Magíster en Ingeniería de Software

"Un modelo del proceso de desarrollo de software guiado por la traceability"

Alumno: Ing. Vanzetti, Juan José

Director: Lic. Oliveros, Alejandro

Tesis presentada a la Facultad de Informática de la Universidad Nacional de La Plata como parte de los requisitos para la obtención del título Magíster en Ingeniería de Software.

La Plata, Septiembre de 2006

Facultad de Informática

UNIVERSIDAD NACIONAL DE LA PLATA ARGENTINA

A todas las personas que de un modo u otro contribuyen para que cada día sea una mejor persona.

Resumen

En esta tesis se presenta la especificación de un proceso de desarrollo de software definido a partir de lo establecido en un modelo de RT. Se ha tomando como referencia el esquema de traceability para los usuarios High End definido por Ramesh y Pohl. El proceso de desarrollo consta de cuatro actividades principales **Modelizar la Organización, Especificar los Requerimientos de Software, Validar la Especificación de Requerimientos de Software y Desarrollar la Arquitectura del Sistema**. Además se especificó la actividad de soporte **Fundamentar**. Es importante recalcar que quedan excluidas de la especificación del proceso de desarrollo de software tanto las actividades involucradas en la confección del diseño detallado, como así también las concernientes a la codificación del sistema

ÍNDICE

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Requirements traceability ha atraído una considerable atención en el campo de la Ingeniería Software y muy especialmente por parte de la comunidad de Ingeniería de Requerimientos. Ello se evidencia a través de numerosos trabajos presentados en publicaciones científicas, congresos y tesis de doctorados así también por la presencia en diversos estándares como IEEE - 830, Mil STD – 2167 y CMM/CMMI, por nombrar sólo algunos.

La implementación de RT en los procesos de desarrollo de software responde a varias razones, en [Kenny, 1996] se agrupan dichas motivaciones en las siguientes cinco categorías:

- Las relacionadas con la calidad: en esta categoría se señala como repercute la implementación de la RT en los componentes de calidad del sistema.
- Las relacionadas con la satisfacción del cliente: la RT representa uno de los mecanismos mediante el cual se permite demostrar o verificar como los requerimientos son validados.
- Las relacionadas con la recuperación de información: la RT puede ayudar a dicha recuperación por medio de la información almacenada en sus traces.
- Las que indican el cumplimiento de estándares, establecidos por diferentes organizaciones.
- Las relacionadas con el mantenimiento, las cuales permiten facilitar el control de cambios.

Si bien hay un importante consenso sobre la RT como medio necesario para alcanzar las mejoras de calidad en el proceso de software, al mismo tiempo existen discrepancias sobre qué es realmente la RT, lo cual implica una amplia variedad en los usos y prácticas actuales de la misma [Pearson, 1996] [Ramesh *et al*., 1995] [Gotel, 1995].

La diversidad de conceptos, técnicas, motivaciones y modelos acerca de la RT, plantea la pregunta de cuál es la función efectiva que cumple en el proceso de desarrollo de software. El autor se sintió motivado a tratar dicho tema debido a que en una clase de Ingeniería de Requerimientos de la Maestría en Ingeniería del Software, dictada por la Universidad de la Plata, se expusieron aspectos relacionados a la RT.

La implementación de RT en los procesos de desarrollo ya ha sido analizada por diferentes autores tales como: Gotel, Pohl, Jarke, Ramesh, Wright, entre otros. Gotel

afirma que la *"RT es crucial para conducir el proceso de RE en sí mismo".*[1] Esta visión se relaciona directamente con el concepto de Pohl: la *"pre-traceability de requerimientos se establece si el proceso de ingeniería de requerimientos en sí mismo es traceable".*[2]

Tomando como punto de partida esas visiones, en esta tesis se propone demostrar que ***"la RT debe conducir el proceso de desarrollo de software",*** extendiendo las ideas de los autores citados anteriormente para afirmar que abarcan la totalidad del proceso de desarrollo de software debido a que la misma se *"utiliza para capturar las relaciones entre los requerimientos, el diseño y la implementación del sistema"*[3]*.*

Para alcanzar ese objetivo se formuló un plan de trabajo con las siguientes pautas básicas:

- Describir las funciones que cumple la RT en el proceso de desarrollo de software.
- Analizar diferentes procesos de desarrollo de software que incorporan RT.
- Seleccionar un modelo de proceso de desarrollo de software.
- Adecuar un modelo con el propósito de especificar un proceso de desarrollo dirigido por traceability.
- Desarrollar una prueba de concepto.

Organización de la Tesis

La presente tesis está organizada de la siguiente forma:

Capítulo 1 Software y Proceso de desarrollo. Se analiza el significado, características, relaciones y diferentes puntos de vistas acerca del software. Luego se exponen temas relacionados con los procesos de desarrollo: definición, características principales y su relación con el software. Se hará además hincapié en la problemática de los procesos de desarrollos para proyectos pequeños.

Capítulo 2 Gestión de los Requerimientos de Software. En este capítulo se tratan los temas relacionados con los la identificación, caracterización y documentación de los requerimientos de software.

Capítulo 3 Traceability de Requerimientos – RT -. Se analizan los vínculos de la RT con las distintas áreas del desarrollo del software.

Capítulo 4 Análisis de los modelos de RT. Se exponen las diferentes técnicas, herramientas, lenguajes y modelos (ESE, PRISM, Estructura de Contribución, etc.) que implementan a la RT.

---

[1] Gotel, Ornela C.Z .Contribution Structures for Requirements Traceability. Ph.D. Thesis. Imperial College of Science, Technology and Medicine. University of London. August 1995. Pág 39.
[2] Pohl, Klaus. PRO ART: Enabling Requirements Pre-Traceability . Pág 3.
[3]Ramesh, B. Stubbs, C. Powers, T. Edwards, M. Lessons Learned from Implementing Requirements Traceability Abril 1995 Pág 1.

Capítulo 5 Modelo de Ramesh. Se describe con mayor detalle el modelo propuesto por Ramesh.

Capítulo 6 Especificación del proceso de desarrollo. En este capítulo se detalla el proceso de desarrollo de software siguiendo los lineamientos establecidos en [Ramesh *et. al*, 2001].

Capítulo 7 Caso de estudio. Se presentan los resultados obtenidos una vez finalizada la puesta en marcha del caso de estudio.

Capítulo 8 Conclusiones y trabajos futuros. En este capítulo se presentan las conclusiones del trabajo realizado. Como así también se detallan los aportes, limitaciones y posibles extensiones del trabajo realizado.

Finalmente en los anexos se detallan todas aquellas cuestiones que se han tratado y facilita el entendimiento de este trabajo.

CAPÍTULO 1 SOFTWARE Y PROCESO DE DESARROLLO

La tarea de definir al proceso de desarrollo de software es una labor bastante compleja. Esto se debe a que existen diferentes puntos de vista y concepciones. Esta problemática puede localizarse en las múltiples concepciones que se poseen acerca del software. Es por ello, que para definirla se examinará el significado y las propiedades distintivas del **software**. Posteriormente, se analizará el concepto de **proceso de desarrollo de software**, sus principales características y actividades. Al finalizar este capítulo se realizará un breve análisis acerca de la problemática en el desarrollo de proyectos catalogados como pequeños.

1.1 El Software

1.1.1 Concepto de software

El objetivo principal de cualquier ingeniería es la construcción de un producto. Entonces cabría preguntarse, ¿cuál es el producto a construir por un ingeniero de sistemas?. El **software** es la respuesta. Éste presenta la propiedad de ser intangible [Ghezzi *et al,* 1991] [McConnell, 2001], pero no obstante, al cumplir una función, se lo cataloga como producto [Ghezzi *et al.*, 1991].

En este apartado se tratará de responder a la pregunta ¿qué es el software?. A primera instancia parece que el interrogante es muy simple. En [Freeman, 1987], [McConnell, 2001] se argumenta que explorar sobre la naturaleza del software es de vital importancia para el buen desempeño de un profesional de la informática. Cabe destacar que los autores que se plantean esta pregunta, también señalan que no es tan simple encontrar la respuesta correcta. Uno de los aspectos en donde puede radicar su complejidad, es que a menudo no se posee el suficiente conocimiento de lo que el software le permitirá manipular [Freeman, 1987].

En el diccionario bilingüe Longman se define el término software como el conjunto de programas que controlan la operación de una computadora.

Pressman define al software como "*un conjunto de instrucciones (programas de computadoras) que cuando se ejecutan proporcionan la función y el rendimiento deseado*,"[4] y también señala que son *"estructuras de datos que permiten a los programas manipular adecuadamente la información y documentos que describen la operación y el uso de los programas".*[5]

[4] Pressman, Roger S. Ingeniería del Software. Un enfoque práctico. Cuarta Edición. McGraw Hill.España. 1998. Pág 7.
[5] Pressman, Roger S. Op.Cit. 1998. Pág 7.

Freeman caracteriza al software como *"el alma y cerebro de la computadora, la corporización de las funciones de un sistema, el conocimiento capturado acerca de un área de aplicación, la colección de los programas, y los datos necesarios para convertir a una computadora en una máquina de propósito especial diseñada para una aplicación particular, y la información producida durante el desarrollo de software"*.[6]

En síntesis, el **software** no sólo hace referencia a los programas ejecutables sino también representa diversas cosas, las cuales difieren en el tiempo, con las personas y principalmente en cómo se lo va a emplear [Freeman, 1987].

1.1.2 Características del software

[Basili, 1989], insta a definir las características del producto software y del proceso que lo elabora, para que su desarrollo sea conducido y concluido con éxito.

Pese a esta disparidad de opiniones referidas a las características del software, diferentes autores, [Ghezzi *et al*., 1991], [Gibbs, 1994] [McConnell, 2001] coinciden que la **maleabilidad** - capacidad de modificación indefinida, constituye su característica distintiva. Esta facultad, la cual se asocia con la aparente facilidad para la adaptación al cambio, produce numerosos inconvenientes. Esto se debe, a que durante el proceso de cambio, no se recuerda que al software hay que pensarlo y gestionarlo **"como un sistema"** [Freeman, 1987]. Teniendo presente este aspecto, se puede afirmar que el cambio es posible e inevitable. Pero también se advierte que al percibir al software como sistema, la gestión del cambio presenta retos de mayor complejidad. El mapear los cambios producidos en los distintos elementos o ítems que existen en el ambiente de desarrollo, representa uno de los principales problemas en el campo de la **Ingeniería del Software** [Madhavji, 1992]. La gestión de los cambios producidos debe contemplar las siguientes actividades [Madhavji, 1992]:

- Identificar las necesidades que produce el cambio.
- Adquirir el adecuado conocimiento referido al cambio producido y el almacenamiento de este conocimiento.
- Evaluar el impacto de los cambios producidos.

Uno de los principales motivos que provocan el cambio en el software se debe a que éste se desarrolla para que refleje o simule lo que acontece en el mundo real, en un dominio específico, el cual puede presentarse como un problema a solucionar. El dominio de problema o la porción de la realidad están en un continuo devenir. El software, para ser fiel

[6] Freeman, Peter.Software Perspectives -The System in the Message-. Addison Wesley. Reading Mass.1987 Pág 5.

reflejo de esta realidad, también debe evolucionar [Ghezzi *et al*., 1991]. El modelado de la realidad, como así también la información inherente a ella se expresa de diversas maneras.

[Jarke *et al.*, 1993] caracteriza este tipo de información en cuatro mundos, los cuales se asocian con diferentes grupos de stakeholders[7]. El primero de ellos se denomina **mundo de la empresa,** en el cual se representa la información del dominio del problema y el comportamiento de la empresa. El segundo es el **mundo del uso,** en el cual se hace hincapié en la interacción hombre – máquina. El tercero es el **mundo del sistema,** en el cual se gestionan los detalles técnicos de desarrollo del software, como ser la especificación de requerimientos, diseño y la implementación de los mismos. Por último, en el **mundo de desarrollo** se detalla la información producida por el proceso de desarrollo del software, los métodos y herramientas que se utilizan, etc.

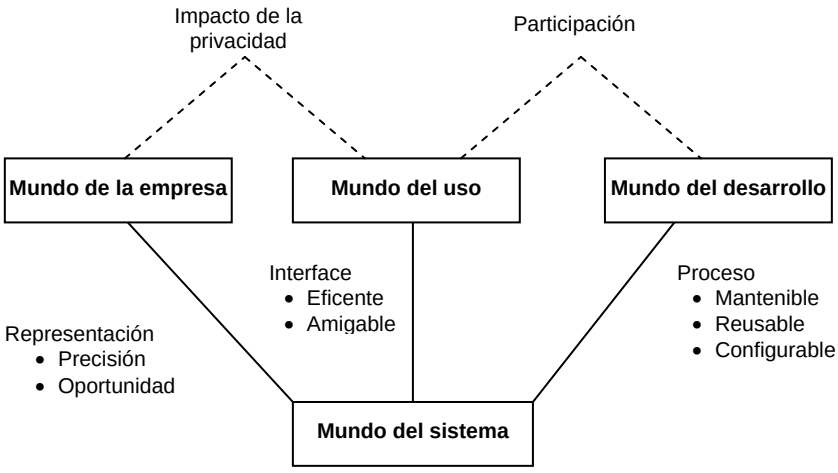


Figura 1.1 Los mundos del modelado del sistema de información [Jarke *et al*., 1993]

Continuando con la descripción de las características distintivas del software, se advierte que la mayoría de los autores también coinciden que la creación del mismo se caracteriza por ser **intensivamente humana.** El proceso de producción de software se ocupa principalmente del diseño e implementación del producto en vez de su fabricación [Ghezzi *et al.*, 1991] [Basili, 1989] [Pressman, 1998].

Se puede analizar al software desde diferentes puntos de vista y desde allí establecer sus propiedades. En [Brooks, 1987] se agrupan las características del software en **esenciales**, las cuales representan las dificultades de la naturaleza del software- y en **accidentales**, las cuales representan las dificultades relacionadas con su producción. Para [Brooks, 1987] **la complejidad, conformidad, el cambio y la invisibilidad** son características esenciales.

Como se verá en los próximos capítulos, la implementación de la RT se vincula con la mayoría de las características esenciales propuestas por Brooks, lo cual no significa que la RT sea considerada como una "*bala de plata"*.[8]

[7] Stakeholders son todas aquellas personas que se benefician o perjudican con el éxito o fracaso del proyecto.
[8] Expresión empleada por Brooks, Frederick en *No Silver Bullits* para argumentar que no existen elementos que atacan a los problemas de las características esenciales del software.

1.1.3 El software y su relación con el conocimiento.

El software refleja las estructuras de conocimiento imbuidas del mundo real o dominio de problema que representa [Sommerville, 1995] [Freeman, 1987]. En este contexto, *"es importante reconocer que la característica principal del producto de software a saber es que es esencialmente conocimiento empaquetado"*[9].

En [Freeman, 1987] se definen tres clases de información que involucran los diferentes aspectos relacionados con el conocimiento y el software. La primera de estas categorías **representación del software,** incluye a los programas, diseños detallados, representaciones de diseño, especificaciones de procesos y requerimientos, entre otras. El principal inconveniente con este tipo de información es la consistencia de los productos. Esta clase de información hace referencia al **software como producto**. Otra clase de información establecida por Freeman es el **conocimiento de la Ingeniería de Software.** Esta categoría se refiere a toda la información relacionada con el desarrollo de software en general (como usar un método determinado) o la información relacionada con un desarrollo específico (cronograma para las pruebas). Por último, la clase de información **dominio del conocimiento específico,** incluye al entendimiento de un domino determinado a ser desarrollado. Un especialista en el área de la aplicación, como ser el ingeniero de control de procesos, un contador entre otros son los encargados de descubrir y plasmar este tipo de información.

1.2 Proceso de Desarrollo de Software

1.2.1 Definición de proceso

En un sentido muy general, "*un proceso define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo"*[10], constituyendo así un marco de trabajo de las tareas a realizar [Pressman, 1998]. En términos concretos, *"los procesos son unas series de pasos que involucran actividades, restricciones y recursos que producen una determinada salida esperada"*[11]. Cabe destacar que estas actividades no se seleccionan aleatoriamente, sino que las mismas se organizan y relacionan con la salida a producir (representa a los objetivos por cumplimentar) [Pfleeger, 2002]. La salida a producir puede ser vista como el valor agregado que genera el proceso.

De lo expuesto, se concluye que la importancia de implementar procesos radica en que los mismos imponen consistencia y estructuras sobre el conjunto de las actividades,

[9] Freeman, Peter. Op.Cit.1987:26.
[10] Jacobson, Ivar. Booch, Grady. Rumbaugh James. El proceso unificado de desarrollo de software. Addison Wesley. España .2000. Pág XVI.
[11] Pfleeger, Shari. Lawrence. Ingeniería de software teoría y práctica. Prentice Hall. 1ra edición Buenos Aires 2002. Pág 52.

permitiendo así capturar experiencias, examinar, comprender, controlar y mejorar las actividades que éste abarca [Pfleeger, 2002].

1.2.2 El proceso de desarrollo

El crear software involucra una serie de pasos y actividades que se especifican en un proceso cuya finalidad es su desarrollo y mantenimiento. Las diversas actividades, métodos y prácticas que se especifican en el proceso, deben estar asociadas a la producción de software [Sommerville, 1995]. Pero debido a la característica distintiva del software, **la maleabilidad,** también se deben detallar en la especificación del proceso de desarrollo de software, las estrategias que acompañan la evolución de éste (gestionar el cambio). Cabe preguntarse cuáles son las tareas o actividades que integran a los procesos de desarrollo de software.[Sommerville, 1995] especifica cuatro actividades que son comunes a todos los procesos de desarrollo:

- **Especificación del Software.** Se engloban todas las actividades orientadas a definir la funcionalidad y las restricciones del software para su operación.
- **Desarrollo del Software.** Involucra todas las actividades dedicadas a la producción del software, según lo especificado en la actividad anterior.
- **Validación del Software.** Esta actividad se centra en asegurar que el sistema proporcione lo que el cliente quiere.
- **Evolución del Software.** Involucra a las actividades que permiten gestionar el cambio.

Las actividades descriptas integran una de las categorías (desarrollo) que se especifican para los procesos denominados **primarios** [Singh Raghu, 1996]. Cabe destacar que los procesos primarios deben ser complementados con los **procesos de soporte** (los cuales dan apoyo a otros procesos realizando alguna función especial) y **organizacionales** (estos establecen, controlan y mejoran los otros procesos) [Singh Raghu, 1996].

Como ocurre con los seres vivos, el software también posee su propio ciclo de vida. En líneas generales lo componen el planeamiento, desarrollo, operación y el mantenimiento del mismo, los cuales constituyen las fases de proceso de desarrollo. Estas fases se caracterizan por las actividades realizadas y los productos elaborados [SEL-81-305, 1992]. Uno de los modelos de ciclo de vida más conocidos es la denominada **cascada pura**, el cual debe su nombre a la secuencialidad de las tareas. Este modelo posee varias extensiones como ser el modelo **cascada modificado, Sashimi y cascada con subproyectos**. También se poseen modelos alternativos, por ejemplo, el **modelo prototipado evolutivo, entrega por etapas, iterativos, espiral**, por nombrar sólo algunos. Tras la existencia de múltiples modelos de ciclo de vida se plantea la discusión sobre qué

modelo utilizar para un proyecto particular. [McConnell, 1997] especifica el modelo más conveniente a emplear dependiendo del contexto de desarrollo que se posea.

El proceso de desarrollo de software implica el *"conjunto de acciones, tareas y procedimientos involucrados en producir un sistema de software a través de su ciclo de vida*"[12]. En este proceso se detallan las tareas y actividades que se realizan en cada fase, como así también las actividades que son continuas [Khodabandeh *et al*., 1994]. Cuando el proceso implica la construcción de algún producto, se suele referir al proceso como su ciclo de vida. Es por este motivo, que al proceso de desarrollo de software suele denominarse comúnmente ciclo de vida del software, debido a que describe la vida de un producto desde su concepción hasta su implementación, entrega, utilización y mantenimiento.

El proceso de desarrollo de software también puede ser visto como la interacción entre tres componentes; **las políticas de la empresa, los mecanismos y estructuras** [Perry *et al.*, 1988]. Estos componentes están fuertemente interrelacionados. El seleccionar uno de estos componentes, tiene como consecuencia implicaciones o limitaciones para los otros dos [Perry *et al.*, 1988].

Otro aspecto importante a tener en cuenta, es lo planteado por [Yourdon, 1992], el cual sugiere que un sistema se modela en tres dimensiones: sus funciones, sus datos y su dinámica. Si bien, se pueden enumerar aún más definiciones y actividades sobre los procesos de desarrollo de software, la implementación de los mismos trae consigo orden y claridad en las actividades de creación del software.

1.2.3 Características del proceso de desarrollo

Se pueden establecer diversas características de los procesos de desarrollo de software, cuya relación depende de cómo el autor define a dicho proceso. Un punto importante para analizar es la **visibilidad** del proceso de desarrollo; no por ser la característica principal sino por estar relacionada con el cambio. La visibilidad de un proceso de desarrollo de software se manifiesta en la documentación confeccionada como resultado de haber puesto en ejecución a dicho proceso.

Continuando con el análisis de las características de este proceso, se señala que *"un buen proceso de software debe ser predecible: se debe conocer la estimación de costos y el cronograma, el producto resultante debe ser robusto y ofrecer la funcionalidad requerida."*[13] Esta postura se relaciona con el punto analizado anteriormente – la visibilidad -. La RT debe desempañar el rol de ser el vehículo por el cual las estimaciones tanto de costo, tiempo o

[12]Khodabandeh, Arash. Palazzi Paolo. Software Development: People, Process, Technology. Proceedings of the 1994 CERN School of Computing, Sopron, Hungary. 1994. Pág 5.

[13] Khodabandeh, Arash. Palazzi, Paolo. Op.Cit.1994:5.

esfuerzo producido en las etapas tempranas del proyecto se constaten con lo ocurrido durante cada una de las etapas del ciclo de vida del mismo.

Comprensión	¿Hasta qué punto se define completamente el proceso y qué tan fácil es comprender su definición?
Visibilidad	¿Las actividades del proceso culminan en resultados claros de tal forma que el progreso del proceso es visible externamente?
Apoyo	¿Hasta qué punto las actividades del proceso pueden apoyarse en las herramientas CASE?
Aceptación	¿El proceso definido es aceptable y utilizable por los ingenieros responsables de producir el producto de software?
Fiabilidad	¿El proceso diseñado es de tal forma que los errores del proceso se evitan o identifican antes de que se conviertan en errores del producto?
Robustez	¿El proceso puede continuar a pesar de los problemas inesperados?
Mantenibilidad	¿El proceso puede evolucionar para reflejar los requerimientos organizacionales cambiantes o las mejoras identificadas del proceso?
Rapidez	¿Qué tan rápido se puede complementar el proceso de construcción de un sistema a partir de una especificación dada?

Tabla 1 Características del proceso de desarrollo de software [Sommerville, 2002]

1.2.4 Proceso de desarrollo orientado a proyectos pequeños

En ocasiones, la importancia de especificar un proceso de desarrollo de software se desestima cuando se llevan a cabo proyectos etiquetados como **pequeños**. Antes de analizar esta conducta errónea, se deben consensuar los factores que inciden en el desarrollo de los proyectos para ser catalogados como **pequeños**. Tales factores se agrupan en [Russ *et al*., 2000]:

- **Tamaño de la organización.** Se tiene en cuenta cómo influye el acceso a una infraestructura de servicios y asesoramiento. Un aspecto importante es el número de personas que desempeñan los distintos roles que se requieren en el proceso de desarrollo.
- **Complejidad del proyecto.** Una de las maneras de clasificar la complejidad es examinar el conocimiento específico del dominio que se requiere para el desarrollo del proyecto.
- **Interacciones.** Las interacciones entre el personal se caracterizan por ser informales y no poseer una estructura de gestión bien definida.

La falta de interés en establecer un proceso para proyectos pequeños se fundamenta en la percepción de que estos necesitan menos coordinación de la que brinda su especificación. Establecer el proceso de desarrollo de software es crítico para el éxito del mismo, tanto en los proyectos catalogados como grandes como también en los pequeños [Russ *et al*, 2000]. En estos últimos la importancia no radica en el gran número de dependencia sino en otros factores tales como [Russ *et al*, 2000]:

- Se presta demasiada atención a la actividad favorita de los desarrolladores.
- Los desarrolladores están atrapados en la rutina y sólo profundizan sus acciones en vez de continuar.
- No hay pautas claras de lo que se tiene que hacer.

Los tres puntos citados destacan la conveniencia de no dejar al azar las actividades a desarrollar en los proyectos pequeños. Otra característica importante en este tipo de desarrollo es lo concerniente al desempeño de los roles, ya que, una o más personas pueden tener roles simultáneos. El tamaño de la organización, es un factor a tener en cuenta en la distribución de estos roles.

Recientes investigaciones han concluido que los proyectos catalogados como pequeños poseen una mayor probabilidad de ser exitosos, dado que tienden a ser más manejables [Standish, 1999].

Tamaño del proyecto	Personas	Tiempo	Ratios de suceso
Menos de $750K	6	6	55%
$750K a $1.5M	12	9	33%
$1.5M a $3M	25	12	25%
$ 3M a $6M	40	18	15%
$ 6M a $10M	+250	+24	8%
$$Más de $10M	+500	+36	0%

Tabla 2 Indicadores que afectan al éxito del proyecto [Standish, 1999]

1.3 Software - Proceso y sus relaciones con la RT

Al analizar las distintas definiciones acerca del software y su relación con la implementación de la RT, se observan ciertos hechos interesantes. Si se define al software sólo como un **conjunto de programas**, la implementación de traceability se traslada al seguimiento del código que se ha desarrollado. Distintos lenguajes de programación ofrecen la opción **trace,** la cual realiza el seguimiento del código (Visual Basic posee la opción **Paso a paso por instrucción**). Sin embargo, al percibir al **software como producto,** se necesitará contar con esquemas de traceability del tipo de UML. Estos esquemas enfatizan el seguimiento de los documentos confeccionados durante el desarrollo. Por último, si se percibe al **software como conocimiento,** se deberá contar con un esquema de traceability en el cual se definan los distintos tipos de traces y el contexto en el cual deben ser capturados [Wieringa, 1995].

Cabe recordar que los beneficios de establecer y documentar un modelo de proceso de desarrollo de software se manifiestan en la comprensión de las actividades, recursos y restricciones involucradas en el desarrollo de software. De esta manera se favorece la comunicación y gestión del grupo de desarrollo al establecer roles y responsabilidades claramente definidas [Pfleeger, 2002] [Curtis *et al.*, 1992]. Los beneficios mencionados no se lograrían sin la adecuada definición de un mecanismo que permita el seguimiento de lo elaborado durante el desarrollo. Es por este motivo que "*el proceso de desarrollo de*

software siempre debería tener como hilo conductor a la trazabilidad de requisitos"[14]. Además se debería contar con un ambiente de desarrollo en el cual se pueda especificar y trabajar con la información de traceability producida, ya sea para la captura, diseño y verificación de los requerimientos de software [Piattini, *et al.*, 2003].

No se debe olvidar que todo proceso debe ser adaptado al contexto (tanto del proyecto como de la organización) en el cual está inmerso el proceso de desarrollo de software, determinando así el tipo de desarrollo a seguir [Basili, 1989], [El Eman, 2001], [Pfleeger, 2002]. Es por ello que se deberán adaptar los mecanismos de traceability a las necesidades específicas del proyecto de software. Estas adaptaciones representan el factor de éxito para la implementación de la RT [Dömges *et al*, 1998].

Por último, se señala que sin la correcta especificación de RT, los procesos de soportes definidos en [Singh Raghu, 1996] no podrán ser conducidos de manera ágil y adecuada.

En síntesis, el software es mucho más que una secuencia de instrucciones. La **intangibilidad** y la **maleabilidad** son sus características distintivas**.** Es importante recalcar que el software involucra importantes aspectos del conocimiento relacionados con el desarrollo del mismo. Si se establece o define un proceso débil, los productos resultantes de dicho proceso van a sufrir indudablemente [Pressman, 1998]. Continuando con el estudio de la relación **producto - proceso**, Basili enfatiza que la misma no puede ser entendida solamente por el análisis, sino que se necesita experimentar y aprender de la propia experiencia para poder configurar adecuadamente el proceso de desarrollo para el logro de resultados óptimos [Basili, 1989]. Para alcanzar dichos resultados se hace evidente hacer hincapié en el modelado del proceso de desarrollo. Éste permitirá encontrar las inconsistencias, redundancias y omisiones de lo especificado [Pfleeger, 2002], logrando además la continua mejora del mismo [Curtis *et al.*, 1992]. En la actualidad se puede afirmar que hay carencias de modelos que permiten razonar acerca del proceso y el producto. La carencia de estos modelos profundizan aún más la característica de invisibilidad del software e impacta negativamente en la posibilidad de la comprensión de las implicaciones del cambio [Basili, 1989].

[14] Piattini, Mario G. García Rubio, Félix O. Calidad en el desarrollo y mantenimiento del software. Ra-Ma Editorial. Madrid, España.2003 Pág 17.

CAPÍTULO 2 GESTIÓN DE LOS REQUERIMIENTOS

Aunque la tesis no tiene el objetivo de ahondar con profundidad temas relacionados con la problemática de los requerimientos de software, es necesario tener una visión general sobre la misma para poder comprender aspectos relacionados con la RT. Es por este motivo que se describirán diversos conceptos referidos a los requerimientos, como así también al proceso por el cual los mismos se identifican, definen y validan.

2.1 Los Requerimientos

2.1.1 Definiciones

El desarrollo de proyectos informáticos presenta una variedad de desafíos pero *"la parte más difícil de construir un sistema de software es precisamente saber qué construir"*[15]. Las actividades o acciones encaminadas a definir el qué construir deben estar claramente definidas y organizadas debido a que *"ninguna otra parte del trabajo afecta tanto al sistema si se realiza incorrectamente. Ninguna es tan difícil de corregir más adelante"*.[16] Estas afirmaciones sintetizan la problemática de establecer los requerimientos de software.

IEEE define a los requerimientos de software como la *"declaración de una capacidad esencial del software a ser desarrollado"*[17]. En otro estándar se señala que un requerimiento es una:

1. *"Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo.*
2. *Condición o capacidad que debe satisfacer o poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.*
3. *Representación documentada de una condición o capacidad como en 1 o 2."* [18]

2.1.2 Caracterización de los requerimientos

El equipo de desarrollo puede organizar a los requerimientos agrupándolos según las características que posean. Como resultado de esta clasificación, la gestión de los mismos se torna más manejable. De las diferentes definiciones y clasificaciones referidas a los requerimientos se observa que difieren principalmente en el grado de abstracción de la

[15] Brooks, Frederick P. "No Silver Bullet: Essence and Accidents of Software Engineering". En: IEEE Computer, 20, (4) April 1987.Pág. 17
[16] Brooks, Frederick P 1987 Op.Cit: 17
[17] IEEE Std-830. 1984 Pág 14.
[18] STD 610.12 IEEE Standard Glossary of Software Engineering Terminology 1990. Pág 62.

declaración de los mismos, como así también desde el punto de vista de los stakeholders que los definen.

[Pinheiro, *et. al* 1996] describe algunas características distintivas referidas a los requerimientos. La primera de ella se refiere a la evolución de los requerimientos. Si los requerimientos se expresan para solucionar algún problema del mundo real, que está en continuo devenir, tales hechos repercutirán en la volatilidad de los mismos [Sommerville, 2002]. Además, *"los requerimientos se encuentran situados, en tanto que dependen de los detalles de una situación particular o contexto del cual emergen."*[19] Este hecho implica adicionar actividades en el proceso de desarrollo, como ser tomar notas sobre aquellos detalles que pueden facilitar la interpretación de los mismos. Por último, Pinheiro señala que "*los requerimientos son negociables en tanto que resultan de las discusiones entre las partes interesadas".*"[20]

Un punto importante a destacar es el ciclo de vida de un requerimiento, el cual presenta una gran variedad de relaciones con información muy útil para la implementación de RT. *"La evolución de los requerimientos es un aspecto crítico del desarrollo de software, de manera tal que el soporte apropiado para la evolución parece ser crucial para el éxito de cualquier sistema de traceability"*[21]*.* Este tipo de información involucra ciertos aspectos, como el saber de donde se extraen, si se hace referencia a una derivación o a un refinamiento desde otro requerimiento. Así también, se debe detallar si el requerimiento fue desarrollado, validado, cancelado o reemplazado por otro.

2.2 Ingeniería de Requerimientos

Tras lo expuesto en el apartado anterior, se percibe la necesidad de disponer de un proceso que permita identificar las necesidades, deseos y expectativas de las distintas personas involucradas en el desarrollo de tales capacidades. La manera en que éstas se plasman en el software con sus diferentes grados de satisfacción, constituye la medida principal de éxito del desarrollo del sistema [Nuseibeh *et al.*, 2000]. A este conjunto de tareas se las encuadran dentro de la **Ingeniería de Requerimientos -RE -**, la cual representa el *"proceso de descubrir el propósito por medio de identificar a los stakeholders, sus necesidades y documentarlas en forma tal que se puedan disponer en el análisis, comunicación y las sucesivas implementaciones"*[22]*.* También se puede definir al proceso de RE como *"el proceso de establecer los servicios que el sistema debe proveer y las*

[19] Pinheiro, Francisco A. C. Goguen, Joseph A. "An Object Oriented Tool of Tracing Requirement" En: IEEE Software. March 1996, Pág 53.
[20] Pinheiro, Francisco A. C. Goguen, Joseph A. Op. Cit:1996. Pág 53.
[21] Pinheiro, Francisco A. C. Goguen, Joseph A. Op. Cit:1996. Pág 60.
[22] Nuseibeh, Bashar. Easterbrook, Steve."Requirement Engineering: A Roadmap".Pág 1.

restricciones bajo las cuales deben operar"[23]*.* Establecer los requerimientos de software para un sistema particular constituye un proceso por el cual se aclaran y especifican las necesidades de los clientes y usuarios del sistema [SEL82-305, 1992].

El proceso de RE es **sistemático**, **iterativo y cooperativo**, e involucra además actividades de análisis, documentación y chequeo [Loucopoulos *et al.*, 1995]. De todas estas características se advierten dos tareas primordiales que engloban a las anteriores, la determinación de los requerimientos y la gestión de los mismos [Jarke *et al*., 1993].

El proceso RE presenta serias dificultades. Los objetivos de los diferentes stakeholders pueden ser divergentes, dependiendo de las perspectivas que posean de la organización y de los trabajos que realizan. Esta situación inevitablemente conducirá a conflictos de intereses cuando estos objetivos no puedan ser articulados [Nuseibeh *et al.*, 2000]. Un típico ejemplo se presenta cuando los intereses de las personas que solventan el sistema a construir son diferentes a los de las personas que van a utilizar el sistema, encontrando allí disímiles situaciones conflictivas a resolver que repercutirán en el proceso de RE [Sommerville, 1995]. Otro hecho bastante peculiar es el advertido por Pearson en [Pearson, 1996], el cual expresa que en determinadas ocasiones el cliente o persona que encarga el sistema puede no saber con claridad lo que desea, acrecentando aún más la complejidad en el proceso de RE.

No todos los problemas del proceso de RE subyacen del lado del cliente. En determinadas ocasiones la falta de conocimiento del dominio del problema por parte de los desarrolladores [Pearson, 1996], perjudican el entendimiento de las necesidades de los clientes. Otros inconvenientes relacionados a los requerimientos surgen por [Sawyer *et al*, 1997]:

- No reflejar las necesidades de los usuarios.
- Ser incompletos o inconsistentes.
- Existir malentendidos o desacuerdos entre los clientes y los encargados de analizar y documentar los requerimientos.

Un aspecto de vital importancia para establecer un buen proceso de RE es la **rationale o fundamentación** asociada a las actividades de definición de los requerimientos [Sommerville, 1995]. La importancia de la fundamentación se refleja en que sin ella, ciertas facilidades que brinda el nuevo sistema, pueden parecer arbitrarias y su importancia no es comprendida por los desarrolladores o - llegado el caso - por las personas que realizan el mantenimiento del sistema [Sommerville, 1995].

[23] Sommerville, Ian. Software Engineering . London. Addison Wesley. 1995. Pág 64.

2.3 La Especificación de Requerimientos de Software -SRS-

Gotel afirma que el objetivo del proceso de RE es producir una descripción, que se establezca como **baseline,** y guíe la construcción de los sistemas. También deberá facilitar la evaluación del sistema construido por parte de los usuarios y desarrolladores [Gotel, 1995]. Una de las actividades principales en el proceso de RE es detallar a los requerimientos identificados en un documento denominado **Especificación de Requerimientos de Software – SRS – .**

Este documento puede ser visto como *"la declaración oficial de lo que se requiere de los desarrolladores del sistema"*[24]*.* IEEE define al documento SRS como *"una especificación para un producto de software particular, programa o conjunto de programas, que realizan ciertas funciones en un ambiente específico".*[25] En dicho documento se debe dejar bien en claro qué funciones van a ser realizadas, qué datos se deben producir o qué resultados se deben obtener, como así también el servicio a ser realizado [IEEE STD-830, 1984]. No obstante, al confeccionar la SRS no se deben especificar ítems de diseño, como las particiones del software en módulos, la descripción del flujo de información o el control entre dos módulos, entre otros [IEEE STD-830, 1984].

Pfleeger expresa la necesidad de asegurar que se comprendan y utilicen correctamente los requerimientos de software desde la óptica de los desarrolladores como también de sus clientes. Esto se hará posible si los requerimientos son de alta calidad [Pfleeger, 2002]. Se han estipulado diferentes características o criterios a seguir para que los requerimientos o la SRS sean de calidad. Una SRS correcta, no ambigua, completa, consistente, ranqueada por importancia, verificable, modificable y traceable es considerada según [IEEE STD-830, 1998] una especificación que posee calidad. Los atributos que reconocen la confección de una SRS de calidad ha sido un tema muy discutido [Wiegers, 1999], aunque un atributo común en todos ellos es que la especificación debe ser traceable, al igual que los requerimientos que la forman.

El entendimiento y la buena gestión de los requerimientos - debidamente expresados en la SRS- representa una de las claves para el éxito del desarrollo de cualquier proyecto informático [Lavazza *et al*., 2000]. Este documento permite entre otras cuestiones mejorar la comunicación entre los desarrolladores y los clientes. Es importante destacar que gran parte de los problemas en el desarrollo del software provienen de las imprecisiones o la falta

[24] Sommerville, Ian. Op.Cit.1995:68.
[25] STD-830 IEEE Recommended Practice for Software Requirements Specifications. Software Engineering Standars Committee of the IEEE Computer Society. 1998 Pág 3.

de información de lo especificado en la SRS [Sommerville, 2002] Un primer paso para despejar dichas imprecisiones es poder consensuar en forma conjunta entre los distintos stakeholders que intervienen en el proyecto, los distintos grados de abstracción con los que se definen y documentan a los requerimientos. Este hecho permitirá un mejor entendimiento entre los participantes del proyecto como así también mejorará la comunicación entre los mismos. El desarrollo de software se caracteriza por la evolución de los requerimientos. Una efectiva gestión de estos requerimientos es de vital importancia para el desarrollo de los proyectos informáticos [Lavazza *et al*., 2000]. Es por esto que la implementación de RT, debe constituirse como el mecanismo principal mediante el cual se logre apoyar la gestión del cambio, la fundamentación de los requerimientos y la identificación de sus fuentes.

CAPÍTULO 3 TRACEABILITY DE REQUERIMIENTOS

En este capítulo se analizan los diferentes aspectos relacionados con la RT. En primer lugar, se comparan las diversas traducciones al castellano del vocablo **traceability** y además, se examinan distintas definiciones referidas a la RT. Luego se revisan de las relaciones de la RT con las diversas áreas del proceso de desarrollo. Por último se describen algunos de los problemas relacionados con la implementación de la RT.

3.1 Concepto de RT

3.1.1 Interpretaciones de los vocablos utilizados

Cuando se traduce el término **traceability** al castellano, se produce cierta confusión al no identificar de forma unívoca cuales son las tareas que involucra este concepto. Esto se debe principalmente a las posibles traducciones **-trazabilidad o rastreabilidad-,** las cuales representan acciones disímiles. Al analizar el impacto de las mismas en el uso e implementación de la RT se observa que:

- La acción de **trazar,** involucra introducir elementos (componentes de información) que identifican alguna situación, hecho o acción para su entendimiento y posterior seguimiento, es decir que posibilitan su rastreo.
- La acción de **rastrear,** involucra recorrer estas trazas, previamente insertadas en lugares estratégicos para encontrar o llegar rápidamente a la información deseada.

El problema radica en que la RT debe enmarcar, como se verá más adelante, un proceso que incluya ambas acciones; introducir las trazas con los componentes de información pertinentes y la confección de mecanismos para la recuperación por medio del rastreo de la información. Entonces cabe preguntarse cuál es la traducción más adecuada que incluya ambas acciones.

Sin la acción de trazar, sería difícil rastrear los requerimientos durante el transcurso del desarrollo del proyecto y el sólo hecho de introducir trazas en este proceso, no alcanza para la definición de un esquema de RT debido a que no se establece cómo recuperar la información que se ha traceado.

Es por ello, que se opta por utilizar el término **traceability,** ya que éste tiene la particularidad de reflejar ambas acciones en un sólo vocablo. Aunque el vocablo **trace** se traduce como huella, rastro o traza, se ha optado por no traducirlo.

3.1.2 Definiciones referidas a la RT

No hay consenso acerca de los diversos temas vinculados a la RT, como ser su definición o las causas de sus principales problemas [Gotel, 1995]. Esta situación se refleja en la gran disparidad de definiciones las cuales inciden en las prácticas e implementaciones actuales de la RT [Kenny, 1996] [Ramesh *et al*., 1995]. En [Gotel, 1995] se clasifican las definiciones en cuatro categorías:

- **Dirigidas por el propósito.** El énfasis se centra en lo que la RT realiza.
- **Dirigidas por la solución.** El énfasis se centra en cómo la RT se implementa.
- **Dirigidas por la información.** Indican qué tipo de información debe ser traceada.
- **Dirigidas por la dirección.** Indican la dirección en la que la RT se implementa.

Una de las contribuciones del trabajo doctoral de Gotel es plantear una definición que cubra la mayoría de las percepciones de las categorías descriptas anteriormente. Gotel define a la RT como "*la habilidad para describir y seguir la vida de un requerimiento tanto hacia adelante como hacia atrás (ej: desde sus orígenes, a través de su desarrollo y especificación, a su puesta en marcha y uso a través de períodos de refinamientos progresivos e iteración en cualquiera de estas fases)*"[26]. De la definición anterior se pueden señalar algunos aspectos importantes:

- Se enfatiza tanto la actividad **proveer o suministrar traces** como también el **usar los traces**. Se deja bien en claro que la RT debe implementar ambas acciones.
- Un aspecto que se remarca es la ***bidireccionalidad*** de la RT **(forward y backward)**, a través de todas las clases de información que puedan ser pertinentes para la gestión de los requerimientos de software.
- La definición no establece cómo se debe implementar la RT.

El estándar STD-830 señala que *"una SRS es traceable si el origen de cada requerimiento es claro y si se facilita el referenciamiento a cada requerimiento en el futuro desarrollo o en incrementos de la documentación"*[27]. Si bien dicho estándar no se específica como se implementa la RT, en el mismo se establecen (al igual que en la definición de Gotel y de otros autores) dos direcciones que implementan los mecanismos de RT:

- **Backward Traceability -Hacia atrás-.** Esta dirección se implementan los mecanismos que vinculan a un determinado producto de trabajo con sus respectivas fuentes. También hace referencia a los mecanismos por los cuales se regresa a las anteriores versiones de producto de trabajo.

[26] Gotel, Ornela. C. Z. Op. Cit.1995:35.
[27] STD-830. Op.Cit.1998:8 .

- **Forward Traceability -Hacia delante-.** Esta dirección se implementan los mecanismos que vinculan a un requerimiento con los documentos desarrollados.

En [Davis, 1990] se establecen cuatro tipos de direcciones de traceability, siendo las mismas:

- **Forward from requirements - Hacia adelante desde los requerimientos -** Esta dirección se implementan los mecanismos mediante los cuales los requerimientos son asignados a los distintos componentes que constituyen el sistema de software.
- **Backward to requirements - Hacia atrás hacia los requerimientos -** Se desarrollan los mecanismos que permitan verificar la conformidad de cada uno de los componentes que integran el sistema de software, evitando de esta manera implementar prestaciones no solicitadas por los usuarios.
- **Forward to requirements - Hacia delante de los requerimientos -** Los cambios en las necesidades de los stakeholders impactan en los requerimientos que se han especificado. Es por ello que en esta dirección se implementan los mecanismos que permitan evaluar y gestionar el cambio producido.
- **Backward from requirements - Hacia atrás desde los requerimientos -** Es de suma importancia identificar el origen de los requerimientos que han sido especificados. Tal información será vital al momento de realizar su validación.

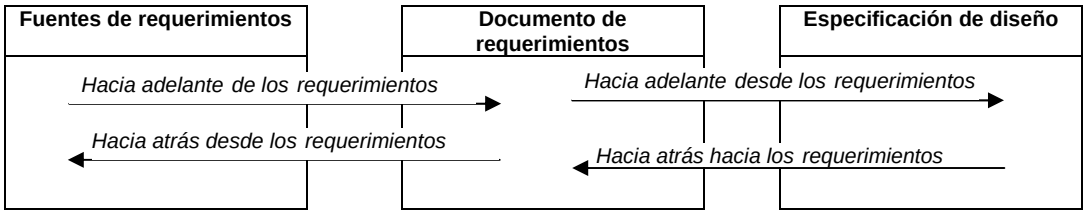


Figura 3.1 Direcciones de traceability

A su vez, Gotel clasifica las direcciones de la RT en:

- **Pre -Traceability de Requerimientos (pre-RT).** Se define como ***"****la habilidad para describir y seguir aquellos aspectos de la vida de un requerimiento anterior a su inclusión a la SRS tanto en la dirección hacia delante como hacia atrás (ej. Producción y refinamiento de requerimientos)"*[28].
- **Post - Traceability de Requerimientos (post-RT).** Se define como *"la habilidad para describir y seguir aquellos aspectos de la vida de un requerimiento que resulta de su inclusión en la SRS tanto de las direcciones hacia delante, como hacia atrás (ej. el despliegue de los requerimientos y el uso)"*[29]

[28] Gotel, Ornela C. Z. Op. Cit.1995:78
[29] Gotel, Ornela C. Z. Op. Cit.1995:78

Esta clasificación es una generalización de las cuatro direcciones propuestas en [Jarke, 1998], ya que los enlaces **Forward from requirements** y **Backward to requirements** implican **Post - RT** y las direcciones **Forward to requirements y Backward from requirements** implican **Pre - RT**.

También se puede describir a la RT teniendo como criterio la fase de la vida del requerimiento [Gotel, 1995]. Si se implementan los mecanismos que vinculan las distintas versiones de un determinado producto de trabajo se denomina **RT horizontal.** Estos mecanismos indican la amplitud del alcance de la misma. Por el contrario, si se implementa los mecanismos que vinculan los productos de trabajos confeccionados en las diferentes etapas del proceso de desarrollo, se denomina **RT vertical.** Estos mecanismos indican la profundidad del alcance de la misma.

