de las tareas de traducción de códigos, migración de datos y pruebas relacionadas.

En conclusión, las herramientas y técnicas automatizadas disponibles hoy en día hacen que la transformación de aplicaciones legacy sea técnica y económicamente factible. El reemplazar o re-escribir es necesario en ciertos casos, pero si la aplicación legacy existente satisface las necesidades actuales de la empresa y la calidad es buena, entonces es posible que el activo pueda ser transformado de manera efectiva para seguir satisfaciendo las necesidades de la empresa en el futuro.

Conclusiones

- No siempre es necesario desechar o reemplazar las aplicaciones legacy. La transformación es una opción factible si las aplicaciones actuales son de buena calidad y satisfacen razonablemente las necesidades de la empresa.
- Hay herramientas automatizadas disponibles para migrar la mayoría de los datos y códigos a plataformas modernas.
- Si se necesita re-escribir por completo, existen herramientas disponibles para ayudar a extraer las reglas existentes de la empresa para utilizarlas como aporte para la re-escritura.
- No ignore las aplicaciones legacy con la esperanza de que desaparezcan. Lleve a cabo una auditoria regular sobre "que tan aptas son con respecto al propósito".
- Avance en los Servicios Web: La evolución de los Servicios Web pondrá mayor presión para ordenar las aplicaciones legacy y hacerlas más accesibles a los clientes y los socios comerciales. Comience a planear desde ahora.
- La transformación requiere tanto planeamiento y participación de la empresa como cualquier otro proyecto de TI.
- Involucre al personal técnico interno en el proyecto de transformación para asegurarse de obtener una transferencia de conocimientos - de lo contrario la organización puede encontrarse nuevamente donde comenzó, con aplicaciones que nadie entiende y que son imposibles de mantener.
- No espere que el personal técnico interno esté contento con la transformación – la mayoría va a preferir re-escribir la aplicación legacy o reemplazarla con el paquete de aplicación más reciente.
- No intente llevar a cabo la transformación usted solo la primera vez - escoja un proveedor con suficiente experiencia en transformación para que trabaje con usted.

1. Introducción

Una aplicación legacy puede ser definida como cualquier aplicación basada en tecnologías y hardware más viejo, tales como mainframes, la cual continua brindando servicios esenciales a una organización. Las aplicaciones legacy frecuentemente son grandes, monolíticas y difíciles de modificar, y el desechar o reemplazar una aplicación legacy muchas veces implica aplicar también reingeniería a los procesos comerciales de la organización.

Este estudio trata de explicar la alternativa de transformación de aplicaciones legacy, la cual mantiene y extiende el valor de la inversión en legacy por medio de la migración hacia nuevas plataformas, y a la vez limita la necesidad de aplicar reingeniería a los procesos comerciales existentes. Este incluye propuestas para una estrategia legacy y examina_aspectos relacionados con el planeamiento y justificación de costos de la transformación. La audiencia a la que está dirigida el estudio incluye Gerentes de Sistemas y sus subalternos directos, integradores de sistemas, y proveedores de paquetes de aplicación comerciales en existencia basados en tecnologías anteriores.

Puntos importantes abordados en este estudio

1) ¿Qué es la transformación de aplicaciones legacy?
2) ¿Cómo puede ayudar la transformación de aplicaciones legacy a enfrentar las presiones comerciales para obtener una mayor funcionalidad, capacidad de repuesta ante el cambio y mejoras en la efectividad en cuanto a costos?
3) ¿Qué tan factible es, y será este el momento correcto para llevarla a cabo?
4) ¿Cuáles estrategias debe adoptar Sistemas de Información para las aplicaciones legacy?
5) ¿Cuáles son los componentes de un proyecto de transformación de aplicaciones legacy?
6) ¿Cómo deben administrarse los proyectos de transformación de aplicaciones legacy?
7) ¿Cómo se puede financiar la transformación de aplicaciones legacy?
8) ¿A quién debe acudir el comprador para obtener asistencia?

Organización del estudio

El capítulo 2 considera la motivación básica para la transformación de aplicaciones legacy, y en el Capítulo 3 aparece una descripción del proceso. El Capítulo 4 brinda una guía sobre cómo tomar una decisión acerca de cuándo es el momento de hacer la transformación, y cuándo desechar o reemplazar la aplicación legacy (o en algunos casos, cuándo hacer una mayor inversión). En particular, se hace un repaso acerca de la dificultad de escoger una plataforma meta. Los capítulos 5 y 6 tratan acerca del planeamiento del proyecto y desarrollo de casos, respectivamente. El capítulo 7 discute

las opciones del lado de la oferta y brinda algunas recomendaciones acerca de cómo escoger socios comerciales para obtener asistencia en la transformación. Finalmente, el capítulo 8 brinda un panorama general de la transformación. Se incluye la bibliografía utilizada y sugerida como lectura adicional. El anexo contiene un glosario breve de abreviaciones y términos utilizados en este estudio.

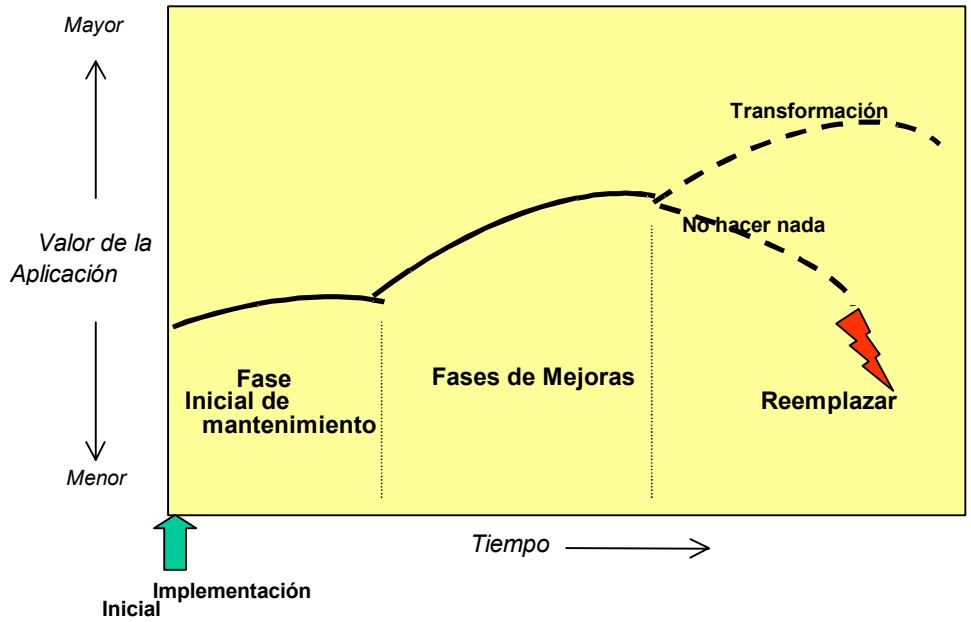
2. El Caso para la Transformación de aplicaciones legacy

Las aplicaciones existentes son el resultado de inversiones de capital hechas en el pasado. El valor de la inversión en la aplicación tiende a bajar con el tiempo conforme el contexto empresarial y tecnológico cambia. Al inicio del ciclo de vida habrá inversiones en mejoras para mantener una alineación cercana con la empresa pero eventualmente llegará a un punto en que esto se hará difícil. Esto puede suceder, por ejemplo, si la infraestructura es reemplazada, si se requiere acceso a la web, o el peso de los cambios en las aplicaciones y la falta de know-how (conocimiento y experiencia) hacen que sea imposible continuar haciendo mejoras.

La insatisfacción con legacy se centra en su falta de flexibilidad (toma mucho tiempo hacer cambios, no se pueden hacer cambios), mantenimiento (no hay documentación, nadie lo entiende, escasez de personal capacitado), accesibilidad (no se puede poner a disposición de los clientes, por ejemplo), costo de operación (corre en infraestructura de macrocomputadora muy costosa, altos costos para las licencias), e interdependencia de aplicación e infraestructura (no se puede actualizar una sin la otra).

En este momento tenemos una opción: ¿Debemos iniciar un proceso de renovación y transformación, o debemos eliminar la aplicación y encontrarle un reemplazo?

Ciclo de Vida de la Aplicación – Adónde dirigirse?



La transformación de aplicaciones legacy tiene que ver con el mantenimiento y extensión del valor de la inversión en legacy por medio de la migración hacia nuevas plataformas. El re-implementar aplicaciones en nuevas plataformas puede traer beneficios al reducir costos operativos, y por medio de las capacidades adicionales de las nuevas tecnologías brinda acceso a valiosas funciones a través de medios más económicos. La migración a una nueva plataforma también brinda la oportunidad de alinear aplicaciones con las necesidades actuales y futuras de las empresas agregando funcionalidad comercial y reestructurando la aplicación.

Los impulsores para la transformación legacy son las reducciones en costos operativos, funciones y adquisiciones, reorganización interna, nueva infraestructura corporativa, necesidad de habilitación para la Web, desempeño y funcionalidad desactualizados, consolidación de datos, y posicionamiento para cambios futuros tales como B2B vía XML y SOAP (Servicios Web).

Los Servicios Web pueden convertirse en un factor clave para forjar el cambio en las aplicaciones legacy. La mayoría de las organizaciones se encuentran en lo que se denomina la etapa de "fase 1" del planeamiento de Servicios Web- quizás haciendo pruebas o simplemente asesorándose en cuanto a la importancia de este concepto para el futuro. Algunas se encuentran en la "fase 2" y ya están explotando las capacidades de integración, muchas veces internamente en su organización. De forma realista, los servicios Web pueden convertirse en un asunto estratégico a corto plazo, aumentando así la urgencia para actuar sobre las aplicaciones legacy que se encontrarán en el centro de la infraestructura del Servicio Web.

La posición en este estudio es que las herramientas y técnicas para soportar la transformación automatizada son tales que la migración es factible tanto técnica como económicamente. El reemplazar y re-escribir es necesario en ciertos casos, pero si la aplicación legacy existente satisface las necesidades actuales de las empresas, entonces es muy probable que este activo legacy pueda ser transformado efectivamente para continuar satisfaciendo las necesidades de las empresas en el futuro.

¿Cuándo se convierte una Aplicación en una Aplicación Legacy?

" Una aplicación legacy es una que ha permanecido con la empresa más tiempo que los programadores que la mantienen, no tiene buena documentación, y contiene código intocable" *Joe Celko* , IT Writer

"¿Cuál es la definición de una aplicación legacy ? Respuesta: Una que funciona." *Amey Stone,* Business Week

"Por supuesto, la definición real de una aplicación legacy es una aplicación que no es dependiente de la Internet." *Amey Stone,* Business Week

"Legacy, en un contexto de TI, usualmente se refiere a una aplicación de macrocomputadora, aunque más recientemente incluso a algunas aplicaciones cliente/servidor les ha sido otorgado este honor equivocadamente." *The Butler Group,* Analistas Europeos de TI

"La gente relaciona el término legacy con mainframes y con IBM, pero la frase se usa cada vez más para incluir toda aplicación que ha existido desde antes del nacimiento de la Web". *Sarah L Roberts-Witt,* Escritora sobre Infraestructura y Servicios de la Internet

"Aunque una aplicación de información puede comenzar su vida con una arquitectura flexible, repetidas olas de piratería informática tienden a petrificar las aplicaciones maduras de información... Una aplicación que haya experimentado la petrificación es denominada una aplicación legacy." *Anthony Lauder, Stuart Kent,* Consultor, Stuart Kent, University of Kent en Canterbury

3. Proceso de Transformación

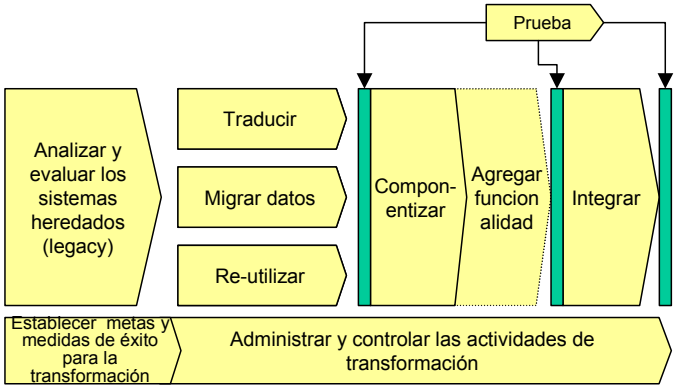
Esta sección presenta los enfoques actuales para la transformación y las herramientas disponibles para ayudar a que esta se dé. Nos brinda un marco modelo para ayudar a distinguir entre la amplia gama de productos disponibles en el Mercado.

El requisito básico para tener una transformación de aplicaciones legacy exitosa es mantener (y agregar) el valor del activo legacy. En la práctica esta transformación puede adoptar varias formas, por ejemplo puede involucrar traducción del código fuente, o cierto nivel de reutilización del código existente, incluyendo una capacidad de conectar el sistema legacy a la Web para brindarle al cliente el acceso requerido por la empresa. A lo largo del documento se asumirá que el objetivo es pasarse a una plataforma abierta/estándar (tal como J2EE o .NET) y que se puede agregar cierta funcionalidad adicional al proceso.

Descripción del proceso

El siguiente diagrama es un modelo del proceso de transformación de aplicaciones legacy incluyendo todas las actividades claves involucradas. La secuencia de actividades puede variar y existirá una iteración a lo largo del proceso a la hora de transformar una cartera de aplicaciones, por ejemplo. Es común completar la traducción de código existente, migración de datos y pruebas asociadas antes de agregar nueva funcionalidad. Esto se hace para probar el resultado final determinando su equivalencia con la aplicación existente y para probar la exactitud de la traducción y de la migración de datos antes de cualquier cambio en la lógica y estructura del programa.

