9. Metodos y modelos de la reingeniería de software

Este Capítulo explica de una forma muy breve los métodos y modelos que son utilizados para llevar a cabo de manera satisfactoria la actividad de reingeniería. Al comienzo del Capítulo se alude al método de análisis de opciones para reingeniería (OAR por sus siglas en ingles de Options Analysis for Reingeneering) dando su definición y explicando la necesidad de aplicarlo, se menciona de forma breve las actividades principales y especializadas de este método así como la estructura de ellas. En los siguientes dos subtemas se trata a dos modelos utilizados para la reingeniería: El modelo herradura y el modelo cíclico, explicando brevemente los niveles o actividades de cada uno.

9.1 El método análisis de opciones para la reingeniería ("OPTIONS ANALYSIS FOR REINGENEERING" (OAR))

9.1.1 Definición y Necesidad Del Análisis De Opciones Para Reingeniería

El Análisis de Opciones para Reingeniería (OAR por sus siglas en ingles de Options Analysis for Reengineering) es un método sistemático, de arquitectura central y de toma de decisiones para la identificación y extracción de componentes dentro de grandes y complejos sistemas de software. La extracción envuelve rehabilitación de partes de un sistema viejo para su re-uso. OAR identifica componentes de arquitectura potencialmente relevantes y analiza los cambios requeridos para usarlos en una línea de producción de software o nuevas arquitecturas de software. En esencia, OAR proporciona un conjunto de opciones de extracción junto con estimación de costos, esfuerzo y riesgos asociados con estas opciones.

Muchas organizaciones se comprometen a esfuerzos para actualizar sus sistemas de software intensivos y para establecer más formas eficientes de desarrollo de software. Una tendencia emergente había sido la implementación de líneas de producción de software para realizar medidas de economías y grandes eficiencias en el desarrollo de de software [Clem 01].

Desde entonces muchas organizaciones tienen una substancial base de software heredado activo, algunas líneas de producción de desarrollo o esfuerzos de migración. Sin embargo, hasta ahora, estas habían sido formas no sistemáticas para identificar componentes para re-uso y para entender los tipos de cambios requeridos para insertar componentes de sistemas heredados dentro de una línea de producción de software o una nueva arquitectura [Mull 00]. En la mayoría de los casos, las opciones habían sido para cualquiera experimentar el costoso y alto riesgoso proceso

de reingeniería de un sistema completo o para simplemente crear los componentes requeridos o sistemas desde cero.

La extracción de componentes casi siempre había sido discutido como una alternativa, pero requería el entendimiento de que tipos de componentes valían la pena extraer y como se debería extraer. Los siguientes puntos son motivos para el cambio:

Componentes existentes casi siempre eran pobremente estructurados y documentados.

Componentes existentes diferían en niveles de granuralidad.

No había una guía clara sobre como salvar componentes.

OAR proporciona un acercamiento sistemático para direccionar esos puntos y tomar decisiones requeridas para el costo efectivo y eficiente de extraer componentes de sistemas heredados.

El método OAR consiste de cinco actividades principales con tareas escalables. Para ver el gráfico seleccione la opción "Descargar" del menú superior

Estas actividades serán abordadas brevemente en las siguientes secciones.

9.1.2 Actividades principales del método

9.1.2.1 Establecimiento del contexto de extracción (ECE).

Es importante para el equipo de OAR entender el contexto para la extracción. Cómo un resultado, la primer actividad de OAR consiste en entrevistar a los accionistas y estudiar la línea de producción de la organización o nuevos requerimientos de sistema, base heredada y expectativas para la extracción de componentes heredados. Estos esfuerzos establecen una línea base de un conjunto de metas, expectaciones y necesidades de componentes. Esto también descubre los controladores de programa y técnicos para la toma de decisiones.

9.1.2.2 Inventario De Componentes (IC).

Después del ECE, el equipo OAR identifica los componentes del sistema heredado que potencialmente pueden ser extraídos para usarlos en una línea de producción o en una nueva arquitectura de software.

Durante esta actividad, los miembros del equipo identifican componentes de líneas de producción necesarios y evalúan los componentes heredados basados en esos

criterios. Aquellos que no descubran los criterios están incapacitados para continuar con el proceso de reingeniería. Esta actividad resulta en un inventario de los componentes heredados candidatos. La actividad de IC tiene seis tareas:

Identificar características de los componentes necesarios:

- Determina las características requeridas de los potenciales componentes heredados.
- Identifica las características satisfactorias de los componentes:
- Crea una tabla de componentes heredados con detalles de sus características.
- Filtra los componentes que no satisfacen las características requeridas.