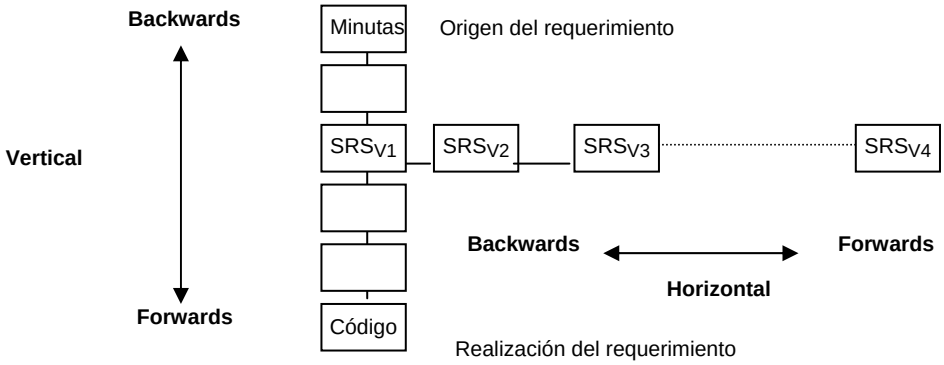


Figura 3.2. Distintas clases de RT [Gotel, 1995]

3.2 Diferentes visiones referidas a la RT

En este apartado se analizan y describen diferentes puntos de vista relacionados con la implementación de RT en los proyectos de software.

3.2.1 RT desde el punto de vista de la calidad

Como cualquier fabricante de un producto, los ingenieros de software también deben especificar los métodos, prácticas y procedimientos para asegurar que los productos desarrollados posean atributos de calidad aceptable. Se percibe a la calidad como el grado en que el software satisface tanto a las necesidades como expectativas de los clientes y usuarios [IEEE Std-610.12, 1990]. La introducción de la calidad en los productos de software ha sido tratada de forma exhaustiva durante el transcurso de la década del 90. Por este motivo se ha caracterizado a esta década como la era de la calidad [Basili *et al*, 1991].

Diversos autores plantean la implementación de RT como una de las estrategias para alcanzar a producir software de calidad. [Kenny, 1996] define dos pasos fundamentales para lograr tal fin. El primero de estos es confeccionar una especificación de requerimientos de software con atributos de alta calidad. El paso posterior es poseer la capacidad de seguir a los requerimientos durante todo el proceso de desarrollo. Otros autores sencillamente

expresan que la RT está íntimamente relacionada con la calidad [Wright, 1991], llegando a ser el principal mecanismo por el cual se puede desarrollar software de alta calidad [Gotel, 1995]. Aunque en la actualidad se dispone de diversa literatura y estándares referidos a modelos de calidad y traceability; la mayoría no proveen un modelo comprensivo sobre la información que debe ser capturada y usada como parte de este esquema de RT [Ramesh *et. al*, 1995b].

En síntesis, existen diversos motivos que fundamentan la implementación de RT en los procesos de desarrollo como parte de una estrategia para desarrollar software de alta calidad.

3.2.2 RT desde el punto de vista de la satisfacción del cliente.

Para asegurar que el software esté acorde a la especificación confeccionada y cumpla con las necesidades de los clientes, la Ingeniería de Software dispone de dos procesos, la verificación y la validación –V&V- [Sommerville, 2002]. El proceso de validación responde al interrogante **¿Estamos construyendo el producto correcto?**, en cambio el proceso de verificación responde a **¿Estamos construyendo correctamente el producto?** [Boehm, 1979]. Aunque existe cierta similitud entre estos conceptos el plantear estos interrogantes ayudan a clarificarlos.

La RT representa una herramienta de gran utilidad cuando se llevan a cabo ambos procesos, debido a que permite verificar las características del sistema. Las mismas se contrastan con la SRS permitiendo encontrar contradicciones o errores en las fuentes de los requerimientos [Pinheiro *et al.*, 1996][Ramesh *et al*, 1995]. Es importante recordar que la validación es más que un simple mecanismo de traceability [Pfleeger, 2002].

3.2.3 Los Stakeholders y la RT

Al implementar traceability se tiene que tener presente la información que suministra la misma a los distintos stakeholders que intervienen en el proyecto de software. Por ejemplo, el **líder de proyecto** hace uso de los mecanismos de RT para realizar el seguimiento y control del proyecto, como así también para evaluar o estimar el impacto de los cambios producidos en los requerimientos. Si se han implementado mecanismos concernientes a la Pre-RT, el líder de proyecto podrá identificar el origen de los requerimientos que han sido especificados. También se podrá monitorear el estado de los requerimientos, si ya han sido elaborados, validados o llegado el caso si un requerimiento fue cancelado o redefinido por otro. En otras palabras, la RT provee visibilidad al proceso de desarrollo, como así también permite gestionar la evolución de los requerimientos [Wieringa, 1995], [Ramesh *et. al*, 1995b]. En definitiva, la implementación de RT le facilita al **líder de proyecto** contestar interrogantes tales como *¿Quién solicitó determinado requerimiento? ¿Cuál es el impacto al*

modificar o eliminar el requerimiento REQ1? ¿En qué grado de avance está la implementación del requerimiento REQ2?

Los **diseñadores** emplean los mecanismos de RT para poder asignar los requerimientos a los componentes confeccionados en el proceso de diseño. Al poseer dicha información, el diseñador podrá rastrear en caso de presentarse dudas o situaciones conflictivas, quien o quienes han solicitado tal requerimiento. La implementación de RT, también permitirá realizar con mayor precisión la estimación del impacto del cambio en sus diseños, como también facilitará la evaluación del reuso de los componentes. Además se podrá verificar con mayor facilidad si estos diseños satisfacen los requerimientos especificados. La información de RT también debe abarcar el registro de las alternativas de diseño confeccionadas, como así también la justificación de lo realizado [Wieringa, 1995], [Ramesh *et. al*, 1995b]. En definitiva, la RT facilita a los **diseñadores** contestar interrogantes tales como *¿Cuáles son los requerimientos afectados al cambio de un determinado componente? ¿Cuáles son las alternativas que se han confeccionado para la implementación de determinado componente? ¿Por qué se decidió implementar tal subsistema de determinada manera? ¿Ayuda a cumplimentar lo expresado en el requerimiento R1 lo implementado en el componente X3?*

Cabe destacar que muchas de estas necesidades de información se comparten entre los distintos stakeholders [Ramesh *et. al*, 1995b]. Los **encargados del mantenimiento** del sistema al igual que el **líder del proyecto** necesitan saber si el cambio de un requerimiento afecta a los componentes ya implementados. Además, deben evaluar el cambio producido en la implementación de un determinado componente, como así también verificar si lo implementado cumplimenta con lo especificado en los requerimientos. Esto último facilitará la relación con los **clientes y/o usuarios**, dado que así se podrá saber con certeza si los requerimientos especificados han sido satisfechos [Wieringa, 1995], [Ramesh *et. al*, 1995b].

Por último se observa que la diversidad de información que necesitan los stakeholders no sólo impacta en la implementación de la RT, sino que también se refleja en las múltiples definiciones de la misma.

3.3 Problemas de la RT

Diversos autores señalan como problemática principal de la RT la existencia de diferentes perspectivas sobre su definición e implementación. Estas discrepancias se manifiestan en la diversidad de implementaciones, muchas de ellas motivadas por las disímiles necesidades de información por parte de los stakeholders. La RT no puede dirigirse a todos los problemas generados en el proceso de desarrollo y en determinadas ocasiones no es la mejor solución para algunos de ellos [Gotel, 1995].

Gotel asevera que el *"principal problema de la RT es la inhabilidad para localizar y acceder a los orígenes (humanos) de los requerimientos; la información relacionada a los requerimientos y al trabajo relacionado con los requerimientos"*[30]. Esta postura señala que la mayor parte de los problemas de la RT surgen de la inadecuada implementación de Pre - RT. Wieringa señala que los principales problemas vinculados con la implementación de la RT se relacionan con la gestión de los requerimientos y la asignación de estos en los respectivos componentes de diseño [Wieringa, 1995].

También se advierte que los problemas vinculados a la implementación de RT se relacionan con el nivel de madurez que posee la empresa que desarrolla el software. Si una empresa posee niveles de madurez bajos, los beneficios de mantener la información de RT no son percibidos. Se desea recalcar que **los principales problemas al realizar traceability son organizativos y no técnicos** [Wieringa, 1995]. Es por este motivo que se considera necesario que la implementación de los mecanismos de traceability dirija el desarrollo del software. De esta manera se obtendrá un proceso de desarrollo que posea en su especificación mecanismos de traceability implícitamente definidos y caracterizados. Como resultado de esta especificación, se podrán generar las unidades de traces (vínculos de traceability) pertinentes de manera transparente, con el propósito de no sobrecargar las actividades que integran el proceso de desarrollo. Además se deberán definir las políticas necesarias para definir la granularidad de la información de RT deseada.

Si bien en este capítulo no se han tratado todos los temas vinculados a la RT, se enfatizó los puntos centrales de la misma. Un aspecto que debe ser debidamente tratado es el análisis del vocablo RT, ya que su implementación estará circunscripta por dicho entendimiento. A pesar de que en reiteradas ocasiones se ha expresado la existencia de variadas y divergentes definiciones referidas a la RT, se ha optado por citar sólo la expresada por Gotel debido principalmente a la profundidad y alcance de tal definición. Esta gran variedad de clasificaciones puede derivar de las múltiples necesidades de información que poseen los distintos stakeholders. Al desarrollar este capítulo, se tuvo la intención de recalcar la idea de [Jarke, 1998], el cual señala que la RT emerge como un puente efectivo entre la evolución del sistema con los cambios de las necesidades de los stakeholders. Tal situación se evidencia en que la RT representa el medio que permite la puesta en marcha de los procesos de soporte y es de gran utilidad para la implementación de los procesos catalogados como primarios y organizacionales.

[30] Gotel, Ornela C:Z. Finkelstein, Anthony. Making Requirements Elicitation Traceable.Papers of the Workshop on Requirements Elicitation for Software – Based Systems(RESS'94) University of Keele, Staffordshire, U.K Pág 13.

CAPÍTULO 4 ESTADO DEL ARTE DE LA RT

Aproximadamente desde el año 1990 se han publicado numerosos trabajos referidos a la RT. En [Pinheiro, 2000], [Wieringa, 1995], [Gotel *et. al.*, 1994], [Gotel, 1995] entre otros, se describen algunas de las técnicas, métodos, lenguajes, herramientas y modelos que posibilitan su implementación. Dada la existencia de esta gran variedad de medios para la puesta en marcha de la RT, este capítulo caracteriza a dichos conceptos. Posteriormente se describen y analizan con mayor profundidad algunos de los modelos que implementan RT.

4.1 La RT y sus soportes

Existen en la actualidad una gran variedad de mecanismos que proveen los distintos medios para la implementación y gestión de la información relacionada con traceability. Estos mecanismos se clasifican en técnicas, métodos, lenguajes, herramientas comerciales y modelos. En este apartado sólo se describen los tres primeros puntos. En el apartado 4.2 se analizan los modelos relacionados con la implementación de RT. El análisis y descripción de las distintas herramientas comerciales relacionadas con la implementación de traceability quedan excluidas del presente trabajo.

4.1.1 Técnicas de tracing

Las **técnicas de tracing** son percibidas como mecanismos explícitos a través de los cuales se lleva a cabo la RT [Gotel, 1995]. Las mismas establecen cómo se representan los vínculos entre los ítems que deben ser relacionados [Wieringa, 1995], como así también especifican la manera en que se etiquetan los requerimientos de software con la información considerada relevante [Pinheiro, 2000].

En [Gotel, 1995] se establecen varios tipos de técnicas, las cuales difieren en la cantidad, diversidad de información gestionada, el número de interconexiones que representan y en cómo se pueden adaptar para reflejar los cambios producidos durante el ciclo de vida del proyecto.

Como se ha mencionado, las técnicas que implementan la RT deben etiquetar al requerimiento de software con la información considerada relevante. La técnica **cross reference** lleva a cabo este objetivo mediante la introducción de la frase ***ver sección xx*** en los distintos productos de trabajo. Otra manera de etiquetar a los requerimientos se realiza por medio de la confección de **enlaces de hipertexto** entre los distintos ítems o información a relacionar, como así también puede realizarse por medio de la **numeración e indexado** de los mismos.

Existen también técnicas **centradas en la estructura,** en las cuales se confeccionan redes o diagramas de la documentación del proyecto, con el propósito de agilizar la actualización y propagación de los cambios producidos en los requerimientos de software [Gotel, 1995]. Las matrices de traceability son un mecanismo que permiten relacionar a los requerimientos con los demás documentos confeccionados durante el desarrollo. Estas representan una de las técnicas más utilizadas.

Es importante destacar que las técnicas mencionadas se emplean para desarrollar los distintos métodos y modelos referidos a la RT. La técnica **cross reference,** se utiliza en el método RADIX, como así también las matrices de traceability se utilizan en el método QFD.

Técnicas	Gotel [Gotel, 1995]	Wieringa [Wieringa, 1995]	Pinheiro [Pinheiro, 2000]
Referencia cruzadas	X	X	X
Matrices de traceability		X	X
Centradas en el documento	X		
Centradas en la estructura	X		
Esquemas de índices			X
Modelos de entidad relación		X	

Tabla 3 Técnicas de tracing

4.1.2 Métodos de tracing

Pinheiro define a los **métodos de tracing** como *"un conjunto de actividades organizadas, conjuntamente con los procedimientos necesarios para crear las relaciones entre cada artefacto siguiendo como base un modelo"*[31]. Cabe aclarar, que la especificación de dichos modelos puede no ser explícita, en tanto que la especificación de los procedimientos debe estar relacionada con algunas de las técnicas de traceability [Pinheiro, 2000].

En [Gotel, 1995] se menciona como ejemplo de estos métodos a **Planning and Design Methodology - PDM -, Requirements Specifications and Traceability Methodology - RESPECT - y Quality Function Deployment - QFD -**, y en [Pinheiro, 2000] a **Radix**

El método **RADIX** en realidad, es una herramienta de software, que fue implementada por la empresa AT&T para el desarrollo de ambientes complejos. Esta herramienta se centra en la documentación que se produce en el desarrollo, especificando un paquete de macros (marcas predefinidas), que poseen el objetivo de caracterizar a los requerimientos de software [Yu, 1994], [Pinheiro, 2000]. Estas macros permiten, entre otras cuestiones, la descomposición del documento en particiones (unidades de información), formatear e imprimir los distintos documentos elaborados en el proceso de desarrollo enfatizando la información considerada crítica [Yu, 1994].

El método de traceability **QFD** especifica un proceso por el cual se estructura la captura y refinamiento de requerimientos de software con el fin de entregar productos de alta calidad.

[31] Pinherio, Francisco A. C. Formal and Informal Aspects of Requirements Tracing 2000. III Workshop de Engenharia de Requisitos, WER'2000, Río de Janeiro, Brazil, Julio 2000 Pág 8.

Este objetivo se lleva a cabo mediante la implementación de RT en los componentes del diseño [Lan Tran *et. al,* 1995]. Cabe destacar, que QFD introduce la idea de calidad desde la perspectiva del cliente en las etapas tempranas del ciclo de vida del proyecto. Al implementar QFD en un proyecto de software, también se posibilita el logro de los siguientes puntos [Lan Tran *et. al,* 1995]:

- Incrementa la calidad del producto enfocándose en los requerimientos y necesidades del cliente.
- La adecuación del producto a las necesidades de los clientes permite enfocarse en los parámetros esenciales del diseño.
- Incrementa la comunicación entre los distintos stakeholders que intervienen en el proyecto.
- Facilita el manejo del cambio de los requerimientos.

4.1.3 Lenguajes de tracing

Los **lenguajes** especifican las notaciones usadas para describir el proceso de desarrollo, junto con las reglas que indican su uso [Gotel, 1995]. Algunos ejemplos de estos lenguajes son: **Requirements Specification Language – RSL - , Prototype System Description Language** [Gotel, 1995] [Pinheiro, 2000]. En la actualidad no se cuenta con lenguajes cuyo objetivo principal sea la implementación de la RT [Pinheiro, 2000]. Sin embargo, por medio de dichos lenguajes se puede implementar la RT. En el lenguaje **RSL** las declaraciones **trace_to** o **trace_back** permite la puesta en marcha de la RT. En tanto **Requirement Modelling Language - RML -,** implementa la RT mediante relaciones semánticas binarias entre los distintos objetos [Gotel, 1995].

4.1.4 Conclusión de los soportes de la RT

En [Gotel, 1995], se sugiere que los trabajos realizados hasta la fecha han sido conducidos bajo la impronta de como debe ser implementada la RT, o como la misma afecta a problemáticas específicas del desarrollo. Además en [Pinheiro, 2000] se expresa que los distintos medios con los cuales se implementa la RT no son los adecuados.

Asimismo se advierte que la información a ser traceada debe especificarse de antemano, en caso contrario no podría llevarse a cabo tal vinculación [Pinheiro, 2000]. Este hecho se agrava porque no es posible anticipar todos los enlaces que se necesitan en el futuro ya que el ambiente de desarrollo del software se caracteriza por el continuo cambio. Además si fuese posible preveer la totalidad de los enlaces, no sería adecuado o práctico el registro de todos ellos [Pinheiro, 2000].

En otro sentido, en distintos trabajos descriptos en [Wieringa, 1995] no sólo proponen representar el enlace de traceability, sino también las condiciones mediante las cuales el enlace se ha establecido, su historial y los atributos necesarios que indiquen si el requerimiento se ha cumplimentado con los elementos de diseños establecidos.

Estas consideraciones describen la necesidad de establecer otros mecanismos, además de las técnicas, métodos y lenguajes descriptos y analizados. Los mecanismos a definir deben tener en cuenta los beneficios y problemas de cada uno de los soportes que implementan la RT, integrando a todos ellos en un contexto mayor.

4.2 Modelos de RT

Teniendo presente las consideraciones expresadas, se advierte la necesidad de contar con estructuras de información denominadas **modelos de tracing.** Estos modelos detallan las técnicas, métodos y lenguajes apropiados para la implementación de RT. Además deben proveer las representaciones de los traces a utilizar, como así también establecer las estructuras y relaciones que contienen a estos elementos [Pinheiro, 2000].

Con el fin de completar dichos modelos se han especificado **procesos de tracing,** que se definen como *"los modelos del proceso de desarrollo que incorporan las maneras de tracear los elementos del proceso"*[32]. La idea subyacente en estos procesos es que los traces se produzcan naturalmente como resultado de las actividades, acciones, decisiones y eventos ocurridos durante el desarrollo de software [Pinheiro, 2000]. De esta manera no se generará sobrecarga de trabajo al realizar las actividades que conforman el proceso de desarrollo. En [Wieringa, 1995] se afirma que para que un proceso de desarrollo sea considerado útil, se debe equiparar las necesidades de los desarrolladores con los recursos disponibles para poder mantener la información de RT. A continuación se describen algunos de los procesos de tracing existentes.

4.2.1 ESE - Evolution Support Environment System -

El continuo cambio de los requerimientos caracteriza al desarrollo de los sistemas de software. El poder comprender el significado y el por qué de cada una de las relaciones entre los componentes que integran el sistema de software posibilita la realización de los cambios producidos de manera correcta y eficiente [Ramamoorthy *et. al*, 1990] [Wieringa, 1995]. Se enfatiza de esta manera la necesidad de recolectar y mantener gran cantidad de información considerada relevante referida a los distintos componentes del software y sus

[32] Pinherio, Francisco A. C Op.Cit.2000:10

relaciones. Como consecuencia de estos hechos se ha especificado el **modelo ESE - Evolution Support Environment System -.**

Este modelo provee el framework necesario que permite capturar, mantener y acceder a la información (entidades semánticas) referida a la evolución de los componentes[33] [Ramamoorthy *et. al*, 1990] [Wieringa, 1995]. Si bien este modelo posee el propósito de resolver distintas situaciones problemáticas que ocurren durante el desarrollo, *"el principal objetivo del proyecto ESE es intentar proveer un soporte integrado para la gestión de la configuración del software y traceability a través del ciclo de vida del software y el control de versión"*[34].

En definitiva, **ESE** provee los mecanismos que posibilitan el seguimiento y mantenimiento de la información de los objetos de software de manera automatizada. De esta manera, soporta la evolución del software minimizando el esfuerzo que se necesita para elaborar nuevas versiones de un determinado objeto [Ramamoorthy *et. al*, 1990]. **ESE** se basa en el modelo de **entidad relación** para organizar a los objetos y proveer traceability entre ellos [Ramamoorthy *et. al*, 1990]. El modelo ESE está compuesto por tres capas**; Kernel ESE, la Librería de Software y la Capa de Aplicación**, las cuales conforman su arquitectura.

- **Kernel ESE**. Esta capa especifica los mecanismos (operaciones básicas) que manipulan a los diferentes objetos que se establecen en el modelo. Las operaciones crear, borrar y modificar son algunos de los ejemplos de estos mecanismos. También forman parte de tales mecanismos las operaciones que permiten la definición de los atributos de los objetos y las relaciones que se establecen entre ellos. Los ejemplos de objetos y enlaces que posee este modelo se describen en la tabla 4 y tabla 5. Debido a la gran cantidad de información que se produce en este modelo, se puede clasificar a la misma en **información específica e información genérica.** Esta última mantiene las propiedades de los objetos que son comunes a todas las versiones. La información catalogada como específica establece las diferencias entre las respectivas versiones del objeto. Estas categorías de información también se trasladan a la implementación de dos tipos de enlaces; los **enlaces genéricos** que se utilizan para vincular información genérica entre dos objetos y los **enlaces específicos** que se utilizan para vincular información específica [Ramamoorthy *et. al*, 1990].
- **Capa de Librería de Software.** Esta capa organiza a los objetos jerárquicamente en **módulos, colección de módulos, y miembros** (colección de módulos) [Wieringa, 1995]. Los **módulos** contienen uno o más procedimientos que se agrupan para implementar una función. En tanto que una **colección de módulos** representa una

[33] Los artefactos elaborados con el modelo ESE se les denominan objetos.
[34] Ramamoorthy, C.V. Usuda, Yutaka. Prakash, Atul. Tsai, W. T. " The Evolution Support Environment System". En IEEE Transactions on Software Engineering. Vol 16 Nro 11 November 1990 pág 1232.

agrupación de módulos que se tratan como una única entidad. Por último, los objetos **miembros** son un tipo especial de colección de módulos, los cuales definen la configuración completa de una aplicación del sistema [Ramamoorthy *et. al*, 1990].

- **Capa de Aplicación.** Esta capa provee el acceso a los servicios de las distintas capas [Ramamoorthy *et. al*, 1990]. En la tabla 6 se especifican las operaciones que se definen en esta capa, las cuales definen el soporte necesario para implementar la traceability.

Sufijo	Tipos de Objetos
.c	C Source File
.h	C hearder File
.r	Especificación de Requerimientos de Software
.d	Especificación de Diseño
.od	Otros documentos

Tabla 4 Algunos ejemplos de tipos de objetos del Kernel [Ramamoorthy *et. al*, 1990]

Nombre del enlace	Tipos de Objetos que conecta
module_testdata	Entre los módulos y sus test.
module_design	Entre los módulos y sus especificaciones de diseño.
design_specs	Entre el diseño y sus especificaciones de requerimientos.

Tabla 5 Algunos ejemplos de enlaces del Kernel [Ramamoorthy *et. al*, 1990]

Nombre	Función
Define Type	Define un tipo de objeto.
Define link	Define el nombre del enlace entre dos tipos de objetos.
Link	Establece el enlace entre dos objetos
Unlink	Elimina el enlace entre dos objetos.
IsLink	Corrobora si existe tal vínculo

Tabla 6 Algunos ejemplos de operaciones de la capa de aplicación [Ramamoorthy *et. al*, 1990]

Con respecto a la implementación de RT, en ESE se especifican tres tipos de enlaces entre los diferentes objetos [Ramamoorthy *et. al*, 1990] [Wieringa, 1995]:

- **Jerárquicos:** Estos enlaces se utilizan para vincular a los objetos según su estructura jerárquica. Establecen de esta manera diferentes niveles de jerarquía.
- **Históricos:** Estos enlaces se utilizan para vincular a los objetos teniendo como base el cambio de su historia. De esta manera se relacionan las diferentes versiones de un objeto.
- **Desarrollo:** Estos enlaces se utilizan para establecer cómo un determinado objeto es producido y utilizado en las diferentes etapas del proceso de desarrollo de software.

4.2.2 The Prism- Model of changes

Con el fin de poder gestionar la problemática relacionada con el cambio en los proyectos de software en [Madhavji, 1991] [Madhavji, 1992] se define una descripción abstracta denominada **Prism – Modelo del Cambio –**. Es importante destacar, que este modelo forma parte de un proyecto de investigación (Proyecto Prism), que estudia los distintos ambientes de desarrollo de software [Madhavji, 1991].

La gestión de los cambios en un proyecto de software involucra la posibilidad de identificar los diversos ítems afectados por éste, como así también la clasificación y registro de los datos relacionados con el cambio ocurrido [Madhavji, 1992]. **Prism** lleva a cabo todas estas actividades especificando un **Modelo del Cambio,** conjuntamente con dos elementos denominados **Estructura de Dependencia y Estructura del Cambio,** los cuales conforman la **Infraestructura del Ambiente** [Madhavji, 1991] [Madhavji, 1992].

El **Modelo del Cambio** representa los diferentes niveles de operación que se implementan en Prism, como así también la infraestructura encargada de gestionar las acciones que se producen debido a la implementación de un cambio ocurrido. Además se definen los procesos que se deben llevar a cabo durante el uso y evolución de los demás elementos que conforman el modelo Prism [Madhavji, 1992]. Este modelo es **explícito** (describe cuáles son los componentes importantes del mismo, conjuntamente con sus interacciones), **incremental** y **continuo.** Esta última característica indica que el modelo abarca todo el ciclo de vida del desarrollo. Como consecuencia de este hecho se mantiene el historial de los cambios producidos durante el desarrollo del proyecto. El modelo presenta dos niveles de operaciones. Aquellas que abarcan el uso de los elementos de **Infraestructura del Ambiente** constituyen el **nivel normal**, el cual se diferencia del **metanivel de operaciones.** Este metanivel utiliza la información generada en el nivel normal con el propósito de mejorar la infraestructura del ambiente [Madhavji, 1992].

La **Estructura de Dependencia** especifica todo lo necesario para poder describir los ítems de cambios y sus interrelaciones. Se logra de esta manera identificar cuáles son los ítems afectados debido a un determinado cambio [Madhavji, 1992]. En esta estructura también se definen los distintos niveles de abstracción (individuo, grupo, proyecto, organización, nación, comunidad) de los ítems de cambio, con el objetivo de poder categorizarlos. En cada uno de estos niveles se especifican los ítems básicos y el conjunto de relaciones, que se denomina **sheet**. Los ítems de cambio se clasifican en personas, políticas, leyes, procesos de desarrollo de software, recursos y resultados producidos. La **Estructura del Cambio,** en cambio, posee el objetivo de facilitar la clasificación, almacenamiento y análisis de los datos relacionados con el cambio producido [Madhavji, 1992] [Wieringa, 1995]. Ésta especifica los mecanismos necesarios para obtener información de retroalimentación relacionada con los distintos ítems de cambio [Madhavji, 1992]. La integración con la **Estructura de Dependencia** representa una de las características principales de esta estructura. Tal integración permite asociar los ítems de cambio de la estructura de dependencia (definidos en el componente sheet) con un conjunto de componentes de la estructura de cambio denominados **CS-sheet** (cada CS-sheet describe un cambio particular) [Madhavji, 1992]. En la **Estructura de Cambio** se especifica un framework para el cambio. El framework representa un esquema de clasificación que se

especializa en capturar la información relacionada con un cambio específico. Éste debe estar descripto por un conjunto de propiedades. Tales propiedades abarcan, entre otras cuestiones, datos que especifican el tipo de cambio (correctivo, perfectivo), la ubicación del ítem en el ambiente, la justificación del cambio (origen de la decisión, ventajas y desventajas, razón), tamaño del cambio (impacto, costo), stakeholders responsables de la decisión, entre otras propiedades [Wieringa, 1995] [Madhavji, 1992].

4.2.3 Nature Project

En el proyecto Nature [Jarke *et. al*, 1993], los requerimientos son vistos como procesos que se trasladan a través de un espacio de tres dimensiones, que se deben tracear.

- **Dimensión de representación**. Este rango va desde lo informal a lo formal. Representa los problemas técnicos.
- **Dimensión del acuerdo.** Este rango va desde lo parcial a lo completo. Representa el proceso social.
- **Dimensión de especificación** Este rango va desde el parcial al completo entendimiento de la especificación. Representa el proceso psicológico.

Cabe mencionar que el proyecto Nature ha desarrollado una herramienta de RT denominada PRO-ART.

4.2.4 Estructuras de Contribución

En [Gotel, 1995] se describe un modelo denominado **Estructura de Contribución - Contribution Structure -**, que fue desarrollado como respuesta a los problemas relacionados con las implementaciones de la RT [Gotel *et al*, 1994], [Gotel *et al*, 1995], [Gotel, 1995]. Este modelo abarca la estructura social subyacente del proceso de RE indicando quien o quienes (individuo o grupos) han contribuido en la elaboración de los productos de trabajo [Gotel, 1995]. Cabe destacar que el modelo propuesto se deriva de trabajos del área de la sociolingüística, particularmente de modelos de interrelación entre el lenguaje y la vida social [Gotel, 1995].

Se consideró importante detallar diversos conceptos que facilitan el entendimiento de dicho modelo. El término **estructura social** se refiere al sistema de agentes que participan en el proceso de RE. El concepto de **artefacto** se refiere a cualquier ocurrencia comunicativa de existencia física del proceso de RE [Gotel, 1995]. Por su parte, el término **agente** se refiere a los participantes humanos en el proceso de RE [Gotel, 1995].

Esta estructura implica relacionar los **artefactos** que son vistos como ***contribuciones*** con los **agentes** que han contribuido con su producción. Estos agentes son vistos como ***contribuyentes***. Este vínculo se materializa al utilizar las ***relaciones de contribución.*** Las

relaciones descriptas se caracterizan por ser bidireccionales (un agente contribuye con un artefacto y un artefacto es contribuido por un agente) [Gotel, 1995].

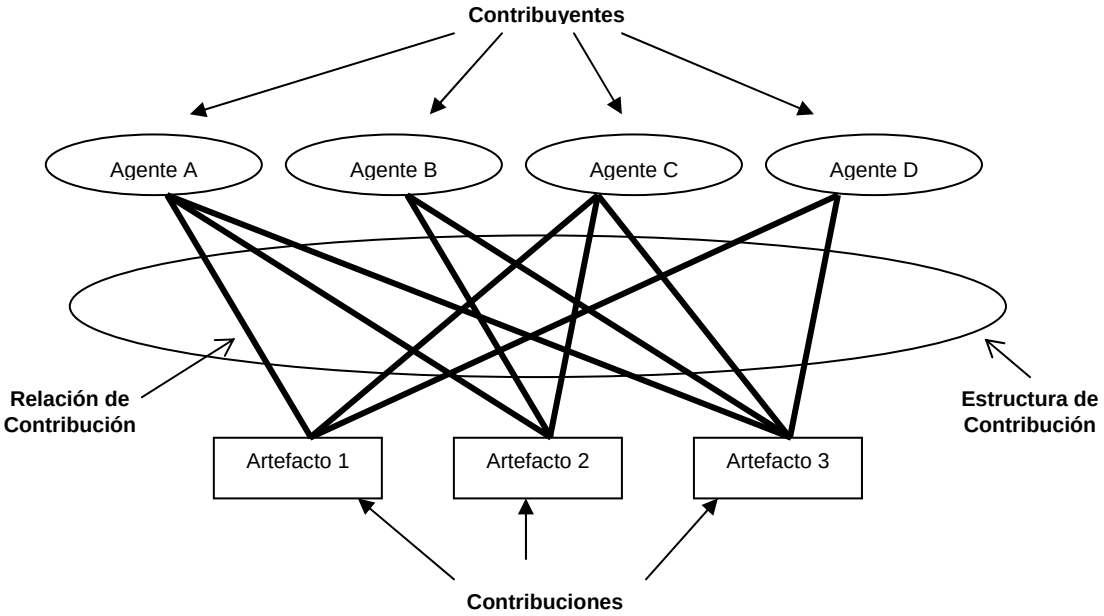


Figura 4.1 Estructura de Contribución [Gotel, 1995]

Para poder definir la naturaleza de las relaciones de contribución entre los agentes y artefactos se emplea el **formato de contribución** [Gotel, 1995]. Los elementos que constituyen este formato delimitan tres capacidades fundamentales por las cuales los agentes pueden contribuir en la elaboración de los artefactos [Gotel, 1995]:

- **Principal**: Representa a los agentes que motivan la producción del artefacto, los cuales se comprometen por lo que éste expresa y son los responsables por sus efectos y consecuencias.
- **Autor:** Representa a los agentes que formulan y organizan el contenido y estructura de la información del artefacto, siendo responsables por su sintaxis y semántica.
- **Documentador:** Representa a los agentes que capturan, almacenan y transcriben la información en el artefacto, siendo responsables por su manifestación física.

La riqueza de las descripciones que se detallan depende fundamentalmente de la forma en que se definen estos enlaces [Gotel *et al.*,1995], conjuntamente con sus atributos. Si se logra elaborar una estructura de contribución rica en relaciones y útiles para el desempeño práctico, la misma podrá suministrar los medios para subsanar la ausencia de la información requerida. Además, de esta manera, se podrá complementar cualquier tipo de información documentada, como así también, gestionar el cambio de los requerimientos desde el punto de vista social [Gotel *et. al.*, 1995]. Lo expuesto es viable en la medida en que se identifique en forma selectiva él o los agentes más apropiados.

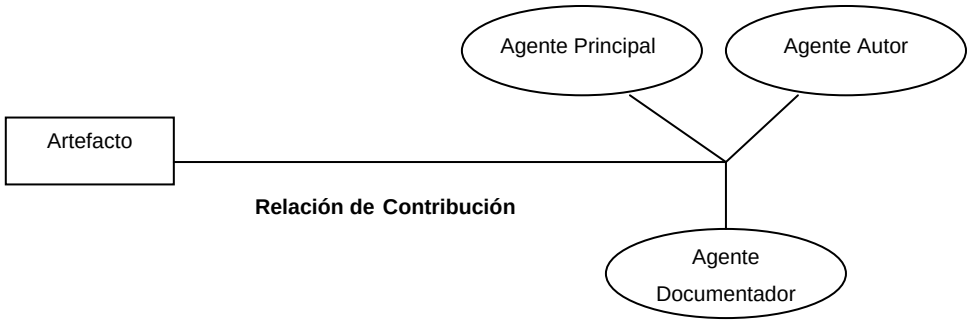


Figura 4.2 Agentes de Contribución [Gotel, 1995]

4.2.5 Modelo de Ramesh

Ramesh y Pohl tipificaron a los usuarios que utilizan a la RT en dos grupos, **Low - End y High - End.** El primer grupo de usuarios define a la RT como la transformación de los documentos de requerimientos al diseño [Ramesh *et. al,* 2001]. En tanto, los **High - End** definen a la RT como el medio por el cual se incrementa la probabilidad de producir un sistema que satisfaga los requerimientos del cliente y además sea sencillo de mantener [Ramesh *et. al,* 2001]. En el próximo capítulo se analizarán las diferencias entre ambos grupos de usuarios.

El esquema de la figura 4.3 describe los metacomponentes y las metarelaciones que intervienen en el metamodelo definido en [Ramesh *et. al*, 2001]. Cada uno de estos elementos deben ser instanciados en relaciones o componentes especializados para crear modelos específicos a ser utilizados en las diversas organizaciones. Debido a la complejidad de cada proyecto, se podrán crear modelos de RT para la confección de proyectos de software específicos [Ramesh *et. al*, 2001].

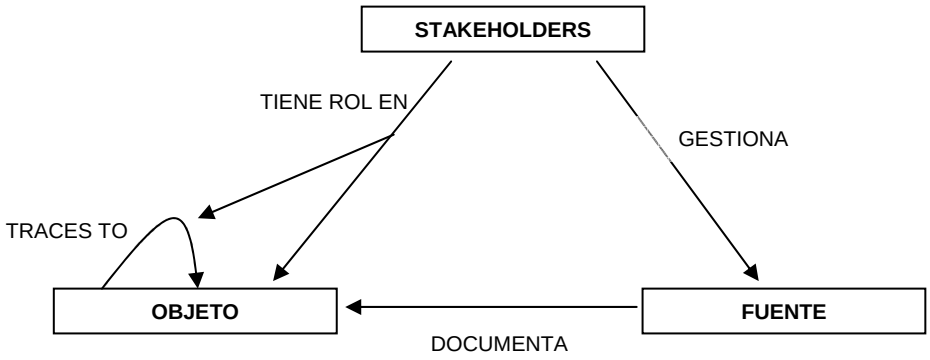


Figura 4.3 Metamodelo de la RT. [Ramesh *et al.,* 1995]

A partir de estos componentes y relaciones se representan las distintas dimensiones de información relacionadas con traceability. El metacomponente **OBJETO** constituye el medio para representar la información ***que*** se utiliza en el modelo. Las instancias de este metacomponente son **Requerimiento, Supuesto, Diseño, Sistemas, Componente,**

Decisión, Fundamentación, etc. El vínculo de traceability entre estos componentes se representa mediante la metarelación ***TRACES_TO*** [Ramesh *et. al*, 2001].

El responder ***quien o quienes*** son los responsables de la creación y mantenimiento de los distintos **OBJETOS,** conjuntamente con sus relaciones, constituye otra de las dimensiones de información de traceability. El metacomponente **STAKEHOLDERS** se utiliza para especificar los roles intervinientes en el desarrollo. Debido a que los stakeholders pueden representar varios roles, se utiliza la metarelación ***TIENE ROL EN*** para reflejar este suceso [Ramesh *et. al*, 2001].

Una importante dimensión de información hace referencia en ***donde*** se documenta lo elaborado. En este modelo todos los **OBJETOS** son documentados por el metacomponente **FUENTE.** Las instancias de este metacomponente son **SRS, Documento de Diseño, Memorandum**, etc. [Ramesh *et. al*, 2001]. Una dimensión de información que está asociada al metacomponente **FUENTE**, se relacionada en ***cómo*** se representa la información.

El preguntar ***por qué*** se ha creado o modificado un determinado metacomponente **OBJETO** representa otra dimensión de información referida a la RT. Dicha dimensión se implementa instanciando al metacomponente **OBJETO** en algún componente cuyo propósito sea el registro de la información de justificación. No se debe olvidar especificar ***cuándo*** la información fue capturada o modificada. Esta información puede ser implementada mediante la especificación de los atributos correspondientes en las instancias del metacomponente **OBJETO** [Ramesh *et. al*, 2001].

El metamodelo puede ser utilizado para representar las dimensiones de información establecidas en [Jarke *et al*, 1993]. Esto se debe porque el metacomponente **OBJETO** cubre los aspectos de la **dimensión del entendimiento.** El metacomponente **STAKEHOLDERS** cubre los aspectos de la **dimensión de agreemment**, y por último los aspectos de la **dimensión de la representación física** puede implementarse con el metacomponente **FUENTE** [Ramesh *et. al.,* 2001].

El metamodelo de RT definido por Ramesh y Pohl enfatiza la importancia del metacomponente **OBJETO.** Dichos autores consideran que este metacomponente constituye el elemento determinante para la efectiva y eficiente implementación del metamodelo. Se advierte que de las seis dimensiones de la información referida a la traceability, tres de ellas definen alguna especialización o atributos del metacomponente **OBJETO**.

En síntesis, el trabajo realizado por Ramesh y Pohl se centraliza en la definición y descripción de los distintos tipos de **OBJETOS** y de sus respectivas relaciones [Ramesh *et. al.,* 2001].

4.3 Comparaciones de los distintos modelos

Este apartado tiene como propósito plasmar las diferencias y similitudes de los modelos de RT que se han descriptos. El primer ítem a comparar se refiere al aspecto central del modelo. El segundo involucra una breve descripción de los modelos y el tercero detalla cuáles son las etapas del proceso de desarrollo que cubre el modelo. Posteriormente, se compara cómo se estructura la información que interviene en el modelo y cuáles son los tipos de enlaces de traceability definidos. Por último se compara si estos modelos poseen criterios que señalen el grado de detalle con que se debe tracear.

ITEM	ESE Evolution Support Enviromment System	Prism Model of Changes	Esquema de Traceability (High End user)	Estructuras de Contribución
Énfasis	Componentes y documentos elaborados en el desarrollo.	Ítems del cambio	Componentes y documentos elaborados en el desarrollo.	Contribución de los agentes en la elaboración de la documentación
Descripción del modelo	Utiliza el modelo de entidad relación. La sintaxis y semántica de los objetos debe estar formalmente definida. Especifica primitivas y operaciones referidas a los objetos definidos.	Establece mecanismos de retroalimentación de la información empleada. El modelo se caracteriza por ser explicito, incremental y continuo. Detalla las propiedades del cambio.	Descripción informal de las entidades y relaciones que integran el esquema de traceability. Se puede adaptar a las necesidades concretas de un proyecto de software	El modelo está influenciado con conceptos de la lingüística. Utiliza notación basada en la teoría de conjuntos Detalla casos de prueba de sus definiciones y la validación de lo propuesto.
Etapas que abarca	Todo el ciclo de desarrollo	Todo el ciclo de desarrollo	Todo el ciclo de desarrollo. (Excluye codificación)	Proceso de RE
Estructura de la información	Kernel Librería de software Capa de aplicación	Infraestructura del cambio Estructura de dependencia Estructura del cambio	Gestión de requerimientos Fundamentación Design/Allocation Compliance Verification	Estructura de Contribución Artefactos Agentes Relaciones de contribución
Tipos de enlaces	Jerárquicos Históricos, Desarrollo	No lo establece	Justificación Evolución Satisfacción Dependencia	Temporales Desarrollo Auxiliares
Criterio detalle	No posee	No posee	CSFs	No posee

Tabla 7 Principales aspectos de los modelos de RT analizados

Los modelos analizados de un modo u otro contemplan la necesidad de describir y almacenar el cambio producido en los distintos elementos que conforman el proceso de desarrollo. Sin embargo, modelos como ESE y Prism especifican con mayor grado de detalle los atributos que describen el cambio producido. El modelo de estructura de contribución puede ser visto como una especialización de los anteriores, ya que especifica cómo han contribuido los distintos stakeholders en la elaboración de los distintos documentos producidos durante el proceso de RE. Este tipo de información (contribución de los stakeholders) no se especifica en el modelo de Ramesh, pero se puede especificar por medio de la definición de atributos tanto en los componentes de información como en las relaciones definidas en dicho modelo.

Un punto importante es el origen del modelo Prism. Al principio fue un proyecto que tenía como objetivo el análisis y comprensión del proceso de desarrollo de software. En trabajos

posteriores se definió un proceso de desarrollo, el cual se detalla en [Madhavji *et. al*, 1991]. Aunque el modelo de Ramesh no esté formalmente definido, se puede inferir del análisis de los componentes de información y de sus relaciones la idea de desarrollo que el autor posee.

El desarrollo de este capítulo centra su atención en la descripción de los diversos medios que permiten implementar la RT. La diversidad de información que aportan las técnicas, métodos y lenguajes que implementan la RT permiten solucionar problemáticas específicas. Es por este motivo, que existe la necesidad de especificar modelos o procesos de desarrollo de software que incorporen a la **RT como hilo conductor** al momento de realizar la especificación de los mismos**.** Estos modelos deben detallar tanto las necesidades de información asociadas a la RT, como así también los mecanismos más adecuados para obtener la información deseada.

CAPÍTULO 5 MODELO RAMESH

El presente capítulo posee el objetivo de describir de manera más exhaustiva la propuesta realizada en [Ramesh *et. al.*, 2001]. En primer término, se analizan los aspectos generales de la investigación conducida por Ramesh y Pohl. Luego se explican los distintos vínculos de traceability que se utilizan en el metamodelo. Por último, se describen los esquemas de traceability propuestos para los distintos usuarios de la RT.

5.1 Aspectos generales de la propuesta de Ramesh

La propuesta de Ramesh y Pohl es el producto de una serie de entrevistas y trabajos grupales. Las personas entrevistadas poseían una vasta experiencia laboral en diversos temas, como por ejemplo, gestión de proyectos, aseguramiento de la calidad, gestión de requerimientos, gestión de la configuración, etc. [Ramesh *et. al.*, 2001]. Tales entrevistas se estructuraron con el propósito de entender las prácticas relacionadas con el uso de RT en las organizaciones. Dichas prácticas se analizaron desde dos perspectivas; la del **usuario de información de traceability** y la del **proveedor de información de traceability** [Ramesh *et. al.*, 2001].

El resultado de esta investigación condujo a la caracterización de dos tipos de usuarios de RT, los **Low - End y High – End.** Además se definieron las distintas categorías de vínculos, como así también los respectivos esquemas de traceability para ambos tipos de usuarios.

5.2 Caracterización de los usuarios Low - End y High - End

Los stakeholders que intervienen en el proyecto de software poseen distintas necesidades de información, las cuales presentan similitudes y diferencias tanto en el uso como en el alcance de la RT. En este apartado se analizan otros factores que inciden en el uso de RT, tanto para los usuarios Low –End como para los High - End**.** El ambiente y la organización, como así también las técnicas que se utilizan para la implementación de traceability constituyen algunos de estos factores.

Característica	Low - End	High - End
Complejidad típica del sistema	Proyectos que deben elaborar alrededor de 1.000 requerimientos.	Proyectos que deben elaborar alrededor de 10.000 requerimientos.
Nivel de experiencia en la RT	Profesionales con 0 - 2 años de experiencia.	Profesionales con 5 – 10 años de experiencia.
Contexto ambiental (tecnologías)	Poseen las tecnologías para la captura y el uso de la información de traceability.	A pesar de que poseen las tecnologías necesarias para la implementación de la RT, desarrollan o adaptan tecnologías para poderlas articular con los objetivos técnicos de la organización.

Contexto organizacional	Se percibe a la RT como una obligación de los patrocinadores del proyecto, como también de los distintos estándares	Se percibe a la RT como un importante mecanismo del sistema de calidad del proceso de desarrollo.
Contexto desarrollo del sistema	Se caracteriza por contar con prácticas y metodología ad hoc. Falta de metodología	Las políticas de desarrollo están apropiadamente especificadas.
Principales aplicaciones	Descomposición de los requerimientos. Allocation de los requerimientos. Compliance Verification. Control de cambios.	Abarca todo el ciclo de vida Captura de información de fundamentación. Captura los traces a través de las dimensiones del producto y el proceso.
Condiciones de Traceability	Utilizan simples esquemas de traceability para modelar las dependencias entre los requerimientos.	Utilizan un esquema en donde se especifican con mayor precisión los distintos traces que integran el modelo. Enfatizan la captura de la información de traceability que facilita el entendimiento del proceso de creación de los distintos documentos y entidades (Rationale).
Adopción y uso de Traceability	Mantienen poca información referida a la Pre- RT. No mantienen un registro detallado de las actividades del proceso por el cual los documentos son confeccionados. Crean documentos estáticos (matrices de traceability). Capturan la información de traceability de manera uniforme a todos los requerimientos.	Mantienen información relacionada a la Pre–RT. Reconocen que los elementos críticos del proceso de desarrollo deben ser traceados. Reconocen la necesidad de mantener información dinámica de traceability que refleje el estado actual del proyecto. Reconocen que no todos los requerimientos son iguales en términos de importancia o criticidad.

