CAPÍTULO I

METODOLOGÍA ORIENTADA A OBJETOS

OBJETIVOS DEL CAPÍTULO:

- Instituir los antecedentes de la metodología orientada a objetos (MOO).
- Detallar los conceptos base de la MOO.
- Explicar los principios e importancia de la MOO
- Realizar ejemplos de aplicación para entender la orientación a la realidad de la metodología en mención.

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector deberá reconocer la importancia de la metodología orientado a objetos.
- El lector diferenciará a la MOO frente a otras metodologías de construcción del software como la metodología estructurada por ejemplo.
- El leedor diferenciará los conceptos de objeto y clase al plantear diversos ejemplos del tema.

1.1. INTRODUCCIÓN

La exigencia de software de calidad, que satisfagan los requerimientos del usuario actual, es todo un reto, ya que solicitan un alto grado de especialización debido al constante cambio de los diversos factores tecnológicos, económicos y sociales que influyen en la organización.

La organización para hacer frente a las exigencias del mercado actual, necesitan soluciones informáticas integrales, preparadas para soportar procesos exigidos por la actual coyuntura. Las soluciones informáticas deben ser construidas en el menor tiempo posible, deben cumplir con estándares de calidad internacional, flexibilidad, robustez y deben ser construidos en base a las necesidades de la organización.

La interrogante más famosa es sin duda: "¿Cómo satisfacer a los requerimientos del usuario actual?".

Después de muchos años de evolución en la construcción de software, encontramos la solución a la interrogante anterior al aplicar la Metodología Orientado a Objetos.

Las características de la Metodología Orientada a Objetos como la herencia, el polimorfismo y el encapsulamiento hacen posible la construcción rápida de software, caracterizado por la flexibilidad a cambios futuros, seguridad y robustez, logrando así satisfacer las exigencias del usuario actual.

El American National Standar Institute (ANSI), crea la organización no gubernamental y sin fines de lucro "Object Management Group (OMG)", dicha institución se encarga de definir los lineamientos y políticas para estandarizar a los procesos, técnicas, elementos, notaciones, etc., basados en la metodología orientada a objetos.

La técnica de modelamiento "Unified Modeling Language (UML)", el proceso de construcción de software "Unified Process" fueron aceptados por la OMG como estándares,

convirtiéndose en la técnica de modelado y el proceso de construcción de software por excelencia de la Metodología Orientada a Objetos.

En este capítulo analizaremos la Metodología Orientada a Objetos desde el punto de vista práctico, explicaré las características y principios de la metodología de manera sencilla y precisa sin necesidad de leer textos adicionales para entender los diferentes tópicos citados en el presente capítulo.

1.2. METODOLOGIA ORIENTADO A OBJETOS

Es un conjunto de métodos, procesos y técnicas que guían la construcción de software; caracterizado por analizar, diseñar y ejecutar lo que sucede en la realidad. Los conceptos base de la Metodología Orientada a Objetos son los objetos y las clases. Permite el desarrollo rápido de software al utilizar la reutilización de componentes característica principal del lenguaje de programación orientado a objetos "JAVA". La Metodología Orientada a Objetos guía el proceso de construcción de software al utilizar el Proceso Unificado además de permitir a los profesionales inmiscuidos en el desarrollo del software expresar su trabajo en términos de diagramas al utilizar el UML.

1.3. BASE CONCEPTUAL DE LA METODOLOGIA ORIENTADO A OBJETOS

1.3.1. OBJETO

Es un ente real ó conceptual que posee características inherentes (atributos) y comportamiento identificable (métodos). El objeto es específico (único).

Son requisitos que deben cumplir los objetos

IDENTIDAD + COMPORTAMIENTO + ESTADO

Carmen Méndez

Carlos Lino

Luz Paz

Figura N° 01, Ejemplos de objetos referidos a la clase Persona.

1.3.2. CLASE

Es la colección de objetos que comparten atributos, funciones y métodos comunes.

Es una abstracción y no se refiere a objeto en particular.

Estas son genéricas, permitiendo modelar el mundo real.

Persona

Universidad

Automóvil

Figura N° 02, Ejemplos de Clases.

1.4. CARACTERISTICAS TEÓRICAS DE LA METODOLOGÍA ORIENTADO A OBJETOS.

1.4.1. ABSTRACCIÓN

Es la representación de las características esenciales de algo, sin incluir detalles irrelevantes.

Cada objeto en el sistema sirve como modelo de un agente abstracto que puede realizar trabajo, informar y cambiar su estado, también puede comunicarse con otros objetos en el sistema informático sin revelar cómo se implementan estas características.

1.4.2. PERSISTENCIA

Se refiere al tiempo de vida de un objeto. Cuando este reside en la memoria RAM, se dice que no es persistente, pero los que se almacenan en un medio permanente, en el disco duro, por ejemplo, se dice que son persistentes.

Ejemplo: La información de la base de datos son considerados persistentes por no alterarse con respeto al tiempo, la única manera de modificarlos es mediante el Structure Query language (SQL).

1.4.3. ENCAPSULAMIENTO

Consiste en contener en una clase datos y funciones, de forma que el acceso a los datos se permite sólo a través de los propios métodos del objeto.

Ninguna otra parte de la aplicación orientada a objetos debe operar directamente sobre los datos de otro objeto.

Empaquetamos en un objeto una pieza de información con comportamiento específico que actúa sobre esta información.

Con esta característica podemos limitar los efectos de cambios sobre el sistema.

1.4.4. POLIMORFISMO

Un mismo método puede presentar diferentes comportamientos, en función al contexto. Esta característica permite lograr la simplicidad y el orden en el ambiente de programación.

También se entiende como comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

1.4.5. HERENCIA

Propiedad que permite a la clase o subclase tener acceso a los atributos y métodos de otra conocida como clase padre o superclase.

La herencia permite a los programadores crear nuevas clases programando solo las diferencias con la clase padre.

Esta característica brinda facilidad de mantenimiento y hace posible la reutilización de componentes.

1.4.6 REUTILIZACIÓN DE COMPONENTES

Producida gracias a la característica de herencia de la Metodología Orientada a Objetos. La reutilización de componentes implica la construcción de software con equipo lógico que ya existe o que construyen terceros.

La ventaja principal que aporta es la generación de aplicaciones eficientes y de gran fiabilidad.

1.4.6.1. COMPONENTES

Son bloques de construcción de aplicaciones. Los "constructores de soluciones" utilizan muchos componentes software para la realización de sus sistemas.

El concepto de reutilización de componentes abarca el equipo lógico existente para tareas básicas y genéricas como impresión, procesadores de textos, hojas de cálculo, gráficos, diagramas de barras y dibujos. Todas estas piezas

deberían estar disponibles como componentes reutilizables para todas las soluciones que los necesiten.

1.4.6.2. OBTENCIÓN DE COMPONENTES

Si se acepta de forma universal el modelo de objetos para la construcción de componentes, significa que aparecerá una nueva industria de creación de componentes genéricos.

Las aplicaciones más habituales (procesadores, gráficos, etc.) se encontrarán disponibles en forma de componentes que se podrán integrar para conseguir nuevas aplicaciones de gran flexibilidad y potencia. Lo único necesario es el lenguaje de programación común para ensamblar los distintos componentes y construir la aplicación.

Para construir una aplicación compleja, se dispondrá de componentes genéricos fabricados por terceros y que se integrarán junto con los componentes específicos desarrollados para la aplicación concreta. Esto permite concentrar el esfuerzo del desarrollador en las partes de la aplicación que son de su competencia y poder así desarrollar soluciones potentes de forma muy rápida.

La idea es construir nuestros propios componentes, ello permite lograr especialización y obviamente la construcción rápida de software.

1.4.7 MODELO DE OBJETOS

El modelo de objetos formaliza la estructura y el comportamiento de los componentes para que puedan trabajar conjuntamente. El modelo ve a los componentes como objetos y

utiliza los conceptos de orientación a objetos para definir el marco de desarrollo de los mismos.

El problema es la falta de uniformidad en el desarrollo de los componentes para que puedan comunicarse y trabajar conjuntamente.

La finalidad del modelo de objetos ó análisis orientados a objetos para la construcción de la base de datos es evitar la presencia de nulos y redundancia logrando además la homogenización de los datos en ella.

1.4.8. MENSAJE:

Los objetos se comunican entre si mediante mensajes.
Cuando un objeto no esta capacitado para realizar una tarea, y otro lo esta; entonces el primer objeto envía un mensaje al segundo. Los mensajes resuelven los problemas derivados del encapsulamiento.

1.4.9. MÉTODO

También conocido como "*operación*" en la etapa de análisis. Son las diversas acciones (comportamientos) a realizar con las características de la clase. Por ejemplo, para el atributo nombre, podemos considerar los siguientes métodos:

agregarNombre()
grabarNombre()
modificarNombre()
eliminarNombre()

1.4.10. MODELO

Representa el sistema software desde una perspectiva específica. Al igual que la planta y el alzado de gráficos en dibujo técnico nos muestran la misma figura, vista desde

distintos ángulos, cada modelo nos permite visualizar un aspecto distinto del sistema.
Un modelo puede ser expresado en los diversos diagramas propuestos por el UML.
El modelo permite a los profesionales inmiscuidos en la construcción de software expresar su trabajo en términos de diagramas.

1.5. RESÚMEN DEL CAPÍTULO

Estimado lector en el capítulo número 01 del texto se revisó conceptos interesantes que forman parte de las generalidades de la metodología orientado a objetos.
Pero ¿solo teoría y más teoría como en los demás textos universitarios?, NO, un rotundo NO ya que, este capítulo introductorio marca el inicio para el desarrollo de un sistema informático basado en el Proceso Unificado. Además en el actual capítulo se citó diversos ejemplos orientados a la realidad que nos ayudaron a diferenciar importantes conceptos y principios de la metodología orientada a objetos.

1.6. EVALUACIÓN DE COMPETENCIAS DEL CAPÍTULO

- ¿Qué entiende usted por orientado a la realidad?
- Realizar un cuadro comparativo en base a ejemplos que permita diferenciar los conceptos objeto y clase.
- ¿Cuál es el papel del modelo de objetos dentro del proceso de construcción de software basado en la metodología orientado a objetos?

CAPITULO II

LENGUAJE DE MODELAMIENTO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Diferenciar los conceptos de vistas y diagramas del UML.
- Mostrar los diversos artefactos utilizados en cada uno de los diagramas considerados por el UML.
- Mostrar la relación existente entre la metodología orientada a objetos y el UML.

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector deberá diferenciar los diversos artefactos existentes en cada uno de los diagramas del UML.
- El leedor conoce los antecedentes del UML y los modelos más importantes que forjaron su existencia.
- El lector reconoce la diferencia entre vistas y diagramas.

2.1. INTRODUCCIÓN

La técnica de modelado estándar UML (Lenguaje de Modelado Unificado), capta cada vez más interés en el mundo de desarrollo de software, ya que permite visualizar, especificar y documentar todo el proceso de construcción del software de manera clara y sencilla.

Así como los arquitectos utilizan los planos para planificar las características de una construcción al detalle, los profesionales que participamos en la construcción de software podemos también planificar el proceso de construcción, obviamente no utilizamos planos pero si los diferentes diagramas del UML.

Con el uso de los diagramas controlamos los detalles de construcción del software, que sin el uso del UML sería complicado por la característica empírica del proceso de construcción sin previo análisis.

En este capítulo analizaremos el detalle del UML desde un punto de vista práctico e ilustrativo.

> La construcción de software es un arte; como toda expresión artística la paciencia y creatividad son habilidades indispensables conducentes al éxito de la construcción del software.

2.2. CONCEPTO

El UML, es una técnica de modelado, ***NO*** *una metodología ó el case de modelado Rational Rose*, muchas personas tienen esa confusión, espero después de esta clara explicación no haya lugar a dudas.

El UML, aparte de permitir la especificación, visualización, construcción y documentación de los elementos de un sistema software, también se utiliza en el modelado de procesos de

negocio u otros sistemas no-software. UML reúne una colección de las mejores prácticas en la ingeniería de software que han sido utilizadas con éxito para modelar sistemas grandes y complejos. Este lenguaje es una notación calificado por la OMG como técnica de modelado estándar.

El UML incrementa la productividad y la calidad del sistema, reduciendo incluso el ciclo de vida de construcción del software al ser utilizado adecuadamente por un proceso de construcción de software como el PROCESO UNIFICADO por ejemplo.

UML será predominante en los próximos años, debido a las siguientes razones:

- Fue creado por expertos en metodología, informática y tecnologías influyentes, sobre la base de las mejores prácticas en construcción de software de todos los tiempos.
- Muchas empresas líderes en tecnología patrocinaron su creación.
- Tiene la aceptación de la OMG como notación estándar.

2.3. ANTECEDENTES DEL UML

El UML ha sido creado por Grady Booch, Ivar Jacobson, y James Rumbaugh, teniendo como principal patrocinador a la corporación "Rational", utilizando información de otros importantes expertos en metodologías, vendedores de software, y usuarios finales.

El objetivo de su creación fue unificar los diversos sistemas que había y crear un lenguaje de modelado con las mejores características de cada uno.

Figura N° 03, Creadores de Lenguaje de Modelamiento Unificado.

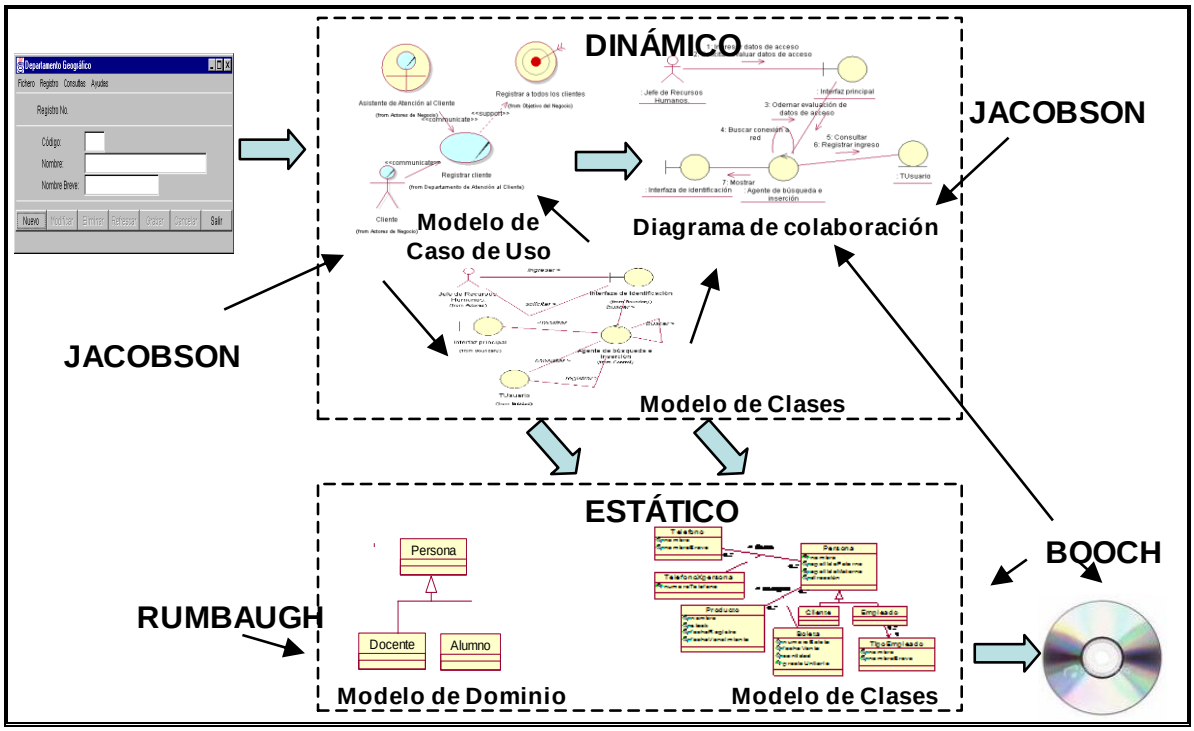


Figura 04, Aportes significativos de los métodos originales de Grady Booch, Ivar Jacobson, y James Rumbaugh al UML.

El UML nace en el año 1995, el detalle de su evolución lo observamos en la figura N° 05.

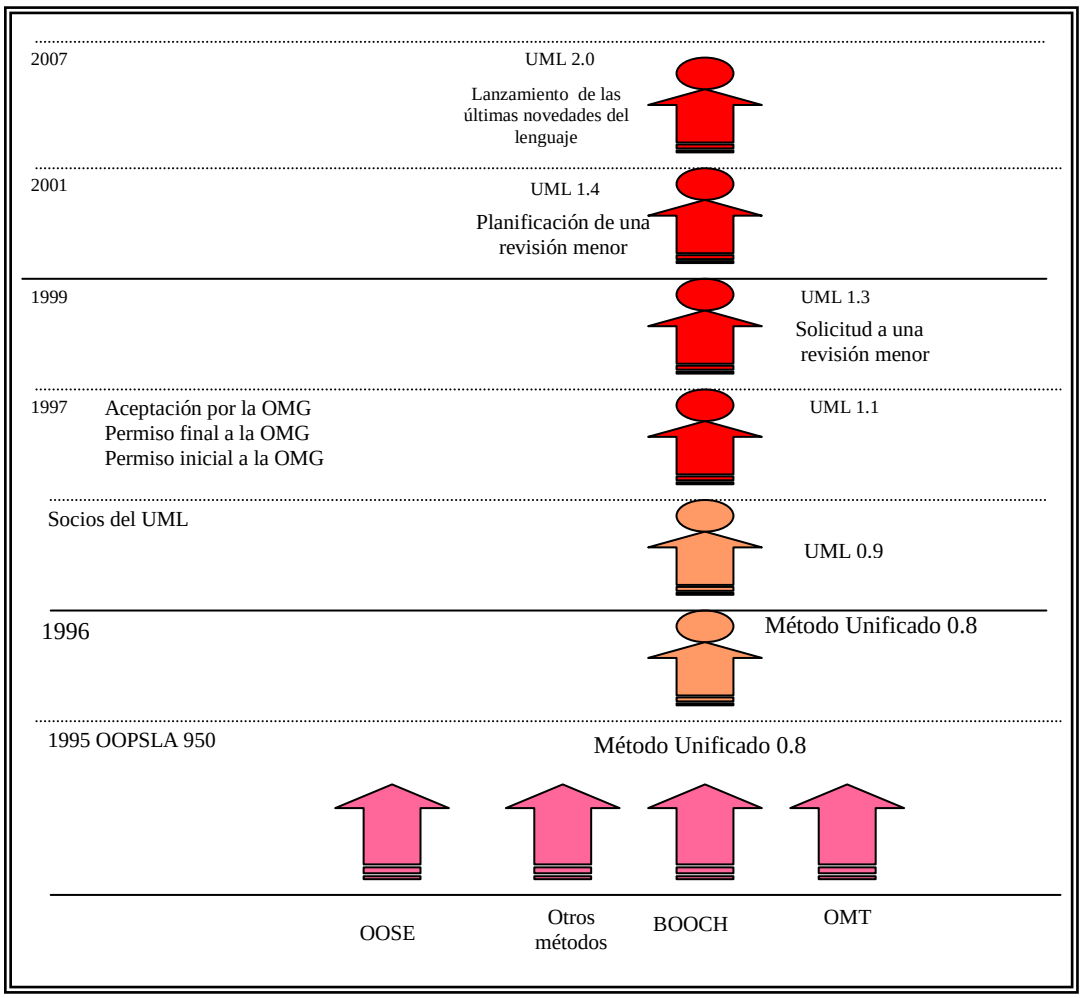


Figura N°05, Evolución del UML desde el advenimiento del Método Unificado 0.8.

2.4. UML AL DETALLE

Los elementos y diagramas UML están basados en el paradigma orientado a objetos.

Podemos dividir al UML en cuatro partes:

2.4.1. VISTAS

Muestran los diferentes aspectos del sistema que son modelados. Una vista no es un gráfico, pero es la abstracción consistente en un número de diagramas.

Consideramos las siguientes vistas:

- Vista de casos de uso.
- Vista lógica.
- Vista de componentes.
- Vista concurrente.
- Vista de despliegue.

Las vistas anteriores, son trabajados en el browser del case de modelamiento Rational Rose, ver figura N° 06.

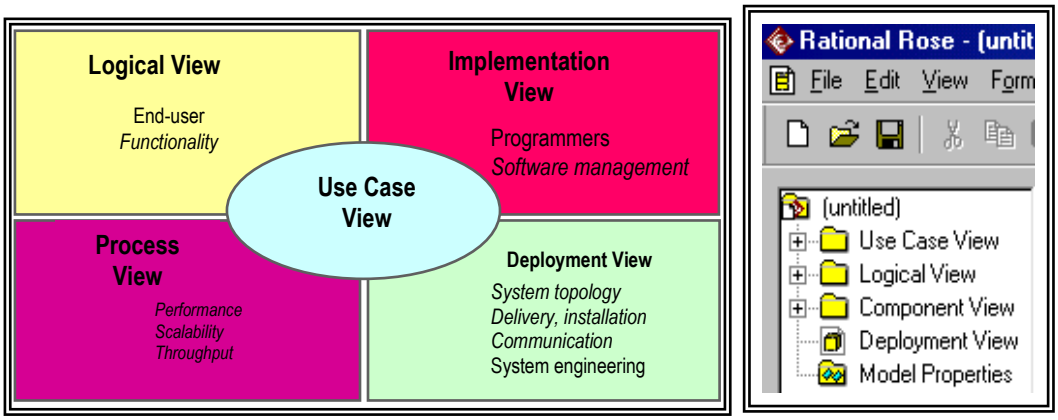


Figura N°06, Vista del UML, desde 2 puntos de vista.

2.4.2. DIAGRAMAS

Son los gráficos que describen el contenido en una vista. UML tiene ocho tipos de diagramas que se usan para proveer todas las vistas del sistema.

2.4.3. MECANISMOS GENERALES

Son símbolos genéricos para información adicional sobre un diagrama, típicamente son los que no pueden ser representados. Son considerados mecanismos generales a los adornos, las notas y las cajas de texto.

2.5. VISTAS DEL UML

2.5.1. VISTAS ESTÁTICAS

Los Diagramas de estructura estática de UML se van a utilizar para representar tanto Modelos Conceptuales como diagramas de clases de diseño, ambos usos son distintos conceptualmente, mientras los primeros modelan elementos del dominio los segundos presentan los elementos de la solución software.

2.5.2. VISTAS DINÁMICAS

Vamos a recordar los diferentes modelos que sirven para representar el aspecto dinámico de un sistema:

- Diagramas de secuencia
- Diagramas de colaboración
- Diagramas de estados
- Diagramas de casos de uso
- Diagramas de actividades

2.6. DESCRIPCION DE LOS DIAGRAMAS DEL UML

2.6.1. DIAGRAMA DE CLASES

2.6.2. CONCEPTO

Un diagrama de clases o estructura estática muestra el conjunto de clases y objetos importantes que forman parte de un sistema informático, junto con las relaciones existentes entre clases y objetos En este diagrama se muestras las clases de forma individual a con relaciones, las relaciones pueden ser: herencia, agregación, composición, asociación y asociación unidireccional.

2.6.3. ELEMENTOS DEL DIAGRAMA DE CLASES

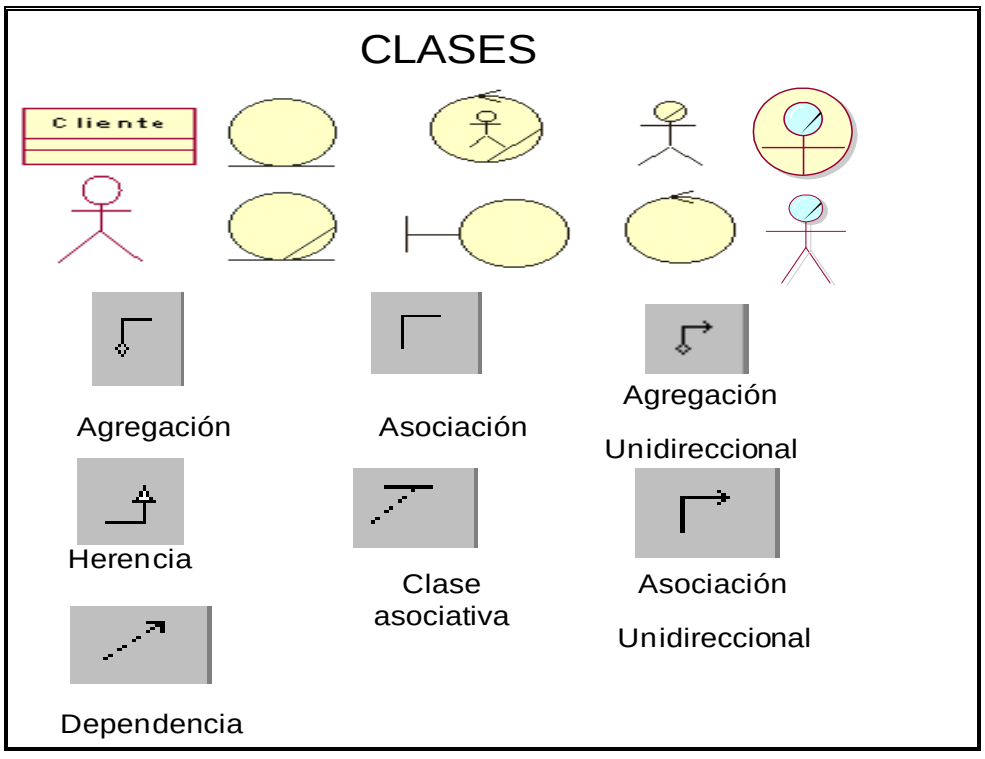


Figura N°07, Elementos en un diagrama de clases.

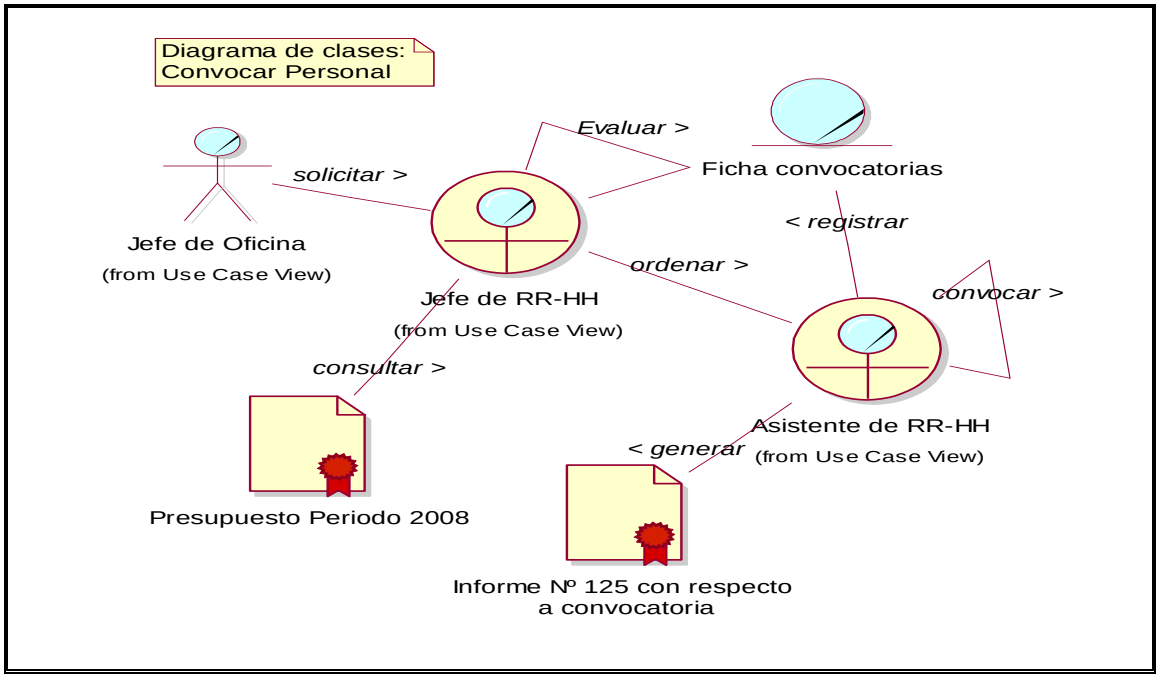


Figura N°08, Ejemplos de Diagramas de Clase.

2.7. DIAGRAMA DE PAQUETES

2.7.1. CONCEPTO

Diagrama del tipo estático, el objetivo es mostrar las dependencias que existen entre paquetes.

2.7.2. ELEMENTOS DEL DIAGRAMA DE PAQUETES

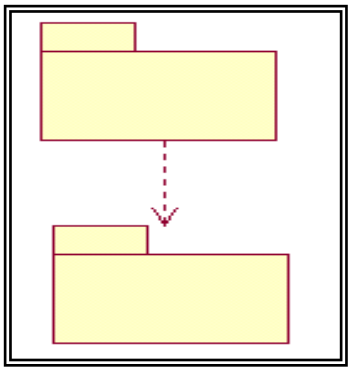


Figura N°09, Elementos y Diagrama de paquetes.

2.8. DIAGRAMA DE CASOS DE USO

2.8.1. CONCEPTO

El diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso tanto en el negocio como en el sistema. Muestran la atomización del sistema en fragmentos funcionales reutilizables, la interacción de los actores con la funcionalidad del sistema. Muestra la definición visual de los requerimientos del usuario. El diagrama de casos de uso también muestra el funcionamiento del proceso empresarial en términos de sus participantes los actores internos y externos ó con respecto a su realización en el ambiente de negocio.

2.8.2. ELEMENTOS DEL DIAGRAMA DE CASOS DE USO

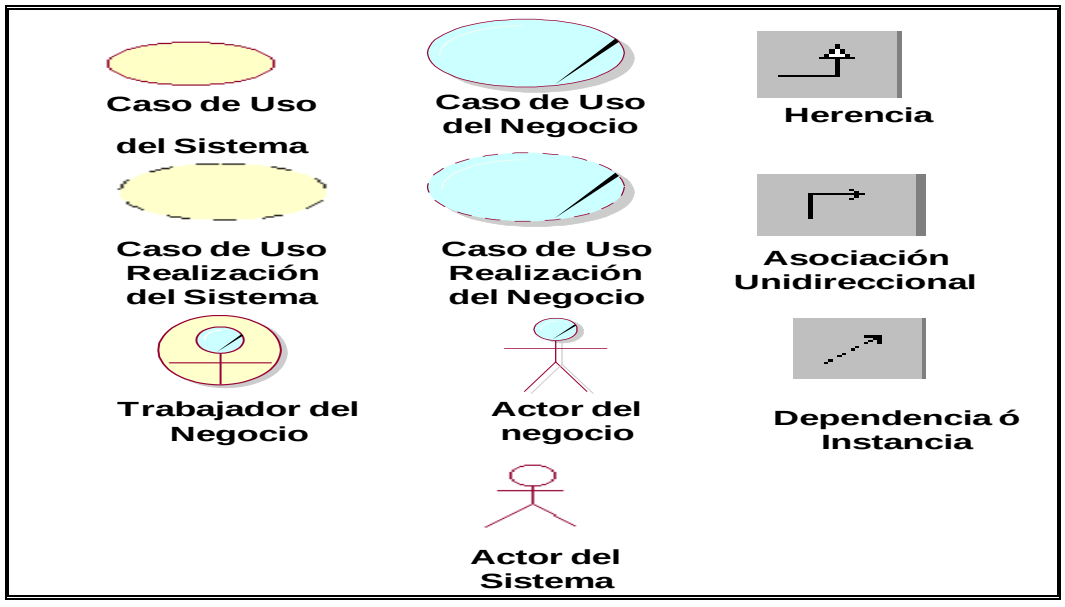


Figura N°10, Elementos a utilizar en un Diagrama de Casos de Uso.

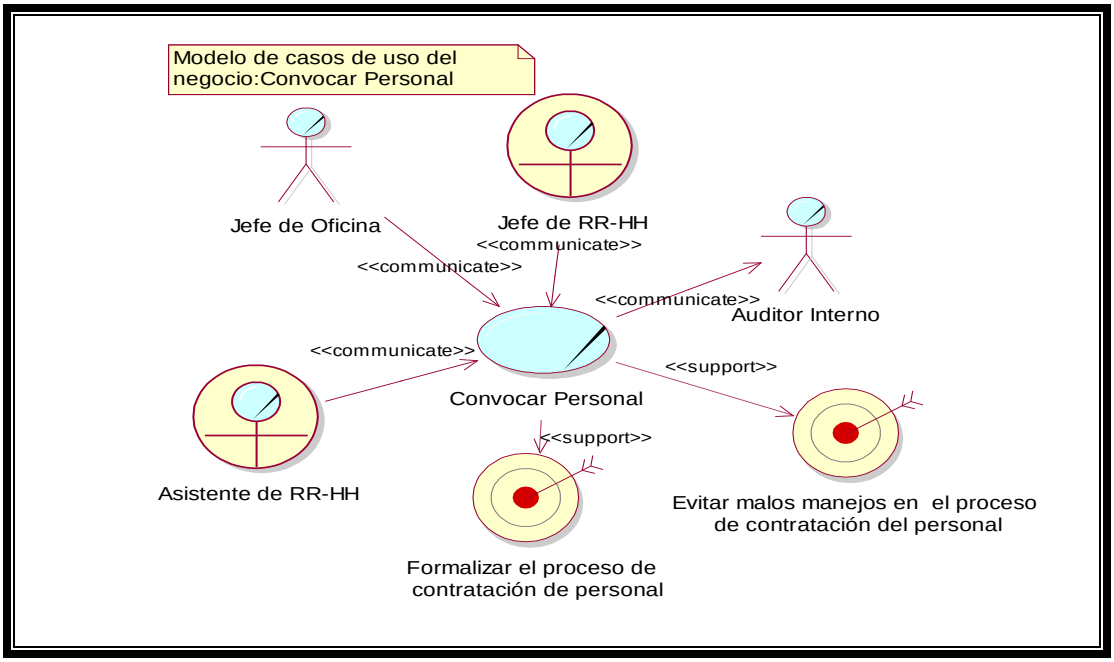


Figura N°11, Ejemplos de Diagramas de Casos de Uso.

2.9. DIAGRAMA ACTIVIDADES

2.9.1. CONCEPTO

Muestra las diversas actividades ejecutados por una persona, una organización, incluso el hardware ó el software.

Su objetivo es comprender qué actividades son necesarias y cuales son sus relaciones de dependencia ó transición de estado.

Se utiliza para representar los distintos escenarios que involucra un Caso de Uso, permite describir las tareas sincronizadas y responsabilidades, resolviendo factores de decisión.

2.9.2. ELEMENTOS DEL DIAGRAMA DE ACTIVIDADES

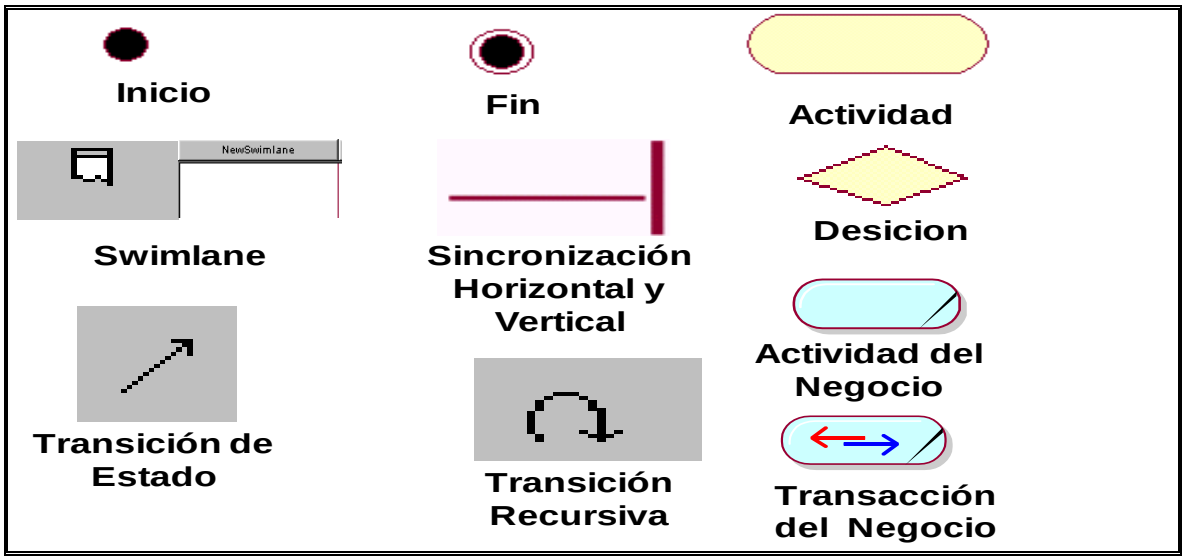


Figura N°12, Elementos del Diagrama de Actividades.

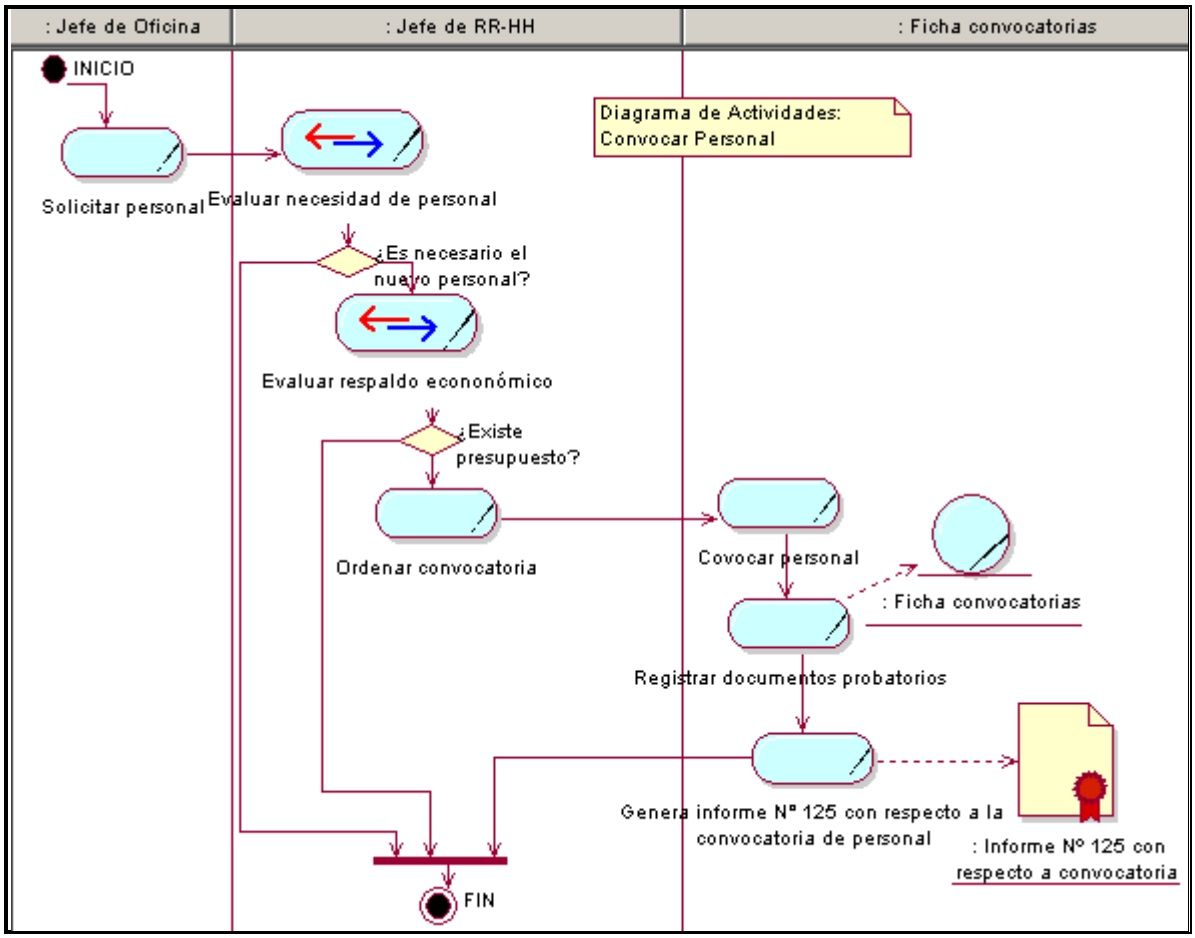


Figura N°13, Ejemplo de Diagrama de Actividades.

2.10. DIAGRAMA DE ESTADOS

2.10.1. CONCEPTO

Muestra la secuencia de estados por los que pasa el caso de uso, el objeto ó el sistema a lo largo de todo el tiempo de vida. En el diagrama de estados se indica qué eventos realizan los casos de uso, los objetos y los sistemas en general para pasar de un estado a otro y cuáles son las respuestas y acciones que genera. En cuanto a la representación, el diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con nombres de los eventos.

2.10.2. ELEMENTOS DEL DIAGRAMA DE ESTADOS

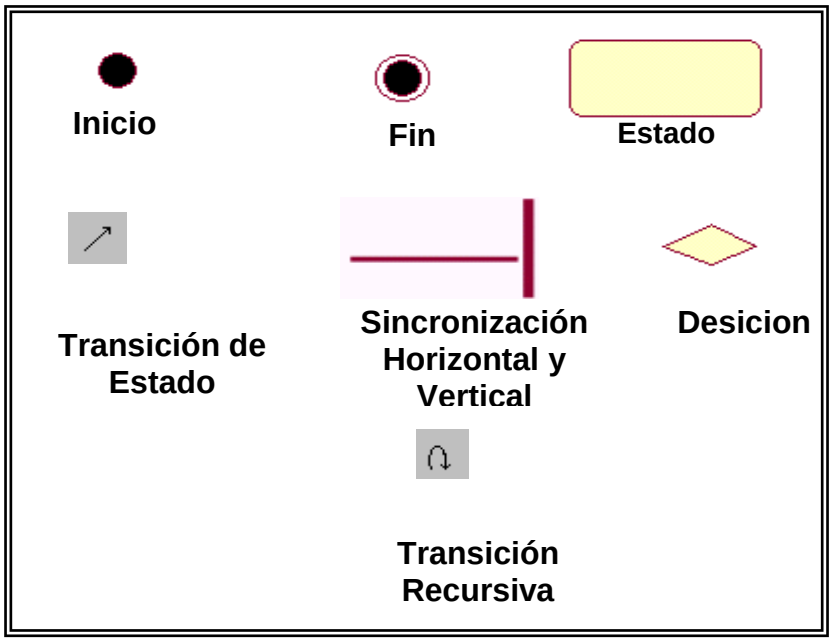


Figura N°14, Elementos del Diagrama de Estados

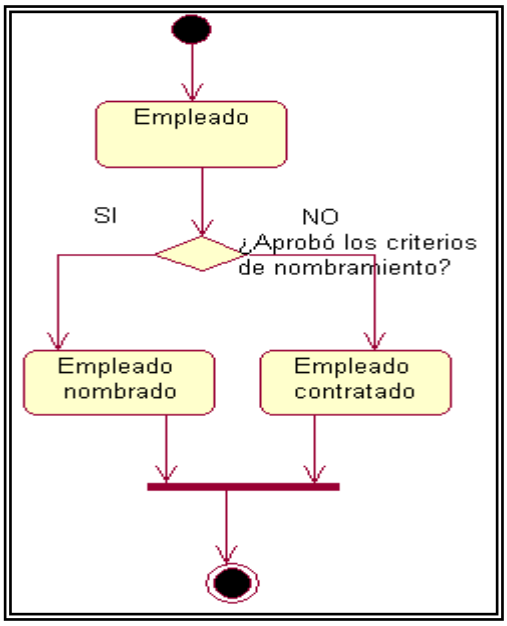


Figura N°15, Ejemplos del Diagrama de Estados.

2.11. DIAGRAMA DE OBJETOS

2.11.1. CONCEPTO

Es el entorno intrínseco de los diagramas del tipo interactivo, tanto en el diagrama de secuencia como en el diagrama de colaboración se realizan los diagramas de objetos, los elementos necesarios para realizar cualquiera de estos diagramas tienen la misma consecuencia.

La única diferencia entre ambos diagramas es la orientación, con respecto al tiempo en el diagrama de secuencia y con respecto al espacio en el diagrama de colaboración; incluso el case de modelado Rational Rose, toma como equivalentes a estos diagramas de interacción.

Convertir el diagrama de secuencia al diagrama de colaboración está a la altura de un "*clic*"; cabe resaltar que también funciona en el otro sentido.

2.12. DIAGRAMAS DE INTERACCIÓN

En los diagramas de interacción se muestra el patrón de interacción entre objetos. Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular. Son diagramas de interacción los ya mencionados diagramas de Secuencia y los diagramas de Colaboración.

2.12.1. DIAGRAMA DE SECUENCIA

2.12.1.1. CONCEPTO

El diagrama de Secuencia muestra la interacción ordenada según la secuencia temporal de eventos, con respecto al tiempo. Muestra los objetos participantes en la interacción y los mensajes que intercambian de manera ordenada y secuencial.

2.12.1.2. ELEMENTOS DEL DIAGRAMA DE SECUENCIA

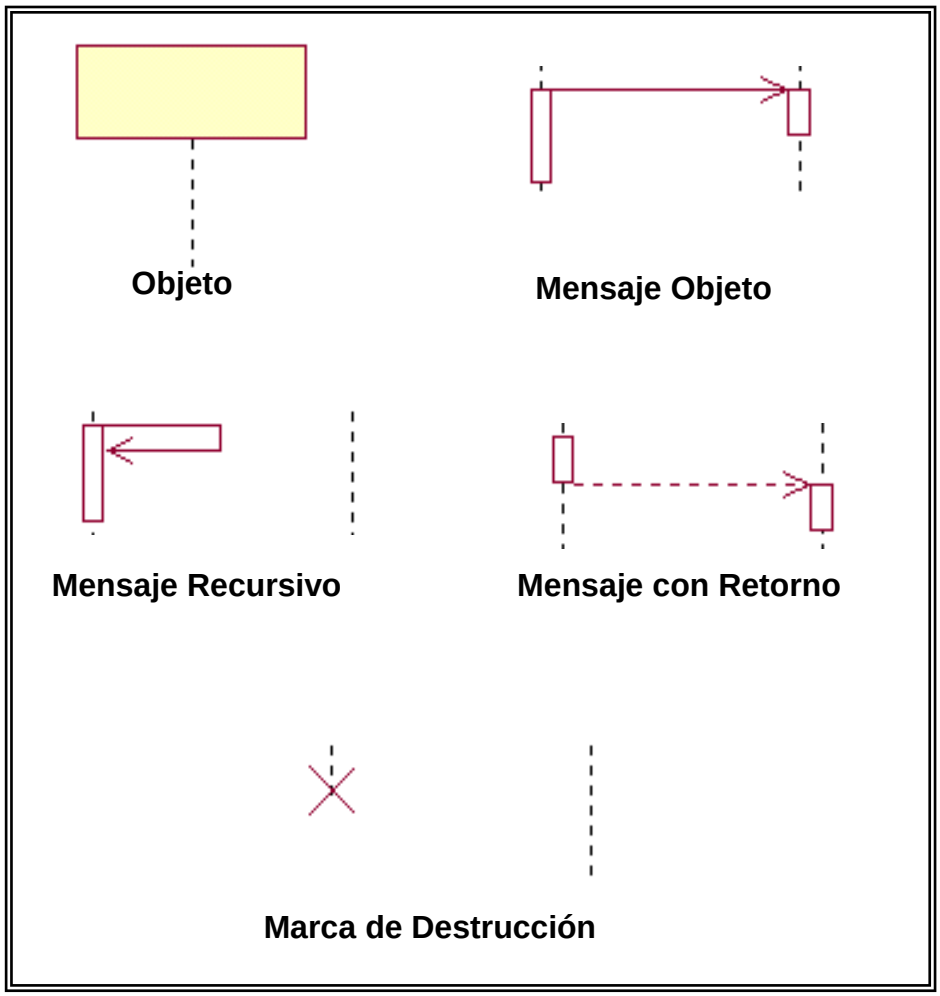


Figura N°16, Elementos del Diagrama de Secuencia.