Compara las necesidades de componentes:

- Compara los componentes heredados en contraste con la línea de producción de componentes necesarios.
- Inventario de componentes candidatos:
- Actualiza la tabla de componentes con más detalles acerca de los componentes candidatos seleccionados.
- Produce tópicos de extracción
- Revisa cualquier tópico de extracción e inquietudes que fueron identificados durante la actividad.

Revisión del calendario OAR:

- Actualiza el calendario de OAR si fuera necesario.
- Análisis de componentes candidatos (ACC).

El siguiente paso de los miembros del equipo es analizar el conjunto de candidatos de componentes heredados para extraer los tipos de cambios que son requeridos. Esta actividad tiene seis tareas:

- Selección de componentes deseables.
- Determinar criterios deseables para cada componente heredado.
- Deja fuere aquellos que no satisfacen esos criterios.
- Identifica los componentes "Tal como están y de caja negra"
- Determina aquellos componentes que pueden ser tapados o usados "tal como están".
- Identifica componentes de caja blanca.
- Determina aquellos componentes que necesitarán ser modificados.

Determina cambios requeridos:

- Determina los tipos de cambio que cada componente necesitará, el costo y esfuerzo involucrados, el nivel de dificultad y riesgo, el costo y esfuerzo comparativo para el desarrollo de componentes desde cero.

Producción de tópicos de extracción:

- Revisa cualquier tópico de extracción e inquietudes que fueron identificados durante la actividad.

Revisa el calendario OAR:

- Actualiza el calendario OAR si fuera necesario.
- Plan de opciones de extracción (POE).

Dado el conjunto de componentes candidatos analizados, el equipo desarrollar alternativas para la extracción basada en consideraciones de calendario, costo, esfuerzo, riesgo y recursos. El equipo OAR también filtra una vez más los componentes candidatos y analiza el impacto de agregación de diferentes componentes.

El POE tiene siete tareas:

- Selecciona componentes favorables:
- Desarrolla criterios, tales como costo o niveles de esfuerzo requeridos.

Ejecución de intercambio de componentes:

- Identifica un componente (o combinación) por linea de producción de componentes necesarios.
- Forma opciones de componentes:
- Desarrolla criterios para agregar componentes.
- Determina costos comparativos y esfuerzos:
- Compara el costo por cada agregación con el costo de desarrollar desde cero.

Analiza dificultad o riesgo:

Determina el nivel de dificultada y el riesgo involucrados por cada agregación.

9.1.2.3 Producción de tópicos de extracción:

Revisa cualquier tópico de extracción e inquietudes que fueron identificados durante la actividad.

9.1.2.4 Revisa el calendario OAR:

Actualiza el calendario OAR si fuera necesario.

Selección de opciones de extracción (SOE).

Finalmente, los miembros del equipo seleccionan la mejor opción de extracción o combinación de opciones para programas y consideraciones técnicas. Después de evaluar cada opción de extracción, ellos preparan un resumen que presenta y justifica sus elecciones.

La actividad SOE tiene cinco tareas:

Elegir la mejor opción:

Determina los controladores para seleccionar entre las opciones.

Selecciona una opción o combinación de ellas.

Verificación de opción:

Guarda todos los detalles acerca de cada una de las opciones escogidas.

Identifica componentes necesarios satisfechos.

Completa la lista final de componentes necesarios satisfechos y no satisfechos a través de las opciones seleccionadas.

Presentación de descubrimientos.

Prepara la presentación de descubrimientos que proporciona detalles acerca de las opciones seleccionadas.

Producción de resumen.

Producción de un reporte final detallando las opciones seleccionadas y las razones para esas elecciones.

9.1.3 Tareas Especializadas.

Cada actividad también tiene un potencial conjunto de actividades especializadas para direccionar circunstancias que pueden de otro modo imposibilitar la

cumplimiento de la actividad. Estas tareas pueden aplicarse bajo las siguientes condiciones:

El criterio existente para las actividades prescritas no han sido satisfechas.

Mas trabajo puede ser requerido para que la actividad sea razonablemente completada en la actividad de las tareas involucradas.

Se requieren datos adicionales para completar una tarea particular o para direccionar necesidades especiales del cliente.