Tabla 8 Caracterización de los usuarios Low - End y High – End [Ramesh,1998].

5.3 Vínculos de traceability

[Ramesh *et. al*, 2001] define un conjunto de **vínculos de traceability** entre los distintos componentes de información, los cuales se agrupan en dos categorías, las relacionadas con el **producto** y las relacionadas con el **proceso de desarrollo**. Los vínculos ***satisfacción*** y ***dependencia*** integran la primera categoría***.*** Dichos vínculos se establecen con el propósito de describir las propiedades y las relaciones de los productos, independientemente de cómo se han creado [Ramesh *et. al*, 2001]. En oposición a los vínculos orientados al producto se especifican los vínculos ***evolución y justificación.*** Dichos vínculos se establecen con el propósito de describir las acciones realizadas durante la ejecución del proceso de desarrollo [Ramesh *et. al*, 2001].

Tipo	Objetivos	Usos
Satisfacción	Asegurar que los requerimientos sean satisfechos por el sistema. Relacionar uno o más objetos de diseño, implementación y objetos de requerimientos verificados por los CVP.	Asegurar la consistencia entre las diferentes salidas o resultados del ciclo de vida. Rastrear el sistema/ subsistema/ componentes hacia los requerimientos. Identificar el sistema/subsistema/ componentes que satisface a los requerimientos. Rastrear los CVPs elaborados para un requerimiento.
Dependencia	Ayudar a gestionar las dependencias entre las entidades.	Rastrear la composición y jerarquía de los objetos. Gestionar las repercusiones de los cambios en uno u otros objetos.
Evolución	Documentar las relaciones de E/S de las principales acciones desde los objetos existentes a los nuevos objetos o a los objetos modificados.	Identificar de dónde provienen los objetos. Rastrear la modificación y la historia del refinamiento de los objetos.
Justificación	Representar la justificación de las entidades. Documentar los motivos que originan la evolución de las entidades	Identificar las razones detrás de la creación de las distintas entidades

Tabla 9 Objetivos de los diferentes vínculos de traceability [Ramesh *et. al*, 2001].

5.4 Esquema de Traceability para los usuarios Low - End

El esquema de traceability establecido para los usuarios Low - End se compone de cuatro componentes de información y siete relaciones. Este tipo de usuario define principalmente las dependencias entre los requerimientos especificados, como también entre los componentes de diseño y la identificación de CVPs propuestos para su validación [Ramesh *et. al*, 2001]. En este esquema de traceability se evidencia la falta de riqueza en la definición de sus relaciones y componentes de información. Uno de los puntos que avala tal aseveración es la ausencia de información de justificación. La información que produce se utiliza generalmente para elaborar la matriz de traceability.

Vínculo	Componentes de información
derive	Requirements
developed for	Compliance_verification_Procedures
satisfy	System_Subsystem_Components
allocated_to	External_Systems
performed_on	
depend_on	
interface_with	

Tabla 10 Vínculos y Componentes de Información

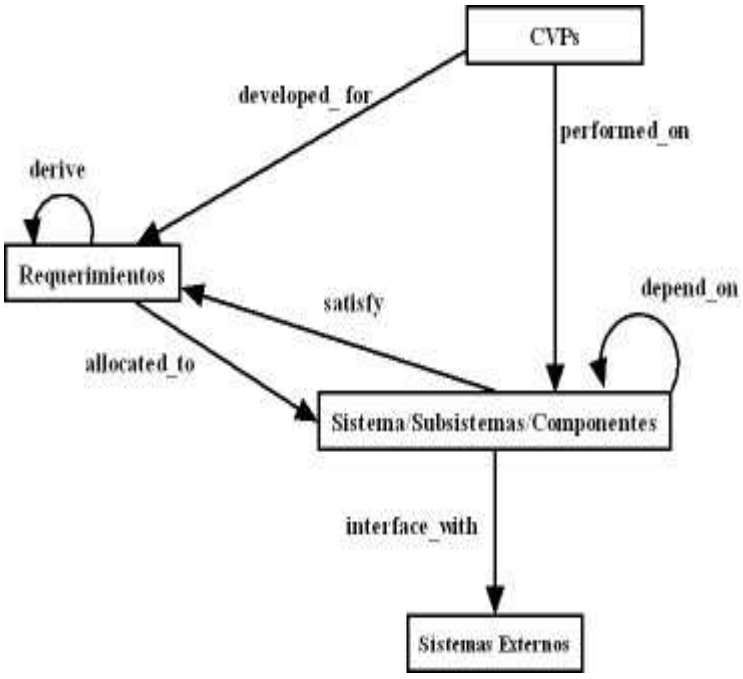


Figura 5.1 Esquema de Traceability para los usuarios Low End [Ramesh *et. al*, 2001].

5.5 Esquema de Traceability para los usuarios High End

El esquema de traceability propuesto para los usuarios High - End se descompone en cuatros submodelos: **Gestión de los Requerimientos, Fundamentación, Design/Allocation y Compliance Verification**. En estos submodelos se especifican entre cincuenta a sesenta componentes de información y sus respectivas relaciones.

5.5.1 Submodelo Gestión de los Requerimientos

En el **submodelo Gestión de los Requerimientos** se especifican los elementos necesarios para la gestión de los requerimientos de software. Siguiendo este lineamiento, los vínculos de traceability descriptos se orientan a reflejar la captura, entendimiento y seguimiento de los requerimientos.

El primer componente de información que se especifica son las **Necesidades Organizacionales,** las cuales representan el propósito principal del sistema a construir. Las necesidades se clasifican en dos tipos, las **Necesidades Estratégicas** (representan necesidades a largo plazo) y las **Necesidades Operacionales** (representan necesidades inmediatas). Esta tipificación se especifica mediante el uso del vínculo ***is_a***. Dichas necesidades se relacionan con los **CSFs, Escenarios y Objetivos del Sistema** [Ramesh *et. al,* 2001].

Los **Factores Críticos de Éxito - CSFs -** representan todos aquellos recursos identificados como críticos para el éxito del proyecto. Estos factores constituyen el principal mecanismo para clasificar y monitorear a los requerimientos [Ramesh et. el, 2001]. Los vínculos existentes entre las **Necesidades Organizacionales** y los **CSFs** se detallan a través del vínculo ***identify***. Los **Escenarios** son utilizados para modelizar las **Necesidades Organizacionales** y los **Objetivos del Sistema**, como así también a los **Requerimientos**. Tal relación se establece mediante el vínculo ***describe***.

Una vez finalizada la identificación de las necesidades con sus respectivos **Escenarios y CSFs,** se determinan los **Objetivos del Sistema.** Dichos objetivos representan el conjunto de metas que el sistema a desarrollar debe lograr. Un aspecto importante de los **Objetivos del Sistema** es que deben estar debidamente justificados por las **Necesidades Organizacionales**. La relación entre estos dos componentes se detalla mediante el uso del vínculo ***justify***.

El componente central de este submodelo es representado por los **Requerimientos.** Este componente posee la información que facilite a los distintos stakeholders entender, describir, y documentar la evolución de los mismos. Para tal fin, se dispone de diversos vínculos de traceability. El primero de estos vínculos, ***generate*** define la relación existente entre un **Objetivo del Sistema** y un **Requerimiento de Software.** Este requerimiento posee un elevado nivel de abstracción, por la cual es necesario poner en marcha actividades con el propósito de obtener requerimientos con un adecuado nivel de detalle. Estas actividades se detallan utilizando el vínculo ***derive.*** El vínculo ***elaborate*** representa el proceso mediante el cual un requerimiento puede ser elaborado por otro, favoreciendo de esta manera su explicación. También se utiliza el vínculo ***modify*** para reflejar las modificaciones realizadas a partir de lo señalado en las **Propuestas de Cambio.** No sólo es importante entender el cómo y cuándo se obtienen los requerimientos sino también es necesario describir las

dependencias entre ellos. Tales dependencias se detallan con los vínculos de traceability ***depend_on, is_a*** **y** ***part_of*****.** Este último vínculo se emplea cuando se poseen requerimientos complejos, los cuales deben ser descompuestos en requerimientos más simples para su mejor tratamiento o desarrollo.

En este submodelo también se especifican cuáles son los **Mandates** (**Estándares, Políticas y Procedimientos**) que se deben utilizar para la correcta elaboración de los requerimientos identificados. Esta información se detalla utilizando el vínculo ***based_on***. Un aspecto que caracteriza a este esquema de gestión de los requerimientos es que las **Restricciones** son tratadas como requerimientos. Se utiliza el vínculo ***is_a*** para detallar la relación entre estos componentes de información.

El nivel de detalle con que se gestionan los requerimientos debe estar determinado según como se vinculen con los **CSFs**. Se utiliza el vínculo *managed_by* para relacionar a los CSFs con los requerimientos.

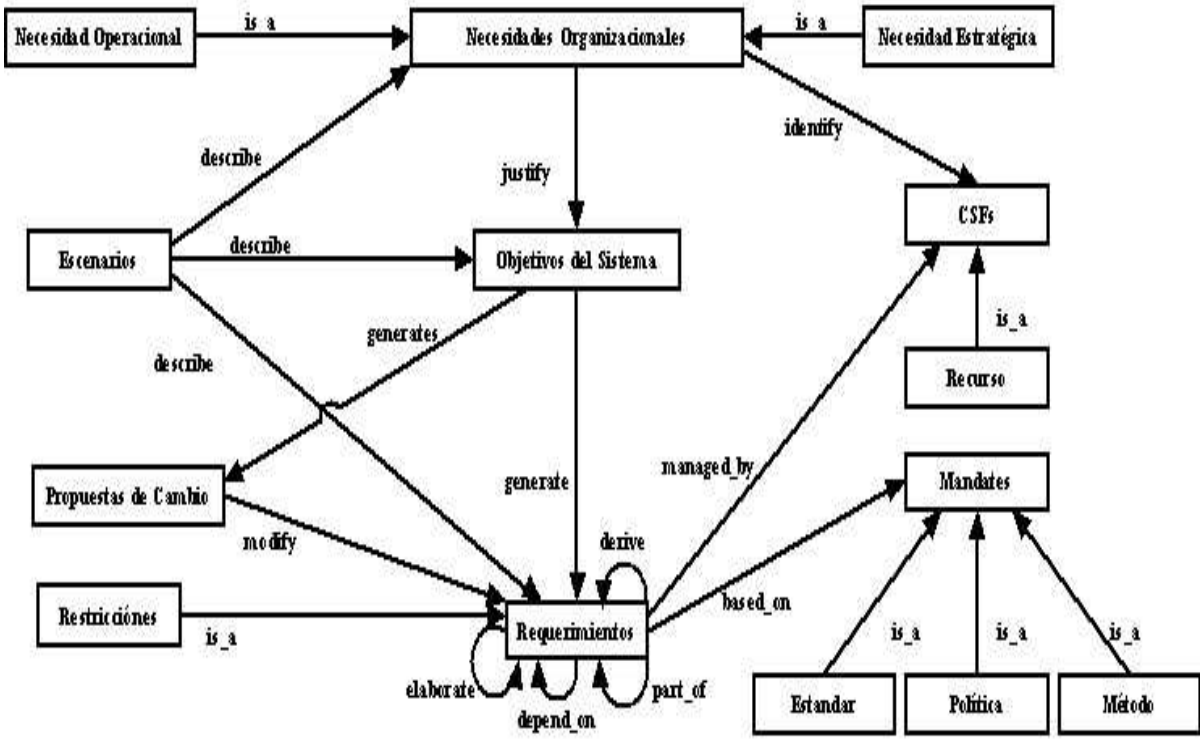


Figura 5.2 Submodelo Gestión de los Requerimientos [Ramesh *et. al*, 2001].

Vínculo	Tipo de Vínculo
based_on	Evolution
depend_on	Dependency
derive	Evolution
describe	Evolution
elaborate	Evolution
generate	Evolution
identify	Evolution
is_a	Dependency
justify	Evolution
managed_by	Rationale
modify	Evolution
part_of	Dependency

Tabla 11 Vínculos del Submodelo Gestión de los Requerimientos

5.5.3 Submodelo Fundamentación

El propósito del **submodelo Fundamentación** radica en mantener la información relacionada con la resolución de las distintas **situaciones problemáticas** (especificadas en los componentes de información **Situación Problemática o Conflicto**) originadas durante el desarrollo.

El primer componente que se describe de este submodelo se denomina **Objeto,** el cual representa cualquier elemento (requerimiento, elemento de diseño, etc.) elaborado en el transcurso del proceso de desarrollo. Tal relación se establece con el vínculo ***is_A****.*

Durante la confección de los distintos **Objetos** se pueden originar distintas **Situaciones Problemáticas o Conflictos** que deben ser resueltas**.** Esta relación se especifica a través del vínculo ***generate****.* Las actividades que intervienen en la resolución del problema, involucran el tratamiento y almacenamiento de distintos componentes de información y vínculos de traceability. Por ejemplo, la información que detalla el curso de acción seleccionado se especifica en el componente de información **Decisión** y se relaciona con la situación problemática mediante el vínculo ***resolve****.* Hay que tener en cuenta que la decisión que se ha tomado puede afectar a determinados requerimientos de software. Es por este motivo, que se recurre al vínculo ***affect*** para describir tal situación. La decisión seleccionada puede estar influenciada por los **CSFs** identificados en la organización. Esta situación se describe por medio del vínculo ***influence***.

El proceso mediante el cual se toma una decisión se caracteriza por el análisis y evaluación de distintos cursos de acción. La información que interviene en la evaluación se especifica en el componente **Alternativa** y se relaciona con la situación problemática mediante los vínculos de traceability ***evaluate y select****.* Cada una de las **Alternativas** especificadas deben estar descriptas por el componente **Argumentos** y se relacionan con las **Alternativas** mediante los vínculos ***oppose y support***. A su vez, los **Argumentos** pueden estar relacionados con distintos supuestos, los cuales se detallan en el componente **Asunción** a través del vínculo ***depend_on****.* Por último, la decisión que se ha efectuado debe estar debidamente fundamentada. Tal fundamentación se registra en el componente **Fundamentación,** y se relaciona con el componente que describe la decisión mediante el vínculo ***based_on****.*

Es importante destacar que no todos los **Objetos** del proceso del desarrollo originan una **Situación Problemática.** En determinadas ocasiones se debe poder especificar el motivo o supuesto que se posee al momento de elaborar determinados **Objetos**. Esta relación se detalla mediante el vínculo ***depend_on*** y relaciona a los **Objetos** con sus **Asunciones.**

Vínculo	Tipo de Vínculo
address	Rationale
affect	Evolution
based_on	Rationale
depend_on	Rationale
depends_on	Rationale
evaluate	Rationale
generate	Rationale
influence	Rationale
oppose	Rationale
resolve	Rationale
select	Rationale
support	Rationale
is_a	Dependency

Tabla 12 Vínculos del Submodelo Fundamentación

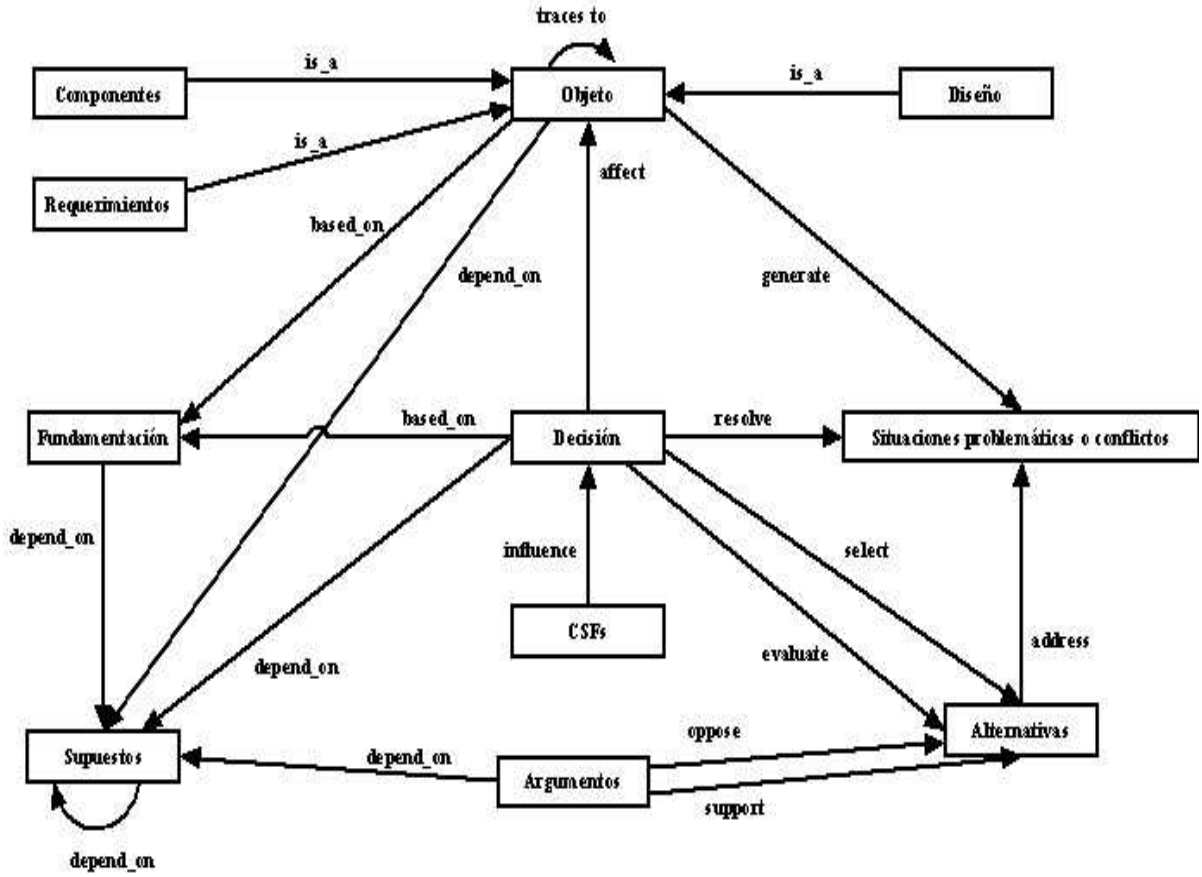


Figura 5.3 Submodelo Fundamentación [Ramesh *et. al*, 2001].

5.5.3 Submodelo Design/Allocation

El submodelo de **Design/Allocation** establece las entidades y relaciones que integran el proceso de diseño. Además se define al proceso de diseño como *"cualquier actividad que crea artefactos*[35]*, incluyendo la implementación"*[36]*.* Sin embargo, tales descripciones

[35] Un artefacto es cualquier información usada o producida por el proceso de desarrollo de software. Unified Modeling Language Specification. UML 1.4 with Action Semantics, Final Adopted Specification, enero 2002.http://www.omg.org.
[36] Ramesh, Bala. Pohl, Klaus. "*Toward Reference Models for Requirements Traceability*" .En: IEEE Transactions on Software Engineering. Vol.27.Nro. January 2001.Pág 73

presentan ciertas dificultades como ser dispone de escasa información relacionada con definición de los componentes de información y de sus relaciones. Esta situación ya fue advertida en otros trabajos, tales como [Piattini, *et al.*, 2003]. Pero, cabe destacar, que en [Ramesh *et. al.*, 2001] se ha especificado un metamodelo, y los autores del mismo, no tienen la obligación de describir su trabajo con el grado de detalle necesario para poder especificar de manera completa y detallada al proceso de desarrollo de software.

Este submodelo propone la confección de la entidad **Diseño**. En dicha entidad se detallan por medio del vínculo ***drive*** los requerimientos que guían las actividades a realizar. Estas actividades involucran la definición (detallado por medio del vínculo ***define***) y creación (detallado por medio del vínculo ***create***) de los **Sistemas, Subsistemas o Componentes** que se especifican en sus respetivos componentes de información. Estas actividades pueden estar basadas en **Mandates.** Las relaciones entre el componente de información que describe al diseño y a los distintos **Mandates** se especifican mediante el vínculo ***based_on****.*

Los vínculos ***part_of y depend_on*** se utilizan para describir la composición y las dependencias entre los distintos componentes que se han creado o definido. Cuando un determinado componente elaborado en el proceso de diseño se relaciona con el sistema externo (descripto por el componente de información **Sistema Externo**) se debe especificar tal relación con el vinculo ***depend_on***. Es de gran utilidad conocer cuáles son los recursos que se utilizan en los distintos componentes de diseño. Esta información se establece mediante el vínculo ***used_by***. El proceso de diseño también involucra el mapeo de los requerimientos de software que se han especificado en los componentes de diseño. Tal actividad se especifica mediante el vínculo ***allocated***. La información que describe las dependencias, composición y allocation de los requerimientos en los componentes de diseño son de gran utilidad al momento de determinar el impacto del cambio de los requerimientos.

Vínculo	Tipo de Vínculo
address	Satisfaction
allocate_to	Satisfaction
based_on	Evolution
create	Evolution
define	Evolution
depends_on	Dependency
depend_on	Dependency
drive	Satisfaction
Is a	Dependency
modify	Evolution
part_of	Dependency
perform	Satisfaction
satisfy	Satisfaction
used_by	Rationale

Tabla 13 Vínculos del Submodelo Design/Allocation

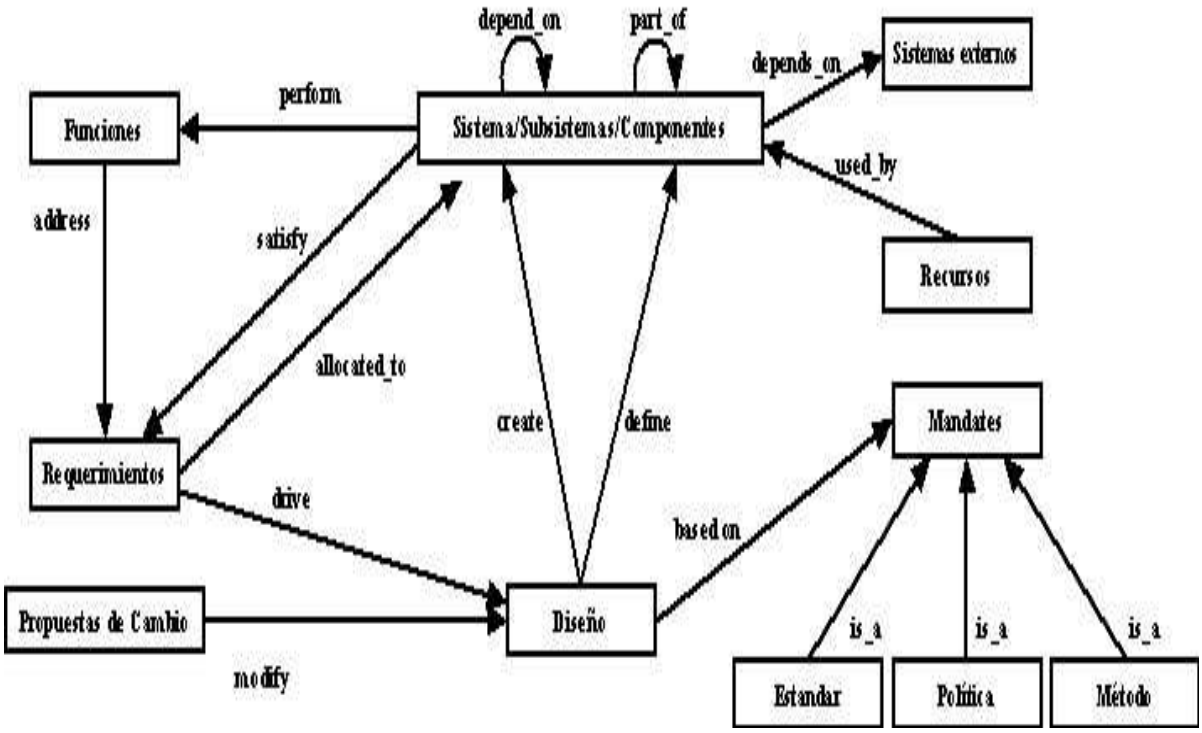


Figura 5.4 Submodelo Design/Allocation[Ramesh *et. al*, 2001].

5.5.4 Submodelo Compliance Verification

Las actividades de validación se encuentran enmarcadas dentro de los siguientes propósitos: contribuir en el chequeo de los requerimientos para detectar la omisión, conflictos o ambigüedades en su especificación [Sommerville *et. al*, 1997], como así también, la implementación de los distintos mecanismos que le aseguran al cliente obtener el producto que ha solicitado [Sommerville, 1995]. En la figura 5.4 se detallan los componentes de información y las relaciones que integran el proceso de validación, los cuales están enmarcados dentro del **submodelo de Compliance Verification**. En este submodelo se establece que "*una variedad de CVP son desarrollados para asegurar que cada uno de los requerimientos estén adecuadamente direccionados".*[37] De tal expresión, se puede afirmar que las actividades propuestas en el submodelo de Compliance Verification se orientan a asegurar que los requerimientos identificados sean los solicitados por los usuarios más que la verificación de los mismos.

El componente de información que caracteriza a este submodelo se denomina **Procedimiento de Verificación de la Conformidad o CVP**. Este componente puede ser elaborado mediante diversas técnicas o métodos. En este submodelo se delimita la manera de llevar a cabo tal validación mediante el uso de cuatro técnicas: **Prototipo, Simulación, Test o Inspección.** Cabe recordar que se debe utilizar el vínculo de traceability ***is_a*** para especificar la técnica utilizada al confeccionar el **CVP**. La selección de la o las técnicas

[37] Ramesh, Bala. Pohl, Klaus. Op. cit 2001.Pág 74

utilizadas para llevar a cabo la validación puede depender de los recursos que posee la organización. Es por tal motivo que se deben detallar mediante el vínculo ***used_by*** los recursos que utiliza el **CVP** elaborado. Esta dependencia genera en algunas ocasiones situaciones problemáticas, debido a que el CVP elaborado puede llegar a utilizar algún recurso crítico, o recursos que no estén disponibles.

Como resultado del desarrollo de las actividades de validación, se confeccionan las **Propuestas de Cambio** (descriptas por el vínculo ***generate***). Estas propuestas representan uno de los elementos de entrada necesarias tanto para la gestión de requerimientos, como para el proceso de diseño. Tras haber culminado favorablemente con la validación del diseño, se detallan los vínculos ***verify_by***. Este vínculo tiene como propósito relacionar al **CVP** con la relación ***satisfy*** *(*esta relación vincula un componente de diseño con un requerimiento de software) establecida en el submodelo de Design/Allocation.

Vínculo	Tipo de Vínculo
based_on	Evolution
developed_for	Satisfaction
generate	Evolution
is_a	Dependency
satisfy	Satisfaction
used_by	Rationale
verify by	Satisfaction
used_by	Rationale

Tabla 14 Vínculos del Submodelo Compliance Verification

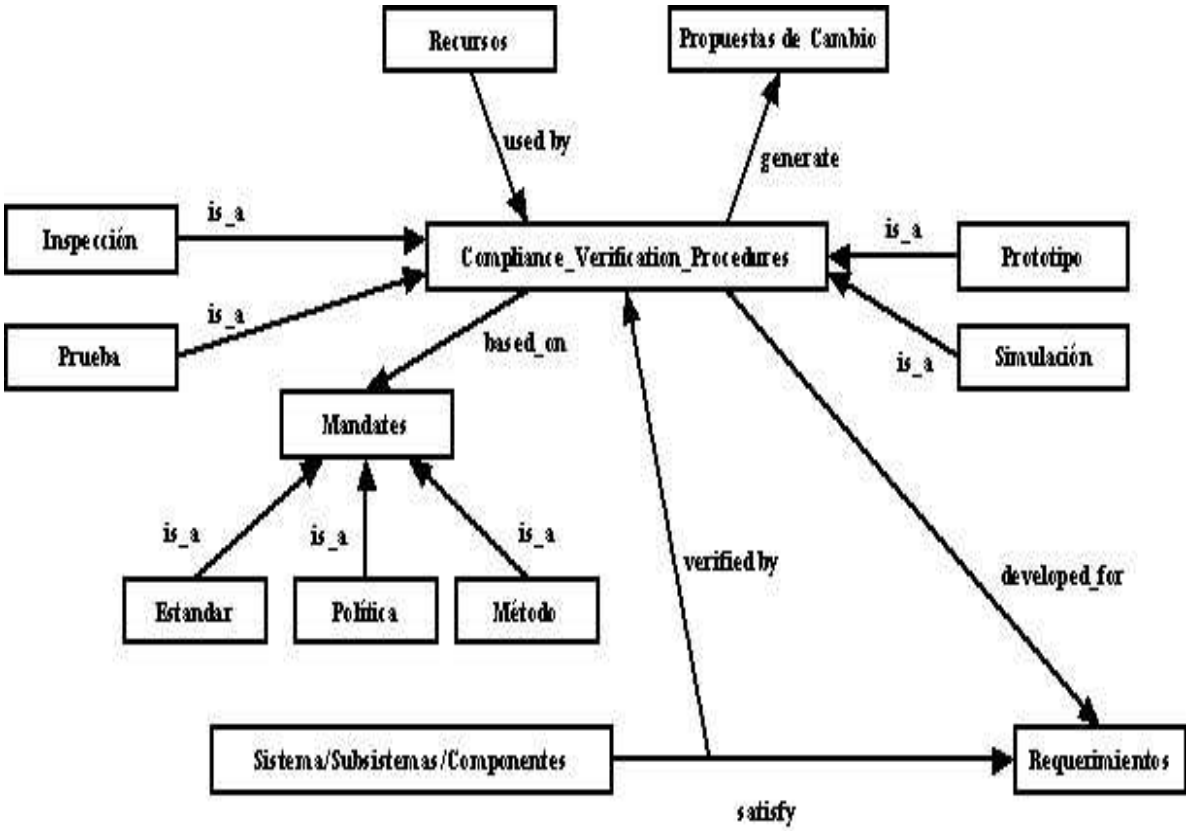


Figura 5.5 Submodelo Compliance Verification[Ramesh *et. al*, 2001].

5.6 Algunas consideraciones

Aunque el esquema de traceability definido para los usuarios High End presenta una extensa especificación de entidades y tipos de enlaces, en determinados aspectos no brinda la suficiente información para poder definir de manera concreta y detallada al proceso de desarrollo. A continuación se señalan algunos de los aspectos contradictorios o deficientes que se han advertido referidos al esquema de traceability para los usuarios High End.

Las **entidades** que se especifican no se describen de manera formal, sino que se representan en esquemas, conjuntamente con sus relaciones. Estas definiciones no incluyen ninguna mención sobre qué tipo de información o atributos deben poseer. Aunque este hecho no represente una restricción para la especificación del proceso de desarrollo, en determinadas entidades, no queda claro qué tipo de información se debe mantener. Este es el caso de la entidad **Diseño.** Dado que el modelo no establece qué información comprende, no se puede afirmar si esta entidad especifica las diferentes alternativas de implementación de diseño o representa las distintas baseline de los productos de trabajo confeccionados en las actividades de diseño. Esta situación se traslada a otras entidades como ser **Sistema, Subsistema y Componentes.** En el submodelo **Design/Allocation**, no se describen las actividades que se deben llevar a cabo para la elaboración de los distintos producto de trabajo (como ser DER, DTE, diagrama de secuencia, etc.). Tampoco se hace mención a la selección y adopción de alguna metodología en especial (estructurado, objeto).

Siguiendo con la problemática referidas a las entidades, se observa cierta ambigüedad en algunas definiciones. Cuando se describe el esquema gestión de requerimientos se detalla que las **Necesidades Organizacionales** justifican a los **Objetivos del Sistema.** En cambio, en el caso práctico descripto en [Ramesh *et. al.*, 2001] se especifica que el componente **Necesidades de Misión** (componente no definido en el esquema) justifica a los **Objetivos del Sistema.** Más aún, en dicho ejemplo, se define como objetivos del sistema a lo que anteriormente se ha definido como necesidades organizacionales.

Tampoco queda bien en claro cómo se deben elaborar los distintos productos de trabajo. Se plantean interrogantes tales como la manera en que debe confeccionarse la SRS o el documento que detalla al diseño elaborado.

Con respecto a la especificación de las relaciones entre las entidades se presenta una seria omisión; la definición de los atributos y de la respectiva cardinalidad de la relación. La cardinalidad de las relaciones puede ser de gran utilidad al momento de especificar las reglas o restricciones del proceso de desarrollo. Se ha establecido que a través de las **Necesidades Organizacionales** se debe identificar a los **CSFs;** pero lo que no se aclara es si hay una correspondencia de uno a uno, de uno a muchos o, si llegado el caso, si ciertas necesidades no poseen **CSFs.** También se nota cierta redundancia en la definición de ciertos vínculos. En el submodelo **Gestión Requerimientos** la relación ***is_a*** para determinar

la jerarquía de los requerimientos puede ser descripta por la relación ***part_of***. También, al no definir las diferencias entre las relaciones ***create y define***, se llega a la conclusión de que no es necesario utilizar ambas relaciones.

5.7 Justificación del esquema de traceability seleccionado

A continuación se exponen los diversos motivos que se tuvieron en cuenta al momento de la elección del metamodelo para constituirse como marco de referencia en la confección de la especificación del proceso de desarrollo de software.

El principal aspecto es la presencia de la entidad **CSF.** Esta entidad representa el mecanismo mediante el cual se permite decidir cuando es esencial tracear, lo que permite establecer el tipo y grado de detalle de la información referida a la RT necesaria en la gestión de los requerimientos [Ramesh *et al*., 2001]. El vincular los requerimientos con los **CSFs** permite poseer una herramienta útil en el proceso de negociación con los stakeholders al determinar los alcances de los cambios y funcionalidad del sistema dependiendo del impacto sobre los **CSF** [Ramesh *et al*., 2001].

Otro motivo que se tuvo en cuenta al seleccionar el metamodelo es la definición de las entidades y relaciones definidas en el submodelo **Gestión de los Requerimientos**. Dichas definiciones permiten seguir la evolución del requerimiento desde sus orígenes hasta su verificación, centrándose en los elementos críticos que afectan al sistema. Además se consideró importante que la definición del submodelo **Fundamentación** establezca los elementos básicos para llevar a cabo tal actividad, independientemente de la envergadura del sistema a implementar.

Por último, dicho esquema no imposibilita que los distintos elementos definidos puedan ser adaptados según el contexto de desarrollo que se posea (la adaptación es fundamental para el éxito de la implementación de los distintos modelos de RT).

Aunque en el apartado anterior se han señalado ciertos problemas que presenta el esquema de traceability seleccionado, estos hechos no constituyen un impedimento para la elección del mismo.

El presente capítulo tuvo como propósito describir con mayor profundidad el metamodelo propuesto por Ramesh, especialmente el esquema definido para los usuarios **High End.** Como se ha advertido, en algunas situaciones los esquemas propuestos no presentan la suficiente información, pero pese a ello se considera que las ausencias de las mismas pueden ser subsanadas.

CAPÍTULO 6 ESPECIFICACIÓN DEL PROCESO DE DESARROLLO

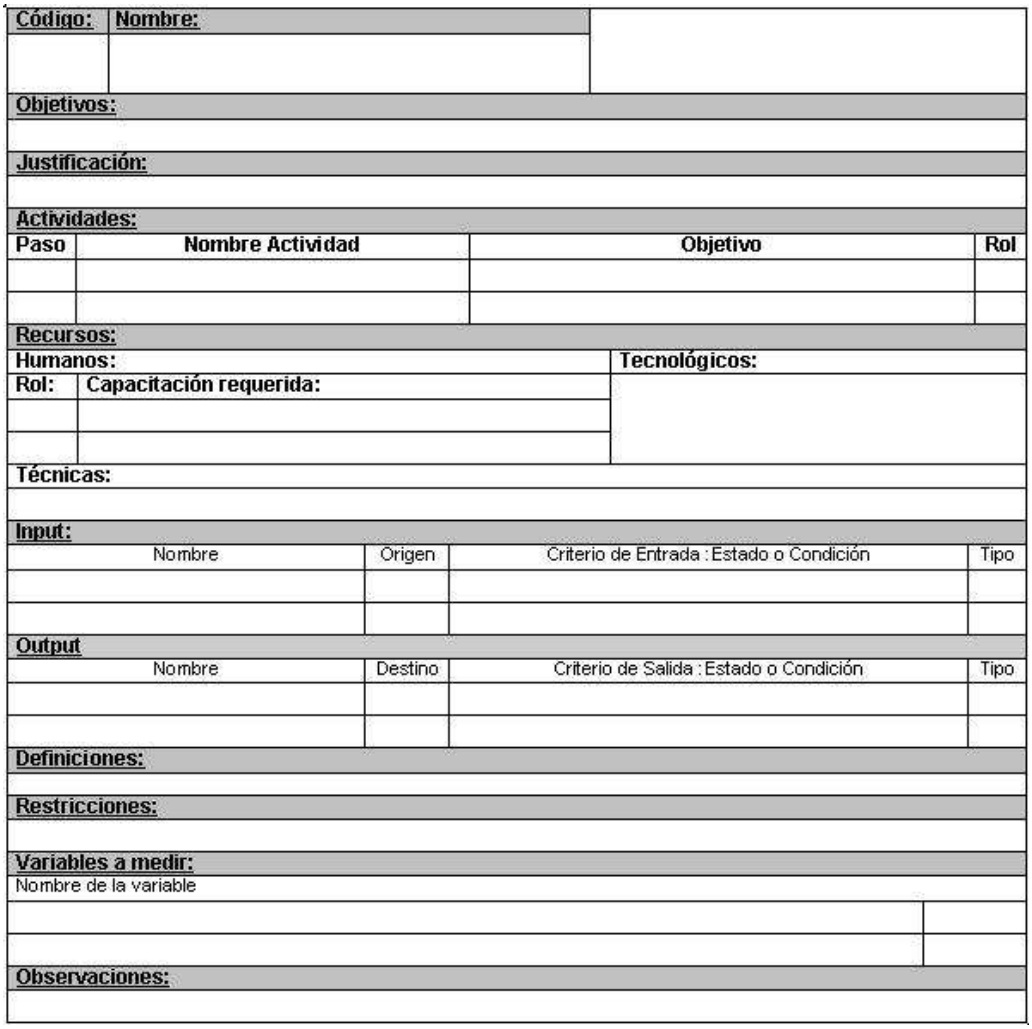
En este capítulo se describe la especificación del proceso de desarrollo de software teniendo como marco de referencia al metamodelo descripto en [Ramesh *et. al.*, 2001].

6.1 Introducción

La importancia de tener como referencia a un metamodelo radica en poder seleccionar sus partes más relevantes. Además, se podrán reducir las alternativas referidas a la especificación de las actividades, entidades y documentos a elaborar [Ramesh *et al*, 2001]. Con el fin de estandarizar la documentación referida a la especificación del proceso de desarrollo de software, se elaboró un **template** siguiendo los lineamientos definidos en **Software Process Framework (SPF)** [Olson *et al*., 1993]. Los componentes que integran el template son:

- **Código:** Es el número con que se identifica a la actividad dentro del proceso de desarrollo. Para identificar una actividad se empleó la siguiente convención:
  - El formato **#.00** se utiliza para identificar una actividad raíz, como ser **1.00 Modelizar la Organización.**
  - El formato **#.0#** se utiliza para identificar las actividades que integran una actividad raíz, como ser **1.02 Desarrollar las Necesidades Organizacionales**
  - El formato **#.##.##** se utiliza para identificar las subactividades, como ser **1.02.02 Identificar los CSFs de la organización**. Al especificar las subactividades se deben identificar las acciones que deben realizarse, las cuales se describen en el componente **técnicas**.
- **Nombre:** Este componente detalla el nombre de la actividad que se está especificando. Se utiliza la forma no personal del verbo para nombrar a las actividades.
- **Referencias:** Se utiliza dicho componente para ubicar la actividad especificada con respecto a las restantes actividades de su misma jerarquía. Aunque este componente no está definido en Software Process Framework se lo agregó al template debido a que brinda legibilidad y favorece al entendimiento de la especificación.
- **Objetivos**: En este componente se detallan las metas que deben alcanzarse al realizar la actividad que se está especificando. Como ser **Determinar el propósito del sistema a construir**.
- **Justificación:** En este componente se detallan las razones por las cuales se tiene que realizar la actividad que se está especificando.

- **Actividades**: Este componente tiene como propósito detallar las acciones que deben realizarse para llevar a cabo la actividad que se está especificando. Para ello se debe completar la siguiente información:
  - *Paso:* Es el orden en que se debe realizar la actividad.
  - *Nombre Actividad:* Nombre que identifica a la actividad.
  - *Objetivo*: En este componente se detallan las metas que deben alcanzarse al realizar la actividad que se está especificando.
  - *Rol:* Se detallan quién o quiénes son los responsables de realizar la actividad.
- **Recursos:** Este componente tiene como propósito describir los recursos humanos, tecnológicos y las técnicas que se emplean en la realización de la actividad que se está especificando. Al especificar los recursos **humanos** sólo se detalla el rol y la capacitación requerida (conocimientos y habilidades necesarias para realizar la actividad). Al definir los recursos **tecnológicos** se detallan las **herramientas** con las que se debería contar para realizar la actividad. Al no contar con una herramienta CASE que gestione las actividades especificadas en el proceso, en este componente se detallan las herramientas que pueden sustituirla. Por último, se detallan como recursos las **técnicas** que deben aplicarse para realizar la actividad que se está especificando.
- **Input:** Estos componentes se utilizan para detallar la información, conocimiento o productos de trabajo (*work product*) que se necesitan para realizar la actividad que se está especificando. Además se detalla el origen de las diversas entradas como así también los criterios, estados o condiciones que deben poseer.
- **Output**: Estos componentes se utilizan para detallar la información, conocimiento o productos de trabajo (*work product*) que se obtienen al realizar la actividad que se está especificando. Además se detalla el destino de estas salidas, como así también los criterios, estados o condiciones que deben poseer.
- **Definiciones**: Este componente se emplea para plasmar la definición de los principales conceptos utilizados por el metamodelo seleccionado.
- **Restricciones:** Este componente se utiliza para detallar las limitaciones que se poseen al realizar la actividad que se está especificando.
- **Variables a medir:** Este componente se utiliza para detallar las distintas métricas especificadas en el plan de mediciones del proyecto de software. Este plan debe estar enmarcado por el conjunto de necesidades de información específicas de cada proyecto particular. Es por este motivo que se ha optado por especificar sólo las variables a medir comunes a todos los proyectos de software.
- **Observaciones:** Este componente se utiliza para indicar los comentarios o aclaraciones que se consideren necesarias para realizar la actividad que se está especificando.



Código:	**Nombre:**	

Objetivos:

Justificación:

Actividades:			
Paso	**Nombre Actividad**	**Objetivo**	**Rol**

Recursos:		
Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	

Técnicas:

Input:			
Nombre	Origen	Criterio de Entrada : Estado o Condición	Tipo

Output			
Nombre	Destino	Criterio de Salida : Estado o Condición	Tipo

Definiciones:

Restricciones:

Variables a medir:	
Nombre de la variable	

Observaciones:

Figura 6.1 Template utilizado para especificar el proceso de desarrollo

6.2 Especificación del proceso de desarrollo de software

El proceso de desarrollo propuesto representa una instancia del metamodelo descripto en [Ramesh *et al*, 2001]. Es por ello que las actividades propuestas están enmarcadas dentro de los cuatro submodelos definidos en [Ramesh *et al*, 2001]:

1. Modelizar la Organización
2. Especificar los Requerimientos de Software
3. Validar la Especificación de Requerimientos de Software
4. Desarrollar la Arquitectura del Sistema

El proceso de desarrollo de software comienza con el análisis de la organización y del contexto en el cual se debe implementar el sistema de software. Al finalizar esta actividad, se tienen identificadas y modeladas las **Necesidades Organizacionales,** como así también,

los **Objetivos del Sistema**. Estos objetivos representan el punto de partida por el cual se identifican los **Requerimientos de Software.** A partir de este momento se pone en marcha la actividad **Especificar los Requerimientos de Software.** Cabe destacar que esta actividad no sólo involucra la identificación de los requerimientos, sino también el análisis, modelización, documentación y gestión de los mismos. Como resultado de estas acciones se obtiene el documento **Especificar los Requerimientos de Software,** el cual debe estar debidamente validado por los miembros de la organización. Finalizada la validación de dicho documento comienza la actividad **Desarrollar la Arquitectura del Sistema**. En esta actividad se agrupan a los requerimientos identificados en **Sistemas, Subsistemas, Módulos o Componentes**. Al finalizar esta actividad se elabora el **Documento de Arquitectura del Sistema** el cual cuenta con la respectiva información de asignación de los requerimientos.

Durante la realización del proceso de desarrollo de software puede ser necesario llevar a cabo la actividad de soporte **Fundamentar.** En dicha actividad se especifican las acciones que deben realizarse para resolver o aclarar las **situaciones conflictivas** que se generan, como así también las que permiten justificar lo realizado.

Aunque se han especificado las actividades de manera secuencial, no significa que se haya adoptado el ciclo de vida cascada, sino todo lo contrario. Se considera al ciclo de vida iterativo e incremental como modelo primordial para el desarrollo de software. Sin embargo no se ha especificado el número de iteraciones a realizar, debido a que depende en gran medida del tipo y características de las necesidades a implementar.

Es importante recalcar que quedan excluidas de la especificación del proceso de desarrollo de software tanto las actividades involucradas en la confección del diseño detallado, como así también las concernientes a la codificación del sistema. El proceso de desarrollo sólo contempla la implementación de los mecanismos de traceability que involucra la dirección **forwards** y **backwards** de la RT catalogada como **vertical.**

En la Tabla 15 se detallan las actividades que integran el proceso de desarrollo de software. Dependiendo de la actividad, se confecciona el respectivo DFD que la describe. Estos diagramas tienen el propósito de detallar las relaciones existentes entre las actividades con su respectivo flujo de información (conocimiento, información o productos de trabajo). La notación empleada en la confección de los DFDs es la siguiente:

- Las elipses representan las distintas actividades.
- Las flechas representan el flujo de información entre las actividades especificadas.
- Los rectángulos representan tanto las entidades como actividades externas al proceso de desarrollo.