El insumo del proceso es una aplicación legacy y el resultado es una aplicación transformada con la mayor parte del activo legacy intacto, posiblemente mejorado e integrado a la cartera general de aplicaciones de la empresa. El proceso está subdividido en varios pasos del proceso que son necesarios para lograr esta transformación. Se debe comprender que así como en la mayoría de los modelos de procesos, pueden haber varias rutas para pasar por todos estos pasos del proceso y pueden haber algunas iteraciones. Por ejemplo, se puede decidir reutilizar algún código legacy “envolviéndolo” en Java, mientras que otros códigos son traducidos directamente a Java- esto implica diferentes rutas a través de los sub procesos. Un ejemplo de iteración es cuando se da la transformación en etapas: Una parte de la aplicación es transformada hasta que esté integrada y entra en operación; luego comienza nuevamente el proceso para transformar la siguiente parte de la aplicación; se requiere más integración y este proceso se repite



Este modelo describe el proceso para transformar aplicaciones legacy- existen varias rutas posibles para que se dé el proceso, y pueden darse algunas iteraciones antes de terminar la transformación

hasta que todas las partes de la aplicación hayan sido transformadas.

Esta lógica también se aplica a paquetes de aplicaciones que comparten los mismos datos o tienen una función empresarial común. Aquí es mejor transformar el paquete como un grupo, pero si esto no es factible, entonces es necesaria la programación adicional para brindar el "andamiaje" (adaptadores) requerido para mantener el paquete de aplicaciones operando durante las fases de transición. En la siguiente tabla aparecen las actividades más comunes que componen estos elementos de proceso.

Sub-proceso	*Actividades Comunes*
Analizar y evaluar las aplicaciones legacy	▪ Hacer un inventario de todas los "artefactos " de aplicación por ejemplo código fuente, "copybooks", JCL, etc. ▪ Minería en la aplicación para extraer las reglas de la empresa ▪ Organización de los elementos de aplicación ▪ Evaluación de la condición del código fuente ▪ Análisis de bases de datos incluyendo, tablas, índices de vistas, procedimientos y "triggers", perfilado de datos
Traducción	▪ "Refactoring" (mejoramiento de estructura de código) ▪ Traducción automática de código ▪ Ajuste de manuales
Migración de datos	▪ Reestructuración del ISAM a un esquema de base de datos de tipo relacional ▪ Creación de un ambiente relacional ▪ Migración de todo lo no relacional a relacional de acuerdo con el modelo de datos
Reutilización	▪ "Wrap" (Envolver) el código COBOL en Java (y exponer únicamente partes limitadas a los procesos Web) ▪ Extraer las reglas de la empresa y objetos y pasarlos a Java, XML ▪ Regeneración del dialecto COBOL al estándar actual
Componentizar	▪ Extraer componentes, alinearlos con la función de la empresa ▪ Reestructurar en presentación, datos persistentes y lógica de la empresa
Agregar funcionalidad	▪ Esta es la etapa en que se puede agregar una mayor funcionalidad para cumplir con los requerimientos específicos de la empresa que aun no han sido cumplidos por la aplicación legacy
Integración	▪ Transferencia de ejecutables, archivos de imágenes, Java etc. a servidores ▪ Habilitar para la Web ▪ Instalación de enlaces para acceso de datos ▪ Sintonización del desempeño(conforme se requiera) ▪ Interfaces MQ y SOAP
Establecimiento de metas y medidas para el éxito de la transformación	▪ Acordar las metas con la administración ▪ Incorporar metas y medidas dentro de los planes del proyecto ▪ Llevar a cabo "revisiones" periódicos para asegurar que se está siguiendo lo básico ▪ Medición después de haber concluido
Administración y	▪ Control y rastreo de la versión

Sub-proceso	*Actividades Comunes*
control de la transformación	▪ Estimación del esfuerzo requerido ▪ 'Diligencia debida' en cuanto a la factibilidad de la conversión del código ▪ Establecimiento de "métrica" (ejemplo de complejidad) ▪ Aplicación de documentos (retro-documento aplicación existente y/o resultado final) ▪ Escoger metodología para regir la transformación
Pruebas	▪ Utilizar un depurador ▪ Create test scripts ▪ Confidence testing ▪ Pruebas de aceptación por parte del cliente

Automatización del proceso de transformación

Existe una gama de herramientas, productos y estándares disponibles para brindar asistencia en el proceso de transformación como aparece descrito anteriormente:

- Herramientas para ayudar a evaluar las complejidades de las aplicaciones existentes y para mapear/evaluar el código e interfases.
- Herramientas para traducir código automáticamente para los lenguajes principales.
- Herramientas para soportar la conversión de estructuras de datos legacy a bases de datos relacionales actuales.
- Productos para ayudar a diseñar el B2B, estructuras Web y cliente-servidor requeridas.
- Estándares tales como XML y JCA (un estándar para una conexión sincrónica de J2EE con aplicaciones y procesadores de transacción) que facilitan la integración.
- Análisis automático para ayudar a identificar y desarrollar componentes estándar.

Estas herramientas, productos y estándares:

- *Ahorran tiempo y costos* reduciendo el aporte manual y agilizando el proceso de transformación.
- *Reducen el riesgo* utilizando herramientas probadas para asegurar la previsibilidad y transparencia del proceso.
- *Preservan la integridad de la aplicación legacy* por medio de criterios de aceptación basados en casos existentes de prueba.
- *Unifican las aplicaciones existentes* migrando de varios lenguajes y/o plataformas a un lenguaje y tipo de plataforma.

- *Preservan el valor de las aplicaciones legacy* transponiendo el código existente a una nueva plataforma y asegurando que la conversión no agregue nuevos errores de programa.

- *Establecen la definición de requerimientos* brindando una línea de base explícita de requerimientos para las aplicaciones existentes sobre las cuales se puede establecer la definición de nuevos requerimientos funcionales.

- *Comprueban que la solución podrá responder a las necesidades futuras* dirigiéndonos hacia plataformas abiertas y mejorando la calidad del código y estructuras de código de manera que la aplicación pueda adaptarse a los cambiantes requerimientos de las empresas.

Superación de las dificultades inherentes a la transformación

Al considerar la factibilidad de las herramientas automatizadas para la transformación, es útil considerar esta lista de obstáculos:

- La aplicación legacy es monolítica. La capa de presentación, el almacenamiento persistente y la lógica de aplicación están entrelazadas.

- La aplicación legacy está ligada a las funciones específicas de un sistema operativo, perjudicando la portabilidad a otras plataformas.

- La tarea implica la transformación de varias aplicaciones que son interfases rígidas, y posiblemente escritas en diferentes lenguajes, en diferentes momentos, y por personas que no se comunicaron entre ellas.

- El código es el único lugar en que las reglas de la empresa están documentadas, las reglas están dispersas en el código y no hay nadie que comprenda la aplicación.

- Los límites de la aplicación están mal definidos- el código fuente es únicamente parte de la aplicación. Su operación puede depender de JCL por ejemplo, o en el caso de 4GLs, de un motor de tiempo de ejecución.

- Código duplicado; código no ejecutado; copias múltiples de interfaces: ¿Cuál es el código correcto? ¿Existen dependencias escondidas?

- La calidad del código es mala.

Específicamente, así son abordados los obstáculos enumerados anteriormente:

Obstáculo	*Cómo es abordado*
Aplicaciones Monolíticas	El modelaje de la arquitectura de las aplicaciones legacy separa la capa de cliente, la lógica de aplicación y los datos persistentes. Una vez que la migración del código es terminada, puede iniciarse la componentización, para alinear los componentes con función empresarial. La representación de aplicaciones y datos como componentes facilita el montaje rápido y fácil de nuevas funciones empresariales. Como regla

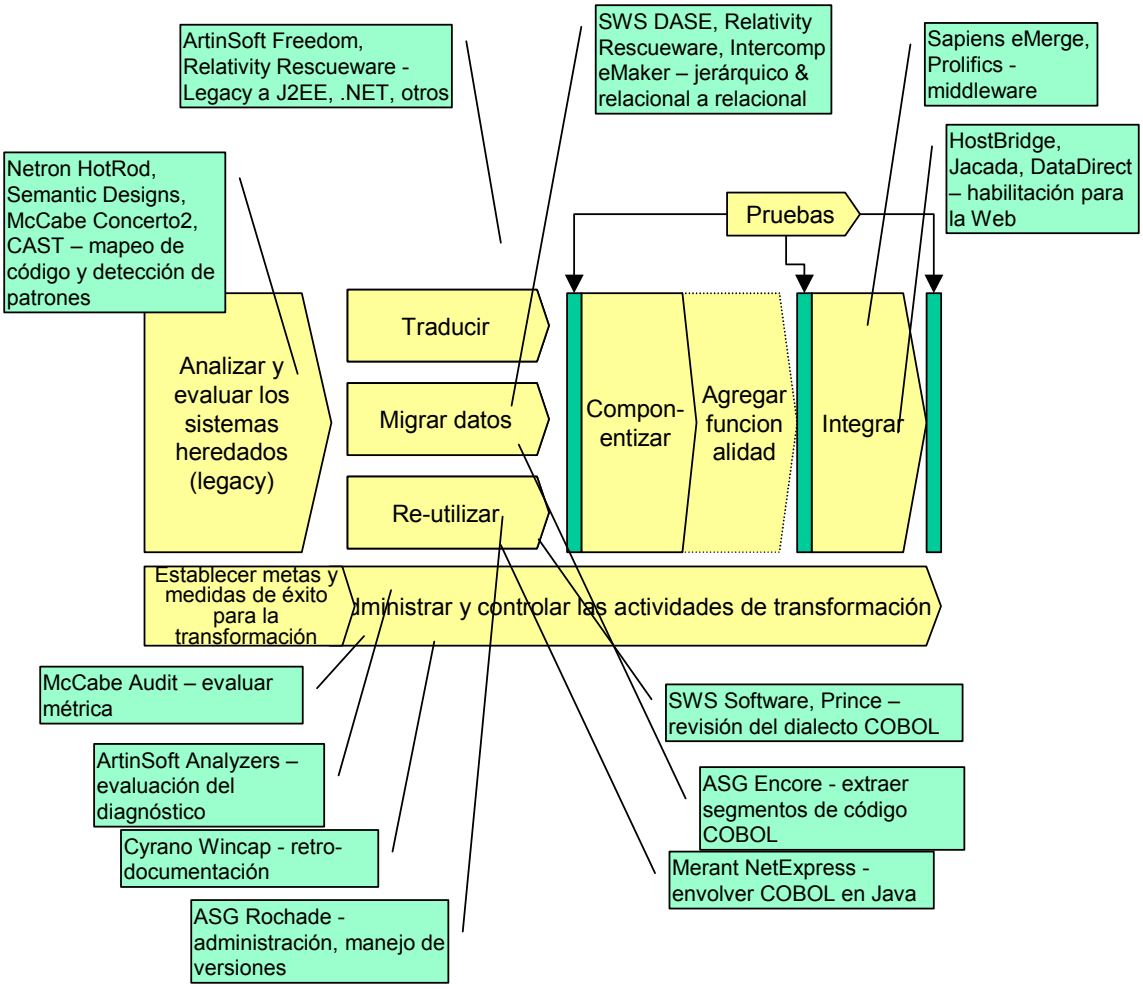
Obstáculo	*Cómo es abordado*
	general los últimos avances limitan la componentización a funciones empresariales, y esta puede estar limitada por la estructura del código original.
Aplicación ligada a funciones específicas de un Sistema Operativo	Los traductores incluyen traducción automática de llamados de programa a Sistema Operativo específico hacia su equivalente en la plataforma meta. Cuando hay una llamada no reconocida, esto se denomina una excepción que será traducida manualmente. Si esto ocurre muchas veces, el proveedor puede reincorporar una nueva solución dentro de la herramienta. Un método que utilizan algunos traductores es un análisis que se corre en código de muestra antes de comenzar la traducción para identificar el posible número de excepciones. Si estas están dentro de un pequeño número de categorías, el traductor puede ser ajustado para manejarlas automáticamente.
Aplicaciones rígidamente acopladas e Incompatibles	La gran ventaja de la plataforma de aplicación moderna y abierta es la habilidad de integrar diversas aplicaciones una vez que la conversión inicial del código se ha llevado a cabo en la plataforma meta. El enfoque de desarrollo evolutivo adoptado por la mayoría de las empresas quiere decir que la transformación se dará en conjunto con aplicaciones, datos e infraestructura existentes. Las aplicaciones transformadas tendrán que coexistir e operar conjuntamente con esas aplicaciones.
El código contiene las reglas de la empresa y nadie las conoce	La minería en la aplicación puede separar código de interfaz, control de flujo, IO, etc. de la pequeña porción del código que representa la lógica de la empresa, por tanto aislando la lógica de la empresa. La búsqueda de lógica de la empresa puede ser reducida mediante búsquedas automatizadas para constructos de código que generalmente albergan las reglas de la empresa. Esto es seguido por los talleres de inspección de código con mantenedores de código. Un enfoque complementario inicia con los datos cuyo valor reflejan la ejecución de una regla y rastrean el código que establece el valor (esto fue muy útil en Y2K, y situaciones como la del Euro pero puede ser generalizado.) Partiendo de estos, los procesos implícitos son mapeados y listos para ingresarlos a la definición de requerimientos o nueva funcionalidad. Paralelamente con este esfuerzo se lleva a cabo un proceso de creación de conocimiento (más adelante se menciona este tema).
¿Cuáles son los límites reales de la aplicación?	Una combinación de minería de aplicación e inspección manual facilita el mapeo de los límites. Se dan pasos específicos en la traducción 4GL, por ejemplo, para tratar con las bibliotecas “run-time” e infraestructura similar. Las metodologías de transformación especifican que para asegurar una equivalencia funcional, todo el código, interfases, control de trabajos y datos necesitan ser considerados. Esto preserva la integridad funcional y reduce las pruebas que se deben llevar a cabo.
Problemas de código – duplicación, código no	La clasificación de esto es un sub producto de la minería de la aplicación mencionada anteriormente- sin embargo,

Obstáculo	*Cómo es abordado*
ejecutado, identificación de la versión "correcta" de una interfaz, etc.	reconociendo que existen situaciones en que el requerimiento es simplemente mover el código de una plataforma más vieja, sin ningún intento de establecer la lógica de la empresa. Las herramientas automatizadas para una conversión de datos persistentes pueden producir definiciones de elementos de datos, determinar cuáles registros de fuentes son utilizados en tablas en la base de datos meta y designar la fuente para los datos en cada tabla.
Problemas de calidad del código	Las herramientas de traducción pueden ser aplicadas a una muestra de código para evaluarlo. Si el problema es más básico, por ejemplo la aplicación no funciona correctamente, entonces puede ser necesario dejarla como candidata para la transformación y tratar de extraer las reglas de la empresa para re-escribirla. Existen herramientas de inspección automatizadas (tales como Advantage) para evaluar la estructura).