Existe una necesidad de complementar tareas estándares OAR para afinar la toma de decisiones y reducir riesgos.

La siguiente sección incluye ejemplos de tareas especializadas.

9.1.4 Estructura De Actividades

Cada actividad esta compuesta de tareas y sub-tareas diseñadas para contestar un conjunto de preguntas de actividades especificas. Esas preguntas definirán la actividad y también servirán como una lista de comprobación para ser incluidas en los criterios de cada actividad.

Las actividades están estructuradas de acuerdo a un formato "EITVOX (Entry Criteria, Inputs, Task, Validation, Outputs, Exit Criteria)" [Berg 01] que se muestra a continuación:

Para ver el gráfico seleccione la opción "Descargar" del menú superior

Las siguientes secciones muestran cómo la actividad ECE es estructurada de acuerdo al el formato EITVOX. Cada una de las otras actividades siguen este formato. Bergey et al., proporciona este nivel de detalle. [Berg 01]

9.1.4.1 Ejemplo De Actividad: Establecimiento Del Contexto De Extracción.

Preguntas fundamentales.

Las tareas de ECE fueron desarrolladas para direccionar un conjunto de cuestiones que incluyen:

¿Qué esperan los administradores del esfuerzo de extracción?

¿Cuáles son los requerimientos (ej. Calidad de atributos) y controladores primarios (ej. Prioridades y restricciones) para la extracción de componentes?

¿Qué restricciones explicitas (e implícitas), reglas y suposiciones aplican?

¿Cuál línea de producción de componentes específicos necesitarán ser direccionados?

¿Cuáles sistemas heredados son relevantes y que documentos están disponibles?

¿Qué documentos disponibles describen los componentes heredados (ej. Funcionalidad, granuralidad e interfaces)?

¿Cuál es el nivel del estado de preparación de las organizaciones en los términos de experiencia, habilidades, recursos disponibles y margen de tiempo de OAR?

El conjunto completo de estas cuestiones fundamentales se revisan durante la validación para asegurar que fueron respondidas satisfactoriamente.

Criterios de entrada.

Los siguientes criterios de entrada se analizan para determinar si hay suficientes datos y tecnológica disponible para ejecutar la actividad exitosamente. Los criterios de entrada para esta actividad son:

La organización ha decidido mover una línea de producción y quiere calcular rápidamente la viabilidad de la extracción de componentes.

La arquitectura de línea de producción ha sido definida y necesita componentes que han sido identificados.

Un conjunto de metas de extracción y objetivos han sido establecidos.

Uno de tres sistemas heredados ha sido identificado como la fuente de componentes para la extracción.

Un "equipo de extracción" ha sido establecido y esta disponible para la duración de actividades de OAR.

Si el criterio de entrada no es satisfecho, una tarea especializada debe ser desarrollada.

Artefactos de entrada.

Cada actividad se basa en un conjunto de artefactos de soporte. Los siguientes artefactos se requieren para el establecimiento del contexto de extracción:

Metas y objetivos para el esfuerzo de extracción.

Requerimientos de la línea de producción y componentes necesarios.

Documentación que describe el alcance de la línea de producción, arquitectura y especificaciones de componentes.

Visión general del sistema heredado y documentación de componentes.

Reportes de experiencia de otros esfuerzos de extracción/reingeniería.

Perfil de calendario OAR típico.*

La actividad ECE se divide en diez tareas. Esas tareas son ejecutadas en orden. Varias de estas tareas son subdivididas. Para ayudar al equipo OAR a ejecutar las tareas y sub-tareas, el SEI (Software Engeneering Institute) desarrollo plantillas de ejecución y datos. Estas plantillas de ejecución muestran los pasos, fundamentos y suposiciones para las tareas y sub-tareas. Las plantillas de datos proporcionan ejemplos típicos y ofrecen varios puntos de inicio para producir información de los clientes.

Las tareas logran lo siguiente:

-**Revisión de metas y objetivos:**

Determina las metas y objetivos de los accionistas para el esfuerzo de extracción.

Revisión de Línea de producción / Nuevos requerimientos de sistema:

Identificar línea de producción o nuevos requerimientos de sistema.

Revisión y selección de componentes necesarios:

Identificar componentes necesarios que deberán ser direccionados para la extracción.

---

* El "Perfil de calendario OAR" es una plantilla proporcionada por el SEI

Estructura de tareas.

Revisión de sistemas heredados y documentación:

Revisión de sistemas heredados y documentación de componentes disponibles.

