10. Reconstruccion de la Arquitectura

En este último capítulo se trata el tema de reconstrucción de la arquitectura de un sistema candidato al proceso de reingeniería. La reconstrucción de la arquitectura es una de las principales actividades que se deben realizar al iniciar un proyecto de reingeniería ya que esta nos permite entender al programa que sufrirá la transformación mediante la creación de abstractas del sistema. En el primer subtema se intenta definir cual es el rol de la reconstrucción de la arquitectura en el proceso de reingeniería para después mencionar algunas recomendaciones para un proyecto de reconstrucción de la arquitectura. Este capítulo finaliza explicando a detalle las fases que conforman la actividad de reconstrucción de la arquitectura dando en algunos casos, ejemplos que ayuden a entender dichas fases.

10.1 El rol de la construcción de la arquitectura

La reconstrucción de la arquitectura según Bergey [Berg 00a] resulta en una representación arquitectural que puede:

Ser usada para documentar la arquitectura existente.

Ser usada para checar la conformidad de la ya implementada arquitectura a la arquitectura diseñada.

Servir como un punto de partida para aplicar reingeniería al sistema para diseñar una nueva arquitectura a través de la estrategia de transformación de la arquitectura.

Ser usada para identificar componentes para establecer un método de línea de aplicación.

Si una organización no tiene documentación actualizada para un sistema existente, este es a menudo una posibilidad para reconstruir la arquitectura del sistema para proporcionar documentación actual. Usando apoyo automatizado, las unidades fuentes que construyen componentes arquitectónicos y las ligas entre ellos sirven como las base para la construcción de la documentación.

En muchos casos, la ya implementada arquitectura de un sistema tendrá que derivarse en la arquitectura diseñada. En algunos casos, reconstruyendo la arquitectura del software ayuda en el chequeo de conformidad de la ya implementada arquitectura contra la arquitectura diseñada.

En otros casos, una organización deseará actualizar y agregar funcionalidad al sistema. La arquitectura ya implementada es entonces reconstruida y usada como la base para la transformación para la nueva arquitectura diseñada.

Cuando se introduce un método a la línea de producción, este normalmente se beneficia al usar los componentes de la arquitectura existente en la línea de producción. En estos casos, la reconstrucción de la arquitectura puede ayudar a identificar componentes comunes que pueden ser el centro activo en la nueva línea de producción de la arquitectura.

10.2 Recomendaciones y fases para la reconstrucción de la arquitectura

10.2.1 Recomendaciones Para La Reconstrucción De Arquitectura

Los siguientes, según Kazman [Kazm 97] son recomendaciones generales para el proyecto de reconstrucción de la arquitectura:

Tener una meta y un conjunto de objetivos o preguntas en mente antes de emprender un proyecto de reconstrucción de datos. Por ejemplo, re-usar partes del sistema en una nueva aplicación puede ser una meta. Sin estas metas u objetivos, gran parte del esfuerzo puede ser gastado en la extracción de la información y generar vistas arquitectónicas que pueden no ser útiles.

Obtener una visión de alto nivel de la arquitectura del sistema antes de comenzar el detallado proceso de reconstrucción. Esta visión guía la:

Procesos de extracción para ayudar a identificar la información que se necesita ser extraída del sistema.

Procesos de reconstrucción para ayudar a determinar que se ve en la arquitectura y que vistas se generarán.

Usar la documentación existente para generar solo vistas de alto nivel de los sistemas. En muchos casos, la documentación existente para un sistema puede no reflejar exactamente el sistema como esta implementado, pero este puede dar una indicación de conceptos de alto nivel.

Involucrar a la gente que esta familiarizado con el sistema en el proyecto para obtener un mejor entendimiento del sistema que será reconstruido. Herramientas pueden ayudar al esfuerzo y acorta los procesos de reconstrucción, pero ellos no pueden ejecutar una reconstrucción completa automáticamente. La reconstrucción

de arquitectura requiere que se involucre la gente (arquitectos, desarrolladores y gente de mantenimiento del software) quienes están familiarizados con el sistema.

Asignar a alguien de tiempo completo para trabajar sobre el proyecto de reconstrucción de arquitectura. La reconstrucción de arquitectura involucra un extenso y detallada análisis de un sistema y requiere un significante esfuerzo.