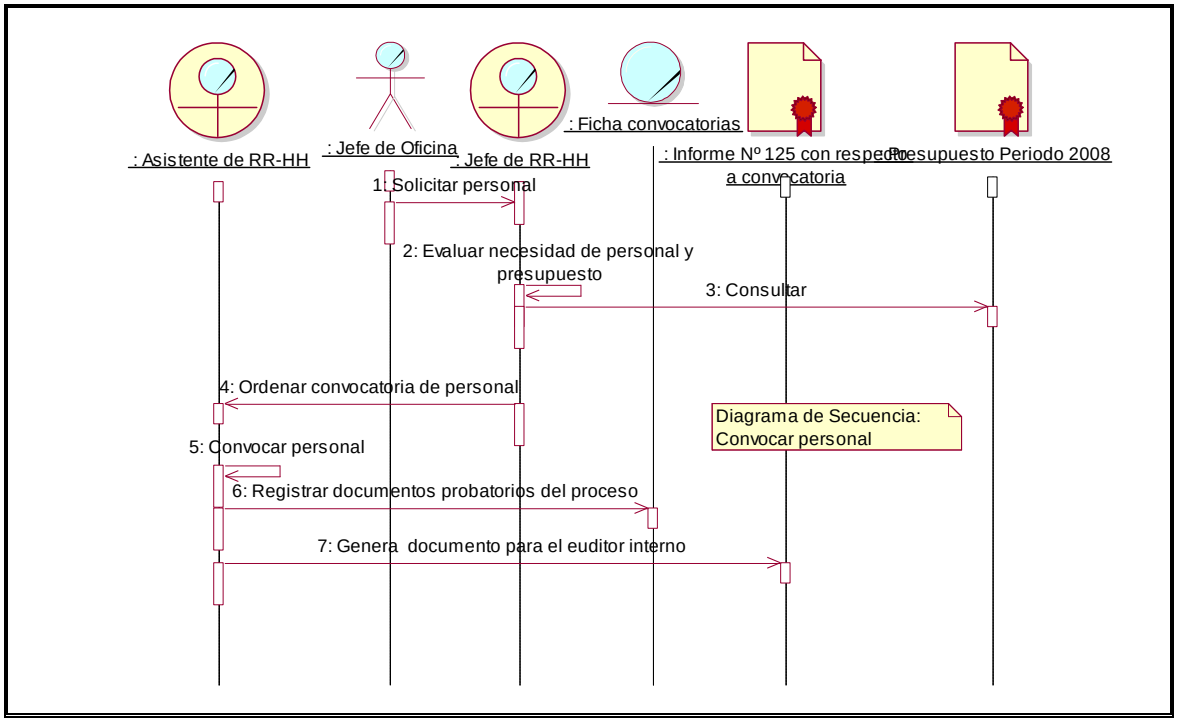


Figura N°17, Ejemplo del Diagrama de Secuencia.

2.12.2. DIAGRAMA DE COLABORACION

2.12.2.1. CONCEPTO

Es un diagrama del tipo dinámico, e interactivo, permite relacionar objetos con otros objetos y entre sí, a través de una secuencia de mensajes con respecto al espacio.

Este diagrama es una forma de representar interacción entre los objetos, es decir, las relaciones entre ellos y la secuencia de los mensajes de las iteraciones que están indicadas por un número.

Los diagramas de colaboración son idénticos a los diagramas de secuencia, la diferencia radica en la orientación.

2.12.2.2. ELEMENTOS DEL DIAGRAMA DE COLABORACION

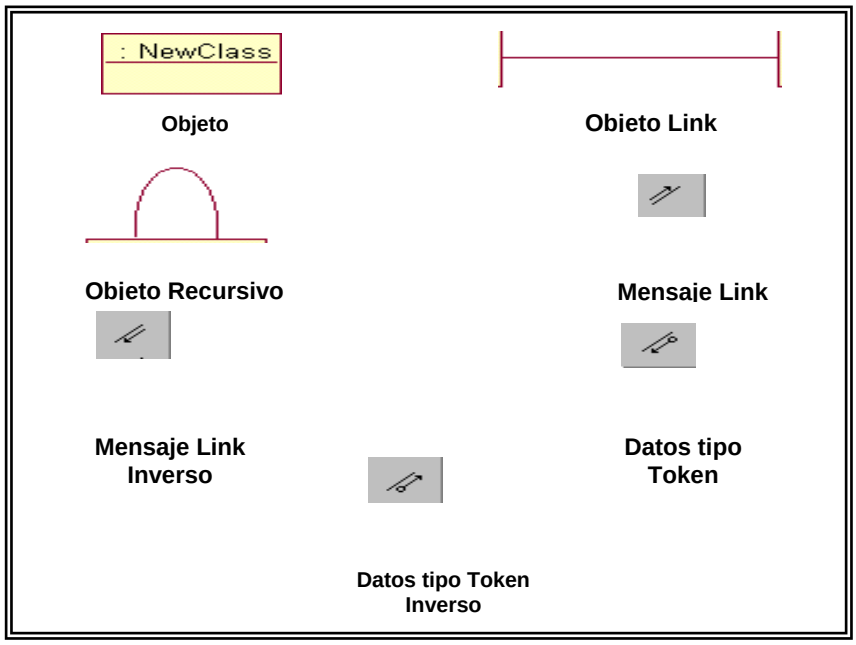


Figura N°18, Elementos del Diagrama de Colaboración.

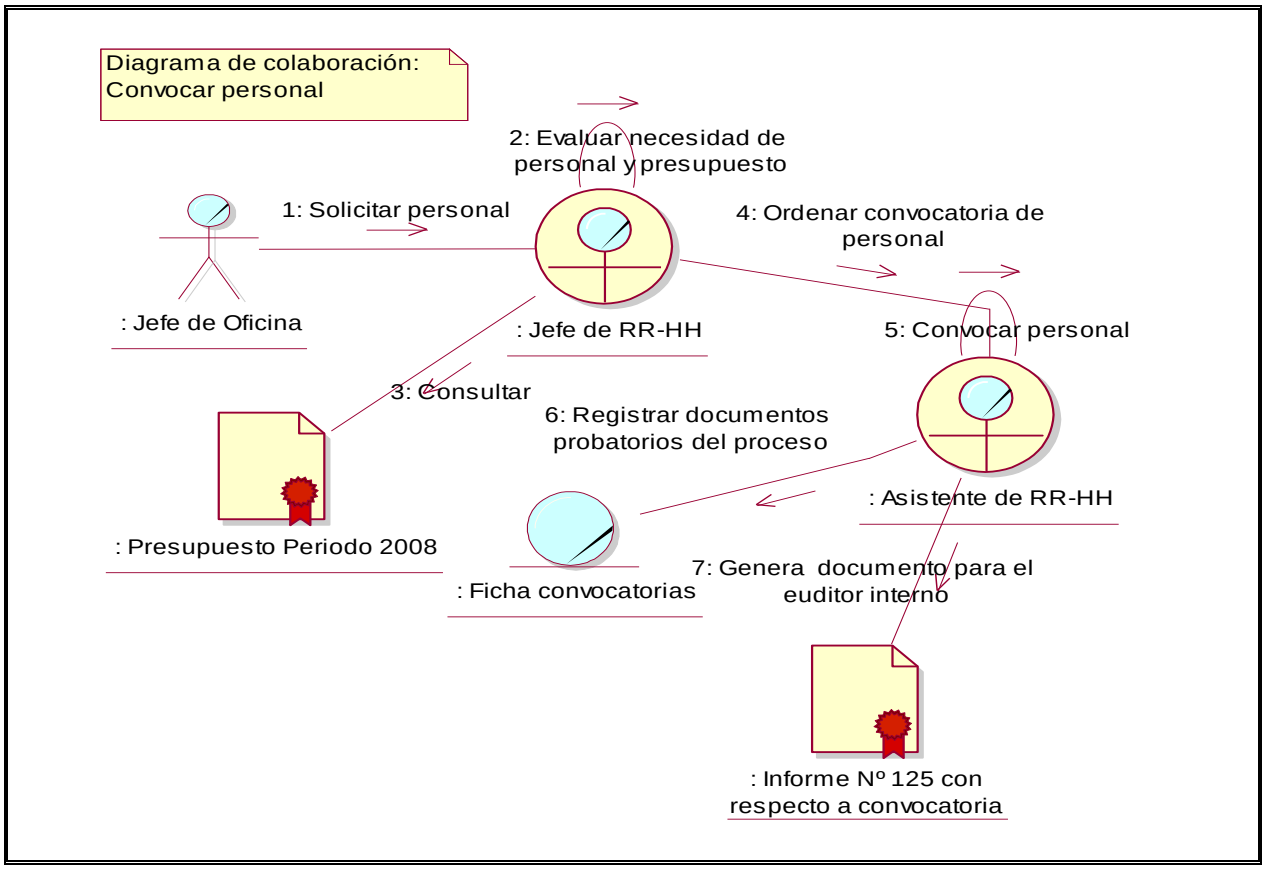


Figura N°19, Ejemplo del Diagrama de Colaboración.

2.13. DIAGRAMA COMPONENTES

2.13.1. CONCEPTO

Los componentes pertenecen al mundo físico, es decir, representan el bloque de construcción al modelar aspectos físicos del sistema informático. La característica básica del componente es que:
"debe definir la abstracción precisa con la interfaz bien definida, permitiendo reemplazar fácilmente los componentes viejos con otros nuevos y compatibles".

En el UML todos los elementos físicos se modelan como componentes.

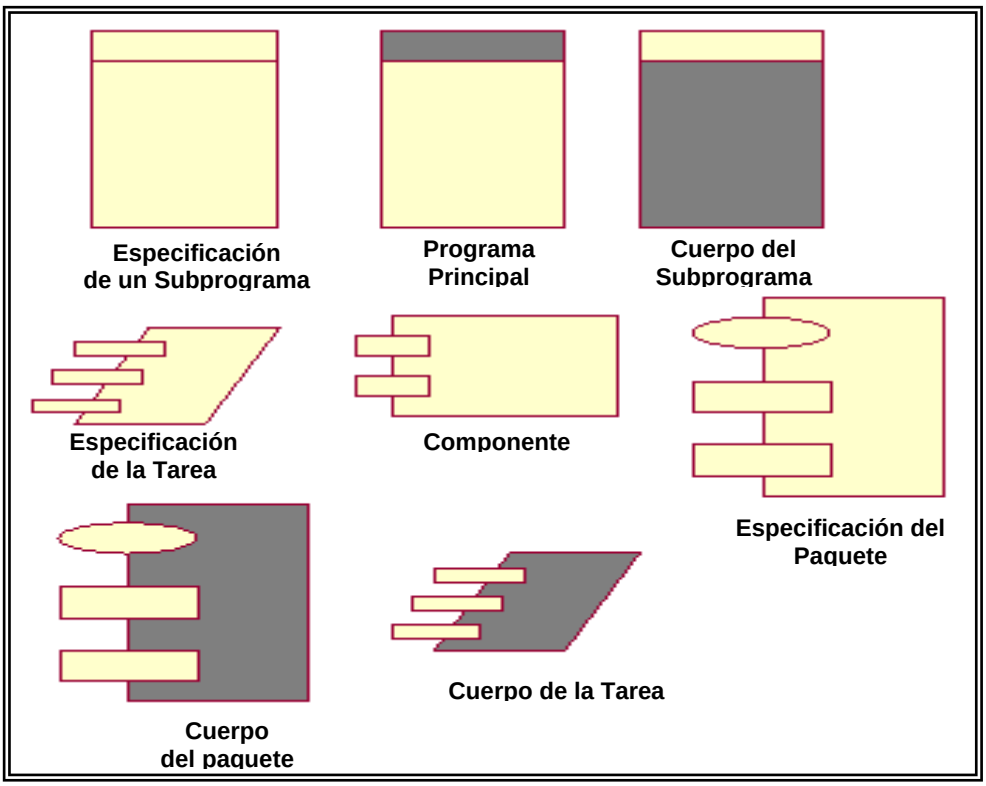


Figura N°20, Elementos del Diagrama de Componentes.

2.14. DIAGRAMA DE DESPLIEGUE

2.14.1. CONCEPTO

Al igual que los componentes los nodos pertenecen al mundo material. Vamos a definir el nodo como un elemento físico, que existe en tiempo de ejecución y representa el recurso computacional que generalmente tiene alguna memoria y, a menudo, capacidad de procesamiento.

Los nodos sirven para modelar la topología del hardware sobre el que se ejecuta el sistema. Un nodo representa normalmente el procesador o el dispositivo sobre el que se pueden desplegar los componentes.

Un nodo debe tener un nombre asignado que lo distinga del resto de nodos.

2.14.2. ELEMENTOS DEL DIAGRAMA DEL DESPLIEGUE

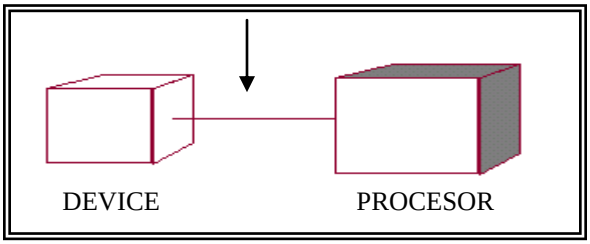


Figura N°21, Elementos del Diagrama del Despliegue.

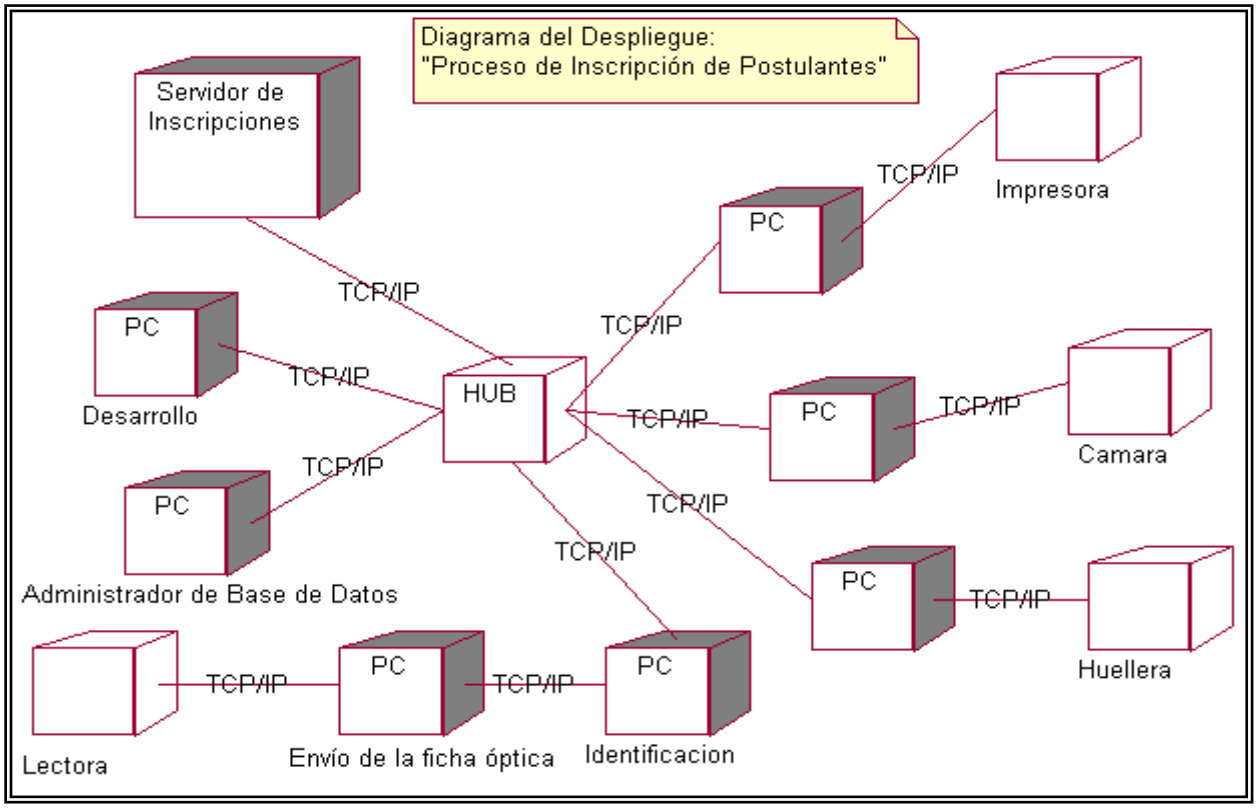


Figura N°22, Ejemplo del Diagrama del Despliegue.

2.13. RESUMEN

En este capítulo detallo las diversas vistas y diagramas del UML.

El UML no es una metodología es una técnica de modelado presente en todas las etapas de construcción de un sistema informático formal (basado en algún proceso, método o metodología).

El UML, sirve para que cada uno de los profesionales que participan en el proceso de construcción informática expresen sus resultados en términos de grafos, notaciones y diagramas.

2.14. PRUEBA DE COMPETENCIAS DEL CAPÍTULO

- ¿Considera usted que el UML es una metodología?, ¿Por qué?
- Realizar un cuadro comparativo entre los diversos diagramas del UML tratados en el capítulo.
- ¿Para qué sirve el factor de decisión en un diagrama de actividades?

CAPITULO III

PROCESO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Explicar la importancia del proceso de construcción de software Proceso Unificado.
- Detallar las diversas fases del proceso bajo la perspectiva de fases y modelos.
- Exponer los principios del proceso.
- Expresa las claves del proceso.

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector deberá diferenciar las características de las diversas fases del Proceso Unificado.
- El leedor describe los principios y claves del Proceso Unificado.
- El lector conoce y diferencia los artefactos utilizados en los diversos modelos propuestos por el Proceso Unificado.

3.1. INTRODUCCIÓN

Si no tenemos una guía ó prototipo que dirija los pasos para el logro de un objetivo en particular difícilmente lograremos el éxito, el mundo real NO funciona en base a criterios y procedimiento empíricos. Basarse en la suerte, en "*lo NO previsto*" y en "*el ojala suceda como pienso*", demuestran la falta de conocimiento e inseguridad, ello repercute en el fracaso del objetivo trazado.

Los profesionales dedicados a la construcción de software, sabemos que la combinación del conocimiento, experiencia, habilidad y creatividad en la aplicación de técnicas, métodos y procesos, nos acerca con certeza al éxito en la creación del software.

"Si no contamos con el plan que guíe el proceso de construcción de software", sin duda caeremos en el fracaso. Un proceso define quién está haciendo qué, cuándo y cómo para lograr el objetivo previsto. En la ingeniería de software el objetivo es construir el software ó mejorar alguno existente.

Para lograr el éxito del proyecto informático NO basta tener la buena administración del conjunto.

Después de un largo proceso de investigación y comparación puedo establecer con certeza, la importancia del un proceso que guíe la construcción del software, el binomio "*administración del proyecto"* y "*proceso de construcción del software"* permite acercarnos al éxito del software en términos de tiempo, costo, calidad y alcance.

Debemos tener cuidado al momento de seleccionar el proceso de construcción, se debe poner especial énfasis en el estudio de los procesos organizacionales y procurar el respaldado por alguna organización estándar.

El advenimiento del Internet, la globalización y el desarrollo agigantado de la tecnología hace que los usuarios soliciten software con características cada vez más sofisticados que les permitan estar a la altura de los constantes cambios internos como externos para permanecer en la carrera competitiva exigida por el mercado actual.

Es necesaria la aplicación del proceso que permita la centralización en los procesos empresariales, adelantarse a los riesgos, centrarse en la arquitectura de desarrollo, pasar por una estricta etapa de pruebas y control de calidad, permitir que cada uno de los integrantes del equipo actué y piense como un solo grupo y analizar el entorno organizacional para asegurar el éxito de la integración.

El Proceso Unificado, es un proceso basado en la metodología orientado a objetos y declarado como proceso estándar por la Object Management Group (OMG) es una alternativa para solucionar muchos de los problemas que aquejan constantemente en la construcción del software.

En el presente capítulo analizaremos los principales aspectos del Proceso Unificado, como fruto de más de un año de investigación; también abordaremos los principios, fases, elementos, desde un punto de vista práctico y didáctico.

3.2. CONCEPTO

El Proceso Unificado (PU) es el proceso de *ingeniería de software*, cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de los márgenes de la planificación y presupuestos establecidos.

El Proceso Unificado, cubre todo el ciclo de vida de desarrollo de software, el propósito es asegurar la producción de software, es decir, que colme las expectativas y exigencias del usuario actual, entregado en el tiempo previsto, con la calidad esperada, que se maneje dentro del presupuesto-costo calculado y que cumpla con los requisitos establecidos en la definición del proyecto de construcción del software.

El Proceso Unificado puede integrar todos los aspectos a tener en cuenta durante el ciclo de desarrollo del software con el objetivo de hacer tangibles todo tipo de proyectos sin interesar su envergadura.

3.3 ANTECEDENTES

Años atrás nuestros colegas especialistas en las construcción de software encontraban muchas dificultades en el proceso de construcción de software, problemas tales como: mantener el hilo conductor del proceso de desarrollo, mantener la retroalimentación constante entre cada una de las etapas de construcción, falta de conocimiento organizacional y falencias en la definición de roles, fueron algunas de las causas de la falta de calidad y performance en el software puesto en producción. Muchas de las dificultades expuestas son solucionadas por el Proceso Unificado.

El Proceso Unificado, nace a partir de la necesidad de contar con un proceso, robusto, potente y flexible que permita dar solución a los requerimientos cada vez más sofisticados del usuario actual donde el punto de entrada más importante es el conocimiento de la organización en base a procesos y sus participantes internos ó externos.

El Proceso Unificado fue creado por Grady Booch, Ivar Jacobson y James Rumbaugh se hace presente en el mercado de desarrollo de software a principios del 1998.

Los orígenes del Proceso Unificado se remonta desde 1967, fecha en que el *método Ericson* era el más respetable método de construcción de software, a partir del *modelo Ericson* el proceso PROCESO UNIFICADO tuvo varias influencias como el *Rational Approch* y el *Objectory Process,* entre otros.

Muchas empresas relacionadas con la tecnología y la informática patrocinaron la creación del Proceso Unificado, menciono algunos para alimentar vuestra cultura y evitar el silencio cuando alguna persona principiante en el apasionado mundo del Proceso Unificado, comienza a tener dudas.

Empresas patrocinadoras para la creación del Proceso Unificado:
IBM, Microsoft, Sun Microsystems, Rational Corporation, Microsoft, HP, Oracle, Texas Instruments, MCI, SystemHouse, entre otras.

3.4. IMPORTANCIA DEL PROCESO UNIFICADO

Resumo la importancia del Proceso Unificado en los siguientes puntos:

- Permite dar solución a los exigentes requerimientos de los usuarios actuales, cada vez más exigentes, debido a los constantes cambios que la misma sociedad y competencias en el mercado exigen.

- Permite obtener los requerimientos y organizarlos, documentar los requerimientos de funcionalidad y restricciones, documentar decisiones, captarlas y por último comunicar los requerimientos del negocio.

- Permite capturar varias de las *mejores prácticas* en el desarrollo moderno de software de forma que sea aplicable en un amplio rango de proyectos y organizaciones.

- Es una guía de cómo utilizar de manera efectiva el *UML*.

La técnica de modelado UML, no se utiliza únicamente para efectos de documentación, gracias al Proceso Unificado, el UML está presente en todas las fases y etapas establecidas por proceso unificado, con UML cada uno de los roles participantes en el proceso de desarrollo de software pueden expresar su trabajo en términos de diagramas.

Los analistas, ingenieros, arquitectos de software, revisores de casos de uso, etc, utilizan los diagramas para mostrar el detalle del proceso de construcción del software.

- Provee a cada miembro de equipo el fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo.

- Crea y mantiene *modelos,* en lugar de enfocarse en la producción de gran cantidad de papeles de documentación.

- Permite que todos los miembros del equipo compartan:

  Conocimiento base, el proceso, la visión de cómo desarrollar software y el lenguaje de modelado.

- Permite la verificación de la calidad del software, mediante las siguientes actividades:

  - ✓ Crea pruebas para cada escenario (casos de uso), asegurando que todos los requerimientos están apropiadamente implementados.

  - ✓ Verifica la calidad del software con respecto a los requerimientos basados en la confiabilidad, funcionalidad, desempeño de la aplicación y del sistema.

  - ✓ Prueba cada iteración.

- ✓ El proceso de Pruebas, sujeto también al modelo iterativo e incremental, permite que cada caso de uso que NO cumpla con el control de calidad pueda corregirse e implementarse en el momento indicado ya que la implementación de la solución obviamente buena, puede no ser la solución idónea si no es implementado en el momento justo.

> Si se desea construir software de calidad, en un tiempo corto, bajo el presupuesto establecido y cumpla con las especificaciones definida por el principal involucrado del proyecto, la alternativa, sin duda es el proceso Unificado.

3.5. PRINCIPIOS DEL PROCESO UNIFICADO

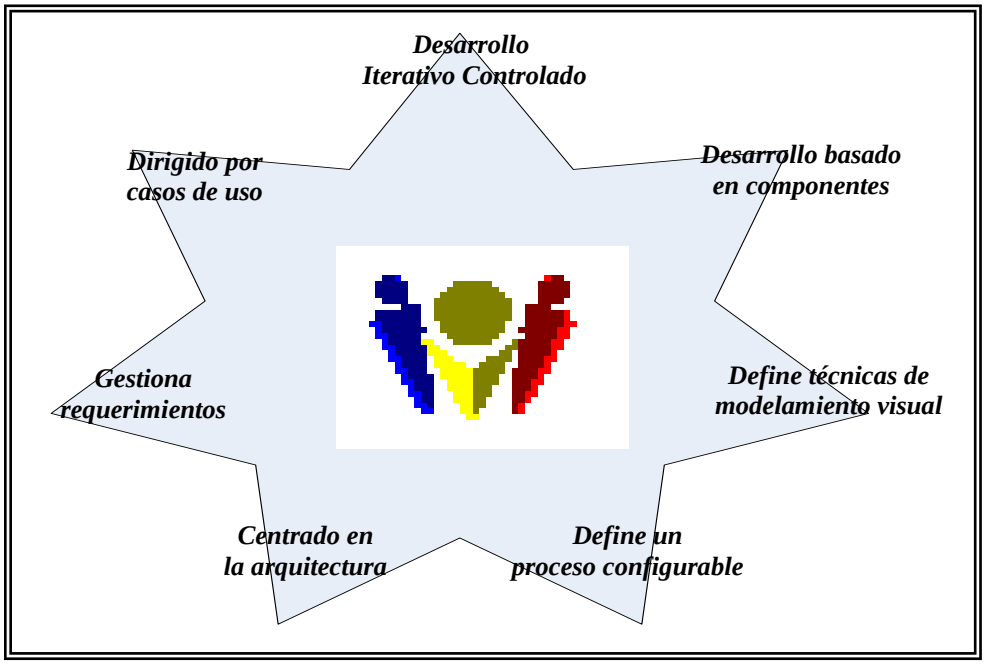


Figura N°23, Principios del Proceso Unificado

Después de analizar más de 22 principios citados por diferentes autores, detallaré 7 principios, los cuales brindan la importancia y diferenciación necesaria al modelo.

Los principios mencionados en la *figura Nº 23*, fueron pautas importantes que obtuve en la investigación y desarrollo en más de 13 proyectos de construcción de software con el Proceso Unificado. Constituyen el corazón del proceso, los cuales por razones que ya expondré son de real utilidad permitiendo el éxito del software, si se logra combinar de una manera inteligente y lógica el proceso de construcción de software con la administración del proyecto.
Ahora empezamos de describir los principios del PROCESO UNIFICADO, conducentes al éxito en la construcción de cualquier software.

3.5.1 GUIADO POR CASOS DE USO

La razón de ser del sistema de software es servir a los usuarios, ya sean humanos u otros sistemas. El ***caso de uso*** ó ***fragmento funcional del sistema***, es la facilidad que el software provee a los actores (personas, software ó hardware) que utilizan ó son utilizados por la plataforma de información del sistema.

Los casos de uso, son importantes ya que definen el comportamiento del futuro sistema, los casos de uso NO son parte de la orientación a objetos tradicional, pero entienden su funcionalidad, cada vez más clara y precisa a medida que se evoluciona; otros métodos orientados a objetos usan los casos de uso como guía pero con nombres diferentes.

Los casos de uso juegan un papel importante en cuatro etapas de los procesos generales del Proceso Unificado: Requisitos, Análisis-Diseño, Codificación y Prueba.

- El modelo de casos de uso es el resultado del análisis en la etapa *de "Requisitos ò Análisis de Requerimientos"*, en esta temprana etapa se necesitan a los casos de uso para conocer que hará el software desde el punto de vista del usuario, los casos de uso constituyen un concepto importante y fundamental, deben ser aceptados por el cliente y el Proceso Unificado desarrollador.

- En la segunda etapa *"Análisis-Diseño"*, los casos de uso son ejecutados en el modelo de diseño, se crea realizaciones de casos de uso, que describa como los casos de uso son realizados en términos de interacción de objetos en el modelo de diseño, este modelo describe en términos de diseño de objetos las diferentes partes de la implementación del sistema y cómo deben actuar ó funcionar las partes.

- En la tercera etapa *"Implementación"*, el modelo de diseño es la especificación de la implementación, porque los casos de uso realizados en el modelo de diseño, son implementados en términos de diseño de clases.

- Durante la cuarta etapa *"Pruebas"*, los casos del uso constituyen la base para identificar la prueba de casos de uso y la prueba de procedimientos, el sistema pasará el control de calidad y la etapa de pruebas solo si

todos los casos de uso especificados y desarrollados en etapas anteriores son parte funcional del sistema final.

- En el modelado de negocio se hallan los diversos procesos empresariales de la organización convirtiéndolas en casos de uso de negocio, posteriormente en casos de uso del sistema informático según el alcance del proyecto.

3.5.2. CENTRADO EN LA ARQUITECTURA

El Proceso Unificado, enfatiza la construcción de sistemas *software robusto*, respetando la arquitectura de construcción, ello disminuye el reinicio del software, aumentado la reutilización y facilita el mantenimiento futuro del mismo.

La arquitectura se utiliza para planificar y administrar el desarrollo del software teniendo en cuenta la reutilización de sus componentes.

La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas software, sistemas operativos, gestores de base de datos, protocolos de comunicación, etc.

El enfoque de iteraciones tempranas, definido con mayor énfasis en la fase de elaboración es producir y validar una arquitectura de software, que el ciclo de desarrollo inicial toma la forma de un prototipo arquitectónico ejecutable el cual evoluciona gradualmente para convertirse en un sistema final en las últimas iteraciones.

3.5.3. RESPETA EL MODELO ITERATIVO E INCREMENTAL

El Proceso Unificado es un proceso iterativo e incremental, el cual permite entender el problema a través de sucesivos refinamientos e incrementar la solución efectiva mediante múltiples interacciones, este acercamiento brinda la mejor opción en acomodar nuevos requerimientos ó cambio de tácticas en los objetivos del negocio y continuar con el proyecto identificado, resolviendo riesgos de manera oportuna.

El Proceso Unificado es un proceso controlado, la característica iterativa solo es posible al menos a través de una cuidadosa administración de requerimientos y control de cambios, asegurando así la comprensión de la funcionalidad del software en el momento adecuado considerando la calidad prevista, además permite el control de la entrega del proyecto dentro del tiempo establecido.

Para hacer más manejable un proyecto se recomienda dividirlo en ciclos, para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como mini-proyecto cuyo núcleo fundamental está constituido por una ó más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo.

La característica iterativa del proceso unificado, permite:

- El entendimiento incremental del problema a través de refinamientos sucesivos.

- Habilitar la fácil retroalimentación del usuario.

- Establecer metas específicas que permiten al equipo de desarrollo mantener su atención en producir resultados.

- La medición del progreso es conforme avanzan las implementaciones.

3.5.4. DIRECCIÓN O ADMINISTRACIÓN DE LOS REQUISITOS

Es el acercamiento sistemático a encontrar resultados, mientras se documenta, se organiza y se rastrea los requisitos de un sistema.

Formalmente es el establecimiento del acuerdo entre los clientes y los trabajadores del proyecto para administrar los cambios de requerimientos en el sistema. Los puntos clave en el manejo de requisitos son el mantenimiento de una visión clara de los requisitos en conjunto con los atributos aplicables y la proyección a otros requisitos y/o artefactos del proyecto.

- Los requerimientos no son fáciles de expresar claramente en palabras.
- La abundancia de requerimientos puede ser difícil de manejar por ende de controlar.
- Los requerimientos cambian con mucha frecuencia.
- Existen diversos tipos de requerimientos y diferentes niveles de detalle.

- Los requisitos no siempre son obvios y tienen diferentes fuentes.

Se debe tener en cuenta las siguientes habilidades para lograr el éxito aun con las dificultades que pueden presentar los requisitos:

- El análisis y entendimiento del problema.
- Comprender las necesidades de cada uno de los involucrados en el proyecto.
- Definir claramente el sistema en base a casos de uso.
- Definir claramente el alcance del proyecto.
- Refinar constantemente la definición del sistema.
- Realizar el seguimiento y control a los requisitos cambiantes.

3.5.5. PROCESO CONFIGURABLE

El Proceso Unificado (PU), es bastante general y completo, puede ser usado en muchas organizaciones de software (organizaciones basadas en proyectos[1] y matriciales balanceadas[2]). En muchas circunstancias, este proceso de ingeniería de software necesitará ser modificado, ajustado, extendido y entallado para acomodarse a las características específicas, circunstancias, entorno cultural, organizacional y político de la organización que lo adopta.

[1] Organización Basada en proyectos: Organización con labores centradas en proyectos.
[2] Organización Matricial Balanceada: Organización con labores funcionales y con proyectos, es una mixtura.

Los elementos del proceso de ingeniería de software que probablemente serán modificados, personalizados, agregados o suprimidos son los siguientes:

- Artefactos
- Actividades
- Flujos de trabajo
- Obreros

3.5.6. TECNICAS DE MODELAMIENTO VISUAL

El Proceso Unificado, al utilizar diferentes técnicas de modelado visual, permite:

- La captura de la estructura, comportamiento de arquitecturas y componentes.
- Mostrar como encajan de forma conjunta los elementos del sistema.
- Mantener la consistencia entre un diseño y su implementación.
- Promover la comunicación no ambigua.

3.5.7. DESARROLLO BASADO EN COMPONENTES

Gracias a la propiedad de herencia, adoptado de la Metodología Orientada a Objetos, el Proceso Unificado, permite el desarrollo de software basado en componentes, el cual brinda ventajas importantes como:

- Permite enfocarse en el pronto desarrollo de una arquitectura ejecutable robusta.

- Es intuitivamente comprensible.
- Promueve la reutilización más efectiva de software.
- Permite la construcción rápida de software. Si tienes gran cantidad de componentes reutilizables se puede finalizar el proyecto informático en un tiempo realmente corto.
- Es derivada a partir de los casos de uso más importantes.

3.6. ESTRUCTURA DEL PROCESO UNIFICADO:

El Proceso Unificado, se divide en dos dimensiones, a razón de dos ejes, ver figura N° 24.

- El eje horizontal representa al tiempo, detalla el aspecto dinámico del proceso, expresado en términos de ciclos, fases, iteraciones y metas.
- El eje vertical representa la característica estática del proceso, detalla como está descrito en términos de *actividades*, *artefactos*, *trabajadores* y *flujos de trabajo*.

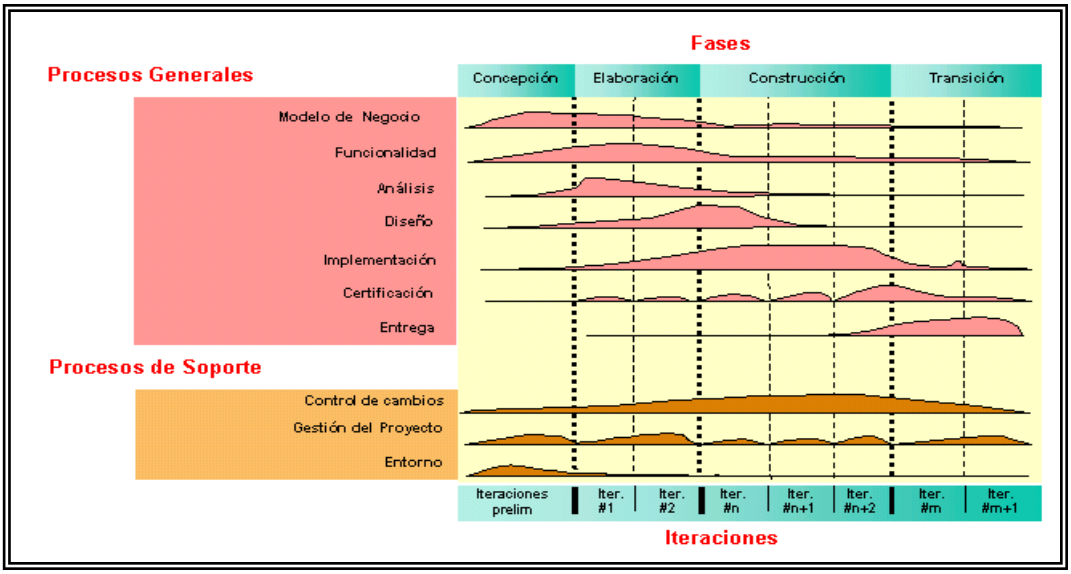


Figura N° 24, Fases y Etapas del Proceso Unificado.

3.7. FASES

3.7.1. GESTACIÓN Ó INCEPCIÓN

Esta fase permite el establecimiento de los objetivos y el plan del proyecto al definir su alcance.

El propósito es establecer los casos de uso de negocio para el nuevo sistema o para alguna actualización importante del sistema existente.

En esta etapa se establece la visión general de los requerimientos del proyecto y de los requerimientos principales para la construcción del software, definiendo el modelo inicial de casos de uso y el modelo del dominio.

Se realiza la evaluación inicial de riesgos y la estimación de los recursos requeridos, para minimizar los riesgos ó evitar que los riesgos se conviertan en problemas.

Esta etapa es ideal para definir el glosario de conceptos, políticas de creación y nombramiento de artefactos.

Esta etapa, permite entender el objetivo del proyecto, determina los casos de uso del negocio y se basa en los requerimientos de los stakeholders[3]

[3] Satkeholder: Representa a los interesados en el éxito o fracaso del proyecto

3.7.2. PREPARACIÓN Ó ELABORACIÓN

Permite la definición de la arquitectura, desarrollo del plan del proyecto y la especificación de características del sistema.

Esta etapa permite definir la *funcionalidad del software* a construir, al clasificar y priorizar los casos de uso del sistema.

En esta fase empezamos a desarrollar la programación lógica expresados en diagramas (realizados en los modelos de análisis), se realiza el análisis & diseño de la base de datos, pasando por el modelo conceptual, el modelo lógico y el modelo físico de la base de datos.

Se realizan las pruebas de certificación y control de calidad hasta llegar al final de la iteración "n", todas las etapas mencionadas se pueden realizar en paralelo, regresando a la etapa anterior ó proyectándose a la etapa siguiente, gracias al modelo iterativo-incremental en el que se basa el proceso.

3.7.3. CONSTRUCCIÓN

En esta fase se desarrolla el código final, se construye el producto final.

El propósito de esta fase es desarrollar incrementalmente el producto software completo el cual estará listo para ser transferido al usuario:

Se producen los siguientes artefactos:

- Después de realizar el proceso de ingeniería reversa, se realiza el modelo completo de diseño y casos de uso, producto del código de la implementación
- Liberaciones de productos ejecutables de funcionalidad incremental (versiones, prototipos)
- Documentación técnica del sistema
- Manuales de usuario
- Se obtiene la versión "beta" del producto

3.7.4. TRANSICIÓN

Se realiza la transición del producto en el entorno usuario. Esta fase, está dedicada a establecer los lineamientos de performance, logrando así cubrir las expectativas de usuario.

Se obtienen los siguientes artefactos:

- Producción de ejecutables de producto
- Modelo de componentes al realizar la compilación y despliegue de componentes en base a la *ingeniería reversa*
- Modelo de diseño actualizado en base a la Ingeniería reversa
- "Pruebas beta" del software para validar el nuevo sistema versus las expectativas del usuario
- Manuales de usuario actualizados
- Documentación de desarrollo actualizada

- Se realiza el proceso de retroalimentación desde el punto de vista del usuario referente a la reciente implementación. Se contesta a las siguientes preguntas ¿el usuario está satisfecho?, ¿los gastos reales de los recursos versus gastos previstos son aceptables?

Uno de los puntos de importancia en esta fase es el desarrollar el plan de soporte y mantenimiento para el sistema de información que acaba de ser puesto en producción, se define cada cuanto tiempo se realizará el mantenimiento, características del equipo de mantenimiento, procesos de supervisión, políticas a seguir en el proceso de copia de seguridad y las políticas de registro a los nuevos usuarios.

3.8. ETAPAS

3.8.1. DISCIPLINAS CENTRALES

3.8.1.1. MODELO DE NEGOCIO

Analiza la organización en términos de procesos y las personas u organizaciones que influencian ó participan en él, de forma directa ó indirecta. El modelo de negocio sirve para determinar cual es el problema de la organización.

Presenta dos etapas:

- ***MODELO DE CASOS DE USO DE NEGOCIO***

Este modelo muestra la relación existente entre un actor de negocio externo ó interno y al caso de uso de negocio, desde el punto de vista general.

- ***MODELO DE ANÁLISIS DEL NEGOCIO***

En esta etapa se define el detalle del negocio en términos de procesos empresariales; el caso de uso de negocio definido en la etapa anterior pasa por el proceso de realización, donde utilizando diferentes vistas del UML como: diagrama de casos de uso (realización del caso de uso), diagrama de clase, diagramas de secuencia, diagramas de colaboración e incluso el diagrama de actividades se llega al detalle de funcionalidad del proceso empresarial que se analiza.

3.8.1.2. FUNCIONALIDAD

En esta etapa se define ¿qué hace el sistema?, para responder la interrogante anterior se realiza diversos procesos para el análisis de los requerimientos, logrando definir cuales serán las opciones del menú principal del sistema incluyendo cada uno de las sub opciones incluso definir las interfaces del sistema final.

3.8.1.3. ANÁLISIS

Esta etapa está dirigida al análisis de la información obtenida en el negocio, después de haber definido la funcionalidad del software que se construye en la etapa anterior, es necesario definir como se realizará la implementación en base a todos los requerimientos establecidos. El Proceso Unificado, propone los denominados, clasificadores de análisis, para realizar la programación lógica; este muestra el detalle de cómo se realizará los procesos de funcionalidad del software final.

Ejemplo: ¿cómo se inserta un registro a una tabla independiente?

Para el desarrollo de la interrogante anterior es necesario conocer las siguientes entradas:

- En qué parte del menú principal se encuentra la opción de inserción del nuevo registro.
- Cuántas y cuáles son las interfaces que participan en el proceso de inserción del registro
- Cuál es la tabla de la base de datos, obviamente, debemos definir la estructura de la futura tabla, que aún no existe.
- Por último se propone un modelo utilizando los clasificadores de análisis y artefactos del UML mostrando la relación lógica de participación e interacción entre cada uno de los artefactos que participan para lograr insertar el registro en una tabla independiente.

3.8.1.4. DISEÑO

Esta etapa causa duda y controversia, entre muchos autores de libros ó artículos en la web, en más del 60% del material bibliográfico investigado para la presente edición, existe dudas con respecto a esta etapa, para muchos autores esta es la etapa de implementación lógica del software, hay algunos que pretenden hacer una comparación con la metodología estructurada y su típico modelo entidad / relación (E/R) para la construcción de la base de datos. Es lamentable que las justificaciones sean sólo teóricas.

La programación lógica ya fue definida en la etapa de análisis, en la etapa de diseño nos dedicamos a la construcción de la base de datos relacional / objeto,
Es importante mencionar que NO existe comunión entre el modelo **E/R** y el modelo de **Objetos**. Es imposible compararlos ya que tienen puntos de partida y consecuencias diferentes.

En diseño tenemos que tomar en cuenta la información proveniente de etapas anteriores, se inicia una de las actividades más importantes en el proceso de construcción de software, "el análisis y diseño de la base de datos", y la primera propuesta del modelo de diseño del futuro software.

Realizaremos el modelo orientado a objetos, definido por etapas iterativas de desarrollo desde el modelo conceptual, pasando por el modelo lógico hasta llegar al modelo físico que es el modelo de base de datos relacional – objeto propiamente dicho.

El objetivo del modelo orientado a objetos es la construcción de una base de datos relacional – objeto que cumpla con todos los criterios de performance y calidad.

Los criterios de calidad de la base son los siguientes: la base de datos No debe permitir nulos, ni redundancia logrando la homogeneidad, sin pasar por el tedioso método de normalización, el cual es un término no existente en el Proceso Unificado.

3.8.1.5. IMPLEMENTACIÓN

En esta etapa se administra la generación de archivos, empezamos la codificación para producir el software final, se debe tener en cuenta los artefactos producidos en las anteriores etapas, sobre todo el modelo de la base de datos relacional – objeto (clases con el estereotipo tabla *relacional – objeto*, estructura, relaciones entre clases, las llaves primarias y los campos foráneos) y el *modelo de análisis* (programación lógica en términos de diagramas).

Ya terminada la etapa de implementación, realizamos el proceso de *ingeniería reversa* para actualizar para primera propuesta del diseño, este proceso es útil para garantizar el cumplimiento de todos los requerimientos del usuario.

3.8.1.6. CERTIFICACIÓN

Esta etapa analizamos los prototipos, por cada caso de uso. Gracias a la característica del modelo iterativo e incremental del PROCESO UNIFICADO, se construyen diversos prototipos, los cuales deberán pasar por un estricto control de calidad, definiendo con certeza cuales son los prototipos que cumplen con los requerimientos del usuario definido en etapas anteriores.

Existen muchos métodos de prueba de calidad de software, recomiendo el proceso de *ingeniería reversa.* Con este proceso probamos que la primera propuesta del diseño realizado en base al modelo de análisis es consecuente con la segunda propuesta del diseño, obtenido como producto del proceso de ingeniería reversa.

3.8.1.7. ENTREGA

Se gestiona el proceso de puesta en marcha del software, este debe estar preparado para la producción siendo flexible para facilitar el proceso de integración con otros sistemas de la organización.

Es requisito indispensable que los sistemas de información hayan aprobado todos los lineamientos de calidad, establecidos en la etapa de ***"pruebas"***, es suficiente la presencia de un error en el sistema de información para No entregar el software resultado.