Determinar controladores de extracción:

Identificar controladores programáticos y técnicos.

Validar metas y objetivos:

Determinar la compatibilidad de los controladores de extracción con las metas y objetivos.

Identificar componentes candidatos:

Determinar criterios para componentes heredados de gran valor.

Selección del conjunto de componentes candidatos.

Producción de tópicos de extracción:

Revisión de cualquier tópico de extracción e inquietudes que fueron identificados durante la actividad.

Evaluar el estado de preparación:

Determinar los niveles de estado de preparación de la organización.

Revisión del calendario OAR:

Actualizar el calendario OAR si fuera necesario.

Algunas tareas tienen sub-tareas. La tarea cuatro por ejemplo es subdividida en:

Revisión de sistemas heredados.

Revisión de documentación de componentes.

Ejemplo de ejecución y plantillas de datos.

La plantilla de, especifica como ejecutar la revisión de documentación de componentes (sub-tarea 4.2).

La plantilla de datos "EMC-4.2-DT" sugiere que los siguientes tipos de datos deben ser recolectados:

Lista de componentes heredados.

Documentación de componentes (ej. Funcionalidad y descripción de interfaces).

Código fuente de componentes heredados.

Historia de mantenimiento para componentes heredados (ej. Costo, modificación, margen de tiempo, recursos de utilización).

Para ver el gráfico seleccione la opción "Descargar" del menú superior

Validación.

Después de que las tareas fueron completadas, las preguntas fundamentales para la actividad son revisadas. Algunos de los criterios de validación para el ECE son los siguientes:

Las expectaciones son esquematizadas y entendidas.

Todos los sistemas de información heredados son identificados.

Un conjunto de 10 a 20 componentes de línea de producción necesarios con alto potencial han sido seleccionadas para ser satisfechas a través de la extracción.

Controladores y prioridades son identificados y entendidos.

Un conjunto de 15 a 30 componentes heredados han sido seleccionados como candidatos para la extracción.

El nivel de preparación de extracción ha sido evaluado.

Artefactos de salida.

Establecer contextos de extracción produce los siguientes artefactos de salida:

Un conjunto de relevantes sistemas de información heredados y documentación.

Un conjunto de componentes de línea de producción necesarios.

Programas principales y controladores técnicos.

Una conjunto de componentes heredados y documentación de componentes asociados.

Evaluación del estado de preparación.

Lista de tópicos de extracción e inquietudes.

Revisión del perfil del calendario OAR.

Criterios de salida.

Antes de completar el establecimiento del contexto de extracción, es necesario cubrir los siguientes criterios de salida:

El equipo de extracción ha identificado un conjunto razonable de líneas de producción necesarias y componentes candidatos.

La expectación de la organización es consistente con los niveles del estado de preparación.

El perfil del calendario OAR ha sido revisado y cualquier cambio necesario ha sido aceptado.

Todos los artefactos de salida de esta actividad han sido producidos.

Todas las preguntas fundamentales para esta actividad han sido contestadas satisfactoriamente.

Ejemplos de tareas especializadas.

Si los criterios de salida no son cumplidos, puede ser apropiado una tarea especializada. Los siguientes son ejemplos de tareas especializadas para la actividad de establecer del contexto de extracción:

Impulsar los esfuerzos para identificar los controladores de extracción y resolver los problemas programáticos y tópicos técnicos.

Impulsar la definición de los requerimientos de línea de producción y componentes necesarios en términos de funcionalidad e interfaces.

Aislando e identificando la funcionalidad e interfaces de los componentes heredados.

Desarrollando los niveles requeridos de documentación para los componentes heredados.

9.2 El modelo herradura

Los tres procesos básicos – análisis de un sistema existente, transformación lógica y desarrollo de un nuevo sistema – forman la base del modelo de herradura. El primer proceso sube el extremo izquierdo de la herradura, el segundo cruza la parte superior y el tercero baja por el extremo derecho de la herradura. La riqueza del modelo de herradura son los tres niveles de abstracción que pueden ser adoptados para las descripciones lógicas. Conceptualmente, este puede ser a través de un conjunto de herraduras anidadas. Las descripciones lógicas pueden ser artefactos tan concretos y simples como el código fuente del sistema o tan complejos y abstractos como la arquitectura del sistema. El propósito de la metáfora visual es para integrar las vistas de reingeniería a nivel de código y arquitectónico del mundo. [Berg 99]

Para ver el gráfico seleccione la opción "Descargar" del menú superior

En su más pura y completa forma, el primer proceso recupera la arquitectura por medio de la extracción de artefactos desde el código fuente. Esta estructura

recuperada es analizada para determinar si esta se adapta a la arquitectura antes diseñada. La arquitectura descubierta también es evaluada con respecto a un número de calidad de atributos tales como rendimiento, modificabilidad, seguridad o confiabilidad.