10.2.2 Fases Para La Reconstrucción De La Arquitectura.

La reconstrucción de datos requiere una serie de actividades y técnicas. Con la gran cantidad de software en la mayoría de sistemas, es casi imposible ejecutar todas las actividades de reconstrucción manualmente. En lugar de eso, un conjunto de herramientas (conocidas como workbench) es necesario para apoyar a las actividades de reconstrucción de la arquitectura.

Un workbench para la reconstrucción de arquitectura debe ser abierto (acoplar fácilmente nuevas herramientas) y proporcionar una fácil integración de un marco de trabajo (conocido como framework) para que las nuevas herramientas agregadas al conjunto no afecten a las herramientas existentes o datos. Un workbench es "ARMIN", el cual remplaza al workbench "Dali architecture Reconstruction" [Kazm 97]. ARMIN se utiliza para extraer información desde el código fuente y otros artefactos del sistema, ARMIN habilita la carga de esta información para que pueda ser usada en el proceso de reconstrucción.

Usando las herramientas proporcionadas por ARMIN, la reconstrucción de la arquitectura según Kazman [Kazm 03] comprende las siguientes fases:

Extracción de la información.

Construcción de la base de datos.

Fusión de vistas.

Composición de las vistas arquitectónicas.

10.2.2.1 Extracción de la Información

La extracción de la información involucra el análisis del diseño existente y artefactos de implementación de un sistema para construir un modelo basado en las vistas de las múltiples fuentes. Desde los artefactos fuente (código, archivos cabecera, archivos construidos) y otros artefactos (ejecución del programa renglón por renglón) de los sistemas, los elementos interesantes y las relaciones entre ellos pueden ser identificados y capturados para producir varias vistas fundamentales del

sistema. La siguiente tabla muestra una lista de elementos típicos y varias relaciones entre los elementos que pueden ser extraídos de un sistema.

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

Cada una de las relaciones entre los elementos constituye una vista diferente del sistema. Las relaciones "calls" entre funciones muestran como interactúan varias funciones en el sistema. Las relaciones "includes" entre archivos muestra las dependencias entre los archivos del sistema. Las relaciones "access_read" y "access_write" entre funciones y variables muestra como son usados los datos en el sistema. Ciertas funciones pueden escribir en un conjunto de datos y otros pueden leerlos. Esta información de relaciones es usada para determinar como son pasados los datos entre varias partes del sistema.

Si el sistema analizado es grande y dividido dentro de una estructura de directorio, capturar la estructura del directorio puede ser importante para el proceso de reconstrucción. Ciertos componentes o subsistemas pueden ser almacenados en directorios, y capturar las relaciones tales como "dir_contains_dir" y "dir_contains_dir" puede ayudar a identificar componentes en el proceso de reconstrucción.

El conjunto de elementos y relaciones extraídos dependerá del tipo de sistema analizado y las herramientas de extracción disponibles. Si el sistema a reconstruir es orientado a objetos, clases y métodos deben ser agregados a la lista de elementos extraídos, y relaciones tales como "Class is_subclass Class" y "Class contains Method" deben ser extraídas y usadas en el proceso de reconstrucción.

Las vistas extraídas pueden ser catalogadas en estáticas o dinámicas. Las vistas estáticas son aquellas obtenidas por observación de los artefactos del sistema, mientras que las vistas dinámicas son las obtenidas por la observación del sistema durante su ejecución. En muchos casos, las vistas estáticas y dinámicas pueden ser combinadas para crear una representación más completa y exacta del sistema. Si la arquitectura del sistema cambia en tiempo de ejecución, por ejemplo, un archivo de configuración es leído por el sistema, y ciertos componentes son cargados en tiempo de ejecución, la configuración en tiempo de ejecución deber ser capturado y usado cuando se lleve a cabo la reconstrucción.

Una vista fuente puede ser extraída aplicando cualquiera de las herramientas disponibles, apropiadas o necesarias para un sistema objetivo dado. El tipo de

herramientas que se usan regularmente en la extracción de información son las siguientes [Kazm 03]:

Parsers (por ejemplo, "Understand for C/C++/java", "Imagix", "SNiFF+", "C++ Information Abstractor [CIA]", "Rigiparse").

Analizadores basado en árboles de sintaxis abstractos (AST) (ejemplo, "Gen++", "Refine").

Analizadores léxicos (por ejemplo, "Lightweight Source Model Extractor [LSME]").

Perfiladores.

Instrumentación de código.