3.8.2. DISCIPLINAS DE SOPORTE

3.8.2.1. CONTROL DE CAMBIOS

Establecimiento de políticas de gestión para la administración de cambios en el proyecto de construcción del software.

Los cambios generalmente vienen de los principales involucrados del proyecto "los clientes", esos cambios son clasificados en 2 categorías:

- **Cambios relevantes**, aquellos que tienes repercusiones serias en el desarrollo del proyecto, incluso se puede modificar la estructura de la base de datos y la propuesta de interfaces.

- **Cambios irrelevantes**, aquellas que pueden ser solucionados sin mayor dificultad, este tipo de cambios no repercute en modificaciones mayores tanto en el ámbito de aplicación como en el ámbito de la base de datos.

Al inicio del proyecto se establecen las políticas de control de cambio, quién es la persona ó personas integrantes del grupo de desarrollo quienes reciben los pedidos de cambio; se resuelven las siguientes interrogantes ¿Cuáles son los formatos de recepción?, ¿Cuáles son los medios de recepción del cambio?, ¿Cuáles son las restricciones y supuestos, con respecto al manejo del cambio?

3.8.2.2. GESTIÓN DE PROYECTOS

El éxito de la construcción del software no solo depende del proceso, es necesario que el binomio administración del proyecto y proceso de construcción, funcionen de forma mancomunada, sólo así se logra el éxito en la construcción de software.

La gerencia de proyectos provee el marco que permite cumplir con los objetivos de la organización usando un proceso estructurado y controlado. Comprende varias técnicas, herramientas y metodología que permiten al gerente y su equipo llevar a cabo un proyecto que cumpla con el "principio del cuarto cuadrante[4]".

[4] Principio del Cuarto Cuadrante: Este principio indica los 4 factores de éxito para un proyecto: el *TIEMPO*, el *COSTO*, la *CALIDAD* y el *ALCANCE*.

El proyecto deberá cumplir:

- Con terminar en el tiempo pactado.
- Dentro de los límites de presupuesto.
- Con la calidad esperada por el cliente.
- Con el alcance establecido en la definición de proyecto.

El rol del gerente de proyectos es de gran responsabilidad, siendo el encargado de dirigir y supervisar el proyecto de principio a fin.

Algunas de sus principales tareas serán:

- **Definir el proyecto**: debe definir el alcance del proyecto, estableciendo sus límites, en otras palabras, se aclara que procesos, departamentos ó elementos de la organización forma parte del proyecto.

  Esto es fundamental para prevenir un crecimiento indeseado del proyecto, a medida que se progresa. Es importante diferenciar claramente aquellos elementos y resultados que son absolutamente necesarios, de aquellos que son deseables.

- **Planificar el proyecto**: planificar el proyecto implica proponer la solución a desarrollar, en base a los objetivos y resultados necesarios, y establecer cómo la desarrollará. Los puntos mas importantes a considerar son: estrategia (cómo se relaciona el proyecto con el plan estratégico de la empresa), recursos (que necesito y con qué cuento), finanzas (cuanto costará y dónde

obtener el dinero) y tiempo (de cuánto tiempo se dispone).

- **Obtener el respaldo de la alta gerencia:** para el éxito de cualquier proyecto, es fundamental el apoyo irrestricto de uno o mas gerentes de alto nivel. Esto hará mucho mas fluido todo el proceso, incluyendo la obtención de recursos, lograr la colaboración de toda la empresa y la resolución de conflictos entre departamentos si es posible.

- **Formar el equipo humano:** Identificar y ubicar a aquellas personas mejor calificadas para las distintas tareas involucradas. Con frecuencia, el equipo se forma con personas provenientes de distintas áreas de la organización, por lo que no reportan directamente al gerente del proyecto. En ocasiones, es necesario reforzar el equipo con personas de fuera del entorno de trabajo, en cuyo caso hay que hacer el reclutamiento.

- **Obtener los recursos:** Es responsabilidad del gerente de proyectos asegurar los recursos (dinero, equipos, personal de apoyo, espacio físico, etc.) que le permita al equipo funcionar en forma efectiva.

- **Definir las operaciones:** Incluye determinar las herramientas a utilizar (ej. software de manejo de proyectos), definir los canales de comunicación, establecer la logística, etc.

- **Controlar el proyecto:** Asegurar que las metas se están logrando y que el proyecto sigue el curso planificado.

En el transcurso del desarrollo del proyecto, surgirán cambios e imprevistos, en cuya circunstancia, es labor del gerente mantener la flexibilidad que le permita adaptarse, corregir y/o ajustar, sin poner en peligro los resultados.

Para evitar problemas de fracaso del proyecto, los gestores del proyecto deben prestar especial atención a las principales razones de fracaso:

- Mala definición o concepción del proyecto
- Cambios en el alcance o definición del proyecto
- Falta de una metodología adecuada para la administración del proyecto
- Falta de planificación en el control de los cambios
- Falta de comunicación entre los miembros del equipo entre ellos y el resto de la empresa
- Falta de claridad del contrato en términos de supuestos y restricciones
- Desacuerdos entre clientes y los gerentes de proyecto

3.8.2.3. ENTORNO

El análisis del entorno, establece criterios ó políticas que permitan el proceso de puesta en marcha del software construido. El nuevo software debe funcionar adecuadamente con los otros sistemas de información de la organización facilitando así el proceso de integración de los sistemas.

El sistema de información necesita cumplir con la factibilidad técnica[5] y operativa[6], para lograr el éxito en la organización.

3.9 ESCENCIA DEL PROCESO UNIFICADO

- El Proceso Unificado se adelanta a los riesgos y continúa con una política de anticipación antes que los riesgos mellen el proceso de construcción del software.
- Permite centrarse en un software ejecutable
- Permite establecer los cambios tempranamente
- Permite construir los sistemas informáticos en base a componentes
- Establece la calidad como un estilo de vida en el proceso de construcción del software, no como una reacción ante problemas o requerimientos.

3.10. ELEMENTOS CLAVE PARA EL MODELO DE CONSTRUCCIÓN DE SOFTWARE BASADO EN EL PROCES UNIFICADO

3.10.1. ROLES

Este elemento determina el "quien" es el encargado de una función en particular dentro del proceso de construcción del sistema informático.

3.10.2. ACTIVIDADES

Esta clave determina el "como" se realizará un determinado modelo dentro del proceso de construcción del software.

[5] Factibilidad Técnica: La organización debe estar preparada técnicamente para asegurar el éxito de la implementación del software en términos de hardware, software, telecomunicaciones y equipos.
[6] Factibilidad Operativa: Se cumple esta factibilidad cuando la construcción del software satisface a los usuarios en términos de requisitos y amigabilidad.

3.10.3. ARTEFACTOS

Este elemento establece el "que" elementos UML se utilizarán en cada modelo del proceso de construcción del software.

3.10.4. FLUJO DE TRABAJO

Esta clave determina el "cuando" exactamente se debe realizar los modelos dentro de cada una de las fases determinadas por el Proceso Unificado (incepción, elaboración, construcción y transición).

3.11. ROLES U OBREROS

Figura N° 25, Actividades y Artefactos de los Obreros según el Proceso Unificado.

Un rol (obrero) define el conjunto de acciones, conductas (en las actividades) y responsabilidades (en los artefactos) que puede ser desempeñado por un individuo o conjunto de individuos en la organización de desarrollo.

Cada obrero realizará un conjunto de actividades relacionadas con el dominio y conocimiento del tema.

Se puede considerar a un obrero como un "*sombrero*" que un individuo puede llevar en el proyecto, el obrero puede vestir

más de un sombrero en el proyecto de construcción del software.

Los obreros NO son los individuos, los obreros describen cómo los individuos deben comportarse en el negocio y qué responsabilidades deben tener.

El Proceso Unificado establece 30 roles diferentes. Una de las preguntas más comunes es ¿necesitamos 30 personas como mínimo para abordar la construcción de un software?, la respuesta obviamente es NO, una persona puede realizar más de un rol en el desarrollo del proyecto, eso dependerá de su experiencia, conocimiento y habilidades; se debe tener especial cuidado en la asignación de roles, este trabajo debe ser realizado por el ingeniero de software con colaboración y comunión del equipo de desarrollo, una persona no puede asumir dos roles dependientes, es decir una misma persona NO puede ser juez y parte.

Por ejemplo:

Si un individuo aborda el rol de "especificador de un elemento", NO podrá abordar el rol de "revisor" del mismo elemento.

El Proceso Unificado, considera los siguientes trabajadores:

3.11.1 TRABAJADOR COMUN (ANY WORKER)

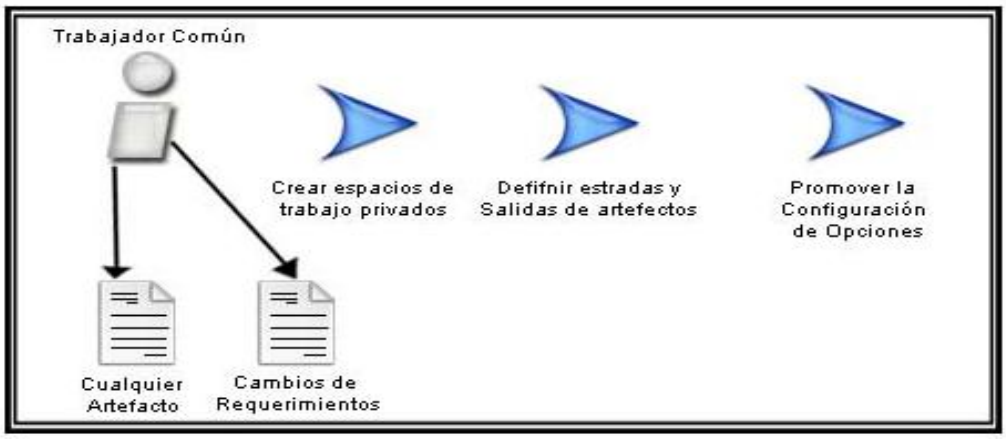


Figura N° 26, Funciones del Trabajador Común.

El trabajador común, representa al operador del sistema informático, quien puede tener niveles de acceso y privilegios, puede ingresar y salir de algún artefacto relacionado con el mantenimiento del sistema. Ver figura N° 26.

3.11.2. ARQUITECTO (ARCHITECT)

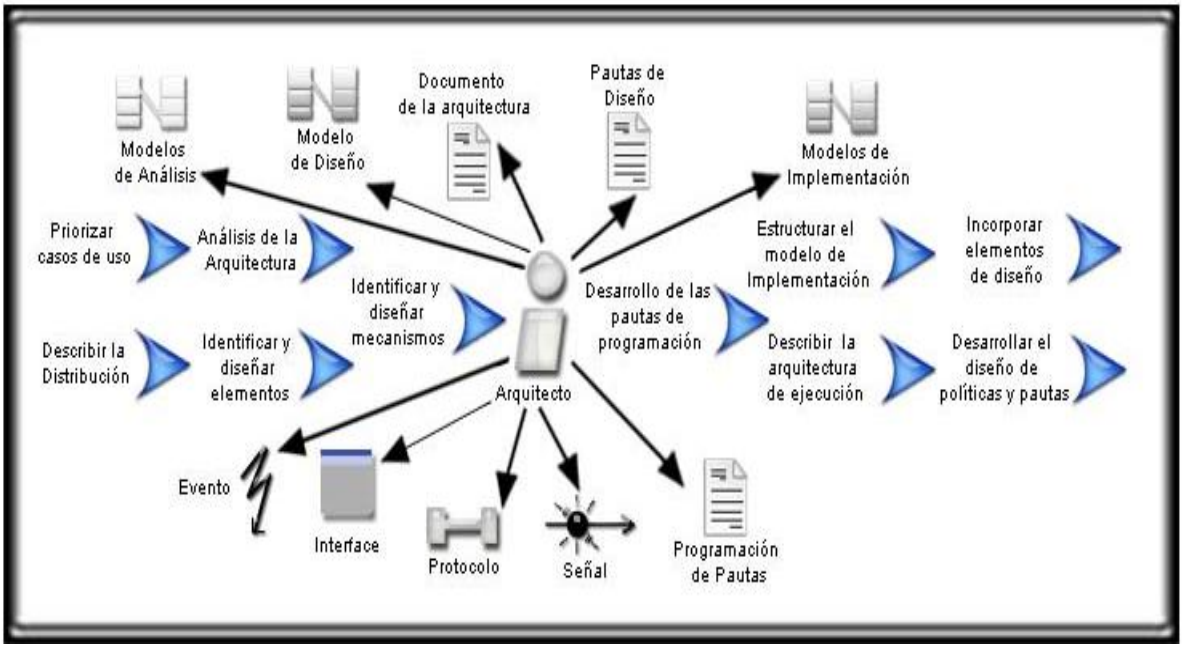


Figura N° 27, Funciones del Arquitecto.

El Arquitecto del sistema informático, dirige y coordina las actividades técnicas y los artefactos a lo largo del proyecto. El Arquitecto establece la estructura global para cada vista arquitectónica de la solución informática:
Realiza labores de descomposición de vistas, la agrupación de elementos de elementos y la comparación de labores realizadas por todos los obreros que participan en el proceso de construcción de la solución informática.
El punto de vista del Arquitecto es profundo y global e imparcial. Busca mejorar el conjunto. Ver figura N° 27.

3.11.3. REVISOR DE LA ARQUITECTURA (ARCHITECTURE REVIEWER)

Figura N° 28, Funciones del Revisor de la Arquitectura.

Planea y conduce la revisión formal de la arquitectura y del software en forma global.
El revisor de la arquitectura es un trabajador con conocimiento y experiencia de la funcionalidad total de la solución informática. Ver figura N° 28.

3.11.4 ANALISTA DE PROCESOS DE NEGOCIO (BUSINESS PROCESS ANALYST)

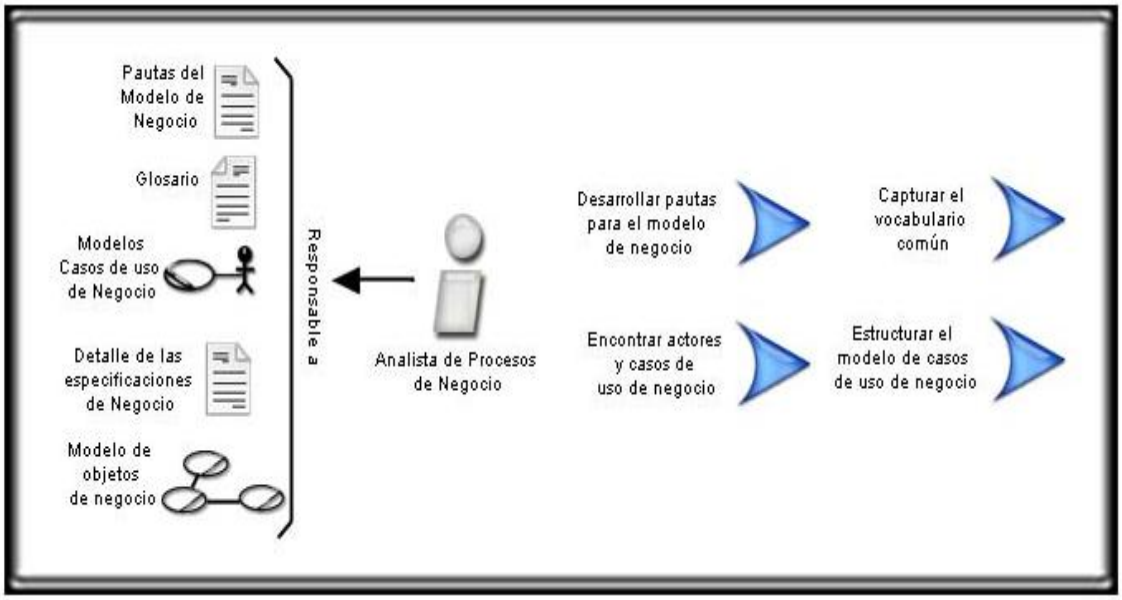


Figura N° 29, Funciones del Analista de Procesos de Negocio.

Lidera y coordina el modelado de los casos de uso de negocio para detallar y delimitar la organización que está siendo modelada, por ejemplo definir que actores internos, externos y que casos de uso de negocio existen. También analizan sus relaciones

Puede realizar labores de identificación de elementos del negocio como actores, trabajadores, documentos, objetivos, etc. Ver figura N° 29.

3.11.5 DISEÑADOR DE NEGOCIO (BUSINESS DESIGNER)

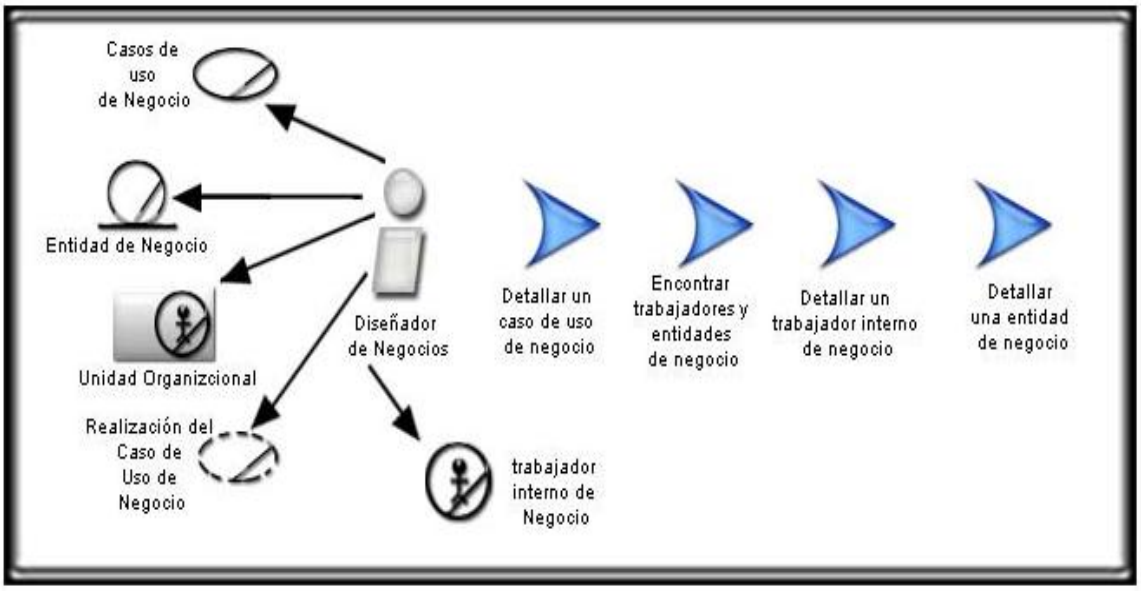


Figura N° 30, Funciones del Diseñador de Negocio.

Detalla la especificación de una parte de la organización para describir el flujo de trabajo de uno ó muchos casos de uso del negocio, se encarga de especificar que actores y entidades de negocio, se necesitan para realizar el caso de uso de negocio describiendo el comportamiento de los casos de uso en los actores y entidades del negocio. El diseñador de negocio, define las responsabilidades, operaciones, atributos y relaciones de uno o muchos trabajadores de negocio con sus respectivas entidades de negocio

Este trabajador establece los objetivos y metas del proceso empresarial, en trabajo mancomunado con el experto del negocio. Ver figura N° 30.

3.11.6. REVISOR DEL MODELO DE NEGOCIO (BUSINESS-MODEL REVIEWER)

Figura N° 31, Funciones del Revisor del Modelo de Negocio.

Participa en revisiones formales de los modelos de casos de uso y los modelos de objetos de negocio.

La persona que asume el rol de "revisor del negocio", No debe participar en las etapas de especificación y diseño de los modelos de negocio. De esta manera se garantiza la imparcialidad en la revisión y el efecto de frescura[7]. Ver figura N° 31

[7] Efecto de frescura: Se refiere al tipo de comportamiento que experimenta la persona cuando observa un acontecimiento libre de algún tipo de influencia o trastorno físico – mental; el resultado generalmente es una opinión adecuada que brinda soluciones al inconveniente que se analiza.

3.11.7. DISEÑADOR DE LA ESTRUCTURA (CAPSULE DESIGNER)

Figura N° 32, Funciones del Diseñador de la Estructura.

La función del diseñador de la estructura es asegurar que el sistema pueda responder a los eventos de manera oportuna, acorde a los requisitos del usuario

Este trabajador debe garantizar que el sistema informático cumpla con la factibilidad operativa de la solución final que se propone. Ver figura N° 32.

3.11.8. CRÍTICO DEL CÓDIGO (CODE REVIEWER)

Figura N° 33, Funciones del Crítico del Código.

Un crítico del código es responsable de:

El crítico de código, debe asegurar la calidad del código fuente a través de la constante revisión del código en cada iteración del modelo de construcción del software. El crítico planear y conduce la revisión del código fuente. Ver figura N° 33.

3.11.9. ADMINISTRADOR DE LA CONFIGURACIÓN (CONFIGURATION MANAGER)

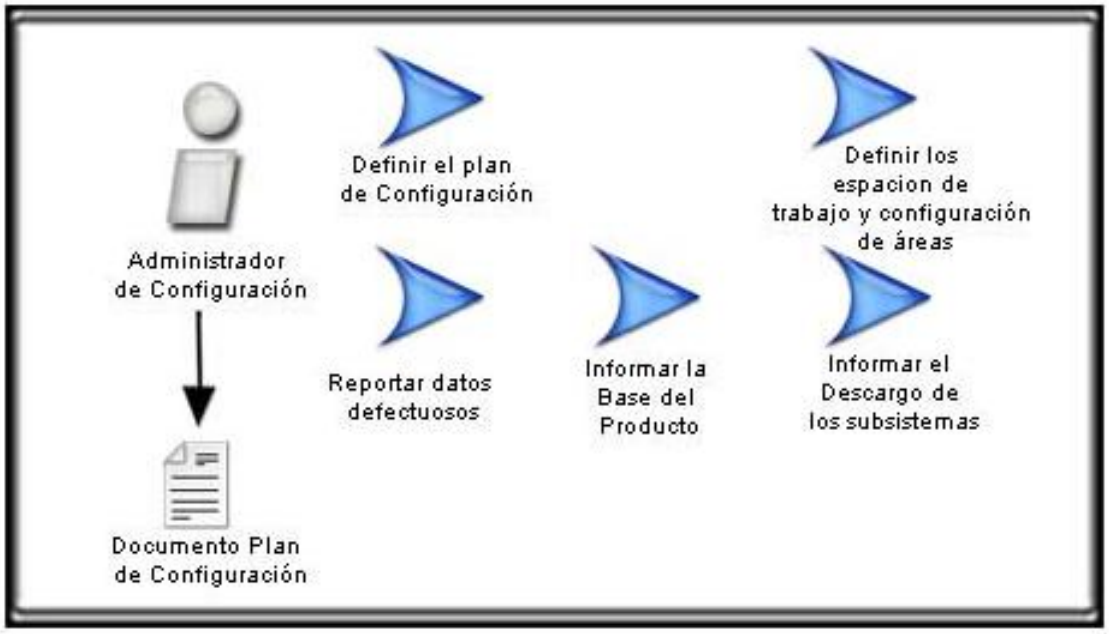


Figura N°34, Funciones del Administrador de la Configuración.

Es responsable para mantener toda la infraestructura y el ambiente que el grupo de desarrollo necesita para probar el producto que construyen. El administrador de la Configuración también es responsable de escribir el plan de control para la demanda de cambios de configuración y las estadísticas de progreso.

Este trabajador debe garantizar que el proyecto informático cumpla con la factibilidad técnica necesaria para garantizar el proceso de pruebas; este proceso es necesario para que el sistema informático ingrese a la etapa de pruebas o puesta en marcha o producción. Ver figura N° 34.

3.11.10. DESARROLLADOR DEL CURSO (COURSE DEVELOPER)

Figura N° 35, Funciones del Desarrollador del Curso.

El diseñador del curso desarrolla el material de entrenamiento para enseñar a los usuarios cómo usar el producto. Esto incluye la creación de diapositivas, notas para el estudiante, cuadros sinópticos, mapas conceptuales, material de exposición, ejemplos, guías didácticas y todo lo necesario para reforzar la comprensión del producto. Ver figura N° 35.

3.11.11. DISEÑADOR DE LA BASE DE DATOS (DATABASE DESIGNER)

Figura N° 36, Funciones del Diseñador de la Base de Datos.

El diseñador de la base de datos define que las tablas, índices, vistas, constraints, triggers, stored procedures, espacios de tablas o parámetros del almacenamiento, y otras estructuras de la base de datos que se necesitan agregar, editar, guardar y recuperar; además de anular los objetos persistentes

Este trabajador realiza la labor de diseño de la base de datos, utilizando salidas de la etapa de análisis de la base de datos, previo a la etapa de diseño. Los artefactos que utiliza este trabajador son por ejemplo: el modelo conceptual, lógico y físico de la base de datos. Ver figura N° 36.

3.11.12. GERENTE DEL DESPLIEGUE (DEPLOYMENT MANAGER)

Figura N° 37, Funciones del Gerente del Despliegue.

El gerente del despliegue es responsable para los planes de transición el producto a la comunidad del usuario, encargado de documentar los planes del despliegue.

Este trabajador planifica, controla y planifica el documento conocido como plan de despliegue, el cual garantiza la puesta en marcha de la solución informática. Ver figura N° 37.

3.11.13. REVISOR DE DISEÑO (DESIGN REVIEWER)

Figura N° 38, Funciones del Revisor de Diseño basado en el Proceso Unificado.

El revisor de diseño planea y establece políticas para las revisiones formales de los Artefactos de diseño
Este trabajador NO puede realizar acciones de especificación de artefactos para el diseño, para garantizar así una acción imparcial donde No sea juez y parte. Ver figura N° 38.

3.11.14. DISEÑADOR (DESIGNER)

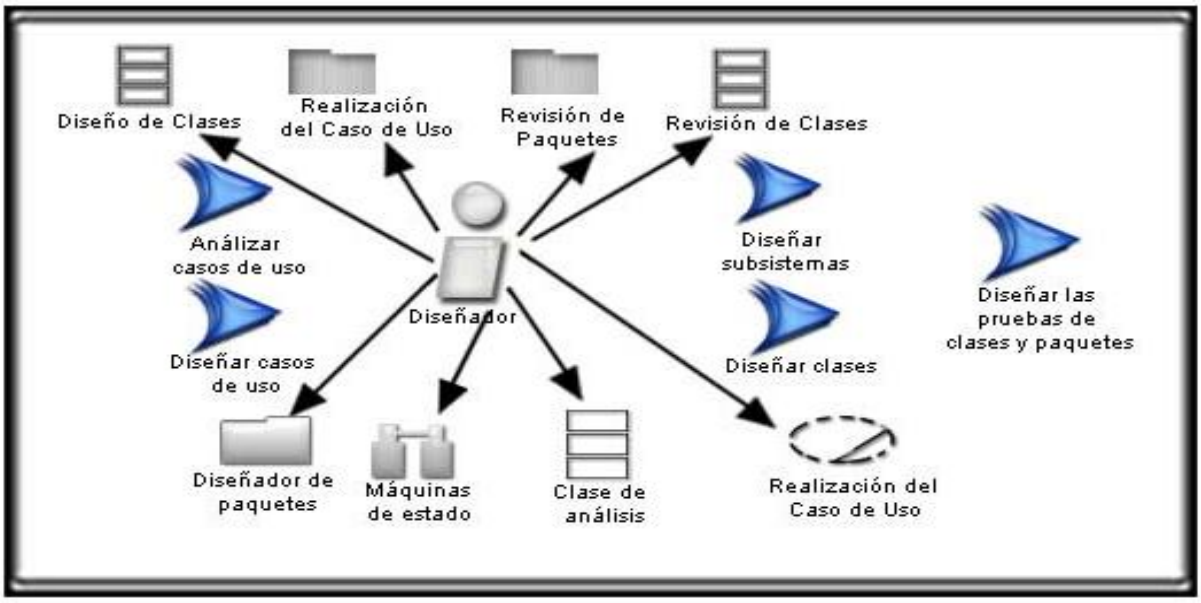


Figura N° 39, Funciones del Diseñador.

El diseñador define las responsabilidades, funcionamientos, atributos, y relaciones de uno o varias clases, determina como deben relacionarse las clases en el ambiente de aplicación.

La labor que realiza este rol es de suma importancia para el modelo de análisis y el análisis & diseño de la base de datos basado en el Proceso Unificado. Ver figura N° 39.

3.11.15. IMPLEMENTADOR (IMPLEMENTER)

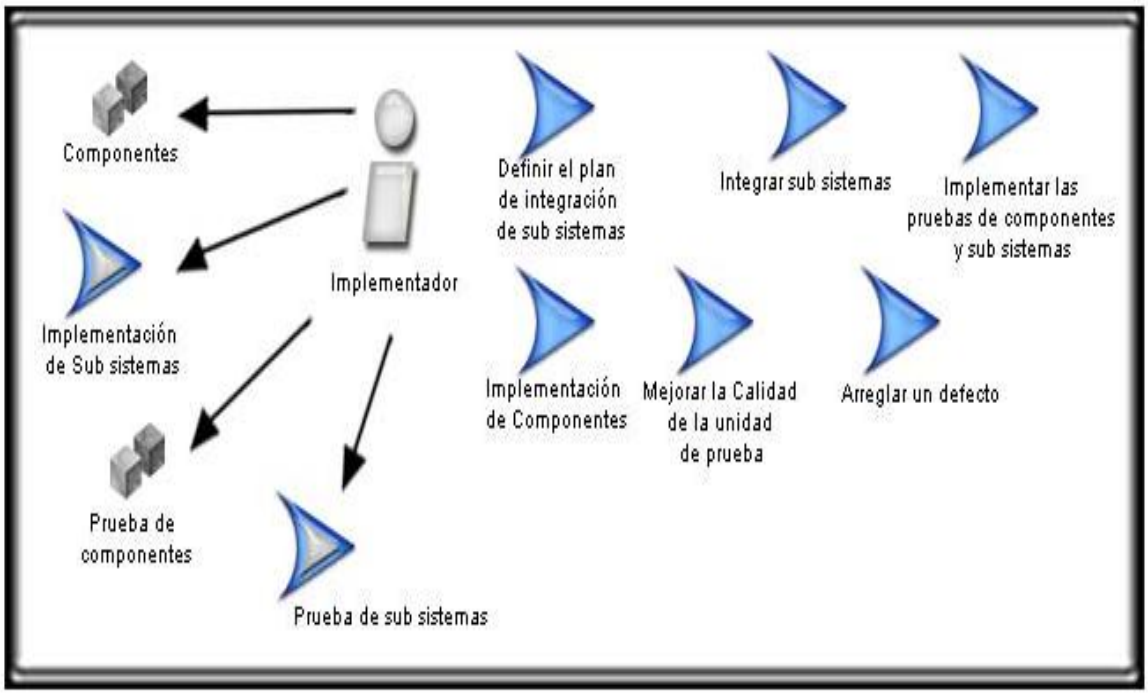


Figura N° 40, Funciones del Implementador.

Un implementador, es responsable de desarrollar y probar los componentes de acuerdo con las normas adoptadas por el proyecto, como la definición de estándares que permita a las clases integrarse con sub sistemas más grandes.

El diagrama de despliegue es el artefacto UML que permite a este trabajador la buena realización de sus funciones. Ver figura N° 40.

3.11.16. CONTROLADOR DE LA INTEGRACIÓN (INTEGRATION TESTER)

Figura N° 41, Funciones del Controlador de la Integración.

El controlador de la integración, es responsable de ejecutar la prueba de integración, sus acciones incluye:

- Realizar la prueba de estructuración y ejecución
- Realizar la evaluación de la prueba de ejecución y recuperación de los errores
- Definir La evaluación de los resultados de la prueba identificando y documentado los defectos. Ver figura N° 41.

3.11.17. CONTROLADOR DE CALIDAD (PERFORMANCE TESTER)

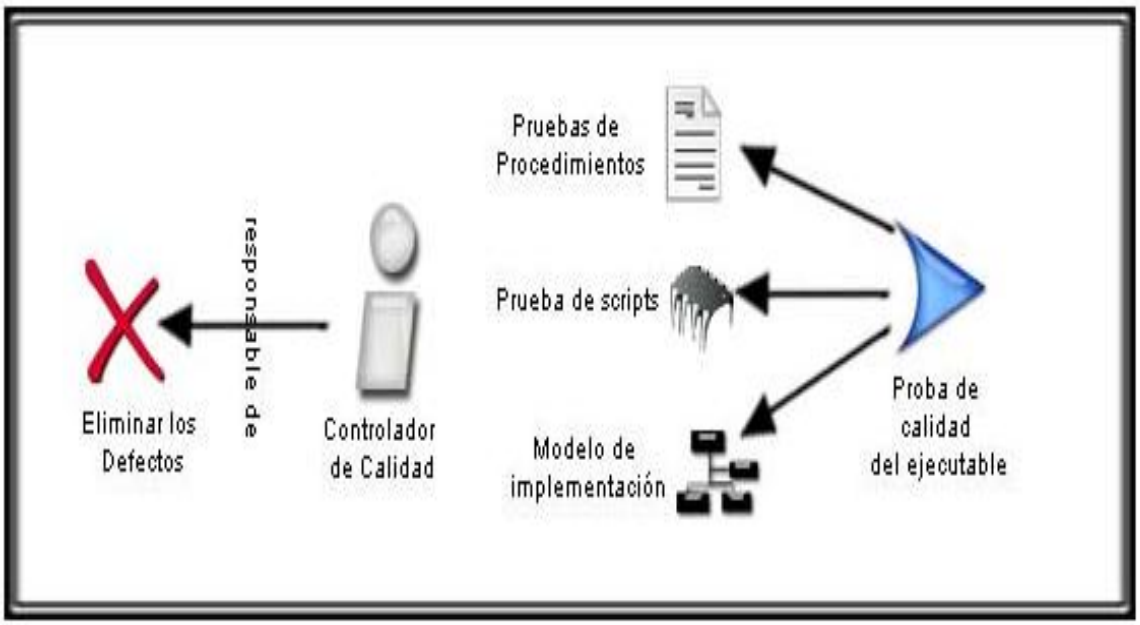


Figura N° 42, Funciones del Controlador de Calidad.

El controlador de calidad es responsable de planificar y ejecutar la prueba de calidad del software, sus acciones incluye:

- Realizar la prueba de estructuración y ejecución, con respecto a la calidad.
- Definir la evaluación de la ejecución y de la prueba de recuperación de errores evaluando los resultados de prueba, identificando y definiendo los factores que afectan a la performance y la calidad del software, ver figura N° 42.

3.11.18. INGENIERO DE PROCESOS (PROCESS ENGINEER)

Figura N° 43, Funciones del Ingeniero de Procesos.

El Ingeniero de Procesos es responsable del proceso de desarrollo de software respectivamente, incluye la definición y configuración del proceso antes del inicio del proyecto y durante el desarrollo del proyecto. Este continúa brindando mejoras en la aplicación del proceso de desarrollo del software.

Este trabajador continua brindando mejoras a la solución informática hasta lograr la homogenización respecto a los proceso de la organización ver figura N° 43.

3.11.19. GESTOR DEL PROYECTO(PROJECT MANAGER)

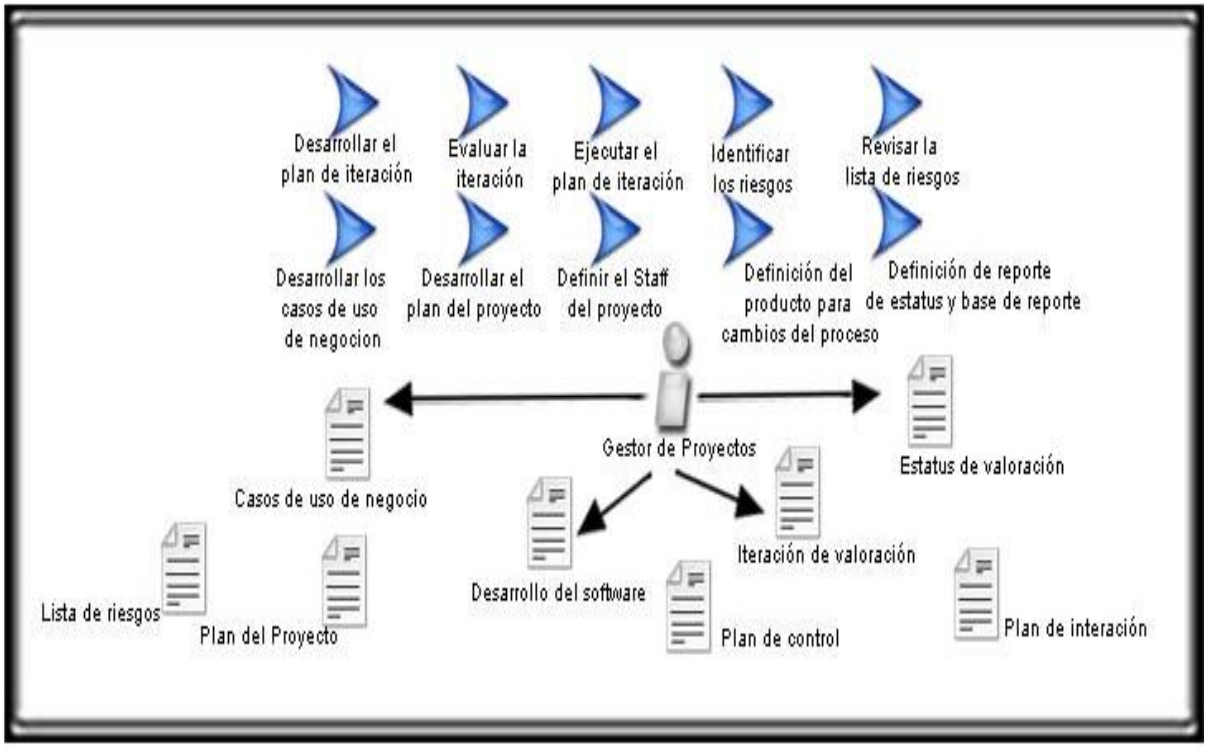


Figura N° 44, Funciones del Gestor del Proyecto.

El gestor de proyectos se encarga de administrar los recursos y prioridades. Coordina interacciones con los clientes y usuarios, hace que el interés del equipo de desarrollo se mantenga centrado con el objetivo del proyecto.

Estable el conjunto de prácticas que asegura la calidad e integridad de los artefactos usado en el proyecto, además es responsable de la efectividad del software en términos de funcionalidad, adaptabilidad, robustez y flexibilidad, ver figura N° 44.

3.11.20. CRÍTICO DE REQUISITOS (REQUIREMENTS REVIEWER)

Figura N° 45, Funciones del Crítico de Requisitos.

Planea y administra la revisión formal de todos los artefactos que participan en el análisis de requerimientos tales como actores del sistema, modelo de escenarios por actor, proceso de abstracción de actores y casos de uso, proceso de clasificación de casos de uso del sistema y actores en base al alcance del proyecto informático y la clasificación de los casos de uso en base a macros. Ver figura N° 45.

3.11.21. INVOLUCRADOS (STAKEHOLDERS)

Los involucrados son individuos y organizaciones que están activamente involucrados con el proyecto o cuyos intereses pueden estar afectados positiva o negativamente por los resultados de la ejecución del proyecto o de la culminación del mismo. Ellos también pueden influenciar sobre el proyecto y sus resultados.

El equipo de gerencia del proyecto debe identificar a los involucrados del proyecto, determinar sus requerimientos,

luego administrar e influenciar esos requerimientos para seguir el éxito del proyecto.

La identificación de los involucrados es a menudo tediosa.

Los involucrados principales del un proyecto, incluyen a:

- Gerente del proyecto
- Cliente
- Organización ejecutora
- Miembros del equipo de desarrollo
- Patrocinador

3.11.22. ADMINISTRADOR DEL SISTEMA (SYSTEM ADMINISTRATOR)

Figura N° 46, Funciones del Administrador del Sistema.

El administrador del sistema mantiene el ambiente de desarrollo, hardware, software, administración del sistema, realiza copias de seguridad, registra a los usuarios estableciendo sus privilegios, ver figura N° 46.

3.11.23. ANALISTA DEL SISTEMA (SYSTEM ANALYST)

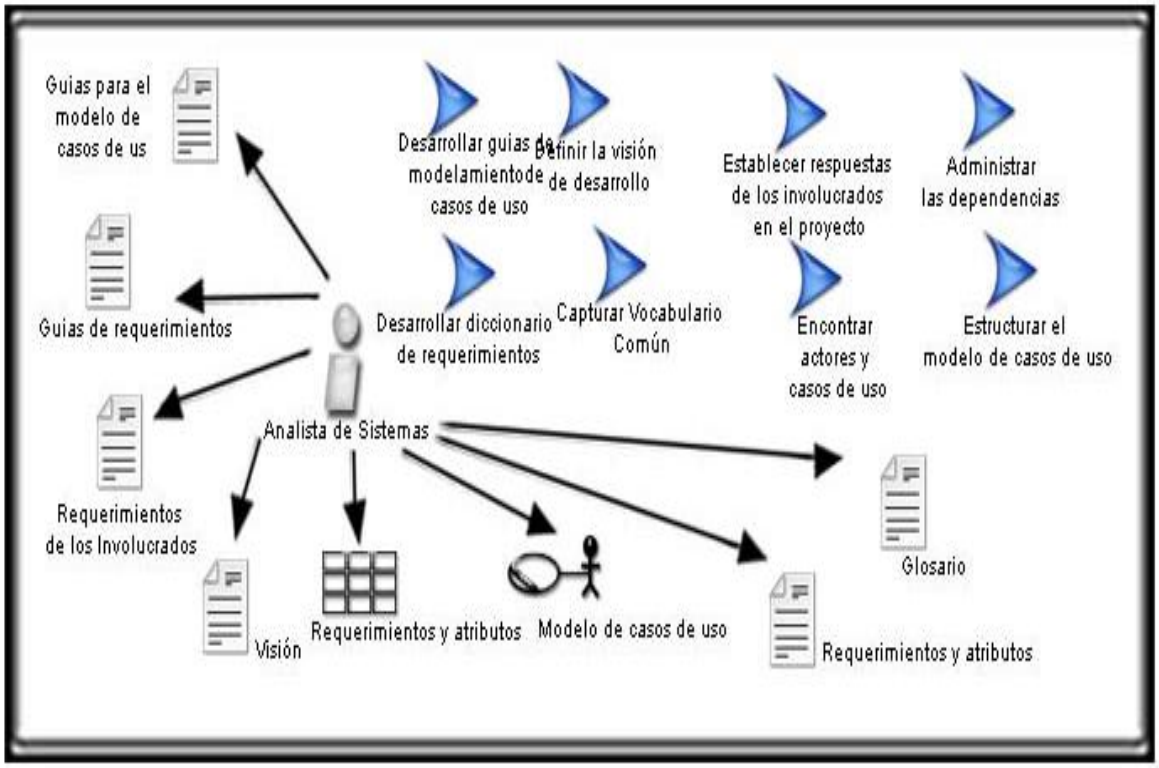


Figura N° 47, Funciones del Analista del Sistema.

El analista del sistema, planea y coordina el proceso de identificación, selección de los modelos de casos de uso, estableciendo la funcionalidad y parámetros del sistema.

Algunas de las actividades son:

- La identificación de los actores y casos de uso
- La realización de los modelos de casos de uso

Para más detalle. Ver la figura N° 47.

3.11.24. INTEGRADOR DEL SISTEMA (SYSTEM INTEGRATOR)

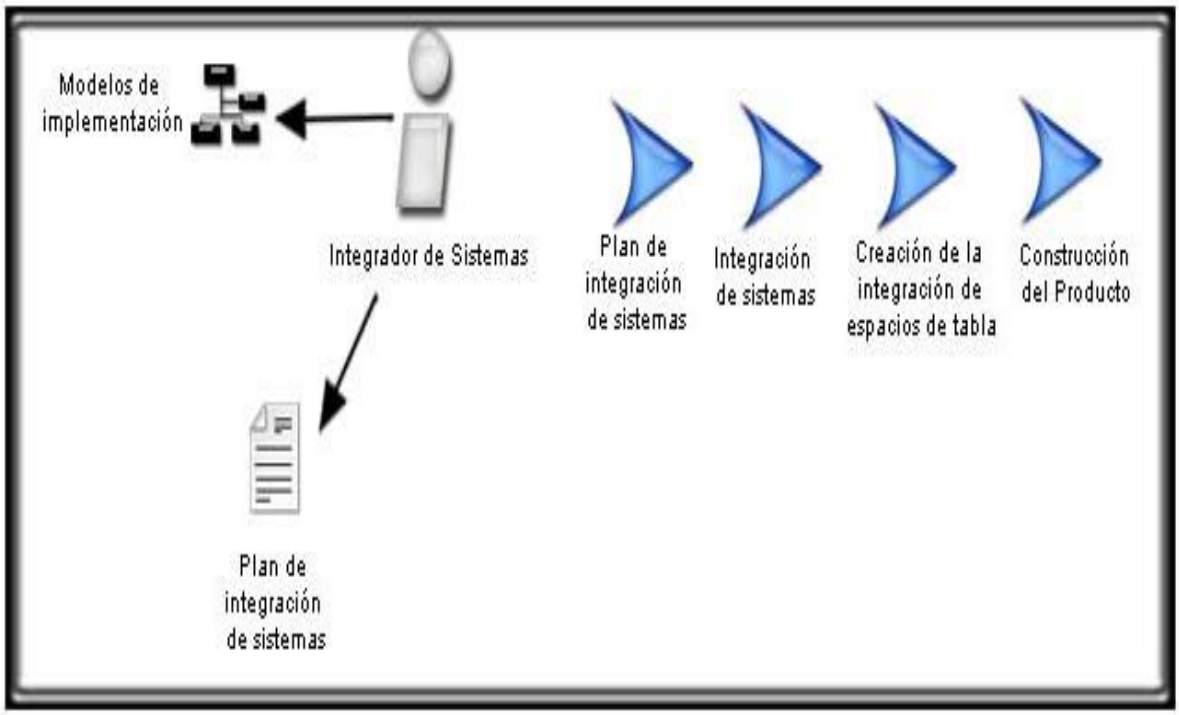


Figura N° 48, Funciones del Integrador de Sistemas.

Es el encargado de planear el proceso de integración del software con otros sistemas informáticos construidos en forma paralela ó con los que ya existen en la organización patrocinadora, el integrador debe establecer políticas de integración con la finalidad de evitar conflictos de funcionamiento global con los software actuales y los futuros en la organización, ver figura N° 48.