El segundo proceso es la transformación de arquitectura. En este caso, la arquitectura antes construida es recuperada y es reingenierada para hacerla una nueva arquitectura deseable. Esta es reevaluada contra las metas de calidad del sistema y sujetas a otras restricciones organizacionales y económicas.

El tercer proceso del modelo de herradura usa el "Architecture-Based Development (ABD)" [Bass99] para ejemplificar la arquitectura deseable. En este proceso, ya empaquetados los tópicos son decididas e interconectadas las estrategias elegidas. Los artefactos a nivel de código del sistema de información heredado son normalmente tapados o reescritos para adaptarlos dentro de la nueva arquitectura.

9.2.1 Los Tres Niveles Del Modelo Herradura

9.2.1.1 En el modelo herradura existen tres niveles:

"Representación de la estructura de código", el cual incluye código fuente y artefactos tales como árboles de sintaxis abstractos (Abstract syntax tree; ASTs) y diagramas de flujo obtenidos a través del análisis gramatical y operaciones analíticas de rutina.

"Representación del nivel funcional", el cual describe la relación entre las funciones del programa (llamadas), datos (funciones y relaciones de datos), y archivos (agrupamiento de funciones y datos).

"Nivel conceptual", el cual representa grupo tanto de funciones y artefactos del nivel de código que son ensamblados dentro de subsistemas de componentes relacionados o conceptos.

El modelo completo no solo hace transformaciones en el nivel de arquitectura, también lo hace en los niveles subsidiarios. A continuación se proporcionan algunos ejemplos simples de transformaciones en cada nivel.

Nivel de código.

En el nivel de código existen dos sub-niveles, transformación textual (o basado en cadena) y transformación basado en el árbol de sintaxis. Mientras algunos métodos hacen distinciones entre transformaciones "1A" textual (sintáctico) y "1B" AST-

based (semántico), el modelo herradura los considera a ambos dentro del contexto de estructura de código.

-Transformación Textual:

En el nivel 1A, las transformaciones textuales son realizadas a través de varias comparaciones de cadenas simples y remplaza métodos. Por ejemplo, el mayor de los esfuerzos de solución al "Year 2000" (Y2K) fueron enfocadas en el código fuente, el cual representaba los años como manipulaciones de dos dígitos (por ejemplo: 99 para 1999), la solución entonces fue remplazarlos con codigo que manipulara cuatro dígitos. Otros ejemplos incluyen transformaciones textuales de nombres o palabras claves cuando portaban un sistema desde una plataforma o sistema operativo a otro. Las transformaciones del nivel 1A son relativamente simples, directas y relativamente baratas. Estas transformaciones son elegidas por las organizaciones cuando el problema es suficientemente simple o cuando se requiere un resultado rápido y sucio (quick and dirty).

-Transformación al árbol de sintaxis:

En el nivel 1B, las transformaciones a la estructura de código basada en árboles de sintaxis (ASTs) soportan cambios que son inmunes a las variaciones superficiales de sintaxis. Así, la representación del código fuente es independiente de las expresiones o la forma en que los lenguajes de programación representan números. Transformaciones a la estructura de código basado en árboles de sintaxis son normalmente usadas para implementar nuevos lenguajes (por ejemplo de COBOL a C++) o para hacer cambios al código automáticamente (por ejemplo, métodos más sofisticados para el problema Y2K).

-Transformaciones a nivel funcional.

Transformaciones a nivel funcional (nivel "2") tiene que ver con el re-empaquetado de funcionalidad (por ejemplo, migrar desde un diseño funcional a un diseño orientado a objeto o migrar desde un modelo de base de datos relacional a un modelo orientado objeto). La encapsulación de un modulo de funcionalidad por un diferente ambiente, es un ejemplo de transformación a nivel funcional. Transformaciones a nivel funcional va más allá de simples transformaciones a la estructura del código, pero no va más allá que una transformación de arquitectura. Ellos son elegidos cuando grandes unidades de funcionalidad pueden ser salvados poniéndolos dentro de un nuevo contexto.