Ad Hoc (por ejemplo, "Grez", "Perl")

Estas herramientas son aplicadas a las líneas del código fuente. Parsers analizan el código y generan representaciones internas del código. Típicamente, es posible salvar esta representación interna para obtener una vista fuente. Los analizadores basados en AST hacen un trabajo similar, pero ellos construyen una representación del árbol explicito de la información analizada.

Analizadores léxicos examinan los artefactos fuentes como cadenas de elementos léxicos o señales. El usuario del analizador léxico puede especificar un conjunto de patrones léxicos que coincidan y los elementos que regresará. Similarmente se usan una colección de herramientas ad hoc tales como Grez y Perl para llevar a cabo comparaciones y búsqueda de patrones léxicos dentro del código en orden de alguna información de salida requerida.

Todas estas herramientas – parsers, analizadores basados en AST, analizadores léxicos y ad Hoc – son usadas para extraer información estática.

Las herramientas perfiladoras y de código son usadas para extraer información acerca del código que esta siendo ejecutado. Usar estas herramientas generalmente no requiere la agregación de cualquier código nuevo al sistema. Por otro lado, instrumentación de código – tales como los aplicados en el campo del testeo – involucra agregar código al sistema para hacer resaltar alguna información especifica (por ejemplo, que procesos se conectan con otros) mientras el sistema se ejecuta [McCa 02]. Estas herramientas y técnicas generan vistas dinámicas del sistema.

10.2.2.2 Construcción de la base de datos.

El conjunto de vistas extraídas son convertidas al formato "Rigi Standard Format" (RSF) o al "Graph eXchange Language" (GXL) y cargadas en ARMIN durante la fase de construcción de bases de datos. Esta conversión es hecha usando Perl scripts que leen los datos y los convierten en un archivo RSF. Las vistas extraídas pueden estar en diferentes formatos dependiendo de las herramientas usadas para extraerlas. Por ejemplo, una herramientas de extracción tal como "Understand for C/C++/Java" o "Imagix-4D", pueden ser usadas para cargar el código fuente dentro de una representación interna, y esta información puede entonces ser descargada a un conjunto de archivos bandera indexados por archivos o función. Estos archivos tienen una estructura uniforme, y los scripts pueden ser desarrollados en Perl para leer esos archivos y extraer información acerca de los elementos y relaciones. La siguiente figura describe este proceso.

Una vez que los elementos y relaciones de archivos (la vista extraída) es convertida a RSF o a GXL, estos pueden ser cargados en ARMIN. La figura 4.2 muestra un extracto de un archivo RSF ejemplo. El archivo completo puede ser cargado dentro de una base de datos en ARMIN.

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

10.2.2.3 Fusión de vistas.

En la fase Fusión de vistas, las vistas extraídas son manipuladas para crear vistas fusionadas. Por ejemplo, una vista estática puede ser fusionada con una vista dinámica. Como se puede notar, una vista estática no puede proporcionar toda la información relevante de la arquitectura. En algunos casos, algunas funciones solo pueden ser identificables en tiempo de ejecución, entonces se necesitará generar una vista dinámica. Estas dos vistas necesitan ser compaginadas y fusionadas para producir la gráfica completa del sistema.

La fase de fusión de vistas compagina y establece conexiones entre las vistas que proporcionan información complementaria. La fusión es ilustrada usando los ejemplos de las siguientes sub-secciones. El primer ejemplo muestra el mejoramiento de una vista estática de un sistema orientado a objetos agregándole información dinámica. El segundo muestra la fusión de varias vistas para identificar funciones llamadas en un sistema.

Mejorando una vista.

Consideramos las dos vistas de código, las cuales son del conjunto de métodos extraídos desde un sistema implementado en C++.

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

Las diferencias entre estas vistas se muestran en:

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

La vista dinámica muestra que List:: getnth es llamada. Sin embargo, este método no esta incluido en el análisis de la vista estática porque este no fue identificado por la herramienta de extracción estática. Esto muestra que la herramienta de extracción estática no es perfecta, haciendo necesario validar los resultados de la información extraída. También, las llamadas a los métodos constructores y destructores de InputValue y List no están incluidas en la vista estática. Esa ausencia de métodos debe ser agregada a la vista de arquitectura compaginada.

En adición, la extracción estática muestra que la clase Primitive Op tiene una llamada al método Compute. La extracción dinámica no muestra tal clase, pero muestra clases tales como ArithmeticOp, AttachOp y StringOp, cada una de las cuales tiene un método Compute y es de echo una subclase de PrimitiveOp, PrimitiveOp es una superclase; este nunca es llamada en la ejecución del programa. Pero este es la llamada a PrimitiveOp que se muestra cuando se ejecuta el extractor estático. Para tener una vista exacta de la arquitectura, las vistas estáticas y dinámicas de PrimitiveOp deben ser compaginadas.