3.11.25. PROBADOR DEL SISTEMA (SYSTEM TESTER)

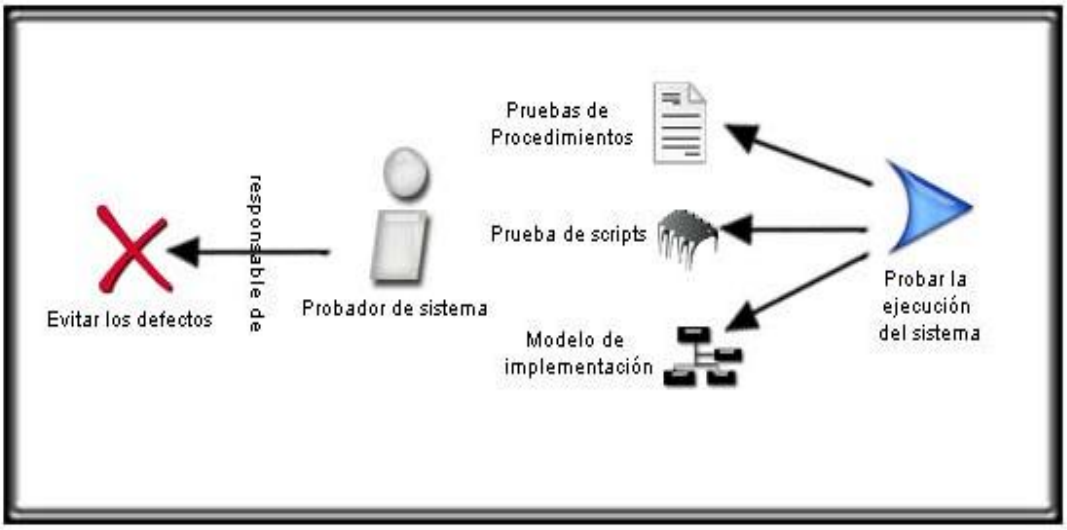


Figura 49, Actividades del Probador del Sistema.

Este trabajador es el encargado de planificar, diseñar y ejecutar las pruebas del sistema de información, las pruebas incluyen:

- La prueba de estructura y ejecución
- La evaluación del proceso de ejecución de pruebas y propuesta de solución de errores
- La evaluación de resultados del proceso de prueba y documentación del conjunto de errores hallados en el proceso. Ver figura N° 49.

3.11.26. DOCUMENTADOR (TECHNICAL WRITER)

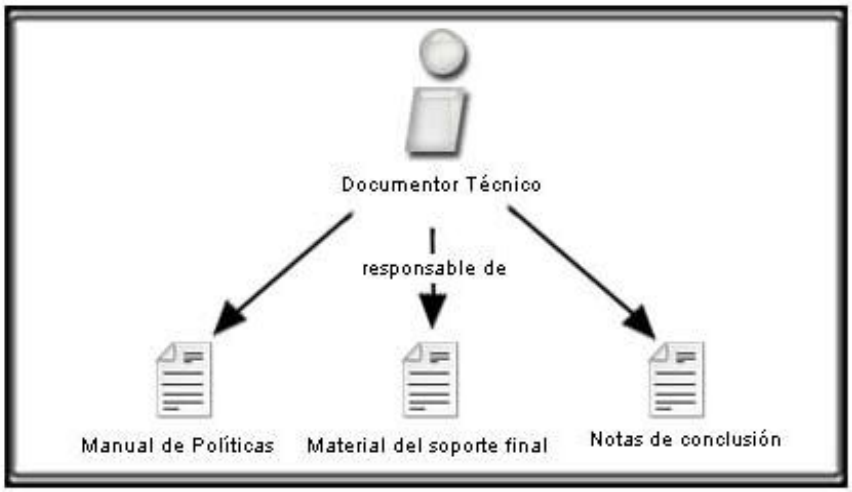


Figura N°50, Funciones del Documentador.

Es el encargado de producir material de apoyo al usuario final como las guías del usuario, los textos de ayuda, las notas del descargo, manuales de uso para el usuario y glosario de términos técnicos. Ver fig. N° 50.

3.11.27. DISEÑADOR DE PRUEBAS (TEST DESIGNER)

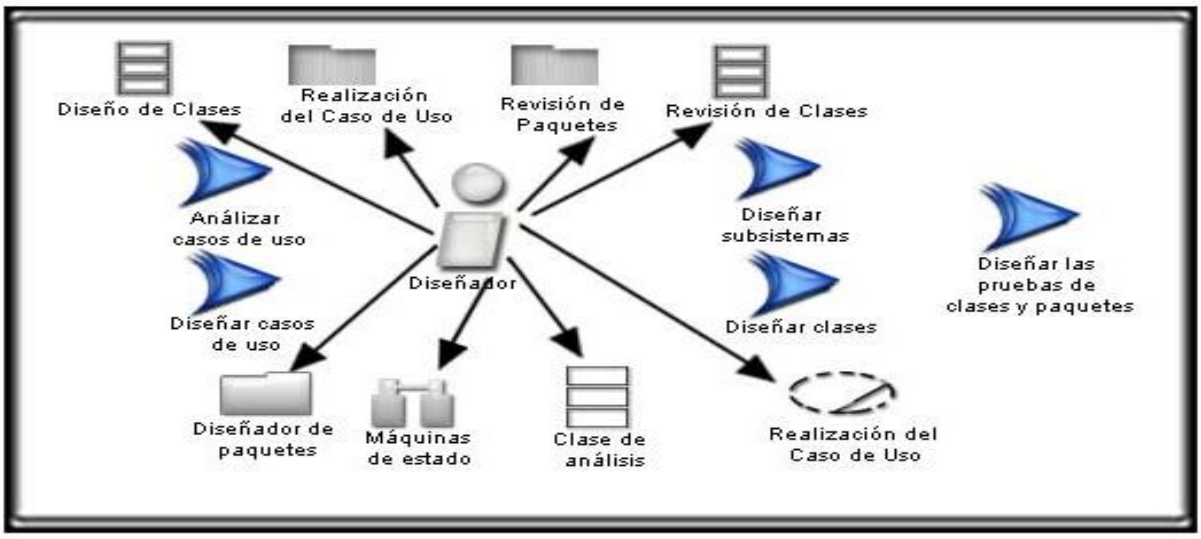


Figura N° 51, Funciones del Diseñador de Pruebas.

El Diseñador de Pruebas es el principal obrero en el proceso de pruebas, es el encargado de la planificación, aplicación y evaluación de las pruebas, incluye las siguientes actividades:

- La generación del plan y modelo de prueba
- La aplicación de procedimientos de prueba
- La evaluación de la estructura, resultados y efectividad de las pruebas, para más detalle, Ver figura N° 51.

3.11.28. ADMINISTRADOR DE HERRAMIENTAS (TOOLSMITH)

Figura N° 52, Funciones del Administrador de Herramientas.

Es el encargado de desarrollar las herramientas de apoyo a necesidades especiales, proporciona herramientas y procesos adicionales como solución a tareas tediosas en la corrección de errores, define además la buena integración entre los

artefactos que se utilizan en el proceso de construcción del software. Ver figura N° 52.

3.11.29. ESPECIFICADOR DE CASOS DE USO (USE-CASE SPECIFIER)

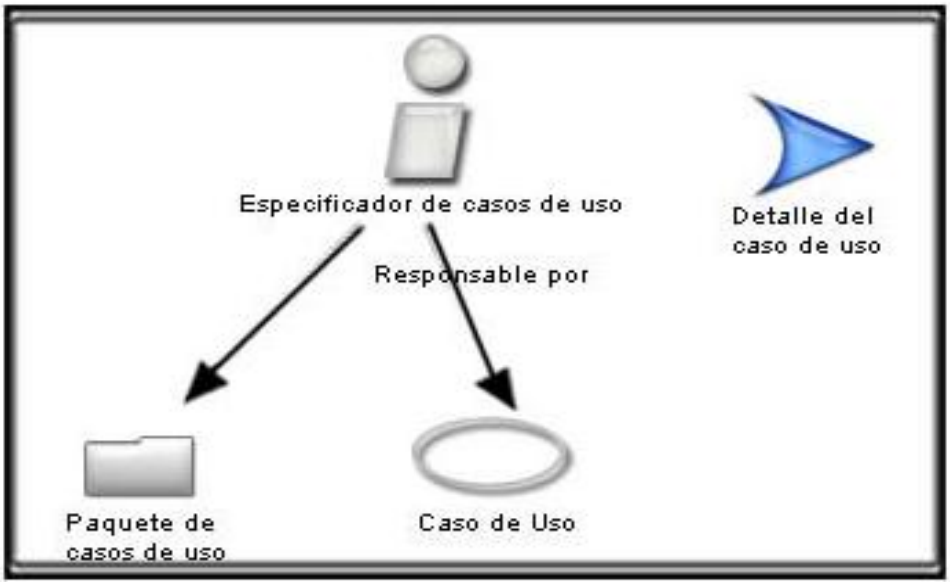


Figura N° 53, Funciones del Especificador de Casos de Uso.

Es el encargado de detallar la especificación de una parte de la funcionalidad del sistema describiendo los aspectos de requerimiento de uno o muchos casos de uso, además es responsable del mantenimiento e integración del paquete de casos de uso (casos de uso, actores, modelo de casos de uso), ver figura N° 53.

3.11.30. DISEÑADOR DE LAS INTERFACES DE USUARIO (USER INTERFACE DESIGNER)

Figura N° 54, Funciones del Diseñador de la Interfaces de Usuario.

Es el encargado de coordinar las actividades de análisis de prototipos con respecto a las interfaces de usuario, mediante:

- La captura de requerimientos para las interfaces de usuario
- La construcción de prototipos de las interfaces de usuario
- La consideración de opiniones de los involucrados del proyecto para la definición de interfaces.

- Las funciones y/o requerimientos de los usuarios son prioridad para este trabajador. Ver figura N° 54.

3.12. ACTIVIDAD

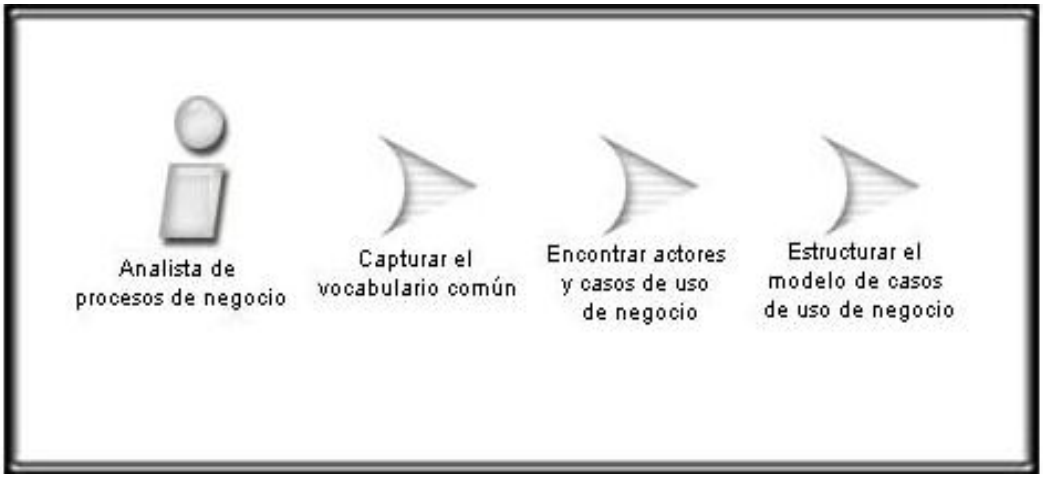


Figura N° 55, Actividades del Analista de Procesos de Negocios.

La actividad, describe una unidad de trabajo que puede ser asignada a un trabajador. Ejemplo "Crear o modificar un artefacto", Ver figura N° 55.

Una actividad lleva entre un par de horas, un par de días ó un poco más dependiendo de la habilidad, experiencia y conocimiento del trabajador, involucra un solo trabajador y un número pequeño de artefactos.

Las actividades se consideran en la planificación y evaluación del progreso del proyecto.

3.13. ARTEFACTO

Definido como la pieza de información que es producida, modificada, ó utilizada por un proceso en particular, son productos tangibles del proyecto, usados por los trabajadores para realizar nuevas actividades y son el resultado de esas actividades. Pueden ser los siguientes:

- El documento, donde se especifiquen a los casos de uso de negocio ó donde se define la arquitectura del software.
- El modelo, como el modelo de caso de uso, modelo de análisis, modelo de diseño, etc.
- Un elemento dentro de un modelo tal como una clase ó un sub sistema.
- Un diagrama que otorgue valor y entendimiento al proceso.

Ahora mencionaré cuales son los artefactos a utilizar en cada de una de etapas de construcción de software según el Proceso Unificado, tanto en las etapas centrales como de soporte.

No se trata de memorizar que artefacto producirá un trabajador, el uso constante de estos hará que sean parte tácita del desarrollo de software, sólo tenemos que aplicarlo.

Para evitar la complejidad que *DETESTO*, aprenderemos los diferentes artefactos por etapas de construcción del software utilizando gráficos relacionados con la notación UML:

3.13.1. ARTEFACTOS DEL MODELO DE NEGOCIO

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de "***MODELO DE NEGOCIO***".

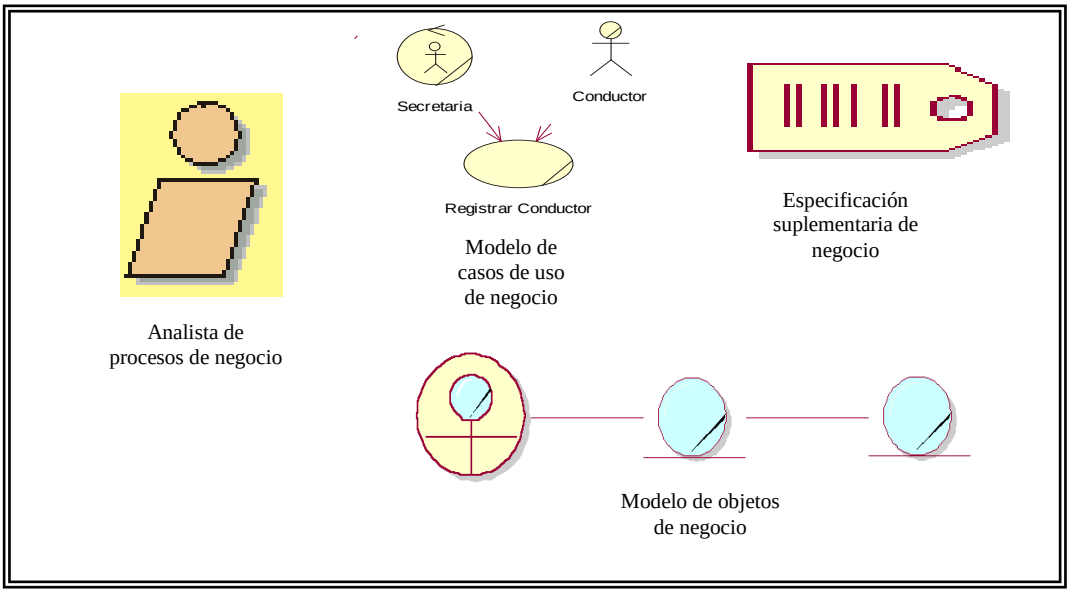


Figura N° 56, Artefactos producidos ó utilizados por el trabajador Analista de procesos de negocio.

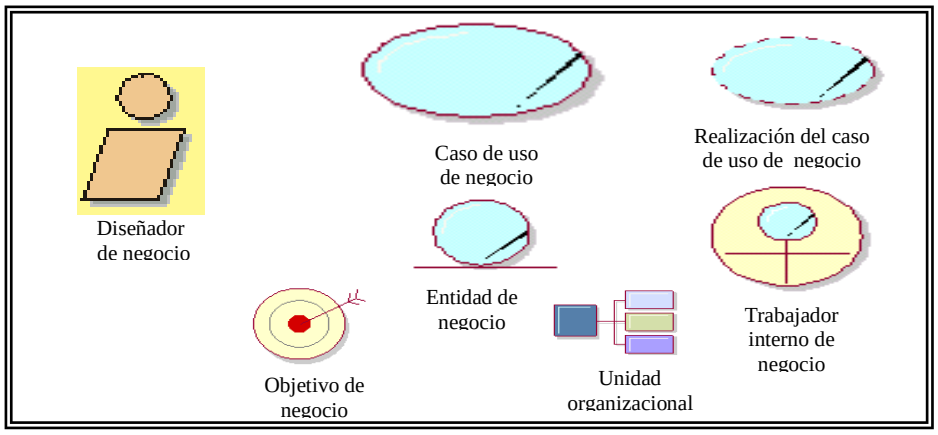


Figura N° 57, Artefactos producidos ó utilizados por el trabajador Diseñador de Negocio. Estos artefactos permiten realizar el modelo de casos de uso del negocio y el modelo de análisis del negocio.

3.13.2. ARTEFACTOS DE REQUERIMIENTOS

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"ANALISIS DE REQUERIMIENTOS"***.

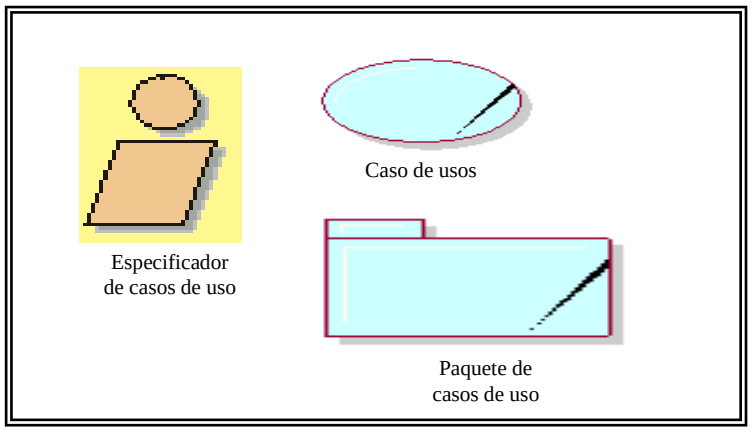


Figura N° 58, Artefactos producidos ó utilizados por el trabajador Especificador de Casos de Uso.

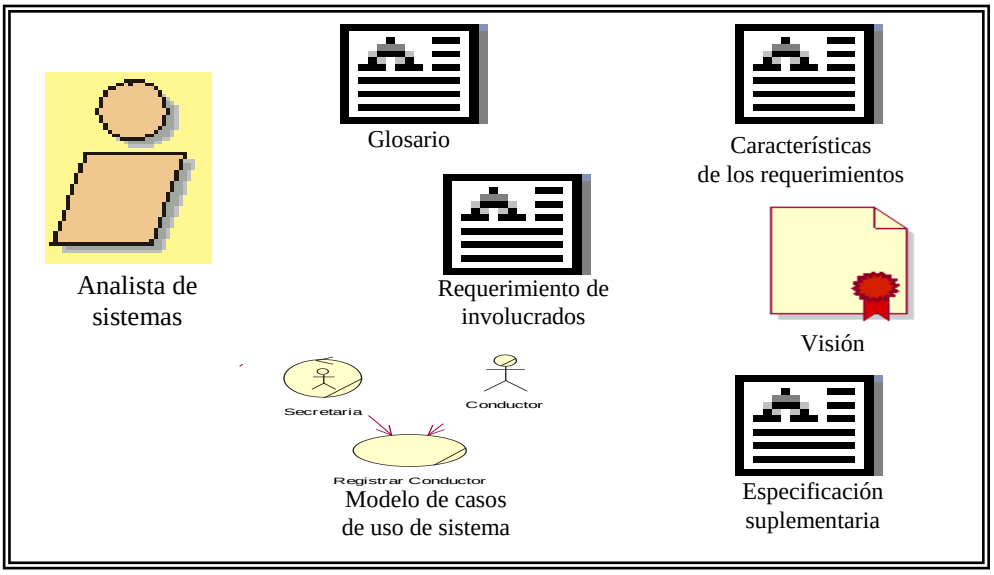


Figura N° 59, Artefactos producidos ó utilizados por el trabajador Analista de Sistemas.

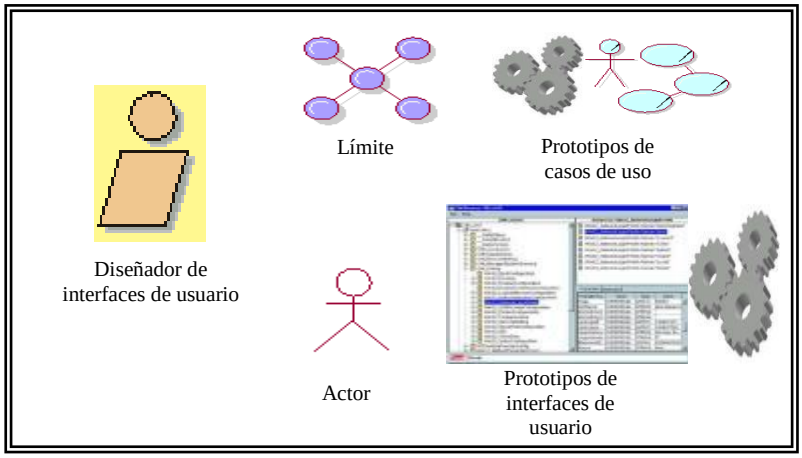


Figura N° 60, Artefactos producidos ó utilizados por el trabajador Diseñador de interfaces de usuario.

3.13.3. ARTEFACTOS DE DISEÑO

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"DISEÑO"***.

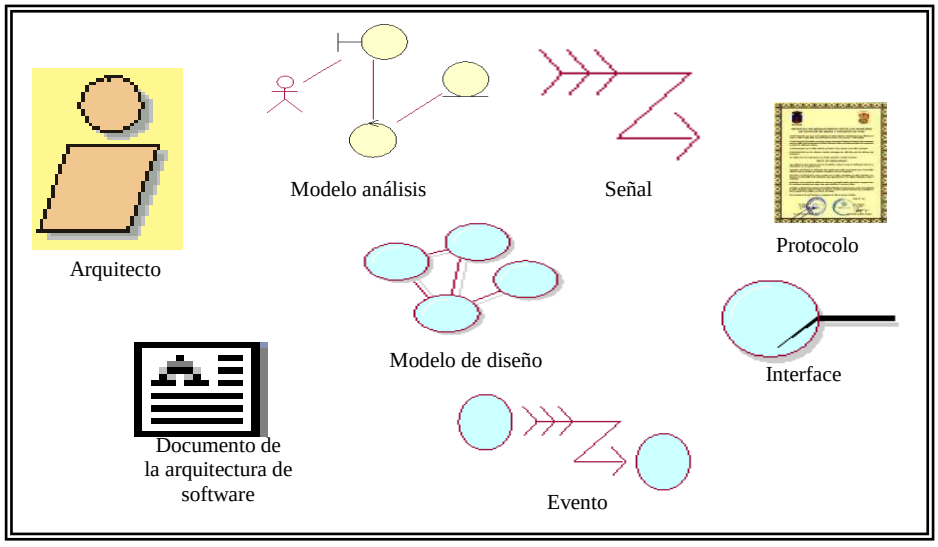


Figura N° 61, Artefactos producidos ó utilizados por el trabajador Arquitecto.

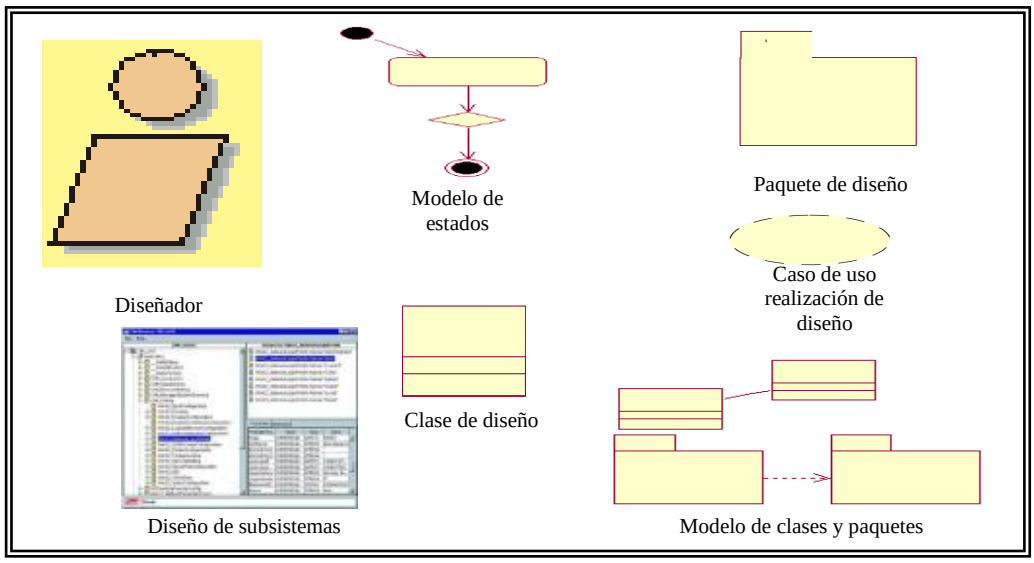


Figura N° 62, Artefactos producidos ó utilizados por el trabajador Diseñador.

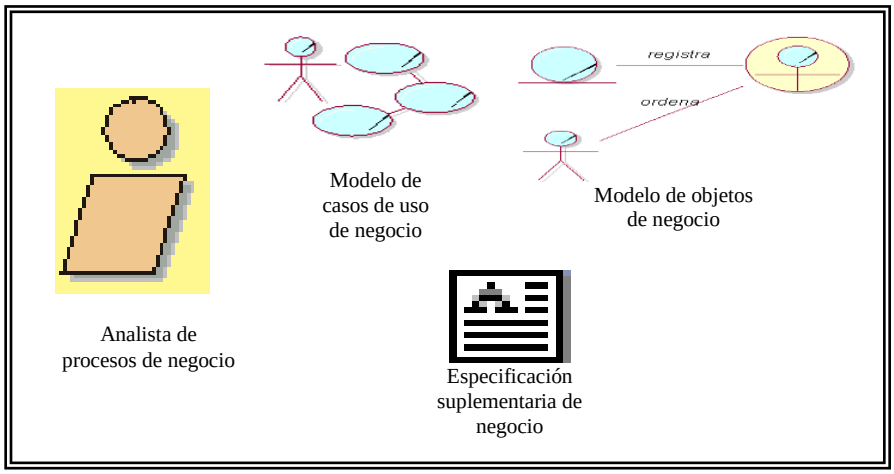


Figura N° 63, Artefactos producidos ó utilizados por el trabajador Analista de Proceso de negocio.

Figura N° 64, Artefactos producidos ó utilizados por el trabajador Diseñador de la estructura.

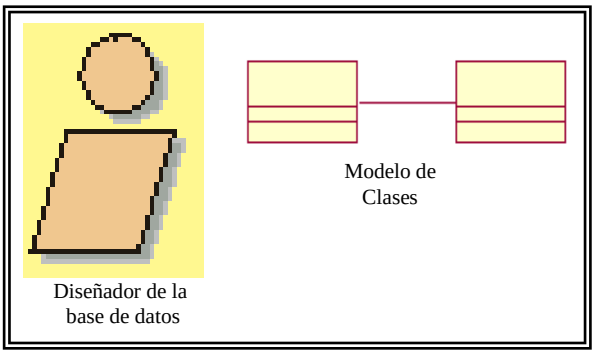


Figura N° 65, Artefactos producidos ó utilizados por el trabajador Diseñador de la Base de Datos.

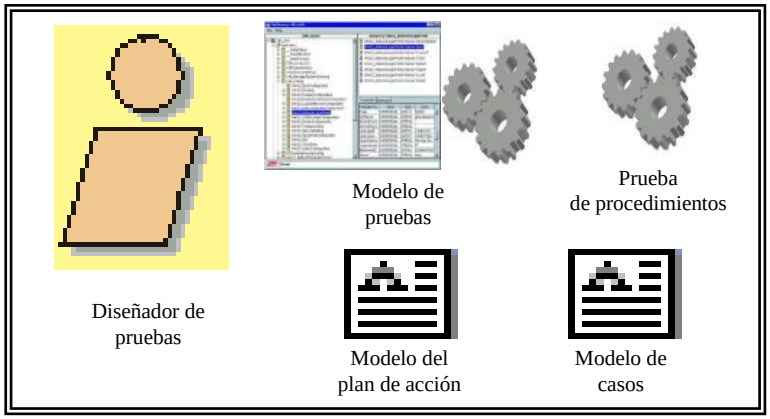


Figura N° 66, Artefactos producidos ó utilizados por el trabajador Diseñador de Pruebas.

3.13.4. ARTEFACTOS DE IMPLEMENTACIÓN

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"IMPLEMENTACIÓN"***.

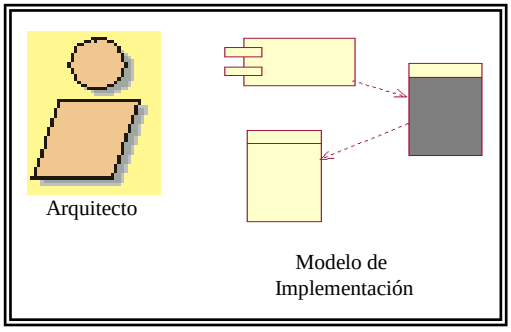


Figura N° 67, Artefactos producidos ó utilizados por el trabajador Arquitecto.

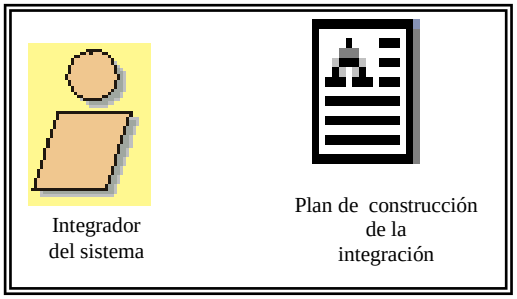


Figura N° 68, Artefactos producidos ó utilizados por el trabajador Integrador del Sistema.

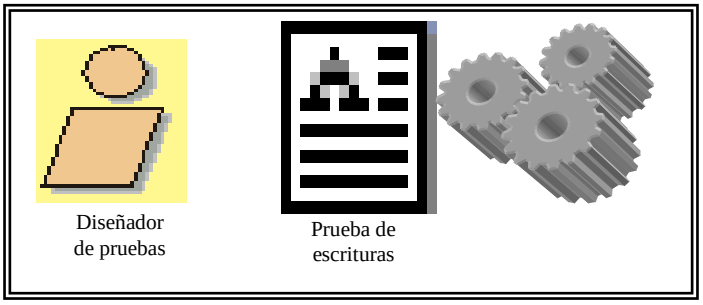


Figura N° 69, Artefactos producidos ó utilizados por el trabajador Diseñador de Pruebas.

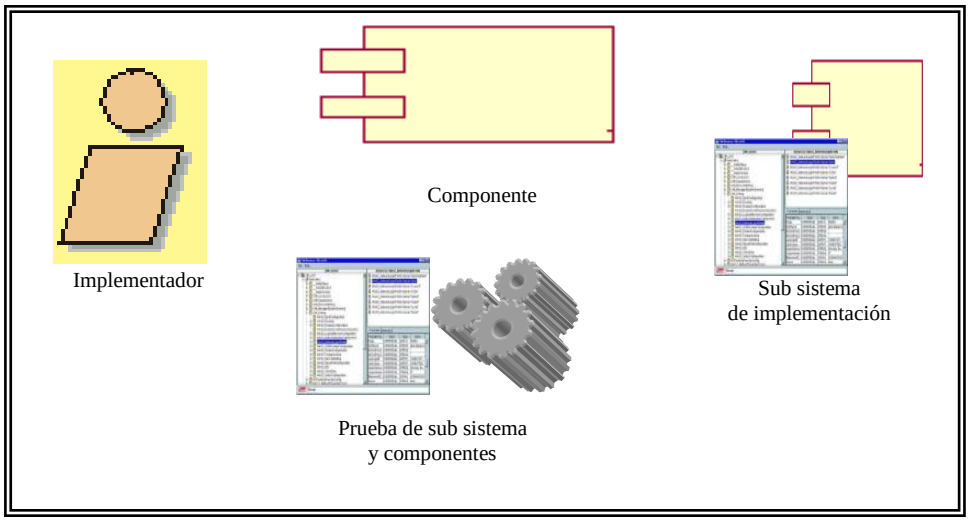


Figura N° 70, Artefactos producidos ó utilizados por el trabajador Implementador.

3.13.5. ARTEFACTOS DE DESPLIEGUE

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"DESPLIEGUE"***.

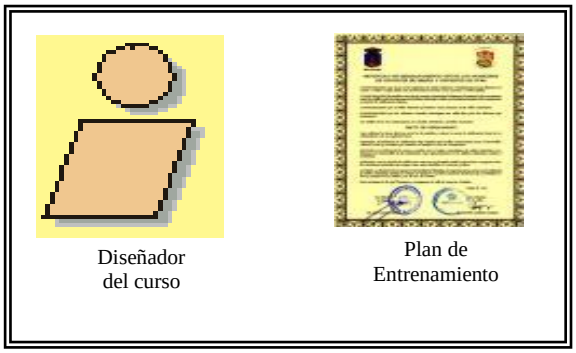


Figura N° 71, Artefactos producidos ó utilizados por el trabajador Diseñador del Curso.

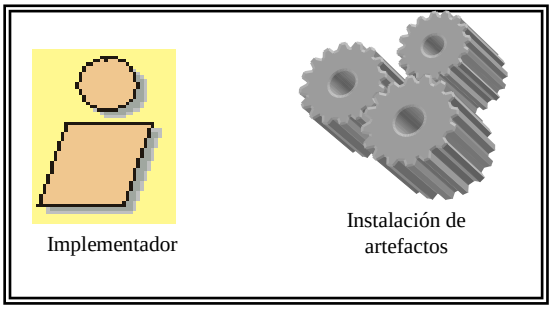


Figura N° 72, Artefactos producidos ó utilizados por el trabajador Implementador.

Figura N° 73, Artefactos producidos ó utilizados por el trabajador Documentador Técnico.

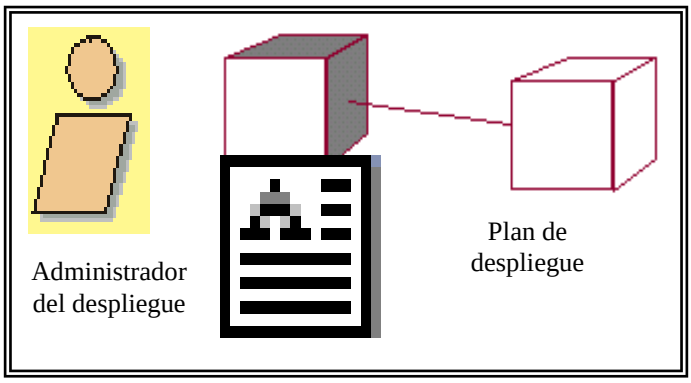


Figura N° 74, Artefactos producidos ó utilizados por el trabajador Administrador del Despliegue.

3.13.6. ARTEFACTOS DE ADMINISTRACIÓN

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"ADMINISTRACIÓN"***.

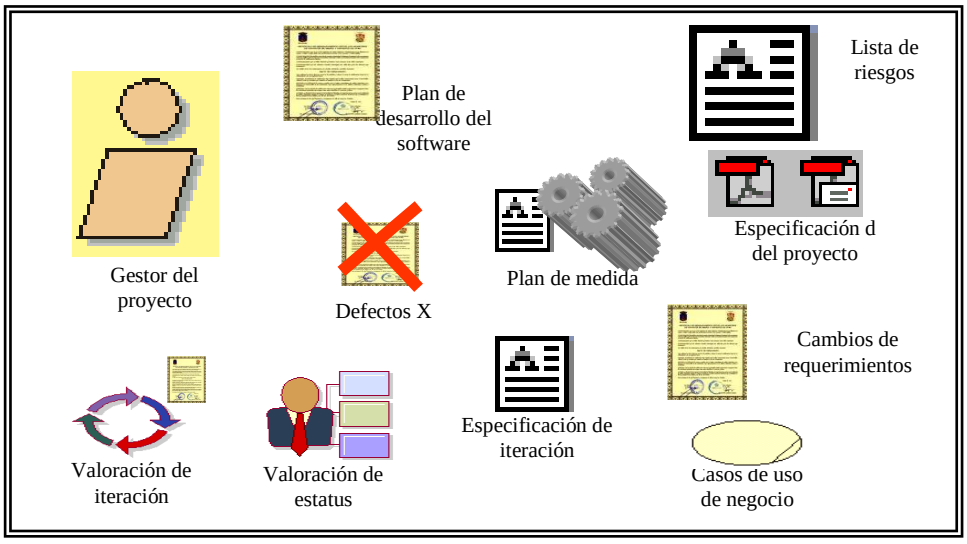


Figura N° 75, Artefactos producidos ó utilizados por el trabajador Gestor del Proyecto.

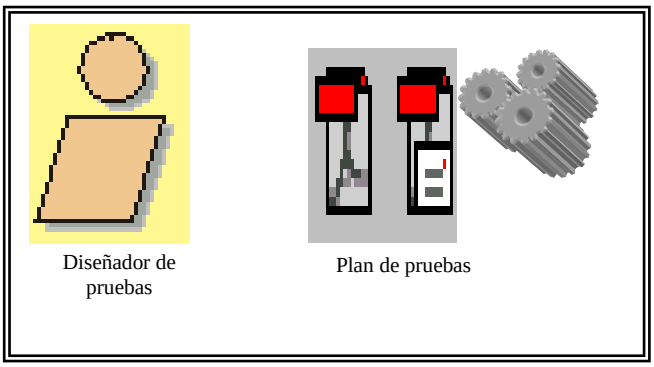


Figura N° 76, Artefactos producidos ó utilizados por el trabajador Diseñador de Pruebas.

Figura N° 77, Artefactos producidos ó utilizados por el trabajador Ingeniero de Procesos.

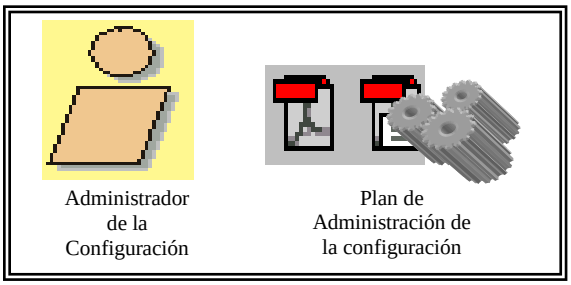


Figura N° 78, Artefactos producidos ó utilizados por el trabajador Administrador de la Configuración.

3.11.7. ARTEFACTOS DE ESPECIFICACIÓN Y PAUTAS

A continuación detallo todos los artefactos creados ó utilizados por un trabajador, con respecto a la etapa de ***"ESPECIFICACIONES"***.

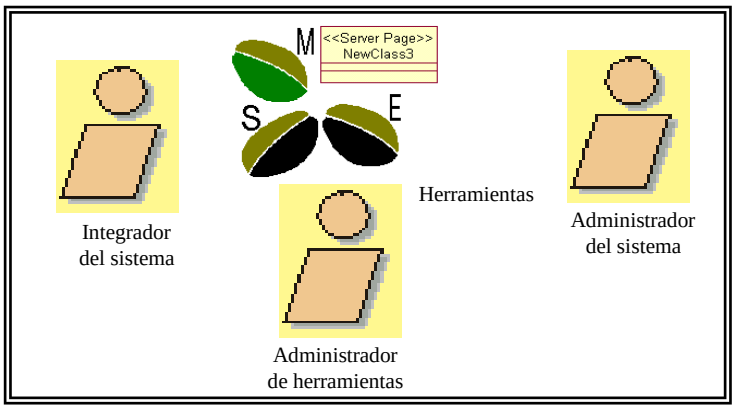


Figura N° 79, Artefactos producidos ó utilizados por los administradores.

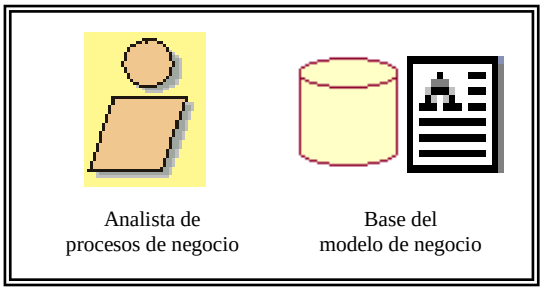


Figura N° 80, Artefactos producidos o utilizados por el Analista de Procesos de Negocio.

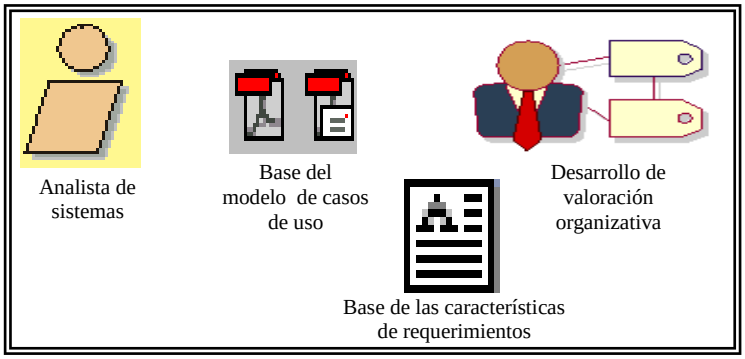


Figura N° 81, Artefactos producidos o utilizados por el trabajador Analista de Sistemas.

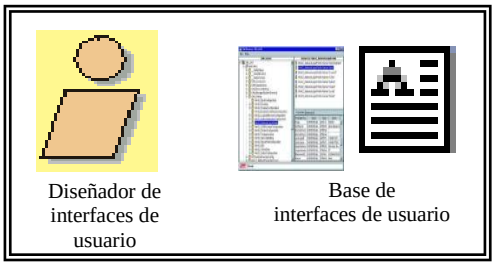


Figura N° 82, Artefactos producidos ó utilizados por el trabajador Diseñador de Interfaces de usuario.

3.14. PROCESO UNIFICADO Y EL CASE DE MODELAMIENTO RATIONAL ROSE, EN EL TRABAJO DE NEGOCIO.

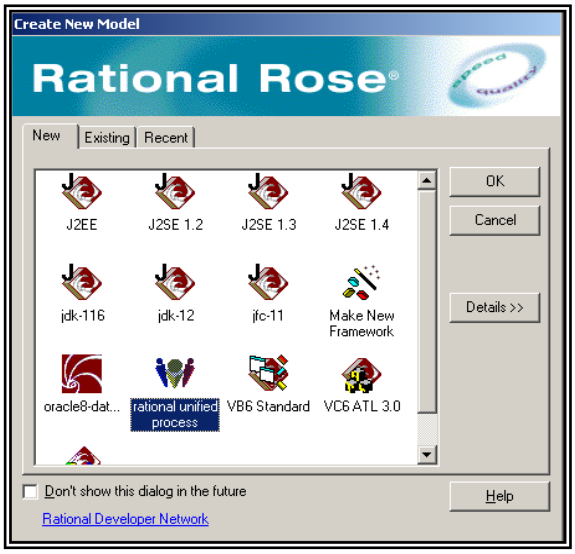


Figura N° 83, Ventana de Creación de nuevos modelos en el Case de Modelado Rational Rose[8].

El Case Rational Rose, es la herramienta de modelado que soporta las fases y etapas del proceso RUP. Para utilizar las ayudas y librerías del Proceso Unificado en Rational Rose, hacer clic en la plantilla Rational Unified Proccess, luego clic en el botón ok, ver figura N° 82.

[8] El case de modelado Rational Rose, ha sido utilizado en el presente texto como referencia en la implementación de los ejemplos, cabe resaltar que su uso es para fines estrictamente académicos.

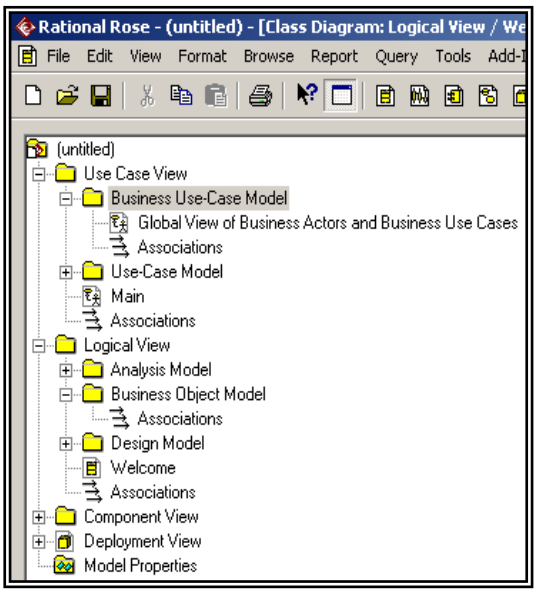


Figura N° 84, Browser del Case Rational Rose, señalando al paquete de casos de uso de negocio.

Después de cagar las librerías del Proceso Unificado, se tiene una plantilla de trabajo, desde un punto de vista general, definido en el paquete *"Business Use Case Model"*, entorno donde se crea los siguientes artefactos:

- Caso de uso de negocio
- Trabajador de negocio
- Actor de negocio
- Unidad organizacional
- Modelo de casos de uso de uso de negocio.

Además se realzan las siguientes actividades:

- Definición de unidades organizacionales
- Selección de actores internos y externos
- Definición de procesos empresariales

- Establecimiento de criterios y políticas de acción para la creación de los modelo de casos de uso de negocio, ver figura N° 83.

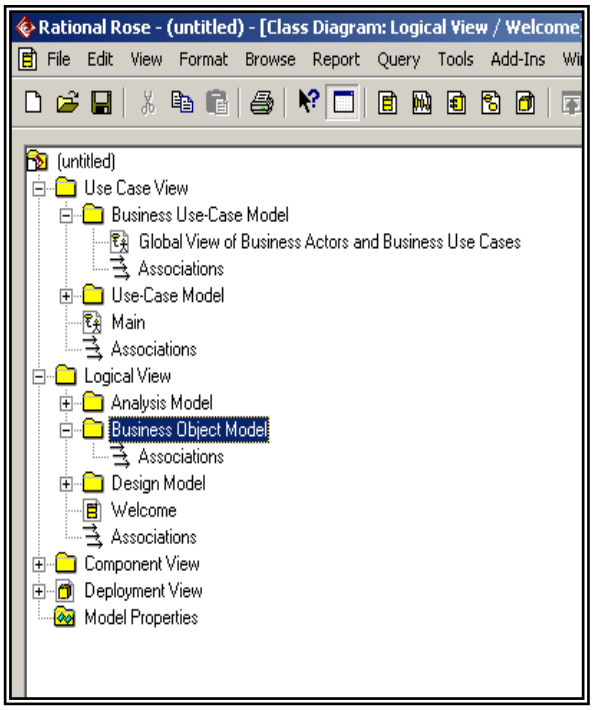


Figura N° 85, Browser del Case Rational Rose, señalando al paquete modelo de objetos de negocio.

El punto de vista detallado, se define en el paquete "*Business Object Model*", en este entorno se crea los siguientes artefactos:

- Diagramas de realización del caso de uso de negocio
- Diagramas de clases
- Diagramas de secuencia
- Diagramas de colaboración
- Entidades de negocio

Además se realizan las siguientes actividades:

- Definición de entidades de negocio
- Realización del detalle de negocio mediante artefactos de creación de clases (diagrama de clases) y de interacción de objetos (diagrama de secuencia ó colaboración), ver figura N° 84.

3.14. RESUMEN

Hasta este capítulo, usted debe comprender las razones de optar por el Proceso Unificado para la construcción de software en estos días, ya debemos estar familiarizados con todos los elementos que implica el Proceso Unificado y listos para adentrarnos en el maravillo y siempre sorprendente mundo de la construcción de software.

Los siguientes conceptos deben ser conocidos al 100% para continuar con el siguiente capítulo:

- Estructura del Proceso Unificado
- Fase y Etapas del Proceso Unificado
- Trabajadores ó roles
- Actividades
- Entorno de trabajo para el Proceso Unificado
- Artefactos por cada trabajador establecido en el Proceso Unificado.