-Transformaciones a nivel de arquitectura.

Las transformaciones a nivel de arquitectura (nivel "3") involucran cambios a los bloques básicos de la arquitectura. Estos incluyen los modelos básicos de interacción incluyendo los tipos de componentes, los conectores usados, la asignación de funcionalidad y el modelo en tiempo de ejecución de control y datos. El nivel de arquitectura es el más abstracto y lejos del alcance de las transformaciones. Las transformaciones a nivel de arquitectura son hechas cuando es necesario un cambio a la estructura principal debido a las principales modificaciones o deficiencias en los sistemas de información heredados. Las transformaciones generalmente traen mayores compromisos de tiempo y recursos, pero también trae consigo grandes beneficios.

-Interacción entre niveles.

Las transformaciones a la estructura del código se pueden dar sin hacer cambios de nivel funcional o cambios a la arquitectura. Las transformaciones del nivel más bajo soportan las transformaciones de niveles más altos. Con esta vista multi-nivel de transformaciones, las transformaciones a la arquitectura son normalmente el contexto en los cuales toman parte niveles mas bajos. Sin embargo, las transformaciones de nivel superior no soportan transformaciones de nivel bajos porque esas transformaciones se pueden dar independientemente de las transformaciones de niveles superiores. De echo, uno de los principales propósitos del modelo herradura es elevar el nivel de abstracción y brindar noticiones de la arquitectura para las tareas de reingeniería.

9.3 El modelo cíclico

Este modelo define [Pres 02] seis actividades. En algunas ocasiones, estas actividades se producen de forma secuencial y lineal, pero esto no siempre es así. Por ejemplo, puede ser que la ingeniería inversa (la comprensión del funcionamiento interno de un programa) tenga que producirse antes de que pueda comenzar la reestructuración de documentos.

El paradigma de la reingeniería mostrado en la figura es un modelo cíclico. Esto significa que cada una de las actividades presentadas como parte del paradigma pueden repetirse en otras ocasiones. Para un ciclo en particular, el proceso puede terminar después de cualquier de estas actividades.

9.3.1 Actividades Del Modelo Cíclico.

En esta sección se dará una breve explicación de las actividades que se definen en el modelo cíclico: Análisis de inventario, Reestructuración de documentos, Ingeniería inversa, Reestructuración de código, Reestructuración de datos e Ingeniería directa.

Análisis de inventario.

Todas las organizaciones de software deberán disponer de un inventario de todas sus aplicaciones. El inventario puede que no sea más que una hoja de calculo con la información que proporciona una descripción detallada (por ejemplo: tamaño, edad, importancia para el negocio) de todas las aplicaciones activas.

Los candidatos a la reingeniería aparecen cuando se ordena esta información en función de su importancia para el negocio, longevidad, mantenibilidad actual y otros criterios localmente importantes. Es entonces cuando es posible asignar recursos a las aplicaciones candidatas para el trabajo de reingeniería.

Es importante destacar que el inventario deberá revisarse con regularidad. El estado de las aplicaciones (por ejemplo, la importancia con respecto al negocio) puede cambiar en función del tiempo y, como resultado, cambiarán también las prioridades para la reingeniería.

9.3.1.1 Reestructuración de documentos.

Una documentación escasa es la marca de muchos sistemas de información heredados. ¿Qué se puede hacer al respecto?

Opción 1: La creación de documentación consume muchísimo tiempo. El sistema funciona, y ya nos ajustaremos con lo que se tiene. En algunos casos, éste es el enfoque correcto. No es posible volver a crear la documentación para cientos de programas de computadoras. Si un programa es relativamente estático está llegando al final de vida útil, y no es probable que experimente muchos cambios: ¡dejémoslo así!.

Opción 2: Es preciso actualizar la documentación, pero se dispone de recursos limitados. Se utilizará un enfoque "del tipo documentar si se modifica". Quizá no es necesario volver a documentar por completo la aplicación. Más bien se documentarán por completo aquellas partes del sistema que estén experimentando cambios en ese momento. La colección de documentos útil y relevante irá evolucionando con el tiempo.

Opción 3: El sistema es fundamental para el negocio, y es preciso volver a documentarlo por completo. En este caso, un enfoque inteligente consiste en reducir la documentación al mínimo necesario.