La siguiente figura muestra los ítems que deben ser agregados a la vista fusionada y aquellos que deben ser removidos desde la vista fusionada.

Despejando la ambigüedad de las funciones llamadas.

En una aplicación multi procesos, es muy probable que ocurra choque de nombres. Por ejemplo, varios de los procesos pueden tener un procedimiento llamada main al cual pueden llamar. Es importante identificar y eliminar la ambigüedad de esas colisiones de nombres dentro de las vistas extraídas. Otra vez, por medio de la fusión de información que puede ser extraída fácilmente, podremos remover las ambigüedades potenciales. En este caso, necesitamos fusionar las vistas estáticas con una vista que contenga archivos/funciones (para determinar cuales funciones están definidos en cuales archivos) y con una vista de dependencias (para determinar que archivos están compilados junto a los ejecutables producidos). La fusión de estas

tres fuentes de información hace a los nombres de procedimientos, métodos y otros elementos nombrados, permitiendo entonces ser referidos sin ambigüedad en el proceso de reconstrucción de la arquitectura. Sin la fusión de vistas, la colisión de nombres puede persistir, y los resultados de la reconstrucción pueden ser ambiguos.

10.2.2.4 Composición de vistas arquitectónicas.

La fase de composición de vistas arquitectónicas consiste en dos áreas de actividad principales:

Visualización e interacción.

Definición de scripts de comandos e interpretación.

Las áreas de visualización e interacción proporcionan un mecanismo que permite al usuario visualizar, explorar y manipular vistas interactivamente. El componente "Aggregator" de ARMIN es usado para presentar vistas al usuario como una gráfica de descomposición jerárquica [Wong 94]. Una presentación ejemplo de una vista arquitectónica. Usando el Aggregator, el usuario puede ver vistas en una variedad de estilos de diseños incluyendo jerárquicos, espiral y ortogonal.

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

La definición de scripts de comandos y la interpretación proporcionan facilidades para la abstracción de información de bajo nivel para generar vistas arquitectónicas. Los scripts de comandos facilitan al usuario para escribir scripts para construir mejores vistas abstractas desde vistas más detalladas por medio de la identificación de agregación de elementos. Los scripts ARMIN son escritos usando el ARL y cualquier editor, y son cargados dentro de ARMIN.

La reconstrucción arquitectónica no es un proceso directo. La construcción arquitectónica no es representada explícitamente en el código fuente, lo que hace que la reconstrucción sea especialmente difícil. Adicionalmente, la construcción arquitectónica es realizada por una diversidad de mecanismos en una implementación. Usualmente estos son colecciones de funciones, clases, archivos, objetos y demás. Cuando un sistema es inicialmente desarrollado, los elementos de la arquitectura/diseño de alto nivel son mapeados para la implementación de elementos. Por consiguiente, cuando los elementos arquitectónicos son "reconstruidos", se aplica un mapeo a la inversa.

La reconstrucción arquitectónica es un proceso interpretativo, interactivo e iterativo, no un proceso automático. Este requiere la habilidad y atención tanto de expertos en

ingeniería inversa y arquitectos (o algunos de los que tienen un conocimiento substancial de la arquitectura). Basándose en los parámetros arquitectónicos que los expertos en la arquitectura pueden encontrar en el sistema, la ingeniería inversa puede construir varios scripts de comandos usando ARMIN. Estos scripts resultan en una nueva agregación que muestra varias abstracciones o agrupamientos de elementos de bajo nivel (los cuales pueden ser artefactos fuentes o abstracciones). Por la interpretación de estas vistas y el análisis de estos, es posible refinar los scripts y agregaciones para producir varias hipótesis de vistas arquitectónicas del sistema. Estas vistas pueden ser interpretadas, refinadas o rechazadas. No existe un criterio universal para este proceso; este está completo cuando la representación arquitectónica es suficiente para apoyar el análisis necesario para los usuarios, entonces las metas de la reconstrucción pueden ser logradas.

Consideremos el subconjunto de elementos y relaciones mostradas en la siguiente tabla:

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

En este ejemplo, las variables "a" y "b" están definidas en la función "f"; esto es, ellas son locales a "f". Esta información la podemos representar gráficamente .