Hora de comenzar con la construcción de un sistema de información en base a los requerimientos del principal involucrado del proyecto "*El cliente*".

¡A TRABAJAR!

3.14. EVALUACIÓN DE COMPETENCIAS DEL CAPÍTUL

- Realizar un cuadro comparativo entre artefacto, rol y actividad.
- Graficar el proceso unificado bajo la perspectiva de modelos y de fases.
- ¿Qué existe entre el proceso unificado y el UML?

CAPITULO IV

MODELO DE NEGOCIO BASADO EN EL PROCESO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Instituir la importancia del análisis de requerimientos dentro del proceso de construcción.
- Explicar los diversos artefactos UML a utilizar en la etapa de construcción del software.
- Establecer diferencia entre los artefactos UML del negocio frente a los artefactos de sistema informático.
- Detallar el proceso de abstracción y clasificación de casos de uso del sistema informático.
- Desarrollar la etapa de análisis de requerimientos referido al caso práctico "empresa de transportes TECSA".

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector deberá reconocer la importancia del análisis de requerimientos.
- El lector reconocerá la relevancia del modelo de negocio como punto de partida para la ejecución del análisis de requerimientos.
- El leyente diferenciará los diversos artefactos UML del negocio de los artefactos del análisis de requerimientos.
- El leedor debe estar en la capacidad de realizar el análisis de requerimientos basado en el Proceso Unificado para cualquier realidad empresarial.

4.1. INTRODUCCIÓN

Ya estamos preparados para dar inicio al proceso de construcción de software basado en Proceso Unificado.

Iniciamos como es de suponer a conocer la organización problema, utilizando para esta tan importante labor la primera etapa del Proceso Unificado, ¿Lo conoces?, si la respuesta es negativa recomiendo revisar el capítulo 3, donde se estudio el corazón del Proceso Unificado.

Para las personas que si prestaron atención a los capítulos anteriores, es tiempo de poner manos a la obra, pero, ¿en base a que caso?, el proceso de construcción estará centrado en el caso empresa de transportes "*TECSA*", a detallarse en el capítulo 4.2.

Sin más preámbulos comenzamos con la etapa "*MODELO DE NEGOCIO*", esta etapa implica 2 sub – etapas:

- *MODELO DE CASOS DE USO DE NEGOCIO Y*
- *MODELO DE ANÁLISIS DEL NEGOCIO*

Utilizaremos el case de modelado Rational Rose 2005, para la presente edición del libro. Con respecto a la versión anterior del Rational Rose (2005), la última versión presenta cambios *significativos*, en el ambiente de *NEGOCIO*. Dedicaremos el tiempo necesario para detallar cuáles son aquellos elementos y notaciones que fueron modificados ó ¿cuáles son las novedades?, como siempre el case de modelado Rational Rose ¡SORPRENDE!

4.2. CASO PRÁCTICO EMPRESA DE TRANSPORTE "*TECSA*"

La empresa de transportes "*TECSA*" tiene varias sucursales ubicados en diversos puntos de la capital y en ciudades ubicados en la zona centro y sur del país. La empresa emplea aproximadamente 500 autobuses de última generación y 1000 conductores. El Departamento de Contabilidad reportó recientemente que la cobertura de seguros para "*TECSA*" está aumentando a razón del 35 por ciento más rápido en comparación con las empresas de transportes similares en toda la nación. Adicionalmente, los ingresos se han quedado atrás en un 22 por ciento con respecto a los pronósticos de la empresa como resultado de menores tarifas y mejora en el servicio realizado por las demás empresas. Los miembros del directorio de TECSA creen que parte de este desempeño desfavorable se debe a una política poco agresiva en la evaluación del desempeño y control de los conductores.

En consecuencia, el gerente de TECSA ha solicitado que se diseñe un sistema informático que evalúe aquellos aspectos del desempeño del conductor que están relacionados más de cerca con las primas de seguros de los autobuses.

Usted, como profesional en construcción de sistemas informáticos de "TECSA", deberá realizar el análisis y diseño orientado a objetos para la construcción de un sistema informático que permita el control del desempeño de los conductores de la empresa; para tal cometido la empresa cuenta con las siguientes estadísticas: Las multas de tránsito han alcanzado el promedio de casi 2800 anualmente durante los tres últimos años. Los autobuses se ven involucrados en un promedio de casi 57 accidentes a la

semana, 45 de los cuales son considerados como golpes menores y 12 de los cuales implican alguna demanda por lesiones personales.

El personal de servicio a clientes maneja entre 25 y 50 quejas de clientes por tiempos de espera largos, lenguaje abusivo, servicio eficiente, etc., de forma diaria. Cada queja reportada, accidente y multa de tráfico se registra por día, conductor, número de autobús y hora del día.

4.3. CONCEPTO DEL MODELO DE NEGOCIO

El ***modelado de negocio*** basado en el Proceso Unificado, permite realizar un estudio exhaustivo de la organización en términos de Procesos de Negocio.

Gracias al ***modelado de negocio***, podemos empezar el desarrollo del Sistema de Información con información certera y de primera mano, pudiendo lograr así la construcción de un Sistema de Información de Calidad.

El ***modelado de negocio***, puede existir en cualquier organización, aun cuando NO cuente con un Sistema de Información.

El resultado del ***modelado de negocio***, es la "ENTRADA" para el Modelo de Desarrollo de Software.

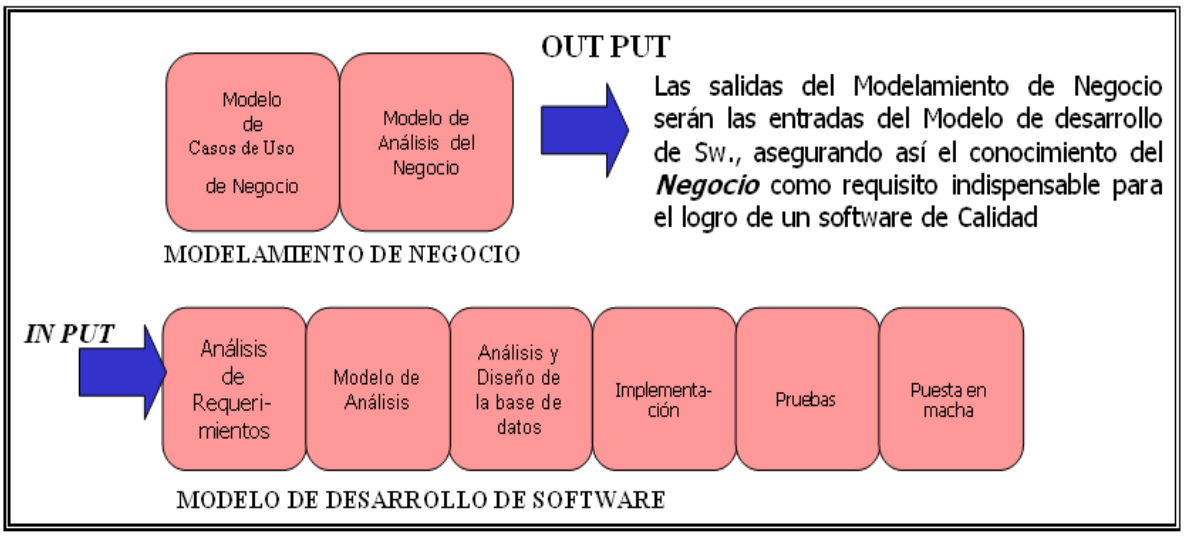


Figura N° 86, Ciclo de desarrollo de software basado en el Proceso Unificado.

4.4. PROPÓSITOS DEL MODELO DE NEGOCIO

- Entender los problemas actuales de la organización.
- Asegurar que clientes, usuarios, equipo de desarrollo y otros involucrados tengan igual entendimiento de la organización.
- Un modelo de negocio provee una vista estática de la estructura de la organización y una vista dinámica dentro de los procesos de la organización.

4.5. ELEMENTOS DE NEGOCIO SEGÚN EL PROCESO UNIFICADO

4.5.1. GLOSARIO DE NEGOCIO

Es de vital importancia acordar la terminología de negocio común desde la definición del proyecto, para lograr estándares y agilidad en la comunicación.

Ejemplo:

"Para que un empleado obtenga los útiles de oficina, mensualmente, tiene que presentar el documento PECOSA"

4.5.2. PARTES DEL GLOSARIO DE NEGOCIO

No solo los documentos son parte de glosario, algunos procesos según el grado de relevancia también son considerados.

El PROCESO UNIFICADO propone la siguiente estructura de descripción del proceso.

- Introducción
- Propósito
- Alcance
- Referencias
- Resumen
- Definiciones

4.5.3. REGLA DEL NEGOCIO

Políticas ó condiciones empresariales a ser respetadas y satisfechas en el modelo de negocio que se realiza como parte del proceso de construcción del sistema informático.

"No se realizará ningún pago sin documento de sustento"
"No se admite como empleado a una persona cuya documentación sea incompleta"

Figura N° 87, Notación UML para la Regla de Negocio.

4.5.4. PARTES DEL DOCUMENTO REGLAS DEL NEGOCIO

- Introducción.
- Propósito.
- Alcance
- Referencias
- Resumen
- Reglas del negocio.

4.5.5. META U OBJETIVO

Es el requisito a ser satisfecho por el negocio, detalla el valor deseado de una medida específica en el futuro, utilizado para planear y administrar las actividades del negocio.

Ejemplo: "Eliminar las tardanzas e inasistencias a diciembre del año 2008".

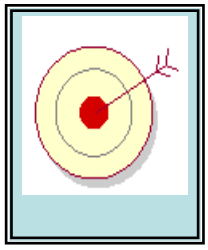


Figura N° 88, Notación UML para la Meta u Objetivo empresarial.

4.5.6. UNIDAD ORGANIZACIONAL

En esencia, es similar al paquete, sirve para organizar los artefactos que permitan explicar los procesos empresariales que se analizan, en términos de actores y casos de uso de negocio.

Ejemplo: Sirve para organizar los modelos de casos de uso de negocio referido a un proceso de nivel macro como:

Ventas, Compras, Control de Personal, etc.

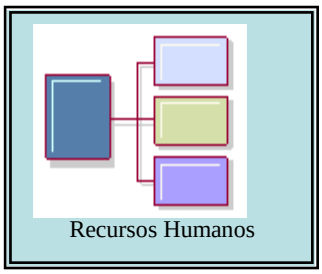


Figura N° 89, Notación UML para una Unidad Organizacional.

4.5.7. CASO DE USO DE NEGOCIO

Representa a un proceso empresarial, aquel conjunto de actividades continuas, necesarias para la existencia de la organización. Los casos de uso de negocio empiezan su definición en verbo.

Ejemplo: Generar Pedido, Generar Orden de Compra, Generar Factura, Generar Boleta, Registrar Personas, etc., ver figura N° 89.

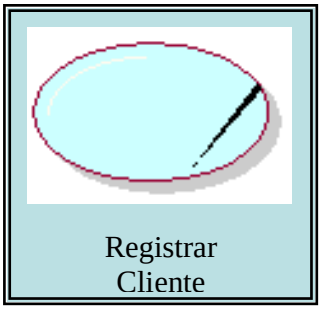


Figura N° 90, Notación UML del Caso de uso de Negocio.

4.5.8. TRABAJADOR DEL NEGOCIO (WORKER)

Conocido también como ***actor interno de negocio***, representa a una persona ó un grupo de personas que tienen relación ***directa*** con el proceso empresarial, su definición depende al caso de uso de negocio que se este analizando.

Ejemplo: Cajero, considerando el proceso "Generar Factura

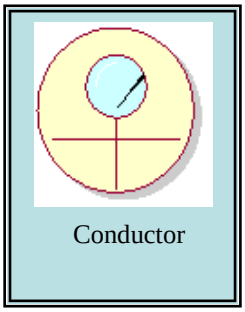


Figura N° 91, Notación UML de un Trabajador del Negocio.

4.5.9. ACTOR DEL NEGOCIO

Representa a una persona ó un grupo de personas que tenga relación ***indirecta*** con el proceso empresarial ó caso de uso de negocio.

La definición del actor externo de negocio depende del caso de uso de negocio que se esté analizando. Ejemplo Proveedor, si consideramos el proceso "Solicitar/Registrar Proforma".

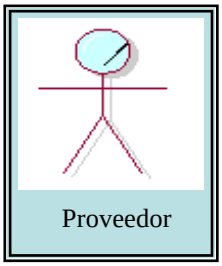


Figura N° 92, Notación UML de un Actor Externo de Negocio.

4.5.10 ENTIDAD DE NEGOCIO

Representa a un documento ó cualquier elemento de información que es usado ó manipulado por un trabajador interno de negocio.

Por ejemplo: En el caso de uso de negocio "Registrar Instructor", registramos los datos del instructor en algún archivo, file, folder ó base de datos, cada uno de los esos elementos donde se almacenan la información del nuevo conductor se denomina entidad de negocio.

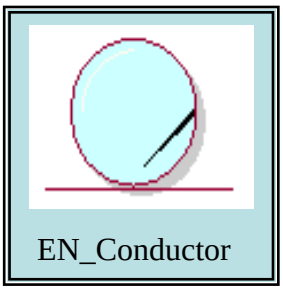


Figura N° 93, Notación UML de una Entidad de Negocio.

4.5.11. REALIZACIÓN DEL CASO DE USO DE NEGOCIO

Sirve como repositorio de todos los artefactos, que tienen como objetivo explicar el funcionamiento al detalle del proceso empresarial que se analiza incluyendo la explicación de los documentos que se utilizan ó generan.

Ejemplo: Realización del caso de uso Registrar Conductor.

Figura N° 94, Notación UML, del caso de uso realización de negocio.

4.5.12. RECURSO

Este elemento representa al recurso de la organización, los recursos y roles actúan de manera conjunta para realización del sistema de negocio.

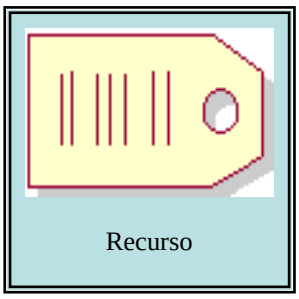


Figura N° 95, Notación UML, del elemento Recurso.

4.5.13. MODELO DE CASOS DE USO DE NEGOCIO

Este elemento representa la Modelo de Casos uso de Negocio.

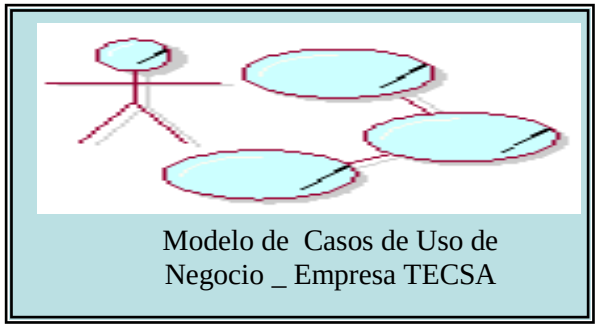


Figura N° 96, Notación UML Modelo de Casos de Uso de Negocio.

4.5.14. DOMINIO DEL NEGOCIO

Este elemento representa al campo de acción ó, ó giro de la organización.

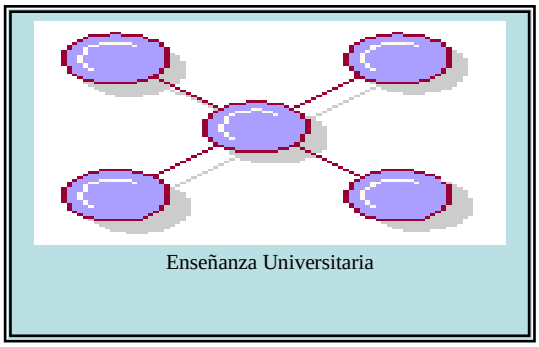


Figura N° 97, Notación UML Dominio del Negocio.

4.5.15 MODELO DE ANALISIS DE NEGOCIO

Este elemento representa a la realización del caso de uso de negocio, a través de artefactos del UML, como diagramas de clases, secuencia y colaboración, detallando la funcionalidad del proceso que se analiza.

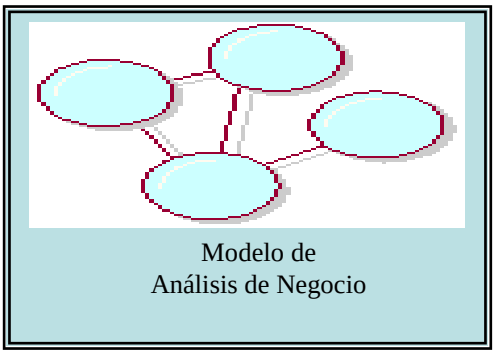


Figura N° 98, Notación UML Modelo de Análisis de Negocio.

4.5.16 TRABAJADOR FISICO

Este elemento representa a una persona ó grupo de personas, que laboran dentro de la organización ocupando puestos claves para la organización.

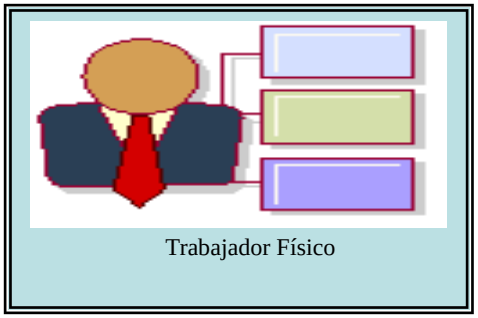


Figura N° 99, Notación UML del Trabajador Físico.

4.5.17. RECURSOS COLABORATIVOS

Este elemento representa al grupo de recursos empresariales, cuya iteración ó relación es necesaria para el éxito de un determinado proceso empresarial.

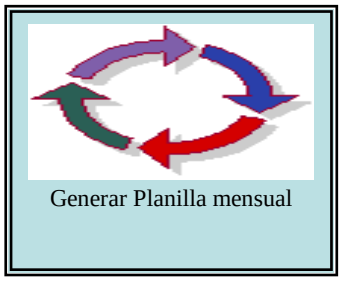


Figura N° 100 Notación UML del Recurso Colaborativo.

4.5.18. SISTEMA DE NEGOCIO

Este elemento representa a unidades empresariales individuales, este encapsula un conjunto de roles y recursos, para el cumplimiento de un propósito en particular, además define un conjunto de responsabilidades mediante los cuales, los propósitos pueden ser alcanzados.

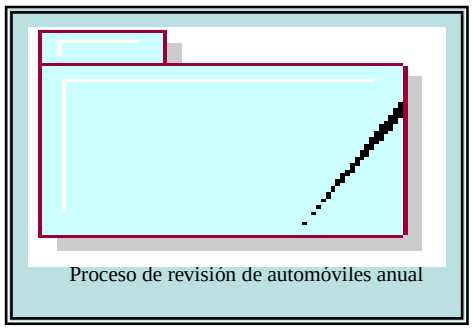


Figura N° 101, Notación UML del Sistema de Negocio.

4.5.19. COMPONENTE DE NEGOCIO

Este elemento representa a un elemento de negocio con código correspondiente a cualquier sistema parte de la organización, desde el inicio del estudio empresarial para la mejora en términos de procesos, implementación de un nuevo sistema de información ó la mejora de alguno existente.

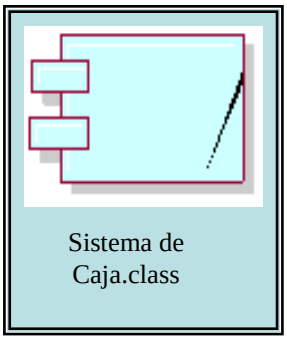


Figura N° 102, Notación UML de un Componente de Negocio.

4.5.20. LOCALIZACIÓN DEL NEGOCIO

Este elemento representa a la ubicación geográfica, la más estratégica para efectos de mercadeo de la organización.

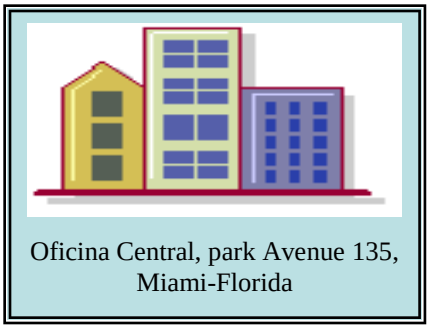


Figura N° 103, Notación UML de la localización física del negocio.

4.5.21. DISEÑADOR DE NEGOCIO

Este elemento es responsable de detallar los eventos comerciales, usándolos para descomponer espacio y tiempo dentro del proceso empresarial que se analiza.

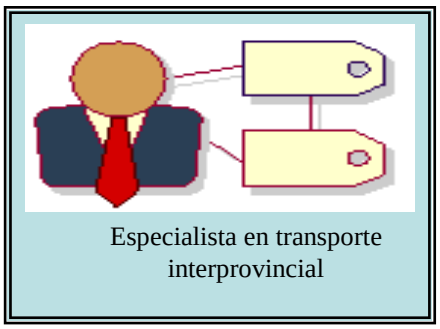


Figura N° 104, Notación UML del Diseñador de Negocio.

4.5.22. EVENTO DE NEGOCIO

Representa al conjunto de sucesos, acciones empresariales que repercuten directamente al proceso que se analiza.

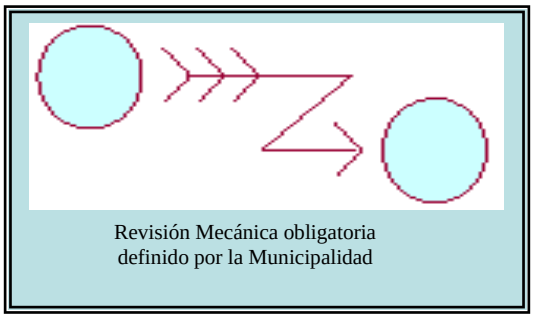


Figura N° 105, Notación UML de un Evento de Negocio.

4.5.23. DOCUMENTO DE NEGOCIO

Representa al documento formal, utilizado para garantizar la funcionalidad del un proceso en particular con referencia a organizaciones supervisoras del buen funcionamiento de la misma.

Ejemplo el documento formal "*Contrato de Trabajo*", entidad supervisora "*Ministerio de Trabajo*".

Figura N° 106, Notación UML del documento de negocio.

4.6 DETERMINACIÓN DE LA SITUACIÓN ACTUAL DE LA ORGANIZACIÓN

Elaborar un listado de términos y definiciones usados comúnmente, en un Glosario de Términos.

Consiste en desarrollar un entendimiento preliminar de los objetivos de la empresa, los cuales son determinados por los stakeholders y responsables del negocio.

Identificar las reglas del negocio, para luego plasmarlos en el documento de Reglas del Negocio. Involucrar a las personas con más experiencia y conocimiento en la organización de la siguiente manera:

- Convertirlos en miembros del equipo de modelado de negocio.
- Entrevistarlos para conocer sus ideas y opiniones basadas en sus experiencias.
- Hacer que revisen nuestros avances.

4.7. MODELO DE CASOS DE USO DE NEGOCIO

4.7.1. CONCEPTO

Este modelo, muestra la relación existente entre un Caso de Uso de Negocio con los diferentes actores de negocio, se realiza en el entorno de trabajo del *diagrama de casos de uso*.

4.7.2. TIPOS DE RELACIONES EN EL MODELO DE CASOS DE USO DE NEGOCIO.

En el ambiente de casos de uso de negocio identificamos los siguientes tipos:

4.7.2.1. RELACION DEL TIPO ASOCIACION UNIDIRECCIONAL

En el modelo de caso de uso de negocio, esta relación indica ***participación***.

4.7.2.2. RELACION DEL TIPO HERENCIA

Este tipo de relación indica que las clases que participan en el modelo de casos de uso de negocio pueden utilizar la característica de *generalización* ó *herencia*.

4.7.3. TIPOS DE ESTERIOTIPOS EN LAS RELACIONES DE MODELOS DE CASOS DE USO DEL NEGOCIO

En el modelo de casos de uso de negocio podemos identificar a los siguientes estereotipos:

4.7.3.1. <<REALIZE>>

El estereotipo <<realize>>, brinda el comportamiento a la relación existente entre un caso de uso ya sea de negocio ó sistema con su respectivo caso de uso de realización.

4.7.3.2. <<IMPORT>>

El estereotipo <<import>>, brinda el comportamiento a la relación existente entre los siguientes artefactos:

- Modelo de casos de uso de negocio y
- Modelo de análisis

4.7.3.3. <<SUPPORT>>

El estereotipo <<support>>, brinda el comportamiento a la relación existente entre artefactos de negocio indicando apoyo ó soporte de acción.

4.7.4. DESARROLLANDO EL MODELO DE CASOS DE USO DE NEGOCIO DEL CASO "TECSA"

4.7.4.1. DEFINICION DE LOS OBJETIVOS DE NEGOCIO

En el paquete "Bussiness Use Case", cargado por defecto al hacer clic en la plantilla "Rational Unified Process" (ver figura N° 82, de la Vista de Casos de Uso del Case Rational Rose, definir los objetivos de negocio, puede crearse dentro de un paquete

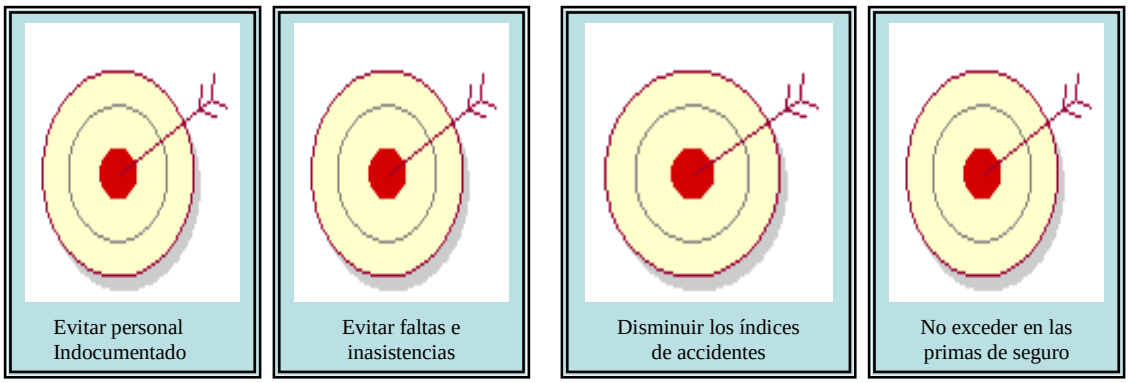


Figura N° 107, Algunos objetivos a cumplir en el caso Empresa de Transportes "TECSA".

4.7.4.2. DEFINICION DE LAS UNIDADES ORGANIZACIONALES

Para el presente caso identificamos las siguientes unidades organizacionales, para más detalle ver la figura n° 108.

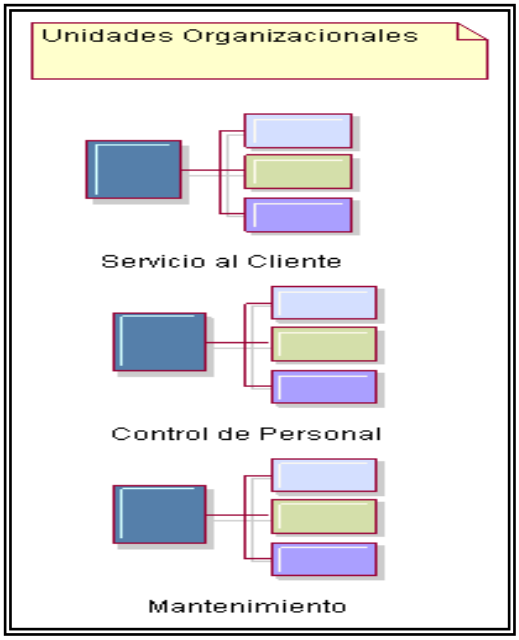


Figura N° 108, Definición de las Unidades Organizacionales con respecto al caso Empresa de Transportes "TECSA".

4.7.4.3. DOCUMENTACIÓN DE LAS UNIDADES ORGANIZACIONALES

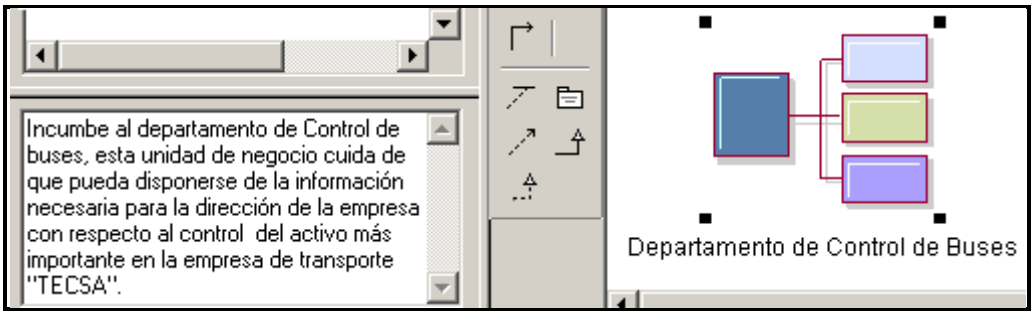


Figura N° 109, Documentación de la unidad organizacional "Control de Buses".

4.7.4.4. DEFINICIÓN DE LOS ACTORES DE NEGOCIO

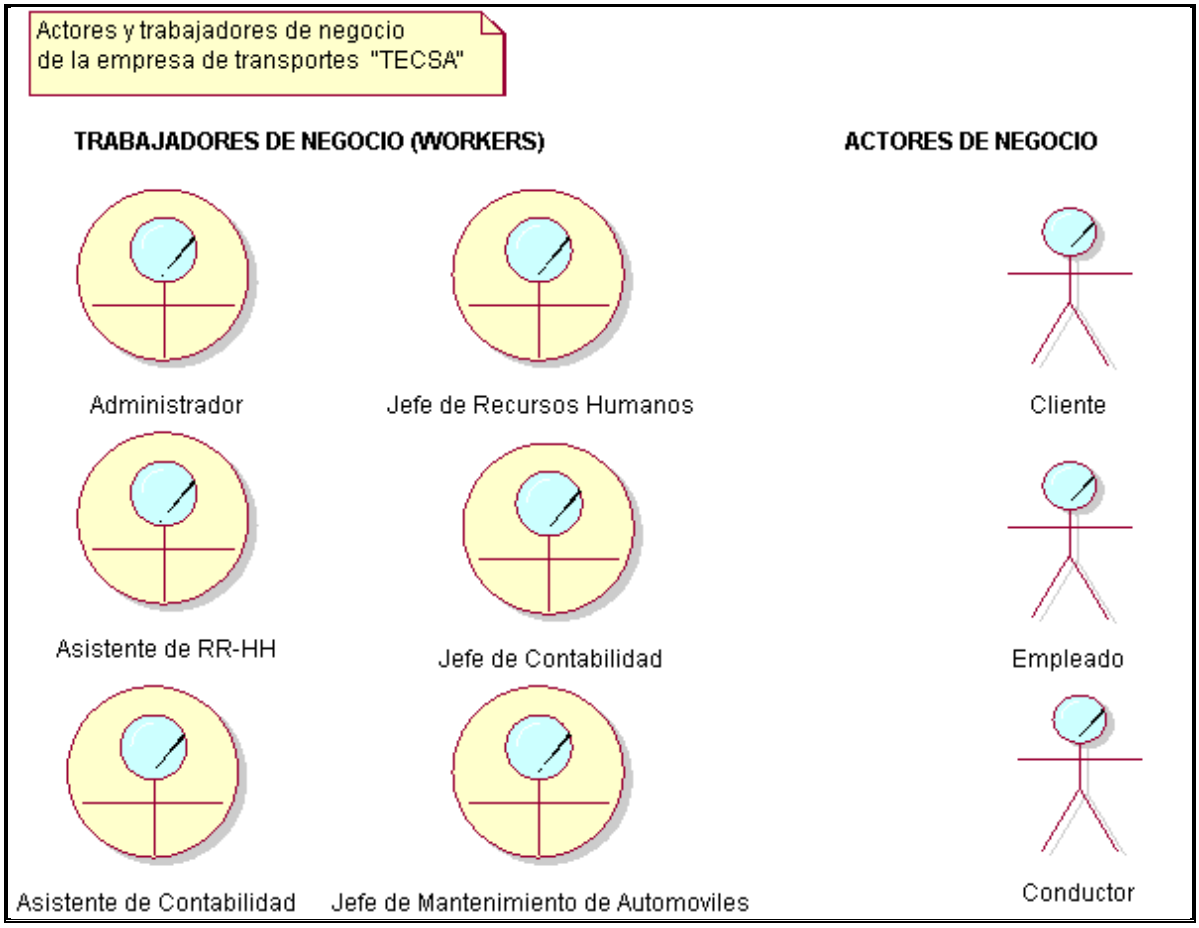


Figura N° 110, Creación de los actores de negocio del caso Empresa de Transportes "TECSA".

En el paquete "Business Use-Case Model", crear un diagrama de casos de uso denominado "Actores de Negocio_TECSA"; en el evento doble clic al diagrama de casos de uso creado, se apertura el entorno de trabajo correspondiente, ahora si podemos comenzar con la creación de los actores internos y externos de negocio.

La idea es tener un repositorio de creación para elementos similares, logrando su fácil ubicación. En el análisis de la empresa "ABC" por ejemplo, se pueden encontrar más de 50 elementos, la idea es encontrarlos en el ambiente preestablecido.

4.7.4.5. DESCRIPCION DE LOS ACTORES Y TRABAJADORES DEL NEGOCIO

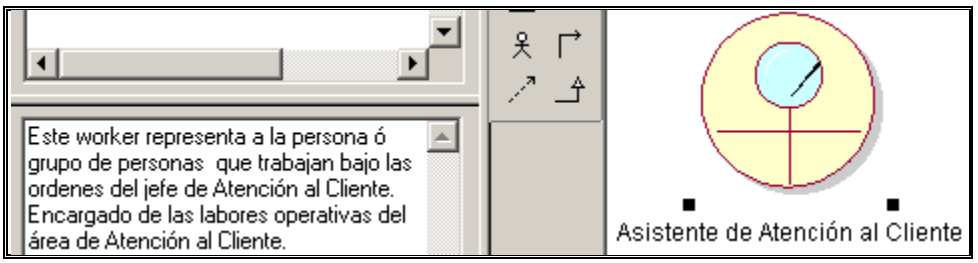


Figura N° 111, Documentación del actor interno de negocio "Asistente de atención al cliente".

El talón de Aquiles de la mayoría de los desarrolladores de software es por la poca importancia dada al proceso de documentación en la construcción del software, se preocupan de la documentación sólo cuando este, YA es un problema, ¡la idea es evitar que el riesgo se convierta en problema!

4.7.4.6. DEFINICION DE LOS CASOS DE USO DE NEGOCIO

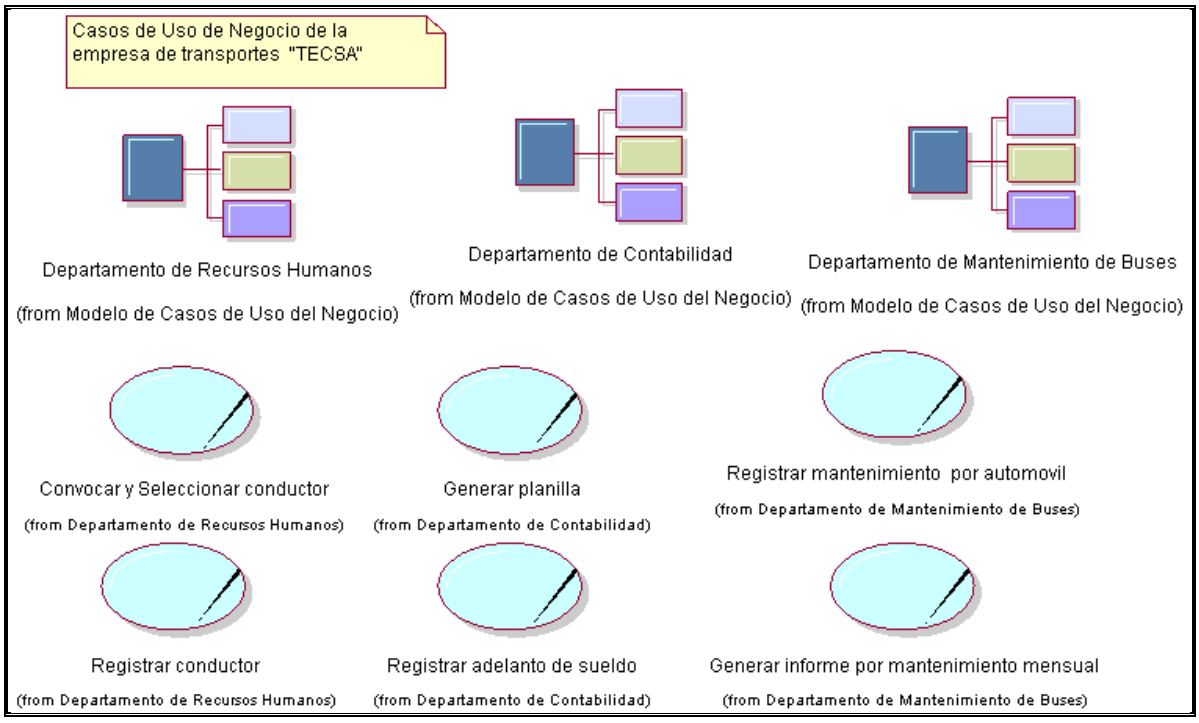


Figura N° 112, Creación de los casos de uso de negocio del caso Empresa de Transportes "TECSA".

4.7.4.7. DESCRIPCION DE LOS CASOS DE USO DE NEGOCIO

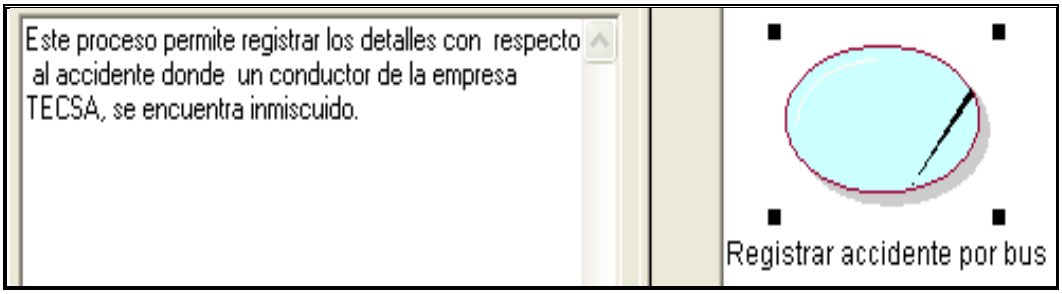


Figura N° 113, Documentación del caso de uso de negocio "Registrar accidente por bus".

4.7.4.8. MODELO DE CASOS DE USO DE NEGOCIO

No olvidemos que el Modelo de Casos de Uso de Negocio, muestra la participación de los trabajadores y actores del negocio con un proceso de negocio en particular. ¿Dónde se crea?, no desespere que a continuación lo menciono.

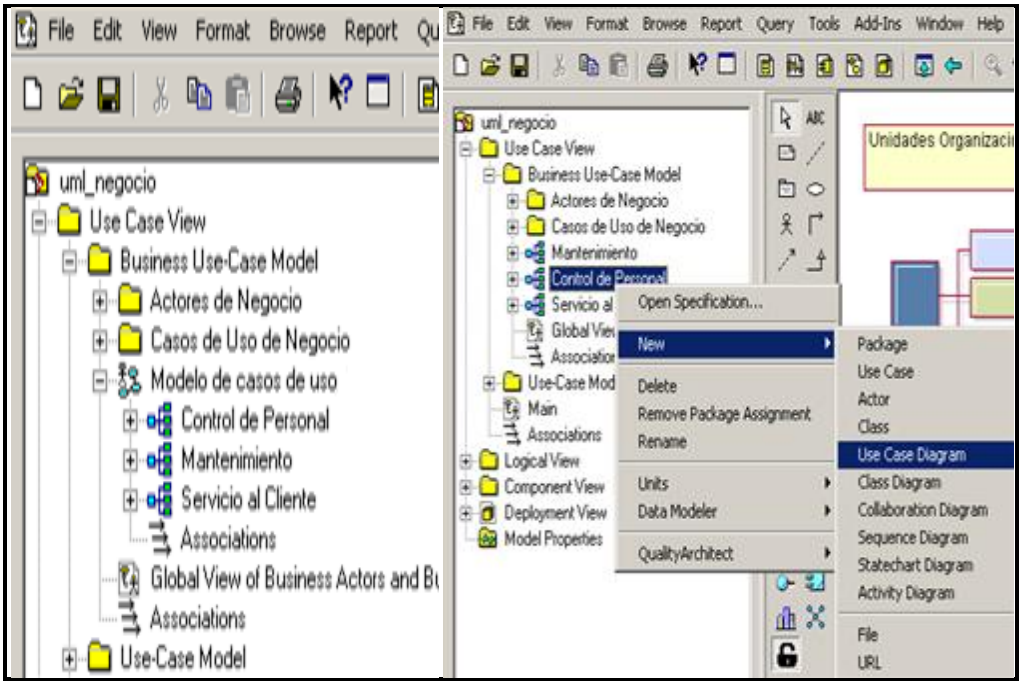


Figura N° 114, Creando un diagrama de casos de uso, para los Modelos de Casos de Uso de Negocio.

Ahora si podemos crear el tan esperado Modelo de Casos de Uso de Negocio para el proceso "***REGISTRAR CONDUCTOR***".

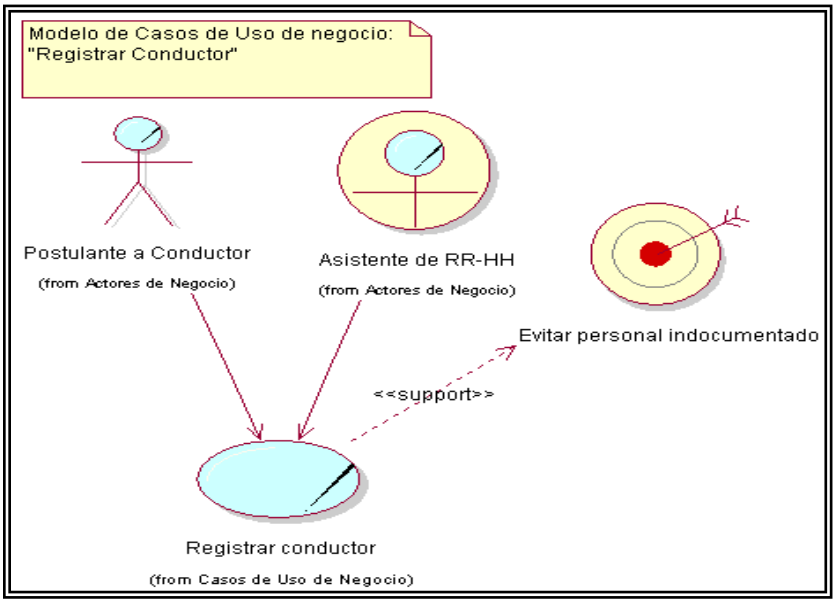


Figura N° 115, Modelo de Casos de Uso de Negocio "Registrar conductor".

Menciono otro ejemplo para mayor ilustración; pero ¿sólo dos?, NO se preocupe, el proceso íntegro, lo detallo en el CD, que acompaña a la presente. ¡***REVISELO***!.

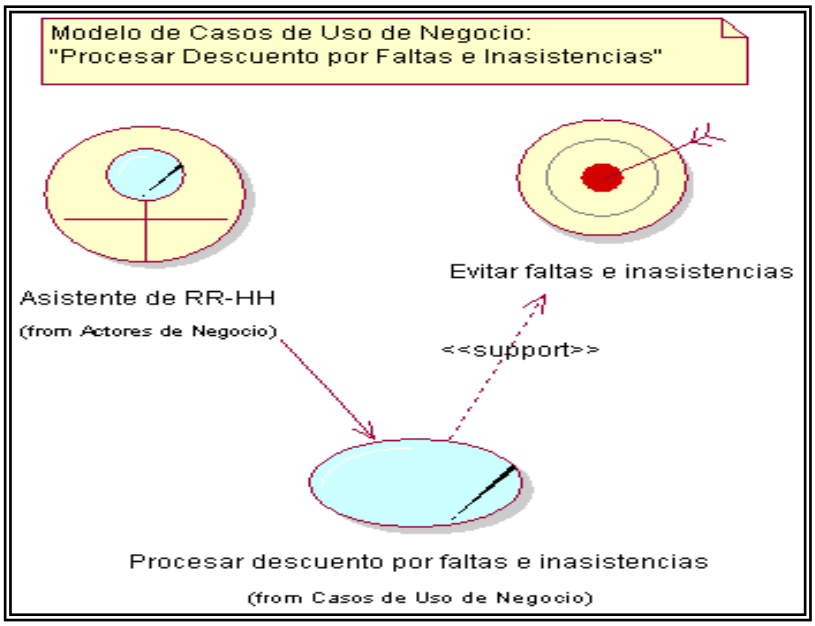


Figura N° 116, Modelo de Casos de Uso de Negocio "Procesar descuento por faltas e inasistencias".

4.8. MODELO DE ANÀLISIS DEL NEGOCIO

4.8.1. CONCEPTO

Este modelo, muestra el ***detalle*** del caso de uso de negocio que se está analizando, como se realiza o desarrolla el caso de uso en mención, para tal cometido el PROCESO UNIFICADO, indica el uso de los siguientes artefactos propios del UML:

- Diagrama de Casos de Uso, para la realización
- Diagrama de Clases
- Diagrama de Secuencia
- Diagrama de Colaboración

4.8.2. TIPOS DE RELACIONES EN EL MODELO DE OBJETOS DE NEGOCIO

4.8.2.1. ASOCIACION BIDIRECCIONAL

Este tipo de asociación contiene a las denominadas "Reglas del negocio".

4.8.2.2. OBJECT LINK

Este tipo de relación presente en el diagrama de secuencia, permite la definición de un mensaje entre dos objetos.

4.8.2.3. OBJECT LINK TO SELF

Muestra la ejecución de un mensaje desde el mismo objeto, presente sólo en diagramas de secuencia.

4.8.2.4. OBJECT MESSAGE

Este tipo de relación presente en el diagrama de colaboración, permite la definición del mensaje entre dos objetos.

4.8.2.5. OBJECT MESSAGE TO SELF

Permite la definición de un mensaje entre el mismo objeto, presente sólo en diagramas de colaboración.

4.8.3. DIAGRAMA DE DEPENDENCIAS ENTRE EL MODELO DE CASOS DE USO DE NEGOCIO Y EL MODELO DE ANALISIS O MODELO DE ANÁLISIS DE NEGOCIO

La idea de este diagrama es demostrar la dependencia entre del Modelo de Análisis ó Modelo de análisis de negocio con respecto al Modelo de Casos de Uso de Negocio.