Ejemplos de Herramientas

El diagrama a continuación muestra el modelo de proceso con ejemplos de productos actuales mapeados sobre este. Esta es únicamente una selección representativa y existen otros productos disponibles y otros proveedores que operan en este espacio. La lista demuestra que las herramientas disponibles cubren todo el proceso de una manera u otra. Es importante notar que la mayoría de estos productos son 'soluciones específicas'. En otras palabras, estos manejan típicamente uno o dos lenguajes de programación específicos, y están restringidos a una pequeña cantidad de ambientes meta. Por ejemplo, los traductores están diseñados para manejar lenguajes específicos, aunque pueden ser extendidos de una manera bastante fácil para que manejen otros lenguajes conforme lo requiera el mercado.

Además, los productos generalmente están limitados a una parte del proceso de transformación. Esta es una consecuencia inevitable de la estructura del proceso de transformación, porque aunque el proceso de transformación es presentado como una transformación monolítica, en realidad los elementos son bastante diferentes unos de otros, y los cambios no son todos consistentes. Por ejemplo, las actividades de traducción cambian el código de un lenguaje a otro, mientras que las actividades de Analizar y Evaluar de la Aplicación Legacy hacen el cambio de un estado de "ignorancia" (relativa) a uno de "conocimiento". Por tanto, es razonable suponer que las herramientas que apoyan la automatización de estos elementos de proceso separados seguirán siendo distintas (aun cuando se acceden por medio de una interfaz común).



Esto no necesariamente es algo malo, pero sí quiere decir que es probable que se requiera de más de una herramienta y será necesario hacer cierta mezcla– con suficientes conocimientos técnicos y una buena administración de proyectos, el proyecto de transformación tendrá éxito. Sin embargo, este sugiere que cierto escepticismo es necesario cuando un proveedor habla acerca de una solución que “se encarga de todo”.

Hay límites en la descomposición automática de las aplicaciones en componentes que pueden convertirse en los componentes básicos para nuevas aplicaciones. En general, las técnicas de transformación actuales llevan las aplicaciones a una nueva plataforma, no a una nueva arquitectura. No es fácil reestructurar manualmente la aplicación transformada para crear componentes por debajo del nivel empresarial funcional. Una meta factible y realista puede ser una arquitectura meta que consista en objetos del negocio que encapsulan la lógica de la empresa de una sola entidad y los datos específicos de esta. Por ultimo, la factibilidad de reutilización depende de la estructura de la aplicación legacy original– el código estructurado y bien formado se presta de mejor manera para cierta componentización, mientras que el código monolítico necesita del “wrapping” (envoltura).

Implicaciones

- La automatización sustancial del proceso es factible, haciendo de la transformación una proposición económicamente atractiva comparada con la re-escritura o reemplazo de la aplicación legacy. Primero que nada, esto asume que la aplicación legacy es “apta para el propósito”.

- Las herramientas disponibles cubren todos los aspectos del proceso, aunque se requiere de alguna intervención manual. Utilizando las herramientas en la práctica dependerá de la escala de la tarea y de si la automatización es necesaria o económica en todos los casos.

- Cada una de las herramientas disponibles en el mercado maneja únicamente parte del proceso de transformación. Consecuentemente habrá necesidad, en la mayoría de los casos, de comprar servicios para ayudar a diseñar y ejecutar la transformación, ya que existen muchas incógnitas que deben superarse sin ayuda de expertos con experiencia previa. La provisión de herramientas puede variar- en algunos casos el código puede ser enviado a otra parte a ser traducido, en otros casos las herramientas reciben licencias con base en su uso. Algunas herramientas requieren ser operadas por personal especializado.

- Los recursos internos deben estar involucrados a todo lo largo, al principio para impartir conocimientos disponibles acerca de las aplicaciones legacy, luego en la etapa de pruebas y de aceptación, y para establecer la base de conocimiento futura.

En el siguiente capítulo se discute una estrategia para llevar a cabo una transformación de aplicaciones legacy.

4. Desarrollo de una Estrategia para la transformación de aplicaciones legacy

El desarrollo en herramientas y técnicas para la transformación de aplicaciones legacy brinda una oportunidad para que la empresa revise su cartera legacy. ¿Transformar o reemplazar, desechar o reinvertir? Estas son las dudas que deben ser abordadas por una estrategia legacy.

Cuando el requerimiento es transformar una sola aplicación legacy, entonces la opción de hacia dónde dirigirse será determinada por la calidad de la aplicación, y tomando en cuenta si existen o no productos empacados disponibles para reemplazarla si la calidad es mala. En la mayoría de los casos el Gerente de Sistemas necesita comenzar con problemas más básicos de la empresa.

Los objetivos

Existen cuatro impulsores comunes para considerar un proyecto de transformación:

- Reducción de costos operativos, del mantenimiento y de los gastos generales de aplicaciones más viejas.

- Mejorar el mantenimiento de la aplicación en situaciones en que nadie conoce la aplicación, o en que existe una alta rotación de personal, falta de recursos calificados y adecuados, o documentación limitada/desactualizada.

- Mejorar el acceso a la aplicación(es) legacy, muchas veces para brindar acceso a la Web para clientes y socios comerciales, o como resultado de una fusión o reestructuración organizacional, en la cual existirán nuevos usuarios y productos.

- Posicionamiento para nuevos proyectos (tales como Servicios Web, o cambio esperado de negocio).

En la práctica algunos de estos impulsores pueden combinarse. Estos impulsores pueden verse como un tipo de espectro, que va desde los factores “push” (lo que quiere decir que la aplicación legacy debe reaccionar para ser adecuada dentro de un nuevo ambiente) o los factores “pull” (la empresa quiere aprovechar una oportunidad para crecer/alcanzar nuevos clientes/agregar productos y servicios).

Transformación de Legacy – Modelo de Valor de la Empresa

Impulsor de la Empresa			
Enfoque interno ⟷ Enfoque externo			
Sistema en Deterioro	Economía	Comercio Electrónico	Preparación para el cambio
Objetivo Principal de la Empresa			
Sobrevivencia, continuidad operativa	Reducir costos operativos	Extender el alcance dentro de la empresa y/o fuera de ella	Posicionar la empresa para el futuro
Dimensiones de valor			
• Mejoras en mantenimiento (documentación, más fácil de arreglar) • Acceso al soporte • Costos operativos más bajos • Acceso a nuevos clientes (proveedor de paquetes) • Sistema más adaptable	• Reducción de costos operativos, costos de licencias, costos de recuperación respaldo/desastres • Oportunidad para contratar externamente • Reducción de la complejidad • Sistema más adaptable	• Mayores ingresos • Nuevos clientes/usuarios • Mejor servicio para clientes/usuarios existentes y nuevos • Reducción en costos de adquisición de clientes • Mejoramiento de marcas	• Sistema adaptable • Mayor integración con socios comerciales • Más servicios para los clientes • Opción de Servicios Web • Re-utilización de componentes • Comprobar respuesta ante las necesidades futuras

En este diagrama se comparan estos impulsores, sugiriendo que puede haber bastante variación en el valor que la empresa estará buscando obtener con una transformación. Los siguientes puntos deben ser tomados en consideración:

- Los impulsores en el lado izquierdo del modelo hacen énfasis en el costo. La justificación para el proyecto de transformación está basada en supervivencia o ahorro de costos. Esto elimina la posibilidad de tener un crecimiento descontrolado de alcances en el proyecto de transformación (por ejemplo). Probablemente los aspectos básicos son la plataforma, lenguajes y estructuras de datos- en otras palabras, la arquitectura.

- Los impulsores del lado derecho tratan acerca de oportunidad comercial, y los problemas son de funcionalidad y capacidad futura.

- Existen diferentes niveles de participación por parte de la empresa. Las plataformas obsolescentes y los impulsores económicos probablemente serán controlados por el departamento de TI, y serán proyectos internos de la empresa. Los impulsores de comercio electrónico y reposicionamiento necesitan la participación de otras áreas de la empresa, tales como mercadeo, ventas, desarrollo de producto, etc., así como a socios comerciales y clientes en algunos casos, y por tanto tener un enfoque externo.

- ….esto también podría determinar quién es el cliente realmente- TI o la empresa.

- En casos en que un paquete de un tercero esté contribuyendo al deterioro de la situación de la aplicación, habrá participación por parte de la empresa en la decisión de la inversión- la necesidad de decidir sobre plataformas meta por ejemplo.

- En casos en que el ahorro de costos es el principal impulsor, la contratación externa puede ser sumada como otra consideración combinada con la transformación de la aplicación legacy.

Evaluación de la cartera

En cualquier empresa existe una distinción obvia entre aplicaciones financieras, procesamiento de ordenes de ventas, recursos humanos y recursos de manufactura, por ejemplo. Pero por supuesto dentro de estas categorías muchas veces hay varios cientos de aplicaciones individuales con millones de líneas de código entre ellas. Es deseable hacer una pre-evaluación de la cartera antes de proceder a tomar decisiones individuales acerca de los métodos de transformación.

El énfasis de la pre-evaluación está en el *valor comercial* (una de las metas de la transformación es abordar la calidad técnica, de manera que esto se pospone hasta más adelante en el proceso). El objetivo es modernizar la cartera de aplicaciones revisando las aplicaciones existentes para determinar si estas continúan brindando valor a la empresa. Si se logran identificar las aplicaciones que se pueden descontinuar esto daría como resultado ahorros inmediatos. Esta modernización requiere de compromiso por parte de la empresa. A continuación aparece una lista de preguntas que se deben hacer acerca de cada aplicación:

- ¿Tiene esto impacto directo sobre los clientes o socios comerciales? Si la aplicación dejara de correr mañana, ¿tendría esto impacto sobre la empresa?

- ¿Cuándo fue la última vez que se le preguntó a los usuarios de la aplicación acerca de su valor? ¿Cuál sería su respuesta si se les preguntara hoy?

- ¿Existe más de una aplicación capaz de brindar la información?

¿Transformar, reemplazar, re-escribir, o reutilizar?

Existen alternativas aparte de la transformación total de la aplicación legacy, y en este punto del desarrollo de la estrategia de transformación estas alternativas deben ser revisadas. Los principales factores de decisión son (I) la calidad de la aplicación legacy, y (ii) la disponibilidad de los paquetes de reemplazo. 'Calidad' en este caso es un término subjetivo aplicado a la aplicación en sí, sin importar la plataforma en que corre o el código utilizado, y debe ser evaluada en términos de parámetros tales como:

- Efectividad actual (por ejemplo errores generados, cantidad de soluciones alternativas, nivel de soporte necesario).

- Estabilidad de las reglas principales de la empresa- ¿permanecerá la lógica de la aplicación igual a mediano plazo? Existe un supuesto importante en la transformación de aplicaciones legacy, y es que el activo del software actual es un activo valioso. Si un modelo de negocios va a cambiar sustancialmente, entonces este supuesto deberá ponerse en duda. Habiendo dicho esto, en la práctica el código muchas veces es únicamente un lugar para colocar las reglas de negocio y estas aparecen dispersas en el código. Por tanto cualquier intento para "comenzar desde cero" necesita reconstruir y documentar los requerimientos capturados en el código legacy y tomar estos requerimientos como punto de partida para la negociación de nuevos requerimientos (utilizando la minería de aplicaciones).