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

Las variables locales no importan durante una reconstrucción de arquitectura porque ellas proporcionan poca comprensión de la arquitectura del sistema. Por consiguiente, instancias de variables locales pueden ser agregadas a la función en la cual ocurren. Un script tal como el que se muestra a continuación se puede escribir para ese propósito.

La primer línea es un comentario el cual comienza con el signo de libra #. La segunda línea obtiene los descendientes de una función (usando el comando desc) – en este caso, variables locales. El comando desc regresa un arreglo tridimensional de la función y sus variables locales. La tercer línea asocia el nombre de la función con las variables locales para cada función y agrega un signo más (+) al final del nombre. Usando el comando collapse, la cuarta línea borrar de la gráfica todos los nombres de variables locales y deja solo el nombre de la función con el +. Esta nueva gráfica es llamada FUNCTION+. La última línea despliega la gráfica en la ventana de Aggregator usando el comando show.

El resultado de aplicar el script es representado gráficamente en la siguiente figura. La mayoría de los scripts de comandos en ARMIN están desarrollados de una manera similar.

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

El principal mecanismo para manipular las vistas es la aplicación de scripts de comandos. La siguiente lista muestra algunos ejemplos de scripts:

Identificar tipos.

Agregar variables locales con funciones.

Agregar miembros a las clases.

Crear elementos del nivel de arquitectura.

Un ejemplo de un script que identifica un componente del nivel de arquitectura, Logical_Interaction, es mostrada en la siguiente figura. Este script dice que si el nombre de la clase es Presentation, Bspline, o Color, o si la clase es una subclase de Presentation, este pertenece al componente Logical_Interaction.

El script esta escrito de esta forma para abstraer información desde la información de bajo nivel para generar vistas a nivel de arquitectura. El reconstructor crea este script para probar la hipótesis acerca del sistema. Si un script no produce los resultados esperados, este puede ser descartado. El reconstructor iteractúa a través de este proceso hasta que obtiene las vistas arquitectónicas útiles.

Conclusiones

El software siempre ha sido y será lo que le puede dar vida y plusvalía a la computación; sin el software, las computadoras serían inservibles y no podrían ser utilizadas para ningún beneficio. Es el software el que realiza los procesos necesarios a los datos introducidos para así obtener nuevos datos con los que se pueden tomar decisiones, en muchos de los casos, decisiones críticas. Conforme pasa el tiempo, hemos visto que las computadoras son utilizadas en casi todos los aspectos de la vida del ser humano, hoy en día se ven computadoras en los bancos, tiendas, automóviles, oficinas gubernamentales, empresas privadas, el hogar y en un futuro cercano hasta en el propio cuerpo humano. Esta simbiosis que el ser humano ha realizado con la computadora es debida en gran parte a que el software, aprovechando la velocidad con que las computadoras pueden procesar datos y arrojar resultados, facilita y agiliza las tareas del hombre.

En la actualidad se puede observar que una falla en el software, en el mejor de los casos puede dar como resultados la pérdida de tiempo pero en algunos otros casos también puede resultar en perdidas de vidas humanas. Conforme pase el tiempo, el hombre es más y más dependiente de las computadoras y sus beneficios, pero siempre existirá la incertidumbre y la intranquilidad de que el software puede fallar y causar serios problemas ya que el software es desarrollado por seres humanos que por naturaleza cometen errores.

El desarrollo de software siempre ha estado en un proceso continuo de evolución, y como en cualquier actividad evolutiva, los primeros años siempre van acompañados de grandes dificultades debidas en gran parte a que no se cuenta con las herramientas ni los conocimientos para realizar bien las tareas que involucra esta actividad. Esto es muy palpable en el ser humano, nadie va a llegar a este mundo sabiendo caminar, este es un proceso en el que se tienen que sufrir tropiezos y caídas, pero que al paso de los años se logra dominar. En los primeros años del desarrollo del software, este fue hecho sin tener absolutamente ningún tipo de conocimiento ni estrategia que permitiera el buen desarrollo, estos años son conocidos como la "crisis de software". Los sistemas que fueron desarrollados en estos años se fueron convirtiendo en parte vital de muchas organizaciones, por lo que se veía la constante necesidad de corregirlos, estas correcciones seguían ejecutándose de una manera no muy convenientes, generando así nuevos errores o poco a poco haciendo imposible la corrección de los sistemas; estos sistemas han recibido el nombre de "sistemas de información heredados".