Actividad		Representada en
1.0	Modelización de la Organización	Figura 6.3
1.01	Analizar la organización y su contexto	
1.02	Desarrollar las Necesidades Organizacionales	Figura 6.4
1.02.01	Confeccionar las Necesidades Organizacionales	
1.02.02	Identificar los CSFs de la organización	
1.02.03	Describir las Necesidades Organizacionales	
1.02.04	Validar las Necesidades Organizacionales	
1.03	Desarrollar los Objetivos del Sistema	Figura 6.5
1.03.01	Confeccionar los Objetivos del Sistema	
1.03.02	Describir los Objetivos del Sistema	
1.03.03	Validar los Objetivos del Sistema	
2.00	Especificación de los Requerimientos de Software	Figura 6.6
2.01	Determinar los Requerimientos de Software	Figura 6.7
2.01.01	Recolectar información del dominio del problema	
2.01.02	Identificar los Requerimientos de Software	
2.01.03	Establecer los vínculos con los CSFs	
2.01.04	Describir los Requerimientos de Software	
2.02	Reformar los Requerimientos de Software	Figura 6.8
2.02.01	Redefinir los Requerimientos de Software	
2.02.02	Modificar los Requerimientos de Software	
2.03	Documentar los Requerimientos de Software	
3.00	Validación de la Especificación de Requerimientos de Software	Figura 6.9
3.01	Determinar los Requerimientos de Software a validar	
3.02	Confeccionar los CVPs	
3.03	Validar los requerimientos seleccionados	
4.00	Desarrollo de la Arquitectura del Sistema	Figura 6.10
4.01	Seleccionar los Requerimientos de Software a diseñar	
4.02	Establecer la estructura del sistema	Figura 6.11
4.02.01	Definir el Sistema a desarrollar	
4.02.02	Confirmar la implementación del Sistema	
4.03	Establecer la estructura de los Subsistemas	Figura 6.12
4.03.01	Definir la estructura de los Subsistemas	
4.03.02	Confirmar la implementación de los Subsistemas	
4.04	Establecer la estructura de los Componentes	Figura 6.13
4.04.01	Definir los Componentes a desarrollar	
4.04.02	Confirmar la implementación de los Componentes	
4.05	Documentar la Arquitectura del Sistema	

Tabla 15 Actividades proceso de desarrollo de software

AC	Actividades codificación
AE	Actividades externas
AP	Actividades posteriores
CA	Cualquier actividad
CON	Conocimiento
DI	Diseñador
ER	Especialista en requerimientos
GS	Gerencia superior
INF	Información
LP	Líder del proyecto
OR	Organización
PDT	Producto de trabajo - Work product -
SRS	Especificación de Requerimientos de Software
TE	Tester
UO	Usuario operativo

Tabla 16 Abreviaturas utilizadas en la especificación

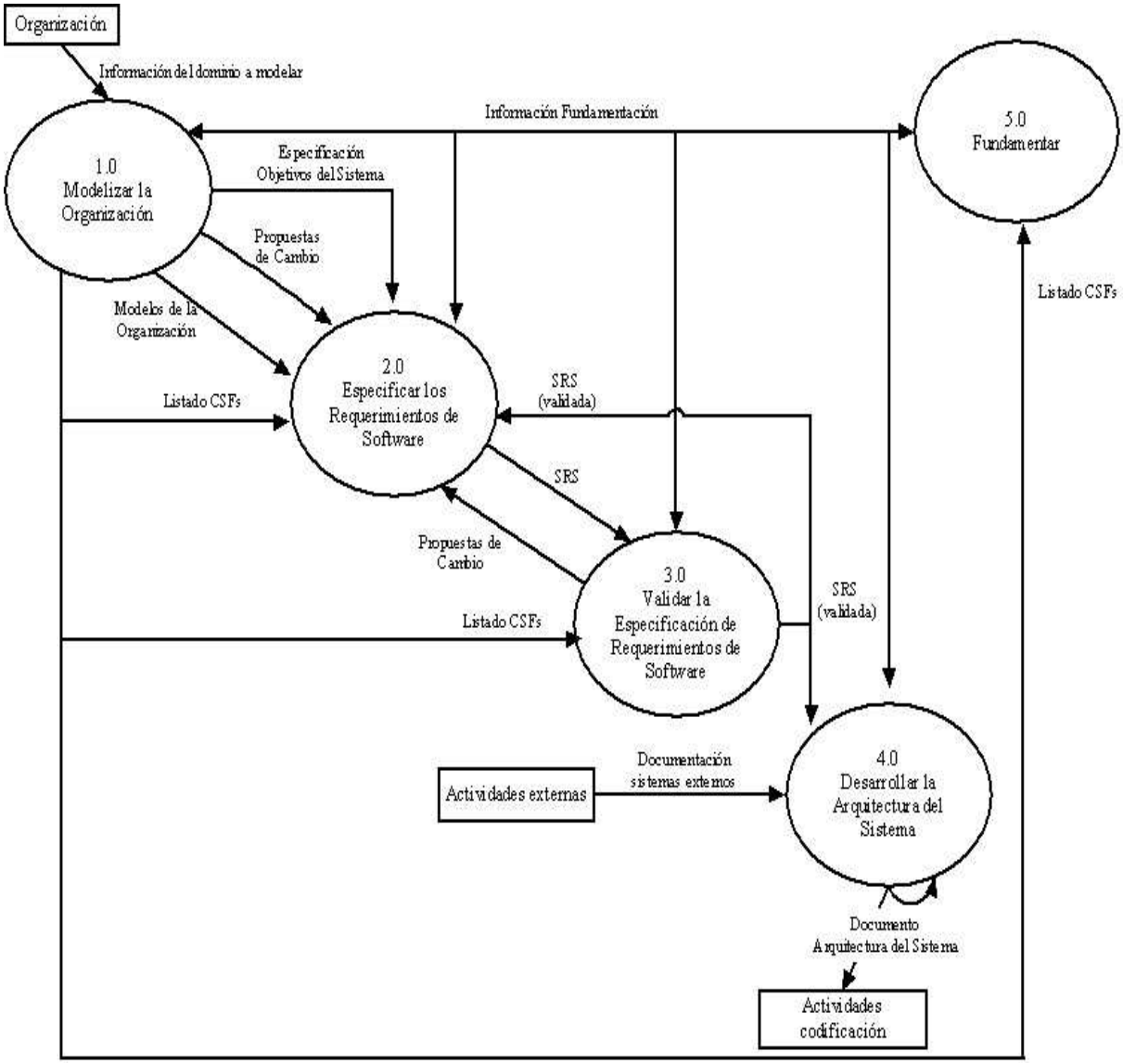


Figura 6.2 Proceso de Desarrollo de Software – Vista General[38]

[38] Con el propósito de simplificar la representación los documentos **Especificación Conflicto** y **Especificación Fundamentación** se definen por medio de la salida **Información Fundamentación**

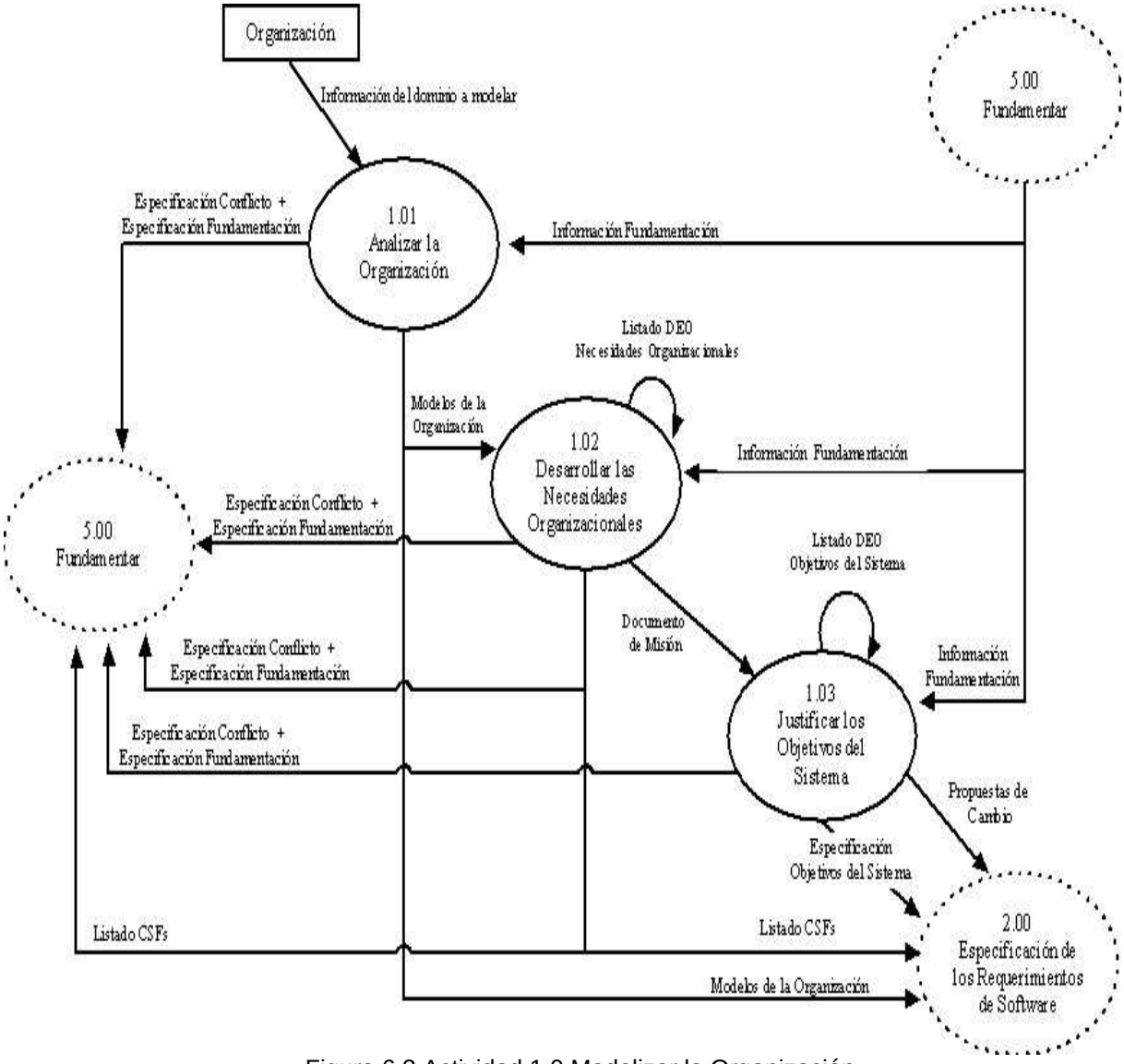


Figura 6.3 Actividad 1.0 Modelizar la Organización

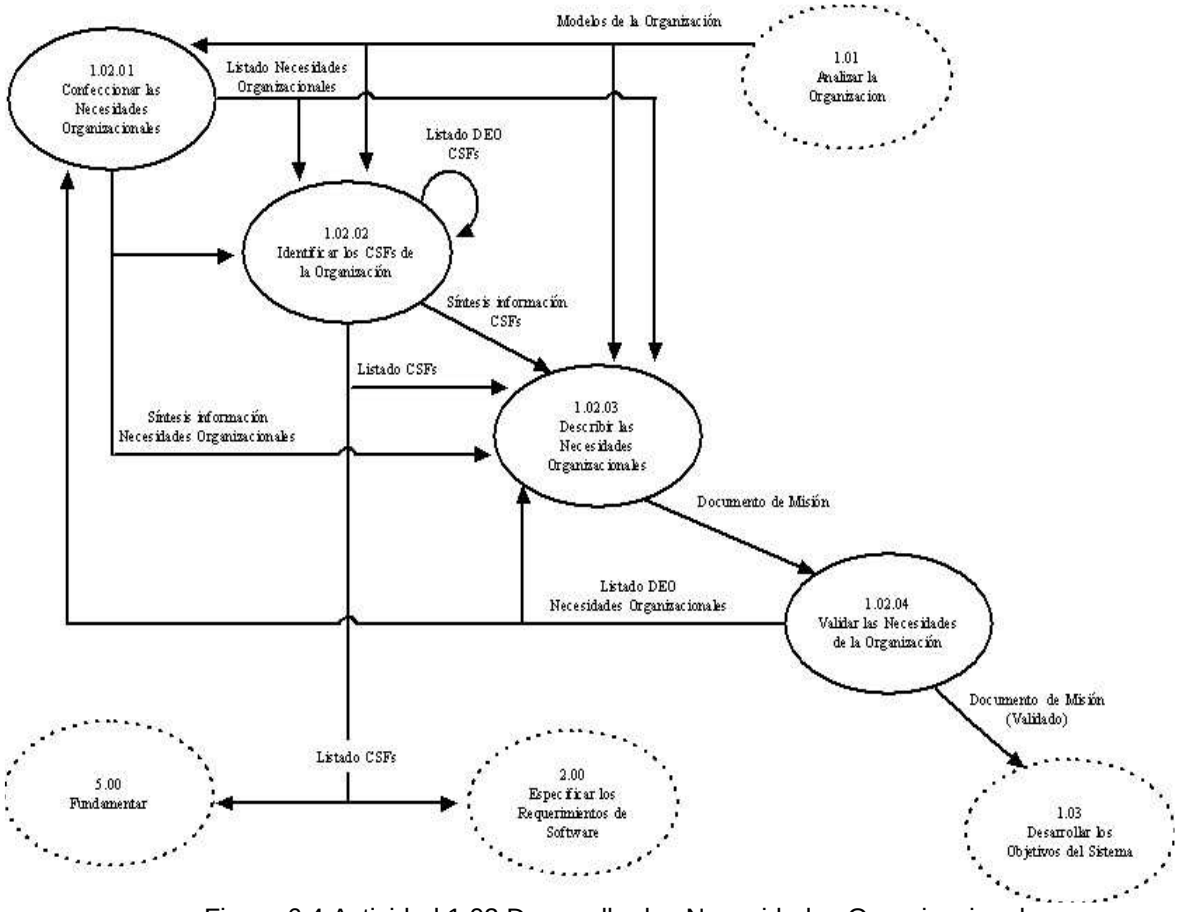


Figura 6.4 Actividad 1.02 Desarrollar las Necesidades Organizacionales

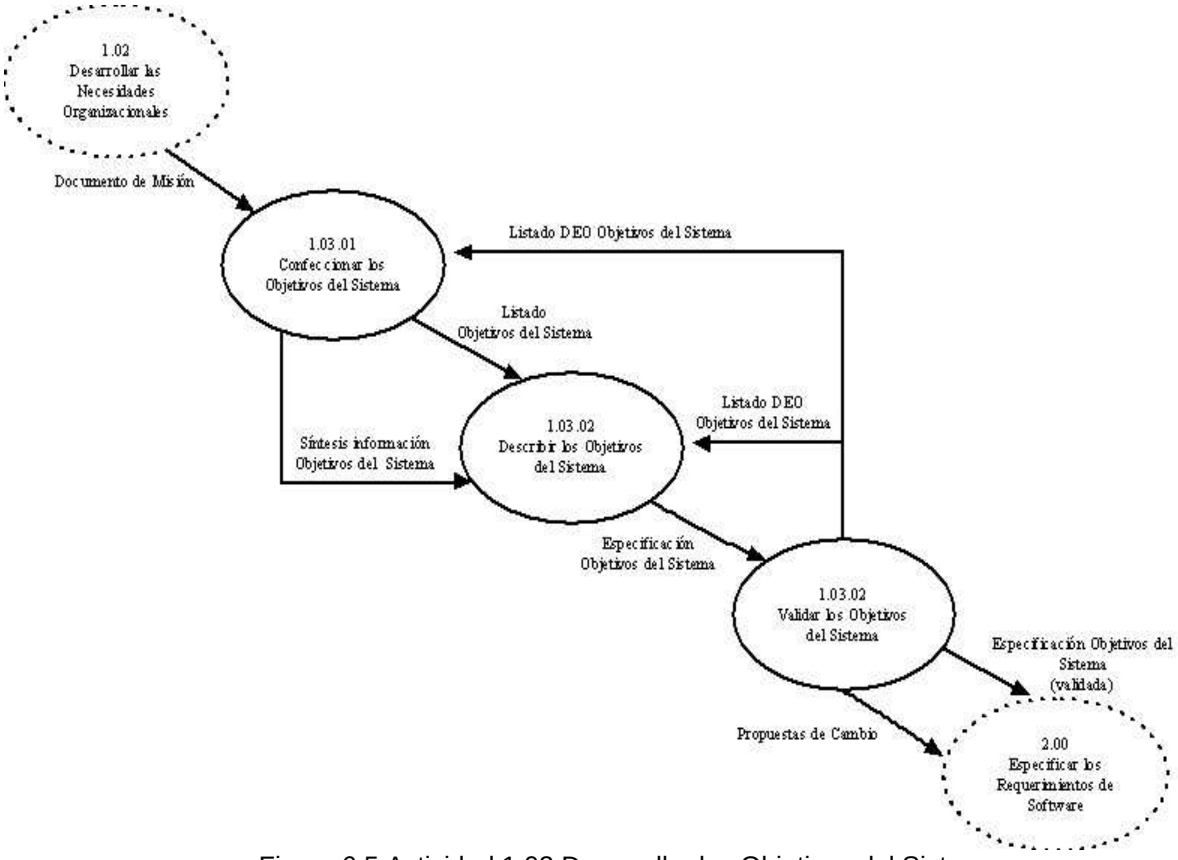


Figura 6.5 Actividad 1.03 Desarrollar los Objetivos del Sistema

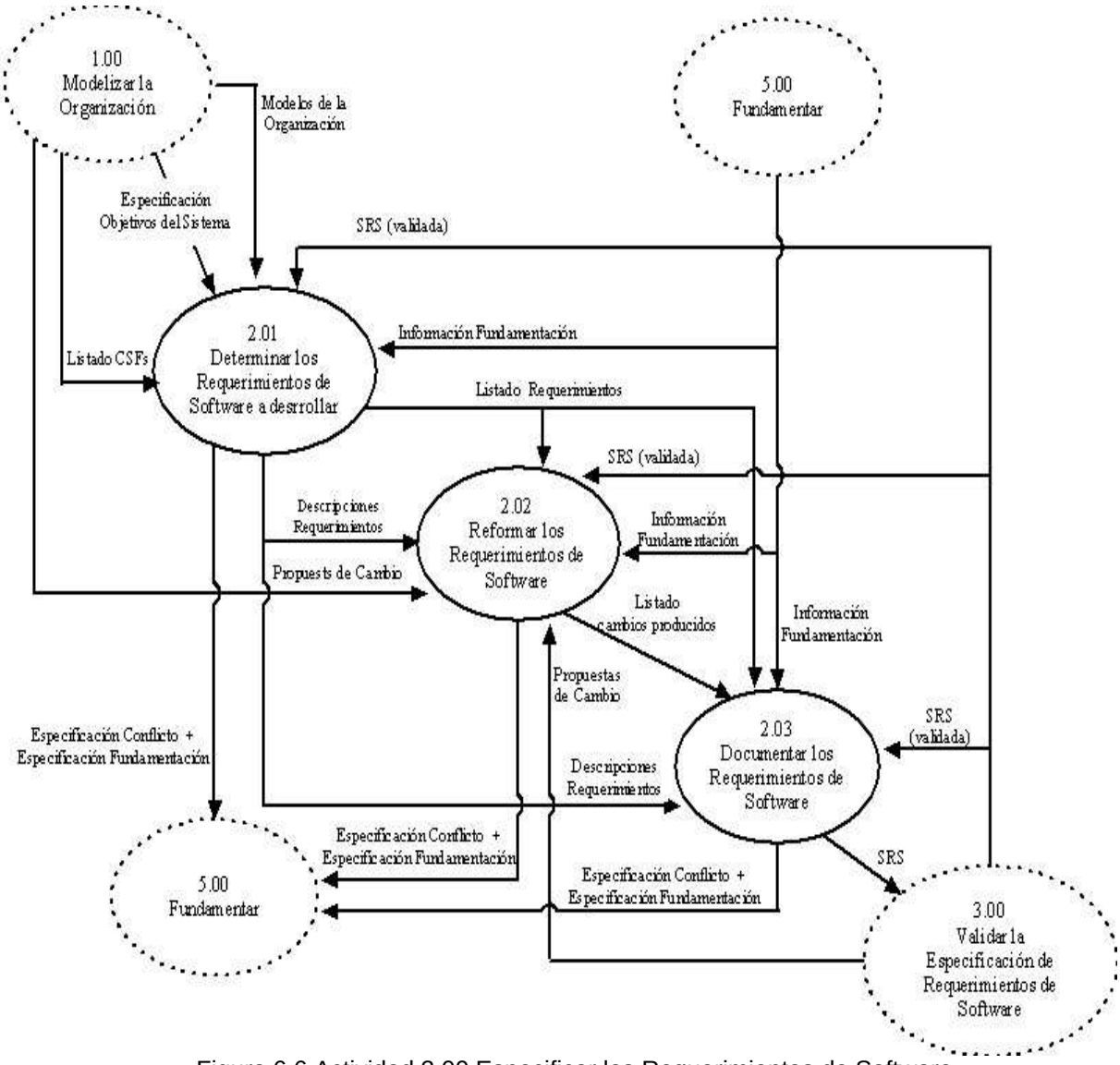


Figura 6.6 Actividad 2.00 Especificar los Requerimientos de Software

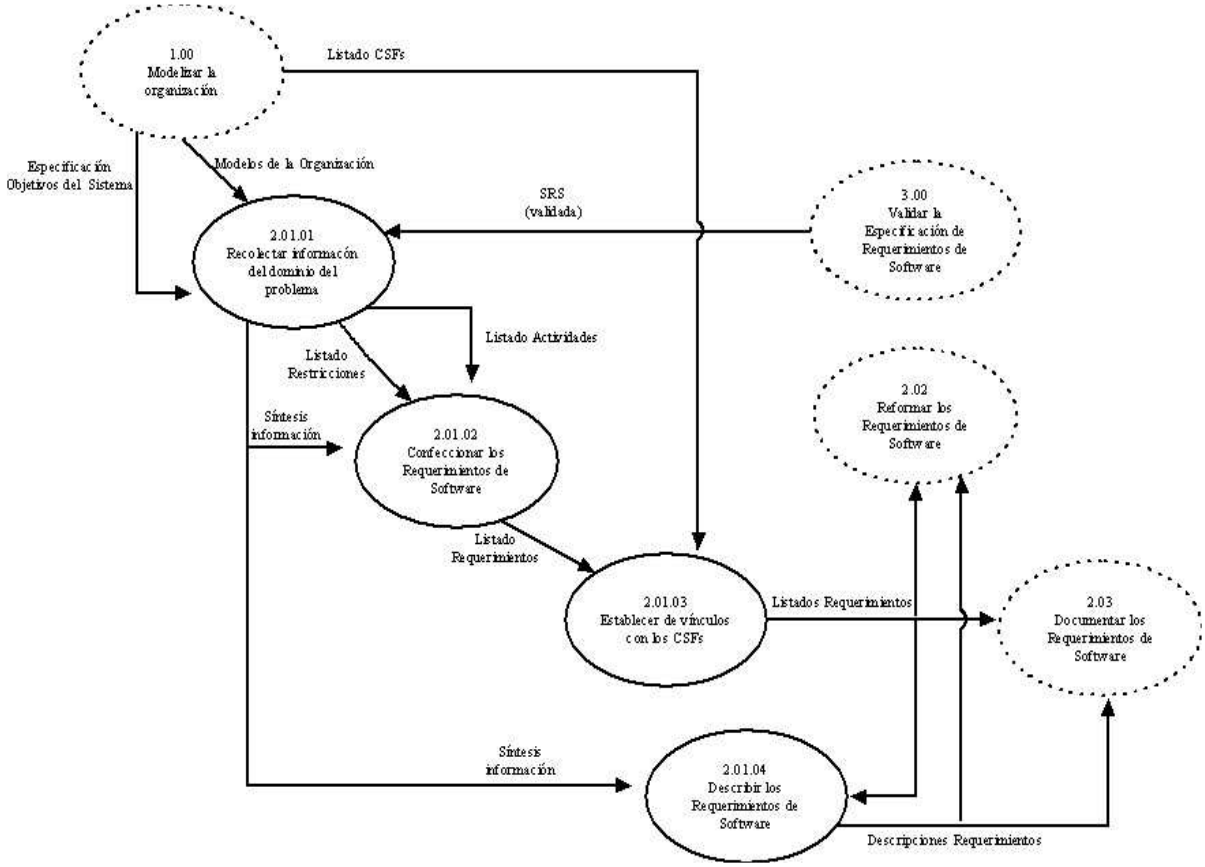


Figura 6.7 Actividad 2.01 Determinar los Requerimientos de Software a desarrollar

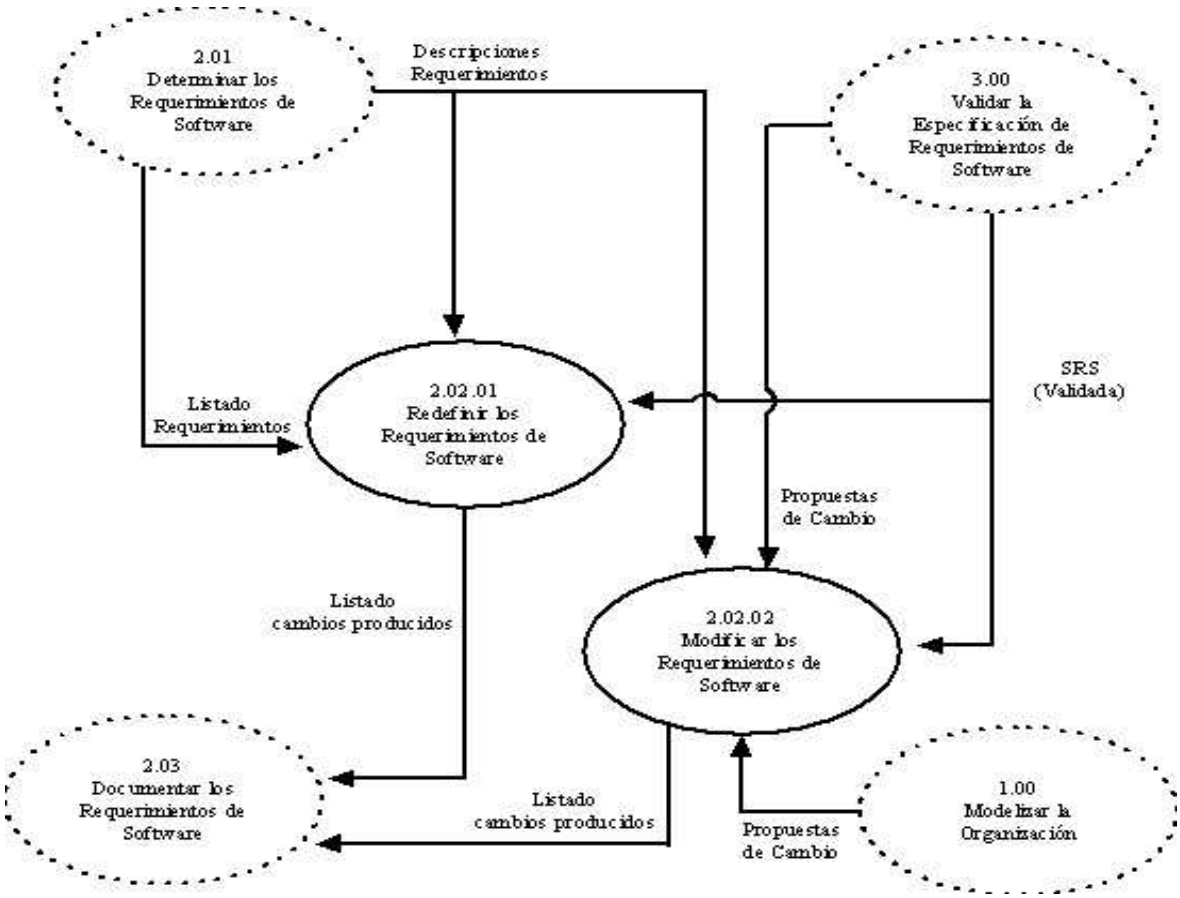


Figura 6.8 Actividad 2.02 Reformar los Requerimientos de Software

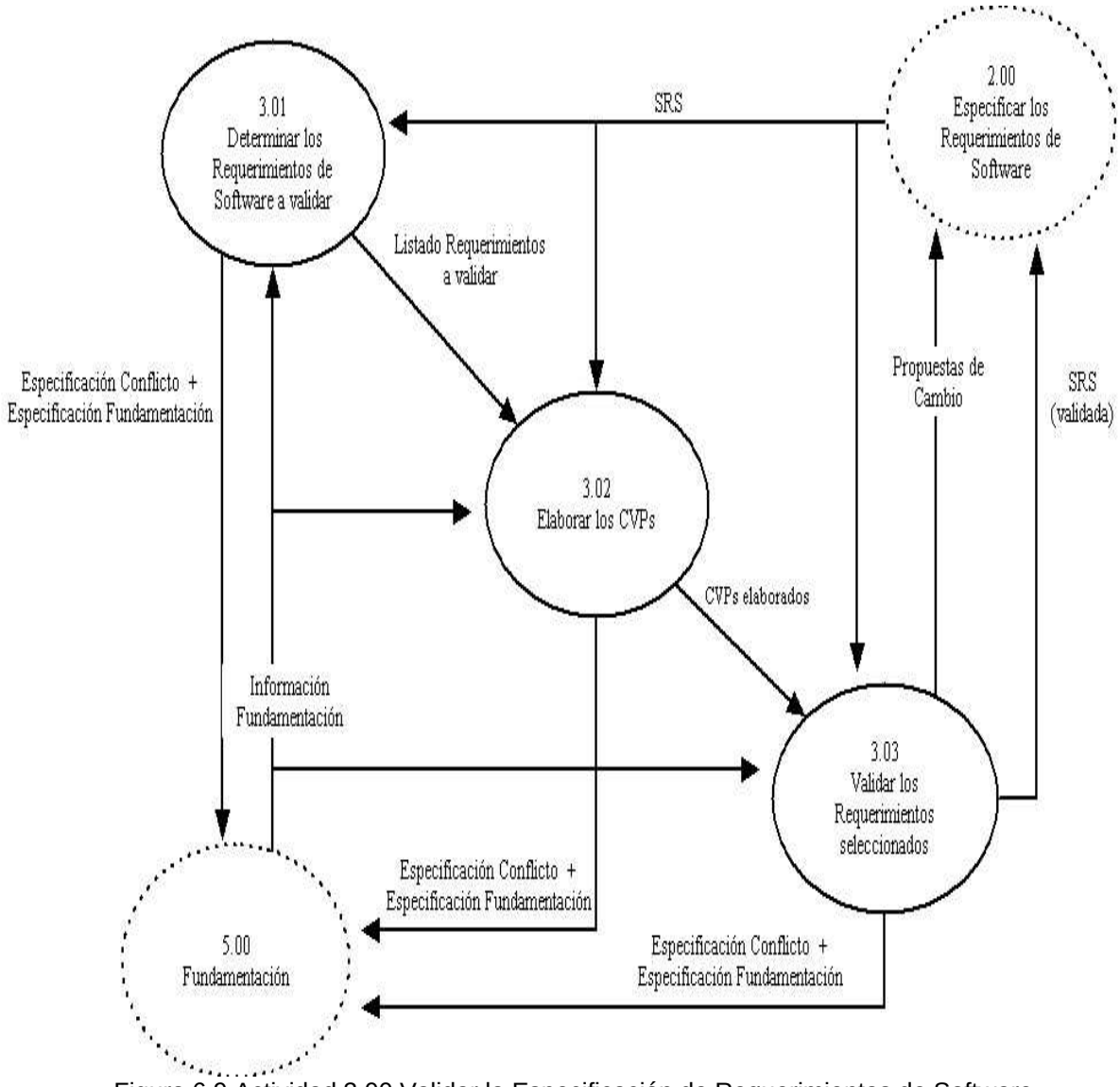


Figura 6.9 Actividad 3.00 Validar la Especificación de Requerimientos de Software

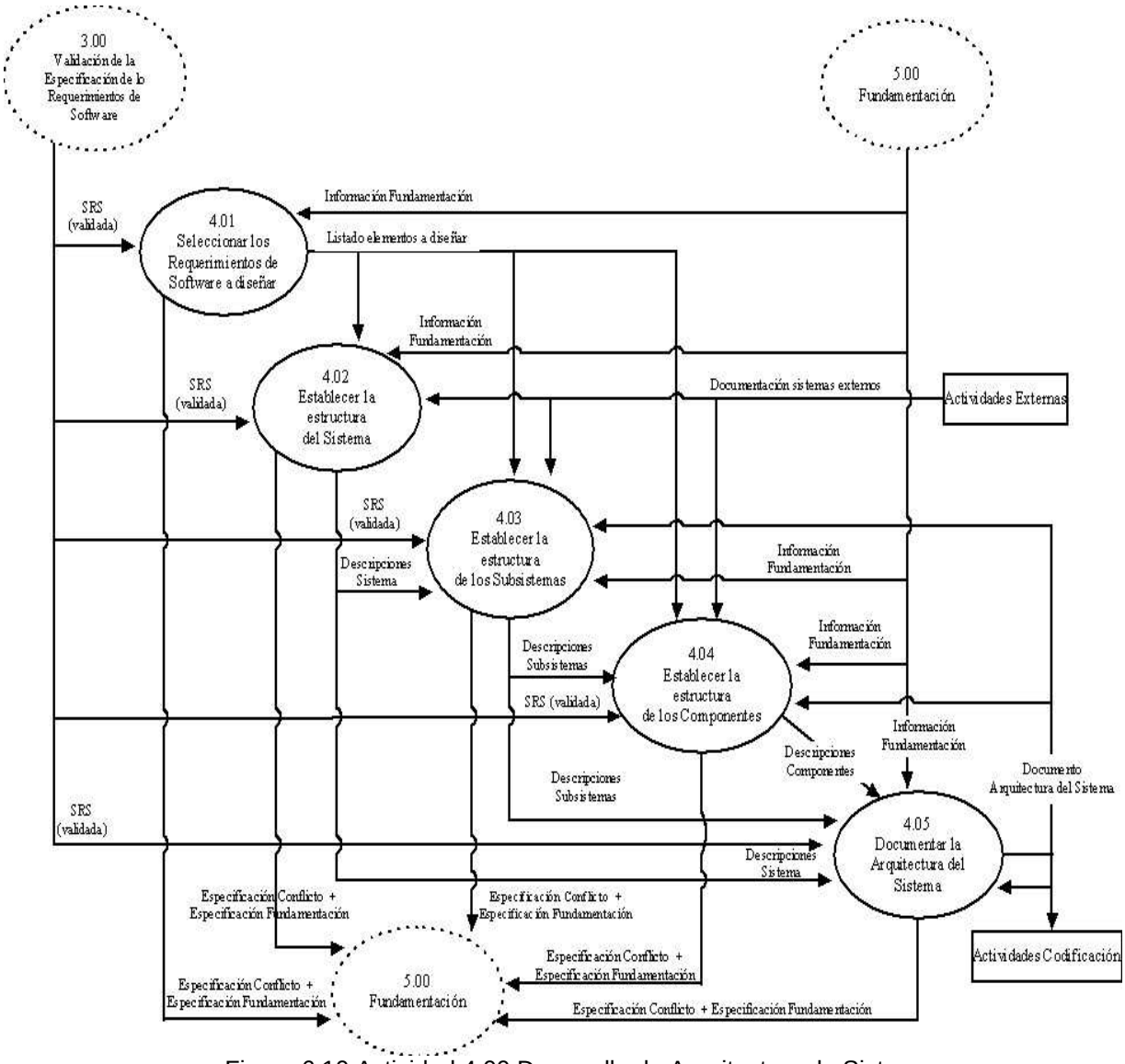


Figura 6.10 Actividad 4.00 Desarrollar la Arquitectura de Sistema

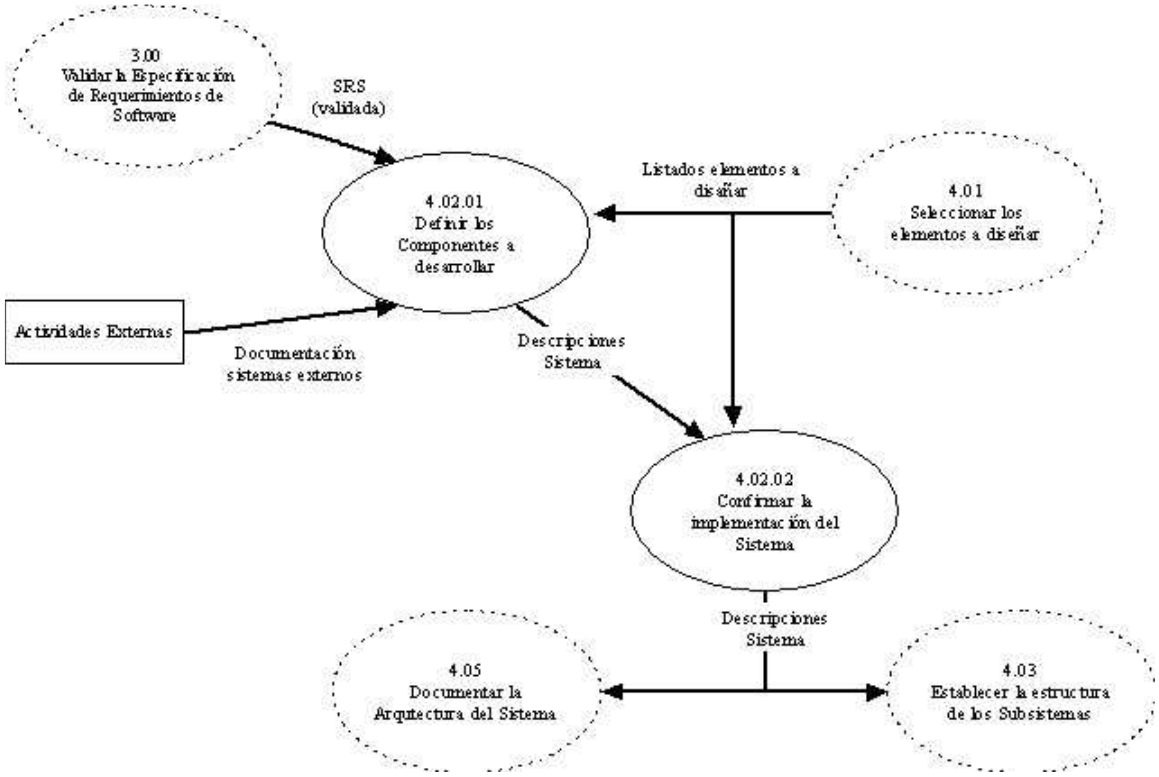


Figura 6.11 Actividad Establecer la estructura del Sistema

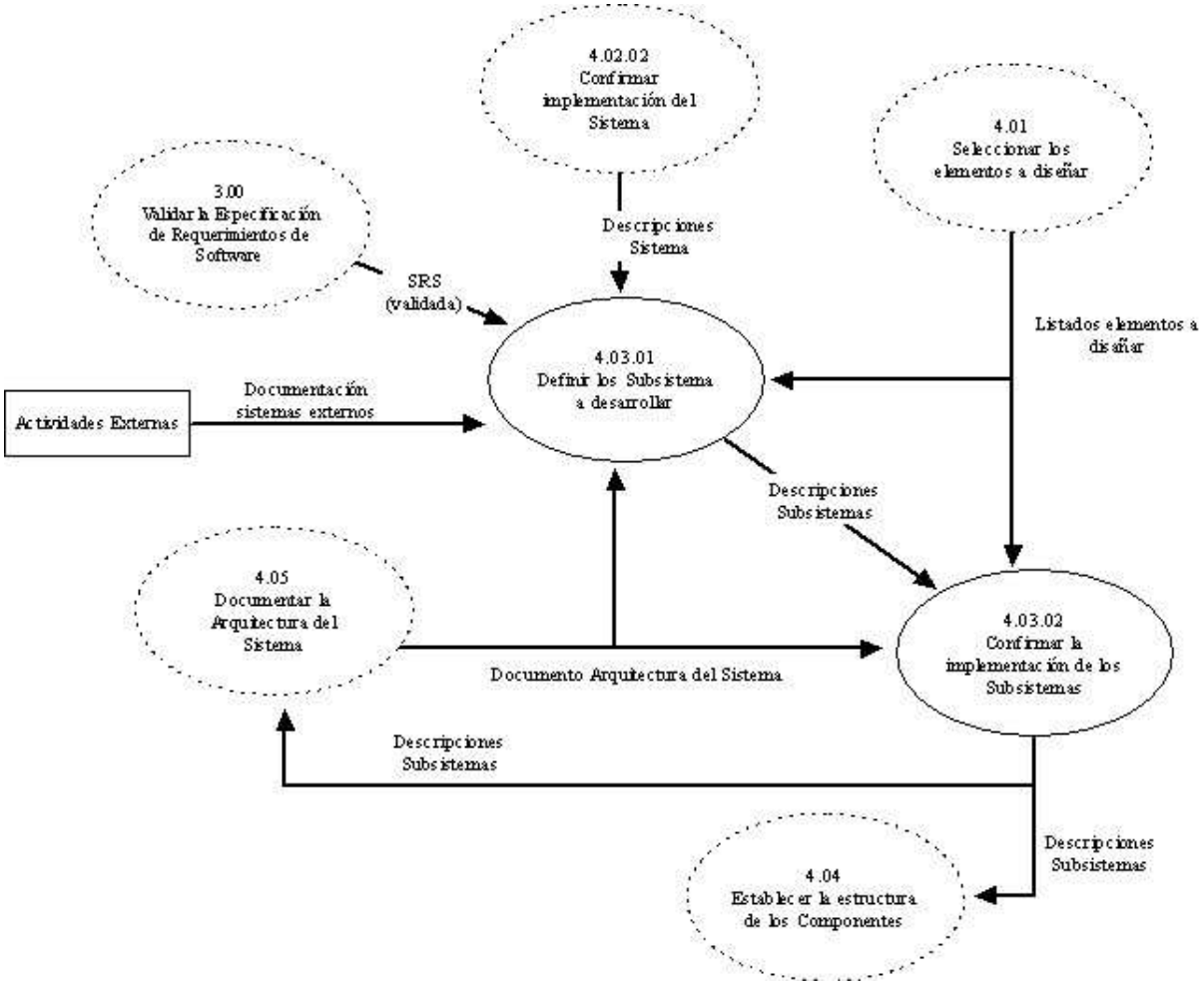


Figura 6.12 Actividad Establecer la estructura de los Subsistemas

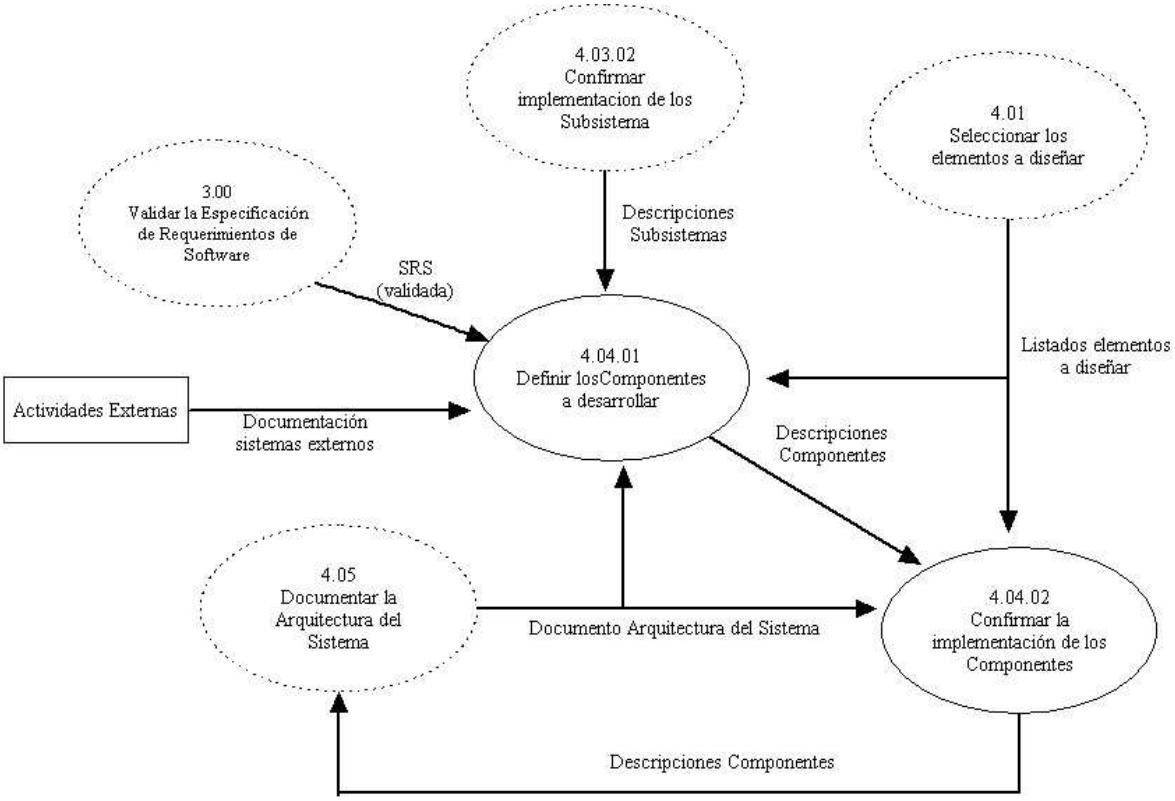


Figura 6.13 Actividad Establecer la estructura de los Componentes

Código:	**Nombre:**
	PROCESO DE DESARROLLO DE SOFTWARE

Objetivos:

Desarrollar un producto de software acorde a las necesidades establecidas por la organización.

Justificación:

Describir las actividades que integran el proceso de desarrollo con sus respectivos input/output, recursos (tanto humanos y tecnológicos) y sus respectivas mediciones facilita el desarrollo del producto software.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Modelizar la Organización	Representar el contexto organizacional y las necesidades organizacionales a desarrollar.	LP
2	Especificar los Requerimientos de Software	Confeccionar el documento Especificación de Requerimientos de Software	ER
3	Validar la Especificación de Requerimientos de Software	Asegurar que los requerimientos especificados cumplan con las necesidades de la organización	TE
4	Desarrollar la Arquitectura del Sistema	Confeccionar el Documento Arquitectura del Sistema	DI

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas de gestión Técnicas de elicitación seleccionadas Técnicas de negociación Conocimientos del dominio de la aplicación
ER	Técnicas de elicitación seleccionadas Técnicas de negociación Conocimientos del dominio de la aplicación
DI	Técnicas de diseño arquitectónico
TE	Técnicas de validación de software seleccionadas
GS	Visión global de la organización Conocimientos del dominio de la aplicación
UO	Conocimientos del dominio de la aplicación

Tecnológicos:

Procesador de Textos
Planilla de Cálculos
Graficadores

Técnicas:

Técnicas de elicitación
Técnicas de diseño arquitectónico
Técnicas de validación y verificación del software
Técnicas de resolución de conflictos

Input:

Nombre	Origen	Criterio de Entrada: Estado o Condición	Tipo
Información del dominio a modelar	ORG	Información de la organización y de su contexto	INF
Documentación sistemas externos	AE	Información sistemas externos	PDT
Información Fundamentación	5.00	Registradas las acciones de fundamentación	PDT

Output

Nombre	Destino	Criterio de Salida: Estado o Condición	Tipo
Listado CSFs[39]	5.00 5.01 5.02 5.02.02	CSFs validados por la gerencia superior	PDT
Documento Arquitectura del Sistema	AC	Descripción de la arquitectura del sistema sin validar	PDT
Especificación Conflicto	5.00 5.01	Identificador, descripción y contexto del conflicto	PDT
Especificación Fundamentación	5.00 5.03	Identificador del objeto a justificar	PDT

Definiciones:

Entidad: Representa los diferentes objetos[40] elaborados durante el proceso de desarrollo. Cabe señalar que en [Ramesh *et al*, 2001] los objetos representan los input/output de proceso, como ser requerimientos, situaciones conflictivas, componentes CSFs etc.