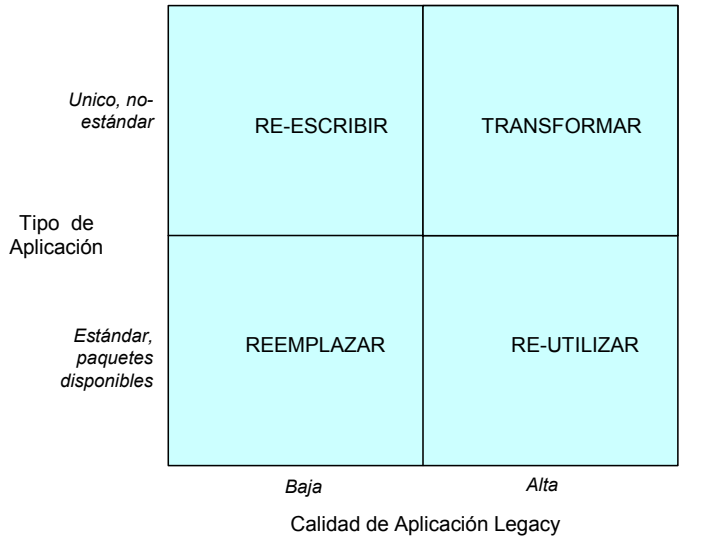
- Brechas en la funcionalidad.

- Etapa del ciclo de vida de las aplicaciones legacy – en las primeras etapas del ciclo de vida, una aplicación legacy probablemente se encuentra cercana a los requerimientos de funcionalidad aunque la plataforma esté obsoleta.

En resumen, la evaluación de "calidad" trata acerca de qué tan apta es la aplicación legacy en términos comerciales y técnicos.

La disponibilidad de un paquete de reemplazo es la otra consideración importante y esto dependerá de la singularidad de la aplicación actual. Si la calidad de la aplicación legacy es mala y existe una funcionalidad comparable disponible en paquetes de software de terceros, tendría sentido reemplazarlo.

Existen cuatro opciones: transformar, reutilizar, re-escribir o reemplazar. Como se explicó anteriormente, algunos o todos los elementos del proceso de transformación aplican a las primeras tres. El siguiente diagrama[1] muestra las combinaciones de factores que pueden llevarnos a estas cuatro opciones.



Adaptado de Erlich

Transformar – Aplicar el proceso de transformación, agregando funcionalidad y alcance empresarial conforme se requiera.

Re-utilización – Existen dos posibilidades en este caso, una en que la aplicación legacy se centra en un paquete de terceros/DBMS, y la otra si la empresa ha desarrollado su propia aplicación partiendo de cero. Si la cartera de aplicación legacy se centra mayormente alrededor de un paquete de terceros, entonces la mejor opción es actualizar hasta la versión más reciente y utilizar técnicas de "wrapping" (envoltura) para brindar el alcance requerido y otras mejoras en funcionalidad. Para aplicaciones internas considere aplicarle el "wrapping" a la aplicación. Brindar acceso directo a datos por parte de usuarios finales sin pasar por la aplicación legacy, por lo general, quiere decir agregar también un "data warehouse". La desventaja de este enfoque es que agrega más

[1] Diagrama adaptado con base en una presentación de Len Erlikh de Relativity Technologies, Inc

elementos a ser mantenidos y dos conjuntos de datos que deben mantenerse sincronizados.

Re-escritura – El activo clave aquí es las reglas de negocio y las estructuras de datos- el problema es la aplicación. Para iniciar la re-escritura, se requiere de la minería de la aplicación, el análisis de lógica del código y estructuras de datos.

Reemplazar – Busque un paquete adecuado o contrate externamente. Esté preparado para hacer cambios al modelo de la empresa para ajustarse a lo que le puede ofrecer el paquete.

Lista de control para escoger la transformación

- La aplicación existente es apta para las necesidades de la empresa
- Se necesitan cambios moderados de funcionalidad en la aplicación existente
- Funcionalidad considerable a ser agregada en una nueva aplicación y se necesita una integración cercana con la existente
- Altos costos operativos de la aplicación existente
- Necesidad de migrar a J2EE o .NET por razones estratégicas
- La visión futura incluye Servicios Web
- Agregar acceso a la Web
- Es difícil encontrar recursos para mantener / modificar las aplicaciones en la plataforma existente

Lista de control para escoger la reutilización

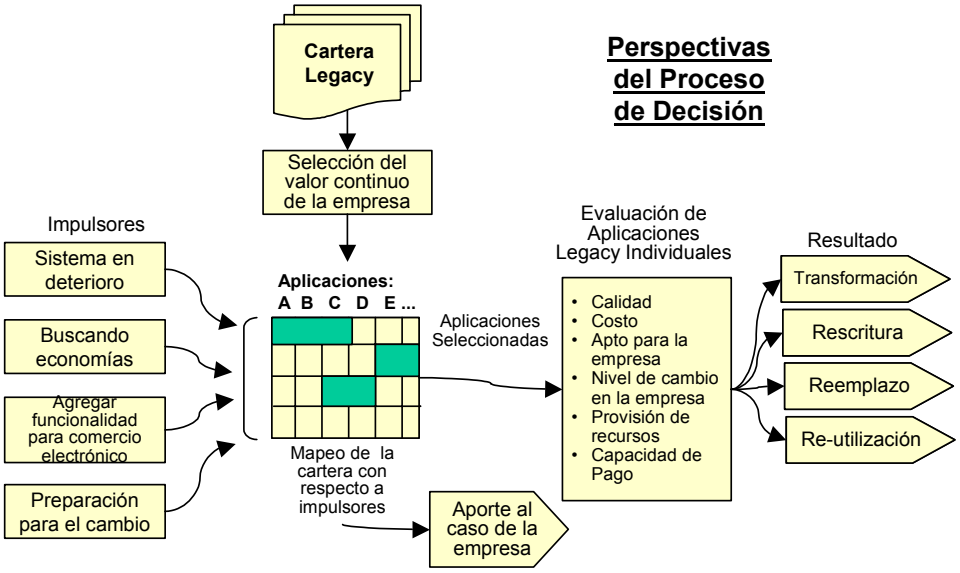
- Reglas de negocios son satisfactorias
- Costos operativos de la aplicación existente son bajos
- Difícil separar las capas de lógica de los datos persistentes y de la presentación
- Se requiere acceso simple a la Web, permitiendo una solución de envoltura (Wrapping)
- Tener los recursos para mantener el legacy principal
- Software de paquete es vital para la aplicación existente, apoyarse en un tercero para obtener soporte y mantenimiento

Lista de control para la re-escritura

- Reglas de negocio son satisfactorias pero necesitan que se les agregue una mayor funcionalidad
- Ninguna solución en existencia se acerca a satisfacer las necesidades
- El código existente es de mala calidad, y tiene altos costos de mantenimiento
- Se cuenta con el tiempo, y se puede incurrir en el costo y la interrupción que esto implica

- Lista de control para reemplazar
- Aplicación considerablemente alejada de las necesidades de la empresa
- Se quieren hacer cambios en el modelo de negocio para adecuarlo a una solución en existencia
- Se cuenta con el tiempo, y se puede incurrir en el costo y la interrupción que esto implica

El diagrama a continuación resume el proceso general de decisión de forma gráfica.



¿Cuál plataforma meta?

Los proyectos de transformación están dirigidos principalmente a la conversión de código o módulos a Java, C++ (or C#), a un ambiente de base de datos relacional, o a una arquitectura HTML – o alguna combinación de estos. La mayor parte del *Nuevo* desarrollo a nivel de empresa en el futuro previsible tendrá lugar en una de estas dos plataformas: plataforma .NET de Microsoft o la Plataforma Java 2, Edición Empresa para multi vendedores (J2EE). Esto no es lo mismo decir que todas las transformaciones legacy deben dirigirse a una de estas dos plataformas, pero existen argumentos contundentes a favor de esto justamente:

- Existen ventajas técnicas para estas plataformas. Por ejemplo, brindan la posibilidad de reutilización, escalabilidad y un amplio acceso a productos y servicios relacionados. La reutilización se deriva de los aspectos que componen estos marcos. La escalabilidad es inherente en las arquitecturas, y todo proveedor importante se apoya en una u otra: .NET y J2EE. Los ambientes de Desarrollo Integrado (IDEs) hacen que la tarea del desarrollo y mantenimiento sea más fácil. Los contenedores de aplicación (ambientes de "tiempo de ejecución") brindan las calidades de servicio necesarias para las aplicaciones de empresa tales como manejo de transacciones, seguridad y servicios de persistencia. Estos factores (más la competencia de mercado por la oferta de plataformas) a final de cuentas reducen los costos operativos.

- Los productos más automáticos de traducción están dirigidos a estas plataformas, aunque con un énfasis actualmente en Java (y por lo tanto tienden a favorecer J2EE). La traducción del dialecto COBOL es una excepción.

- Existe una creciente base de conocimientos, que hace más fácil el reclutamiento de personal, y debido a que estas plataformas se consideran como tecnologías de punta son atractivas para el personal existente que quiere ampliar sus propios conocimientos y avanzar en sus carreras.

- Facilitan la publicación de función de aplicación a una red utilizando protocolos estándar basados en XML para ser utilizados por otras aplicaciones (usualmente conocidos como "Servicios Web").

Los factores que hacen estas plataformas adecuadas para los Servicios Web son por supuesto de gran interés en la transformación de aplicaciones legacy, por la manera en que la integración puede ser facilitada. La mayoría de las aplicaciones legacy forman parte de una cartera existente de aplicaciones interrelacionadas e interdependientes y los Servicios Web son el siguiente paso en la evolución de integración de aplicación. Las desventajas incluyen la evolución continua de estándares para Servicios Web y la solución para aspectos relacionados con la seguridad.

La opción de.NET versus J2EE es un tema altamente debatido. Ambos han evolucionado de una tecnología existente de servidor de aplicación. J2EE es bastante madura y se corre en aplicaciones empresariales de gran escala. .NET es el recién llegado, pero definitivamente vinó para quedarse. Hoy en día, las soluciones basadas en Microsoft están limitadas casi en su totalidad a plataformas clase Wintel – en otras palabras, la opción de .NET implícitamente escoge la plataforma, middleware y el sistema operativo y se puede decir que es más difícil crecer a varios cientos de usuarios concurrentes que con una implementación similar J2EE . Por otra parte una organización más pequeña puede escoger una plataforma como su estándar casi universal y utilizar .NET por su bajo costo de entrada y enfoque en el desarrollo rápido de aplicación. Ambos J2EE y .NET están siendo reposicionados para distribuir la visión de Servicios Web.

Los argumentos estratégicos para cada una son familiares. La portabilidad de plataforma versus estar amarrado a un proveedor, ventajas en cuanto a costos de un único proveedor versus el trato con varios proveedores, una mejor arquitectura versus el riesgo de inestabilidad mientras la arquitectura es implementada.

La conclusión es que cualquiera de las plataformas es una opción adecuada para la transformación de aplicaciones legacy y la escogencia entre ellas se hace de mejor forma basándose en las estrategias empresariales y técnicas en general.

Pueden haber consideraciones que nos llevan a otras opciones meta. Algunas posibilidades son:

- Migrar el nivel persistente de datos: Pasando de bases de datos de "mainframes" jerárquicas, de redes o relacionales a un ambiente RDBMS. Será necesario hacer algunos cambios de código correspondientes en la aplicación.

- Pasar a acceso via browser , mientras se le aplica la reingeniería a la aplicación para dar soporte a la habilitación para la Web: Agregar funcionalidad de autoservicio para cubrir actividades que normalmente son llevadas a cabo por agentes internos de la empresa; eliminar capacidades que deben estar disponibles únicamente internamente- por ejemplo cambio de descuentos, agregar datos, capacidades necesarias para la Web- por ejemplo ingresar direcciones de correo electrónico, migrar a bases de datos compartidas; crear un "data warehouse" para soportar el acceso Web (por ejemplo, la base de datos legacy puede ser no relacional o que no cumple con ODBC); separar la lógica de presentación.

- Crear soluciones complementarias alrededor de las aplicaciones legacy CICS: CICS TS V2 brinda un ambiente de ejecución EJB, que permite el uso de soporte CICS para la sesión EJB , habilitando el uso de soporte CICS para “beans” de sesión EJB para brindar acceso de clientes a las transacciones CICS, programas y recursos via IIOP. (El servidor CICS IIOP brinda el ambiente “run-time” en el cual el contenedor y a su vez los “enterprise beans” ejecutan y es el ambiente desde el cual pueden interactuar con otros servicios y recursos CICS) Las posibilidades incluyen presentación HTML o XML del lado del cliente, con ejecución de “servlets/beans” en ambiente WebSphere, e invocaciones del método Java luego pasan a IIOP para ejecutar “enterprise beans” que corren bajo CICS. Los “Java beans” bajo CICS de hecho ‘envuelven’ las aplicaciones existentes.

- Migración a cliente-servidor: Esta probablemente estará confinada a situaciones en que se requiere una interfase de usuario de alta función. El cliente servidor puede dar como resultado costos de soporte mayores para ambientes de computo distribuidos unido a ambientes de aplicación cliente/servidor más complejos. Esto no siempre tiene sentido hoy en día con J2EE y .NET disponibles.

- Migración a COBOL: Existen traductores para ciertos 4GLs a COBOL, lo que puede cumplir con el requisito mínimo para salirse de una plataforma obsoleta particular. Algunos ejemplos específicos son conversión de 4GL a COBOL (por ejemplo, CA-Easytrieve Plus, DYL-260/280, DataAnalyzer, etc.) — también conversión de BAL a COBOL, de CSP a COBOL, y OS/VS y VS COBOL II a COBOL para OS/390. Otros traductores identificados incluyen PL/I a Cobol, Natural a Cobol, también Cobol II, 74, o 85 a OS/390 Cobol, OS/390 Cobol a Cobol Cliente/Servidor.