Al hablar de sistemas de información heredados puede hacerse una analogía con la obra de uno de los más grandes autores de la literatura latinoamericana, Gabriel García Márquez y su obra "Cien años de soledad". El relato adopta una apariencia virtualmente lineal pero en realidad el tiempo de la novela no es sucesivo o cronológico, sino cerrado. El presente, el pasado y el futuro pueden ser narrados en un tiempo a cualquier tiempo por el narrador. Por eso, el tiempo en Cien años de soledad es circular. La novela tiene una declaración que se desarrolla y explica de una manera lógica, que ninguna otra explicación puede ser posible.

El pasado se repite en el presente y el futuro es previsible porque, de alguna manera, ya ocurrió. El tiempo no existe en Macondo (lugar donde se desarrolla la mayor parte de la obra), está congelado. Ursula (uno de los personajes principales) es el que tiene la mas clara conciencia de vivir en una dimensión intemporal, propia de los sueños: cuando José Arcadio Segundo concibe el loco proyecto de establecer un sistema de

navegación, el comentario de Ursula es " ya esto me lo se de memoria". Es como si el tiempo diera vueltas en redondo y hubiéramos vuelto al principio (como la historia de la humanidad, quien comete los mismos errores una y otra vez). La acción concentra la espesa historia de Macondo en un tiempo inmóvil, donde mil cosas pasan y mil cosas vuelven, y sostiene la presencia de varios protagonistas, que se alternan en el primer plano y el trasfondo temporal, sin perder en ningún momento la tensión narrativa.

Así como en Cien años de Soledad, el mantenimiento de algunos sistemas ha seguido una línea en el tiempo circular, en la cual se siguen cometiendo los mismos errores que se cometieron al crear el sistema, heredando así año con años los errores cometidos. Es en este momento, en el cual el mantenimiento de un sistema se hace imposible donde la "Reingeniería de software" se vislumbra como única solución a los sistemas de información heredados. La reingeniería de software pretende terminar de una vez por todas con la vida útil del sistema, no sin antes extraer de el la mayor parte del conocimiento que pueda brindar, conocimientos que serán utilizados para crear un nuevo sistema aplicando las buenas practicas que nos brinda la ingeniería de software.

La reingeniería no es una actividad que se lleva a cabo de la noche a la mañana, además implica una gran inversión de esfuerzo, tiempo y recursos. Es por ello que es necesario planear de una manera muy cuidadosa todo el proceso. Este proceso se inicia con la justificación del proyecto de reingeniería, en el cual se tiene que analizar cada sistema candidato para así obtener una lista de aplicaciones ordenada según sus prioridades. Una vez obtenida la lista, se hace la estimación de costes que serán necesarios para modificar todos los componentes. Por último se comparan los costes con los beneficios esperados.

En la reingeniería de software existen métodos y modelos que permiten que esta actividad se pueda desarrollar de una manera bien direccionada para poder crear una nueva aplicación con un alto nivel de calidad, fiabilidad y plusvalía. En estos métodos y modelos se puede observar varios puntos en común siendo uno de los más importantes la "reconstrucción de la arquitectura", ya que esta es la que nos va a brindar la compresión del sistema al que se le va aplicar reingeniería para poder crear una clara imagen de lo que se va a rediseñar y así poder planificar las actividades que se llevaran a cabo durante toda la actividad de reingeniería.

Ninguna de las actividades llevadas a cabo durante la reingeniería de un sistema es fácil, es por ellos que cada una debe de estar muy bien planeadas antes de

emprenderse y la reconstrucción de la arquitectura no es la excepción. Es por ello que para realizar la reconstrucción de la arquitectura de un sistema de información heredado se recomienda tener bien definidas las metas y objetivos, obtener una visión general de la arquitectura, identificar y usar la documentación existente, e involucrar a personas familiarizadas con el sistema.

La reconstrucción de la arquitectura genera una representación de la arquitectura del sistema que puede ser usado de diversas formas. El principal uso de esta representación es para documentar la arquitectura de un sistema. Si la documentación existente o disponible es obsoleta, la recuperación de la arquitectura puede ser utilizada como base para la redocumentación de la arquitectura. La representación de la arquitectura también puede ser usada como punto de partida para la reingeniería del sistema. Por último, esta representación puede ser útil para identificar la funcionalidad de los componentes para reutilizarlos o establecerlos como la arquitectura base.