Semántica vínculo traceability: Los vínculos de traceability deben poseer como mínimo los siguientes atributos:
Denominación: <Nombre del vínculo>
Tipo de vínculo: <Evolución | Dependencia | Justificación | Satisfacción >
Entidad Origen: <Identificador de la entidad que origina el vínculo >
Entidad Destino: <Identificador de la entidad > Representa tanto una nueva entidad que se ha creado como consecuencia de una acción, o establece la entidad que se vincula con la entidad origen
Cardinalidad: Representa el número de entidades que se vinculan
La semántica de las relaciones puede variar dependiendo del vínculo y de las entidades afectadas.

Información Fundamentación: Representa la información recolectada y documentada en las actividades de

[39] Aunque en el Documento de Misión se incluye el Listado CSFs, se ha optado por especificar a este listado como un producto de salida, debido a que en determinadas actividades sólo es necesario contar con dicha información y no con la totalidad de las descripciones que integran el Documento de Misión.

fundamentación. Dicha información incluye la documentación referida a la resolución de los conflictos y las situaciones problemáticas (Especificación Conflicto) que se han presentado, como así también a las justificaciones (Especificación Fundamentación) realizadas durante el proceso de desarrollo de software.
Restricciones:
Las provenientes del dominio del problema. Las actividades especificadas representan una instancia del metamodelo definido en [Ramesh *et al*, 2001]. No se especifican las actividades concernientes a la codificación del sistema.
Variables a medir:
Nombre de la variable
Variables definidas en 1.0, 2.0, 3.0 y 4.0.
Observaciones:
Se ha optado por especificar sólo en las actividades 1.0, 2.0, 3.0 y 4.0 las entradas y salidas relacionadas con el proceso Fundamentar. De lo contrario, se tendrían que haber especificado en cada una de las actividades que integran el proceso de desarrollo de software.

6.2.1 Actividad 1.0 Modelizar la Organización

El proceso de desarrollo de software comienza con el análisis y descripción de los principales aspectos que caracterizan a la organización destinataria del software que se desarrolla. Las actividades que integran **Modelizar la Organización** son las siguientes:

1. Analizar la Organización
2. Desarrollar las Necesidades Organizacionales
3. Desarrollar los Objetivos del Sistema

Al finalizar la actividad se debe haber identificado y modelizado tanto las **necesidades organizacionales** como los **objetivos del sistema** a cumplimentar. Cabe recordar que estos objetivos deben justificar a las necesidades identificadas. Tales relaciones se representan mediante el vínculo *justify.* El metamodelo descripto en [Ramesh *et al*, 2001] contempla la confección de **escenarios** que modelen tanto a las necesidades organizacionales como a los objetivos del sistema. Cuando fuese necesaria la elaboración de un escenario se vinculará con la respectiva entidad por medio del vínculo *describe*. Las entidades y relaciones especificadas en la actividad **Modelizar la Organización** corresponden al submodelo **Gestión de Requerimientos** en [Ramesh *et al*, 2001].

[40] El término objeto no hace referencia a la metodología de desarrollo de software.

Código:	**Nombre:**	**Modelizar la Organización** Especificar los Requerimientos de Software Validar la Especificación de Requerimientos de Software Desarrollar la Arquitectura del Sistema
1.00	**Modelizar la Organización**	

Objetivos:

Representar el contexto organizacional y las necesidades organizacionales a desarrollar.

Justificación:

Lo elaborado en esta actividad permitirá identificar y modelizar los requerimientos de software como así también facilitará la validación y gestión de los mismos.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar la Organización	Elaborar modelos descriptivos de la organización y del contexto en la cual está inserta	LP
2	Desarrollar las Necesidades Organizacionales	Determinar el propósito del sistema a construir	LP
3	Desarrollar los Objetivos del Sistema	Identificar los objetivos del sistema a desarrollar y asegurar su correcta justificación	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas de gestión Técnicas de elicitación seleccionadas Técnicas de negociación Conocimientos del dominio de la aplicación
ER	Técnicas de elicitación seleccionadas Conocimientos del dominio de la aplicación
GS	Visión global de la empresa Conocimientos del dominio de la aplicación
UO	Conocimientos del dominio de la aplicación

Tecnológicos:

Procesador de Textos
Planilla de Cálculos
Graficadores

Técnicas:

Técnicas de elicitación

Input:

Nombre	Origen	Criterio de Entrada: Estado o Condición	Tipo
Información del dominio a modelar	ORG	Información de la organización y de su contexto	INF
Información Fundamentación	5.00	Registradas las acciones de fundamentación	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación Objetivos del Sistema	2.00 2.01 2.01.01	Especificación validada por la gerencia superior	PDT
Listado CSFs	2.00 2.01 2.01.03 5.00 5.01 5.02 5.02.02	CSFs validados por la gerencia superior	PDT
Modelos de la Organización	2.00 2.01 2.01.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Propuestas de Cambio	2.00 2.02 2.02.02	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Especificación Conflicto	5.00 5.01	Identificador, descripción y contexto del conflicto	PDT
Especificación Fundamentación	5.00 5.03	Identificador del objeto a justificar	PDT

Definiciones:

Necesidad Organizacional: Representan el propósito principal del sistema a construir [Ramesh *et al,* 2001]. Los criterios de clasificación de las necesidades organizacionales:

- ***Estratégica****:* Representan las necesidades de largo plazo de la organización.
- ***Operacional*****:** Representan las necesidades inmediatas de la organización.

Objetivo del Sistema: se emplea para representar las metas que el sistema debe lograr. Están expresado con un alto nivel de abstracción para ser tratados como requerimientos [Ramesh *et al,* 2001].

Restricciones:

Variables a medir:

Nombre de la variable

Variables definidas en 1.01,1.02 y 1.03

Observaciones:

Al realizar la actividad raíz se confeccionan los siguientes documentos: **Modelos de la Organización, Documento de Misión, Especificación Objetivos del Sistema y Propuestas de Cambio**. Es importante destacar la activa participación y el compromiso de la gerencia superior al realizar dicha actividad. Esto se debe a que se describen los principales aspectos del negocio. Por último, en esta actividad pueden originarse conflictos, o presentarse hechos o acciones que necesitan ser justificadas. Ambas circunstancias deben estar debidamente documentadas.

6.2.1.1 Actividad 1.01 Analizar la Organización

El grado de detalle con que se encare la actividad **Analizar la Organización** depende en gran medida del entendimiento que el grupo de desarrollo posea acerca de la organización y del dominio del problema. Aunque esta actividad no esté contemplada en el metamodelo seleccionado, su realización facilitará tanto la identificación como la comprensión de las necesidades organizacionales y los objetivos del sistema a desarrollar. Esta actividad se descompone en:

1. Recolectar y analizar la información de la organización
2. Documentar la información recolectada

Código:	**Nombre:**	Modelizar la Organización **Analizar la Organización** Desarrollar Necesidades Organizacionales Desarrollar los Objetivos del Sistema
1.01	**Analizar la Organización**	

Objetivos:

Elaborar modelos descriptivos de la organización y del contexto en la cual está inserta.

Justificación:

Identificar, analizar y modelizar los principales aspectos de la organización permitirá en la próxima actividad desarrollar las necesidades organizacionales.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Recolectar y analizar la información de la organización	Aprender sobre la organización y su contexto	LP
2	Documentar la información recolectada	Elaborar el documento Modelos de la Organización	LP

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Procesador de Textos Planilla de Cálculos Herramientas que soporten la realización de Organigramas
ER	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema	
LP	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema	
GS	Visión global de la organización y de su contexto	
UO	Conocimientos del dominio de la aplicación	

Técnicas:

Entrevistas

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Información del dominio a modelar	ORG	Información de la organización y de su contexto	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Modelos de la Organización	1.02 1.02.01 1.02.02 1.02.03 2.00 2.01 2.01.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
Cantidad de objetivos, estrategias y políticas de la organización	
Cantidad de procesos y productos analizados y descriptos	

Cantidad de recursos afectados por procesos analizados	
Total de personal detectado por área	
Entrevistas realizadas	
Entrevistas realizadas por personal	
Cantidad de documentos consultados	
Observaciones:	

6.2.1.2 Actividad 1.02 Desarrollar las Necesidades Organizacionales

Finalizado el análisis de los aspectos principales de la organización, se deben identificar y modelizar las **necesidades organizacionales**. Esta actividad se descompone en:

1. Confeccionar las Necesidades Organizacionales
2. Identificar los CSFs de la organización
3. Describir las Necesidades Organizacionales
4. Validar las Necesidades Organizacionales

Las entidades confeccionadas en la actividad **Desarrollar las Necesidades Organizacionales** corresponden a las **necesidades organizacionales**. Un punto importante a tener en cuenta al realizar la actividad es la identificación de los **CSFs** de la organización. Estas entidades representan los recursos críticos que afectan al logro de las necesidades organizacionales que han sido identificadas. La relación existente entre las necesidades y los CSFs se representan por medio del vínculo *identify*. Cuando fuese necesaria la elaboración de un escenario, se vinculará con la respectiva necesidad por medio del vínculo *describe*. Las salidas correspondientes a la actividad se materializan en el **Documento de Misión** y en el **Listado CSFs**

Código:	**Nombre:**	Modelizar la Organización Analizar la Organización **Desarrollar las Necesidades Organizacionales** Desarrollar los Objetivos del Sistema
1.02	**Desarrollar las Necesidades Organizacionales**	

Objetivos:

Determinar el propósito del sistema a construir.

Justificación:

Todos los productos elaborados durante el proceso de desarrollo estarán legitimados por el resultado de esta actividad.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Confeccionar las Necesidades Organizacionales	Identificar y clasificar las necesidades de la organización	LP
2	Identificar los CSFs de la organización	Confeccionar el listado CSFs	LP
3	Describir las Necesidades Organizacionales	Elaborar el Documento de Misión	LP
4	Validar las Necesidades Organizacionales	Asegurar la adecuada identificación de las necesidades organizacionales	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema
LP	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema Técnicas de negociación
GS	Visión global de la organización y de su contexto

Tecnológicos:

Procesador de Textos
Planilla de Cálculos

Técnicas:

Escenarios
Entrevistas

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Listado DEO Necesidades Organizacionales	1.02.04	Si no está vacía	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Documento de Misión	1.03 1.03.01	Documento validado por la Gerencia Superior	PDT
Listado CSFs	2.00 2.01 2.01.03 5.00 5.01 5.02 5.02.02	CSFs validados por la Gerencia Superior	PDT
Listado DEO Necesidades Organizacionales	1.02.01 1.02.03	Si no está vacía	INF

Definiciones:

CSF: Representan todos aquellos elementos identificados como críticos para el éxito del proyecto [Ramesh *et. el*, 2001].

Restricciones:

Un proyecto catalogado como pequeño puede contar con siete necesidades organizacionales como máximo.
El cambio de alguna Necesidad Organizacional conduce a elaborar otro proyecto de software.

Variables a medir:

Nombre de la variable	
Entrevistas realizadas	
Documentos consultados	
Necesidades Organizacionales identificadas/eliminadas/definitivas	
Necesidades Organizacionales por criterio de clasificación	
CSFs identificados/modificados/eliminados/definitivos	
CSFs agrupados por necesidad organizacional	
Vínculos *identify* identificados/modificados/eliminados/definitivos	
Escenarios identificados/modificados/eliminados/definitivos	
DEO identificados	

Observaciones:

En esta actividad se confeccionan las entidades **Necesidades Organizacionales, CSFs, Escenarios, DEO.**

Código:	**Nombre:**	Desarrollar las Necesidades Organizacionales **Confeccionar las Necesidades Organizacionales** Identificar los CSFs de la organización Describir las Necesidades Organizacionales Validar las Necesidades Organizacionales
1.02.01	**Confeccionar las Necesidades Organizacionales**	

Objetivos:

Identificar y clasificar las necesidades de la organización.

Justificación:

Identificar las necesidades de la organización constituye uno de los mecanismos por el cual se puede justificar lo elaborado durante el desarrollo.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Identificar las Necesidades de la Organización	Elaborar el listado necesidades organizacionales	LP
2	Clasificar las Necesidades Organizacionales	Agrupar las necesidades identificadas mediante la selección de un criterio	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema
LP	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de negociación
GS	Visión global de la empresa Expectativas del sistema a modelar

Tecnológicos:

Planilla de Cálculos

Técnicas:

Entrevistas

Identificar las Necesidades de la Organización. Por cada **Necesidad Organizacional** identificada se debe:

1. Confeccionar la entidad correspondiente.
2. Completar el componente de información **Identificador.**
3. Con la información obtenida se debe elaborar el **Listado Necesidades Organizacionales**.

Clasificar las Necesidades Organizacionales. Las acciones que componen esta actividad son:

1. Seleccionar el criterio de clasificación.
2. Agrupar a las necesidades identificadas según el criterio establecido.
3. Por cada **Necesidad Organizacional** se completa el componente **Tipo** con el vínculo *is_a* de acuerdo a la clasificación realizada.
4. Con la información obtenida se debe completar el **Listado Necesidades Organizacionales**.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Listado DEO Necesidades Organizacionales	1.02.04	Si no está vacía	INF
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Síntesis información Necesidades Organizacionales	AP	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado Necesidades Organizacionales	1.02.02 1.02.03	Completo los componentes Identificador y Tipo de las necesidades identificadas	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
Entrevistas realizadas	
Documentos consultados	
Necesidades Organizacionales identificadas/eliminadas/definitivas	
Necesidades Organizacionales por criterio de clasificación	

Observaciones:

En esta actividad se confecciona el vínculo *is_a*. Las necesidades identificadas deben estar alineadas con las políticas y estrategias vigentes en la organización.

Si se tiene como entrada al Listado DEO Necesidades Organizacionales se deben realizar las respectivas acciones de modificación (identificar o eliminar necesidades organizacionales).

Código:	**Nombre:**	Desarrollar las Necesidades Organizacionales Confeccionar las Necesidades Organizacionales **Identificar los CSFs de la organización** Describir las Necesidades Organizacionales Validar las Necesidades Organizacionales
1.02.02	**Identificar los CSFs de la Organización**	

Objetivos:

Confeccionar el listado de los CSFs.

Justificación:

El haber identificado los CSFs que afectan al logro de las necesidades organizacionales facilitará la gestión de los requerimientos.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Determinar los CSFs de la organización	Elaborar el listado de los recursos críticos que afectan el logro de las necesidades identificadas	LP
2	Describir los CSFs	Completar los componentes de información de los CSFs	LP
3	Validar los CSFs	Asegurar la adecuada identificación y descripción de los CSFs identificados	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de Escenarios
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de Escenarios
GS	Visión global de la empresa Conocimientos referidos a los recursos críticos de la organización

Tecnológicos:

Planilla de Cálculos

Técnicas:

Análisis de la documentación

Entrevistas

Determinar los CSFs de la organización. A partir de la información recolectada se debe:

1. Elaborar un listado que contenga los recursos identificados como críticos para el logro de las necesidades identificadas.
2. Por cada recurso del listado se confecciona una entidad **CSF** con la información pertinente.

Describir a los CSF. Esta actividad debe ser llevada a cabo por cada uno de los CSFs confeccionados

1. Completar los restantes componentes de información de los CSFs identificados.
2. Confeccionar el vínculo *identify* y registrarlo tanto en el componente **Identificado por** de los CSFs identificados, como en el componente **Identifica a** de las Necesidades Organizacionales afectadas.

Validar los CSFs. Por cada uno de los CSFs identificados se debe:

1. Chequear la correcta identificación
2. Chequear la correcta descripción
3. Por cada omisión, error o deficiencia que se encuentre, se debe confeccionar una entidad **DEO** y se registra en el **Listado DEO CSFs**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Listado Necesidades Organizacionales	1.02.01	Completo los componentes Identificador y Tipo de las necesidades identificadas	INF
Listado DEO CSFs	1.02.02	Si no está vacía	INF
Síntesis información Necesidades Organizacionales	1.02.01	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Síntesis información CSFs	AP	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado DEO CSFs	1.02.02	Si no está vacía	INF
Listado CSFs	1.02.03 2.00 2.01 2.02.03 5.00 5.01 5.02 5.02.02	CSFs validados por la gerencia superior	PDT

Definiciones:

Para modelar a los CSFs sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[41].

Restricciones:

[41] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Los CSFs son definidos como recursos, por ser lo establecido en el metamodelo seleccionado.
Una Necesidad Organizacional puede poseer uno o muchos CSFs y un CSF puede estar vinculado a una o muchas Necesidades Organizacionales.
Los CSFs identificados no cambian durante el transcurso del desarrollo del proyecto.

Variables a medir:

Nombre de la variable	
CSFs identificados/modificados/eliminados/definitivos	
CSFs agrupados por necesidad organizacional	
Vínculos *identify* identificados/modificados/eliminados/definitivos	

Observaciones:

En esta actividad se confecciona el vínculo *identify***.** El vínculo *is_a* definido en el metamodelo, el cual relaciona a un recurso crítico con un CSF no fue definido por considerarlo innecesario. La actividad **Validar los CSFs** debe ser llevada a cabo con la activa participación de la gerencia superior de la organización.

Código:	**Nombre:**	Desarrollar las Necesidades Organizacionales Confeccionar las Necesidades Organizacionales Identificar los CSFs de la organización **Describir las Necesidades Organizacionales** Validar las Necesidades Organizacionales
1.02.03	**Describir las Necesidades Organizacionales**	

Objetivos:

Elaborar el Documento de Misión

Justificación:

A partir de las descripciones realizadas a las necesidades organizacionales se confecciona el Documento de Misión. Este documento debe poseer la información necesaria para poder identificar los objetivos de sistema a implementar

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Caracterizar las Necesidades Organizacionales	Completar los componentes de información de las necesidades organizacionales	LP
2	Desarrollar los Escenarios de las Necesidades Organizacionales	Describir con mayor grado de detalle a las necesidades organizacionales	LP
3	Confeccionar el Documento de Misión	Generar un documento que contenga la información necesaria para comprender a las necesidades identificadas	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de escenarios
LP	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de escenarios Técnicas de negociación

Tecnológicos:

Planilla de Cálculos
Herramienta que soporten escenarios

Técnicas:

Caracterizar las Necesidades Organizacionales. Por cada una de las necesidades identificadas se debe:
1. Establecer los vínculos existentes con las políticas, estrategias y metas de la organización.
2. Completar los restantes componentes que describe a la necesidad organizacional.
3. Con la información obtenida se debe completar el **Listado Necesidades Organizacionales**

Desarrollar los Escenarios de las Necesidades Organizacionales. Por cada necesidad que deba ser descripta con mayor grado de detalle se debe:
1. Confeccionar el escenario pertinente
2. Confeccionar el vínculo *describe* y registrarlo en el componente **Modelado por** de la respectiva necesidad.
3. Verificar que los escenarios confeccionados estén elaborados según el estándar establecido

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Listado Necesidades Organizacionales	1.02.01	Completo los componentes Identificador y Tipo de las necesidades identificadas	INF
Listado CSFs	1.02.02	CSFs validados por la gerencia superior	PDT
Listado DEO Necesidades Organizacionales	1.02.04	Si no está vacía	INF
Síntesis información CSFs	1.02.02	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Síntesis información Necesidades Organizacionales	1.02.01	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Documento de Misión	1.02.04	Necesidades Organizacionales y CSFs identificados y descriptos	PDT

Definiciones:

Para el desarrollo de los escenarios se toma como referencia lo formalizado en el proceso LEC para la actividad **3.0**

Construcción de Escenarios definido en [Zuñiga, 2004].
Restricciones:
Para modelar a las Necesidades Organizacionales sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[42].
Variables a medir:
Nombre de la variable
Escenarios identificados/modificados/eliminados/definitivos
Observaciones:
En esta actividad se confecciona el vínculo *describe*. Dependiendo del grado de entendimiento que se posea el paso dos puede ser optativo.

Código:	**Nombre:**	Desarrollar las Necesidades Organizacionales Confeccionar las Necesidades Organizacionales Identificar los CSFs de la organización Describir las Necesidades Organizacionales **Validar las Necesidades Organizacionales**
1.02.04	**Validar las Necesidades Organizacionales**	

Objetivos:

Asegurar la adecuada identificación de las necesidades de la organización

Justificación:

La aprobación por parte de la gerencia superior de lo elaborado hasta el momento asegura la correcta identificación y descripción de las necesidades de la organización.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Verificar las Necesidades Organizacionales identificadas	Asegurar la correcta identificación y descripción de las necesidades organizacionales	LP
2	Verificar los Escenarios confeccionados	Asegurar que los escenarios sean consistentes con la postura de la gerencia superior	LP

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
LP	Técnicas para realizar entrevistas Técnicas de construcción de escenarios	
ER	Técnicas para realizar entrevistas Técnicas de construcción de escenarios	
GS	Visión global de la empresa Expectativas del sistema a modelar	

Técnicas:

Verificar las Necesidades Organizacionales identificadas. Por cada una de las necesidades identificadas se debe:

1. Chequear lo detallado en el componente **Identificador**
2. Chequear lo detallado en componente **Descripción**
3. Chequear la coherencia entre el componente **Identificador y Descripción**
4. Chequear la clasificación realizada
5. Chequear el contenido de los restantes componente de información
6. Por cada omisión, error o deficiencia que se encuentre, se debe confeccionar una entidad **DEO** y se registra en el **Listado DEO Necesidades Organizacionales**

Verificar los Escenarios confeccionados. Por cada uno de los escenarios confeccionados se debe:

1. Chequear el contenido del escenario
2. Chequear el contenido del vínculo *describe*
3. Por cada omisión, error o deficiencia que se encuentre, se debe confeccionar una entidad **DEO** y se registra en el **Listado DEO Necesidades Organizacionales**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento de Misión	1.02.03	Necesidades Organizacionales y CSFs identificados y descriptos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Documento de Misión	1.03 1.03.01	Documento validado por la gerencia superior	PDT
Listado DEO Necesidades Organizacionales	1.02 1.02.01 1.02.03	Si no está vacía	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable

[42] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Ítem DEO identificados	
Observaciones:	
Si el listado DEO presenta algún ítem, el proceso de desarrollo continúa en la actividad 1.02.01 o 1.02.03 dependiendo de los errores encontrados. En cambio si la lista se encontrara vacía, el proceso continúa con la siguiente actividad.	

6.2.1.3 Actividad 1.03 Desarrollar los Objetivos del Sistema

Concluída la identificación de las **necesidades organizacionales**, se deben establecer cuáles son los **objetivos del sistema** necesarios para cumplimentar el logro de las mismas. Estas actividades se detallan en:

1. Confeccionar los Objetivos del Sistema
2. Describir los Objetivos del Sistema
3. Validar los Objetivos del Sistema

En esta actividad no sólo se confeccionan las entidades objetivos del sistema, sino también se especifica la relación existente entre el objetivo del sistema y la necesidad organizacional que lo justifica. Esta relación se representa por medio del vínculo *justify*. Cuando fuese necesaria la elaboración de un escenario, se vinculará con el respectivo objetivo por medio del vínculo *describe*. La salida correspondiente a la actividad se materializa en el documento **Especificación Objetivos del Sistema**. Al realizar la validación de los objetivos del sistema se pueden detectar deficiencias, omisiones o errores que afectan a los requerimientos especificados. Con dicha información se confecciona el documento **Propuestas de Cambio**, el cual sirve de elemento de entrada para las actividades de modificación de los requerimientos. Esta situación sólo acontece durante las iteraciones tardías en donde se ha adicionado, modificado o eliminado algún objetivo del sistema.

Código:	**Nombre:**	Modelizar la Organización Analizar la Organización Desarrollar las Necesidades Organizacionales **Desarrollar los Objetivos del Sistema**
1.03	**Desarrollar los Objetivos del Sistema**	

Objetivos:

Identificar los objetivos del sistema a desarrollar y asegurar su correcta justificación.

Justificación:

Esta actividad produce la Especificación de los Objetivos del Sistema, la cual servirá como punto de partida para la identificación de los requerimientos de software a desarrollar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Confeccionar los Objetivos del Sistema	Identificar los objetivos del sistema que justifican a las necesidades organizacionales	LP
2	Describir los Objetivos del Sistema	Elaborar el documento Especificación de los Objetivos del Sistema	LP
3	Validar los Objetivos del Sistema	Asegurar la adecuada identificación de los objetivos del sistema	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema
ER	Conocimiento del dominio del problema Técnicas de elicitación seleccionadas
GS	Visión global de la organización Expectativas del sistema a modelar
UO	Conocimientos del dominio de la aplicación

Tecnológicos:

Planilla de Cálculos
Procesador de Textos

Técnicas:

Escenarios
Entrevistas

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Listado DEO Objetivos del Sistema	1.03.03	Si no está vacía	INF
Documento de Misión	1.02.04	Documento validado por la gerencia superior	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Propuestas de Cambio	2.00 2.02 2.02.02	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Especificación Objetivos del Sistema	2.00 2.01 2.01.01	Especificación validada por la gerencia superior	PDT
Listado DEO Objetivos del Sistema	1.03.03	Si no está vacía	INF

Definiciones:

Restricciones:

Los Objetivos del Sistema deben estar justificados por las Necesidades Organizacionales.

Variables a medir:

Nombre de la variable	
Objetivos del Sistema identificados/eliminados/definitivos	
Vínculos *justify* identificados por necesidad	
Escenarios identificados/modificados/eliminados/definitivos	
DEO identificados	
DEO por objetivo del sistema	
DEO por requerimiento de software	

Observaciones:

En esta actividad se confeccionan los componentes de información **Objetivos del Sistema.**

Código:	**Nombre:**	Desarrollar los Objetivos del Sistema **Confeccionar los Objetivos del Sistema** Describir los Objetivos del Sistema Validar los Objetivos del Sistema
1.03.01	**Confeccionar los Objetivos del Sistema**	

Objetivos:

Identificar los objetivos del sistema que justifican a las necesidades organizacionales.

Justificación:

En esta actividad se identifican las metas que el sistema a desarrollar debe lograr

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar la información producida	Comprender el alcance y complejidad de las necesidades organizacionales identificadas	LP
2	Identificar los Objetivos del Sistema	Elaborar el listado objetivos del sistema	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas para realizar entrevistas Conocimiento del dominio del problema
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema
GS	Visión global de la organización Expectativas del sistema a modelar
UO	Conocimientos del dominio de la aplicación

Tecnológicos:

Planilla de Cálculos
Procesador de Textos

Técnicas:

Análisis de la documentación

Entrevistas

Identificar los Objetivos del Sistema .Por cada **Necesidad Organizacional** identificada se debe:

1. Identificar el o los **Objetivo del Sistema** que se relaciona al logro de la necesidad identificada.
2. Por cada objetivo identificado se confecciona la entidad correspondiente y se completa el componente **Identificador**.
3. Por cada objetivo identificado se confecciona el vínculo *justify* con la información pertinente. Estos vínculos se deben registrar tanto en el componente **Justificado a** de las necesidades que lo justifica, como así también en el componente **Justificado por** del objetivo identificado.
4. Con la información obtenida se debe generar el **Listado de los Objetivos del Sistema**.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento de Misión	1.02.04	Documento validado por la gerencia superior	PDT
Listado DEO Objetivos del Sistema	1.03.03	Si no está vacía	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Síntesis información Objetivos del Sistema	1.03.02	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado Objetivos del Sistema	1.03.02	Completo los componentes de información Identificador y Justificado por	INF

Definiciones:

Restricciones:

Un Objetivo del Sistema está sólo vinculado con una Necesidad Organizacional.
Se suprimen aquellos Objetivos del Sistema que no estén justificados por las Necesidades Organizacionales.

Variables a medir:

Nombre de la variable	
Objetivos del Sistema identificados/eliminados/definitivos	
Vínculos *justify* identificados por necesidad	

Observaciones:

En esta actividad se confecciona el vínculo *justify*.

Código:	**Nombre:**	Desarrollar los Objetivos del Sistema Identificar los Objetivos del Sistema **Describir los Objetivos del Sistema** Validar los Objetivos del Sistema
1.03.02	**Describir los Objetivos del Sistema**	

Objetivos:

Elaborar el documento Especificación de los Objetivos del Sistema.

Justificación:

Esta actividad permite modelizar los objetivos del sistema que han sido identificados.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Caracterizar los Objetivos del Sistema	Completar los componentes de información de los objetivos del sistema	LP
2	Desarrollar los Escenarios de los Objetivos del Sistema	Describir con mayor grado de detalle a los objetivos del sistema	LP
3	Confeccionar el documento Especificación de los Objetivos del Sistema	Generar un documento que contenga la información necesaria para comprender a los objetivos del sistema	LP

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de escenarios
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema Técnicas de construcción de escenarios Técnicas de negociación

Tecnológicos:

Planilla de Cálculos
Herramienta que soporten escenarios

Técnicas:

Caracterizar los Objetivos del Sistema. Por cada uno de los objetivos identificados se debe:
1. Completar los restantes componentes que describen al objetivo del sistema identificado.
2. Con la información obtenida se debe completar el **Listado de los Objetivos del Sistema**

Desarrollar los Escenarios de los Objetivos del Sistema. Por cada objetivo que deba ser descripto con mayor grado de detalle se debe:
1. Confeccionar el escenario pertinente
2. Confeccionar el vínculo *describe* y registrarlo en el componente **Modelado por** del respectivo objetivo del sistema
3. Verificar que los escenarios confeccionados estén elaborados según el estándar establecido

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Síntesis información Objetivos del Sistema	1.03.01	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado DEO Objetivos del Sistema	1.03.03	Si no está vacía	INF
Listado Objetivos del Sistema	1.03.01	Completo los componentes de información Identificador y Justificado por	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación Objetivos del Sistema	1.03.03	Objetivos del Sistema identificados y descriptos	PDT

Definiciones:

Para el desarrollo de los escenarios se toma como parámetro lo formalizado para la actividad **3.0 Construcción de Escenarios** del proceso LEC definido en [Zuñiga, 2004].

Restricciones:

Para modelar los Objetivos del Sistema sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[43].

Variables a medir:

Nombre de la variable	
Escenarios identificados/modificados/eliminados/definitivos	

Observaciones:

En esta actividad se confecciona el vínculo *describe*. Dependiendo del grado de entendimiento que se posea, el paso dos puede ser optativo.

[43] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Código:	**Nombre:**	Desarrollar los Objetivos del Sistema Identificar los Objetivos del Sistema Describir los Objetivos del Sistema **Validar los Objetivos del Sistema**
1.03.03	**Validar los Objetivos del Sistema**	

Objetivos:

Asegurar la adecuada identificación de los objetivos del sistema.

Justificación:

La correcta identificación y descripción de los objetivos del sistema asegura que los requerimientos generados desde los mismos, sean relevantes para la organización.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Verificar los Objetivos del Sistema	Asegurar la correcta identificación y descripción de los objetivos del sistema	LP
2	Verificar los Escenarios confeccionados	Asegurar que los escenarios sean consistentes con la postura de la gerencia superior	LP

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
LP	Técnicas para realizar entrevistas Técnicas de construcción de escenarios	
ER	Técnicas para realizar entrevistas Técnicas de construcción de escenarios	
GS	Visión global de la empresa Expectativas del sistema a modelar	
UO	Expectativas del sistema a modelar	

Técnicas:

Verificar los Objetivos del Sistema identificados. Por cada uno de los objetivos del sistema identificados se debe:

1. Chequear lo detallado en el componente **Identificador.**
2. Chequear lo detallado en el componente **Descripción.**
3. Chequear la coherencia entre el componente **Identificador y Descripción**
4. Chequear la relación con la necesidad que lo justifica
5. Chequear el contenido de los restantes componente de información
6. Por cada omisión, error o deficiencia que se encuentre, se debe confecciona una entidad **DEO.** Si el contenido de la entidad DEO afecta a algún requerimiento se registra en el DEO en el documento **Propuestas de Cambio** y sino se registra en el **Listado DEO Objetivos del Sistema**

Verificar los Escenarios confeccionados. Por cada uno de los escenarios confeccionados se debe:

1. Chequear el contenido del escenario
2. Chequear el contenido del vínculo *describe*
3. Por cada omisión, error o deficiencia que se encuentre, se debe confecciona una entidad **DEO.** Si el contenido de la entidad DEO afecta a algún requerimiento se registra en el DEO en el documento **Propuestas de Cambio** y sino se registra en el **Listado DEO Objetivos del Sistema**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación Objetivos del Sistema	1.03.02	Objetivos del Sistema identificados y descriptos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación Objetivos del Sistema	2.00 2.01 2.01.01	Especificación validada por la gerencia superior	PDT
Propuestas de Cambio	2.00 2.02 2.02.02	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Listado DEO Objetivos del Sistema	1.03 1.03.01 1.03.02	Si no está vacía	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
DEO identificados	
DEO por objetivo del sistema	
DEO por requerimiento de software	

Observaciones:

En esta actividad se confecciona el vínculo *generates*. La diferencia entre el documento Propuestas de Cambio y el Listado DEO Objetivos del Sistema radica en que en el primer documento se detallan hechos que afectan a los requerimientos. En cambio, en el Listado DEO Objetivos del Sistema se detallan hechos que no afectan a los requerimientos. Si este listado presenta algún ítem, el proceso continúa dependiendo de los errores encontrados en la actividad 1.03.01 y/o 1.03.02.

La siguiente tabla muestra los distintos productos de trabajos generados en cada una de estas actividades.

	Actividad	**Input**	**Origen**	**Output**	**Destino**
1.00	Modelizar la Organización	Información del dominio a modelar	ORG	Especificación Objetivos del Sistema (validada)	2.00 2.01 2.01.01
		Información Fundamentación	5.00	Listado de CSFs	2.00 2.01 2.01.03 5.00 5.01 5.02 5.02.02
				Modelos de la Organización	2.00 2.01 2.01.01
				Propuestas de Cambio	2.00 2.02 2.02.02
				Especificación Fundamentación	5.00 5.03
				Identificación Conflicto	5.00 5.01
1.01	Analizar la Organización	Información del dominio a modelar	ORG	Modelos de la Organización	1.02 1.02.01 1.02.02 1.02.03 2.00 2.01 2.01.01 2.01.04
1.02	Desarrollar las Necesidades Organizacionales	Listado DEO Necesidades Organizacionales	1.02.04	Documento de Misión	1.03 1.03.01
		Modelos de la Organización	1.01	Listado DEO Necesidades Organizacionales	1.02.01 1.02.03
				Listado CSFs	2.00 2.01 2.01.03 5.00 5.01 5.02 5.02.02
1.02.01	Confeccionar las Necesidades Organizacionales	Modelos de la Organización	1.01	Síntesis información Necesidades Organizacionales	1.02.02
		Listado DEO Necesidades Organizacionales	1.02.04	Listado Necesidades Organizacionales	1.02.02 1.02.03
1.02.02	Identificar los CSFs de la organización	Modelos de la Organización	1.01	Síntesis de la información CSFs	CA
		Listado DEO CSFs	1.02.02	Listado CSFs	2.00 2.01 2.01.03 5.00 5.01 5.02 5.02.02
		Síntesis información Necesidades Organizacionales	1.02.01		
		Listado Necesidades Organizacionales	1.02.01		
				Listado DEO CSFs	1.01.02
1.02.03	Describir las Necesidades Organizacionales	Modelos de la Organización	1.01	Documento de Misión	1.02.03
		Listado Necesidades Organizacionales	1.02.01		
		Listado CSFs	1.02.02		
		Listado DEO Necesidades Organizacionales	1.02.04		
		Síntesis información CSFs	1.02.02		
		Síntesis información Necesidades Organizacionales	1.02.01		
1.02.04	Validar las Necesidades Organizacionales	Documento de Misión	1.02.02	Listado DEO Necesidades Organizacionales	1.02 1.02.01 1.02.03
				Documento de Misión validado	1.03 1.03.01

1.03	Desarrollar los Objetivos del Sistema	Documento de Misión validado	1.02.03	Propuestas de Cambio	2.00 2.02 2.02.02
		Listado DEO Objetivos del Sistema	1.03.03	Especificación Objetivos del Sistema	2.00 2.01 2.01.01
				Listado DEO Objetivos del Sistema	1.03.03
1.03.01	Confeccionar los Objetivos del Sistema	Documento de Misión validado	1.02.03	Listado Objetivos del Sistema	1.03.02
		Listado DEO Objetivos del Sistema	1.03.03	Síntesis información Objetivos del Sistema	1.03.02
1.03.02	Describir los Objetivos del Sistema	Listado Objetivos del Sistema	1.03.01	Especificación Objetivos del Sistema	1.03.03
		Síntesis información Objetivos del Sistema	1.03.01		
		Listado DEO Objetivos del Sistema	1.03.02		
1.03.03	Validar los Objetivos del Sistema.	Especificación Objetivos del Sistema	1.03.02	Especificación Objetivos del Sistema(validada)	2.00 2.01 2.01.01
				Propuestas de Cambio	2.00 2.02 2.02.02
				Listado DEO Objetivos del Sistema	1.03 1.03.01 1.03.02

Tabla 17 Entradas/salidas actividad Modelizar la Organización

6.2.2 Actividad 2.0 Especificar los Requerimientos de Software

La actividad raíz **Especificar los Requerimientos de Software** involucra la realización de diferentes actividades que tienen como propósito la captura, entendimiento y documentación de los requerimientos. Estas actividades se especifican en:

1. Determinar los Requerimientos de Software a desarrollar
2. Reformar los Requerimientos de Software
3. Documentar los Requerimientos de Software

Dichas actividades implementan las acciones por la cuales se confeccionan las entidades y relaciones que faltan definir del submodelo de **Gestión de los Requerimientos** propuesto por Ramesh. En la primer actividad **-Determinar los Requerimientos de Software a desarrollar -**, se especifican las acciones que involucran la captura (representado por medio de los vínculos *generate y derive*) y modelado de los requerimientos. Un aspecto central en el modelado de los requerimientos consiste en vincularlos con los CSFs de la organización. Tal relación se representa por medio del vínculo *managed_by*. Por otra parte, en la actividad **Reformar los Requerimientos de Software**, se especifican las diversas acciones relacionadas con la modificación de los mismos. Dichas acciones se describen por medio de los vínculos *elaborate, modify, part_of*. En la actividad **Documentar los Requerimientos de Software** se confecciona el documento **Especificación de Requerimientos de Software**. Cabe aclarar que en las distintas iteraciones del proceso de desarrollo se puede alterar el orden de ejecución de las actividades especificadas, o llegado el caso, puede no ser necesaria la puesta en marcha de algunas de ellas.

Código:	**Nombre:**	Modelizar la Organización **Especificar los Requerimientos de Software** Validar la Especificación de Requerimientos de Software Desarrollar la Arquitectura del Sistema
2.00	**Especificar los Requerimientos de Software**	

Objetivos:

Confeccionar el documento Especificación de Requerimientos de Software.

Justificación:

La identificación correcta de los requerimientos, su posterior análisis, documentación y su continua gestión representan uno de los factores críticos para el éxito de cualquier proyecto de desarrollo de software.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Determinar los Requerimientos de Software a desarrollar	Identificar y modelizar los requerimientos de software a desarrollar.	ER
2	Reformar los Requerimientos de Software	Reelaborar los requerimientos de software a desarrollar	ER
3	Documentar los Requerimientos de Software	Elaborar el documento Especificación de Requerimientos de Software	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos Procesador de Textos
ER	Técnicas de elicitación seleccionadas Alto nivel de abstracción Conocimientos del dominio de la aplicación Conocimiento de los estándares referidos al documento SRS	
GS	Visión global de la empresa Conocimientos del dominio de la aplicación	
UO	Conocimientos del dominio de la aplicación	

Técnicas:

Técnicas de elicitación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Listado CSFs	1.02.02	CSFs validados por la gerencia superior	PDT
Especificación Objetivos del Sistema	1.03.03	Especificación validada por la gerencia superior	PDT
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Propuestas de Cambio	1.03.03 3.03	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Información Fundamentación	5.00	Registradas las acciones de fundamentación	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación de Requerimientos de Software	3.00 3.01 3.02 3.03	Especificación de Requerimientos de Software sin validar	PDT
Especificación Conflicto	5.00 5.01	Identificador, descripción y contexto del conflicto	PDT
Especificación Fundamentación	5.00 5.03	Identificador del objeto a justificar	PDT

Definiciones:

Requerimiento de Software

1. *"Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo.*
2. *Condición o capacidad que debe satisfacer o poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.*
3. *Representación documentada de una condición o capacidad como en 1 o 2."* [44]

Restricciones:

La gestión de los Requerimientos de Software está influenciada por los CSFs de la organización.

Variables a medir:

Nombre de la variable

Variables definidas en 2.01, 2.02 y 2.03

Observaciones:

En esta actividad se confecciona el documento **Especificación de Requerimientos de Software**. Al realizar dicha actividad pueden originarse conflictos, o presentarse hechos o acciones que necesitan ser justificadas. Ambas circunstancias deben estar debidamente documentadas.

[44] STD 610.12 IEEE Standard Glossary of Software Engineering Terminology 1990. Pág 62.

6.2.2.1 Actividad 2.01 Determinar los Requerimientos de Software a desarrollar

En actividades anteriores se han identificado y modelado los **Objetivos del Sistema**, que están vinculados al logro de las **Necesidades Organizacionales**. Como dichos objetivos están descriptos con alto grado de abstracción, no pueden implementarse (confección del diseño y/o codificación). Es por este motivo que en la actividad **Determinar los Requerimientos de Software a desarrollar** se especifican las acciones mediante las cuales se identifican a los requerimientos a partir de los objetivos del sistema. Este hecho se representa a través del vínculo *generate*. Así mismo, a partir de los requerimientos ya identificados, se derivan nuevos requerimientos que se detallan por medio del vínculo *derive*. La implementación de estos vínculos permite detallar la evolución de los requerimientos. La continua elaboración y entendimiento de los requerimientos se especifican en las siguientes actividades:

1. Recolectar información del dominio del problema
2. Identificar los Requerimientos de Software
3. Establecer los vínculos con los CSFs
4. Describir los Requerimientos de Software

El hecho de haber establecido los vínculos (*managed_by*) entre los requerimientos y los CSFs de la organización permitirá gestionar el grado de detalle con que se describan a los requerimientos. Tales descripciones abarcan la identificación (*based_on*) de los **Mandates** necesarios para la elaboración de los mismos, el establecimiento de las dependencias entre los requerimientos y la confección de escenarios.

Código:	**Nombre:**	Especificar los Requerimientos de Software
2.01	**Determinar los Requerimientos de Software a desarrollar**	**Determinar los Requerimientos de Software a desarrollar** Reformar los Requerimientos de Software Documentar los Requerimientos de Software

Objetivos:

Identificar y modelizar los requerimientos de software a desarrollar.

Justificación:

Al finalizar esta actividad se debe contar con el listado de requerimientos a implementar, conjuntamente con la debida descripción de estos. Dichas descripciones permitirán comprender el significado y alcance de los requerimientos identificados.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Recolectar información del dominio del problema	Comprender el domino del problema	ER
2	Confeccionar los Requerimientos de Software	Establecer los requerimientos de software a implementar	ER
3	Establecer los vínculos con los CSFs	Identificar y describir los vínculos entre los requerimientos y los CSFs	ER
4	Describir los Requerimientos de Software	Modelizar los requerimientos de software identificados	ER

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema
GS	Conocimiento específico del dominio del problema Expectativas del sistema a modelar
UO	Conocimiento específico del dominio del problema Expectativas del sistema a modelar

Tecnológicos:

Planilla de Cálculos
Procesador de Textos

Técnicas:

Escenarios
Entrevistas

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Listado CSFs	1.02.02	CSFs validados por la gerencia superior	PDT
Especificación Objetivos del Sistema	1.03.03	Especificación validada por la gerencia superior	PDT
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listado Requerimientos	2.02 2.02.01 2.02.02 2.03	Completa la información requerida en la definición del listado	INF
Descripciones Requerimientos	2.02 2.02.01 2.02.02 2.03	Completos los componentes de información dependiendo si el requerimiento se vincula con los CSFs	INF

Definiciones:

Mandate: Representa las entidades en las que se basa el requerimiento para su elaboración. Abarca tanto los estándares, políticas o métodos implementados por la organización

Restricciones:

Para modelar los Requerimientos de Software sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[45].
El grado de detalle de las descripciones de los Requerimientos de Software depende si se vinculan con los CSFs.
Las Restricciones del Sistema se tratan como Requerimientos de Software.

Variables a medir:

Nombre de la variable	
Documentos consultados.	
Personal entrevistado.	
Entrevistas realizadas por personal.	
Actividades identificadas por Objetivo del Sistema/Requerimiento de Software	
Restricciones identificadas	
Mandates identificados por actividad/ categoría	
Requerimientos identificados	
Vínculos *generate* identificados	

[45] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Vínculos *derive* identificados	
Vínculos *is_a* (Restricción identificados)	
Vínculos *based_on* identificados/ categorías	
Vínculos *rmanaged_by* identificados	
Vínculos *depend_on* identificados	
Vínculos *describe* identificados/eliminados/modificados/aceptados	

Observaciones:

En esta actividad se confecciona la entidad **Requerimientos de Software**. En la primera iteración se generan los requerimientos desde la Especificación de los Objetivos del Sistema. En posteriores iteraciones, se derivan los requerimientos desde la **Especificación de Requerimientos de Software**. Exceptuando el caso que en iteraciones tardías se modifiquen o adicionen objetivos del sistema. En dicho caso, se deben derivar los requerimientos a partir del objetivo identificado.

Código:	**Nombre:**	Determinar los Requerimientos de Software a desarrollar **Recolectar información del dominio del problema** Confeccionar los Requerimientos de Software Establecer los vínculos con los CSFs Describir los Requerimientos de Software
2.01.01	**Recolectar información del dominio del problema**	

Objetivos:

Comprender el domino del problema

Justificación:

La información que se recolecta permite, en actividades posteriores, la identificación y modelado de los requerimientos de software.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Determinar y recolectar la información	Obtener la información necesaria para poder identificar y modelar a los requerimientos de software	ER
2	Identificar las Restricciones que afectan al sistema	Elaborar el listado restricciones	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos Procesador de Textos
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema	
GS	Conocimiento referido a los estándares, políticas y métodos implementados por la organización	
UO	Conocimiento referido a los estándares, políticas y métodos implementados por la organización	

Técnicas:

Determinar y recolectar la información. Las acciones que orientan la puesta en marcha de la actividad son:

1. Elaborar una lista de las actividades que involucran la realización de los objetivos del sistema y/o de los requerimientos.
2. Identificar el personal de la organización que estén afectados a las actividades identificados.
3. Entrevistar al personal identificado con el propósito de recabar la información que describa las actividades identificadas.

Identificar las restricciones que afectan al sistema. Con la información obtenida hasta el momento se debe elaborar un listado con las restricciones que afectan al desarrollo del sistema.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Modelos de la Organización	1.01	Estructura, objetivos, políticas y estrategias de la organización identificadas y comprendidas	PDT
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Especificación Objetivos del Sistema	1.03.03	Especificación validada por la gerencia superior	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Síntesis Información	2.01.02 2.01.04	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	CON
Listado Actividades	2.01.02	Todas las actividades deben vincularse con un objetivo del sistema o un requerimiento	INF
Listado Restricciones	2.01.02	Identificador y denominación de la restricción	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
Documentos consultados.	
Personal entrevistado.	
Entrevistas realizadas por personal.	