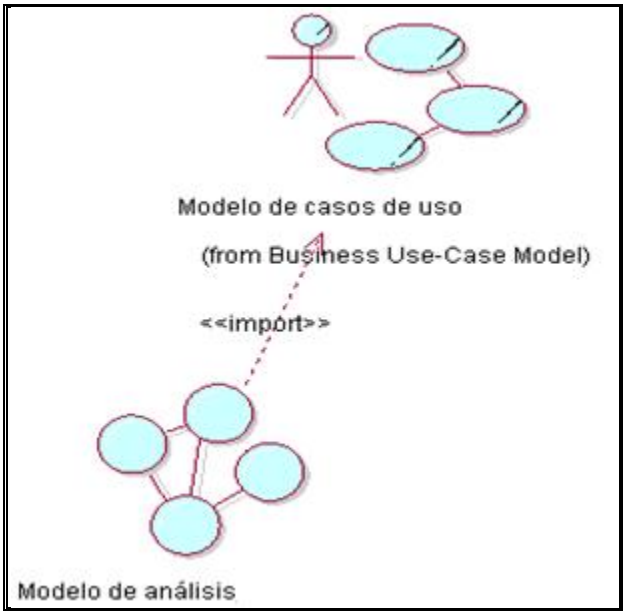
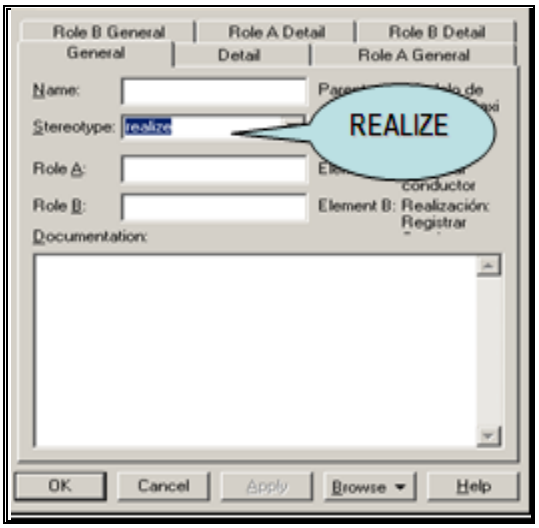
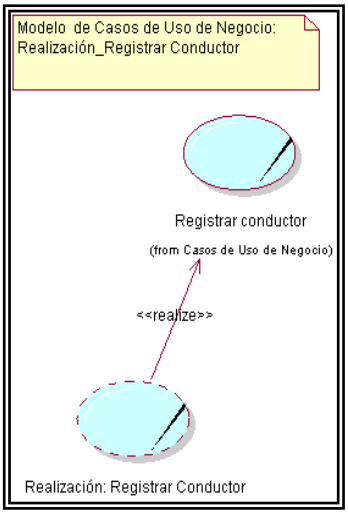


Figura N° 117, Dependencia del Modelo de Análisis con respecto al Modelo de Casos de Uso, referido al caso Empresa de Transporte "TECSA".

4.8.4. PROCESO DE REALIZACION DEL CASO DE USO DE NEGOCIO





El caso de uso realización de negocio, permite explicar al detalle como se realiza un determinado caso de uso de negocio, utilizando artefactos como diagrama de clases, secuencia y colaboración.

Figura N° 118, Diagrama de casos de uso, mostrando la realización del proceso "Registrar Conductor".

El comportamiento de la relación está dado por el estereotipo <<realize>>, se puede activar haciendo doble click en la relación y seleccionado la opción deseada.

Podemos crear un "Sistema de Negocio", es un paquete que ayuda a organizar los artefactos para la realización de un proceso empresarial. Para nuestro caso se denomina "*Registrar Conductor*", contiene al diagrama de casos de uso, contenedor del modelo de realización, al caso de uso "Realización: Registrar Conductor" y sus diagramas correspondientes.

Figura N° 119, Distribución del browser del case de modelado Rational Rose, en el ambiente de Modelo de Análisis de Negocio.

4.8.4.1. DIAGRAMA DE CLASES

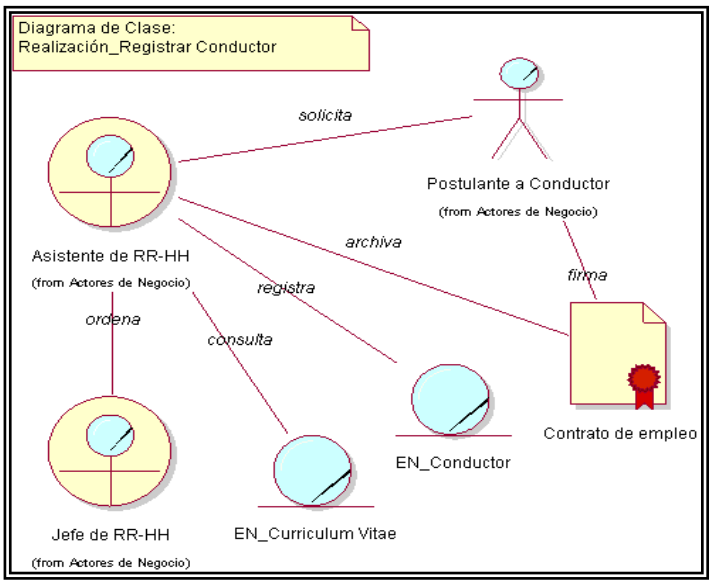


Figura N° 120, Diagrama de Clases del proceso "Registrar Conductor", con respecto al caso Empresa de Transporte "TECSA".

No olvidemos que el Diagrama de Clases, es una estructura estática, muestra las clases y sus relaciones.

En el negocio utilizamos la ya descrita "Entidad de Negocio", el cual representa a cualquier documento, ficha, archivo, etc.; creado, manipulado por un trabajador interno de negocio.

Como cualquier elemento deberá ser contenido en un repositorio, si se necesita la misma entidad para la realización de otro proceso, puede ser ubicado con mucha facilidad.

4.8.4.1.1. DEFINIENDO UN REPOSITORIO PARA ENTIDADES DE NEGOCIO.

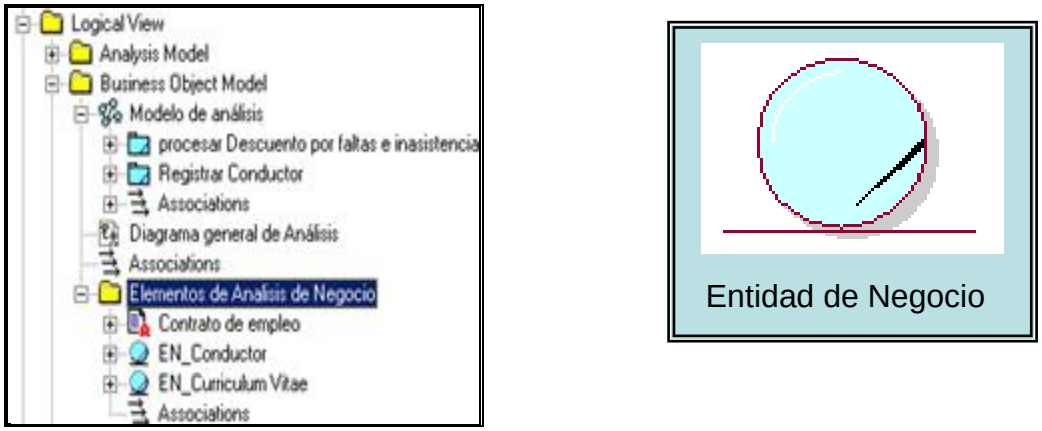


Figura N° 121, Creando un repositorio para las entidades de negocio.

4.8.4.2. DIAGRAMA DE COLABORACIÓN

Sabemos que el diagrama de colaboración es del tipo dinámico e interactivo, muestra como cada uno de los objetos, se comunican mediante una secuencia de mensajes para explicar el detalle de un proceso en particular. No olviden que la distribución es con respecto al espacio.

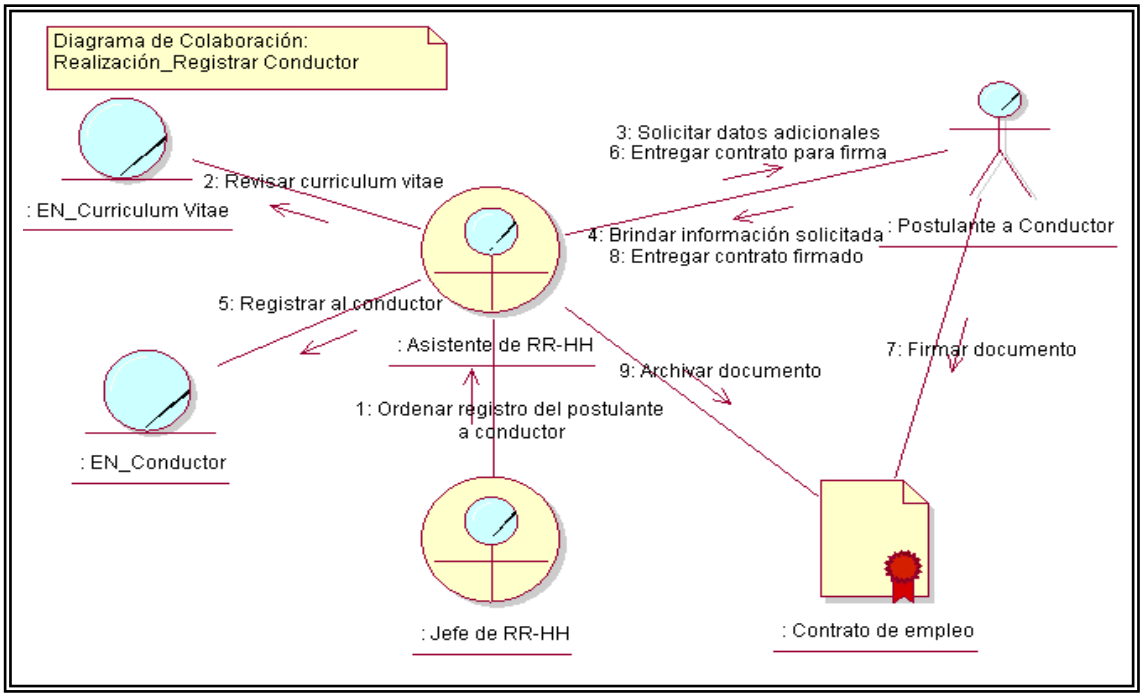


Figura N° 122, Diagrama Colaboración para la realización del proceso "Registrar Conductor".

4.8.4.3. DIAGRAMA DE SECUENCIA

Este diagrama es equivalente al diagrama de secuencia, la diferencia radica en la distribución, el diagrama de secuencia presenta la distribución con respecto al tiempo.

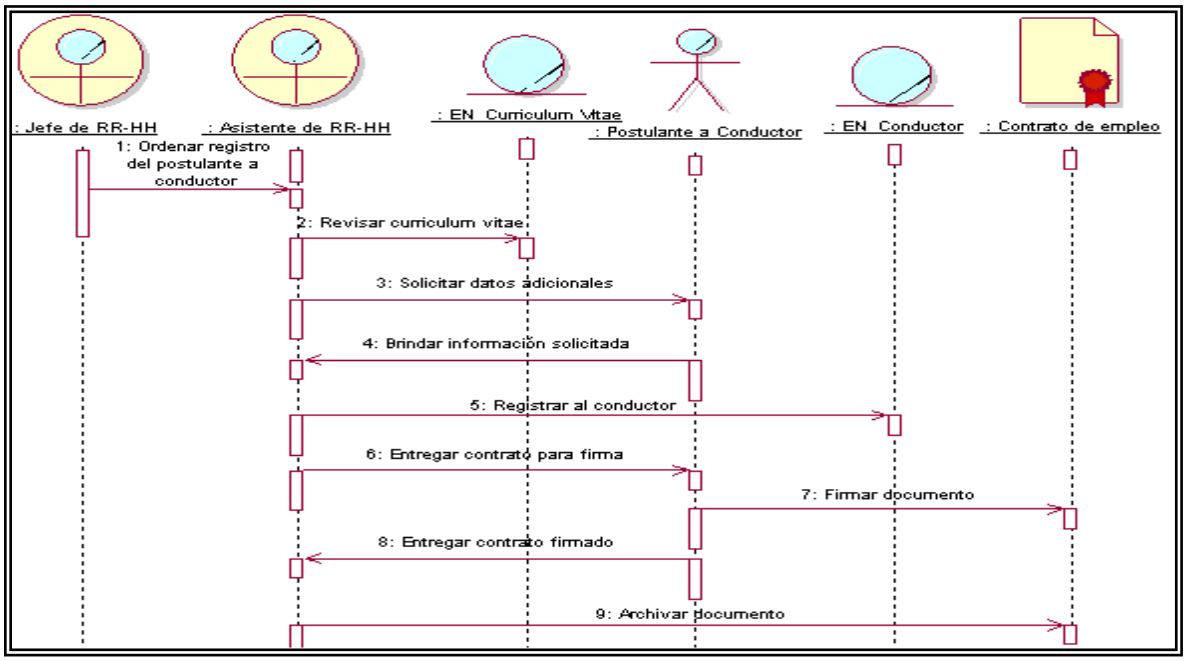


Figura N° 123, Diagrama Secuencia para la realización del proceso "Registrar Conductor".

4.8.4.4. DIAGRAMA DE ACTIVIDADES

Es necesario considerar los diagramas de actividades en el estudio de la organización para resolver los ***"FACTORES DE DESICIÓN"***, los cuales No son considerados por los artefactos de negocio ya que se asume la afirmación.

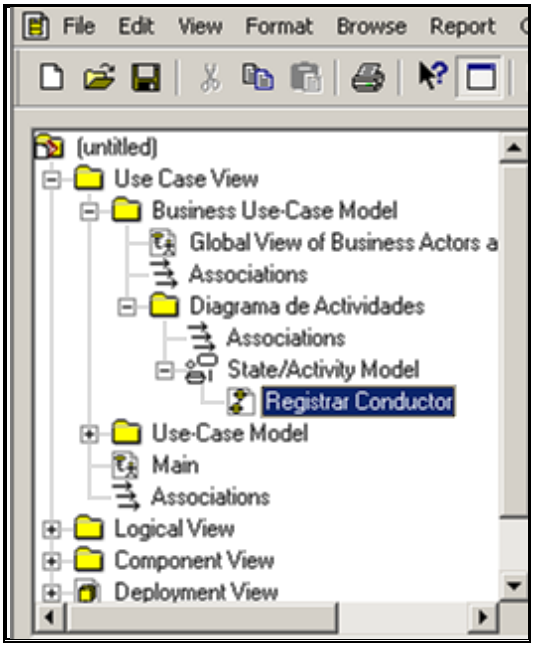


Figura N° 124, Creando el Diagrama de Actividades "Registrar Conductor".

En el paquete Business Object Model crear el paquete Diagrama de Actividades, el cual será repositorio de todos los diagramas de actividades dirigido a negocio, para el caso que desarrollamos, se denomina: "Registrar Conductor".

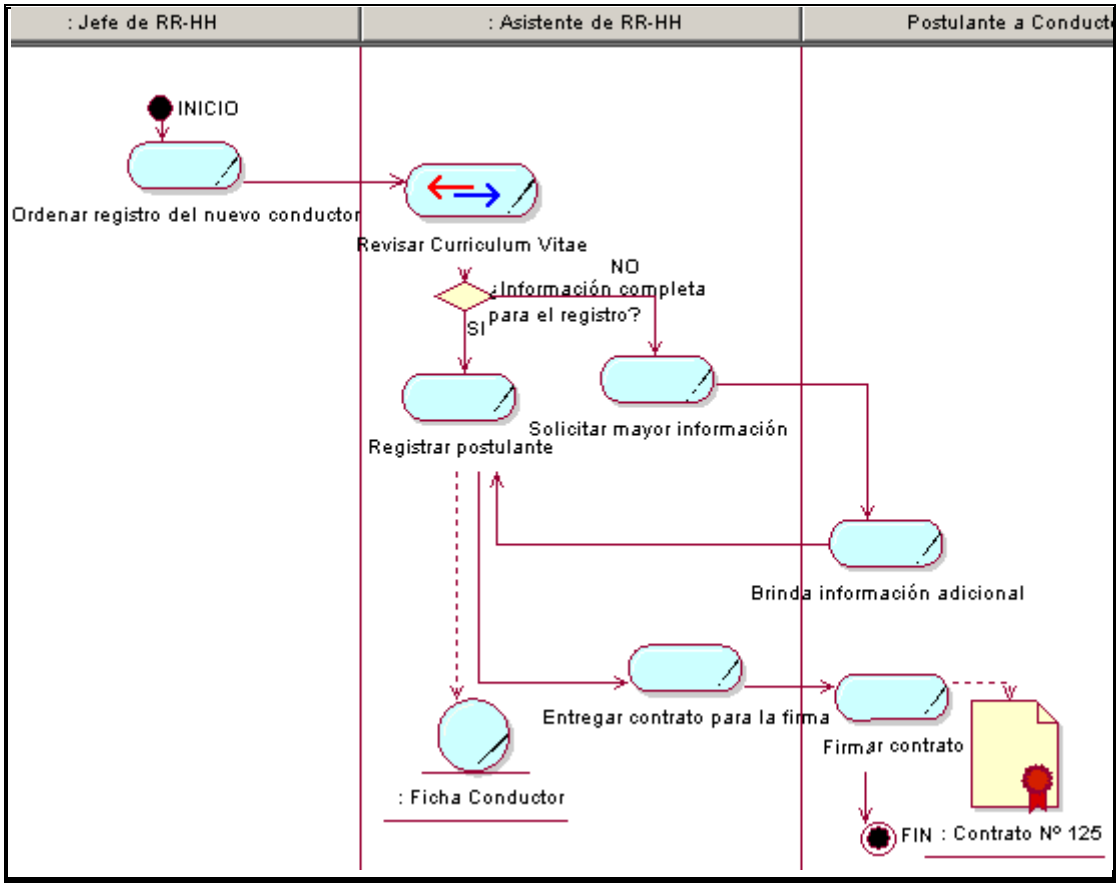


Figura N° 125, Creando el Diagrama de Actividades "Registrar Conductor".

4.9. RESUMEN

El modelo de negocio permite conocer a la organización en un 100%. El proceso unificado brinda especial importancia al análisis empresarial por considerarla crucial para el éxito de la solución informática.

Se analizó el negocio desde un punto de vista genérico y detallado a través de los diversos artefactos proporcionados por el UML.

El procedo unificado establece el perfecto uso de la técnica de modelado UML, especialmente en el análisis organizacional.

4.10. PREGUNTAS DE COMPETENCIA DEL CAPÍTULO

- Explicar la diferencia existente entre el modelo de casos de uso del negocio y el modelo de análisis del negocio.
- Mencionar la importancia de los elementos documento y entidad del negocio.
- Explicar la importancia del artefacto objetivo del negocio.

CAPITULO V

ANÁLISIS DE REQUIRIMIENTOS BASADO EN EL PROCESO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Instituir la importancia del análisis de requerimientos dentro del proceso de construcción.
- Explicar los diversos artefactos UML a utilizar en la etapa de construcción del software.
- Establecer diferencias entre los artefactos UML del negocio frente a los artefactos de sistema informático.
- Detallar el proceso de abstracción y clasificación de casos de uso del sistema informático.
- Desarrollar la etapa de análisis de requerimientos referido al caso práctico "empresa de transportes TECSA".

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector deberá reconocer la importancia del análisis de requerimientos.
- El lector reconocerá la relevancia del modelo de negocio como punto de partida para la ejecución del análisis de requerimientos.
- El leyente diferenciará los diversos artefactos UML del negocio de los artefactos del análisis de requerimientos.
- El leedor debe estar en la capacidad de realizar el análisis de requerimientos basado en el Proceso Unificado para cualquier realidad empresarial.

5.1. INTRODUCCIÓN

El resultado del modelo de negocio permite conocer a la organización en un 100%, esta salida hace posible el inicio del de la segunda gran etapa en el proceso de construcción del software conocido como "modelo de construcción del software". Este modelo guía a los diversos trabajadores del proyecto informático en la construcción del sistema informático desde el análisis de requerimientos hasta la puesta en marcha del sistema.

La etapa de análisis de requerimientos permite el análisis de las funciones y/o requerimientos de todos aquellos elementos que interactúan con la solución informática.

Esta etapa formal del Proceso Unificado permite satisfacer las necesidades exactas del usuario "ON DEMAND"[9].

El análisis de requerimientos marca el inicio de la construcción técnica formal del sistema informático basado en el Proceso Unificado, donde los diversos artefactos UML que se utilizan en esta etapa marcan un modelo claro, estándar y preciso, como punto de partida para la construcción de la solución informática.

5.2. IMPORTANCIA

Los profesionales que asumen roles para elaborar el análisis de requerimientos del sistema informático, asumen uno de los retos trascendentes para el éxito del sistema informático el

[9] On Demand: Este párrafo califica adecuadamente a los usuarios que demandan, aquellos quienes reconocen y exigen calidad en los sistemas informáticos, requieren de sistemas informáticos construidos a la medida de sus necesidades.

de TRADUCIR en términos de unidades y fragmentos funcionales los requerimientos y necesidades de la organización mencionados y expuestos por los trabajadores y expertos de negocio de la empresa de forma coloquial y poco técnica.

El análisis de requerimientos permite el diseño de la presentación final del sistema informático, basado en las necesidades exactas del usuario, para garantizar que la funcionalidad del sistema final coincida con las todas las etapas previas del modelo de construcción del software, lo cual es un requisito del Proceso Unificado, ya este esta basado en el modelo iterativo e incremental, pasarán todas las iteraciones necesarias para garantizar un modelo completamente homogéneo.

5.3. ELEMENTOS UML DEL ANÁLISIS DE REQUERIMIENTOS

5.3.1. ACTOR DEL SISTEMA INFORMÁTICO

Esta notación UML, representa a una persona, conjunto de personas, hardware, software y a cualquier componente que interactúa con la solución informática basado en una necesidad.

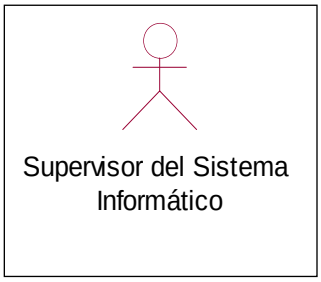


Figura N° 126, Notación UML, representa a un actor del sistem

5.3.2. CASO DE USO DEL SISTEMA INFORMÁTICO

Notación UML que representa a un fragmento o unidad funcional del sistema informático.
El caso de uso mencionado en las etapas de análisis del Proceso Unificado deberá ser considerado en la solución informática final como elemento que muestra respuesta (acción) frente a un requerimiento o solicitud.

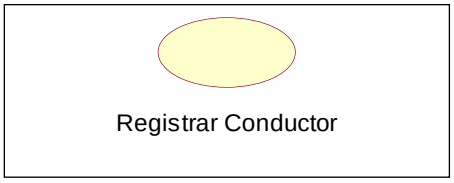


Figura N° 127, Notación UML, representa a un caso de uso del sistema

5.4. ETAPAS DEL ANÁLISIS DE REQUERIMIENTOS

Para garantizar la formalidad y perfecta realización de los diversos casos de uso del sistema informático en base a las necesidades exactas de los usuarios, recomiendo trabajar en base a la siguiente estructura de desarrollo; los cuales serán mencionados en orden de ocurrencia y por grado de dificultad.

5.4.1. IDENTIFICACION DE ACTORES DEL SISTEMA INFORMÁTICO

Esta etapa permite determinar a los diversos actores del sistema informático, el punto discriminador es el nivel de interacción con la solución informática.

Los actores serán considerados por representar un conjunto de acciones en el negocio. Si 2 personas en la empresa tienen un conjunto de acciones idénticas deberán ser representados por un solo actor genérico; pero si 2 personas tienen un conjunto de funciones idénticos y solo se diferencian por un o unos pocos requerimientos deberán ser representados por 2 actores diferentes.

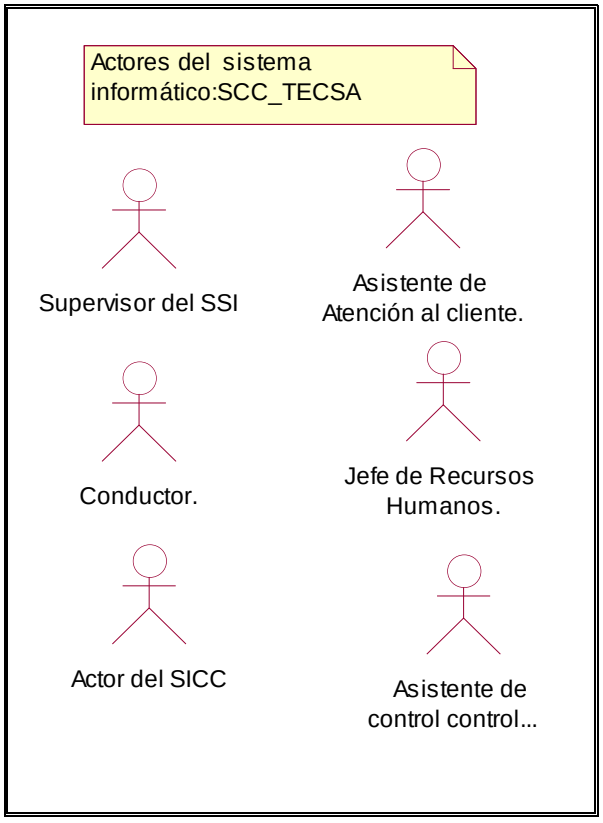


Figura N° 128, Esta figura muestra a los diversos actores del caso empresa de transporte "TECSA"

5.4.2. MODELO DE ESCENARIOS POR ACTOR

El modelo de escenarios muestra en un entorno claro y literal los diversos requerimientos y funciones del actor frente al sistema informático.

El modelo de escenarios representa la funcionalidad total del sistema informático.

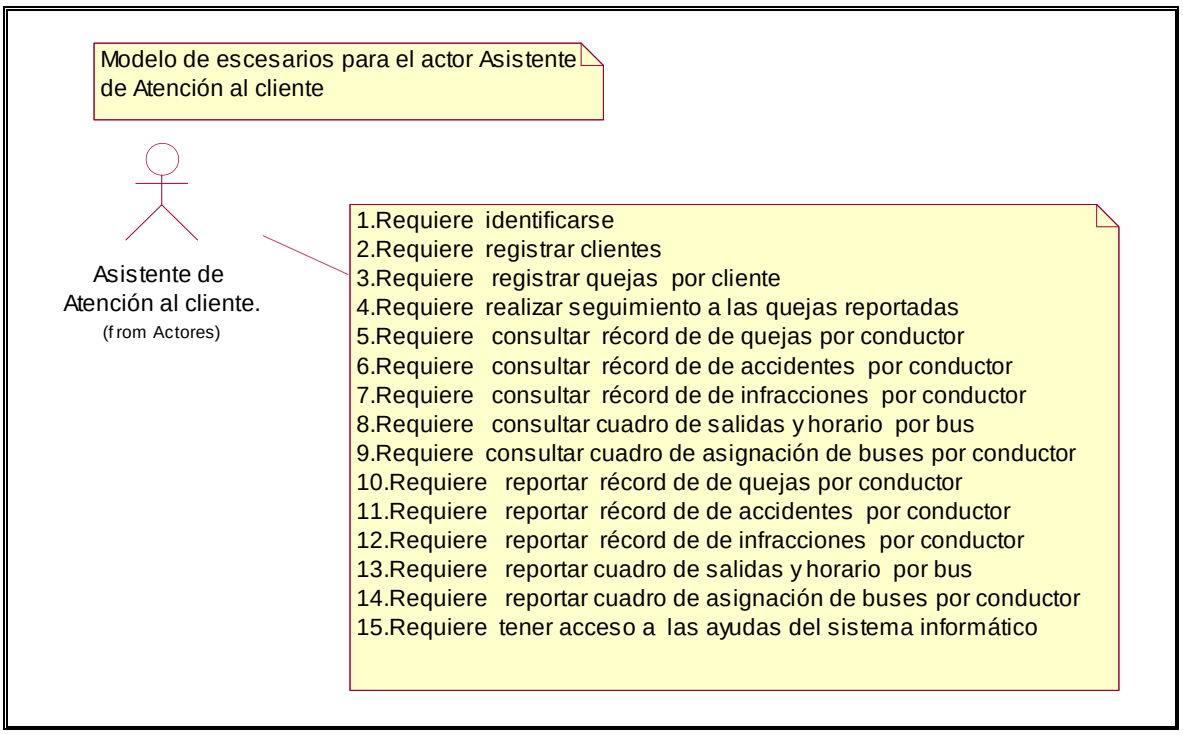


Figura N° 129, Esta figura muestra el modelo de escenarios del actor asistente del área atención al cliente

5.4.3. DEFINICION DEL ALCANCE DEL SISTEMA INFORMÁTICO

El alcance será definido en base al PMI[10], donde el alcance es definido como el objetivo principal de proyecto informático, en su definición el alcance debe tener algunas características como:

El alcance debe ser formulado de manera clara, sencilla y completamente delimitado, se de debe asegurar que los puntos de vista del desarrollador y del cliente coincidan sin dar espacio a ambigüedades de ningún tipo.

[10] PMI: Siglas del Project Managemet Institute, organización no gubernamental sin fines de lucro, establece criterios y lineamientos estándares en la admnistración de proyectos.

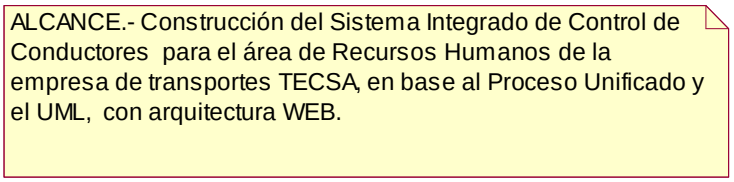


Figura N° 130, Ejemplo de alcance referido al caso empresa de transporte "TECSA"

5.4.4. PROCESO DE ABSTRACCIÓN

El proceso de abstracción[11] permite seleccionar diversos elementos de negocio que evolucionan para convertirse en elementos del sistema informático.

El proceso de clasificación es en base al ALCANCE del sistema informático.

Este proceso puede ser realizado para cualquier artefacto del negocio. Todos los artefactos de negocio son sensibles a ser seleccionados párale entorno del sistema informático final.

El proceso de abstracción puede realizarse de forma grupal o específica.

[11] ABSTRACCIÓN: Proceso de clasificación de elementos importantes o relevantes dentro de un universo de factores heterogéneos, donde la abundante cantidad de elementos innecesarios repercuten en el entendimiento del concepto.

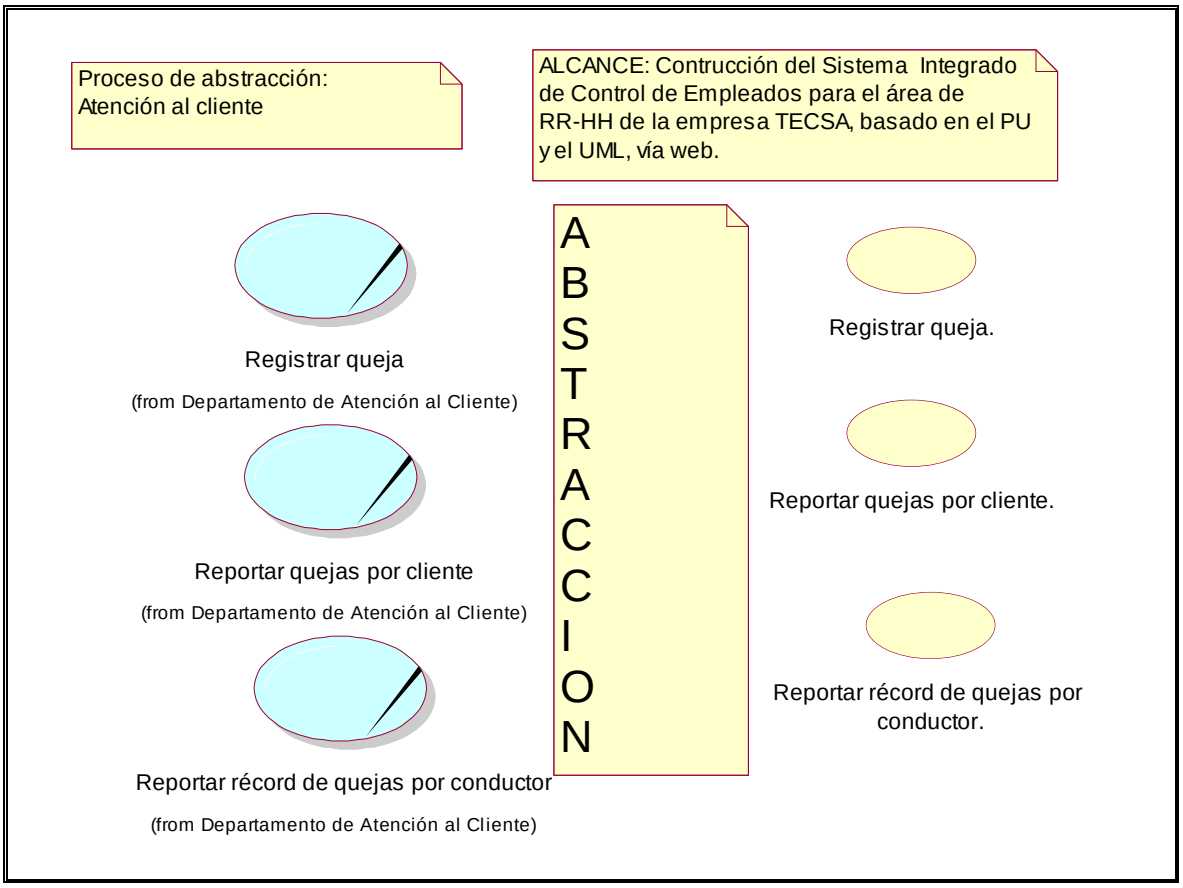
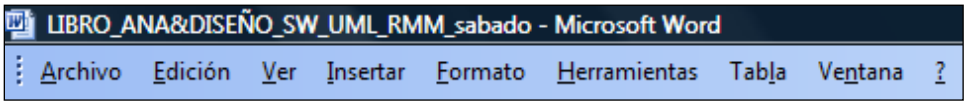


Figura N° 131, Ejemplo del proceso de abstracción referido al caso empresa de transporte "TECSA"

5.4.5. PROCESO DE CLASIFICACIÓN DE LOS CASOS DE USO DEL SISTEMA INFORMÁTICO EN BASE A MACROS.

Los casos de uso MACRO representan a los fragmentos funcionales generales del sistema informático final, aquellos que representan a opciones del menú principal. Un ejemplo claro lo podemos observar en las opciones generales del software de oficina Microsoft Word donde Archivo, Edición, Ver, etc pueden ser representado por el artefacto *caso de uso* y llevar la denominación *macro* para efectos de clasificación.



Ejemplo de casos de uso macro: Archivo, Edición, etc son representados por un caso de uso y llevar la denominación macro.

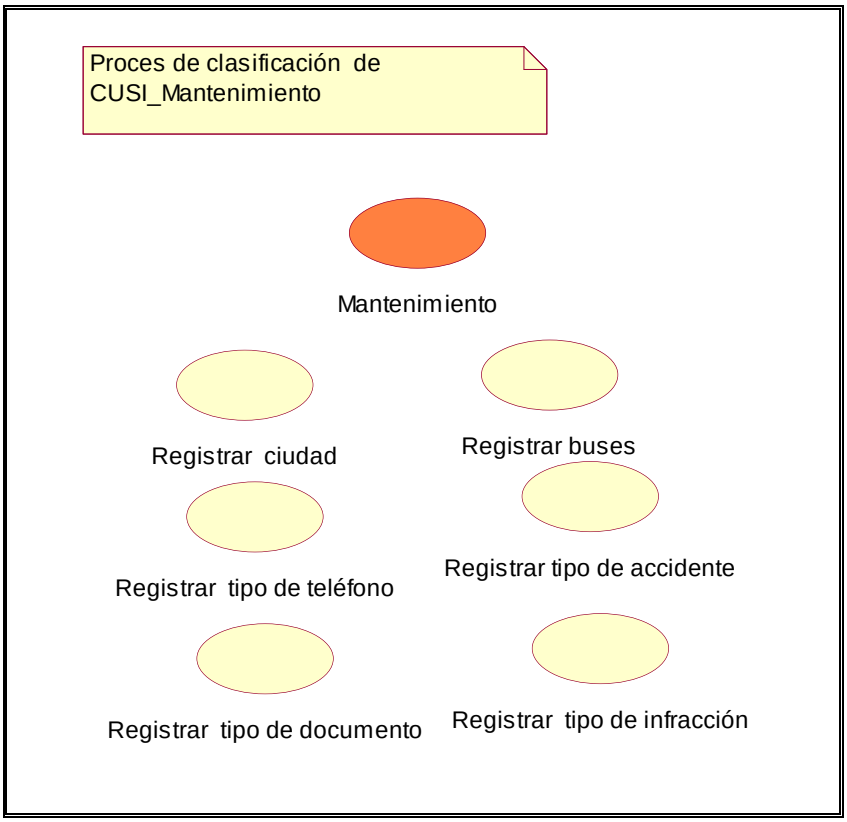


Figura N° 132, Ejemplo del proceso de clasificación en base a macros, referido al caso empresa de transporte "TECSA"

5.4.6. MODELO DE CASOS DE USO DEL SISTEMA INFORMÁTICO

En esta etapa se relaciona a los actores con los diversos casos de uso del sistema informático.

Este modelo se da a través del diagrama de Casos de Uso del UML, muestra las distintas operaciones que se esperan de una

aplicación o sistema informático y cómo se relaciona con su entorno (usuario u otras aplicaciones).

5.5. TIPOS DE RELACIONES EN EL MODELO DE CASOS DE USO DEL SISTEMA

 Asociación unidireccional, el cual permite relacionar al actor con el caso de uso.

 Generalización, este tipo de relación se da entre los actores del sistema, el cual significa herencia.

 Dependencia o instancia, este tipo de relación se da entre casos de uso, significa dependencia de inclusión.

 Realización, esta relación permite establecer la dependencia entre un caso de uso y su correspondiente caso de uso realización.

Cada de una de las relaciones anteriores están acompañadas de los denominados esteriotipos, los cuales brindan el comportamiento de una determinada relación. Los esteriotipos son considerados dentro de los caracteres << >>.

El comportamiento está preestablecido por los siguientes esteriotipos del UML.

5.6. ESTERIOTIPOS DEL UML

1. <<communicate>>, este esteriotipo significa comunicación, el cual se establece entre el actor y su correspondiente caso de uso.

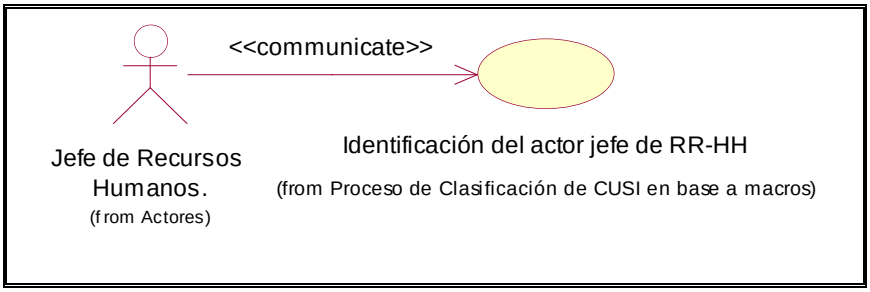


Figura N° 133, Ejemplo del esteriotipo <<communicate>> referido al caso empresa de transporte "TECSA"

2. << extend>>, utilizando en relaciones del tipo asociación y asociación unidireccional, esta relación obedece a la unión lógica del tipo "OR", representa una unión opcional.

   Esta relación entre dos casos, denota cuando un caso de uso es una especialización de otro, extiende comportamiento.

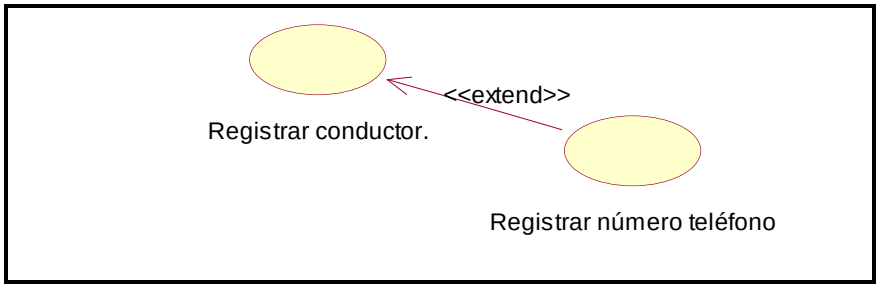


Figura N° 134, Ejemplo del esteriotipo <<extend>> referido al caso empresa de transporte "TECSA"

3. <<include>>, utilizado en la relación dependencia o instancia, esta relación obedece a la unión lógica del tipo "AND", significa una relación obligatoria.
   Esta relación se da entre dos casos de uso, denota la inclusión del comportamiento de un escenario en otro.

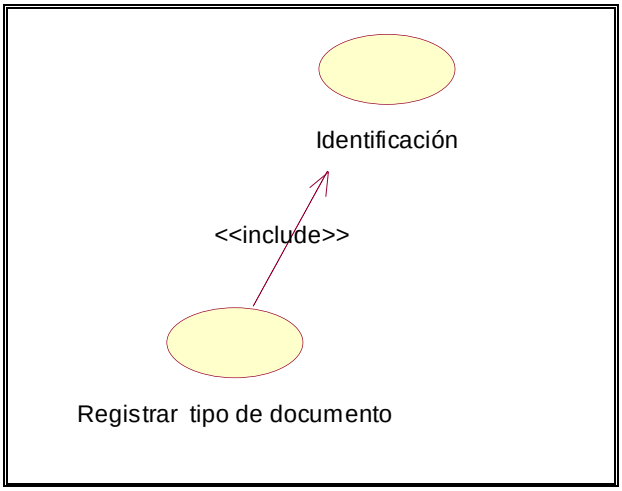


Figura N° 135, Ejemplo del esteriotipo <<include>> referido al caso empresa de transporte "TECSA"

4. << realize>>, esta relación está presenta entre el caso de uso y su correspondiente caso de uso realización, significa dependencia en el desarrollo.

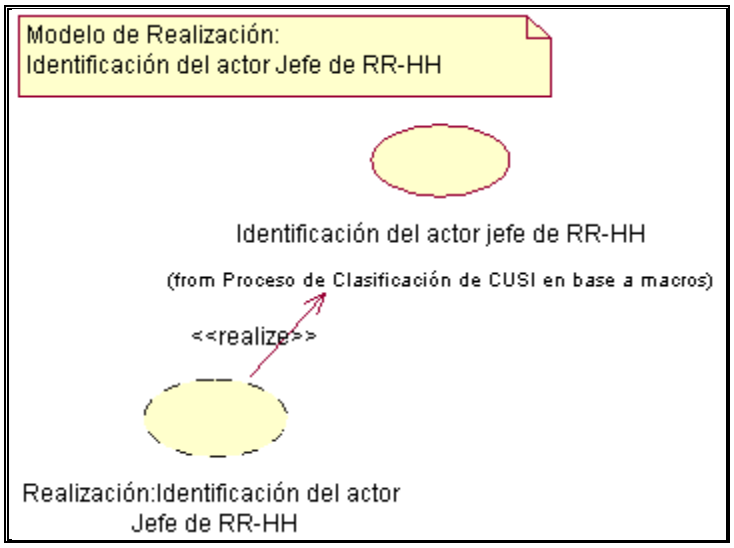


Figura N° 136, Ejemplo del esteriotipo <<realize>> referido al caso empresa de transporte "TECSA"

El modelo de casos de uso del sistema informático, se realiza en base a los requerimientos y/o funciones del actor, establecidos en el modelo de escenarios.

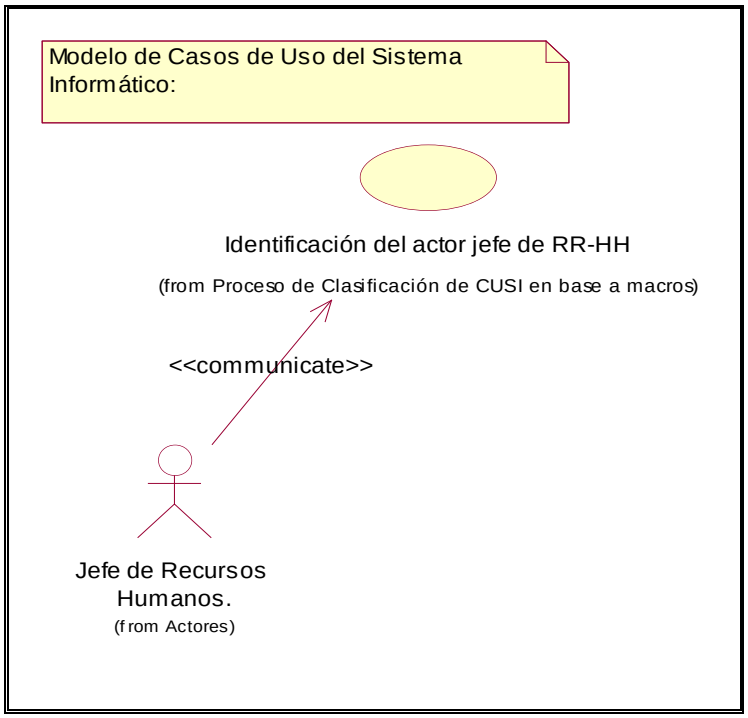


Figura N° 137, Modelo de casos de uso del sistema para la identificación del actor jefe del área recursos humanos

5.7. RESUMEN DEL CAPÍTULO

Estimado lector, en el presente capítulo brindé diversos conceptos y ejemplos para iniciar el proceso de construcción del software en base al Proceso Unificado de forma práctica, el cual es uno de los objetivos del presente texto.

La etapa de análisis de requerimientos deberá realizarse inmediatamente después de realizar el modelo de negocio, siendo este un análisis previo de suma importancia para garantizar que la solución informática haya sido construida en base a la realidad organizacional.

NO olvidemos que el desarrollo completo del caso práctico empresa de transportes TECSA de la actual etapa está ubicado en el CD, recomiendo revisar el archivo correspondiente, para despejar cualquier tipo de duda, NO olviden los tres grandes secretos para el logro de la especialización son 1.- PRACTICAR, 2.- PRACTICAR y 3.- PRACTICAR.

5.8. PREGUNTAS DE COMPETENCIA DEL CAPÍTULO

- Realizar un cuadro comparativo entre los artefactos UML del negocio y el análisis de requerimientos.
- ¿Por qué es necesario realizar el proceso de realización del caso de uso del sistema informático?
- ¿Qué determina el modelo de escenarios por actor?

CAPITULO VI

MODELO DE ANÁLISIS BASADO EN EL PROCESO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Instituir la importancia de los diversos clasificadores de análisis para la construcción de la solución informática basado en el Proceso Unificado.
- Explicar el proceso de realización de cada caso de uso del sistema informático.
- Detallar el modelo de análisis utilizando mas una entidad del sistema.

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector reconoce la funcionalidad de los diversos clasificadores de análisis.
- El leedor está en la capacidad de realizar modelos de análisis en base a varias entidades del sistema informático.
- El lector entiende la importancia del modelo de actividades para resolver factores de decisión.

6.1. INTRODUCCIÓN

El modelo de análisis es una etapa de transición entre el análisis de requerimientos y el análisis y diseño de la base de datos.

En esta etapa ya se conoce el alcance del proyecto informático, las funciones y/o requerimientos por cada uno de los actores; por lo tanto se tiene certeza de la funcionalidad total del sistema informático.