Todas y cada una de estas opciones son viables. Las organizaciones del software deberán seleccionar aquella que resulte más adecuada para cada caso.

9.3.1.2 Ingeniería inversa.

El término "ingeniería inversa" tiene sus orígines en el mundo del hardware. Una cierta compañía desensambla un producto de hardware competitivo en un esfuerzo por comprender los "secretos" del diseño y fabricación de su competidor. Estos secretos se podrán comprender más fácilmente si se obtuvieran las especificaciones de diseño y fabricación del mismo. Pero estos documentos son privados, y no están disponibles para la compañía que efectúa la ingeniería inversa. En esencia, una ingeniería inversa con éxito precede de una o más especificaciones de diseño y fabricación para el producto, mediante el examen de ejemplos reales de ese producto.

La ingeniería inversa del software es algo bastante similar. Sin embargo, en la mayoría de los casos, el programa del cual hay que hacer una ingeniería inversa no es el de un rival, sino, más bien, el propio trabajo de la compañía (con frecuencia efectuado hace muchos años). Los "secretos" que hay que comprender resultan incomprensibles porque nunca se llegó a desarrollar una especificación. Consiguientemente, la ingeniería inversa del software es el proceso de análisis de un programa con el fin de crear una representación de programa con un nivel de abstracción más elevado que el código fuente. La ingeniería inversa se extraerá del programa existente información del diseño arquitectónico y de proceso, e información de los datos.

9.3.1.3 Reestructuración del código.

El tipo más común de reingeniería es la reestructuración del código. Algunos sistemas heredados tienen una arquitectura de programa relativamente sólida, pero los módulos individuales han sido codificados de una forma que hace difícil comprenderlos, comprobarlos y mantenerlos. En estos casos, se puede reestructurar el código ubicado dentro de los módulos sospechosos.

Para llevar a cabo esta actividad, se analiza el código fuente mediante una herramienta de reestructuración, se indican las violaciones de las estructuras de programación estructurada, y entonces se reestructura el código (esto se puede hacer automáticamente). El código reestructurado resultante se revisa y se comprueba para

asegurar que no se hayan introducido anomalías. Se actualiza la documentación interna del código.

9.3.1.4 Reestructuración de datos.

Un programa que posea una estructura de datos débil será difícil de adaptar y de mejorar. De hecho, para muchas aplicaciones, la arquitectura de datos tiene más que ver con la viabilidad a largo plazo del programa que el propio código fuente.

A diferencia de la reestructuración de código, que se produce en un nivel relativamente bajo de abstracción, la estructuración de datos es una actividad de reingeniería a gran escala. En la mayoria de los casos, la reestructuración de datos comienza por una actividad de ingeniería inversa. La arquitectura de datos actual se analiza minuciosamente y se definen los modelos de datos necesarios. Se identifican los objetos de datos y atributos y, a continuación, se revisan las estructuras de datos a efectos de calidad.

Cuando la estructura de datos es débil (por ejemplo, actualmente se implementan archivos planos, cuando un enfoque relacional simplificaría muchísimo el procesamiento), se aplica una reingeniería a los datos.

Dado que la arquitectura de datos tiene una gran influencia sobre la arquitectura del programa, y también sobre los algoritmos que los pueblan, los cambios en datos darán lugar invariablemente a cambios o bien de arquitectura o bien de código.

9.3.1.5 Ingeniería directa (foward engineering).

En un mundo ideal, las aplicaciones se reconstruyen utilizando un "motor de reingeniería" automatizado. En el motor se insertaría el programa viejo, que lo analizaría, reestructuraría y después regeneraría la forma de exhibir los mejores aspectos de la calidad del software. Después de un espacio de tiempo corto, es probable que llegue a aparecer este "motor", pero los fabricantes de CASE han presentado herramientas que proporcionan un subconjunto limitado de estas capacidades y que se enfrentan con dominios de aplicaciones específicos (por ejemplo, aplicaciones que han sido implementadas empleando un sistema de bases de datos específico). Lo que es más importante, estas herramientas de reingeniería cada vez son más sofisticadas.

La ingeniería directa, que se denomina también renovación o reclamación [Chi 90], no solamente recupera la información de diseño de un software ya existente, sino que, además, utiliza esta información en un esfuerzo por mejorar su calidad global.

En la mayoría de los casos, el software procedente de una reingeniería vuelve a implementar la funcionalidad del sistema existente, y añade además nuevas funciones y/o mejora el rendimiento global.