Actividades identificadas por Objetivo del Sistema/Requerimiento de Software	
Restricciones identificadas	
Mandates identificados por categoría	

Observaciones:

Código:	**Nombre:**	Determinar los Requerimientos de Software a desarrollar Recolectar información del dominio del problema **Confeccionar los Requerimientos de Software** Establecer los vínculos con los CSFs Describir los Requerimientos de Software
2.01.02	**Confeccionar los Requerimientos de Software**	

Objetivos:

Establecer los requerimientos de software a implementar

Justificación:

En esta actividad se genera el listado de los requerimientos a desarrollar. En dicho listado se debe especificar el origen, como así también los mandates a cumplimentar para la elaboración de los mismos.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar la documentación elaborada	Comprender cada uno de los ítems que integra el listado de actividades y restricciones	ER
2	Identificar los Requerimientos de Software	Elaborar el listado requerimientos identificados	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
ER	Conocimiento del dominio del problema	

Técnicas:

Análisis de la documentación

Identificar los Requerimientos de Software. A partir de los listados referidos a las actividades y restricciones se debe:

1. Elaborar una lista con los requerimientos de software que se identifican.
2. Por cada elemento de la lista confeccionar las respectivas entidades **Requerimientos de Software** y completar los componentes **Identificador y Descripción**
3. Por cada requerimiento que integra la lista se debe verificar:
   - Si se identificó a partir de un Objetivo del Sistema, se debe registrar el vínculo *generate* con la información correspondiente tanto en el componente **Origen** del respectivo requerimiento**,** como así también en el componente **Genera a** del Objetivo del Sistema que lo genera.
   - Si se identificó a partir de un requerimiento, se debe registrar el vínculo *derive* con la información correspondiente en el componente de información **Origen** del respectivo requerimiento.
   - Si se necesita algún Mandate para su elaboración, se debe registrar el vínculo *based_on* con la información correspondiente en el componente **Basado en** del respectivo requerimiento. Al especificar el **Mandate** también se debe crear el vínculo *is_a*, el cual determina si es un **Estándar, Política o Procedimiento.**
4. Los requerimientos identificados que no posean el vínculo *generate o derive,* se eliminan de la lista.
5. Las restricciones identificadas deben ser registradas y descriptas como requerimientos. El componente **Origen** del requerimiento debe poseer el vínculo *is_a* con la correspondiente información.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Síntesis información	2.01.01	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado Actividades	2.01.01	Todas las actividades deben vincularse con un objetivo del sistema o requerimiento	INF
Listado Restricciones	2.01.01	Identificador y denominación de la restricción	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listado Requerimientos	2.02.03	Información del listado excepto el ítem Gestionado Por.	INF

Definiciones:

Restricciones:

Un Objetivo del Sistema puede generar uno o muchos Requerimientos de Software.
Un Requerimiento de Software debe estar generado por un sólo Objetivo del Sistema o derivar sólo de un requerimiento.

Variables a medir:

Nombre de la variable	
Requerimientos identificados	
Vínculos *generate* identificados	
Vínculos *derive* identificados	
Vínculos *is_a* (Restricciones identificadas)	
Vínculos *based_on* identificados por categorías	

Observaciones:

En esta actividad se confeccionan los vínculos *generate, derive, based_on, is_a*.

Código:	**Nombre:**	Determinar los Requerimientos de Software a desarrollar Recolectar información del dominio del problema Confeccionar los Requerimientos de Software **Establecer los vínculos con los CSFs** Describir los Requerimientos de Software
2.01.03	**Establecer los vínculos con los CSFs**	

Objetivos:

Identificar y describir los vínculos entre los requerimientos y los CSFs

Justificación:

Poseer esta información permitirá gestionar a los requerimientos a través de las cuestiones que son críticas para la organización.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Especificar los recursos afectados en la elaboración de los Requerimientos de Software	Elaborar una lista de los recursos afectados al desarrollo de los requerimientos.	ER
2	Comparar los recursos identificados con los CSFs	Identificar las relaciones entre los requerimientos y los CSFs.	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
ER	Técnicas para realizar entrevistas Conocimiento específico del dominio del problema	
UO	Conocimiento específico del dominio del problema	

Técnica:

Entrevistas

Comparar los recursos identificados con los CSFs. Si un recurso que se ha identificado para la elaboración del requerimiento se encuentra en el **Listado de los CSFs** se debe confeccionar el vínculo *managed_by* y debe ser registrado en el componente de información **Gestionado por** del requerimiento y en el componente **Gestiona a** del CSF respectivo.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Listado CSFs	1.02.02	CSFs validados por la Gerencia Superior	PDT
Listado Requerimientos	2.01.02	Información del listado excepto el ítem Gestionado Por.	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listado Requerimientos	2.01.04 2.02 2.02.01 2.02.02 2.03	Información del listado completa.	INF

Definiciones:

CSFs: representan todos aquellos recursos de la organización identificados como críticos para el éxito del proyecto [Ramesh *et al,* 2001].

Restricciones:

Para modelar los CSFs sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[46].

Variables a medir:

Nombre de la variable	
Vínculo *managed_by* identificados	
Entrevistas realizadas	
Entrevistas realizadas por personal	

Observaciones:

En esta actividad se confecciona el vínculo *managed_by*.

[46] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Código:	**Nombre:**	Determinar los Requerimientos de Software a desarrollar Recolectar información del dominio del problema Confeccionar los Requerimientos de Software Establecer los vínculos con los CSFs **Describir los Requerimientos de Software**
2.01.04	**Describir los Requerimientos de Software**	

Objetivos:

Modelizar los requerimientos de software Identificados.

Justificación:

La información que se produce en esta actividad permite representar a los requerimientos que han sido identificados

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Caracterizar los Requerimientos de Software	Completar los componentes de información de los requerimientos de software	ER
2	Desarrollar los escenarios de los Requerimientos de Software	Describir con mayor grado de detalle a los requerimientos de software.	ER
3	Identificar las dependencias entre los Requerimientos de Software	Determinar la existencia de dependencia entre los requerimientos identificados.	ER

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas de construcción de escenarios Técnicas para realizar entrevistas Conocimiento del dominio del problema
UO	Conocimiento específico del dominio del problema Conocimiento referido a la técnica de escenarios

Tecnológicos:

Planilla de Cálculos
Herramientas que soporte escenario

Técnicas:

Escenarios

Caracterizar los Requerimientos de Software. Por cada ítem del **Listado Requerimientos** se debe completar los restantes componentes de información dependiendo si estos se vinculan con algún CSFs.

Desarrollar los Escenarios de los Requerimientos de Software. Por cada requerimiento que deba ser descripto con mayor grado de detalle se debe:

1. Confeccionar el escenario pertinente
2. Confeccionar el vínculo *describe* y registrarlo en el componente **Modelado por** del respectivo requerimiento.
3. Verificar que los escenarios confeccionados estén elaborados según el estándar establecido.

Identificar las dependencias entre los Requerimientos de Software.

1. Seleccionar el criterio para establecer las dependencias.
2. Identificar las dependencias que cumplan con el criterio seleccionado.
3. Por cada dependencia identificada se confecciona el vínculo *depend_on.* Dicho vínculo debe ser registrado en el componente **Depende de** pertenecientes a los requerimientos afectados en tales relaciones.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Síntesis información	2.01.01	Resumen del conocimiento adquirido en la actividad. Incluye las minutas de las distintas reuniones	INF
Listado Requerimientos	2.01.03	Información del listado completa.	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Requerimientos	2.02 2.02.01 2.02.02 2.03	Completos los componentes de información dependiendo si se vinculan con los CSFs	INF

Definiciones:

Para el desarrollo de los Escenarios se toma como parámetro lo formalizado para la actividad **3.0 Construcción de Escenarios** del proceso LEC definido en [Zuñiga, 2004].

Restricciones:

La granularidad de las descripciones de los requerimientos se realiza dependiendo si se vinculan con los CSFs.
Para modelar los Requerimientos sólo se utilizan los escenarios y los componentes de información especificados en el template de la entidad[47].

Variables a medir:

Nombre de la variable	
Vínculos *depend_on* identificados	
Vínculos *describe* identificados/eliminados/modificados/aceptados	

Observaciones:

En esta actividad se confeccionan los vínculos *describe, depend_on*. Dependiendo del grado de entendimiento que se posea, el paso dos puede ser optativo. Al finalizar esta actividad se puede optar por realizar nuevamente la actividad **2.01 Determinar los Requerimientos de Software**, o si hubiese requerimientos que necesiten ser clarificados o modificados se debe continuar con la actividad **2.02 Reformar los Requerimientos de Software**. De lo contrario, se realiza la actividad **2.03 Documentar los Requerimientos de Software.**

[47] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

6.2.2.2 Actividad 2.02 Reformar los Requerimientos de Software

La dificultad que encierra la evolución de los requerimientos radica en tener que identificar y mantener la información relacionada con el cambio producido. Las acciones que detallan el cambio producido se especifican en las siguientes actividades:

1. Redefinir los Requerimientos de Software
2. Modificar los Requerimientos de Software

En la actividad **Determinar los Requerimientos de Software a desarrollar** se han especificado las acciones que detallan la evolución de los requerimientos a partir de cómo fueron identificados. En cambio, en la actividad **Reformar los Requerimientos de Software** se especifican las acciones relacionadas a la evolución de los mismos. Si tal evolución se relaciona al progreso natural del entendimiento – disminución del grado de abstracción en la identificación y descripción de los requerimientos-, se está en presencia de la actividad **Redefinir los Requerimientos de Software**. En dicha actividad, se llevan a cabo las acciones de clarificación (vínculo *elaborate*) y descomposición del requerimiento (vínculo *part_of*). Pero si tal evolución tiene como origen el error, la omisión o deficiencias detectadas en las actividades de validación se debe realizar la actividad **Modificar los Requerimientos de Software**. Tales modificaciones se representan por medio de los vínculos *modify.*

Código:	**Nombre:**	Especificar los Requerimientos de Software Determinar los Requerimientos de Software a desarrollar **Reformar los Requerimientos de Software** Documentar los Requerimientos de Software
2.02	**Reformar los Requerimientos de Software**	

Objetivos:

Reelaborar los requerimientos de software a desarrollar

Justificación:

El poder discriminar los motivos que provocan el cambio en las descripciones de los requerimientos posibilita el entendimiento de la evolución de los mismos.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Redefinir los Requerimientos de Software	Clarificar las descripciones de los requerimientos	ER
2	Modificar los Requerimientos de Software	Realizar las modificaciones advertidas en el documento Propuestas de Cambio	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
ER	Técnicas de elicitación seleccionadas Conocimiento del dominio del problema	
UO	Conocimiento específico del dominio del problema Expectativas del sistema a modelar	
GS	Conocimiento específico del dominio del problema Expectativas del sistema a modelar	

Técnicas:

Análisis de la documentación
Técnicas de elicitación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Propuestas de Cambio	1.03.03 3.03	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Descripciones Requerimientos	2.01.04	Completos los componentes de información dependiendo si se vinculan con los CSFs	INF

Listado Requerimientos	2.01.03	Información del listado completa.	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listados cambios producidos	2.03	Identificador y denominación del requerimiento, vínculos generados en esta actividad	INF

Definiciones:

Restricciones:

En la actividad **Redefinir los Requerimientos de Software**, sólo se clarifican los componentes de información **Identificador y/o Descripción.**
Sólo se modifican los Requerimientos de Software afectados por el contenido de la Propuestas de Cambio

Variables a medir:

Nombre de la variable	
Vínculos *elaborate* identificados	
Vínculos *part_of* identificados	
Vínculos *depend_on* modificados/eliminados/agregados	
Vínculos *part_of* modificados/eliminados/agregados	
Vínculos *managed_by* modificados/eliminados/agregados	
Requerimientos modificados/eliminados/agregados	

Observaciones:

La jerarquía entre los requerimientos padre/hijos está representada en el metamodelo con el vínculo de dependencia *is_a.* Se ha optado por no implementar este tipo de relación, debido a que con la información especificada en el vínculo *part_of* se puede deducir a dicha jerarquía. Si se contase con una herramienta automatizada, los cambios efectuados en esta actividad se realizarían de manera automática en la **Especificación de Requerimientos de Software** y no se tendría que confeccionar el **Listado cambios producidos**.

Código:	**Nombre:**	Reformar los Requerimientos de Software **Redefinir los Requerimientos de Software** Modificar los Requerimientos de Software
2.02.01	**Redefinir los Requerimientos de Software**	

Objetivos:

Clarificar las descripciones de los requerimientos.

Justificación:

Reelaborar un requerimiento por medio de la clarificación o descomposición permite a los distintos stakeholders que intervienen en el proyecto poseer una mayor comprensión y entendimiento de los mismos.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Reelaborar los a Requerimientos de Software	Detallar los componentes de información de los requerimiento en términos más legibles	ER
2	Descomponer a los Requerimientos de Software	Particionar a los requerimientos en requerimientos con descripciones de menor complejidad	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema	
UO	Conocimiento específico del dominio del problema Expectativas del sistema a modelar	
GS	Conocimiento específico del dominio del problema Expectativas del sistema a modelar	

Técnicas:

Reelaborar los Requerimientos de Software.

1. Seleccionar el requerimiento de software que necesite ser reelaborado.
2. Recolectar y analizar la información relacionada al requerimiento seleccionado.
3. Con la información obtenida, se debe reelaborar el componente **Identificador textual** y/ o el componente **Descripción.**
4. Crear una nueva entidad **Requerimiento de Software** con los componentes de información elaborados en el ítem 3.
5. Registrar en el componente **Estado** del requerimiento seleccionado en el ítem 1, el vínculo *elaborate*. El mismo se debe completar con la información correspondiente.
6. Analizar cuáles de los aspectos informativos del requerimiento seleccionado en el ítem 1, que tendrán vigencia en el nuevo requerimiento.
7. Completar los restantes componentes de información del requerimiento. En el componente **Origen** del nuevo requerimiento, se debe registrar el vínculo *elaborate* especificado en el ítem 5.
8. Registrar lo confeccionado en el **Listado cambios producidos**.
9. Volver al ítem 1, si es que todavía hay requerimientos que necesiten ser reelaborados.

Particionar a los Requerimientos Software.

1. Seleccionar el requerimiento de software que necesite ser particionado.
2. Recolectar y analizar la información relacionada al requerimiento seleccionado.
3. Elaborar una lista de las actividades que involucran la realización del requerimiento de software seleccionado.
4. Analizar y describir cada una de las actividades identificadas
5. A partir de las actividades detalladas en el paso 3 y 4 se debe confeccionar las distintas entidades de requerimiento de

software que la describen. Por cada requerimiento identificado se debe completar los componentes de información **Identificador y Descripción**.
6. Confeccionar los vínculos *part_of* los cuales detallan la descomposición realizada. Además se debe registrar dichos vínculos tanto en el componente de información **Compuesto por** del requerimiento seleccionado en el ítem 1, como en el componente **Origen** de los nuevos requerimientos identificados.
7. Analizar y completar los componentes de información de los requerimientos identificados en el paso 5.
8. Registrar lo confeccionado en el **Listado cambios producidos**.
9. Volver al paso 1, si es que todavía hay requerimientos que necesiten ser particionados.

Input:

Nombre	Origen	Criterio de Entrada: Estado o Condición	Tipo
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Descripciones Requerimientos	2.02.04	Completos los componentes de información dependiendo si se vinculan con los CSFs	NF
Listado Requerimientos	2.02.03	Información completa, sólo los requerimientos que se vinculan con los CSFs	NF

Output

Nombre	Destino	Criterio de Salida: Estado o Condición	Tipo
Listados cambios producidos	2.03	Completos los componentes del listado	INF

Definiciones:

Restricciones:

Un requerimiento puede ser reemplazado por sólo un requerimiento de software.

Variables a medir:

Nombre de la variable	
Vínculos *elaborate* identificados	
Vínculos *part_of* identificados	

Observaciones:

En esta actividad se confeccionan los vínculos *elaborate, part_of*. El resultado de la actividad de elaboración produce el reemplazo de un requerimiento por otro. Podría suceder que esta actividad sólo involucre un cambio de forma de redacción del requerimiento y por ello no se alteran las dependencias y las relaciones con los CSFs que el requerimiento poseía. Pero, no hay que descartar el hecho, que al realizar la reelaboración, se deba llevar a cabo nuevamente las actividades **2.01.03 Establecer las vínculos con las CSFs y/o 2.01.04 Describir los Requerimientos de Software.**

Código:	Nombre:	Reformar los Requerimientos de Software Redefinir los Requerimientos de Software **Modificar los Requerimientos de Software**
2.02.02	**Modificar los Requerimientos de Software**	

Objetivos:

Realizar las modificaciones advertidas en el documento Propuestas de Cambio.

Justificación:

Esta actividad detalla las acciones necesarias para llevar a cabo lo detallado en el documento Propuestas de Cambio elaborado en las actividades de validación.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Modificar los aspectos descriptivos de los Requerimientos de Software	Reelaborar el contenido de algún componente de información del requerimiento	ER
2	Eliminar aspectos descriptivos de los Requerimientos de Software	Eliminar el contenido de algún componente de información del requerimiento	ER

Recursos:

Humanos:

Rol:	Capacitación requerida:
ER	Técnicas para realizar entrevistas Conocimiento del dominio del problema
UO	Conocimiento específico del dominio del problema Expectativas del sistema a modelar
GS	Conocimiento específico del dominio del problema Expectativas del sistema a modelar

Tecnológicos:

Planilla de Cálculos

Técnicas:

Modificar los aspectos descriptivos de los Requerimientos de Software. En esta actividad se puede realizar las siguientes acciones:
1. Modificar el componente **Identificador y/o Descripción.**
2. Agregar o modificar el escenario que modele al requerimiento de software.
3. Agregar, o modificar las relaciones con los CSFs.
4. Agregar o modificar la composición del requerimiento de software.
5. Agregar o modificar las relaciones de dependencias que posee el requerimiento.
6. Agregar o modificar cualquier aspecto restante que describe al requerimiento.
7. Registrar lo confeccionado en el **Listado cambios producidos**

Eliminar aspectos descriptivos de los Requerimientos de Software: En esta actividad se puede realizar las siguientes acciones:

1. Eliminar un requerimiento.
2. Eliminar el escenario que modele a un requerimiento.
3. Eliminar las relaciones con los CSFs.
4. Eliminar la composición del requerimiento de software.
5. Eliminar las relaciones de dependencias que posee el requerimiento.
6. Eliminar cualquier aspecto restante que describe al requerimiento.
7. Registrar lo confeccionado en el **Listado cambios producidos**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Propuestas de Cambio	1.03.03 3.03	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Descripciones Requerimientos	2.02.04	Completos los componentes de información dependiendo si se vinculan con los CSFs	INF
Especificación de Requerimientos de software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listados cambios producidos	2.03	Completos los componentes del listado	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
Vínculos *depend_on* modificados/eliminados/agregados	
Vínculos *part_of* modificados/eliminados/agregados	
Vínculos *managed_by* modificados/eliminados/agregados	
Requerimientos modificados/eliminados/agregados	

Observaciones:

En esta actividad se confeccionan los vínculos

6.2.2.3 Actividad 2.03 Documentar los Requerimientos de Software

Una vez finalizadas las actividades de identificación y modelado de los requerimientos se debe confeccionar un documento **Especificación de Requerimientos de Software.** Este documento posee toda la información recolectada y analizada hasta el momento, la cual se detalla de manera estructurada y consistente.

Código:	**Nombre:**	Especificar los Requerimientos de Software Determinar los Requerimientos de Software a desarrollar Reformar los Requerimientos de Software **Documentar los Requerimientos de Software**
2.03	**Documentar los Requerimientos de Software**	

Objetivos:

Elaborar el documento Especificación de Requerimientos de Software

Justificación:

Con la información producida hasta el momento se debe confeccionar un documento en donde se registren y describan los requerimientos de software identificados. Este documento permitirá evaluar el producto software desarrollado.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Confeccionar la Especificación de Requerimientos de Software	Crear un documento que contenga las descripciones de los requerimientos identificados y la información necesaria para su comprensión	ER

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Procesador de Textos
ER	Conocimiento de los estándares referidos al documento SRS	

Técnica:

Confeccionar la Especificación de los Requerimientos de Software

- En la primera iteración del proceso de desarrollo se elabora la especificación de requerimientos desde el Listado de los Requerimientos y el Listado de los Cambios producidos.
- En iteraciones posteriores se elabora la especificación tomando como base la Especificación de Requerimientos de Software de la iteración anterior, integrando a esta los ítems del Listado de los Requerimientos y el Listado de los cambios producidos en la iteración.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Descripciones Requerimientos	2.02.04	Completos los componentes de información de los requerimientos dependiendo si se vinculan con los CSFs	INF
Listado cambios producidos	2.02.01 2.02.02	Completos los componentes del listado	INF
Listado Requerimientos	2.02.03	Información completa, sólo los requerimientos que se vinculan con los CSFs	INF
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Output			
Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación de Requerimientos de Software	3.00 3.01 3.02 3.03	Con toda la información producida en las actividades posteriores	PDT
Definiciones:			
Restricciones:			
Variables a medir:			
Nombre de la variable			
Cantidad de requerimientos adicionados/eliminados/modificados			
Observaciones:			
En esta actividad se confecciona el documento **Especificación de Requerimientos de Software –SRS-**			

La siguiente tabla muestra los distintos productos de trabajos generados en cada una de estas actividades.

	Actividad	Input	Origen	Output	Destino
2.00	Especificar los Requerimientos de Software	SRS (validada)	3.03	SRS	3.00 3.01 3.02 3.03
		Listado CSFs	1.02.02		
		Modelos de la Organización	1.01		
		Especificación Objetivos del Sistema	1.03.03	Especificación Conflicto	5.00 5.01
		Propuesta s de Cambio	1.03.03 3.03	Especificación Fundamentación	5.00 5.03
		Información Fundamentación	5.00		
2.01	Determinar los Requerimientos de Software	Listado CSFs	1.02.02	Listado Requerimientos	2.02 2.02.01 2.02.02 2.03
		Modelos de la Organización	1.01		
		Especificación Objetivos del Sistema	1.03.03		
		SRS (validada)	3.03	Descripciones Requerimientos	2.02 2.02.01 2.02.02 2.03
2.01.01	Recolectar información del dominio del problema	Especificación Objetivos del Sistema	1.03.03	Listado Actividades	2.01.02 2.01.04
		SRS (validada)	3.03	Síntesis Información	2.01.02
		Modelos de la Organización	1.01	Listado Restricciones	2.01.02
2.01.02	Identificar los Requerimientos de Software	Listado Actividades	2.01.01	Listado Requerimientos	2.01.03
		Listado Restricciones	2.01.01		
		Síntesis de la Información	2.01.01		
2.01.03	Establecer los vínculos con los CSFs	Listado de los CSFs	1.02.02	Listado Requerimientos	2.01.04 2.02 2.02.01 2.02.02 2.03
		Listado Requerimientos	2.01.02		
2.01.04	Describir los Requerimientos de Software	Síntesis información	2.01.01	Descripciones Requerimientos	2.02 2.02.01 2.02.02 2.03
		Listado Requerimientos	2.01.03		
2.02	Reformar los Requerimientos de Software	SRS (validada)	3.03	Listado cambios producidos	2.03
		Propuestas de Cambio	1.03.03 3.03		
		Descripciones Requerimientos	2.01.04		
		Listado Requerimientos	2.01.03		

2.02.01	Redefinir los Requerimientos de Software	Descripciones Requerimientos	2.01.04	Listado cambios producidos	2.03
		Listado Requerimientos	2.01.03		
		SRS (validada)	3.03		
2.02.02	Modificar los Requerimientos de Software	SRS (validada)	3.03	Listado cambios producidos	2.03
		Propuestas de Cambio	1.03.03 3.03		
		Descripciones Requerimientos	2.01.04		
2.03	Documentar los Requerimientos de Software	Listado cambios producidos	2.02.01 2.02.02	SRS	3.00 3.01 3.02 3.03
		Descripciones Requerimientos	2.01.04		
		Listado Requerimientos	2.01.03		
		SRS (validada)	3.03		

Tabla 18 Entradas/salidas actividad Especificar los Requerimientos de Software

6.2.3 Actividad 3.0 Validar la Especificación de Requerimientos de Software

Las actividades de validación no fueron descriptas con el mismo grado de detalle que las demás actividades que integran la especificación. Esto se debe a que las mismas se caracterizan por carecer de estructura, ya que no se puede aplicar un algoritmo para realizar tal actividad [Loucopoulos *et al*, 1995]. Las actividades que integran **Validar la Especificación de Requerimientos de Software** se detallan en:

1. Determinar los Requerimientos de Software a validar
2. Elaborar los CVPs
3. Validar los Requerimientos seleccionados

Las acciones que se especifican corresponden a lo definido en el submodelo de **Compliance Verification** descripto en [Ramesh *et al*, 2001]. Este submodelo describe tanto las actividades que posibilitan la validación de los requerimientos como también la del diseño elaborado. La especificación realizada se limita sólo a la validación de los requerimientos, motivo por el cual se han excluido las acciones involucradas en la confección del vínculo *verify* definido en el submodelo de **Compliance Verification**.

Los resultados obtenidos al realizar las validaciones deben estar documentados en las respectivas entidades **DEO**. En dichas entidades se registran las deficiencias, errores u omisiones que han sido detectadas. Una vez registrados tales hechos en la entidad DEO se confeccionan los vínculos *generate.* Dichos vínculos permiten detallar las relaciones existentes entre las entidades DEOs y los CVPs. Finalizadas las validaciones se debe confeccionar el documento **Propuestas de Cambio**. En este documento se especifican los DEOs que han sido confeccionados, como así también toda aquella información elaborada durante la realización de las actividades de validación.

Código:	**Nombre:**	Modelizar la Organización Especificar los Requerimientos de Software **Validar la Especificación de Requerimientos de Software** Desarrollar la Arquitectura del Sistema
3.00	**Validar la Especificación de Requerimientos de Software**	

Objetivos:

Asegurar que los requerimientos especificados satisfagan las necesidades de la organización

Justificación:

Los errores producidos durante la especificación de los requerimientos conducen a costos y retrabajo excesivos en las restantes actividades del proceso de desarrollo, si es que los mismos no son corregidos una vez finalizada tal especificación.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Determinar los Requerimientos de Software a validar	Identificar los requerimientos de software a validar	LP
2	Confeccionar los CVPs	Elaborar los distintos CVPs	TE
3	Validar los Requerimientos seleccionados	Realizar las validaciones de los elementos seleccionados con sus respectivos CVPs	TE

Recursos:

Humanos:

Rol:	Capacitación requerida:
LP	Conocimiento del cronograma de las actividades
TE	Técnicas de validación seleccionadas Conocimiento de los estándares referidos a las técnicas de validación
UO	Conocimientos del dominio de la aplicación
GS	Conocimientos del dominio de la aplicación

Tecnológicos:

Lenguaje de Programación
Planilla de Cálculos

Técnicas:

Técnicas de Validación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación de Requerimientos de Software	2.03	Con toda la información producida en las actividades posteriores	PDT
Información Fundamentación	5.00	Registradas las acciones de fundamentación	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Propuestas de Cambio	2.00 2.02 2.02.02	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT
Especificación de Requerimientos de Software	2.00 2.01 2.01.01 2.02 2.02.01 2.02.02 2.03 4.00 4.01 4.02 4.02.01 4.03 4.03.01 4.04 4.04.01	Especificación validada por la Gerencia superior y los usuarios operativos	PDT
Especificación del Conflicto	5.00 5.01	Identificador, descripción y contexto del conflicto	PDT
Especificación de la Fundamentación	5.00 5.03	Identificador del objeto a justificar	PDT

Definiciones:

CVP: Entidad en donde se detalla la información relacionada con el procedimiento confeccionado para realizar la validación

Restricciones:

Las actividades que se detallan tienen como alcance sólo la validación del documento SRS

Variables a medir:

Nombre de la variable

Variables definidas en 3.01,3.02 y 3.03

Observaciones:

En esta actividad se confecciona el documento **Propuestas de Cambio.**

6.2.3.1 Actividad 3.01 Determinar los Requerimientos de Software a validar

El propósito de esta actividad es poder seleccionar (previamente se ha establecido algún criterio de selección) los requerimientos que deben ser validados.

<table>
<tr><td>Código:
3.01</td><td colspan="2">Nombre:
Determinar los Requerimientos de Software a validar</td><td>Validar la Especificación de Requerimientos de Software
Determinar los Requerimientos de Software a validar
Confeccionar los CVPs
Validar los requerimientos seleccionados</td></tr>
<tr><td colspan="4">Objetivos:
Identificar los requerimientos de software a validar</td></tr>
<tr><td colspan="4">Justificación:
Esta actividad sirve de punto de partida dado que se identifican los distintos requerimientos a validar.</td></tr>
<tr><td colspan="4">Actividades:
<table>
<tr><th>Paso</th><th>Nombre Actividad</th><th>Objetivo</th><th>Rol</th></tr>
<tr><td>1</td><td>Seleccionar los requerimientos a validar</td><td>Elaborar el listado requerimientos a validar</td><td>LP</td></tr>
</table></td></tr>
<tr><td colspan="4">Recursos:
<table>
<tr><td colspan="2">Humanos:</td><td>Tecnológicos:</td></tr>
<tr><td>Rol:</td><td>Capacitación requerida:</td><td rowspan="2">Planilla de Cálculos</td></tr>
<tr><td>LP</td><td>Conocimiento del cronograma de las actividades</td></tr>
</table></td></tr>
<tr><td colspan="4">Técnicas:
Seleccionar los Requerimientos a validar
• En primer término, se debe escoger el criterio a emplear para realizar la selección de los requerimientos a validar.
• Posteriormente, se confecciona una lista con los requerimientos que cumplimentan el criterio seleccionado.</td></tr>
<tr><td colspan="4">Input:
<table>
<tr><th>Nombre</th><th>Origen</th><th>Criterio de Entrada: Estado o Condición</th><th>Tipo</th></tr>
<tr><td>Especificación de Requerimientos de software</td><td>2.03</td><td>Con toda la información producida en las actividades posteriores</td><td>PDT</td></tr>
</table></td></tr>
<tr><td colspan="4">Output
<table>
<tr><th>Nombre</th><th>Destino</th><th>Criterio de Salida: Estado o Condición</th><th>Tipo</th></tr>
<tr><td>Listado Requerimientos a validar</td><td>3.02</td><td>Identificador del requerimiento seleccionado</td><td>INF</td></tr>
</table></td></tr>
<tr><td colspan="4">Definiciones:
.</td></tr>
<tr><td colspan="4">Restricciones:
Sólo se seleccionan los requerimientos que se relacionen con los CSFs.</td></tr>
<tr><td colspan="4">Variables a medir:
<table>
<tr><td>Nombre de la variable</td><td></td></tr>
<tr><td>Cantidad de Requerimientos de Software seleccionados</td><td></td></tr>
</table></td></tr>
<tr><td colspan="4">Observaciones:
Si existiera algún motivo para validar a un determinado requerimiento (falta de entendimiento, mayor complejidad, etc.) se deberá incorporar al mismo al listado correspondiente.</td></tr>
</table>

6.2.3.2 Actividad 3.02 Confeccionar los CVPs

Una vez identificados los requerimientos a ser validados, se confeccionan los CVPs más adecuados para efectuar tal validación. Estas acciones se describen en:

1. Seleccionar las técnicas de validación
2. Determinar los Mandates
3. Determinar los recursos necesarios para la elaboración de los CVPs
4. Confeccionar los CVPs

Al confeccionar los CVPs se deben identificar por medio de los vínculos *developed_for* los requerimientos que motivaron su elaboración. Además se identifica la **técnica** empleada (vínculo *is_a*), como así también los **mandates** (vínculo *based_on*) y **recursos** necesarios (vínculo *used_by*) para realizar la validación.

Código:	**Nombre:**	Validar la Especificación de Requerimientos de Software Determinar los Requerimientos de Software a validar **Confeccionar los CVPs** Validar los requerimientos seleccionados
3.02	**Confeccionar los CVPs**	

Objetivos:

Elaborarlos distintos CVPs

Justificación:

Esta actividad tiene como propósito construir los diversos CVPs para los requerimientos que han sido seleccionados. Además se debe haber identificado los mandates y recursos necesarios para poder llevar a cabo el CVP realizado

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar las técnicas de validación	Identificar la técnicas de validación más adecuadas	TE
2	Determinar los Mandates a cumplimentar	Identificar los mandates necesarios para la elaboración de los CVPs	TE
3	Determinar los recursos	Identificar los recursos necesarios para la elaboración y puesta en marcha de los CVPs	TE
4	Construir los CVPs	Crear los CVPs para los requerimientos seleccionados	TE

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos Lenguajes de Programación
TE	Técnicas de validación seleccionadas Conocimiento de los estándares referidos a las técnicas de validación	

Técnicas:

Seleccionar las técnicas de validación. Por cada requerimiento detallado en el **Listado requerimientos a validar** se debe:

1. Confeccionar una entidad **CVP** y completar el componente **Identificador**. Además se debe realizar una breve descripción del motivo de la validación.
2. Confeccionar el vínculo *develop_for* , el cual se registra en el componente **Desarrollado para**.
3. Seleccionar la técnica de validación a realizar. Para representar este hecho se confecciona el vinculo *is_a.,* el cual debe registrarse en el componente **Modelado por**.

Determinar los Mandates a cumplimentar. Por cada CVP se debe:

1. Identificar si las técnicas seleccionadas deben cumplir con algún mandate.
2. Por cada mandate identificado se confecciona el vinculo *based_on,* y se registra en el componente **Basado en.**

Determinar los Recursos necesarios Por cada CVP se debe:

1. Identificar los recursos necesarios para llevar a cabo la validación.
2. Por cada recurso identificado se confecciona el vínculo *used_by*, y se registra en el componente **Recursos utilizados**.

Construir los CVPs. Dependiendo de la técnica de validación seleccionada se debe realizar la modelización de los distintos CVPs que han sido identificados. Los identificadores de tales modelizaciones deben ser registradas en el componente **Modelado por** del respectivo CVP.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Listado Requerimientos a validar	3.01	Identificador del requerimiento seleccionado	INF
Especificación de Requerimientos de Software	2.03	Con toda la información producida en las actividades posteriores	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
CVPs elaborados	3.03	Completo los componentes Identificador, descripción, recursos utilizados, desarrollado para, modelado por	INF

Definiciones:

Restricciones:

Para modelar los CVPs se utilizan los componentes de información especificados en el template de la entidad[48].
Un CVP puede ser elaborado sólo mediante las técnicas de prototipado, inspección, simulación o test.

Variables a medir:

Nombre de la variable	
Vínculos *develop_for* identificados/modificados/eliminados	
Vínculos *is_a* identificados/modificados/eliminados	
Vínculos *based_on* identificados/modificados/eliminados	
Vínculos *used_by* identificados/modificados/eliminados	

Observaciones:

En esta actividad se confeccionan las entidades **CVP**, y los vínculos de traceability *develop_for, is_a, based_on, used_by*

[48] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

6.2.3.3 Actividad 3.03 Validar los requerimientos seleccionados

Finalizada las actividades anteriores se realizan las validaciones correspondientes. Los resultados de las mismas deben estar debidamente documentados en el documento **Propuestas de Cambio.** Tales acciones se especifican en:

1. Poner en marcha al CVP
2. Confeccionar la Propuestas de Cambio

Código:	**Nombre:**	Validar la Especificación de Requerimientos de Software Determinar los Requerimientos de Software a validar Confeccionar los CVPs **Validar los requerimientos seleccionados**
3.03	**Validar los requerimientos seleccionados**	

Objetivos:

Realizar las validaciones de los elementos seleccionados con sus respectivos CVPs

Justificación:

Esta actividad permite asegurar que los requerimientos identificados satisfacen tanto a los objetivos del sistema, como a las necesidades organizacionales.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Poner en marcha a los CVPs.	Realizar las validaciones correspondientes	TE
2	Documentar los resultados obtenidos	Registrar los resultados obtenidos	TE

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Lenguaje de Programación Planilla de Cálculos
TE	Técnicas de validación seleccionadas	
UO	Conocimientos del dominio de la aplicación	
GS	Conocimientos del dominio de la aplicación	

Técnicas:

Documentar los resultados obtenidos. Al realizar las validaciones se deben ir registrando los resultados obtenidos. Si se identificase alguna deficiencia, error u omisión se deberá confeccionar una entidad **DEO** que será registrada en el documento **Propuestas de Cambio**. En caso que las validaciones hayan sido efectuadas de manera positiva, se debe registrar la aprobación del documento **Especificación de Requerimiento de Software**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación de los Requerimientos de Software	2.03	Con toda la información producida en las actividades posteriores	PDT
CVPs elaborados	3.02	Completo los componentes Identificador, descripción, recursos utilizados, desarrollado para, modelado por	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Especificación de Requerimientos de Software	2.00 2.01 2.01.01 2.02 2.02.01 2.02.02 2.03 4.00 4.01 4.02 4.02.01 4.03 4.03.01 4.04 4.04.01	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Propuestas de Cambio	2.00 2.02 2.02.02	Errores, omisiones y/o deficiencias identificadas y descriptas en las respectivas entidades DEO	PDT

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable

Vínculos *generate* identificados/modificados/eliminados	
Observaciones:	
En esta actividad se confeccionan los vínculos de traceability *generate*	

La siguiente tabla muestra los distintos productos de trabajos generados en cada una de estas actividades.

	Actividad	Input	Origen	Output	Destino
3.00	Validar la Especificación de Requerimientos de Software	SRS	2.03	Propuestas de Cambio	2.0 2.02 2.02.02
		Información de Fundamentación	5.00		
				SRS(validada)	2.00 2.01 2.01.01 2.02 2.02.01 2.02.02 2.03 4.00 4.01 4.02 4.02.01 4.03 4.03.01 4.04 4.04.01
				Especificación del Conflicto	5.00 5.01
				Especificación de la Fundamentación	5.00 5.03
3.01	Determinar los Requerimien-tos de Software a validar	SRS	2.03	Listado Requerimientos a validar	3.02
3.02	Confeccionar los CVPs	SRS	2.03	CVPs elaborados	3.03
		Listado Requerimientos a validar	3.01		
3.03	Validar los Requerimientos seleccionados	SRS	2.03	SRS (validada)	2.00 2.01 2.01.01 2.02 2.02.01 2.02.02 2.03 4.00 4.01 4.02 4.02.01 4.03 4.03.01 4.04 4.04.01
		CVPs elaborados	3.02	Propuesta s de Cambio	2.00 2.02 2.02.02

Tabla 19 Entradas/salidas actividad Validar la Especificación de Requerimientos de Software

6.2.4 Actividad 4.0 Desarrollar la Arquitectura del Sistema

Como se ha mencionando en el capítulo cinco, la descripción del **submodelo de Design/Allocation** carece de detalles específicos que caractericen al proceso de diseño a realizar. Después de analizar las entidades y relaciones que integran el submodelo mencionado, se concluye que las actividades implicadas en tales descripciones hacen referencia a lo que se denomina **Diseño Arquitectónico.** Teniendo en cuenta tales carencias, se ha optado por especificar la actividad **Desarrollar la Arquitectura del Sistema** de la siguiente manera:

1. Seleccionar los elementos a diseñar
2. Establecer la estructura del Sistema
3. Establecer la estructura de los Subsistemas
4. Establecer la estructura de los Componentes
5. Documentar la arquitectura del Sistema

La actividad **Desarrollar la Arquitectura del Sistema** comienza realizando la selección tanto de los requerimientos de software a diseñar como de las entidades de diseño elaboradas en iteraciones anteriores. Una vez identificados tales elementos se debe determinar la próxima actividad a realizar. Dicha actividad puede contemplar tanto la definición del sistema y subsistemas que lo conforman, como así también la de los distintos componentes que integran a los subsistemas. Al realizar estas actividades se pueden identificar distintas alternativas para la elaboración del sistema, subsistema o componente a implementar. Dichas alternativas se detallan en las respectivas entidades denominadas **Diseño** por medio del vínculo *define*. Independientemente de la entidad confeccionada, se debe detallar la información de asignación de los requerimientos de software. Además se debe identificar la alternativa que ha sido confirmada para su implementación. Tal información se registra en la respectiva entidad **Diseño** por medio del vínculo *create*. Por último, se confecciona el **Documento Arquitectura del Sistema**.

Código:	**Nombre:**	Modelizar la Organización Especificar los Requerimientos de Software Validar la Especificación de Requerimientos de Software **Desarrollar la Arquitectura del Sistema**
4.00	**Desarrollar la Arquitectura del Sistema**	

Objetivos:

Confeccionar el Documento de Arquitectura del Sistema

Justificación:

El diseño arquitectónico permite estructurar y organizar a los requerimientos que integran la Especificación de Requerimientos de Software con la pertinente información de asignación de los requerimientos de software.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar los elementos a diseñar	Confeccionar un listado con los distintos requerimientos a diseñar	LP/ DI
2	Establecer la estructura del Sistema	Describir la estructura del sistema a desarrollar	DI
3	Establecer la estructura de los Subsistemas	Describir la estructura de los subsistemas	DI
4	Establecer la estructura de los Componentes	Describir la estructura de los componentes	DI
5	Documentar la Arquitectura del Sistema	Crear un documento que identifique y describa la arquitectura del sistema a implementar	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
LP	Cronograma de las actividades Visión global del dominio del problema	
DI	Técnicas de diseño Conocimiento de los estándares referidos al documento arquitectura del sistema Conocimiento específico del dominio del problema	
UO	Conocimiento específico del dominio del problema	

Técnicas:

Técnicas de diseño

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación de Requerimientos de	3.03	Especificación validada por la gerencia superior y los	PDT

Software		usuarios operativos	
Documentación sistemas externos	AE	Información sistemas externos	PDT
Información Fundamentación	5.00	Registradas las acciones de fundamentación	PDT
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Output			
Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.00 4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02 4.05 AC	Documento de arquitectura del sistema sin validar	PDT
Especificación Conflicto	5.00 5.01	Identificador, descripción y contexto del conflicto	PDT
Especificación Fundamentación	5.00 5.03	Identificador del objeto a justificar	PDT
Definiciones:			
Diseño Arquitectónico: Abarca la identificación de los distintos subsistemas, con el propósito de establecer el marco de trabajo para el control y comunicación de los mismos [Sommerville, 2002]. **Componente o módulo:** Suministra y utiliza uno o más servicios implementados en los otros componentes. Un componente puede estar compuesto por componentes más simples [Sommerville, 2002]. **Descomposición Funcional:** Los desarrolladores identifican las principales funciones del sistema y las refinan sucesivamente en componentes con limitada funcionalidad [SEL-81-305, 1992] . **Asignación de Requerimientos de Software:** Representa las actividades que mapean los Requerimientos de Software en los documentos elaborados [Wright, 1991]. **Sistema externo:** Cualquier sistema (interno o externo) que tenga vinculación con el sistema a desarrollar.			
Restricciones:			
No se implementan las acciones que se relacionan con el vínculo *modify* descripto en el metamodelo. No se especifican las acciones relacionadas con la validación de la arquitectura del sistema confeccionada.			
Variables a medir:			
Nombre de la variable			
Variables definidas en 4.01,4.02, 4.03 y 4.04			
Observaciones:			
En esta actividad se confecciona el **Documento Arquitectura del Sistema.** La responsabilidad compartida en la actividad **Seleccionar los elementos a diseñar** se debe a que el Líder del Proyecto posee la capacidad de coordinar los esfuerzos y recursos durante la realización del proceso de desarrollo. Al realizar dicha actividad pueden originarse conflictos, o presentarse hechos o acciones que necesitan ser justificadas. Ambas circunstancias deben estar debidamente documentadas.			

6.2.4.1 Actividad 4.01 Seleccionar los elementos a diseñar

Al especificar esta actividad, se tuvieron en cuenta los diversos motivos presentes en la selección de los diversos elementos a diseñar. En determinadas situaciones se debe seleccionar un conjunto de requerimientos los cuales pueden o no poseer entidades de diseño ya elaboradas. Al especificar la actividad también se consideró el hecho de que sólo se necesiten seleccionar entidades de diseño elaboradas en iteraciones previas. Por lo tanto, tal selección involucra las siguientes acciones:

1. Seleccionar los Requerimientos de Software a diseñar
2. Seleccionar las entidades de diseño
3. Determinar la actividad de diseño a realizar

Aunque en dicha actividad no se confecciona ninguna entidad o vínculo de traceability, la información que se produce permitirá en actividades posteriores, especificar cuáles son los requerimientos que dirigen la confección de las entidades de diseño (sistema, subsistema o componente) y la respectiva información de asignación de requerimientos.

Código:	**Nombre:**	Desarrollar la Arquitectura del Sistema **Seleccionar los elementos a diseñar** Establecer la estructura del Sistema Establecer la estructura de los Subsistemas Establecer la estructura de los Componentes Documentar la arquitectura del Sistema
4.01	**Seleccionar los elementos a diseñar**	

Objetivos:

Confeccionar un listado con los distintos requerimientos a diseñar.

Justificación:

Esta actividad permite identificar los elementos necesarios para dar comienzo a las actividades de diseño.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar los Requerimientos de Software a diseñar	Confeccionar un listado de los requerimientos que direccionan las actividades de diseño	LP/ DI
2	Seleccionar las entidades de diseño	Confeccionar un listado con las entidades de diseño afectadas a las actividades de diseño	LP/ DI
3	Determinar la actividad de diseño a realizar	Seleccionar la próxima actividad de diseño	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento de las técnicas de diseño	
LP	Conocimiento del cronograma del proyecto de software	

Técnicas:

Seleccionar los Requerimientos de Software a diseñar. Según el criterio de selección elegido se debe confeccionar una lista con los requerimientos a diseñar.

Seleccionar las entidades de diseño. A partir del listado confeccionado se debe:

1. Verificar si los requerimientos que integran el listado poseen entidades de diseño elaboradas. Tal verificación se realiza chequeando si los mismos poseen el vínculo *allocation* en el componente **Información de asignación** de los requerimientos seleccionados. Si existen, se debe agregar al listado los identificadores de las entidades de diseño afectadas.
2. Además se puede seleccionar las entidades de diseño que se consideren necesarias.