5. Proyecto de Transformación

Un proyecto de transformación presenta muchas de las características de proyectos tradicionales de desarrollo tales como establecimiento de objetivos, participación por parte del usuario, pruebas, programación, monitoreo, etc. Sin embargo en un proyecto de transformación existen algunos factores diferentes:

- *La solución es desarrollada sobre las bases de un activo legacy,* en lugar de comenzar por descubrir los requisitos de la empresa. Por supuesto que pueden haber requerimientos funcionales adicionales a ser agregados, pero el procedimiento usual es agregar esta funcionalidad *después* de haber terminado la transformación.

- *Las tecnologías habilitantes necesitan ser gestionadas para la traducción, migración de datos, y reutilización, o se debe identificar un socio adecuado para brindar las tecnologías*

- Ya que la aplicación legacy ya es parte de las operaciones comerciales de hoy en día, *es necesario tener una transición sin dificultades*.

- *Se debe ir desarrollando el Know-how a lo largo del proyecto* de manera que las capacidades de soporte se encuentren en su lugar al terminar. Hasta cierto punto este es el caso para cualquier proyecto de desarrollo pero debido a documentación desactualizada y una comprensión limitada de la aplicación legacy, junto con el traslado a una nueva arquitectura es necesario prestar especial atención para desarrollar know-how y hacerlo accesible.

- *Será necesario hacer ajustes a las metodologías existentes de desarrollo* para asegurar que el trabajo está estructurado para satisfacer las necesidades de un proyecto de transformación y alcanzar los objetivos empresariales y técnicos, cronograma y presupuesto.

Desglose del método para el proyecto

Un proyecto de desarrollo clásico sigue un patrón de cuatro etapas de Planeamiento, Análisis, Diseño e Implementación. El mayor riesgo es que el proyecto de transformación será visto como implementación únicamente, con poca necesidad de planeación, análisis o diseño. El siguiente diagrama ilustra una manera de comprender la totalidad del trabajo involucrado.

Esta es una presentación lógica de las tareas y actividades involucradas, no una manera definitiva de terminar el proyecto. Cualquier organización que esté llevando a cabo un proyecto de transformación debe revisar sus planes con respecto a este diagrama para asegurarse de no haber olvidado nada. El diagrama muestra que el trabajo inicial es necesario antes de avanzar en el proyecto de transformación. Por ejemplo, las tecnologías de transformación disponibles pueden tener que ser provistas por múltiples proveedores, y es probable que sea necesario obtener asesoría por parte de consultores, al menos para un primer proyecto de transformación. Segundo, si existe incertidumbre acerca de la calidad de la aplicación legacy podría valer la pena correr una prueba del concepto en un mini-proyecto para confirmar su factibilidad y (talvez igualmente importante) para establecer en firme los costos totales del proyecto. Subsecuentemente la

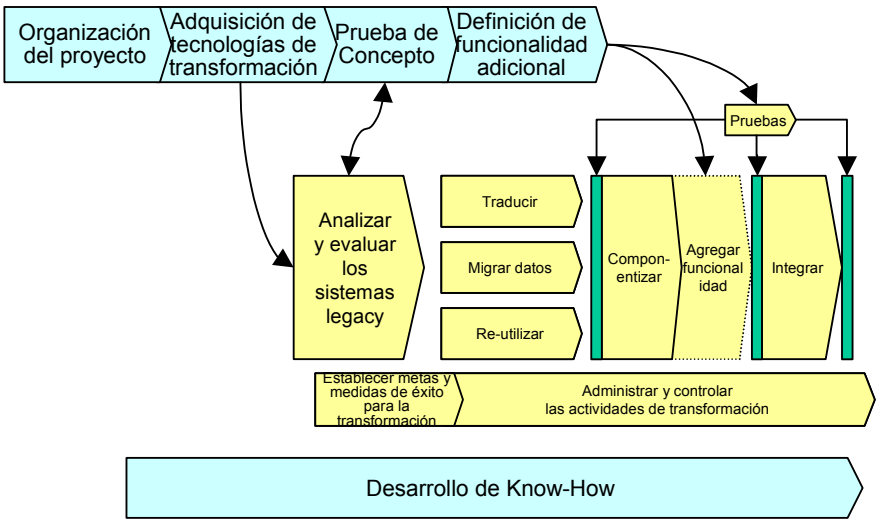
necesidad de tener funcionalidad adicional puede ser adquirida siguiendo procedimientos de desarrollo estándar, con la condición de que esta funcionalidad esté siendo agregada a una aplicación existente, de manera que los usuarios empresariales no estén partiendo de una hoja en blanco.

Aquí se hace énfasis en el desarrollo del know-how debido a las peculiaridades del activo legacy. Aunque este no es siempre el caso, a las aplicaciones legacy por lo general les falta documentación, los usuarios empresariales no están familiarizados con la índole exacta de las reglas de la empresa que están incluidas dentro de sus sistemas, y los conocimientos técnicos se ven limitados a una o dos personas en el departamento de Sistemas de Información. Las actividades del *Desarrollo del Know-How* podrían incluir las siguientes:

- Confirmar las reglas de la empresa, y hacerlas accesibles por medio de documentación o acceso HTML. La minería de aplicaciones es una tecnología útil para buscar en partes del código legacy que probablemente contienen las reglas.
- Identificar cualquier brecha en las reglas- esto es especialmente importante en casos en que el acceso a la aplicación debe ser extendido (ejemplo via habilitación para la Web) – y debe trabajar para llenar estas brechas.
- Ensamblar un modelo de usuario que pueda ser utilizado como la base para la aceptación de la aplicación transformada. El "modelo de usuario" contiene descripciones de la funcionalidad a ser presentada por la aplicación, las limitaciones que debe enfrentar (ejemplo versión de browser, hardware), y las propiedades que debe poseer la aplicación, tal como portabilidad, capacidad de mantenimiento, seguridad, etc. Revisar los escritos de prueba disponibles y extenderlos para asegurar que el "modelo de usuario" pueda ser verificado.
- Actualizar al personal de desarrollo y soporte en cuanto a tecnologías meta.
- Asegurar la transferencia del know-how relevante de los consultores, sobre todo si es el primer proyecto de transformación para la empresa.

Características específicas de los proyectos de transformación

El 'método' descrito en el párrafo anterior describe un marco para la transformación. Las características específicas enumeradas anteriormente son en parte abordadas por este



marco y por actividades específicas adicionales:

Desarrollo sobre las bases del activo legacy, en lugar de comenzar por descubrir los requerimientos de la empresa– El proceso de transformación descrito incluye una fase de evaluación y análisis. Para situaciones en que las reglas de la empresa en el legacy sean inciertas, estas pueden ser aclaradas en esta fase- o en casos en que sea necesaria la reescritura estas pueden formar la base de requerimientos para el desarrollo. Las herramientas de minería de aplicaciones pueden ser útiles en este proceso. La inclusión de actividades para desarrollar Know-how son complementarias a este trabajo y aseguran que el activo legacy sea bien comprendido.

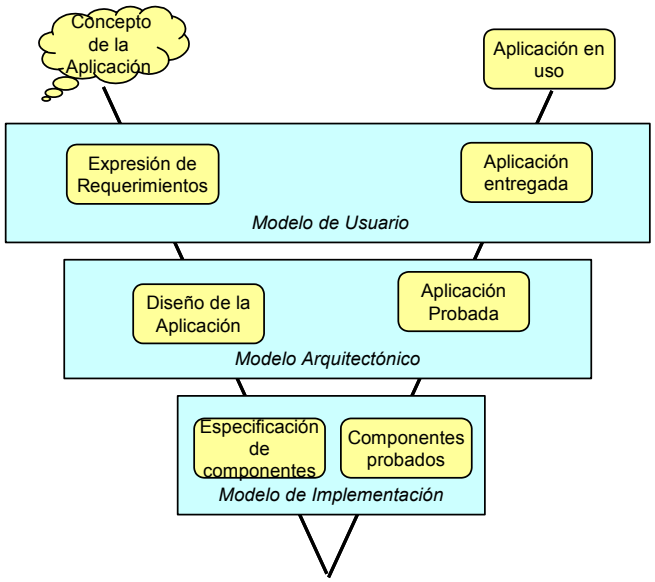
Obtención de tecnologías habilitantes para traducción, migración de datos y reutilización o un socio adecuado identificado para aportar las tecnologías– El marco anterior cubre esto.

Lograr una transición sin problemas – No existe una receta sencilla para hacer la transición de manera que no se sienta. Un método que ha sido utilizado con éxito es migrar grupos de función de la empresa a un ambiente vivo, utilizando middleware para integrarse con la función que debe ser migrada. Esto sucede en cuatro o cinco iteraciones hasta que la migración de toda la función de la empresa haya sido completada. Los datos necesitan ser sincronizados. Con los datos, un enfoque es tener una aplicación heredada lógica (logical legacy) y bases de datos nuevas que se mantengan sincronizadas utilizando un producto como DC Metalink. Como alternativa se puede usar middleware para brindar acceso directo desde legacy. El enfoque de acceso directo puede requerir de cierta modificación en el código de aplicación legacy y es más adecuado si no es necesario migrar el almacenamiento de datos legacy.

Desarrollo de know-how a lo largo del proyecto– El concepto aquí es que el plan del proyecto necesita desarrollar gradualmente conocimientos acerca de las aplicaciones existentes y las aplicaciones meta y crear el conocimiento necesario para dar soporte a largo plazo- esto puede mantenerse documentado y en las cabezas de la gente. La diferencia entre este y el enfoque usual son las tareas explícitas identificadas para que esto suceda. Es importante notar que existen diferentes tipos de know how– cliente (usuario), programadores (para dar un soporte continuo), y personal de soporte (ejemplo servicio de asistencia a distancia).

Es recomendable involucrar un socio consultor, al menos para el primer proyecto- alguien que pueda prever los muchos problemas que no se pueden anticipar y que sepa como manejar el ambiente meta. A continuación presentamos una lista de control: ¿Tienen una metodología? ¿Tienen acceso a las herramientas necesarias? ¿Cuál es su historial ? Conocen bien el .NET, J2EE o cualquiera que sea la plataforma meta que usted planea adoptar?

Haciendo ajustes a las metodologías de desarrollo existentes – Los proyectos de transformación requieren de estructura y (entregables) como cualquier otro proyecto. Cualquier metodología basada en el modelo V trabajará efectivamente, aunque podrían ser necesarias ciertas modificaciones en los procedimientos para seguir los principios de usuario, pruebas modelo de la arquitectura, y de implementación.



El llamado modelo V describe el desarrollo de aplicación como una progresión de etapas desde la presentación de requerimientos, diseño y creación hasta aceptación. Por medio de pruebas se demuestra que los entregables de las últimas fases son implementaciones de las especificaciones correspondientes del otro lado de la V.

Las metodologías en cascada son adecuadas para el patrón V igualmente que la mayoría de metodologías "livianas".

Las metodologías livianas son reemplazadas por los enfoques tradicionales de cascada en muchos talleres de desarrollo. Estos métodos livianos (ágiles) son adaptativos y aceptan el cambio bien, y están orientados hacia la gente. Estos son adecuados para la transformación de aplicaciones legacy por la manera en que los proyectos de transformación están enfocados (es necesario superar la fase de conversión de al menos parte de la cartera existente antes de que se pueda predecir con certeza el costo de la introducción de la nueva funcionalidad), el poco esfuerzo requerido en el diseño (los requerimientos son altamente inherentes en legacy), y proporcionalmente el mayor esfuerzo necesario para las pruebas y la puesta en marcha. Si un departamento de desarrollo ya tiene una metodología liviana documentada pueden ser necesarios algunos ajustes para que esta sea adecuada para un proyecto de transformación. Por ejemplo, el alcance y definición de las fases tempranas y entregables tendrán que ser ajustados y las actividades así como las pruebas deberán ser cambiadas.

La necesidad de tener un defensor dentro de la empresa

Existe una percepción común en las empresas de que los sistemas legacy son las soluciones del pasado. La aplicación legacy no es vista como una base para seguir adelante, sino como una aplicación desactualizada, difícil de mantener, y posiblemente una que no satisface plenamente las necesidades de las empresas. Un defensor es esencial para promover la opción de transformación y presentar el caso.

Esto no será fácil ya que existirán intereses opuestos a la transformación y a favor de otras opciones. El personal técnico interno tendrá poco incentivo para mantener la aplicación legacy pero tenderá a querer re-escribir o reemplazar mediante paquetes porque estas opciones brindan nuevas experiencias de desarrollo y la oportunidad para aprender nuevas destrezas. Los proveedores existentes de plataformas querrán mantener el estatus quo. Los gerentes serán los receptores de las campañas de mercadeo de los ERP y los proveedores de paquetes, y todas estas prometen beneficios por medio del reemplazo y rediseño de procesos. Los usuarios finales pueden ser más

positivos porque la aplicación legacy hace el trabajo que se espera de ella hoy en día, aunque el costo puede ser mayor de lo que quisieran que fuera y le faltan ciertas capacidades (por ejemplo no tiene acceso Web). Sin embargo, encontrarán que es difícil evaluar las ventajas técnicas y los riesgos de opciones que son muy diferentes entre sí (re-escribir versus reemplazar con un paquete versus transformación).

Debido a estas consideraciones, deberá de ser el Gerente de Sistemas quien desarrolle el caso de la transformación y quien explique los puntos a favor y en contra de las diferentes opciones que tienen los gerentes. El Gerente de Sistemas tendrá que incentivar a los analistas internos para que busquen todas las opciones. La experiencia muestra que ellos prefieren el diseño de una nueva solución por encima de la reutilización de la anterior y esta tendencia necesita ser contrarrestada por el Gerente de Sistemas. Alternativamente, puede ser necesario que se involucre a terceros para asegurar que se lleve a cabo un análisis objetivo.