Dadas las entradas es tiempo de planificar técnicamente y en base a diagramas la implementación de los casos de uso del sistema informático ya considerados.

Así como los arquitectos utilizan los planos para asegurar la calidad de sus edificaciones, los profesionales en la informática también podemos realizar esos planos a través del modelo de análisis.

El modelo de análisis determina la etapa de codificación y el diseño de la base de datos, ya que en sus diversos diagramas se planteará la solución para que cada uno de los casos de uso del sistema se realice a la medida de las necesidades del usuario.

6.2. IMPORTANCIA

La planificación y el trabajo metódico-sistémico es la clave para el éxito planteado por el Proceso Unificado. Para garantizar la performance adecuada de la solución informática final se debe probar que los casos de uso del sistema informático estén realizados en base a las necesidades del usuario; ello se garantiza a través de los diversos planos (diagramas) realizados en el modelo de análisis.

El modelo de análisis a través de sus clasificadores permite la realización de cada caso de uso preparando el terreno hacia el diseño de la base de datos y la codificación del sistema informático, garantizando el cumplimiento del modelo iterativo e incremental lo cual también garantiza la homogenización de la propuesta.

6.3. ELEMENTOS UML PARA EL MODELO DE ANÁLISIS

6.3.1. CONTROL

Este clasificador de análisis representa un elemento de control transaccional en programación, puede representar a un objeto, un agente, un elemento que busca conexión con la base de datos o con la red. También representa la coordinación, secuencia y control de otros objetos, y que se usa a menudo para encapsular control referido a un determinado caso de uso.

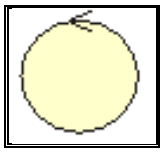


Figura N° 138, Notación UML, representa al clasificador de análisis "CONTROL"

6.3.2. BOUNDARY

Clase del modelo de análisis usado para modelar la interacción entre el sistema y sus actores, esto es, usuarios y sistemas externos.

Esta notación UML que representa a una interfaz (ventana) del sistema informático.

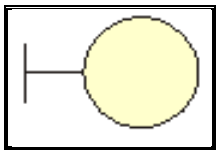


Figura N° 139, Notación UML, representa al clasificador de análisis "BOUNDARY"

6.3.3. ENTIDAD

Clase del modelo de análisis usado para modelar información de larga duración y a menudo persistente. Esta notación representa a una tabla de la base de datos, puede representar a un simple repositorio o una estructura completamente definida como un procedimiento almacenado por ejemplo.

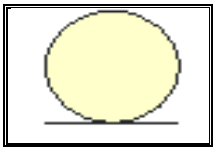


Figura N° 140, Notación UML, representa al clasificador de análisis "ENTIDAD"

6.4. ETAPAS DEL MODELO DE ANÁLISIS

6.4.1. MODELO DE REALIZACIÓN

Este modelo es una representación gráfica y didáctica en el cual se muestra la dependencia del caso de uso con su correspondiente caso de uso realización.

El caso de uso realización será el contenedor de todos los diagramas para el desarrollo del caso de uso que se analiza.

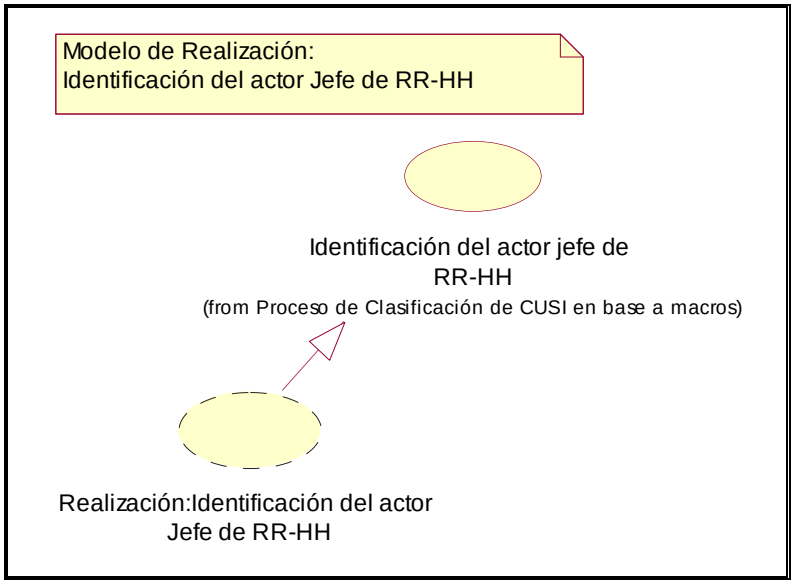


Figura N° 141, Modelo de realización para la identificación del actor jefe de recursos humanos, aplicado al caso empresa de transporte "TECSA"

6.4.2. MODELO DE CLASES

Este modelo muestra a las diversas clases del modelo relacionadas a través de la relación del tipo "asociación", el cual es bidireccional por defecto, esta relación NO significa dependencia simplemente participación y relación de algún tipo. El nombre de las asociaciones se denomina "reglas del software", aquellas que dirigen y/o controlan el proceso de la construcción informática.

Los clasificadores son básicamente clases, es en esencia, un conjunto de objetos con características comunes.

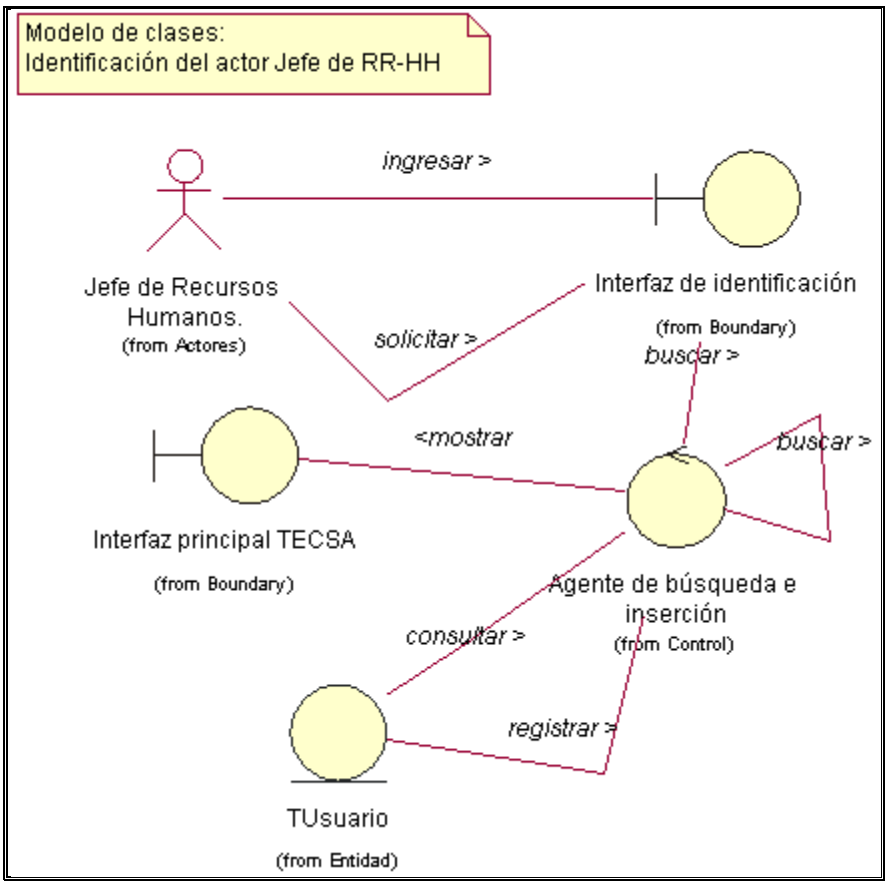


Figura N° 142, Modelo de clases para la identificación del actor jefe de recursos humanos, aplicado al caso empresa de transporte "TECSA"

6.4.3. MODELO DE INTERACCIÓN

Este modelo puede realizarse en un diagrama de secuencias o de colaboración del UML, ambos diagramas como se mencionó en el capítulo número 2 del presente texto, ambos diagramas son equivalentes variando solo en la distribución.

Cada uno de los objetos que participan en este modelo se comunica a través de un conjunto de mensajes los cuales pueden ser lineales o recursivos; el modelo de interacción

muestra en forma clara y concisa la realización del caso de uso asumiendo la NO existencia de factores de decisión, se asume la ocurrencia afirmativa de todas las acciones.

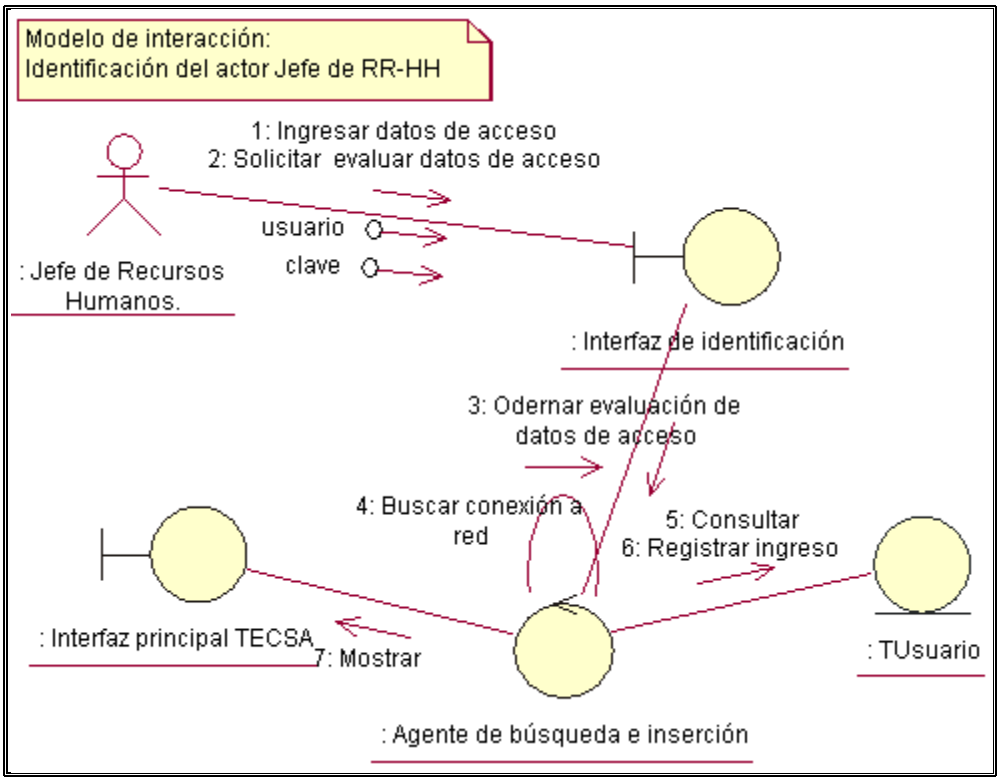


Figura N° 143, Modelo de interacción para la identificación del actor jefe de recursos humanos, aplicado al caso empresa de transporte "TECSA"

6.4.4. MODELO DE ACTIVIDADES

El modelo permite resolver factores de decisión a través del factor "desicion" del diagrama de actividades, el factor en cuestión tiene salidas del tipo boolen(2 alternativas de respuesta, puede ser verdadero o falso).

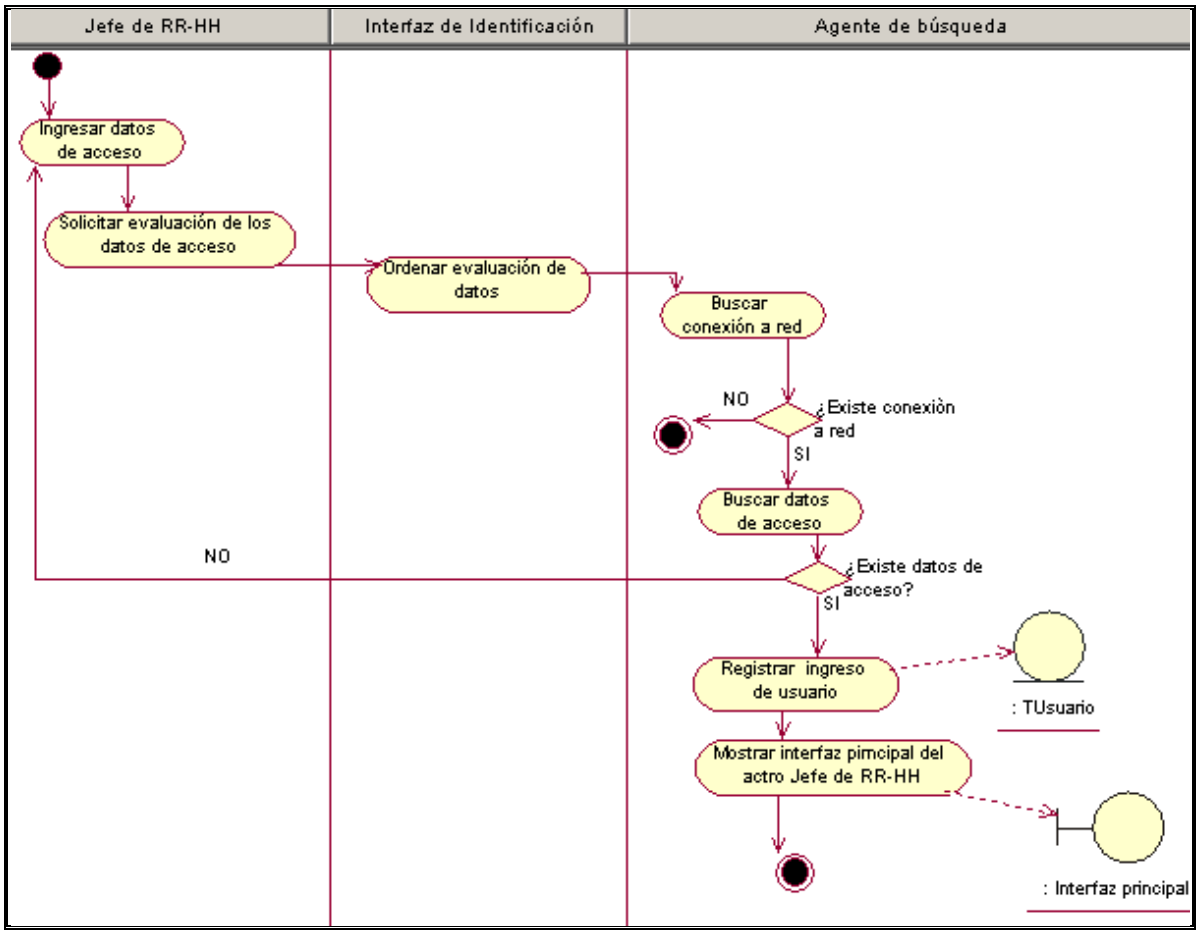


Figura N° 144, Modelo de actividades para la identificación del actor jefe de recursos humanos, aplicado al caso empresa de transporte "TECSA"

Estimado lector, NO olvide que el desarrollo completo del proyecto se encuentra en el CD que acompaña al texto, recomiendo revisar el proyecto para que no quede duda alguna respecto a la utilidad del modelo de análisis.

6.5. CUADRO COMPARATIVO ENTRE EL MODELO DE CASOS DE USO Y DEL MODELO DE ANÁLISIS BASADO EN EL PROCESO UNIFICADO

MODELO DE CASOS DE USO	MODELO DE ANÁLISIS
Descrito en el lenguaje del cliente (coloquial, no técnico).	Descrito en el lenguaje técnico, propio del desarrollador.
Vista externa del sistema informático.	Vista interna del sistema informático
Estructurado por los casos de uso, proporciona la estructura a la vista interna.	Estructurado por clases y paquetes esteriotipados; proporciona la estructura de la vista externa.
Utilizado como contrato entre el cliente y los desarrolladores, referido a los límites funcionales del sistema informático.	Utilizado por los desarrolladores para comprender el cómo debería darse diseñarse e implementarse el sistema informático.
Captura la funcionalidad del sistema informático, incluyendo el conjunto de funciones dirigidas a la arquitectura.	Esboza el como llevare a cabo la funcionalidad significativa para la arquitectura, es la primera aproximación al diseño.
Define casos de uso que se realizarán al detalle en el modelo de análisis	Define realizaciones de caso de uso, cada una de ellas representa el análisis de un caso de

	uso planteado en el modelo de casos de uso.

Fuente: Jacobson, Ivar
1999 The Unified Software Development Process 1ra edición EE.UU., Addison Wesley

6.6. RESUMEN DEL CAPÍTULO

El modelo de análisis permite planificar la etapa de análisis & diseño de la base de datos y la etapa de codificación. Esta etapa garantiza el cumplimiento del modelo iterativo e incremental logrando un modelo completamente homogéneo.
El modelo de análisis hace posible la realización de todos los casos de uso planteados en la etapa de análisis de requerimientos.

6.7. PREGUNTAS DE COMPETENCIA DEL CAPÍTULO

- Explicar la funcionalidad de cada uno de los clasificadores de análisis.
- ¿Cuáles son las etapas del modelo de análisis?
- ¿Porque el actor del sistema informático cumple un papel crucial en la etapa de análisis?

CAPITULO VII

ANÁLISIS Y DISEÑO DE LA BASE DE DATOS BASADO EN EL PROCESO UNIFICADO

OBJETIVOS DEL CAPÍTULO

- Detallar los modelos necesarios para la construcción de la base de datos.
- Determinar la importancia del modelo lógico.
- Detallar el proceso de conversión de un modelo de objetos a un modelo de datos y viceversa.

COMPETENCIAS CONCEPTUALES Y PROCEDIMENTALES A LOGRAR EN EL LECTOR

- El lector reconoce la importancia de los modelos conceptual, lógico y físico para la construcción de la base de datos.
- El leído conoce acerca de las repercusiones de una base de datos con presencia de datos nulos y redundantes.
- El lector conoce las características de los atributos de una clase diferenciándolos de los objetos.

7.1. INTRODUCCIÓN

En el presente capítulo analizaremos los criterios y pasos necesarios para la creación de una base de datos en base a conceptos orientados a objetos.

Iniciaremos con el entorno conceptual donde los artefactos UML para esta etapa del proceso unificado marcan un aspecto crucial.

Los modelos orientados a objetos utilizados en la fase de análisis de la base de datos como el modelo conceptual y lógico, son analizados en forma individual y teniendo en cuenta criterios de comparación por ser dependientes desde el punto de vista iterativo e incremental[12].

Muchas de las organizaciones peruanas e incluso internacionales basan sus modelos de base de datos basados en el metodología estructurada[13], donde el modelo entidad relación o modelo de datos es un elemento de suma importancia; para lograr congruencia entre las propuestas de datos de estas empresas y la propuesta orientada a objetos actual, se trabajará con un conjunto de procedimientos que permitan la conversión de un modelo de objetos a un modelo de datos y viceversa, conocido como la opción "data modeler"

7.2. IMPORTANCIA

La buena definición de una base de datos permite garantizar el buen funcionamiento y calidad del sistema informático.

La base de datos creada a partir de un modelo de objetos tiene ventajas importantes en la robustez y eficacia de la

[12] Modelo iterativo e incremental: Modelo de construcción de software donde las iteraciones se refieren a pasos en el flujo de trabajo, y los incrementos a un crecimiento en el producto. Este modelo garantiza la homogenización del producto software.

[13] Metodología estructurada: Conjunto de métodos para construcción de sistemas informáticos basado en estructuras. La base conceptual de esta metodología es el dato.

base de datos. A través del modelo de objetos se logra una propuesta donde NO es necesario el engorroso proceso de normalización[14]; en el modelo de objetos la propuesta ya está normalizada por NO poseer datos nulos ni datos redundantes; es una estructura completamente homogénea.

7.3. ELEMENTOS UML ORIENTADO AL DISEÑO DE LA BASES DE DATOS

7.3.1. CLASE

Notación UML que representa a un conjunto de objetos con características comunes.

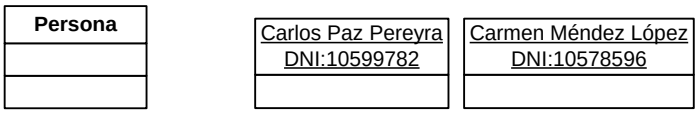


Figura N° 145, Ejemplos de la clase persona y sus objetos realizados en el case de modelado Microsoft Visio

7.4. TIPOS DE RELACIONES ENTRE CLASES

7.4.1. HERENCIA

Este tipo de relación indica la generalización o herencia entre una clase padre o súper clase y sus diversas clases hijas.

Sin un conjunto de clases hijas contiene elementos en común como atributos, operaciones o incluso relaciones, estos deben ser absorbidos por el padre ya que la repetición es

[14] Proceso de Normalización: Consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo , este proceso sirve para evitar la redundancia de los datos, evitar problemas de actualización de los datos en las tablas y para garantizar la integridad de los datos.

considerado redundancia por la metodología orientado a objetos.

Por lo tanto la generalización es el proceso de identificar las características comunes y definir relaciones entre una Superclase (genérico) y Subclases (conceptos especializados, específicos). Una clase hija puede ser reconocida mediante las palabras reservadas "*Es un tipo de*".

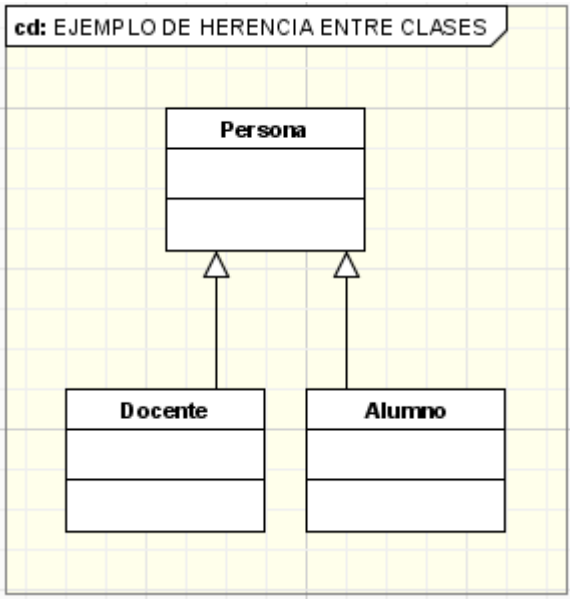


Figura N° 146, Notación UML del Caso de uso de Negocio. Ejemplo de herencia realizado en el case de modelado Poseidón[15]

7.4.2. COMPOSICIÓN

Esta tipo de relación permite representar a los componentes (partes) conocido como maestro con sus diversas partes conocido como esclavos.

La Agregación puede ser reconocido mediante las palabras reservadas "*Es parte de*".

[15] Poseidón: Es una herramienta case, soporta a la técnica de modelado UML, esta herramienta puede simplificar la compleja tarea de desarrollo de software ayudando a estructurar pensamientos, a clarificar la comunicación, y a encontrar la correcta abstracción.

La composición modela las partes de un todo tal cual existe en la realidad.

Este tipo de relación implica que la llave primaria de una clase esclavo siempre será compuesta por la llave primaria de la clase maestro más el campo identificador de la clase esclavo.

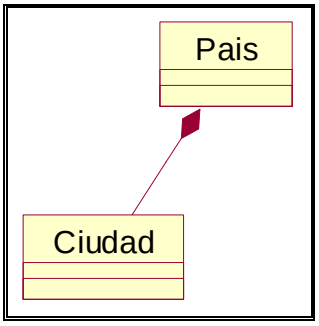
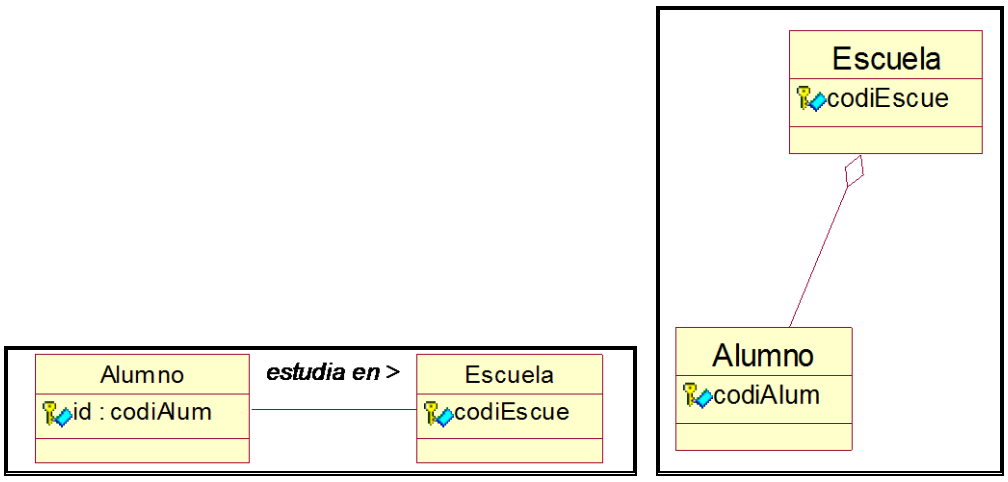


Figura N° 147, Ejemplo de composición realizado en el case de modelado Rational Rose. La llave primaria de la futura tabla esclavo (Ciudad) deberá estar compuesto por el campo identificador del maestro mas el campo identificador del esclavo (codipais, codiciudad)

7.4.3. AGREGACIÓN

Conceptualmente en idéntico a la composición, la diferencia radica que la relación del tipo agregación sirve para forzar la conformación de la llave primaria, vale decir cuando es necesaria la característica compuesta de la llave. Por ejemplo la clase Alumno y la clase Escuela puede relacionarse a través de la relación del tipo "asociación", en este caso, las llaves primarias de cada una de esas clases son complemente independientes es decir sencillas, pero si el analista considera necesaria que la llave

primaria sea compuesta se puede forzar la relación a través de la agregación.



ASOCIACIÓN **COMPOSICIÓN**

Figura N° 148, Ejemplo Las clases Alumno y Escuela puede relacionarse a través de la asociación o la composición, la consecuencia de las llaves primarias se visualizan en los cuadros anteriores.

7.4.4. ASOCIACIÓN

Este tipo de relación permite asociar a 2 clases independientes por algún motivo de control, el producto de la asociación entre las clases se denomina clase asociativa o atributo de enlace.

La asociación es bidireccional por defecto y sensible a criterios de multiplicidad.

Las asociaciones tienen nombre propio, este es considerado como la "regla del software".

La convención de lectura de la asociación es de izquierda a derecha, pero en caso se complique la lectura por la gran cantidad de artefactos que se puede considerar, pueden

utilizar los símbolos de mayor y menor (>, <) para direccionar la lectura.

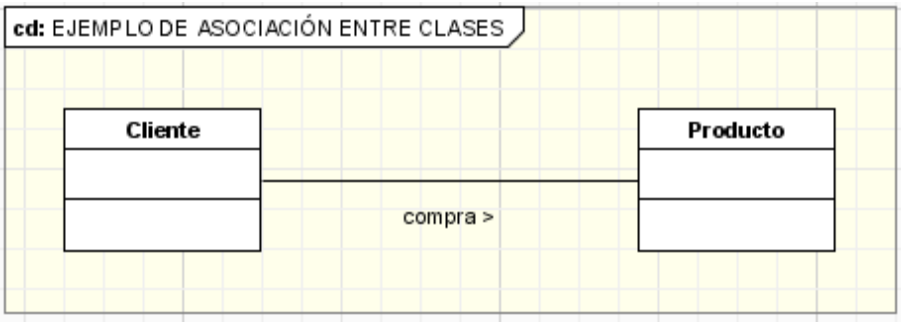


Figura N° 149, Ejemplo de la relación del tipo "asociación", realizado en el case de modelado Poseidón 5.0

7.4.5. ASOCIACIÓN UNIDIRECCIONAL

Este tipo de relación deriva de la relación del tipo asociación; para convertirlo a una asociación unidireccional basta con desactivar la opción de navegabilidad en la asociación. La relación unidireccional significa dependencia y permite evitar nulos en la base de datos. La clase a la cual indica la asociación es conocida como la clase independiente, esta debe existir en la base de datos incluyendo sus valores (registros) antes de insertar algunos datos en la clase dependiente (aquella de donde sale la asociación unidireccional), ya que puede repercutir en datos nulos.

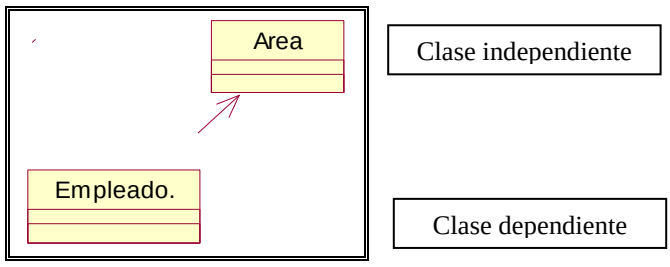


Figura N° 150, Ejemplo de la asociación unidireccional. La clase independiente (área) debe existir (deberá contener por

lo menos un registro) para insertar algún dato en la clase Empleado, de esta manera se evita la presencia de datos nulos en la clase dependiente (Empleado).

7.4.6. CLASE ASOCIATIVA

Este tipo de relación conocido también como atributo de enlace está presente en las relaciones del tipo asociación cuyas multiplicidades son de muchos a muchos.

Otro criterio indicador para considerar la presencia del atributo de enlace es cuando se tiene un dato, el cual no puede ser ubicado en ninguna de las clases padre; es ese caso de debe crear una tercera clase denominado atributo de enlace o clase asociativa.

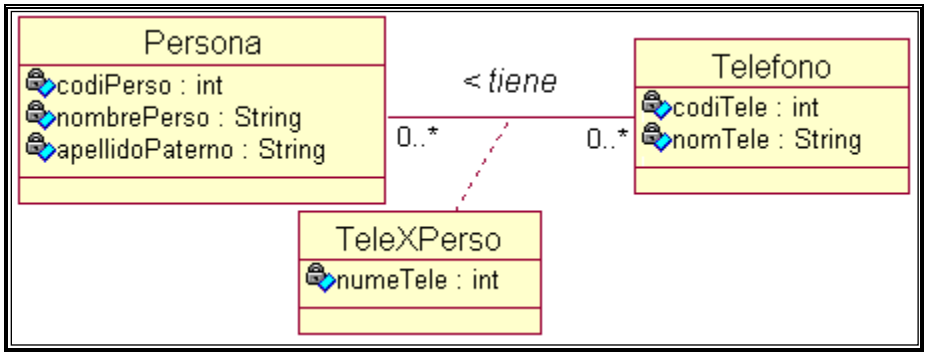


Figura N° 151, Ejemplo de atributo de enlace, el atributo "numeTele", no puede ser ubicado en la clase Persona (causa datos nulos), ni el la clase Teléfono (no se puede conocer a quien pertenece al número de teléfono), es un necesidad considerar la tercera clase TeleXPerso, para ubicar el atributo en cuestión

7.5. MULTIPLICIDAD

Es una característica de las asociaciones, muestra los diversos tipos de interacción entre cada uno de los objetos que participan en la relación.

Existen diversos tipos de multiplicidad, como lo podemos observar en la figura N°

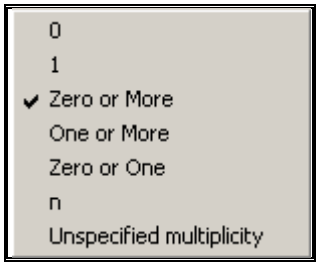


Figura N° 152, Diversos tipos de multiplicidad

La lectura de las multiplicidades se explicará en base a la figura N°

Una Persona puede tener cero o muchos tipos de teléfono, por lo tanto la multiplicidad que corresponde es 0..*, fíjese que esa multiplicidad se coloca al costado de la clase Teléfono.

Ahora analicemos la lectura en otra dirección, un tipo de teléfono puede ser poseído por cero o muchas personas, nuevamente la multiplicidad que corresponde es 0..*, nótese que está ubicado al costado de la clase Persona.

7.6. ETAPAS DEL ANÁLISIS Y DISEÑO DE LA BASE DE DATOS

7.6.1. MODELO CONCEPTUAL

Este modelo marca el inicio del análisis y diseño de la base de datos relacional/objeto.

El modelo conceptual se caracteriza por ser sencillo y orientado al usuario, considera en su ámbito a las clases relevantes[16].

Las clases en este entorno están relacionadas a través asociaciones, herencias, agregaciones y composiciones.

Es de suma importancia nombrar las diversas reglas del software ya que de ellas depende el éxito de las base de datos.

"Carlitos's Pizza ", es un empresa dedicada a la venta de pizzas y bebidas gaseosas. La empresa desea controlar los registros de pedidos para pizzas y bebidas. Los clientes llaman por teléfono a realizar su pedido, se les solicita su número telefónico. Cuando el número es tecleado en una computadora aparecen automáticamente el nombre, dirección y fecha del último pedido en la pantalla. Una vez que es tomada la orden se calcula el total, incluyendo el impuesto. Luego, se pasa el pedido a la cocina. Se imprime un recibo. Los choferes que hacen la entrega le dan a los clientes una copia del comprobante de pago y reciben el dinero.

Figura N° 153, Caso de estudio Carlitos's Pizza.

[16] Clases Relevantes: Clases independientes las cuales no requieren de una relación previa para su existencia

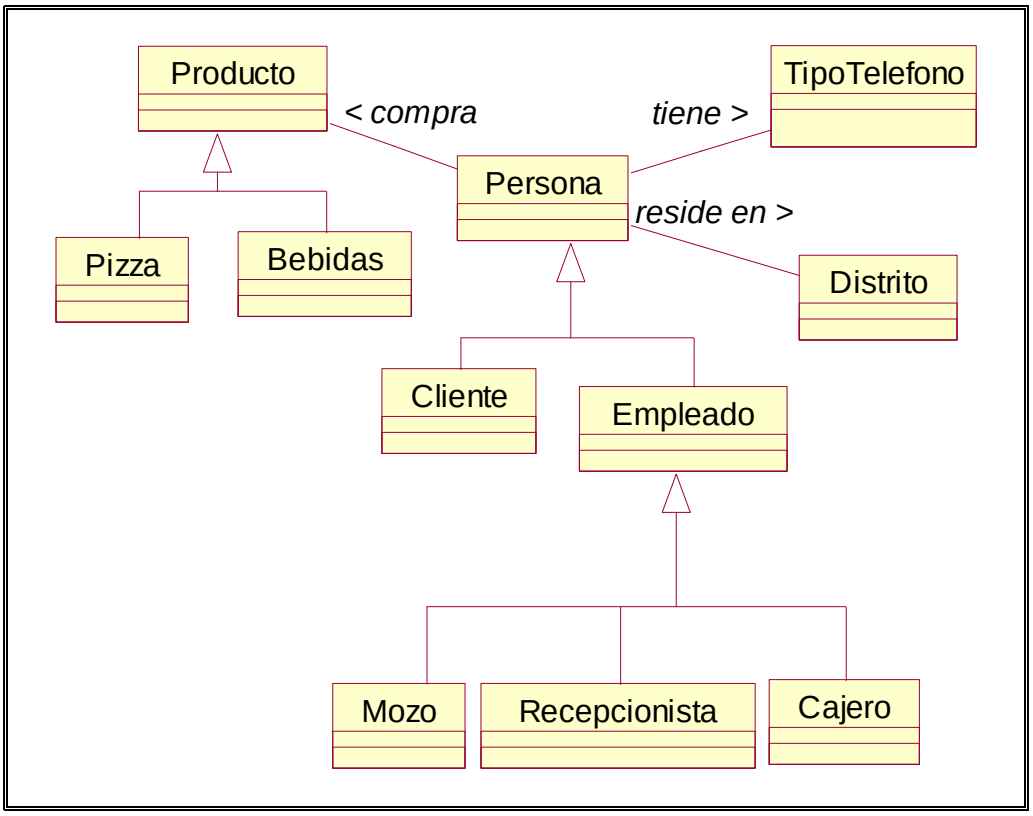


Figura N° 154, Modelo conceptual aplicado al caso empresa expendedora de pizzas Carlitos's Pizza. Se trabaja bajo el supuesto que la persona compra solo un producto.

7.6.2. MODELO LÒGICO

Esta etapa es realizada en base al modelo conceptual, se caracteriza por estar orientado a los miembros del equipo de desarrollo del sistema informático por su alto contenido técnico.

En el modelo lógico se resuelven problemas de datos nulos, redundantes y heterogéneos. En esta propuesta el concepto de normalización no es utilizado ya que el modelo lógico ya que el modela no presenta nulos ni redundancia.

En el proceso de transición del modelo conceptual al modelo lógico es necesaria la aplicación de los siguientes principios:

7.6.2.1. PRINCIPIOS DE CONVERSIÓN DEL MODELO CONCEPTUAL AL MODELO LÓGICO

- Todas las clases del modelo conceptual que tenga relación directa con una regla del software pasará directamente al modelo lógico.

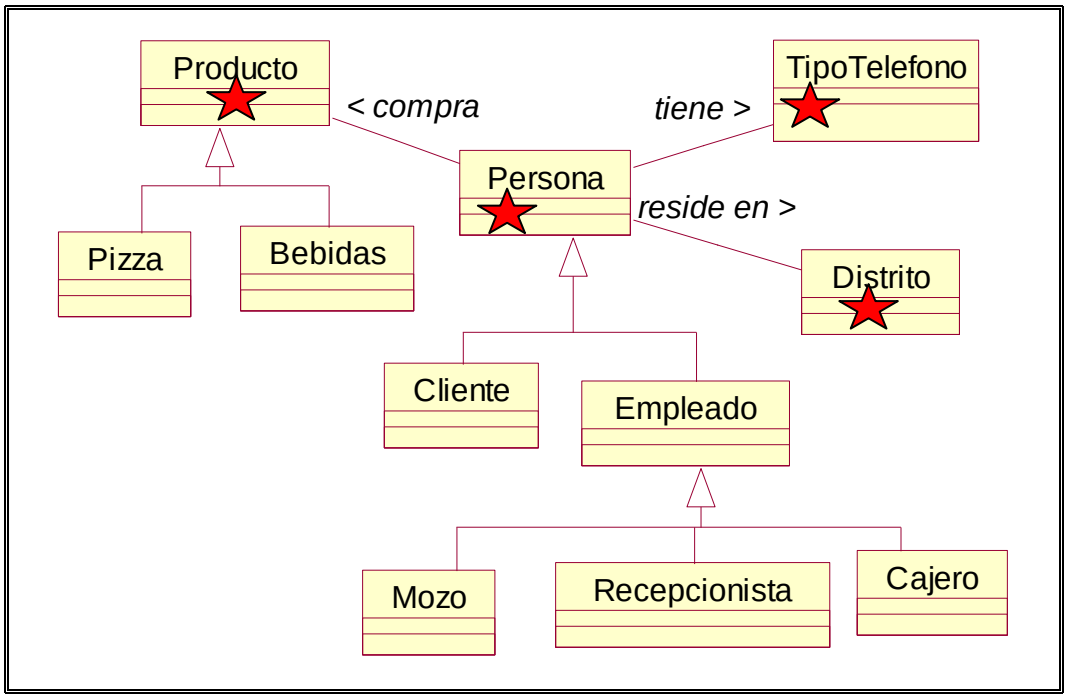


Figura N° 155, La figura muestra a aquellas clases que pasarán al modelo lógico directamente. Son las que están identificadas a través de estrellas.

- Todas las clases del modelo lógico, presentan atributo[17]s

[17] Atributos: Características inherentes a la clase, aquellas con las que la clase nace.

Figura N° 156, Clase con atributos

- Las clases que derivan de una herencia o agregación y no tienen relación directa con una regla del software serán sensibles a la teoría de conversión.

CONVERSIÓN ASCENDENTE

Proceso de conversión de un atributo a una clase.

CONVERSIÓN DESCENDENTE

Proceso de conversión de una clase a un atributo.

CONVERSIÓN MIXTA

Este tipo de primitiva se presenta cuando se aplica en un caso en particular ambos teorías.

Para entender mejor este principio planteo el siguiente ejemplo, el cual estará resuelto paso por paso.

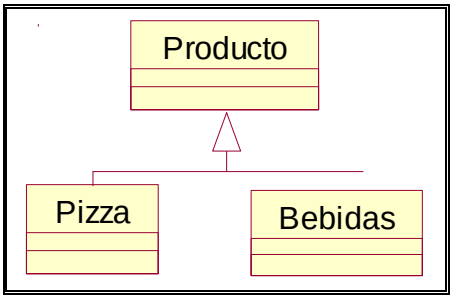


Figura N° 157, Ejemplo de clases hijas (en base a la herencia) que no tienen relación con una regla de software.

Dada la figura N°, tenemos dos clases hijas que no tienen relación directa con una regla del software, este en un buen caso para aplicar las teorías de conversión.

PASO 01: Establecer un nombre que permita identificar a cada una de las clases hijas en forma genérica, en este caso "TIPO PRODUCTO".

PASO 02: Las clases hijas son reemplazadas por un término nuevo, tipo producto el cual se convierte en atributo, la transición de conversión de clase de atributo se le denomina conversión descendente.

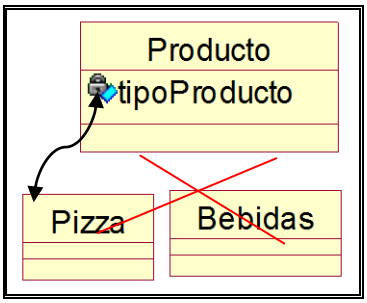


Figura N° 158, Teoría de conversiones

PASO 03: Analizamos al nuevo atributo, si el atributo puede convertirse en clase (solo si tiene objetos) entonces se produce una conversión ascendente. El atributo "tipoProducto", se convertirá en clase y estará relacionada con la antigua clase padre Producto a través de una relación del tipo asociación unidireccional. Los objetos de la nueva clase TTipoProducto serán pizza y bebidas.

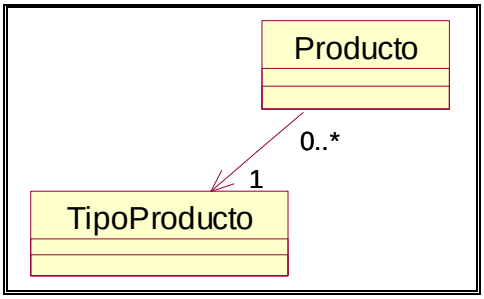


Figura N° 159, Resultado después de aplicar la teoría de conversiones.

Para este ejemplo se utilizaron ambas teorías, por lo tanto se habla de una conversión mixta.

- En las relaciones del tipo asociación considerar características de multiplicidad.
- Para aquellas relaciones cuyas multiplicidades son de muchos a muchos considerar a los atributos de enlace conocidos también como clases asociativas.
- Para las relaciones del tipo asociación con multiplicidades diferentes a muchos a muchos, considerar a las asociaciones unidireccionales, las cuales significan dependencia y generalmente coinciden con la multiplicidad de 1. La relación del tipo asociación unidireccional permite evitar datos nulos en las base de datos.
- En el modelo lógico se puede agregar o eliminar clases a criterio del diseñador de la base de datos.

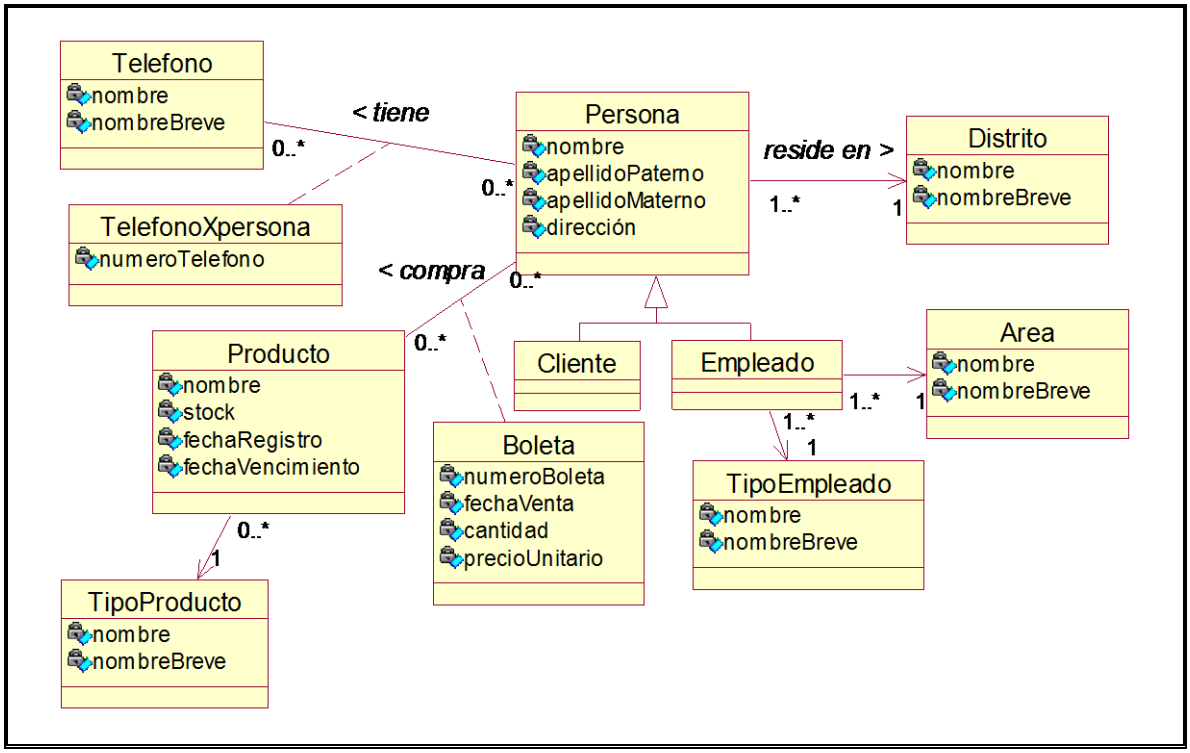


Figura N° 160, Modelo Lógico aplicado al caso Carlitos's Pizza

7.6.3. MODELO FÍSICO

Este modelo, es la propuesta física de la base de datos, donde se considera a los campos que forman parte de la llave primaria de clase.

Las clases del modelo físico tienen es esteriotipo tabla relacional, motivo por el cual se consideran características de tipo de dato y longitud por cada campo.

El modelo físico será construido en base al wizard del del gestor de base de datos oracle, ubicado como una opción mas en el case de modelado Rational Rose.

7.6.3.1. CONSTRUCCIÓN DE TABLAS EN BASE AL MODELO DE OBJETOS

A continuación explicaré como construir la tabla utilizando la opción mencionada.

PASO N° 01

Realizar el orden de carga de cada una de las tablas, se inicia con las clases independientes para finalizar con la clase que tenga la mayor cantidad de dependencias posibles.