Determinar la actividad de diseño a realizar. Para seleccionar la próxima actividad se verifica que

1. Si los requerimientos seleccionados involucran la definición, modificación o eliminación de detalles que corresponden a la especificación del sistema, se debe seleccionar la actividad **4.02 Establecer la estructura del sistema**.
2. Si los requerimientos seleccionados involucran la definición, modificación o eliminación de detalles que corresponden a la especificación de algún Subsistema, se debe seleccionar la actividad **4.03 Establecer la estructura de los Subsistemas.**
3. Si los requerimientos seleccionados involucran la definición, modificación o eliminación de detalles que corresponden a la especificación de algún componente, se debe seleccionar la actividad **4.04 Establecer la estructura de los Componentes.**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Listado elementos a diseñar	4.02 4.02.01 4.02.02 4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02	Completo los componentes del listado	INF

Definiciones:

Restricciones:

Los Requerimientos de Software seleccionados deben pertenecer a una SRS validada y aprobada

Variables a medir:

Nombre de la variable	
Cantidad de Requerimientos de Software seleccionados	
Cantidad de Entidades de diseño seleccionadas	
Cantidad de Requerimientos de Software con Entidades de diseño ya elaboradas	

Observaciones:

6.2.4.2 Actividad 4.02 Establecer la estructura del Sistema

Esta actividad debe ser llevada a cabo cuando los requerimientos seleccionados poseen la magnitud o alcance necesario para definir aspectos que conciernen a la descripción del sistema a desarrollar. Tales acciones de detallan en:

1. Definir el Sistema a desarrollar
2. Confirmar la implementación del Sistema

Al detallar la estructura del sistema se debe confeccionar la entidad **Diseño.** En dicha entidad, se especifican por medio de los vínculos *drive* los requerimientos que direccionan la confección del sistema. Si fuese necesario, se especificarán (vínculos *based_on*) los **Mandates** con los cuales el diseñador se base para confeccionar el sistema. Las distintas alternativas identificadas se detallan mediante las entidades **Sistema,** las cuales se relacionan con la entidad **Diseño** a través del vínculo *define*. Es importante destacar que sólo una alternativa debe ser confirmada (vínculo *create)* para realizar la confección de los distintos subsistemas.

Por cada alternativa identificada se describen las funciones que el sistema debe realizar, como así también, se especifican (vínculos *used_by*) los recursos necesarios para llevar a cabo la alternativa propuesta. Por último, se especifica la información relacionada con la asignación de requerimientos. Tal especificación se realiza a través de los vínculos *allocated_to* y *satisfy*.

Código:	**Nombre:**	Desarrollar la Arquitectura del Sistema Seleccionar elementos a diseñar **Establecer la estructura del Sistema** Establecer la estructura de los Subsistemas Establecer la estructura de los Componentes Documentar la arquitectura del Sistema
4.02	**Establecer la estructura del Sistema**	

Objetivos:

Describir la estructura del sistema a desarrollar.

Justificación:

La información producida en esta actividad permite visualizar tanto la complejidad, alcance, y dependencias que presenta el sistema a desarrollar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Definir el Sistema a desarrollar	Determinar y representar el sistema a implementar	DI
2	Confirmar la implementación del Sistema	Establecer la alternativa del sistema más apropiada a implementar	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento referido a las técnicas de diseño seleccionadas	

Técnicas:

Descomposición Funcional
Análisis de la documentación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Especificación de Requerimientos de Software	3.03	Especificación validada por la gerencia superior y los usuarios operativos	PDT
Documentación sistemas externos	AE	Información sistemas externos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Sistema	4.03	Definición del Sistema comprendida y confirmada	INF

	4.03.01 4.03.02 4.05		

Definiciones:

Restricciones:

Para modelar el Sistema sólo se utilizan los componentes de información especificados en el template de la entidad[49].

Variables a medir:

Nombre de la variable	
Vínculos *drive* identificados/modificados/eliminados	
Vínculos *based_on* identificados/modificados/eliminados	
Vínculos *define* identificados/modificados/eliminados	
Vínculos *performs* identificados/modificados/eliminados	
Vínculos *address* identificados/modificados/eliminados	
Vínculos *used_by* identificados/modificados/eliminados	
Vínculos *depends_on* identificados/modificados/eliminados	
Vínculos *allocated_to* identificados/modificados/eliminados	
Vínculos *satisfy* identificados/modificados/eliminados	
Vínculos *create* identificados	

Observaciones:

En esta actividad se confecciona la entidad **Diseño, Sistema y Función.** El Documento Arquitectura del Sistema no se ha detallado como input de la actividad debido a que no se especifican las acciones relacionadas con la modificación del diseño.

Código:	**Nombre:**	Establecer la estructura del Sistema **Definir el Sistema a desarrollar** Confirmar la implementación del Sistema
4.02.01	**Definir el Sistema a desarrollar**	

Objetivos:

Determinar y representar el sistema a implementar.

Justificación:

La información elaborada en esta actividad permitirá describir y delimitar el alcance del sistema a desarrollar

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar las descripciones de los elementos seleccionados	Comprender el significado y alcance de los distintos aspectos relacionados con el sistema a desarrollar.	DI
2	Identificar las Funciones a realizar	Detallar las funciones a realizar por el sistema	DI
3	Identificar los Recursos a utilizar	Detallar los recursos necesarios para la implementación del sistema	DI
4	Establecer dependencias con Sistemas Externos	Detallar las relaciones de dependencias con los sistemas existentes.	DI/ LP
5	Establecer la asignación de los requerimientos seleccionados	Detallar la asignación de los requerimientos seleccionados	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento en técnicas de descomposición funcional Capacidad de análisis Conocimiento de mandates de diseño	

Técnicas:

Analizar las descripciones de los elementos seleccionados A partir del análisis realizado se debe:

1. Confeccionar la entidad **Diseño** y completar los componentes **Identificador, Descripción**.
2. Por cada ítem que integra el **Listado elementos a diseñar** se confecciona el vínculo *drive* con la información pertinente. Tal vínculo se registra en el componente **Trata de** perteneciente a la entidad **Diseño** confeccionada.
3. Por cada **Mandates** de diseño identificado se confecciona el vinculo *based_on* con la información pertinente. Estos vínculos se registran en el componente **Basado en** de la respectiva entidad **Diseño**.
4. Por cada alternativa de diseño identificada para elaborar el sistema se debe:
   I. Confeccionar la entidad **Sistema** y completar los componentes **Identificador y Descripción.**
   II. Especificar el vínculo *define* con la información pertinente. Tal vínculo debe ser registrado tanto en el componente **Alternativas identificadas** de la respectiva entidad **Diseño**, como así también en el componente **Define a** de la respectiva entidad **Sistema**.
   III. Si fuese necesario se puede modelar la alternativa identificada. El identificador de dicho modelo debe ser registrado en el componente **Modelado por** de la respectiva entidad **Sistema**.

Definir las Funciones a realizar por el sistema. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Confeccionar la entidad **Función** y completar los componentes **Identificador y Descripción**.
2. Especificar a que alternativa de diseño pertenece la función identificada. Tal relación se materializa con el vínculo *performs*. Tal vínculo se registra tanto en el componente **Identificada por** de la respectiva entidad **Función**, como así también, en el componente **Funciones identificadas** de la respectiva entidad **Sistema**.

[49] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

3. Establecer los vínculos correspondientes entre la función establecida y los requerimientos. Tales relaciones se especifican mediante el vinculo *address,* el cual debe ser registrado en el componente **Trata de** perteneciente a la entidad **Función** confeccionada.
4. Si fuese necesario se pueden modelar las funciones identificadas.

Identificar los recursos a utilizar. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Determinar los recursos necesarios para la implementación del sistema
2. Confeccionar el vínculo de traceability *used_by*. Tal vínculo se registra en el componente **Recursos a utilizar** de la respectiva entidad **Sistema**.

Establecer las dependencias con los Sistemas Externos. Si existen dependencias con sistemas externos se debe especificar el o los vínculos *depends_on* con la información correspondiente. Tales vínculos deben registrarse en el componente **Vínculos con sistemas** de la respectiva entidad **Sistema**.

Establecer la asignación de los requerimientos seleccionados. Por cada uno de los requerimientos presentes en la confección del sistema se debe:

1. Confeccionar el vínculo *allocated_to* con la información correspondiente. Tal vínculo se registra en el componente **Información de asignación** del respectivo requerimiento.
2. Confeccionar el vínculo *satisfy* con la información correspondiente. Tal vínculo se debe registra en el componente **Satisface a** de la respectiva entidad **Sistema.**

Input:

Nombre	Origen	Criterio de Entrada:Estado o Condición	Tipo
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Documentación sistemas externos	AE	Información sistemas externos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Sistema	4.02.02	Completos los componentes de información que describen al sistema	INF

Definiciones:

Restricciones:

Las funciones identificadas deben estar vinculadas con algún Requerimiento de Software.

Variables a medir:

Nombre de la variable	
Vínculos *drive* identificados	
Vínculos *based_on* identificados	
Vínculos *define* identificados	
Vínculos *performs* identificados	
Vínculos *address* identificados	
Vínculos *used_by* identificados	
Vínculos *depends_on* identificados	
Vínculos *allocated_to* identificados	
Vínculos *satisfiy* identificados	

Observaciones:

En esta actividad se confeccionan los vínculos de traceability *drive, based_on, define performs address, used_by, depends_on, allocated_to, satisfy.*Cabe aclarar que la identificación y descripción de las funciones a realizar por el sistema es optativa.

Código:	**Nombre:**	Establecer la estructura del Sistema Definir el Sistema a desarrollar **Confirmar la implementación del Sistema**
4.02.02	**Confirmar la implementación del Sistema**	

Objetivos:

Establecer la alternativa del sistema más apropiada a implementar

Justificación:

Una vez identificada las alternativas de diseño que describen el sistema a implementar se deberá seleccionar una de estas para poder continuar con las restantes actividades de diseño.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar la alternativa del Sistema a implementar	Determinar cual de las distintas alternativas referidas al sistema se selecciona para su implementación	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Capacidad de análisis	

Técnicas:

Seleccionar la alternativa del Sistema a implementar. Una vez seleccionada la alternativa se debe confeccionar el vínculo *create*. Tal vínculo debe ser registrado tanto en el componente **Alternativa confirmada** de la entidad **Diseño** que describe al sistema, como en el componente **Define a** de la entidad **Sistema** que describe alternativa seleccionada.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Descripciones Sistema	4.02.02	Completos los componentes de información que describen al sistema	INF
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Output			
Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Sistema	4.03 4.03.01 4.03.02 4.05	Definición del Sistema comprendido y confirmado	INF
Definiciones:			
Restricciones:			
Variables a medir:			
Nombre de la variable			
Vínculos *create* identificados			
Observaciones:			
En esta actividad se confeccionan los vínculos de traceability *create*			

6.2.4.3 Actividad 4.03 Establecer la estructura de los Subsistemas

En las actividades anteriores se ha detallado cómo se debe describir el sistema a implementar. A partir de este momento se especifican las acciones que caracterizan a la descomposición del sistema en subsistemas y posteriormente componentes. Dicha descomposición permite gestionar la complejidad en entidades más manejables y entendibles. Dichas acciones se especifican en:

1. Definir los Subsistemas a desarrollar
2. Confirmar la implementación de los Subsistemas

Al detallar la estructura de los distintos subsistemas se deben confeccionar diferentes entidades **Diseño.** En estas entidades, se especifican por medio de los vínculos *drive* los requerimientos que direccionan la confección de los respectivos subsistemas. Además, si fuese necesario se especificarán (vínculos *based_on*) los **Mandates** con los cuales el diseñador debe basarse para confeccionar el respectivo subsistema. Las distintas alternativas identificadas se detallan mediante las entidades **Subsistema,** las cuales se relacionan con la entidad **Diseño** a través del vínculo *define*. Es importante destacar que sólo una alternativa debe ser confirmada (vínculo *create)* para realizar la confección de los distintos componentes. Al realizarse tal acción, se debe especificar que el sistema ha sido particionado por medio del vínculo *part_of*.

Por cada alternativa identificada se describen las funciones que el subsistema debe realizar, como así también se especifican (vínculos *used_by*) los recursos necesarios para llevar a cabo la alternativa propuesta. Por último, se especifica la información relacionada con la asignación de requerimientos a través de los vínculos *allocated_to* y *satisfy*.

Código:	**Nombre:**	Desarrollar la Arquitectura del Sistema Seleccionar los elementos a diseñar Establecer la estructura del Sistema **Establecer la estructura de los Subsistemas** Establecer la estructura de los Componentes Documentar la arquitectura del Sistema
4.03	**Establecer la estructura de los Subsistemas**	

Objetivos:

Describir la estructura de los subsistemas.

Justificación:

La información producida en esta actividad permite visualizar tanto la complejidad, alcance, y dependencias que presenta el subsistema a desarrollar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Definir los Subsistemas a desarrollar	Determinar y representar los subsistemas a implementar	DI
2	Confirmar la implementación de los Subsistemas	Determinar por cada subsistema identificado cual de las distintas alternativas se selecciona para su implementación	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento referido a las técnicas de diseño seleccionadas	

Técnicas:

Descomposición Funcional
Análisis de la documentación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Especificación de Requerimientos de Software	3.03	Especificación validada y aprobada por la gerencia superior y los usuarios operativos	PDT
Descripciones Sistema	4.02.01	Definición del Sistema comprendida y confirmada	INF
Documentación sistemas externos	AE	Información sistemas externos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Subsistemas	4.04 4.04.01 4.04.02 4.05	Completos los componentes de información que describen a los subsistemas	INF

Definiciones:

Restricciones:

Para modelar los Subsistemas sólo se utilizan los componentes de información especificados en el template de la entidad[50].

Variables a medir:

Nombre de la variable	
Vínculos *drive* identificados	
Vínculos *based_on* identificados	
Vínculos *define* identificados	
Vínculos *performs* identificados	
Vínculos *address* identificados	
Vínculos *used_by* identificados	
Vínculos *depend_on* identificados	
Vínculos *depends_on* identificados	
Vínculos *allocated _to* identificados	
Vínculos *satisfiy* identificados	
Vínculos *create* identificados	
Vínculos *part_of* identificados	

Observaciones:

En esta actividad se confecciona la entidad **Diseño, Subsistema y Función.**

[50] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Código:	**Nombre:**	Establecer la estructura de los Subsistemas **Definir los Subsistemas a desarrollar** Confirmar la implementación de los Subsistemas
4.03.01	**Definir los Subsistemas a desarrollar**	

Objetivos:

Determinar y representar los subsistemas a implementar.

Justificación:

La información producida en esta actividad se debe utilizar para poder describir y delimitar el alcance de los distintos subsistemas a implementar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar las descripciones de los elementos seleccionados	Comprender el significado y alcance de los distintos aspectos relacionados con los subsistemas a desarrollar.	DI
2	Identificar las Funciones a realizar	Detallar las funciones a realizar por el subsistema	DI
3	Identificar los Recursos a utilizar	Detallar los recursos necesarios para la implementación del subsistema	DI
4	Establecer dependencias del Subsistema identificado	Detallar cómo se relaciona el subsistema con los demás subsistema y sistemas externos.	DI/ LP
5	Establecer la asignación de los requerimientos seleccionados	Detallar la asignación de los requerimientos seleccionados	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento en técnicas de descomposición funcional Capacidad de análisis Conocimiento de mandates de diseño	

Técnicas:

Analizar las descripciones de los elementos seleccionados A partir del análisis realizado se debe:

1. Confeccionar la entidad **Diseño** y completar los componentes **Identificador, Descripción**.
2. Por cada ítem que integra el **Listado elementos a diseñar** se confecciona el vínculo *drive* con la información pertinente. Tal vínculo se registra en el componente **Trata de** perteneciente a la entidad **Diseño** confeccionada.
3. Por cada **Mandates** de diseño identificado se confecciona el vinculo *based_on* con la información pertinente. Estos vínculos se registran en el componente **Basado en** de la respectiva entidad **Diseño**.
4. Por cada alternativa de diseño identificada para elaborar el sistema se debe:
   I. Confeccionar la entidad **Subsistema** y completar los componentes **Identificador y Descripción.**
   II. Especificar el vínculo *define* con la información pertinente. Tal vínculo debe ser registrado tanto en el componente **Alternativas identificadas** de la respectiva entidad **Diseño**, como así también en el componente **Define a** de la respectiva entidad **Subsistema**.
   III. Si fuese necesario se puede modelar la alternativa identificada. El identificador de dicho modelo debe ser registrado en el componente **Modelado por** de la respectiva entidad **Subsistema**.

Definir las Funciones a realizar por el sistema. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Confeccionar la entidad **Función** y completar los componentes **Identificador y Descripción**.
2. Especificar a que alternativa de diseño pertenece la función identificada. Tal relación se materializa con el vínculo *performs*. Tal vínculo se registra tanto en el componente **Identificada por** de la respectiva entidad **Función**, como así también, en el componente **Funciones identificadas** de la respectiva entidad **Subsistema**.
3. Establecer los vínculos correspondientes entre la función establecida y los requerimientos. Tales relaciones se especifican mediante el vinculo *address,* el cual debe ser registrado en el componente **Trata de** perteneciente a la entidad **Función** confeccionada.
4. Si fuese necesario se pueden modelar las funciones identificadas.

Identificar los recursos a utilizar. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Determinar los recursos necesarios para la implementación del sistema
2. Confeccionar el vínculo de traceability *used_by*. Tal vínculo se registra en el componente **Recursos a utilizar** de la respectiva entidad **Subsistema**.

Establecer las dependencias con los Sistemas Externos. Si existen dependencias con sistemas externos se debe especificar el o los vínculos *depends_on* con la información correspondiente. Tales vínculos deben registrarse en el componente **Vínculos con sistemas** de la respectiva entidad **Subsistema**.

Establecer la asignación de los requerimientos seleccionados. Por cada uno de los requerimientos presentes en la confección del sistema se debe:

1. Confeccionar el vínculo *allocated_to* con la información correspondiente. Tal vínculo se registra en el componente **Información de asignación** del respectivo requerimiento.
2. Confeccionar el vínculo *satisfy* con la información correspondiente. Tal vínculo se debe registra en el componente **Satisface a** de la respectiva entidad **Subsistema.**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Descripciones Sistema	4.02.02	Definición del Sistema comprendida y confirmada	INF
Documentación sistemas externos	AE	Información sistemas externos	PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo

Descripciones Subsistemas	4.03.02	Completos los componentes de información que describen a los subsistema	INF

Definiciones:

Restricciones:

Las funciones identificadas deben estar vinculadas con algún Requerimiento de Software.

Variables a medir:

Nombre de la variable	
Vínculos *drive* identificados	
Vínculos *based_on* identificados	
Vínculos *define* identificados	
Vínculos *performs* identificados	
Vínculos *address* identificados	
Vínculos *used_by* identificados	
Vínculos *depend_on* identificados	
Vínculos *depends_on* identificados	
Vínculos *allocated_to* identificados	
Vínculos *satisfy* identificados	

Observaciones:

En esta actividad se confeccionan los vínculos de traceability *drive, based_on, define performs address, used_by, depends_on,depend_on, allocated_to, satisfy*. Las actividades especificadas deben ser realizadas por cada uno de los subsistemas identificados. Cabe aclarar que la identificación y descripción de las funciones a realizar por el subsistema es optativa.

Código:	**Nombre:**	Establecer la estructura de los Subsistemas Definir los Subsistemas a desarrollar **Confirmar la implementación de los Subsistemas**
4.03.02	**Confirmar la implementación de los Subsistemas**	

Objetivos:

Determinar por cada subsistema identificado cual de las distintas alternativas se selecciona para su implementación.

Justificación:

Una vez identificada las alternativas de diseño que describen a un determinado subsistema a implementar se deberá seleccionar una de estas para poder continuar con las restantes actividades de diseño.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar la alternativa del Subsistema a implementar	Establecer la alternativa a implementar para un determinado subsistema	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Capacidad de análisis	

Técnicas:

Seleccionar la alternativa del Subsistema a implementar

1. Al seleccionar la alternativa a implementar se debe confeccionar el vínculo *create*. Este vínculo debe ser registrado en el componente **Alternativa confirmada** de la entidad **Diseño** que describe a la entidad **Subsistema**, como así también debe registrase en el componente de información **Define a** de la entidad seleccionada.
2. Una vez seleccionada la alternativa a implementar se debe confeccionar el vínculo *part_of.* Este se debe registrar en el componente **Compuesto por** de la entidad **Sistema** que se referencia en dicho vínculo

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Descripciones Subsistemas	4.03.01	Completos los componentes de información que describen a los subsistema	INF
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Subsistemas	4.04 4.04.01 4.04.02 4.05	Definiciones de los Subsistemas comprendidos y confirmados	INF

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	
Vínculos *create* identificados	
Vínculos *part_of* identificados	

Observaciones:
En esta actividad se confeccionan los vínculos de traceability *create y part_of* .Las actividades especificadas deben ser realizadas por cada uno de los subsistemas identificados.

6.2.4.4 Actividad 4.04 Establecer la estructura de los Componentes

Algunos subsistemas se pueden particionar en distintos componentes debido a la complejidad que poseen. Dichas acciones se especifican en:

1. Definir los Componentes a desarrollar
2. Confirmar la implementación de los Componentes

Al detallar la estructura de los distintos componentes se deben confeccionar las entidades **Diseño.** En dichas entidades, se especifican por medio de los vínculos *drive* los requerimientos que direccionan la confección del respectivo componente. Además, si fuese necesario se especificarán (vínculos *based_on*) los **Mandates** con los cuales el diseñador debe basarse para poder confeccionar al respectivo componente. Las distintas alternativas identificadas se detallan mediante las entidades **Componentes**. Estas entidades se relacionan con la entidad **Diseño** a través del vínculo *define*

Es importante destacar que sólo una alternativa debe ser confirmada (vínculo *create)* para realizar la confección de los distintos componentes Al realizarse tal acción, se debe especificar que el subsistema ha sido particionado por medio del vínculo *part_of*.

Por cada alternativa identificada se describen las funciones que el componente debe realizar. Se detallan también (vínculos *used_by*) los recursos necesarios para llevar a cabo la alternativa propuesta. Por último, se especifica la información relacionada con la asignación de requerimientos a través de la confección de los vínculos *allocated_to* y *satisfy*.

Código:	**Nombre:**	Desarrollar la Arquitectura del Sistema Seleccionar los elementos a diseñar Establecer la estructura del Sistema Establecer la estructura de los Subsistemas **Establecer la estructura de los Componentes** Documentar la arquitectura del Sistema
4.04	**Establecer la estructura de los Componentes**	

Objetivos:
Crear un documento que identifique y describa la arquitectura del sistema a implementar
Justificación:
La información producida en esta actividad permite visualizar tanto la complejidad, alcance, y dependencias que presenta el componente a desarrollar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
2	Definir los Componentes a desarrollar	Determinar y representar los componentes a implementar	DI
3	Confirmar la implementación de los Componentes	Determinar por cada componente identificado cual de las distintas alternativas se selecciona para su implementación	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento referido a las técnicas de diseño seleccionadas	

Técnicas:
Descomposición Funcional Análisis de la documentación

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT

Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Especificación de Requerimientos de Software	3.03	Especificación validada y aprobada por la gerencia superior y los usuarios operativos	PDT
Descripciones Subsistemas	4.03.02	Completos los componentes de información que describen a los subsistemas	INF
Documentación sistemas externos	AE		PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Componentes	4.05	Completos los componentes de información que describen los componentes	INF

Definiciones:

Restricciones:

Para modelar los Componentes sólo se utilizan los componentes de información especificados en el template de la entidad[51].

Variables a medir:

Nombre de la variable	
Cantidad de Mandates identificados por categoría	
Vínculos *drive* identificados	
Vínculos *based_on* identificados	
Vínculos *define* identificados	
Cantidad de entidades de Componentes identificadas	
Cantidad de Funciones identificadas	
Vínculos *performs* identificados	
Vínculos *address* identificados	
Vínculos *used_by* identificados	
Vínculos *depend_on* identificados	
Vínculos *depends_on* identificados	
Vínculos *allocated _to* identificados	
Vínculos *satisfy* identificados	
Vínculos *create* identificados	
Vínculos *part_of* identificados	

Observaciones:

En esta actividad se confecciona la entidad **Diseño, Componente y Función**.

Código:	**Nombre:**	Establecer la estructura de los Componentes **Definir los Componentes a desarrollar** Confirmar la implementación de los Componentes
4.04.01	**Definir los Componentes a desarrollar**	

Objetivos:

Determinar y representar los componentes a implementar.

Justificación:

La información producida en esta actividad se debe utilizar para poder describir y delimitar el alcance de los componentes a implementar.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Analizar las descripciones de los elementos seleccionados	Comprender el significado y alcance de los distintos aspectos relacionados con los componentes a desarrollar	DI
2	Identificar las Funciones a realizar	Detallar las funciones a realizar por el componente	DI
3	Identificar los Recursos a utilizar	Detallar los recursos necesarios para la implementación del subsistema	DI
4	Establecer dependencias del Componente identificado	Detallar cómo se relaciona el componente con los demás componentes, subsistemas y sistemas externos.	DI/ LP
5	Establecer la asignación de los requerimientos seleccionados	Detallar la asignación de los requerimientos seleccionados	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Conocimiento en técnicas de descomposición funcional Conocimiento de mandates de diseño	

Técnicas:

Analizar los descripciones de los elementos seleccionados A partir del análisis realizado se debe:

1. Confeccionar la entidad **Diseño** y completar los componentes **Identificador, Descripción**.
2. Por cada ítem que integra el **Listado elementos a diseñar** se confecciona el vínculo *drive* con la información pertinente.

[51] Ver sección III.2 Definición de Entidades del Anexo III Elementos de soporte

Tal vínculo se registra en el componente **Trata de** perteneciente a la entidad **Diseño** confeccionada.

3. Por cada **Mandates** de diseño identificado se confecciona el vinculo *based_on* con la información pertinente. Estos vínculos se registran en el componente **Basado en** de la respectiva entidad **Diseño**.
4. Por cada alternativa de diseño identificada para elaborar el sistema se debe:
   I. Confeccionar la entidad **Componente** y completar los componentes **Identificador y Descripción.**
   II. Especificar el vínculo *define* con la información pertinente. Tal vínculo debe ser registrado tanto en el componente **Alternativas identificadas** de la respectiva entidad **Diseño**, como así también en el componente de información **Define a** de la respectiva entidad **Componente**.
   III. Si fuese necesario se puede modelar la alternativa identificada. El identificador de dicho modelo debe ser registrado en el componente **Modelado por** de la respectiva entidad **Subsistema**.

Definir las Funciones a realizar por el sistema. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Confeccionar la entidad **Función** y completar los componentes **Identificador y Descripción**.
2. Especificar a que alternativa de diseño pertenece la función identificada. Tal relación se materializa con el vínculo *performs*. Tal vínculo se registra tanto en el componente **Identificada por** de la respectiva entidad **Función**, como así también, en el componente **Funciones identificadas** de la respectiva entidad **Componente**.
3. Establecer los vínculos correspondientes entre la función establecida y los requerimientos. Tales relaciones se especifican mediante el vinculo *address,* el cual debe ser registrado en el componente **Trata de** perteneciente a la entidad **Función** confeccionada.
4. Si fuese necesario se pueden modelar las funciones identificadas.

Identificar los recursos a utilizar. Esta actividad debe ser llevada a cabo por cada una de las alternativas de diseño que se han identificado.

1. Determinar los recursos necesarios para la implementación del sistema
2. Confeccionar el vínculo de traceability *used_by*. Tal vínculo se registra en el componente **Recursos a utilizar** de la respectiva entidad **Componente**.

Establecer las dependencias con los Sistemas Externos. Si existen dependencias con sistemas externos se debe especificar el o los vínculos *depends_on* con la información correspondiente. Tales vínculos deben registrarse en el componente **Vínculos con sistemas** de la respectiva entidad **Componente**.

Establecer la asignación de los requerimientos seleccionados. Por cada uno de los requerimientos presentes en la confección del sistema se debe:

1. Confeccionar el vínculo *allocated_to* con la información correspondiente. Tal vínculo se registra en el componente **Información de asignación** del respectivo requerimiento.
2. Confeccionar el vínculo *satisfy* con la información correspondiente. Tal vínculo se debe registra en el componente **Satisface a** de la respectiva entidad **Componente.**

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF
Descripciones Subsistemas	4.03.02	Completos los componentes de información que describen los subsistemas	INF
Documentación sistemas externos	AE		PDT

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Componentes	4.04.02	Completos los componentes de información que describen a los componentes	INF

Definiciones:

Restricciones:

Las funciones identificadas deben estar vinculadas con algún Requerimiento de Software.

Variables a medir:

Nombre de la variable	
Vínculos *drive* identificados	
Vínculos *based_on* identificados	
Vínculos *define* identificados	
Cantidad de entidades de Componentes identificadas	
Cantidad de Funciones identificadas	
Vínculos *performs* identificados	
Vínculos *address* identificados	
Vínculos *used_by* identificados	
Vínculos *depends_on* identificados	
Vínculos *depend_on* identificados	
Vínculos *allocated_to* identificados	
Vínculos *satisfy* identificados	

Observaciones:

En esta actividad se confeccionan los vínculos *drive, based_on, define performs, address, used_by, depends_on, depende_on allocated_to, satisfy*, Las actividades especificadas deben ser realizadas por cada uno de los componentes identificados. Cabe aclarar que la identificación y descripción de las funciones a realizar por el componente optativa.

Código:	**Nombre:**	Establecer la estructura de los Componentes Definir los Componentes a desarrollar **Confirmar la implementación de los Componentes**
4.04.02	**Confirmar la implementación del Componente**	

Objetivos:

Determinar por cada componente identificado cual de las distintas alternativas se selecciona para su implementación.

Justificación:

Una vez identificada las alternativas de diseño que describen a un determinado subsistema a implementar se deberá seleccionar una de estas para poder continuar con las restantes actividades de diseño.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Seleccionar la alternativa del Componente a implementar	Establecer la alternativa a implementar para un determinado componente	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Planilla de Cálculos
DI	Capacidad de análisis	

Técnicas:

Seleccionar la alternativa del Componente a implementar

1. Al seleccionar la alternativa a implementar se debe confeccionar el vínculo *create*. Este vínculo debe ser registrado en el componente **Alternativa confirmada** de la entidad **Diseño** que describe a la entidad **Componente**, como así también debe registrase en el componente de información **Define a** de la entidad seleccionada.
2. Una vez seleccionada la alternativa a implementar se debe confeccionar el vínculo *part_of.* Este se debe registrar en el componente **Compuesto por** de la entidad **Subsistema** que se referencia en dicho vínculo.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Descripciones Componentes	4.04.02	Completos los componentes de información que describen a los componentes	INF
Listado elementos a diseñar	4.01	Completo los componentes del listado	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Descripciones Componentes	4.05	Definiciones de los Componentes comprendidos y confirmados	INF

Definiciones:

En esta actividad se confeccionan los vínculos de traceability *create y part_of*

Restricciones:

Variables a medir:

Nombre de la variable	
Vínculos *create* identificados	
Vínculos *create* identificados	

Observaciones:

Las actividades especificadas deben ser realizadas por cada uno de los componentes identificados.

6.2.4.5 Actividad 4.05 Documentar la arquitectura del Sistema

Una vez finalizadas las actividades relacionadas con la identificación y descripción del sistema, subsistemas y componentes a desarrollar, se debe confeccionar el documento denominado **Arquitectura del Sistema**.

Código:	**Nombre:**	Desarrollar la Arquitectura del Sistema Seleccionar los elementos a diseñar Establecer la estructura del Sistema Establecer la estructura de los Subsistemas Establecer la estructura de los Componentes **Documentar la arquitectura del Sistema**
4.05	**Documentar la arquitectura del Sistema**	

Objetivos:

Crear un documento que identifique y describa la arquitectura del sistema a implementar.

Justificación:

La información producida en las distintas actividades que especifican el proceso de diseño deben estar debidamente documenta para su posterior uso en la programación del sistema.

Actividades:

Paso	Nombre Actividad	Objetivo	Rol
1	Generar el Documento de Arquitectura del Sistema	Crear un documento que contenga lo elaborado en el proceso de diseño	DI

Recursos:

Humanos:		**Tecnológicos:**
Rol:	**Capacitación requerida:**	Procesador de Textos
DI	Conocimiento de los estándares referidos al documento arquitectura del sistema	

Técnicas:

Generar el Documento de Arquitectura del Sistema

- En la primera iteración del proceso de desarrollo se elabora el documento a partir de las descripciones elaboradas.
- En iteraciones posteriores se elabora el documento tomando como base el documento confeccionado en la iteración anterior. Además se sebe integran al mismo las descripciones elaboradas en la presente iteración.

Input:

Nombre	Origen	Criterio de Entrada :Estado o Condición	Tipo
Documento de Arquitectura del Sistema	4.05	Documento de arquitectura del sistema sin validar	PDT
Descripciones del Sistema	4.02.02	Definición del Sistema comprendido y confirmado	INF
Descripciones de los Subsistemas	4.03.02	Definiciones de los Subsistemas comprendidos y confirmados	INF
Descripciones de los Componentes	4.04.02	Definiciones de los Componentes comprendidos y confirmados	INF

Output

Nombre	Destino	Criterio de Salida :Estado o Condición	Tipo
Documento de Arquitectura del Sistema	4.00 4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02 4.05 AC	Documento de arquitectura sin validar	PDT

Definiciones:

Restricciones:

Variables a medir:

Nombre de la variable	

Observaciones:

La siguiente tabla muestra los distintos productos de trabajos generados en cada una de estas actividades.

	Actividad	Input	Origen	Output	Destino
4.00	Desarrollar la Arquitectura del Sistema	Documento Arquitectura del Sistema	4.05	Documento Arquitectura del sistema	4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02 4.05 AC
		Información Fundamentación	5.00	Especificación Fundamentación	5.00 5.03
		Documentación sistemas externos	AE	Especificación Conflicto	5.00
		SRS	3.03		5.01
4.01	Seleccionar los elementos diseñar	SRS	3.03	Listado elementos a diseñar	4.02 4.02.01 4.02.02 4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02
4.02	Establecer la estructura del sistema	SRS	3.03	Descripciones Sistema	4.03
		Listado elementos a diseñar	4.01		4.03.01
		Documentación sistemas externos	AE		4.03.02 4.05
4.02.01	Definir el Sistema a desarrollar	Listado elementos a diseñar	4.01	Descripciones Sistema	4.02.02
		Documentación sistemas externos	AE		

4.02.02	Confirmar la implementación del Sistema	Listado elementos a diseñar Descripciones Sistema	4.01 4.02.02	Descripciones Sistema	4.03 4.03.01 4.03.02 4.05
4.03	Establecer la estructura de los Subsistemas	Listado elementos a diseñar SRS Descripciones Sistema Documento Arquitectura del Sistema Documentación sistemas externos	4.01 3.03 4.02.02 4.05 AE	Descripciones Subsistemas	4.04 4.04.01 4.04.02 4.05
4.03.01	Definir la estructura de los Subsistemas	Documento Arquitectura del Sistema Listado elementos a diseñar Documentación sistemas externos Descripciones Sistema	4.05 4.01 AE 4.02.02	Descripciones Subsistemas	4.03.02
4.03.02	Confirmar la implementación de los Subsistemas	Listado elementos a diseñar Documento Arquitectura del Sistema Descripciones Subsistemas	4.01 4.05 4.03.02	Descripciones Subsistemas	4.04 4.04.01 4.04.02 4.05
4.04	Establecer la estructura de los Componentes	Listado elementos a diseñar Documento Arquitectura del Sistema SRS Descripciones Sistema Documentación sistemas externos	4.01 4.05 3.03 4.02.02 AE	Descripciones Componentes	4.05
4.04.01	Definir los Componentes a desarrollar	Listado elementos a diseñar Documento Arquitectura del Sistema Documentación sistemas externos Descripciones Subsistemas	4.01 4.05 AE 4.03.02	Descripciones Componentes	4.04.02
4.04.02	Confirmar la implementación de los Componentes	SRS Documento Arquitectura del Sistema Síntesis información	3.03 4.05 4.04.02	Descripciones Componentes	4.05
4.05	Documentar la arquitectura del Sistema	Descripciones Sistema Documento Arquitectura del Sistema Descripciones Subsistemas Descripciones Componentes	4.02.01 4.05 4.03.02 4.04.02	Documento Arquitectura del Sistema	4.03 4.03.01 4.03.02 4.04 4.04.01 4.04.02 4.05 AC

Tabla 20 Entradas/salidas actividad Desarrollar la Arquitectura del Sistema

En este capítulo se ha detallado la especificación del proceso de desarrollo de software a partir de lo señalado por el esquema de traceability definido para los usuarios High End [Ramesh *et al*, 2001]. En el próximo capitulo se detalla el caso de prueba realizado para constatar lo especificado.

CAPÍTULO 7 CASO DE ESTUDIO

En este capítulo se detalla el caso de estudio referido a la validación del proceso de desarrollo de software especificado en el capítulo seis. Tal validación tiene como propósito poner en práctica lo descripto en el proceso de desarrollo para encontrar las debilidades y fortalezas del mismo. En primer término se describe la empresa seleccionada para realizar el caso de estudio. Posteriormente, se detalla lo realizado durante las distintas actividades que integran el proceso de desarrollo. Por último, se analiza la información relacionada con la implementación de los mecanismos de traceability.

7.1 Caso de Estudio Evergreen School of English

Se ha seleccionado un instituto de enseñanza del idioma inglés **Evergreen School of English** para poner en práctica lo especificado en el proceso de desarrollo. Este instituto cuenta con una matrícula de aproximadamente 250 alumnos.

Las actividades diarias como ser el registro de las asistencias de los docentes y alumnos, el registro de las calificaciones, entre otras actividades, se efectúan de manera manual. Este hecho representa una de las razones por la cual la Coordinadora Pedagógica de Evergreen solicita la elaboración de un software que automatice tales actividades. Además se contemplarán los mecanismos necesarios para realizar el seguimiento académico de los alumnos, como así también realizar el seguimiento de las distintas actividades realizadas por los docentes. En síntesis, el **sistema de gestión académica** debe alcanzar los siguientes objetivos:

- Monitorear la evolución académica de los alumnos.
- Planificar y seguir los pasos académicos de los alumnos.
- Automatizar las actividades diarias de la institución.

Los recursos humanos que participan durante la puesta en marcha del caso se dividen en dos categorías, el **equipo de desarrollo** y los **usuarios participantes**. El equipo de desarrollo está constituido por un ingeniero de sistema y un alumno de tercer año de la carrera Ingeniería de Sistemas de la Universidad Tecnológica Nacional Regional Villa María. Por otra parte los usuarios que participaron en la elaboración del sistema fueron la Coordinadora Pedagógica y la Secretaria Administrativa del instituto.

La documentación diaria de la institución y los usuarios que participaron en el proceso de desarrollo constituyeron las distintas fuentes de información utilizadas para la elaboración

del sistema. Para elicitar tal información se realizaron tanto entrevistas individuales y grupales.

7.2 Puesta en marcha del proceso de desarrollo

En esta sección se describe lo realizado durante la realización del proceso de desarrollo. Así también se detallarán distintas observaciones relacionadas con los hechos acontecidos. Los respectivos documentos y listados confeccionados al realizar las actividades del proceso de desarrollo se definen en el **Anexo IV Documentación Caso de Estudio.**

7.2.1 Modelizar la Organización

Cabe recordar que el proceso **1.0 Modelizar la Organización** está conformado por las siguientes actividades **1.01 Analizar la Organización, 1.02 Desarrollar las Necesidades Organizacionales y 1.03 Desarrollar los Objetivos del Sistema.**

En primer término se puso en marcha la actividad **1.01 Analizar la organización.** En la misma se recolectaron diversos documentos que caracterizan la actividad diaria de la empresa. Posteriormente se realizó una entrevista con la Coordinadora Pedagógica con el fin de poder recabar la información necesaria para cumplimentar lo especificado en la actividad.

Tras analizar los datos recolectados se confeccionó el documento **Modelos de la Organización**. En este documento se especifican los objetivos, estrategias y políticas del instituto. Se advierte que tales conceptos no se encuentran debidamente especificados en un documento. Sin embargo, a partir de lo expuesto por la Coordinadora Pedagógica en la entrevista realizada y del análisis de la documentación (proyecto educativo de la institución – contrato pedagógico) se pudieron establecer tales requisitos. Es importante destacar que durante la puesta en marcha de esta actividad no se detectaron conflictos relacionados con la problemática del dominio. En cambio, al momento de realizar las respectivas mediciones surgió el siguiente inconveniente. Al momento de medir la variable **Cantidad de documentos consultados** no se contó con la debida definición de lo que comprende la misma. Por ejemplo, en el **legajo del alumno** se archivan las distintas libretas de calificaciones, notas de seguimiento como así también los datos identificatorios de los mismos. El inconveniente en este ejemplo radica en cómo se computan los documentos consultados (un documento o cuatro). Se tomó la decisión que en la variable **cantidad de documentos consultados** se compute la totalidad de documentos que se han consultado aunque estos formen parte de otro.

Durante la entrevista descripta en la actividad anterior también se identificaron las **Necesidades Organizacionales** y los **CSFs** que afectan el logro de las mismas. Tal

información cumplimenta con lo especificado en las actividades **1.02.01 Confeccionar las Necesidades Organizacionales y 1.02.02 Identificar los CSFs de la Organización**. Finalizadas estas actividades se planteó la alternativa en la cual los CSFs no sólo fuesen **recursos críticos**. Tras un breve intercambio de opiniones, se identificó la actividad **Preparar a los alumnos para rendir exámenes internacionales** como el CSF más significativo de la institución. Dicho CSF no se tendrá en cuenta debido a que en el metamodelo detallado en [Ramesh *et al*, 2001] sólo se define a los CSFs como recursos críticos de la organización.

Al realizar la actividad **1.02.03 Describir las Necesidades Organizacionales** no se vio la necesidad de confeccionar escenarios que modelen a dichas necesidades. En cambio, se enfatizó el poder vincular las necesidades identificadas con los objetivos, estrategias y políticas definidas en el instituto. En futuros trabajos se deberán especificar las pertinentes variables a medir para poder caracterizar tales vínculos. Se desea señalar que al finalizar la descripción de las necesidades organizacionales se tomó la libertad de no seguir secuencialmente lo definido en el proceso y se realizó la actividad **1.03.01 Confeccionar los Objetivos del Sistema**.

En la actividad **1.02.04 Validar las Necesidades Organizacionales** se detectó un error -la incorrecta identificación de una necesidad organizacional-. Para ello, se tuvo que volver a la actividad **1.02.01 Confeccionar las Necesidades Organizacionales** para eliminar ***NEC3 Mirada hacia la comunidad.***

Al concluir con la actividad **1.03.03 Validación de los Objetivos del Sistema** no fue necesaria la confección del **Listado de DEO y/o Propuestas de cambio,** ya que sólo se tuvo que modificar el contenido de ciertos componentes de información (errores ortográficos y de redacción).

Nombre de la variable	
Cantidad de Objetivos Cantidad de Estrategias Cantidad de Políticas	4 4 5
Cantidad de procesos analizados y descriptos	2
Cantidad de recursos afectados por procesos analizados Formar grupos para el dictado de clases Seleccionar examen internacional a rendir por alumno	 4 3
Total de personal detectado por área Coordinadora Pedagógica Secretaria Administrativa Docentes Maestranza	 1 1 6 1
Entrevistas realizadas	1
Entrevistas realizadas por personal Coordinadora Pedagógica	 1
Cantidad de documentos consultados.	13

Tabla 21 Mediciones actividad 1.01 Analizar la organización

1.02.01 Confeccionar las Necesidades Organizacionales	
Entrevistas realizadas	1
Documentos consultados	4
Necesidades Organizacionales	
Identificadas	3
Eliminadas	1
Definitivas	2
Necesidades Organizacionales por criterio	
Estratégica	1
Operacional	1
1.02.02 Identificar los CSFs de la Organización	
CSFs	
Identificados	2
Modificados	0
Eliminados	0
Definitivos	2
CSFs agrupados por necesidad organizacional	
NEC 1 Mirada holística sobre el cliente	2
NEC 2 Mirada sobre la organización interna	2
NEC 3 Mirada hacia la comunidad	1
Vínculos identify	
Identificados	5
Modificados	0
Eliminados	1
Definitivos	4
1.02.03 Describir las Necesidades Organizacionales	
Escenarios identificados/modificados/eliminados/definitivos	0
1.02.04 Validar las Necesidades Organizacionales	
Ítem DEO identificados	1

Tabla 22 Mediciones 1.02 Desarrollar las Necesidades Organizacionales

1.03.01 Confeccionar los Objetivos del Sistema	
Objetivos del Sistema	
Identificados	6
Eliminados	0
Definitivos	6
Vínculos *justify* identificados por necesidad	
NEC 1 Mirada holística sobre el cliente	3
NEC 2 Mirada sobre la organización interna	3
1.03.02 Describir los Objetivos del Sistema	
Escenarios	
Identificados	2
Modificados	0
Eliminados	0
Definitivos	2
1.03.03 Validación de los Objetivos del Sistema	
Ítem DEO identificados	0
Ítem DEO por objetivo del sistema	0
Ítem DEO por requerimiento de software	0

Tabla 23 Mediciones actividad 1.03 Desarrollar los Objetivos del Sistema

Finalizadas dichas actividades se realizó una inspección de lo elaborado conjuntamente con un agente externo al desarrollo, en este caso el tutor de la tesis. En tal revisión se decidió reincorporar la necesidad organizacional **Mirada hacia la comunidad** que se había eliminado en una primera instancia y además, se determinó que se debía agregar una nueva necesidad, la cual se denominó **Mirada de desempeño**. Dichas incorporaciones produjeron cambios en lo elaborado, como ser la confección de dos nuevas entidades (necesidades organizacionales) con sus respectivas descripciones, adicionando tres vínculos *identify*. Por último, al realizar nuevamente la actividad **1.03 Desarrollar los Objetivos del Sistema** se eliminó y agregaron dos vínculos *justify*. Las conclusiones obtenidas al concluir con la actividad **1.0 Modelizar la Organización** fueron las siguientes:

- El conocer el funcionamiento de la empresa como también el dominio del problema facilitó la realización de lo especificado para dicha actividad. La evidencia de este hecho radica en que sólo se tuvieron que realizar dos entrevistas para poder recolectar toda la información necesaria.
- Se deberá incorporar a la especificación del proceso de desarrollo una variable a medir que indique el total de entrevistas realizadas. También se deberán definir los instrumentos necesarios para poder valorizar el tiempo invertido en cada una de estas actividades.
- Se advierte la necesidad de establecer tanto las relaciones de dependencias entre los distintos **Objetivos del Sistema,** como así también las relaciones con los **CSFs** de la organización. Estas relaciones no están definidas en el metamodelo descripto en [Ramesh *et al*, 2001].
- No se ha especificado ningún mecanismo que refleje lo sucedido con la necesidad **Mirada hacia la comunidad**. En una primera instancia se decidió eliminar esta necesidad y luego recuperarla.

Actividades Entrevista 1	**Duración aproximada**
1.01 Analizar la Organización	15 minutos
1.02.01 Confeccionar las Necesidades Organizacionales	15 minutos
1.02.02 Identificar los CSFs de la Organización	20 minutos
1.02.03 Describir las Necesidades Organizacionales	10 minutos
1.03.01 Confeccionar los Objetivos del Sistema	15 minutos
1.03.02 Describir los Objetivos del Sistema	10 minutos
Actividades Entrevista 2	**Duración aproximada**
1.02.03 Validar las Necesidades Organizacionales	20 minutos
1.03.03 Validar los Objetivos del Sistema	10 minutos

Tabla 24 Detalle entrevistas realizadas

7.2.2 Especificar los Requerimientos de Software

Al poner en práctica lo especificado en la actividad **2.00 Especificar los Requerimientos de Software,** se llevó a cabo en primer lugar la actividad **2.01 Determinar los Requerimientos de Software a desarrollar**. Finalizada la misma se realizó la actividad **2.02 Reformar los Requerimientos de Software** y posteriormente la actividad **2.03 Documentar los Requerimientos de Software**.