Debemos sin embargo notar que los gerentes de las empresas necesitan defender el proyecto de transformación en general. Puede parecer un reemplazo de igual a igual de la aplicación existente, pero será necesario aceptar los resultados finales como equivalentes, y será necesario participar en cualquier cambio funcional. Y cuando hay problemas o retrasos (como siempre hay) deberá haber alguien que le recuerde a todos porqué se está haciendo el trabajo y los beneficios que obtendrá la empresa como resultado de esto.

6. Desarrollo del Caso

Estos son momentos difíciles para hablar de cualquier tipo de inversión para las empresas. La situación se agrava para las inversiones en legacy porque muchos de los supuestos de la alta gerencia y los modelos empresariales que han sido honrados por el tiempo son inadecuados para comprender que está sucediendo. La clave es encontrar la manera correcta de presentar el problema a la alta gerencia, explicándoles las desventajas de no invertir, y contrastando los riesgos y beneficios de las diferentes alternativas.

¿Qué hace que justificar una transformación de aplicaciones legacy sea tan difícil? Primero que nada, los gerentes están más cómodos con la idea de gastar dinero para comprar algo nuevo y de primera entrada la transformación de una aplicación legacy parece ser un proyecto para 'arreglar' algo que todavía funciona. Segundo, el proyecto legacy muchas veces es una inversión 'habilitadora' – o sea, que posiciona la empresa para lograr algo que no se lograba anteriormente. Por ejemplo, uno de los beneficios puede ser que sea más fácil agregar nueva funcionalidad, insinuando que la funcionalidad puede ser agregada de otra manera posiblemente menos costosa. Los beneficios indirectos de este tipo siempre son más difíciles de cuantificar y justificar. Tercero, los costos del proyecto de transformación deben ser aclarados, mientras que el costo real de la aplicación legacy que se tiene actualmente (interrupciones, mantenimiento, operaciones costosas, etc.) se encuentra oculto en otros presupuestos. Estos son los puntos que se presentan en esta sección.

"Si no está dañado, no lo repare"

Esta tiende a ser la primera reacción del gerente ante una solicitud para gastar dinero en la aplicación legacy. Como en otras situaciones en la vida, la gente se acostumbra a sus aplicaciones legacy, las incómodas interfases, las soluciones alternativas, el tiempo que se requiere para hacer cambios, el costo de mantenimiento y operación, etc. Cada usuario ha trabajado más de la cuenta y tiene menos personal del necesario y la gente no tiene tiempo para pensar en los posibles arreglos y mejoras. Puede ser que se quejen pero están muy ocupados para hacer algo al respecto. Entonces al menos que se dé una crisis, deberán comenzar el trabajo enseguida, muchas gracias.

La gerencia escucha las quejas pero al menos que los clientes exijan algo o se pasen a la competencia es poco probable que reaccionen. La gerencia está acostumbrada al tamaño de su presupuesto de mantenimiento y parece ser más fácil aprobar el mismo presupuesto para el año siguiente, sin tener que discutir al respecto. Por lo tanto es probable que cualquier propuesta para transformar una aplicación legacy va a recibir una respuesta negativa.

La superación de esta inercia es el primer paso para avanzar en la propuesta de transformación de aplicaciones legacy, y para hacer esto es necesario explicar los riesgos y costos de oportunidad de no hacer nada. Por ejemplo, el riesgo propuesto por el personal de mantenimiento saliente: Las aplicaciones legacy por su naturaleza son opacas, no tan fáciles de trabajar debido a las capas de cambios que se han llevado a cabo a lo largo de los años, y la falta de claridad en la documentación. (La documentación no siempre es creada para ayudar a los mantenedores, existe para ayudar al usuario, y para explicar cómo y porqué la aplicación fue creada de esa manera). De manare que toma tiempo conocer de cerca los aspectos importantes acerca de estas aplicaciones, y por eso existe la dependencia de la gente que brinda el mantenimiento y

les hace mejoras. Por supuesto que la gente puede ser reemplazada. Pero esto toma tiempo, existe una curva de aprendizaje involucrada, y si el lenguaje de programación es relativamente confuso, podría ser difícil conseguir el personal con las aptitudes necesarias. Otros riesgos potenciales son que los proveedores de componentes para plataformas retiren su soporte y la falta de sustituciones y actualizaciones para hardware y software.

A nivel de la empresa pueden haber riesgos para los futuros planes empresariales que están aun en el tapete. ¿Puede la aplicación legacy manejar un 50% más de clientes? ¿Podrá la gente de pre-ventas manejar tantas cotizaciones considerando la condición actual de la aplicación?

Un ejemplo de los costos de oportunidad es el tiempo del personal que podría ahorrarse con solo utilizar herramientas actualizadas. El personal actual de mantenimiento está ocupado dando mantenimiento (con dificultad) a una aplicación que está envejeciendo- este trabajo puede tomar menos tiempo y esfuerzo con una aplicación transformada debido a la disponibilidad de herramientas y su productividad inherente. Los cambios y mejoras serían ejecutadas más rápida y eficientemente y el personal en cuestión puede utilizar el tiempo ahorrado en nuevos desarrollos.

Estos argumentos basados en riesgo y costos de oportunidad(los costos de no hacer nada)deben ser complementados con una explicación de los beneficios específicos y cómo la inversión en legacy logra avances hacia las metas empresariales de la organización. Mientras que estos dependen del contexto, los beneficios típicos pueden incluir reducciones de costos a largo plazo, mejoras en la rapidez de entrada al mercado y un alcance empresarial extendido, solo para mencionar tres.

La inversión en el ciclo de vida del software

Después de las inversiones en aplicaciones legacy para el Y2K, la administración puede estar preocupada de gastar más en las mismas aplicaciones. La misma reacción es probable ante una solicitud de transformación para una aplicación legacy que tiene dos o tres años de existir. (Esta situación puede surgir, por ejemplo cuando el desarrollo de una aplicación tuvo inicio hace varios años, pero debido a diversos tipos de demoras, ya esta empieza a ser obsoleta.)

En cierta forma este es un problema similar al de la sección anterior, pero con una mayor dimensión ya que implica que existe una historia detrás de este asunto de aplicaciones legacy. El problema básico aquí es que a la gerencia no se le ha brindado un panorama acerca del futuro. En otras palabras, las expectativas de la gerencia no han sido administradas de forma adecuada. Esto se origina en la manera en que hemos tratado los proyectos grandes en el pasado, como proyectos relativamente aislados, eventos únicos, en lugar de que formen parte de un programa empresarial continuo. Las empresas han adquirido el hábito de pensar en los proyectos de aplicación como algo que se acaba cuando la aplicación es entregada, el equipo de creación desaparece, los consultores se van, y los usuarios empiezan el trabajo.

El argumento de justificación necesita enfocarse en la manera como se trata la inversión en software y transmitir la noción de que el software es (o debería ser) un activo no perecedero. Las empresas han hecho inversiones sustanciales en las aplicaciones legacy y su operación efectiva es esencial para manejar las empresas de hoy en día. Tiene sentido mantener una aplicación legacy si esta continúa satisfaciendo las necesidades de la empresa. Sin embargo el cambio en las fuerzas de la empresa, la

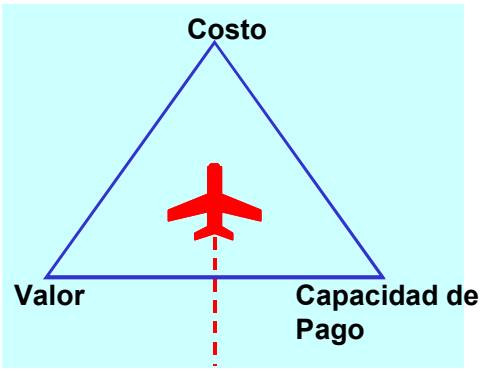
obsolescencia de la tecnología y el deterioro del know-how interno ponen las aplicaciones en riesgo.

Por eso es que las empresas invierten en mantenimiento y mejoras. Pero es la inversión en legacy como un carro, que debe ser reemplazado por el ultimo modelo cada cierto tiempo, o será más parecida a la compra de una casa, que requiere de atención y renovación regularmente, y esta se agranda cuando la familia crece? Los recientes desarrollos en herramientas y tecnología de plataforma hacen que sea práctico y juicioso adoptar la segunda similitud. Viéndolo desde esta perspectiva, la transformación de aplicaciones legacy se convierte en una manera de prolongar la inversión en software y muy posiblemente una manera de entregar mejores datos y mayor flexibilidad para actualizaciones futuras de expansión – y también cuesta menos.

Costo, Valor y Capacidad de Pago

No todas las justificaciones de las aplicaciones legacy comienzan en el mismo punto. Es necesario comprender dónde se encuentra este punto de inicio dentro de lo que podemos llamar el "Triángulo de las Bermudas" de Costo, Valor y Capacidad de Pago. La confusión y dificultad más común que encuentra el Departamento de Sistemas de Información para justificar las inversiones en proyectos 'habilitantes' (y la transformación de aplicaciones legacy es uno de estos, más que cualquier otra cosa) es esta incertidumbre. La razón detrás de esta dificultad es que una vez que se han abordado las objeciones con respecto a los costos, entonces se pasa a los asuntos de valor (valdrá la pena?) y cuando se ha resuelto esta incógnita, surge la duda de la capacidad de pago (tenemos la capacidad de pago?). Después de esto, la discusión pasa a ser porqué es tan caro. Y por tanto el argumento sigue dando vueltas.

La salida del triángulo se encuentra identificando la posición actual y trabajando luego en forma sistemática para lograr un balance adecuado entre los argumentos. En el ápice de "costo" la principal discusión es acerca de cuánto se debe gastar, cuáles son las opciones, y cuál es la opción correcta. En el ápice de "valor" por otra parte, el enfoque se hace sobre qué tan rentable sea la inversión- la objeción vuelve a ser "si no está dañado, no lo repare". Finalmente, en el ápice de "capacidad de pago", se establece un balance favorable entre costo y valor y las objeciones se centran en prioridades y otras oportunidades para gastar este dinero en la empresa.



El Triangulo de las Bermudas ilustra la posible trampa y naturaleza cambiante de las objeciones que surgen ante una propuesta de inversión en legacy. La salida del triángulo se encuentra identificando la posición actual y trabajando luego de forma sistemática para lograr el balance adecuado entre los argumentos.

Ubicación	*Punto principal*	*Síntomas*	*Forma de avanzar*
Ápice de costo	Solución escogida	• "Será este el monto correcto a pagar?" • Indecisión acerca de la opción escogida: "Reemplacemos la aplicación legacy" " No, debemos re-escribirla" • Incertidumbre acerca de cuánto costará la transformación de aplicaciones legacy	• Comparar las diferentes opciones • Incluir la opción de "no hacer nada" • Incluir los costos operativos, no solo los costos del proyecto • Pedir una cotización de precio fijo
Ápice de valor	Beneficios	• Incertidumbre acerca de su valor • "Si no está dañado, no lo repare" • "Existen muchos supuestos- no sabemos cuál va a ser el futuro de la empresa"	• Explicar los riesgos y costos de oportunidad de no hacer nada • Pedir a los usuarios que den un valor a la flexibilidad, confiabilidad y otras mejoras anticipadas. • Considerar las oportunidades para aprovechar la transformación a lo largo de la empresa
Ápice de capacidad de pago	Prioridades	• "Es una buena idea, pero será esta la mejor manera de gastar el dinero?" • "Tenemos otras prioridades este año" • "No hay suficiente presupuesto" • "Ya estamos gastando suficiente "	• Adoptar una perspectiva de administración de cartera- calificar los proyectos con respecto a su contribución para la empresa, el nivel actual de satisfacción técnica/costos para demostrar los beneficios relativos • Demostrar los ahorros • Compensar el gasto de capital uniéndole al proyecto de transformación algo en que todos estén de acuerdo (por ejemplo Acceso web por parte de los clientes)

Punta del iceberg

Un punto común en la discusión anterior ha sido el costo de no hacer nada. Una gran parte del costo de las aplicaciones legacy que se tienen actualmente se encuentra escondido en lo que podemos llamar el "efecto iceberg". En la superficie, los únicos costos son los recursos de mantenimiento. Pero los costos reales de la aplicación legacy incluyen equipo, software, personal (operaciones, manejo de problemas, coordinación, mantenimiento, capacitación, reclutamiento, etc.) comunicaciones (aranceles, soporte), e instalaciones(espacio, movimientos y cambios, energía, enfriamiento) . Los costos de la empresa pueden incluir interrupción del trabajo de la empresa, tiempo improductivo, horas extra del personal, trabajo que debe hacerse dos veces debido a fallas/errores, o impactos negativos del cliente.

La gerencia muchas veces no está consciente de los costos sustanciales de las aplicaciones legacy actuales– incluyendo su impacto comercial.

La razón por la cual es importante sacar esto a la luz es porque los costos de la propuesta de inversión en legacy serán visibles y podrían parecer muy altos si se comparan con los costos visibles de la aplicación actual. Es necesario comparar "manzanas con manzanas". De acuerdo con Gartner, de un 60 a un 80 por ciento del presupuesto típico de Tecnología de la Información se gasta en mantenimiento de aplicaciones de mainframe y las aplicaciones que corren sobre estas. Si la transformación de aplicaciones legacy puede cambiar esta cifra en tan siquiera un 10 por ciento, el impacto sería considerable. Se pueden obtener otras reducciones de la consolidación de servidores y almacenamiento– Gartner ha mencionado que los gastos en hardware son un 18 por ciento del presupuesto típico (y el mismo monto para personal operativo). La consolidación puede llevar a una reducción de un 20 por ciento en estos costos (debido a las mejoras en utilización y a que es posible compartir la carga). Por tanto es importante mostrar cuál es la cifra comparable para las aplicaciones legacy existentes.