La actividad **2.01 Determinar los Requerimientos de Software a desarrollar** comenzó con una entrevista con la Coordinadora Pedagógica, la cual tuvo como propósito identificar las actividades que deben realizarse para cumplimentar los Objetivos del Sistema que se han establecido. Tras la finalización de la misma se entrevistó a la Secretaria Administrativa. En este encuentro, sólo se analizaron las actividades en las cuales la secretaria está afectada. Para realizar esta actividad se consultaron los productos de trabajo **Especificación de los Objetivos del Sistema y Modelos de la Organización**. Las acciones descriptas se especifican en la actividad **2.01.01 Recolectar información del**

dominio del problema. Tras haber realizado las mediciones pertinentes se observó que la variable **Mandates identificados** indica mandates que pueden estar repetidos. A la política ***"Los distintos grupos se forman de acuerdo al nivel académico que presentan los alumnos al inicio del ciclo lectivo"*** se la identificó en cuatro ocasiones. Es por ello que se consideró necesario incorporar a la especificación la variable a medir **Mandates efectivos,** la cual detalla de manera no repetitiva los distintos mandates que afectan al desarrollo de las actividades identificadas. En el caso de estudio dicha variable tiene una valoración de tres políticas.

Con la información obtenida se llevó a cabo la actividad **2.01.02 Identificar los Requerimientos de Software** y posteriormente la actividad **2.01.03 Establecer los vínculos con los CSFs.** Para llevar a cabo esta última actividad se realizó una entrevista conjuntamente con la Coordinadora Pedagógica y la Secretaria Administrativa, en la cual se analizaron la incidencia de los recursos necesarios para la elaboración de los requerimientos que se habían identificado. Del análisis de las mediciones realizadas se consideró necesario incorporar a la especificación una variable a medir que refleje la cantidad de requerimientos afectados por los CSFs, **Cantidad de Requerimientos de Software afectados,** la cual posee una valoración de seis requerimientos.

En esta entrevista también se recolectó información para llevar a cabo la actividad **2.01.04 Describir los Requerimientos de Software**. En la misma se detallaron los componentes de información **urgencia, importancia y estabilidad.** Cabe aclarar que sólo se modelaron y se especificaron las dependencias correspondientes a los requerimientos que están gestionados por algún CSF. Estas dependencias detallan cuáles son los requerimientos que se necesitan para poder implementar a un determinado requerimiento. En el desarrollo de esta actividad se ha observado que el componente **urgencia** ha generado malos entendidos. Es por ello que se ha optado por cambiar el nombre de este componente por el de **prioridad.**

2.01.01 Recolectar información del dominio del problema	
Documentos consultados.	2
Personal entrevistado.	2
Entrevistas realizadas por personal Coordinadora Pedagógica Secretaria Administrativa	 1 1
Actividades identificadas	28
Actividades identificadas por Objetivo del Sistema Requerimiento de Software	 28 0
Restricciones identificadas	0
Mandates identificados por categoría Políticas	 9
2.01.02 Identificar los Requerimientos de Software	
Entrevistas realizadas	1
Entrevistas realizadas por personal Coordinadora Pedagógica Secretaria Administrativa	 1 1
Requerimientos identificados	16
Vínculos *generate* identificados	16

Vínculos *derive* identificados	-
Vínculos *is_a* (Restricciones identificadas)	-
Vínculos *based_on* identificados por categorías Políticas	 7
2.01.03 Establecer los vínculos con los CSFs	
Vínculo *managed_by* identificados	10
Entrevistas realizadas	1
Entrevistas realizadas por personal Coordinadora Pedagógica Secretaria Administrativa	 1 1
2.01.04 Describir los Requerimientos de Software	
Vínculos *depend_on* identificados	5
Vínculos *describe* identificados/eliminados/modificados/aceptados	-

Tabla 25 Mediciones actividad 2.01 Determinar los Requerimientos de Software a desarrollar

La actividad **2.02 Reformar los Requerimientos de Software** está conformada por dos acciones disímiles, - redefinición y modificación de los requerimientos de software -. Esta última debe ser llevada a cabo teniendo en cuenta lo señalado en el documento **Propuestas de cambio** confeccionado en la actividad **3.00 Validar la Especificación de la Especificación de Requerimientos de Software**. Como hasta el momento no se ha llevado a cabo la validación se los requerimientos, sólo se puso en marcha la actividad **2.02.01 Redefinir los Requerimientos de Software.** Cabe recordar que esta actividad centra su atención en la reelaboración de los requerimientos, como así también en la descomposición en requerimientos más simples. Al finalizar la actividad, se confeccionó un listado que contiene los cambios realizados.

Nombre de la variable	
Vínculos *elaborate* identificados	1
Vínculos *part_of* identificados	6

Tabla 26 Mediciones actividad 2.02.01 Redefinir los Requerimientos de Software

Al analizar los valores de las mediciones relacionadas con el vínculo *part_of*, se observó la necesidad de incorporar una variable a medir que indique el número de requerimientos que han sido particionados. En este caso de estudio la variable **requerimientos afectados** tiene una valoración de dos.

Al finalizar las actividades **2.01 Determinar los Requerimientos de Software a desarrollar y 2.02 Reformar los Requerimientos de Software** se advirtió que se había omitido la identificación de las restricciones que afectan al sistema a implementar. Para solucionar tal omisión se realizó nuevamente la actividad **2.01.02 Recolectar información del dominio del problema** con el objetivo de confeccionar el **Listado Restricciones.** Cabe aclarar que para identificar y detallar las restricciones, no hubo que realizar ninguna entrevista, sólo se utilizaron los conocimientos adquiridos durante las actividades que integran el proceso de requerimientos. El listado confeccionado fue tomado como input para realizar la actividad **2.01.02 Identificar los Requerimientos de Software**. Tales restricciones serán tratadas como requerimientos no funcionales.

Con la información recolectada y elaborada en las distintas actividades se confeccionó el documento de **Especificación de Requerimientos de Software**.

2.01.01 Recolectar la información del dominio del problema-	
Documentos consultados.	-
Personal entrevistado.	-
Entrevistas realizadas por personal	-
Actividades identificadas	-
Actividades identificadas por Objetivo del Sistema Requerimiento de Software	- - -
Restricciones identificadas	3
Mandates identificados por categoría	-
2.01.02 Identificar los Requerimientos de Software	
Entrevistas realizadas	-
Entrevistas realizadas por personal	-
Requerimientos identificados	4
Vínculos *generate* identificados	-
Vínculos *derive* identificados	-
Vínculos *is_a* (Restricciones identificadas)	4
Vínculos *based_on* identificados por categorías	-

Tabla 27 Mediciones obtenidas al identificar las restricciones

Las conclusiones obtenidas al poner en marcha el proceso **2.0 Especificar los Requerimientos de Software** fueron las siguientes:

- Se deberán incorporar a las especificaciones de los listados confeccionados el componente de información **Iteración**, el cual debe indicar el número de iteración en que se ha realizado.
- El componente de información **Estado** asociado a la entidad **Requerimiento de software** está definido de manera incompleta. La escala de valores que comprende este componente debe abarcar por lo menos algunos de los siguientes ítems; especificado, especificado y validado, en desarrollo, en codificación, terminado, terminado y validado, eliminado, cancelado.
- Un punto negativo observado es no haber especificado diferentes criterios para establecer las relaciones de dependencia entre los requerimientos, como así también se ha omitido especificar el criterio para establecer las fortalezas de las mismas.
- Se observó además que la escala utilizada para valorizar los componentes **Prioridad, Importancia y Estabilidad** es inadecuada (alta-media-baja).
- Se debe agregar a la especificación de la entidad **Requerimiento de Software** el componente de información **Tipo**, el cual debe indicar si el requerimiento es o no funcional.
- Al llevar a cabo dicho proceso, se evidenció la falta de una herramienta que automatice la gestión de los requerimientos. Si se hubiese contado con esta herramienta se podría haber agilizado el accionar de las tareas realizadas.

7.2.3 Validar la Especificación de Requerimientos de Software

Cabe recordar que a la actividad **3.00 Validar la Especificación de Requerimientos de Software** la integran las actividades **3.01 Determinar los Requerimientos de Software a validar, 3.02 Elaborar los CVP** y por último**, 3.03 Validar los requerimientos seleccionados.** Se estableció como criterio de selección de los distintos requerimientos a validar la relación existente entre estos y los CSFs. Finalizada la selección de los requerimientos a validar se confeccionaron los distintos CVPs. Para el caso de estudio se optó por emplear al prototipo como la técnica a utilizar para efectuar todas las validaciones que fuesen necesarias.

Tras haber concluido con las validaciones se confeccionó el documento **Propuestas de Cambio**, que detalla las observaciones, omisiones o errores que han sido encontrados. La mayoría de las consideraciones realizadas se refirieron a la modificación del contenido del componente **datos específicos** que poseen los requerimientos validados. Sólo se advirtió en la validación del **CVP5 Validación asistencia docente** la incorporación de funcionalidades que no habrían sido debidamente especificadas.

3.01 Determinar los Requerimientos de Software a validar	
Cantidad de Requerimientos de Software seleccionados	6
3.02 Elaborar los CVP	
Vínculos *develop_for*	
Identificados	6
Modificados	-
Eliminados	-
Vínculos *is_a*	
Identificados	5
Modificados	-
Eliminados	-
Vínculos *based_on*	-
Vínculos *used_by*	
Identificados	12
Modificados	-
Eliminados	-
3.03 Validar los elementos seleccionados	
Vínculos *generate*	
Identificados	5
Modificados	-
Eliminados	-

Tabla 28 Mediciones actividad 3.00 Validar la Especificación de Requerimientos de Software

Al analizar las mediciones realizadas, se decidió incorporar dos nuevas variables a medir. La primera, debe indicar la cantidad de recursos afectados para realizar las validaciones. Esta medición se incorpora debido a que en la especificación de los distintos vínculos *used_by* puede repetirse un recurso a utilizar. Para el caso de estudio dicha variable posee un valor de tres recursos. La segunda variable a medir que se adiciona es **vínculos generate por tipo.** Esta medición se incorpora debido a que un ítem DEO puede ser categorizado como una simple observación, una omisión o un error.

Finalizado el proceso de validación se llevó a cabo la actividad **2.02.02 Modificar los Requerimientos de Software y 2.03 Documentar los Requerimientos de Software,** para

realizar las modificaciones necesarias debido a lo señalado en el documento **Propuestas de Cambio**.

2.02.02 Modificar los Requerimientos de Software a validar	
Vínculos *depend_on*	
Modificados	-
Eliminados	-
Agregados	2
Vínculos *part_of*	
Modificados	-
Eliminados	-
Agregados	2
Vínculos *managed_by*	
Modificados	-
Eliminados	-
Agregados	-
Vínculos *allocated_to*	
Modificados	-
Eliminados	-
Agregados	4
Requerimientos	
Modificados	5
Eliminados	-
Agregados	2

Tabla 29 Mediciones actividad 2.02.02 Modificar los Requerimientos de Software

7.2.4 Desarrollar la Arquitectura del Sistema

La actividad **Desarrollar la Arquitectura del Sistema** comienza con la actividad **4.01 Seleccionar los elementos a diseñar**. Al ser la primera iteración de este proceso, el listado que se confecciona en esta actividad, sólo contiene requerimientos de software. Posteriormente se determinó que la próxima actividad a seguir correspondería a **4.02 Establecer la estructura del Sistema,** la cual comienza con el análisis de los requerimientos que han sido seleccionados. Cabe señalar que no se identificó ningún **mandate** a tener en cuenta para la confección del diseño del sistema. Prosiguiendo con lo especificado en el proceso de desarrollo, se identificó y confeccionó sólo una alternativa referida al sistema a implementar. En dicha alternativa se han identificado las funciones que el sistema debe realizar y la respectiva información de asignación de requerimientos. Cabe mencionar que en esta actividad no se ha especificado ningún vínculo *used_by* debido a que la confección del sistema no emplea recursos con características especiales. Es importante destacar que la alternativa descripta no se relaciona con ningún sistema externo. Como consecuencia de haber identificado una única alternativa, la misma fue inmediatamente confirmanda. Para detallar tal confirmación se confeccionó el vínculo *create* correspondiente.

Finalizada la definición de la estructura del sistema, se llevó a cabo nuevamente la actividad **Seleccionar los elementos a diseñar**, pero tal selección tuvo como propósito la definición de los subsistemas. Para realizar la selección de requerimientos se tuvo en cuenta si los mismos se relacionaban con las funciones **FNC1 Automatizar la información relacionada con el accionar académico de los alumnos y FNC2 Automatizar la**

información relacionada con la actividad docente. En la definición del subsistema no se detectaron **mandates** para la confección de los subsistemas y se identificó sólo una alternativa por subsistema a desarrollar. En esta actividad no se identificaron las funciones que los subsistemas deben realizar. Cabe recordar que la identificación de las funciones es optativa según lo especificado en el proceso de desarrollo. Al confirmar las alternativas de los subsistemas, se especificaron los vínculos *part_of* y se registraron en el componente de información **Compuesto por** del sistema correspondiente. Además se confeccionó un diagrama de paquetes con el propósito de graficar la arquitectura del sistema resultante. El identificador del diagrama elaborado se registró en el componente **Modelado por** de la entidad sistema.

Finalizada la definición de los dos subsistemas, se optó por seleccionar los requerimientos involucrados en la confección de los componentes de diseño necesarios para implementar el subsistema **Gestión de Alumnos**. Tal como en actividades anteriores, no se detectaron **mandates** a cumplimentar para la confección de dichos componentes. Tras el análisis de los requerimientos, se identificaron cuatro componentes para la definición del subsistema. Por cada componente se identificaron las funciones, las dependencias entre los componentes y la respectiva información de asignación de requerimientos. El criterio empleado para establecer las dependencias se limitó a verificar si los requerimientos que dirigen la confección de los componentes dependen entre sí. Al confirmar la implementación de los componentes se especificaron los vínculos *part_of* los cuales fueron registrados en la entidad subsistema correspondiente.

Al finalizar cada una de las iteraciones correspondientes a la actividad **4.00 Desarrollar la Arquitectura del Sistema** se efectuaron las mediciones correspondientes. Al analizar dichas mediciones se ha observado la necesidad de adicionar nuevas variables a medir como ser **Vínculos *performs* por Subsistema** o llegado el caso **Vínculos *perfoms* por Componente.** Cabe aclarar que la variable a medir **Cantidad de Entidades de diseño seleccionadas** de la actividad **4.01 Seleccionar los elementos a diseñar** sólo indica las entidades seleccionadas sin contar la repetición de las mismas.

Primera micro iteración	
4.01 Seleccionar los Requerimientos de Software a diseñar	
Cantidad de Requerimientos de Software seleccionados	16
Cantidad de Entidades de diseño seleccionadas	-
Cantidad de Requerimientos de Software con Entidades de diseño ya elaboradas	-
4.02 Establecer la estructura del Sistema	
Cantidad de Mandates identificados por categoría	-
Vínculos *drive* identificados	16
Vínculos *based_on* identificados	-
Vínculos *define* identificados	1
Entidades de Sistema identificadas	1
Vínculos *performs* identificados	5
Vínculos *address* identificados	15
Vínculos *used_by* identificados	-
Vínculos *depends_on* identificados	-
Vínculos *allocated_to* identificados	22

Vínculos *satisfy* identificados	16
Segunda micro iteración	
4.01 Seleccionar los Requerimientos de Software a diseñar	
Cantidad de Requerimientos de Software seleccionados	9
Cantidad de Entidades de diseño seleccionadas	1
Cantidad de Requerimientos de Software con Entidades de diseño ya elaboradas	9
4.03 Establecer la estructura de los Subsistemas	
Cantidad de Mandates identificados por categoría	-
Vínculos *drive* identificados	9
Vínculos *based_on* identificados	-
Vínculos define identificados	2
Vínculos *performs* identificados	-
Vínculos *address* identificados	-
Vínculos *used_by* identificados	-
Vínculos *depend_on* identificados	
Vínculos *depends_on* identificados	-
Vínculos *allocated_to* identificados	11
Vínculos *satisfy* identificados	9
Vínculos *create* identificados	2
Vínculos *part_of* identificados	2
Tercer micro iteración	
4.01 Seleccionar los Requerimientos de Software a diseñar	
Cantidad de Requerimientos de Software seleccionados	7
Cantidad de Entidades de diseño seleccionadas	2
Cantidad de Requerimientos de Software con Entidades de diseño ya elaboradas	7
4.04 Establecer la estructura de los Componentes	
Cantidad de Mandates identificados por categoría	-
Vínculos *drive* identificados	7
Vínculos *based_on* identificados	-
Vínculos define identificados	4
Cantidad de entidades de Componentes identificadas	4
Cantidad de Funciones identificadas	9
Vínculos *performs* identificados	9
Vínculos *address* identificados	11
Vínculos *used_by* identificados	-
Vínculos *depend_on* identificados	-
Vínculos *depends_on* identificados	-
Vínculos *allocated* identificados	11
Vínculos *satisfied* identificados	7
Vínculos *create* identificados	4
Vínculos *part_of* identificados	4

Tabla 30 Mediciones actividad 4.00Desarrollar la Arquitectura del Sistema

Las conclusiones obtenidas al finalizar la actividad **4.0 Desarrollar la Arquitectura del Sistema** fueron las siguientes:

- Se evidenció la necesidad de contar con una herramienta que automatice las actividades relacionadas con la asignación de los requerimientos.
- En la especificación del proceso no se han descripto con suficiente detalle las acciones que se relacionan con la confección de los vínculos *drive, allocated_to, satisfy, address* y *depend_on* cuando se selecciona un requerimiento que ha sido particionado o reelaborado por otro. Esta situación se ha resuelto de dos maneras. Cuando se confecciona el vínculo *allocated_to* correspondiente a un requerimiento que está particionado, también se confeccionan los respectivos vínculos *allocated_to* por cada uno de los requerimientos que intervienen en la partición. Para la confección de los restantes vínculos de traceability no se considera si el requerimiento que interviene en el vínculo ha sido particionado.

- La sola definición del componente de información **Modelado por** en las entidades **Sistema, Subsistema y Componente** no caracteriza de manera detallada la relación entre estas entidades con lo elaborado al seleccionar una metodología de diseño específica.

Al finalizar el caso de estudio se obtuvieron las siguientes conclusiones:

- El caso de estudio ha servido para demostrar la viabilidad de la utilización del proceso de desarrollo propuesto.
- Dicho proceso puede ser fácilmente adaptado a las particularidades de la aplicación a desarrollar, especialmente para los proyectos catalogados como pequeños.
- Su implementación se realizó fácilmente, ya que no se necesita contar con una estructura organizacional compleja.

Sin embargo se han observado algunos aspectos negativos en la especificación del proceso de desarrollo:

- La documentación obtenida de las actividades definidas en el proceso de desarrollo no reflejan lo acontecido en las distintas iteraciones.
- Se ha evidenciado la falta de detalle en la definición de las entidades y vínculos que se confeccionan durante el desarrollo. El componente de información **Estado** no especifica las posibles tipificaciones que puede tener. Otro ejemplo lo representa el componente de información **Autor.** Este componente debería haberse definido como una **relación de contribución** [Gotel, 1995]. Además, sólo se han especificado los atributos elementales de los vínculos de traceability definidos en el metamodelo.
- Al especificar las variables a medir no se detallaron los aspectos más significativos a computar.
- Un aspecto que influyó negativamente en el desarrollo del caso de estudio fue no contar con una herramienta que automatice lo confeccionado en las distintas actividades.
- En el caso de estudio no se realizaron actividades de fundamentación. Esto se debió principalmente al entendimiento que se posee del dominio del problema y a la falta de una herramienta que agilice dichas actividades.

7.3 Ejemplificando la traceability en el proceso de desarrollo definido

En este apartado se ejemplifica el funcionamiento de los mecanismos de traceability implementados en el proceso de desarrollo. En primer término se analizan los mecanismos

que proporcionan información referida a la gestión del cambio de los requerimientos. Posteriormente, se analizan los mecanismos que proporcionan la información relacionada con la asignación de los requerimientos en los componentes elaborados durante las actividades de diseño.

7.3.1 Ejemplo Gestión cambio de requerimientos

El cambio representa una de las características principales de los requerimientos. Es por esta razón que la gestión de dichos cambios debe contemplar entre otras cuestiones la evaluación del impacto que se ha producido [Madhavji, 1992]. Al responder el interrogante ***¿qué entidades son afectadas por el cambio de un determinado objetivo del sistema?*** se muestra la implementación en el proceso de desarrollo de los mecanismos de traceability necesarios para realizar tal evaluación.

En primer lugar, para poder contestar este interrogante se debe identificar y analizar el **Objetivo del Sistema** afectado al cambio producido. Por el momento, es indistinto saber si el cambio hace referencia a alguna modificación o a la eliminación del objetivo del sistema. La entidad que detalla la información referida al objetivo del sistema se especifica en el componente de información **Genera a.** En tal componente se encuentran los vínculos *generate,* los cuales indican los requerimientos generados a partir de dicho objetivo. Una vez identificados los requerimientos, se deben analizar cuáles son las diversas entidades de diseño que se vinculan con tales requerimientos. Dicha información se obtiene examinando el contenido del componente **Información Asignación** (vínculos *allocated_to)* de los respectivos requerimientos seleccionados. Esta información conjuntamente con los requerimientos de software identificados, indican las distintas entidades afectadas por el cambio realizado a un objetivo del sistema.

Para ejemplificar la evaluación de impacto a partir de lo realizado en el caso de estudio, se selecciona el objetivo del sistema **OBJ1 Seguimiento de los alumnos.** Al examinar el componente **Genera a** se observa la presencia de tres vínculos *generate*. En dichos vínculos se especifican los **REQ1 Información correspondiente a los alumnos, REQ2 Confeccionar la libreta de calificaciones de los alumnos, REQ3 Disponer de información cualitativa referida al desempeño de los alumnos**.

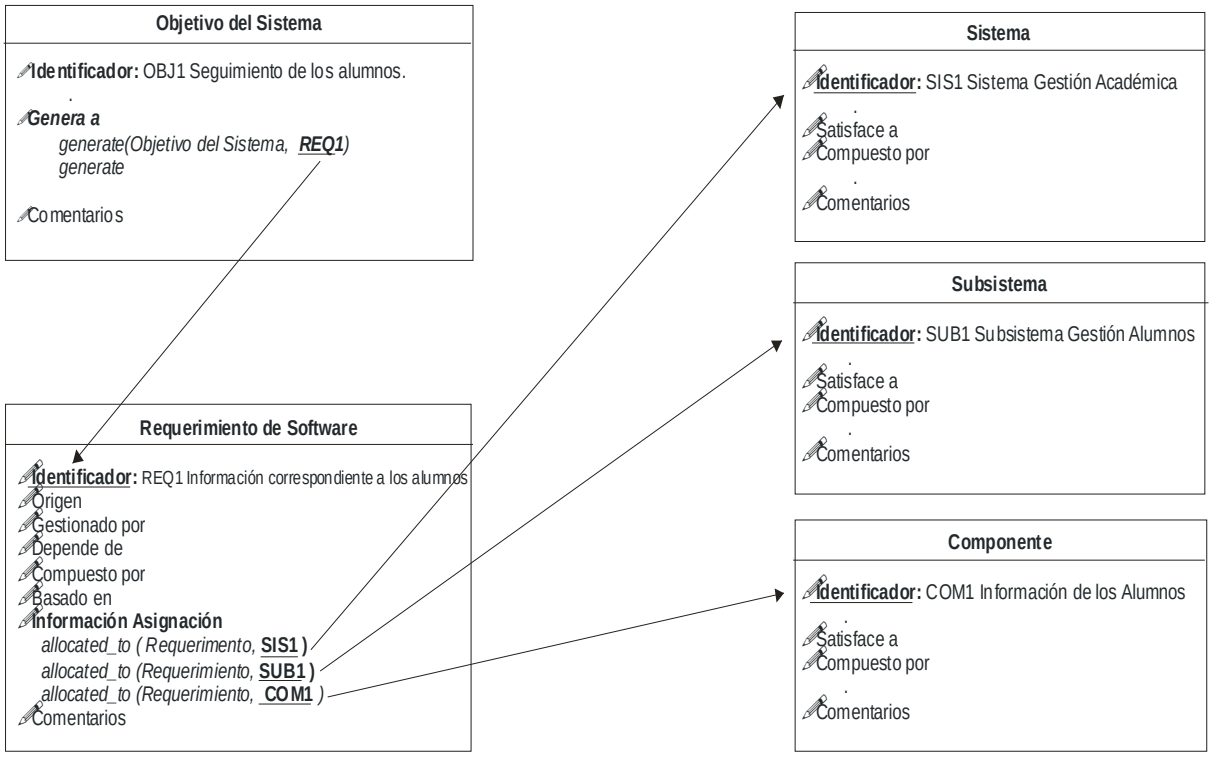


Figura 7.1 Evaluación de Impacto- Cambio de un Objetivo del Sistema

Una vez precisados los requerimientos generados a partir del **OBJ1 Seguimiento de los alumnos,** se deben identificar las diversas entidades confeccionadas en las actividades de diseño. Para obtener tal información se debe examinar por cada requerimiento identificado el contenido del componente **Información Asignación.** En estos componentes se han registrado once vínculos *allocated_to,* los cuales corresponden a las entidades **SIS1 Sistema Gestión Académica, SUB1 SubSistema Gestión Alumnos, COM1 Información de alumnos y COM2 Desempeño académico**.

El análisis de evaluación de impacto realizado hasta el momento no ha contemplado la evolución de las distintas entidades. La información relacionada con la evolución de los requerimientos se describe por medio de los vínculos *derive, elaborate y part_of*. Los dos primeros vínculos se registran en el componente **Origen**, en vez los vínculos *part_of* se registran en el componente **Compuesto por**.

Siguiendo con el ejemplo, si al examinar el componente **Origen** de un determinado requerimiento posee el vínculo *derive* y éste posee en la entidad destino el requerimiento generado a partir del objetivo del sistema seleccionado, se lo debe identificar como una entidad afectada por el cambio realizado. En cambio, si al examinar el componente **Origen** posee el vínculo *elaborate,* se identifican como entidades afectadas por el cambio aquellos requerimientos que en la entidad destino del vínculo *elaborate* tengan especificado el requerimiento generado a partir del objetivo del sistema seleccionado.

Si un requerimiento afectado por el cambio realizado ha sido particionado, los requerimientos que lo componen también deben seleccionarse para analizar la evaluación de impacto del cambio realizado. Tal información se describe por medio de los vínculos *part_of* los cuales están registrados en el componente **Compuesto por** del respectivo requerimiento. Los requerimientos que se detallan en la entidad destino de los vínculos *part_of* también deben identificarse como entidades afectadas por el cambio producido.

Sin embargo, para obtener mayor grado de detalle referido al alcance del cambio es necesario analizar si los requerimientos dependen de otros requerimientos. Si existiesen tales relaciones se especifican con los respectivos vínculos *depend_on.* Una vez identificados los requerimientos involucrados en tales dependencias, se debe analizar si los mismos resultan o no afectados por la modificación de objetivo del sistema. Una vez individualizados los requerimientos afectados por el cambio de un objetivo del sistema, se debe examinar en cada uno de ellos el contenido del componente **Información Asignación.**

También hay que analizar cómo repercute la evolución de la arquitectura del sistema en la evaluación de impacto de los cambios realizados. La primera observación de este hecho es similar a lo expuesto anteriormente. Un sistema, subsistema o componente puede ser particionado o depender de otros. En tal caso hay que examinar y analizar las respectivas relaciones y evaluar si las entidades identificadas en dichas relaciones se ven afectadas por el cambio producido. Además hay que tener en cuenta que se pueden confeccionar diversas alternativas para la confección de un sistema, subsistema o componente y sólo una de estas puede ser confirmada. Es por esta razón que no sólo hay que examinar el contenido del componente **Información Asignación** de los requerimientos, sino también verificar que las entidades especificadas en los vínculos *allocated_to* estén confirmadas.

7.3.2 Ejemplo Asignación de requerimientos

[Wright, 1991] sugiere incorporar a los existentes modelos de ciclo de vida del software una nueva etapa - **Asignación (Allocation)** -. Esta actividad deberá mapear los requerimientos en los distintos productos entregables que han sido confeccionados [Wright, 1991]. Al responder el interrogante **¿cómo contribuye un determinado componente en el logro de las necesidades organizacionales?** se muestra la implementación en el proceso de desarrollo de los mecanismos de traceability necesarios para realizar tal asignación.

Como en el ejemplo, para poder responder tal interrogante se debe empezar por identificar y analizar la entidad **Componente** seleccionada**.** En dicha entidad se especifica el componente de información **satisface a.** En esta entidad se encuentran los vínculos *satisfy*. Este vínculo tiene como propósito indicar el requerimiento que la implementación del componente seleccionado debe satisfacer. Por cada uno de los requerimientos identificados en estos vínculos se deben analizar aquellos que en el contendido del componente **Origen**

posean en el vínculo *generate* (identifica el objetivo del sistema que lo ha generado). Por cada uno de los objetivos del sistema identificados en los vínculos *generate* se debe examinar el contenido del componente **Justificado por.** En dicho componente se encuentra el vínculo *justify*, el cual especifica la necesidad organizacional que lo justifica. De este modo se logra vincular el componente elaborado en el proceso de desarrollo de los requerimientos con las necesidades organizacionales.

Es importante destacar dos restricciones del proceso de desarrollo que afecta a la elaboración de esta información. La primera es que un objetivo del sistema puede estar justificado por una necesidad organizacional y en segundo lugar un requerimiento puede ser generado por un solo objetivo del sistema.

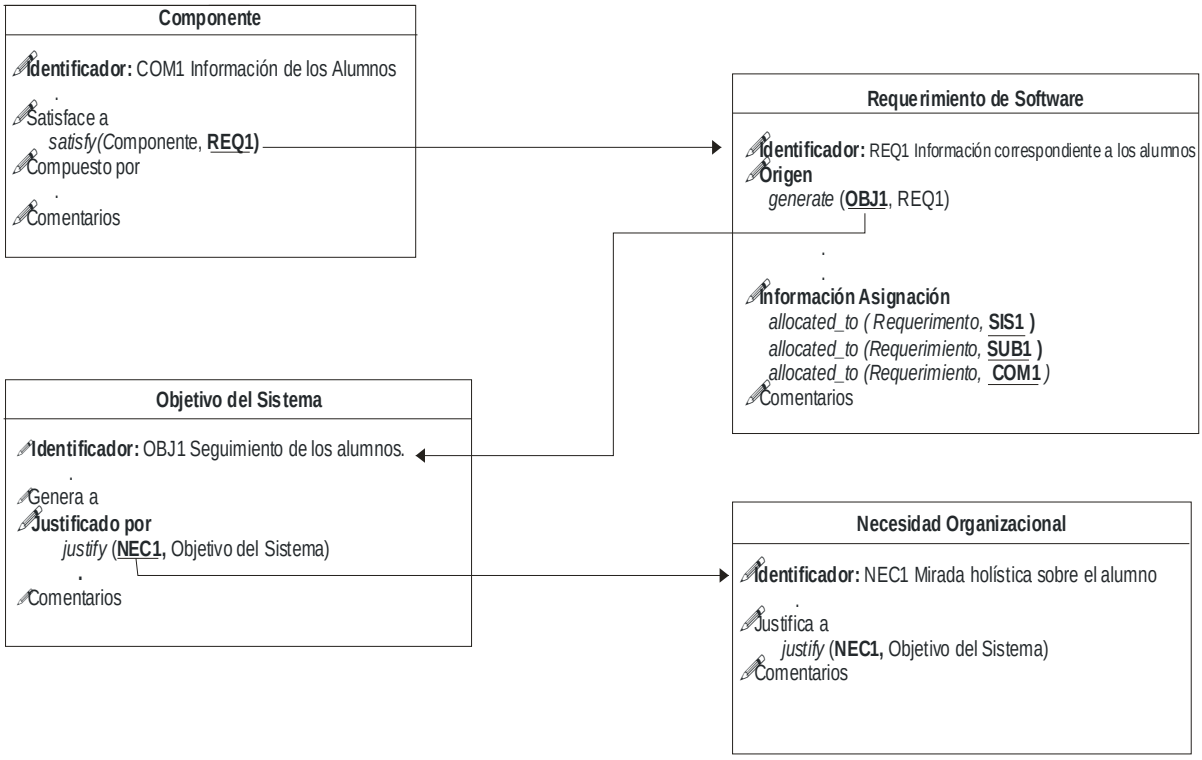


Figura 7.2 Información Asignación Requerimientos

Para responder dicho interrogante con lo confeccionado en el caso de estudio se selecciona el componente **COM1 Información de Alumnos** que posee en el componente **Satisface a** dos vínculos *satisfy.* Dichos vínculos detallan en la entidad destino los requerimientos **REQ1 información correspondiente a los alumnos y REQ11 Información relacionada al pago de la mensualidad de los alumnos** .Por cada uno de los requerimientos se examina el componente de información **Origen.** En ambos componentes se encuentran los vínculos *generate,* los cuales detallan el objetivo del sistema que lo ha generado. El REQ1 ha sido generado a partir del objetivo **OBJ1 Seguimiento de los alumnos** y el REQ11 ha sido generado a partir del objetivo **OBJ4 Gestión de los grupos.**

Una vez identificados los objetivos, se examina el componente **Justificado por**, el cual tiene el propósito de detallar por medio del vínculo *justify* la necesidad organizacional que lo justifica. En el OBJ1 se detalla a la necesidad **NEC1 Mirada holística sobre el alumno** y en el objetivo OBJ4 se detalla a **NEC2 Mirada sobre la organización interna.**

En síntesis, el componente **COM1 Información de Alumnos** contribuye al logro tanto de la necesidad **NEC1 Mirada holística sobre el alumno** como a la **NEC2 Mirada sobre la organización interna.**

En este apartado se describe lo realizado en la prueba de concepto, que tuvo como caso de estudio la implementación de un sistema de gestión académica para el instituto **Evergreen School of English**. Finalizado el mismo, se ejemplificó la manera en que se implementaron los mecanismos de traceability en el proceso de desarrollo definido en el capítulo seis

CAPÍTULO 8 CONCLUSIONES Y TRABAJOS FUTUROS

8.1 Conclusiones

El trabajo realizado consistió en la especificación de un proceso de desarrollo de software que implemente implícitamente la RT utilizando como referencia al metamodelo descripto en [Ramesh *et. al*, 2001]. Para validar los resultados alcanzados se desarrolló una aplicación de software (para un instituto de enseñanza del idioma ingles) siguiendo el modelo de proceso especificado. Esta prueba de concepto del modelo propuesto no incluyó las actividades de codificación.

8.2 Contribuciones

Como se mencionó en la Introducción, el objetivo de esta tesis ha sido demostrar la factibilidad de implementar un proceso que implemente la RT. Básicamente ello consiste en:

- Definir explícitamente un proceso de desarrollo de software basado e implementado a partir de un metamodelo.
- Definir un proceso de desarrollo con mecanismos traceability implícitamente definidos, sin generar sobrecarga de trabajo en la actividades que lo conforman.
- Ejecutar una prueba de concepto del proceso propuesto mediante un desarrollo de un sistema específico.

8.3 Futuros trabajos

El conocimiento de la organización en la que se inserta un sistema software es una pieza clave en todos los pasos posteriores. Un camino a investigar para mejorar ese conocimiento es la incorporación de un proceso de un LEL (Léxico Extendido del Lenguaje) referido a la organización afectada.

También se advierte la importancia de no sólo percibir a los CSFs como recursos críticos. Se deberá redefinir la actividad encargada de identificar los CSFs para llevarla a cabo durante la identificación de las necesidades, objetivos del sistema y requerimientos de software.

Además, se deberá alinear el proceso con las pautas establecidas en la norma ISO/IEC 12207, en particular en lo referido a los *procesos de soporte*. Se estima comenzar por los procesos de documentación y de gestión de la configuración. La implementación de los

distintos *procesos organizacionales* de la norma mencionada se encarará posteriormente. También deberán introducirse las mejoras identificadas en el caso de estudio.

Tales extensiones tienen como propósito la confección de un **Ambiente de Desarrollo de Software,** en donde los procesos y componentes que se definan tengan implícitamente incorporados los mecanismos de traceability.

BIBLIOGRAFÍA

[Basili *et al*, 1991] Basili Victor, Musa John. "The Future Engineering of Software: A Management Perspective". En: IEEE Computer September, 1991, pág 90-96.

[Basili, 1989] Basili Victor, "The Software Business" Chapter 1 En: http://www.cs.umd.edu/users/mvz/mswe609/book/chapter1.pdf

[Boehm, 1979] Boehm, B. W. "Software engineering: R&D trends and defense needs". In *Research Directions in Software technology*. P Wegner, ED. Cambridge, MA: MIT Press, 1979

[Brooks, 1987] Brooks, Frederick P. "No Silver Bullet: Essence and Accidents of Software Engineering". En: IEEE Computer, 20, (4) April 1987, pág 10-19.

[Curtis *et al*, 1992] Curtis, Bill. Kellmer, Marc. Over, Jim. "Process Modelling.". En: Communications of the ACM, 35(9),1992, pág 75-90

[Davis, 1990] Davis, A.M. The analysis and specification of systems and software requirements. In *Systems and Software Requirements Engineering.* IEEE Computer Society Press, 1990, 119–144.

[Dömges *et al*, 1998] Dömges, Ralf. Pohl, Klaus. "Adapting Traceability to project specific need environments". Communications of the ACM. December 1998/Vol.41,Nro12. pag54-62.

[Durán Toro *et al*, 2000] Durán Toro, Amador. Bernárdez Jiménez Beatriz. "Metodología para la Elicitación de Requisitos de Sistemas Software. Universidad de Sevilla. Informe Técnico LSI-2000-10. Departamento de Lenguajes y Sistemas Informáticos. Facultad de Informática y Estadística. Sevilla, Octubre de 2000.

[El Eman, 2001] El Eman, Khaled. *Description of the SWEBOK Knowledge Area Software Engineering Process (Version 0.9).* National Research Council of Canada. Institute for Information Technology, NCR 44188, March 2001.

[Freeman, 1987] Freeman, Peter. "Psst, What Is Software, Anyway?". En: *Software Perspectives. The System in the Message,* Addison–Wesley. Reading Mass, 1987.

[Ghezzi *et al.*, 1991] Ghezzi, Carlo. Jazayeri, Mazayeri. Mandrioli, Dino. *Fundamentales of Software Engineering*. Prentice-Hall International, Englewood Cliffs, New Jersey,1991.

[Gibbs 1994] Gibbs, Wayt. "Scientific American" En : Scientific American. September. 1994.

[Gotel *et al*, 1994] Gotel, Ornela C. Z. Finkelstein, Anthony. "An Analysis of the Requirements Traceability problem" En: Proceedings of the First International Conference on Requirements Engineering (ICRE'94), IEEE Computer Society Press, Colorado Springs, Colorado, USA, April 1994.Páginas 71-81

[Gotel *et al*, 1995] Gotel, Ornela C.Z. Finkelstein, Anthony. "Contribution Structures." En: Proceedings of the Second IEEE International Symposium on Requirements Engineering(RE'95), IEEE Computer Society Press, York, U.K. March 27-29,1995. páginas 100-107

[Gotel, 1995] Gotel, O. C.Z .Contribution Structures for Requirements Traceability. Ph.D. Thesis. Imperial College of Science, Technology and Medicine. University of London. August 1995

[IEEE Std-830, 1984] "An American National Standard IEEE Guide to software Requirements Specifications Std-830".1984.

[IEEE Std-610.12, 1990] "Standard glossary of Software Engineering Terminology". 1990.

[IEEE STD-830, 1998] "Recommended Practice for Software Requirements Specifications. Software Engineering Standards". Committee of the IEEE Computer Society, 1998.

[Jarke *et al*., 1993] Jarke, Matthias. Pohl, Klaus. "Establishing visions in context:Towards a Model of Requirements Process". En: Proc. of the Int. Conference on Information Systems, Orlando, Florida,1993.

[Jarke, 1998] "Requirements tracing". En Communications of the ACM. Vol 41 Nro12.,December 1998, pág 32-36

[Kenny, 1996] Kenny Christy. "Requirements Traceability Cmpt 856 Project". 1996

[Khodabandeh *et al,* 1994] Khodabandeh, Arash. Palazzi Paolo. "Software development: People, Process, Technology". En: Proceedings of the 1994 CERN School of Computing, Sopron, Hungary.

[Lan Tran et. al, 1995] Lan Tran, Tuyet. Sherif Joseph S. "Quality Function Deployment (QFD): An Effective Technique For Requirements Acquisition and Reuse." IEEE 1995

[Lavazza *et al.*, 2000] Lavazza, Luigi. Valetto, Giuseppe. "Enhancing Requirements and Change Management through Process Modelling and Measurement", Fourth International Conference on Requirements Engineering, ICRE'2000, Schaumburg, Illinois, USA, Junio 19-23, 2000, pp 106-115.

[Loucopoulos *et al*.,1995] Loucopoulos, P. Karakostas, V . *System Requirements Engineering.* London, McGraw-Hill, 1995.

[Madhavji *et. al* , 1991] Madhavji. Nazim H. Schäfer Wilhelm. "Prism – Methodology and Process-Oriented Environment". IEEE Transactions on Software Engineering, Vol 17. Nro 12 December 1991.

[Madhavji, 1991] Madhavji. Nazim H. "The Prism Model of Changes". in Proc. 13th Int. Conf. on Software Pro. Los Alamitos, CA: IEEE Computer Soc. Press, 1991, pp 166-177.

[Madhavji, 1992] Madhavji. Nazim H. "Environment Evolution: The Prism Model of Changes". IEEE Transactions on Software Engineering, Vol 18. Nro 5 May 1992.

[McConnell, 1997] McConnell, Steven. Desarrollo y gestión de proyectos informáticos. McGraw – Hill España.

[McConnell, 2001] McConnell, Steven. "Who needs software engineering?". En: IEEE Software, January/February, 2001, pág 5-8

[Nuseibeh *et al.*, 2000] Nuseibeh, Bashar. Easterbrook, Steve. "Requirement Engineering: A Roadmap" International Conference on Software Engineering, ICSE 2000, Future of SE Track 2000 (2000) 35-46.

[Olson *et al*., 1993] Olson, T. Parker Gates, L. Mullaney, J. Over, J. Reizer, N. Kellner,M. Phillips, R. DiGennaro,S. Special Report CMU/SEI-93-SR-007 Carnegie Mellon – Software Engineering Institute. June 1993.

[Pearson, 1996] Pearson, Justin K. "Requirements Traceability and Formal Software Development or a Further Analysis of Requirements Traceability.". En: Computer Science Department Royal Holloway University of London. Egham October 1996

[Perry *et al.*, 1988] Perry, Dewayne. Kaiser, Gail. "Models of software development environments". January 1988.

[Pfleeger, 2002] Pfleeger Shari Lawrence. *Ingeniería de software teoría y práctica*. Buenos Aires, Prentice Hall, 2002.1ra edición.

[Piattini, *et al.*, 2003] Piattini, Mario G. García Rubio, Félix O. Calidad en el desarrollo y mantenimiento del software. Ra-Ma Editorial. Madrid, España.2003

[Pinheiro *et al.*, 1996] Pinheiro, Francisco A. C. Goguen, Joseph A. "An Object Oriented Tool of Tracing Requirement" En: IEEE Software. March 1996, páginas 52-64

[Pinheiro, 2000] Pinheiro, Francisco A C. "Formal and Informal Aspects of Requirements Tracing"- III Workshop de Engenharia de Requisitos, WER'2000, Rio de Janeiro, Brazil, Julio 2000.

[Pressman, 1998] Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico.* Madrid, Mac Graw. Hill, 1998. Cuarta Edición

[Ramamoorthy *et. al*, 1990] Ramamoorthy, C.V. Usuda, Yutaka. Prakash, Atul. Tsai, W. T. "The Evolution Support Environment System". En IEEE Transactions on Software Engineering. Vol 16 Nro 11 November 1990.

[Ramesh *et. al*, 1995] Ramesh, Bala. Stubbs, Curtis. Powers, Timothy. Eduards, Michael. "Lessons Learned from Implementing Requirements Traceability ".- April 1995.

[Ramesh *et. al*, 1995b] Ramesh, Bala. Stubbs, Curtis. Powers, Timothy. Eduards, Michael. "Implementing Requirements Traceability: A Case Study". Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE95) 1995.

[Ramesh, 1998] Ramesh Balasubramaniam. *"Factors Influencing Requirements Traceability Practice".* Communications of the ACM. December 1998/Vol 41. nro 12. Pág 38 - 44.

[Ramesh *et. al*, 2001] Ramesh, Bala. Pohl, Klaus. "*Toward Reference Models for Requirements Traceability"* .En: IEEE Transactions on Software Engineering. Vol.27.Nro. January 2001.

[Russ *et al*, 2000] Russ Melissa L. McGregor John D. "A Software Development Process for Small Projects." En: IEEE Software, September/October 2000, pág 96-101

[Sawyer *et al*, 1997] Sawyer P. Sommerville I. Viller S. Requirements Process Improvement through The Phased Introduction of Good Practice. Software Process Improvement and Practice. Vol 3, Issue1, pp. 19-34 March 1997.

[SEL-81-305, 1992] "Recommended Approach to Software Development. Revision 3". En Software Engineering Laboratory Series. NASA, June 1992.

[Singh Raghu, 1996] Singh Raghu. International Standard ISO/IEC 12207 Software Life Cycle Processes. Software Process - Improvement and Practice, Vol.2, pp 35-50. 1996.

[Sommerville, 1995] Sommerville Ian. *Software Engineering*. England, Addison Wesley, 1995. Fifth Edition

[Sommerville *et. al,* 1997] Sommerville Ian. Sawyer P. *Requirements Engineering. A Good Practice Guide*. 1997. Chichester: John Wiley and Sons.

[Sommerville, 2002] Sommerville Ian. *Ingeniería de software*. México, Pearson Educación, 2002. Sexta Edición

[Standish, 1999] The Standish Group International, INC. "The CHAOS report. The 3 pillars of project success" 1999.

[Wiegers, 1999] Wiegers, Karl. "Writing Good Requirements" may 1999 software Development magazine.

[Wieringa, 1995] Wieringa, Roel. "An Introduction to Requirements Traceability"- Technical report, Faculty of Mathematics and Computer Science, Vrije Universiteit, Esprit Project 2RARE, November 1 (1995).

[Wright, 1991] Wright, S. Requirements Traceability - What? Why? and How?, Tools and Techniques for Maintaining Traceability During Design, IEEE Colloquium, Computing and Control Division, Professional Group C1 (Software Engineering), U.K., Digest Number:1991/180, December 2, pp. 1/1-1/2. 1991.

[Yourdon, 1992] Yourdon, Edward*. Decline an Fall of the American Programmer.* Englewood Cliffs, NJ. USA, Prentice Hall Inc, 1992.

[Yu *et al*, 1994] Yu, E. Mylopoulos, J. Understanding 'Why' in Software Process Modeling, Analysis and Design, Proc. 16th Int Conf. Software Eng., pp. 159-168, 1994.

[Yu, 1994] Yu, E. Weider D. "*Verifying Software Requirements: A Requirement Tracing Methodology and Its Software Tool-RADIX"* IEEE Journal On Selected Areas In Communications, Vol. 12, No. 2, 234-240-February 1994.