La definición de costo en los libros de texto se refiere a " tanto los recursos mensurables y los difíciles de medir para producir bienes y entregar servicios.. el costo completo de cualquier objeto de costo…... es el costo de los recursos utilizados directamente para ese objeto más una parte del costo de los recursos utilizados en común para hacer todos los objetos"[2]. En este caso los costos están siendo recolectados para tomar una decisión, de manera que la asignación detallada de los recursos es secundaria. La asignación de costos operativos de esta forma puede ser un ejercicio complejo, pero la idea es mostrar la escala de los costos involucrados, y no sacar todos y cada uno de los detalles. La asignación de recursos deberá concentrarse en las áreas en que los costos varían si el proyecto de transformación avanza.

Aunque parezca obvio la administración tendrá la última palabra, y las cifras están disponibles para informar esta decisión.

[2] CH Brandon, RE Drtina *Management Accounting* McGraw-Hill (1997)

Este asunto de costos visibles/invisibles aplica igualmente a otras opciones que no son la transformación de aplicaciones legacy. Un ejemplo que hace al caso es el reemplazo legacy: Por ejemplo, los analistas que estudian otras opciones le pueden haber pedido al proveedor de ERP una cotización para reemplazar la aplicación (es) legacy. Pero es necesario ir más allá de esta cotización, para considerar los otros costos involucrados (muchas veces considerables), los cuales muchas veces están por encima del precio cotizado por el proveedor para hacer el trabajo. Estos son otra forma de "efecto iceberg" aunque es menos probable que aplique la regla 80/20:

- Crecientes costos de capacitación y para la administración del cambio debido a cambios en los roles y responsabilidades que imponen los nuevos modelos empresariales implícitos en el paquete ERP.

- La conversión de datos no es de igual a igual, por tanto el esfuerzo requerido es mayor y se distrae al personal operativo de sus tareas diarias.

- Volver a capacitar al usuario.

- Los datos en el nuevo paquete no quedarán uno a uno con la cobertura de la aplicación actual, y podría ser necesario hacer un análisis extenso para manejar estas brechas. Podría ser necesario hacer ajustes a otras aplicaciones desarrolladas internamente.

- La configuración y pruebas del paquete, especialmente de interfases a aplicaciones desarrolladas internamente, es altamente impredecible. Se deben tener reservas para imprevistos.

- Los costos de consultoría cotizados son para el trabajo como se describe. Aparte del obvio crecimiento descontrolado de los alcances, no poder transferir el know how al personal interno puede significar tener que apoyarse en los consultores a largo plazo y por tanto los costos de consultoría resultarían muy altos.

Uniéndolo todo

La transformación de las aplicaciones legacy es una tarea con riesgos y recompensas. Es más fácil apoyarse en lo que parecen aplicaciones estables y esperar que sean adecuadas para la empresa al menos a mediano plazo. Pero estas aplicaciones legacy están en el corazón de las operaciones de hoy y si se separan mucho de las necesidades de las empresas el impacto será considerable, y posiblemente catastrófico. El reto para el Gerente de Sistemas es presentar los argumentos para la inversión en legacy de la mejor manera posible pero también brindar a la administración el panorama completo de estos riesgos y recompensas de manera que estos puedan tomar la decisión con toda la información necesaria a su disposición. Por último, la transformación de aplicaciones legacy es un proyecto 'habilitante' que permite que sucedan otras cosas, pero a su vez tiene sus propios beneficios directos.

- Para venderle de forma exitosa un proyecto de transformación de gran escala a la administración, es necesario presentar evidencia decisiva de que el proyecto ahorrará dinero y fortalecerá a la empresa.
- Gastar dinero para mantener las aplicaciones legacy "así como están" puede ser un error. Hacer planes empresariales basados en estas aplicaciones legacy es otro error. Los costos y riesgos asociados con la opción de "no hacer cambios" deben ser detallados.

- El rendimiento sobre la inversión esperado del proyecto de transformación provendrá en parte de la reducción proyectada en costos de mantenimiento y costos de funcionamiento de la nueva aplicación en comparación con la aplicación legacy.
- La justificación más importante para comenzar este tipo de proyecto es sin embargo la necesidad de la empresa.
- Previamente se deben elaborar los argumentos adecuados sobre costo, valor y capacidad de pago para poder argumentar las posibles objeciones.
- Las aplicaciones no están terminadas cuando el proyecto se termina. Los ciclos de vida de las aplicaciones requieren de que se haga un gasto continuo para asegurar la efectividad empresarial y técnica. La transformación de las aplicaciones legacy debe ser presentada como parte normal de mantenimiento y renovación de activos, no como una única misión de rescate poco común, para así decirlo.
- Actualizar el caso de la empresa conforme el trabajo avanza para asegurar visibilidad a los que velan por la empresa(la gerencia, usuarios, equipo del proyecto, proveedores) y para facilitar que se hagan revisiones periódicos.

7. La Situación de la Oferta

La discusión se ha concentrado hasta ahora en tecnología y proyectos de transformación. En esta sección la discusión pasa a la situación de la oferta y dónde deben buscar los gerentes para comprar la tecnología de transformación y servicios relacionados. El concepto de "cadena de valor de la transformación de aplicaciones legacy" es introducido como una forma para comprender los roles de los diferentes participantes del lado de la oferta. Esta es una manera de pensar acerca del rol que cada tipo de participante adopta en la cadena de valores- partiendo de esto el posible comprador puede ver el panorama de cómo puede seleccionar proveedores adecuados.

Comprendiendo la situación de la oferta

El trabajo necesario para lograr la transformación tiene varias dimensiones- definición de requisitos, traducción de programas, pruebas, administración del cambio, instalación de nuevas plataformas, etc. Como se explico anteriormente, el estado actual del desarrollo del mercado quiere decir que hay algunas situaciones en que el comprador puede contar con encontrar un solo proveedor que satisfaga todas sus necesidades, y el know how especializado necesario por lo general hace que sea poco práctico hacer el trabajo internamente. El comprador estará buscando comprar algunos (o todos) productos y servicios necesarios, pero no querrá tratar con muchos proveedores, para evitar los riesgos de las responsabilidades compartidas y la confusión acerca de quién hace qué. ¿Entonces con quién debe tratar el comprador?

Un buen punto de inicio para comprender la situación de la oferta es la "cadena de valor de la transformación de aplicaciones legacy". Este diagrama muestra una cadena de valor de una transformación de una aplicación legacy. Esta cadena de valor muestra diferentes tipos de participantes- los proveedores de plataformas y proyectos habilitantes, proveedores de productos de transformación y servicios, y los consumidores- el departamento de Sistemas de Información y el usuario final (y posiblemente contratistas externos).

Las categorías son un poco arbitrarias pero esencialmente reflejan la fuente de ingresos del proveedor. Nótese que un solo proveedor puede adoptar más de un rol. Por ejemplo, vender días consultor así como licencias de software.

La mejor manera de categorizar los proveedores es considerar de dónde provienen sus ingresos. Cada participante en la cadena de valor tiene diferentes generadores de ingresos. Los participantes en la parte superior de la cadena de valor (a la izquierda del diagrama) están buscando ingresos continuos, de licencias y actualizaciones, hardware y sistemas operativos, capacitación y soporte, etc. Los participantes del medio están involucrados principalmente en la transformación del proyecto en sí, de manera que obtendrán esencialmente ingresos únicos. Por su parte los contratistas externos y el departamento interno de Sistemas están preocupados por el costo de operación y los ingresos provenientes de los usuarios finales y la empresa.

La cadena de valor tiene tres implicaciones para el desempeño de la situación de la oferta:

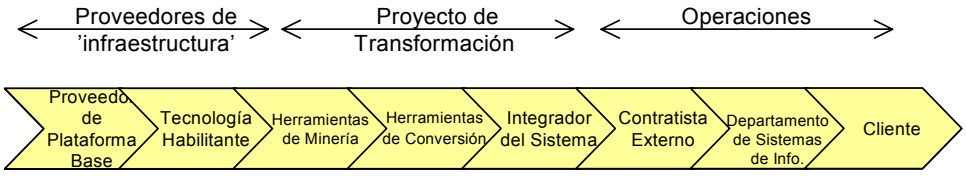
- *La posición en la cadena de valor es el impulsor para las estrategias del proveedor.* Por ejemplo, a los proveedores del lado de la infraestructura les interesa regalar herramientas de migración "gratis" ya que sus ganancias provienen de las licencias

de software, actualizaciones de productos y soporte en general– todos son ingresos iterativos a lo largo del ciclo de vida de las aplicaciones transformadas. De la misma forma, los integradores pueden encontrar que es rentable hacer lo mismo, ya que sus ganancias provienen de los servicios prestados. Esto puede resultar ser una restricción para los proveedores de juegos de herramientas para la transformación. La desventaja para el comprador es que la escogencia de tecnologías puede ser determinada por las relaciones empresariales existentes, en lugar de ser determinada por la mejor herramienta para la tarea que hay que realizar.

- *Sociedades en la cadena de valor.* Los participantes en la parte superior de la cadena de valor establecerán sociedades con participantes más cercanos al comprador tales como los integradores, que esencialmente son "dueños" de la relación con el cliente. El integrador que maneja el proyecto de transformación tenderá a tomar decisiones claves sobre herramientas, infraestructura, etc. El inverso puede ser el caso ocasionalmente, por ejemplo si se ha hecho la venta de infraestructura, y el vendedor de infraestructura está tratando de ayudar al comprador a mover todas sus aplicaciones con la ayuda de un socio integrador.

- *La automatización no es el principal interés de los proveedores de infraestructura y los integradores,* pero menos automatización equivale a mayor costo y mayor probabilidad de que haya errores: los proveedores de infraestructura y los integradores tenderán a tomar una decisión que les favorezca. Por ejemplo, el integrador gana más con el esfuerzo manual (ya que su ingreso es una función de los días hombre que se trabajen), pero esto le cuesta más al comprador, y la intervención manual puede hacer que se introduzcan nuevos errores en la aplicación. Los proveedores de infraestructura tienen un know how limitado de la aplicación y sus honorarios profesionales son caros y orientados hacia sus productos. De una u otra manera, el comprador será siempre quien pierda si no se le da el mejor uso a las herramientas y tecnologías de transformación.

Cadena de Valor de la transformación de aplicaciones Legacy



Ejemplos	Microsoft Oracle Sun SAP IBM Jacada	IBM Websphere Acucorp Oracle9iAS BEA	Relativity Netron Cyrano Allen Systems SEEC CAST	ArtinSoft Tominy SWS S/W Services	CMG Progeni Adpac SEER Computing LegacyJ Modis Solution NIIT S/W Solns	EDS Capita		
Impulsores de ingresos *(ejemplos)*	No. usuarios Actualizaciones Tamaño de la instalación	Tráfico esperado en la Web No. usuarios Actualizaciones	No. de participantes en el equipo de proyecto	Líneas de código convertido Días trabajados en el proyecto	Días trabajados en el proyecto	Volúmenes de Transacción procesados	Presupuesto basado en costo continuo de operación	
Duración de Ingresos	Continuos	Continuos	Proyecto	Proyecto	Proyecto	Contrato		
Fortaleza Competitiva	Mercadeo Propietario de la Tecnología	Limita el grado de cambio requerido	Especialista con técnicas propietarias	Especialista promete resultados rápidos	Buenas relaciones interpersonales, experiencia	Economías de Escala	Portero	
…y debilidades	Lejos del cliente	Solución específica, no aborda los problemas básicos de la transformación	Requiere personal altamente especializado Deja pendiente mucho del trabajo que debe hacerse	Únicamente parte de la solución requerida Cierta intervención manual es necesaria	Historial rara vez calza de manera adecuada, por lo tanto la credibilidades limitada	No hay historial o este es muy limitado	Recursos limitados (cantidad de gente, destrezas) No se cuenta con experiencia	

Implicaciones para los compradores

La mejor estrategia es obtener un balance entre los participantes orientados hacia el proyecto (los que se encargarán de la transformación), y los proveedores de infraestructura y personal interno quienes tienen que hacer que el resultado final funcione diariamente, y asegurar que los conocimientos técnicos en transformación estén en el corazón de la solución que se va a entregar.

Al mismo tiempo, el comprador querrá evitar tener que tratar directamente con una serie de vendedores y el riesgo de que los proveedores puedan "echarle la culpa a otros" cuando surgen dificultades.

La mejor estrategia para el comprador es la siguiente:

- Mantener la administración del proyecto independiente. Nombrar un gerente de proyecto interno o utilizar un contratista o consultor en que se tenga confianza.

- Planificar el proyecto, agrupar la traducción de código, migración de datos y pruebas relacionadas, y mantener el resto del trabajo separado. Esto aclarará a quién corresponden las responsabilidades y logrará que se haga un mejor uso de los conocimientos especializados disponibles.

- Investigar las herramientas de traducción más adecuadas, migración de datos y actividades de reutilización, basadas en la situación específica del comprador y ya sea encontrar un integrador calificado, o tratar directamente con los proveedores involucrados si es un proyecto de transformación de tamaño considerable. Por ejemplo, un trabajo considerable de traducción algunas veces puede beneficiarse de las adiciones hechas a la medida para los algoritmos de traducción, y el proveedor está en la mejor posición para hacer esto.

- Escoger un socio de integración que involucre al personal interno en los cambios y en la integración subsecuente de manera que haya transferencia de know-how. Utilizar el integrador para subcontratar servicios de proveedores de infraestructura, tanto para administrarlos más eficientemente como para compartir con el integrador el riesgo de sobrepasarse en costos.

- Considere correr un proyecto de "prueba del concepto" en una aplicación más pequeña o menos crítica para la empresa para poder evaluar el desempeño de los proveedores, y lograr que el equipo interno avance en la curva de aprendizaje con un ambiente de bajo riesgo. Esto también nos dará una mejor indicación de los costos.

8. Perspectivas para la transformación de aplicaciones Legacy

Las aplicaciones legacy tienen un papel vital en el panorama del comercio. El servicio que estas brindan es crítico para la función diaria de la industria en general. Por ejemplo concebir soluciones de comercio electrónico sin explotar las aplicaciones legacy sería ignorar la clave para hacer que el comercio electrónico integrado de principio a fin se convierta en una realidad.

La queja más común acerca de las aplicaciones legacy es que son muy viejas, muy inflexibles y muy desactualizadas para agregar valor real a la evolución de las soluciones de sistemas de la industria. Mientras que en un mundo ideal la tecnología del año anterior sería descartada cada año para reemplazarla con lo último en diseño de sistemas y funcionalidad, la realidad y la prudencia evitan que esto suceda. En el caso de sistemas empresariales en que la misión es crítica, esto no es ni práctico ni prudente. Es necesario repensar cómo el ciclo de vida de las aplicaciones legacy es administrado. Este estudio plantea que la transformación es factible, se está haciendo más fácil, y con la escogencia adecuada de estrategia y administración de proyectos presenta una alternativa real para reemplazar o reescribir aplicaciones partiendo de cero.

La transformación no puede darse de un día para otro porque involucra la colaboración de muchos componentes con valiosa información y funcionalidad en juego. Sin embargo la disponibilidad de las herramientas y procesos puede simplificar y agilizar el proceso y, como incluso una breve consideración de factores lo puede revelar, las alternativas pueden ser aun más costosas y consumir más tiempo y ser más riesgosas para la empresa.

¿Cuál es el panorama para el futuro? Las estrategias actuales de los vendedores tienen un enfoque un poco limitado y los proveedores están más interesados en elaborar sus propias herramientas y servicios. Sin más esfuerzo cooperativo a través de la cadena de valor, esta situación continuará perjudicando la aceptación de la transformación como una manera legitima y efectiva para avanzar.

Por otra parte, la demanda por tener acceso a los beneficios de las herramientas de desarrollo actuales para ambientes Java y .NET pueden impulsar la transformación y aumentar la demanda por productos y servicios de transformación. Estos ambientes incluyen generadores de código, editores inteligentes con “wizards” incorporados, herramientas de rastreo de lógica y medios de depuración. Los analistas pueden crear modelos de diseño que suministran especificaciones a los productos de desarrollo. Estas herramientas son parte de los ambientes de desarrollo integrado que sincronizan los modelos empresariales, especificaciones y lógica del programa a lo largo del ciclo de desarrollo. Los cambios en reglas de la empresa en un modelo de diseño se ven reflejadas en el código fuente del sistema y un cambio de código también se ve reflejado dentro de un modelo de diseño. Dicho desarrollo y mantenimiento impulsados por el modelo son una manera muy efectiva y eficiente de desarrollar sistemas en el largo plazo.

Otro factor es la falta de recursos calificados: Existen muchas aplicaciones legacy críticas y muy pocos técnicos capacitados para trabajar con estas. La transformación brinda la oportunidad de aumentar la efectividad de los programadores de las aplicaciones legacy por medio de mejores herramientas de análisis, desarrollo, actualización, depuración y componentización. Tiene sentido migrar las aplicaciones legacy a estos ambientes en lugar de buscar las herramientas para ser retromodificadas para los ambientes legacy.

Al poner estos factores en perspectiva, la transformación de aplicaciones legacy actualmente se encuentra en una etapa temprana de aceptación, pero si el interés en J2EE y .NET sigue aumentando y los usuarios aprovechan los beneficios de estas plataformas, entonces el futuro de la tecnología es positivo.

Bibliografía Adicional

Los siguientes artículos y reportes me ayudaron a formar mis ideas:

Anon *Application Mining – what* 24th Agosto 2001 http://www.it-analysis.com/article_pf.php?id=1575

John Bergey, Liam O'Brien, Dennis Smith *DoD Software Migration Planning* Technical Note CMU/SEI-2001-TN-012 Carnegie Mellon University, Agosto 2001

A Bainbridge, J Colgrave, A Colyer, G Normington *CICS and Enterprise Java Beans* IBM SYSTEMS JOURNAL, VOL 40, NO 1, 2001 http://www.ibm.com

Michael L Brodie, Michael Stonebraker *DARWIN: On the incremental migration of legacy information systems* Technical memorandum TR-0222-10-92-165 Electronics Research Laboratory, College of Engineering, University of California, Berkeley, Marzo 1993

Joe Celko *Breaking tradition* Intelligent Enterprise, Febrero 21, 2002 http://www.intelligententerprise.com

Arie van Deursen, Paul Klint, Chris Verhoef *Research Issues in the Renovation of Legacy Systems* ETAPS Conference, 1999

Jim Duggan *Graceful Retirement: Your Applications, Not You* AV-15-4789 Gartner Group, Febrero 2002-08-15 http://www.gartner.com

Len Erlikh *Unlock the power of your legacy systems* Relativity Technologies Inc, 2000 http://www.relativity.com

Len Erlikh, Mike Ferris *Business-Rule Extraction from Cobol to Java* 1998 http://www.devx.com/premier/mgznarch/javapro/1998/JP_junjul_98/le0698/le0698.asp

Franck Gonzales *What's ahead for client/server architecture?* Owendo, Francia 2001 http://www.owendo.com

Simone Kaplan *Despite the sluggish economy and uncertain business climate, right now is the perfect time to tear down your legacy applications and start over.* CIO Magazine Marzo 15 2002 http://www.cio.com/archive/031502/infrastructure_content.html?printversion=yes

Anthony Lauder, Stuart Kent *Legacy system anti-patterns and a pattern-oriented migration response* Computing Laboratory, University of Kent en Canterbury, 2000 http://www.ukc.ac.uk/research/publications/2000/compsci.pdf

Jerry Loza *Is a switch to Java a good move for you?* Junio 27, 2001 http://www.techrepublic.com/article.jhtml?id=r00820010627ggp01.htm&src=bc

Amey Stone *Keeping legacy software alive* Business Week, Junio 14, 2001 http://www.businessweek.com

Scott Tilley *The Net Effects of Product Lines* SEI Interactive, Septiembre 1999 http://interactive.sei.cmu.edu

Chad Vawter, Ed Roman *J2EE vs Microsoft.NET: A comparison of building XML-based web services* The Middleware Company, 2001 http://www.middleware-company.com

D. Vecchio, J. Sinur, J. Duggan *Legacy Evolution: Not as Black and White as It Seems* TU-10-1478 Gartner Group, 24 Febrero 2000 http://www.gartner.com

Richard Veryard *Identifying Web Services* Interact, CBDi Forum, Febrero 2002 http://www.cbdiforum.com

Ben Wilson *Chickens and turkeys migrate, but not necessarily in IT* ANUBEX, Febrero 2002 http://www.falconsoft.be

Legacy Value Legacy Value Restoration Cognizant Technology Solutions, White Paper, November 2001 www.cognizant.com

Parallel Operation of Software: Is it a Desirable Transition Technique? Joint Financial Management Improvement Program, Julio, 2001 http://www.jfmip.gov

Web Services Triple Tree Spotlight Report, Febrero 15, 2002 http://www.triple-tree.com

WebIT™ Web-Enabling of Legacy Applications Intercomp White Paper, Agosto 2001 http://www.intercomp.com

Glosario de términos y abreviaciones

Contenedor de Aplicación	Los componentes bajo el estándar J2EE están agrupados en y corren dentro de los contenedores de aplicación. (los componentes son archivos de código que son accesados por la aplicación durante el tiempo de ejecución. Un contenedor brinda apoyo durante el tiempo de ejecución para los componentes y brinda un panorama unificado de los servicios J2EE. Cada tipo de componente (por ejemplo, EJB, JSP, "servlet", "applet", o aplicación de cliente) tiene un contenedor específico para el tipo de componente. Bajo .NET, los componentes corren dentro del contexto del "Common Language Runtime" (CLR). El CLR brinda muchas de las mismas funciones que el contenedor J2EE.
Minería de Aplicación	La minería de aplicación es utilizada para extraer las reglas de la empresa del código legacy.
B2B	"Business to Business". (Empresa a Empresa)
" Copybook"	Un fragmento común de código fuente diseñado para ser copiado dentro de muchos programas fuente. Comúnmente utilizado por programadores de COBOL . En IBM OS, se les llama "bibliotecas" y son implementados como "conjuntos de datos divididos'.
EJB	"Enterprise Server Beans". Los "beans" son componentes J2EE individuales que brindan funcionalidad comercial. Pueden administrar su propia persistencia o delegar esta función a su contenedor. Existen 3 tipos: "Session beans", "entity beans", y "message beans". El Contenedor EJB (frecuentemente denominado el servidor de aplicación) brinda administración de transacciones, seguridad, procesamiento a distancia (por ejemplo llamar objetos distribuidos), ciclo de vida de los objetos, y centralización de servicios de conexión a los "beans". Los "beans" son automáticamente agrupados por el contenedor cuando es apropiado, eliminando la necesidad de crear y destruir continuamente.
IDE	"Integrated Development Environment." Visual Studio .NET es un buen ejemplo - este elimina del desarrollo las tareas más difíciles y que consumen más tiempo . Una serie de IDEs son los que soportan la plataforma J2EE. El Visual Age de IBM y BEA's WebGain Studio son dos IDEs, Borland JBuilder también es utilizado.
IIOP	El Internet Inter-ORB Protocol. Protocolo que habilita CORBA por medio de la Internet. Habilita los "browsers" y servidores para intercambiar números enteros, tablas y objetos más complejos, a diferencia de HTTP, que solo soporta la transmisión de texto.
IO	Entrada- Salida (Input-Output).

ISAM	Indexed Sequential Access Method. (Método de Acceso Secuencial Indexado)
Java	Un conjunto de tecnologías para crear y correr programas de software . Una marca registrada de Sun.
JSP	Java Server Page. (Página de Servidor Java)
JCA	Java Connector Architecture. (Arquitectura de Conector Java) Esta diseñada para conectar ambientes J2EE sincrónicamente con aplicaciones y procesadores de transacción. Es una adaptación del Marco Común de Conectores de IBM.
JCL	"Job Control Language". Un conjunto de declaraciones que controlan la ejecución de una serie de tareas.
J2EE	"Java 2 Platform, Enterprise Edition". Habilita las aplicaciones JAVA de gran escala para ser creadas consistentemente en un ambiente distribuido. J2EE brinda un contenedor que administra componentes, un conjunto de especificaciones en cuanto a cómo operan estos componentes, y un conjunto de interfases estándar.
Aplicación Legacy	Una aplicación legacy puede ser definida como una aplicación basada en tecnologías y hardware más viejos, tales como "mainframes", la cual continua brindando servicios esenciales a una organización.
Rastreo de Lógica	Una función de un IDE que rastrea la ejecución de código de programa utilizado para depuración.
MQ	"Message Queue" (Fila de Mensajes). WebSphere MQ de IBM (anteriormente MQSeries) y el MSMQ de Microsoft hacen posible que los mensajes sean enviados de una aplicación o servicio a otro con una entrega garantizada.
.NET	.NET no son siglas pero se usan en mayúscula. Se refiere al conjunto de herramientas de Microsoft para aplicaciones de desarrollo así como estándares y la filosofía basada en la Web para todos los productos y servicios de Microsoft. Las implementaciones .NET actuales corren en plataformas de Windows.
"Refactoring"	Restructuración de código de programa para mejorar su capacidad de mantenimiento, legibilidad, etc.
SOAP	"Simplified Object Access Protocol". (Protocolo Simplificado de Acceso a Objetos) Protocolo de comunicación de Servicios Web brindando un formato con base en XML para transportar mensajes e invocar servicios por medio de Internet.
XML	"Extensible Mark-up Language" – Formato de datos que pueden ser leídos por personas y máquinas. Valores de datos y meta datos ambos están incluidos en los datos para brindar una sintaxis auto descriptiva. Estándar publicado por el "World Wide Web Consortium".
Servicios Web	Una colección de servicios o capacidades empresariales tomadas de una sola o múltiples aplicaciones que pueden ser publicadas en una red utilizando protocolos basados en XML, para ser accesados por otras aplicaciones.

Wizard	Un utilitario en un programa que desglosa una serie de tareas secuenciales para establecer una porción del programa.
"Wrapping o wrappers"	"Wrapping o wrappers" (Envolturas) habilitan el acceso a la función legacy desde un ambiente orientado a objetos. El "object wrapper" opera como un método de llamado de un objeto, y luego utiliza un procedimiento tradicional para ejecutar una función de aplicación existente. Es un método de caja negra que aísla las complejidades internas de los programas legacy. Puede crear problemas de mantenimiento a largo plazo.