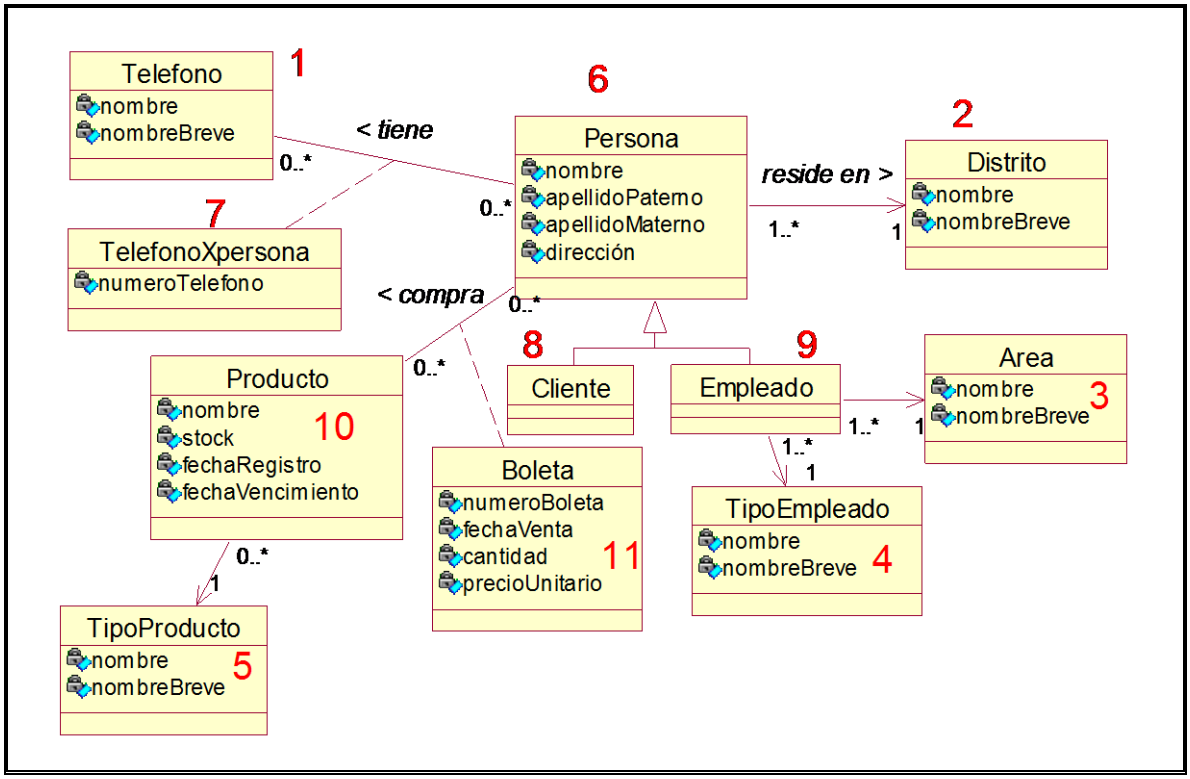


Figura N° 161, La figura anterior muestra el orden de creación y conversión de las clases a tablas.

PASO N° 02

Todas las relaciones se convierten a asociaciones unidireccionales. La diferencia radica en la composición de la llave primaria.

Las tablas independientes tienen llaves simples, conformadas por solo un campo.

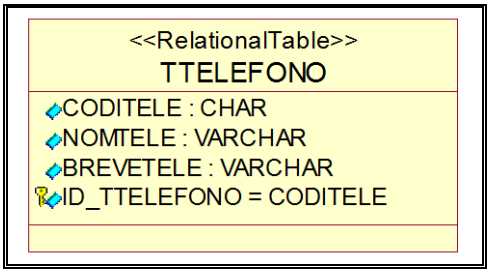


Figura N° 162, La figura muestra a la tabla independiente TTELEFONO

Las tablas dependientes tienen llaves compuestas, dependiendo del caso.

7.6.3.2. DEFINICIÒN DE LLAVES EN TABLAS DEPENDIENTES

A continuación detallaré cada uno de los casos donde las llaves primarias son **COMPUESTAS.**

CASO N° 01 CLASE ASOCIATIVA

Si la clase es un atributo de enlace, la llave primaria de la tercera clase (clase asociativa) es compuesta, estará formado por la llave primaria de cada una de las clases padre.

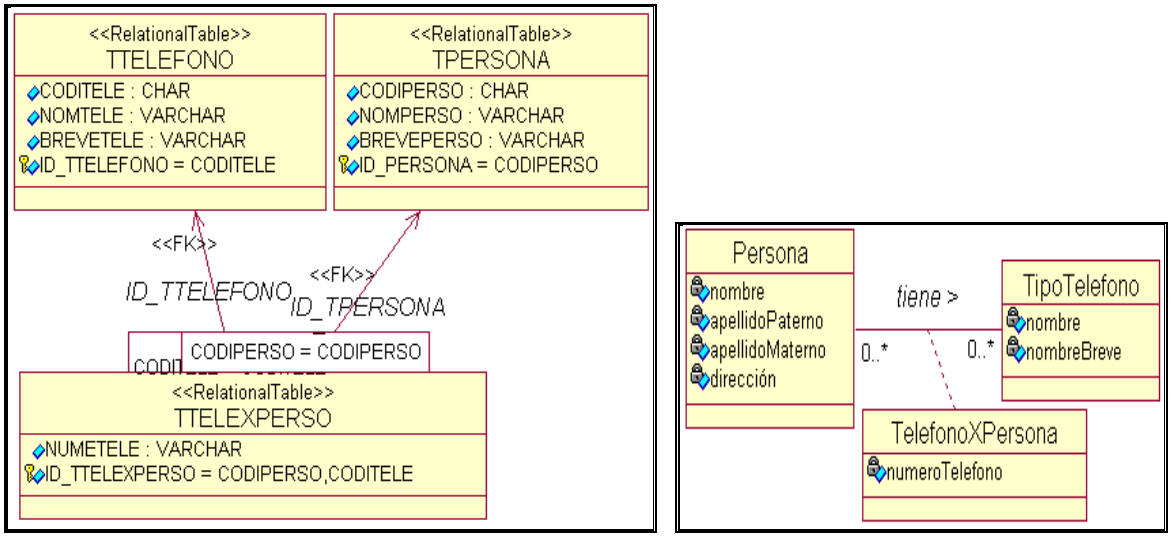


Figura N° 162, La figura muestra el proceso de conversión a tabla de un atributo de enlace. Nótese que la llave primaria de la tabla dependiente (TTELEXPERSO) está formado por los campos identificadores de las tablas independientes (TTELEFONO y TPERSONA)

CASO N° 02 CLASES QUE DERIVAN DE LA RELACIÓN DEL TIPO HERENCIA

Las llaves primarias de las clases hijas están formadas por la llave primaria del padre; esta llave puede ser editada para su fácil ubicación, pero la referencia a la clase padre NO puede cambiar.

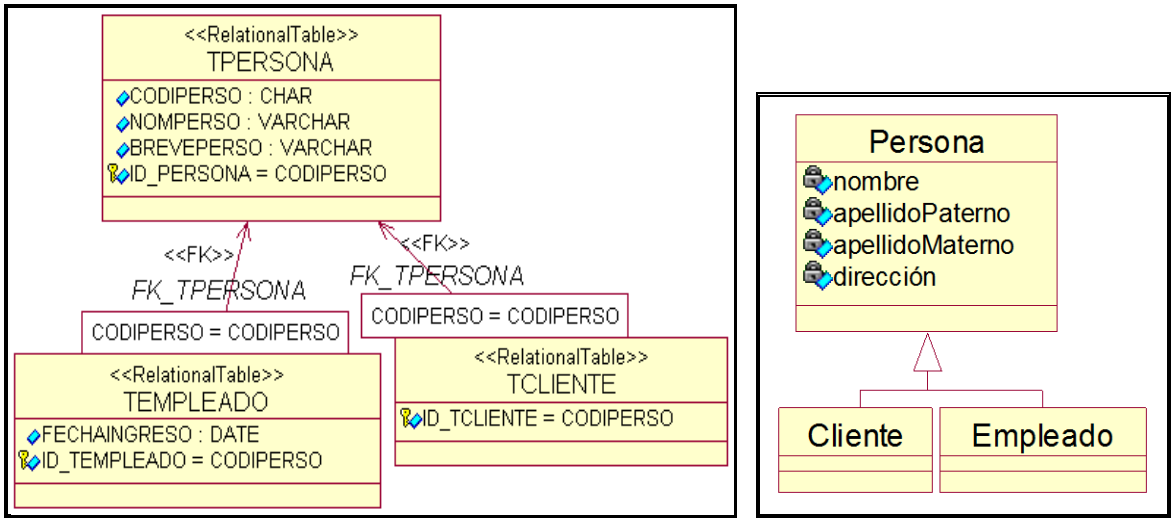
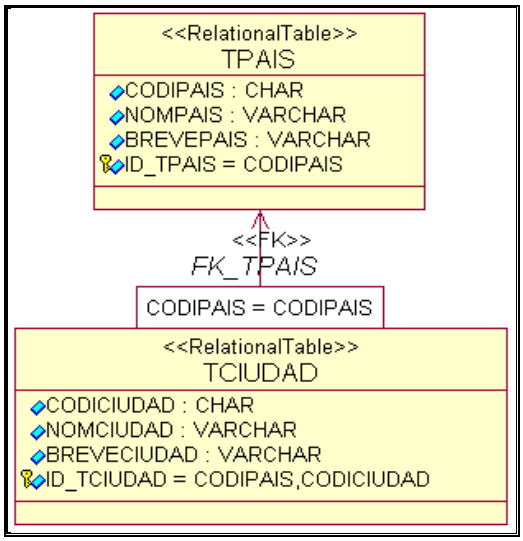


Figura N° 163, La figura muestra el proceso de conversión a tabla de clases hijas (Cliente y Empleado)

CASO N° 03 CLASES QUE DERIVAN DE LA RELACIÓN DEL TIPO AGREGACIÓN Y COMPOSICIÓN

La llave primaria de las tablas esclavo que derivan tanto de agregación y/o composición siempre con compuestas. Estarán

formados por la llave primaria del maestro más el campo identificador del esclavo.



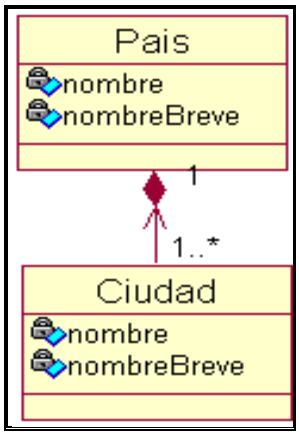


Figura N° 164, La figura muestra el proceso de conversión a tabla de una clase producto de la relación del tipo composición. Nótese que la llave primaria de la tabla esclavo (TCIUDAD) está formado por la llave primaria de la tala maestro (TPAIS) mas el campo identificador del esclavo.

CASO N° 04 CLASES QUE DERIVAN DE LA RELACIÓN DEL TIPO ASOCIACIÓN UNIDIRECCIONAL

Las llaves primarias de las tablas independientes serán campos foráneos en las tablas dependientes. Es criterio del analista de la base de datos, determinar si el nuevo campo foráneo (llave secundaria) será parte de llave primaria de la clase dependiente o no.

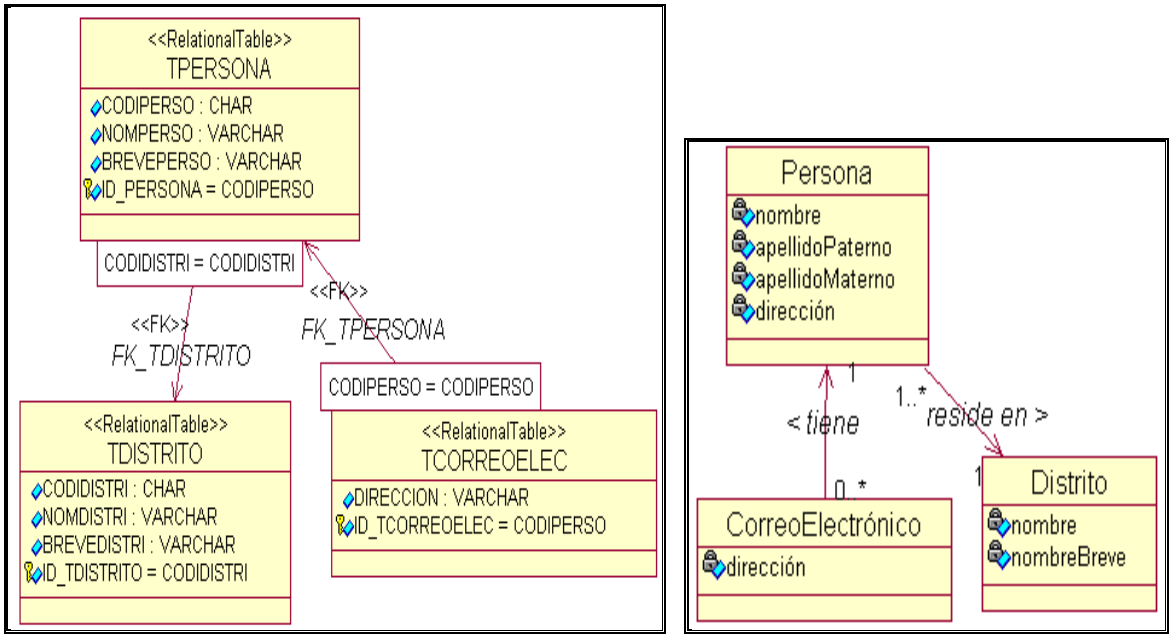


Figura N° 165, Nótese que la llave primaria de la clase TPERSONA no está determinada por el campo foráneo "CODIDISTRI"; pero la clase TCORREOELEC si está determinada por la llave foránea "CODIPERSO".

No olviden que la decisión acerca del contenido de las llaves primarias es responsabilidad y decisión del analista de la base de datos.

7.6.3.3. CREACIÒN DE CLASES CON EL ESTERIOTIPO TABLA RELACIONAL EN BASE AL WIZARD DEL ORACLE8 EN EL CASE DE MODELADO RATIONAL ROSE

PASO N° 01

Seleccionar la opción que señala el foco de la figura N° .

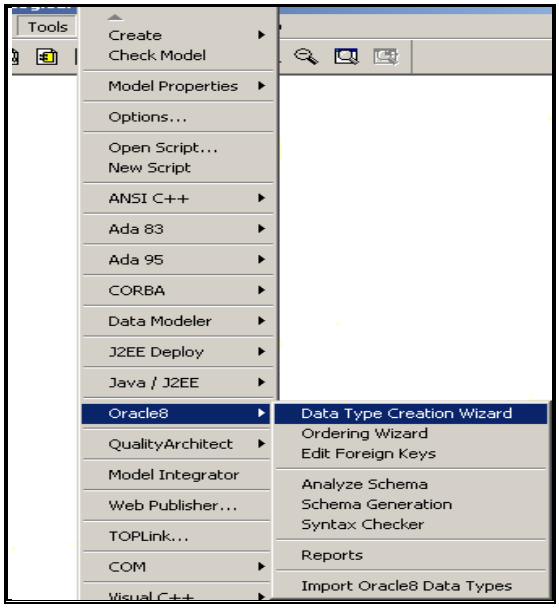


Figura N° 166, Esta figura muestra el ingreso a al opción que permite crear el tipo de dato "tabla relacional" en base a un modelo de objetos.

PASO N° 02

Seleccionar la opción que señala el foco de la figura N°.

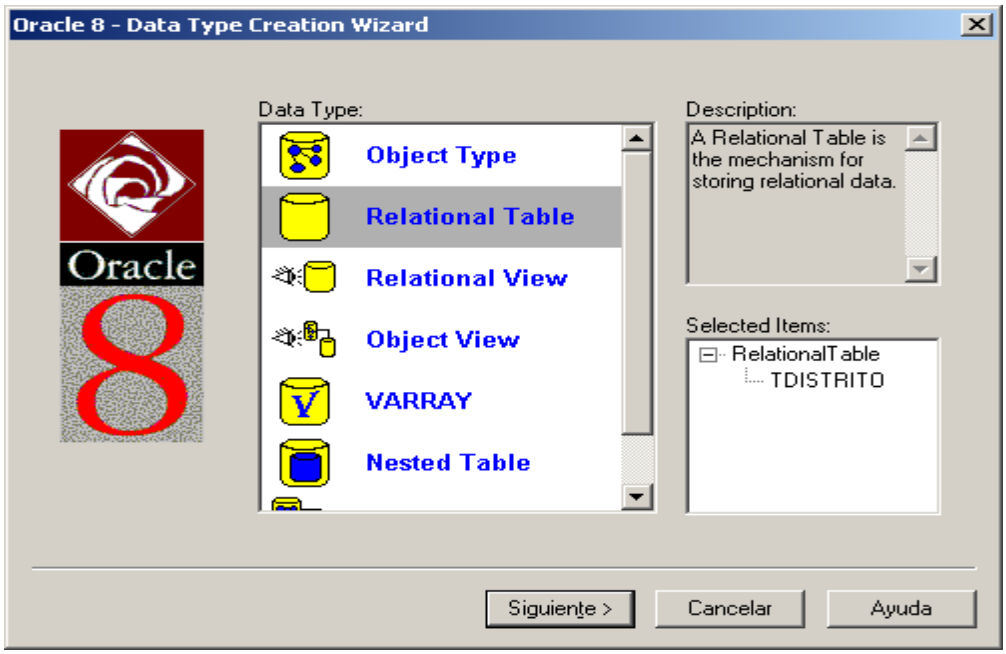


Figura N° 167, La figura muestra la selección del tipo de dato.

PASO N° 03

Colocar el nombre de la tabla, el "Schema", indica el nombre de la base de datos; por último seleccionar el entorno de trabajo donde se creará la clase con el estereotipo tabla relacional.

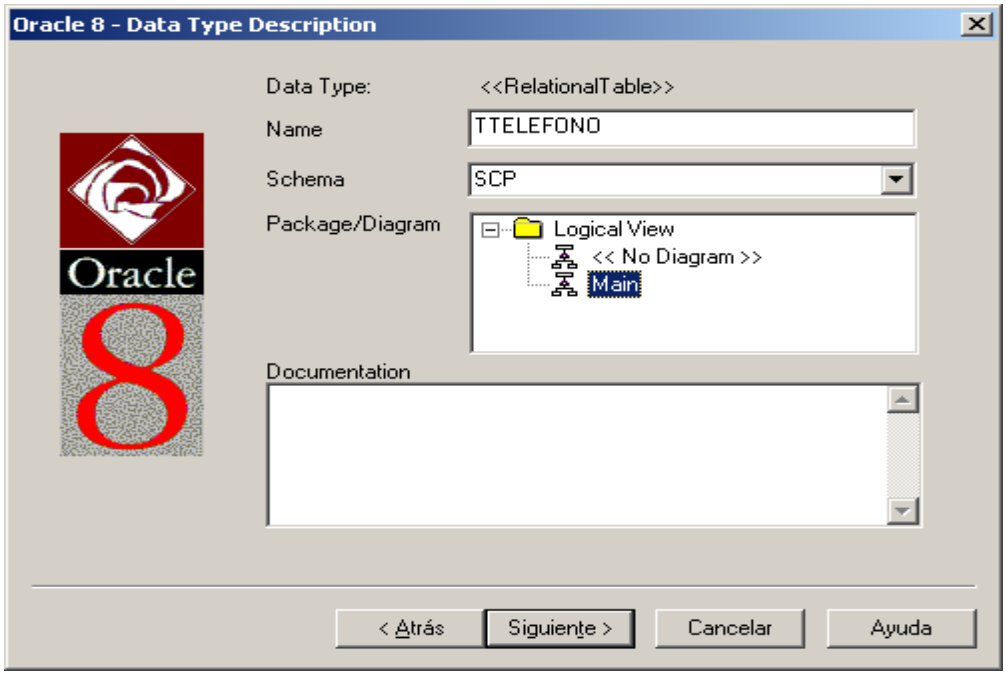


Figura N° 168, En la ventana anterior se establece el nombre, el esquema (nombre de la base de datos) y el entorno de trabajo donde se creará la nueva tabla relacional.

PASO N° 04

Crear cada uno de los campos de la tabla en base a lo establecido en el modelo lógico. En esta ventana se debe considerar el tipo de dato y la longitud del campo.

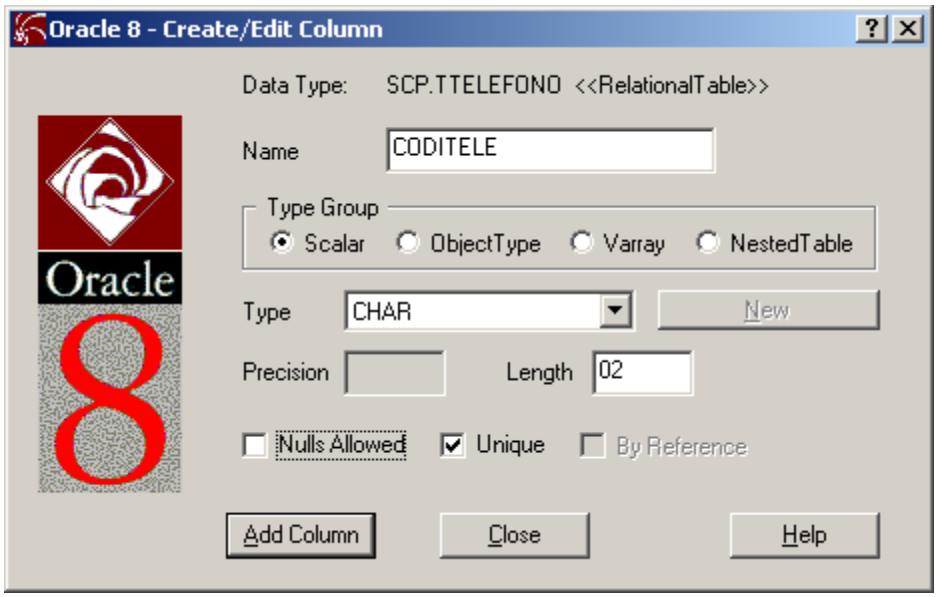


Figura N° 169, En esta ventana se determina el tipo de dato del campo incluyendo la longitud del mismo.

PASO N° 05

Señalar el campo que forma parte de la llave primaria de la tabla, para el ejemplo, la llave primaria de la tabla TTELEFONO es en campo CODITELE.

Figura N° 170, La figura muéstrala elección de la llave primaria de la tabla TTELEFONO

PASO N° 06

Una vez seleccionado el campo se procede a crear el índice como se indica en la figuara siguiente.

Figura N° 171, La figura muestra el proceso de creación de los índices de la tabla TTELEFONO

PASO N° 07

La siguiente ventana permite trabajar con los campos foráneos, para el ejemplo No es necesario por tratarse de una clase independiente.

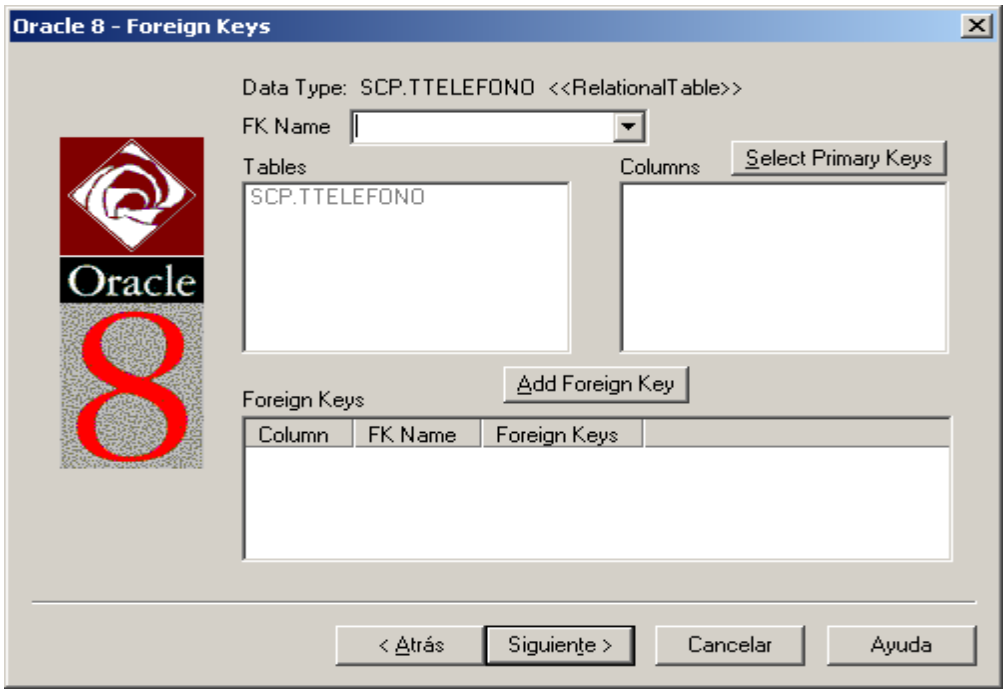


Figura N° 172, En esta ventana permite trabajar con los campos foráneos. La ventana No presenta campos foráneos por tratarse de una tabla independiente.

PASO N° 08

En la siguiente ventana, se procede a ordenar los campos, esta opción es usual cuando se trabaja con tablas dependientes, para el ejemplo referido a la tabla TTELEFONO no es necesario cambiar el orden de los campos.

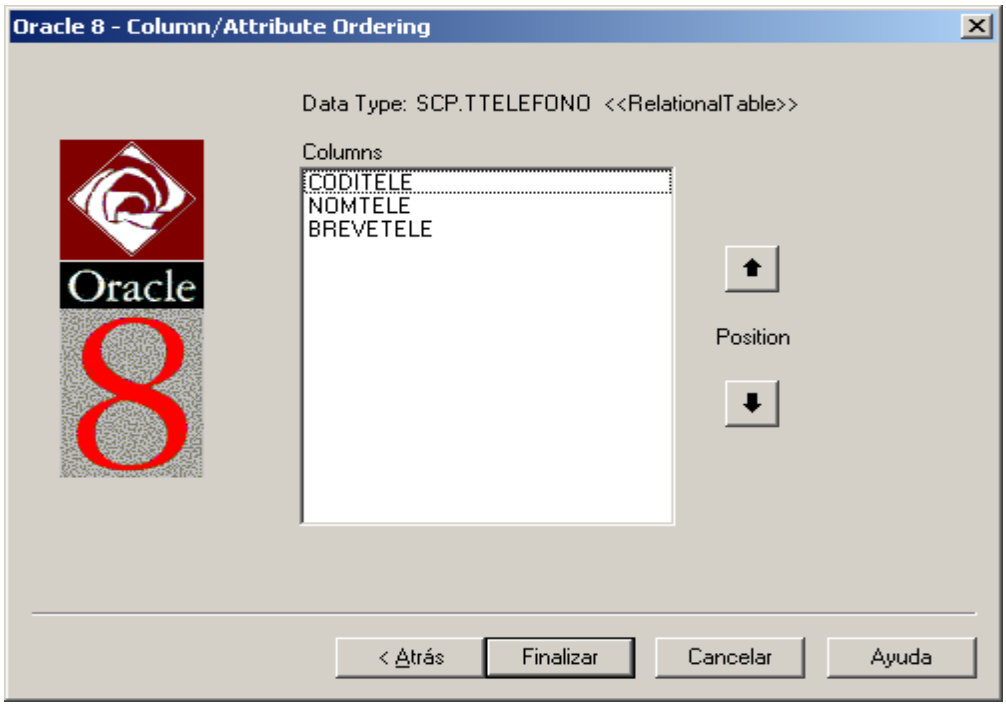


Figura N° 173, Esta ventana permite ordenar los campos de la tabla.

PASO N° 09

Finalmente, señalar como llave primaria al campo seleccionado. Este procedimiento es manual, se obtiene haciendo clic izquierdo en el campo índice y seleccionar el ícono en forma de llave.

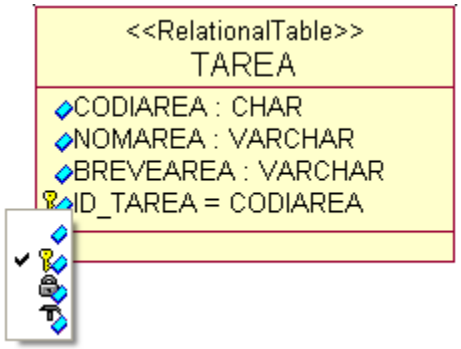


Figura N° 174, La figura muestra el proceso de identificación de la llave primaria de la tabla.

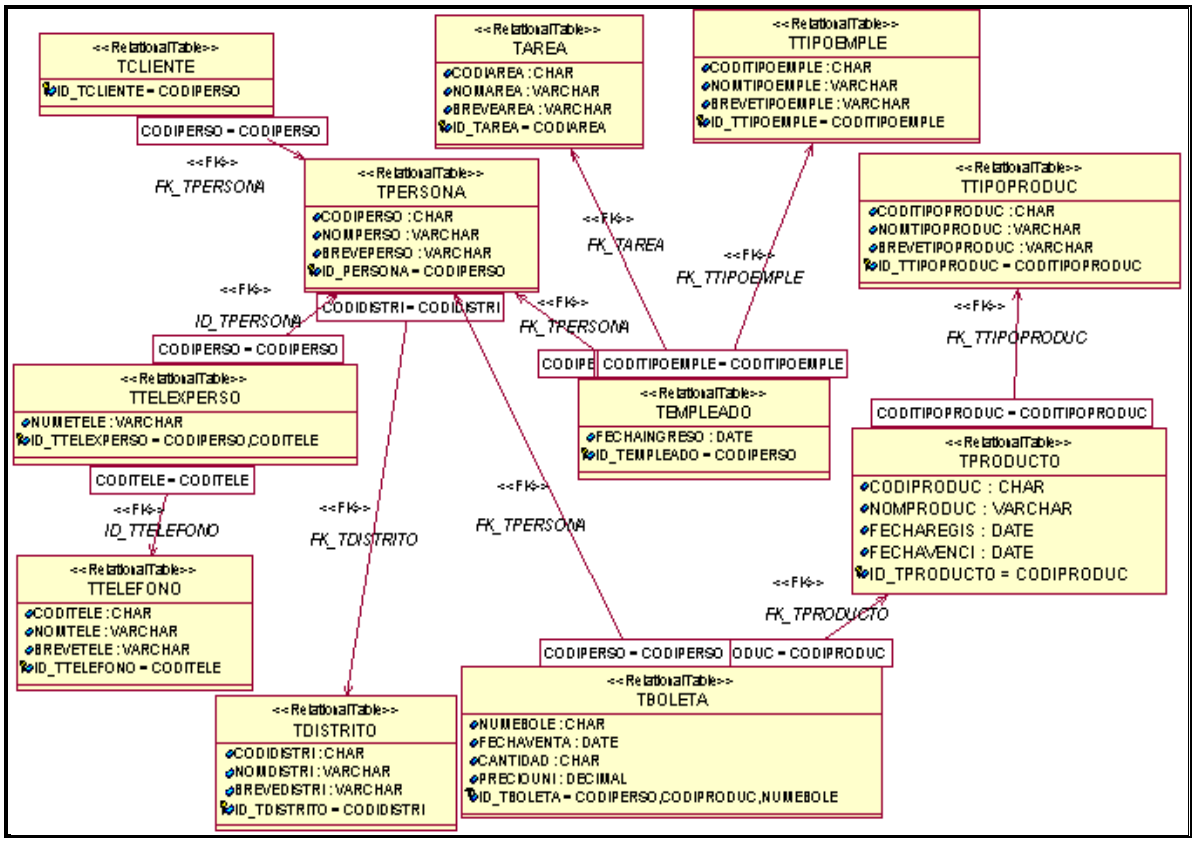


Figura N° 175, Modelo Físico del caso Carlitos's Pizza, realizado en base al wizard del Oracle8, del case de modelado Rational Rose.

7.6.3.4. GENERACIÓN DE CÓDIGO SQL DEL MODELO FÍSICO CARLITOS'S PIZZA

Al seguir la ruta Tools/Oracle8/Schema Generation, ingresar a la opción para la generación de código SQL de forma automática.

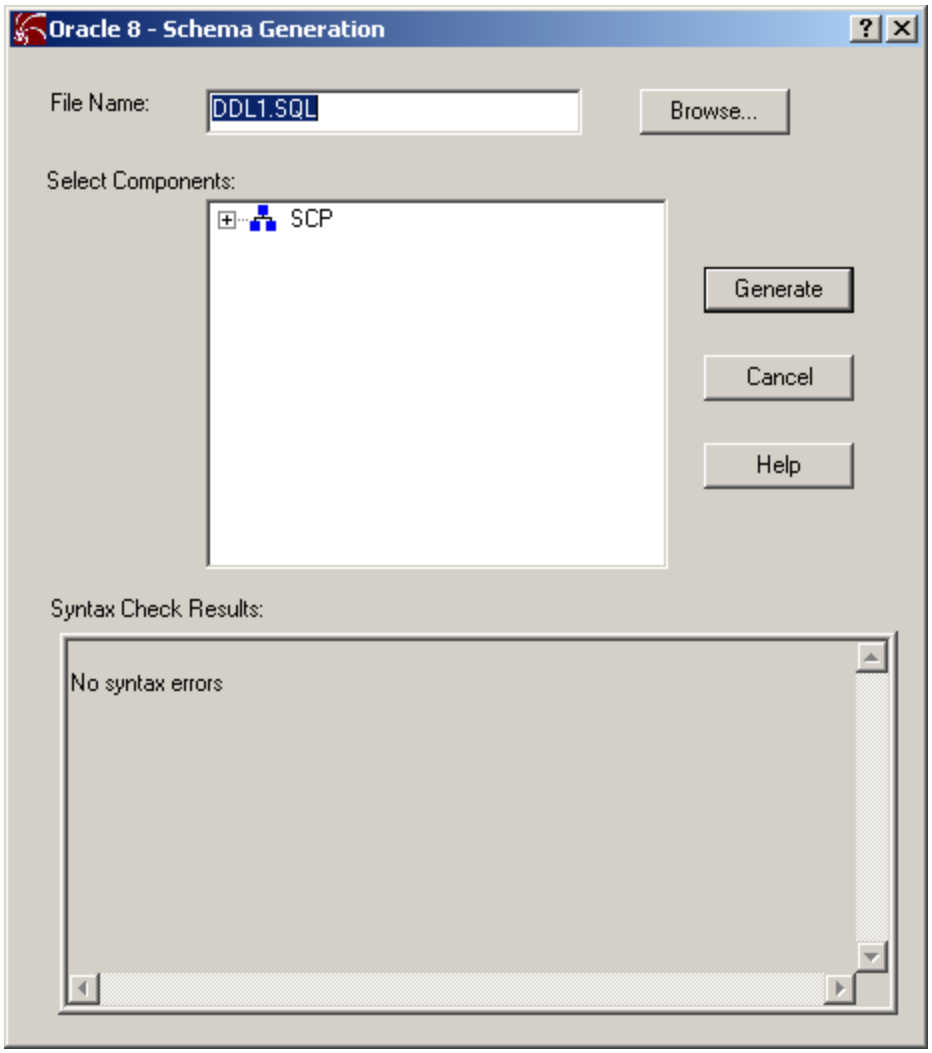


Figura N° 176, Ventana de generación de código SQL, nótese el mensaje "No syntax errors", el cual indica satisfacción en el modelo físico realizado

```
CREATE TABLE SCP.TTELEFONO (
     CODITELE CHAR(02) NOT NULL UNIQUE,
     NOMTELE VARCHAR(35) NOT NULL,
     BREVETELE VARCHAR(25) NOT NULL,
     CONSTRAINT ID_TTELEFONO PRIMARY KEY (CODITELE));

CREATE TABLE SCP.TDISTRITO (
     CODIDISTRI CHAR(03) NOT NULL UNIQUE,
```

```
      NOMDISTRI VARCHAR(35) NOT NULL,
      BREVEDISTRI VARCHAR(25) NOT NULL,
      CONSTRAINT ID_TDISTRITO PRIMARY KEY (CODIDISTRI));

CREATE TABLE SCP.TAREA (
      CODIAREA CHAR(03) NOT NULL UNIQUE,
      NOMAREA VARCHAR(35) NOT NULL,
      BREVEAREA VARCHAR(25) NOT NULL,
      CONSTRAINT ID_TAREA PRIMARY KEY (CODIAREA));

CREATE TABLE SCP.TTIPOEMPLE (
      CODITIPOEMPLE CHAR(02) NOT NULL UNIQUE,
      NOMTIPOEMPLE VARCHAR(35) NOT NULL,
      BREVETIPOEMPLE VARCHAR(25) NOT NULL,
      CONSTRAINT ID_TTIPOEMPLE PRIMARY KEY (CODITIPOEMPLE));

CREATE TABLE SCP.TTIPOPRODUC (
      CODITIPOPRODUC CHAR(02) NOT NULL UNIQUE,
      NOMTIPOPRODUC VARCHAR(35) NOT NULL,
      BREVETIPOPRODUC VARCHAR(25) NOT NULL,
      CONSTRAINT ID_TTIPOPRODUC PRIMARY KEY (CODITIPOPRODUC));

CREATE TABLE SCP.TPERSONA (
      CODIPERSO CHAR(06) NOT NULL UNIQUE,
      NOMPERSO VARCHAR(35) NOT NULL,
      BREVEPERSO VARCHAR(25) NOT NULL,
      CODIDISTRI CHAR(03),
      CONSTRAINT ID_PERSONA PRIMARY KEY (CODIPERSO),
      CONSTRAINT    FK_TDISTRITO    FOREIGN    KEY(CODIDISTRI)
REFERENCES SCP.TDISTRITO(CODIDISTRI));

CREATE TABLE SCP.TTELEXPERSO (
```

```
     NUMETELE VARCHAR(10) NOT NULL,
     CODITELE CHAR(02),
     CODIPERSO CHAR(06),
     CONSTRAINT ID_TTELEXPERSO PRIMARY KEY
(CODIPERSO,CODITELE),
     CONSTRAINT ID_TTELEFONO FOREIGN KEY(CODITELE) REFERENCES
SCP.TTELEFONO(CODITELE),
     CONSTRAINT ID_TPERSONA FOREIGN KEY(CODIPERSO) REFERENCES
SCP.TPERSONA(CODIPERSO));

CREATE TABLE SCP.TPRODUCTO (
     CODIPRODUC CHAR(06) NOT NULL UNIQUE,
     NOMPRODUC VARCHAR(35) NOT NULL,
     FECHAREGIS DATE NOT NULL,
     FECHAVENCI DATE NOT NULL,
     CODITIPOPRODUC CHAR(02),
     CONSTRAINT ID_TPRODUCTO PRIMARY KEY (CODIPRODUC),
     CONSTRAINT FK_TTIPOPRODUC FOREIGN KEY(CODITIPOPRODUC)
REFERENCES SCP.TTIPOPRODUC(CODITIPOPRODUC));

CREATE TABLE SCP.TCLIENTE (
     CODIPERSO CHAR(06),
     CONSTRAINT ID_TCLIENTE PRIMARY KEY (CODIPERSO),
     CONSTRAINT FK_TPERSONA FOREIGN KEY(CODIPERSO) REFERENCES
SCP.TPERSONA(CODIPERSO));

CREATE TABLE SCP.TEMPLEADO (
     FECHAINGRESO DATE NOT NULL,
     CODIPERSO CHAR(06),
     CODIAREA CHAR(03),
     CODITIPOEMPLE CHAR(02),
     CONSTRAINT ID_TEMPLEADO PRIMARY KEY (CODIPERSO),
```

```
     CONSTRAINT FK_TPERSONA FOREIGN KEY(CODIPERSO) REFERENCES
SCP.TPERSONA(CODIPERSO),
     CONSTRAINT FK_TAREA FOREIGN KEY(CODIAREA) REFERENCES
SCP.TAREA(CODIAREA),
     CONSTRAINT FK_TTIPOEMPLE FOREIGN KEY(CODITIPOEMPLE)
REFERENCES SCP.TTIPOEMPLE(CODITIPOEMPLE));

CREATE TABLE SCP.TBOLETA (
     CODIPERSO CHAR(06),
     CODIPRODUC CHAR(06),
     NUMEBOLE CHAR(06) NOT NULL UNIQUE,
     FECHAVENTA DATE NOT NULL,
     CANTIDAD CHAR(2),
     PRECIOUNI DECIMAL(2,6) NOT NULL,
     CONSTRAINT ID_TBOLETA PRIMARY KEY
(CODIPERSO,CODIPRODUC,NUMEBOLE),
     CONSTRAINT FK_TPRODUCTO FOREIGN KEY(CODIPRODUC)
REFERENCES SCP.TPRODUCTO(CODIPRODUC),
     CONSTRAINT FK_TPERSONA FOREIGN KEY(CODIPERSO) REFERENCES
SCP.TPERSONA(CODIPERSO));
```

7.6.4. PROCESO DE CONVERSIÒN DEL MODELO DE OBJETOS AL MODELO DE DATOS Y VICEVERSA

En la actualidad, muchas de las empresas nacionales e internacionales mantiene sistemas informáticos basados en la metodología estructurada, por ende utilizaron modelos entidad-relación (modelo de datos). Para facilitar el engorroso proceso de migración para la revisión, mantenimiento o una nueva propuesta es necesario el proceso

de conversión de un modelo de objetos al modelo de datos y viceversa.

Muchas herramientas de modelado que soportan UML, presentan alternativas de solución para este proceso. A mi criterio, la herramienta case Rational Rose de IBM es una de las alternativas más potentes para la migración del modelo de datos al modelo de objetos.

7.6.4.1. PROCESO DE MIGRACIÒN DE UN MODELO DE OBJETOS AL MODELO DE DATOS

A continuación explicaré paso a paso el proceso de migración, tendiendo en cuenta los criterios técnicos e ingenieriles que nos permitan optimizar el proceso de conversión.

PASO N° 01

Las clases 00del modelo lógico deben estar preparadas para la conversión a tablas relacionales. Las clases deben presentar atributos considerando el tipo de dato.

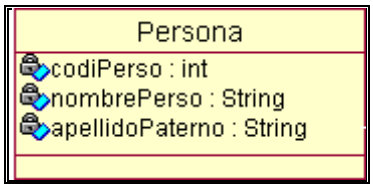


Figura N° 177, Nótese la característica de los campos y los tipos de dato

PASO N° 02

Las clases que se convertirán a tablas relacionales deben ser persistentes.

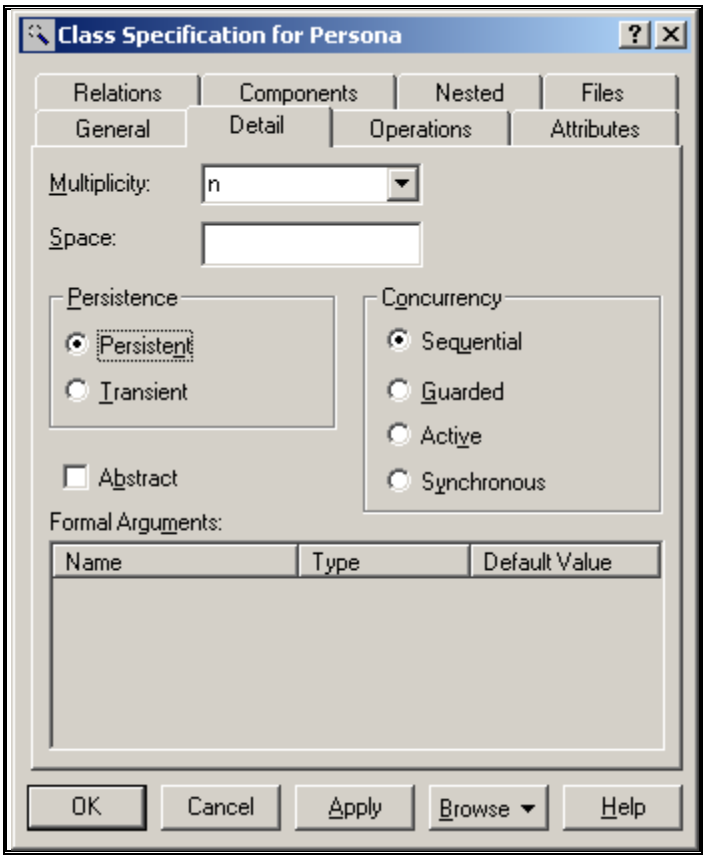


Figura N° 178, en la ventana de especificación de la clase Persona, seleccionar la opción "Persistent"

PASO N° 03

El atributo que formará parte de la llave primaria de la futura tabla relacional debe ser evaluado por su valor, se debe indicar manualmente que el campo elegido será el identificador principal de la tabla.

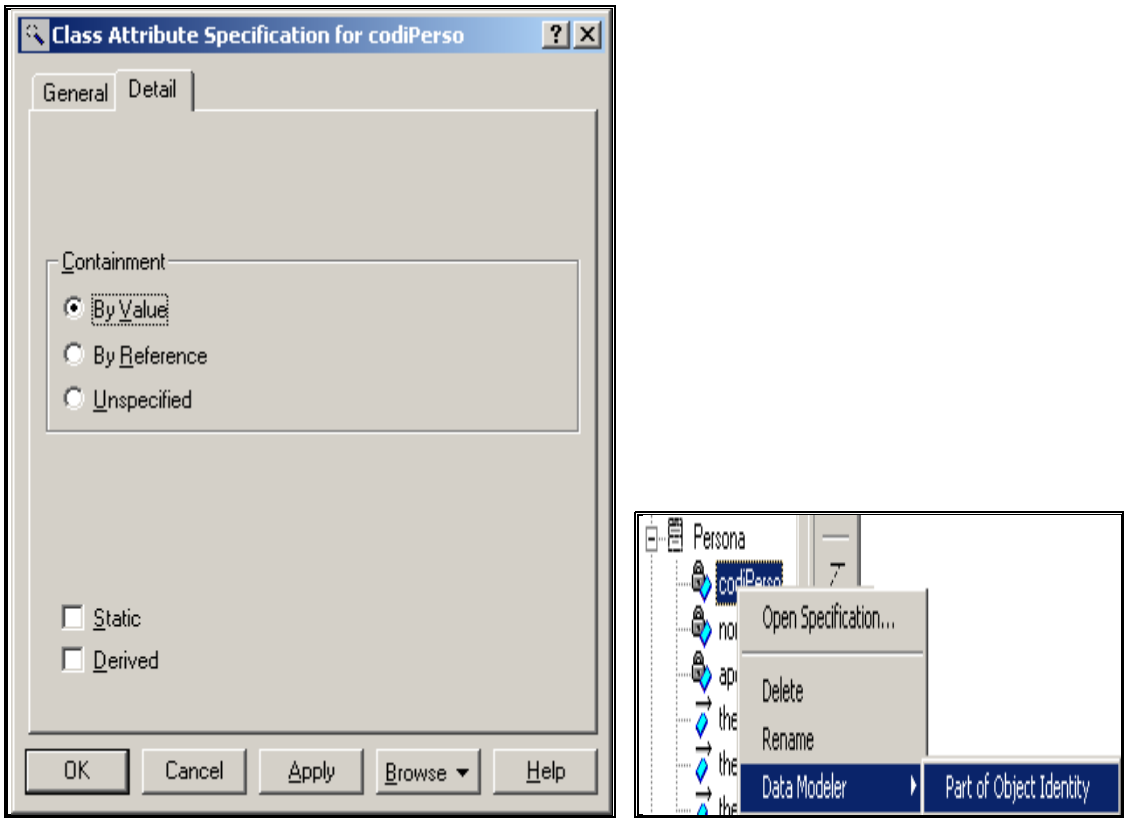


Figura N° 179, Se debe seleccionar la opción "By Value" para evaluar por el valor al campo identificador, además se debe seleccionar la llave primaria de la tabla en forma manual, a través de la opción "part of object identity".

PASO N° 04

Estimado lector, estamos listos para iniciar con el proceso de transformación del modelo lógico al modelo de datos.

Crear una nueva base de datos en el paquete de Componentes

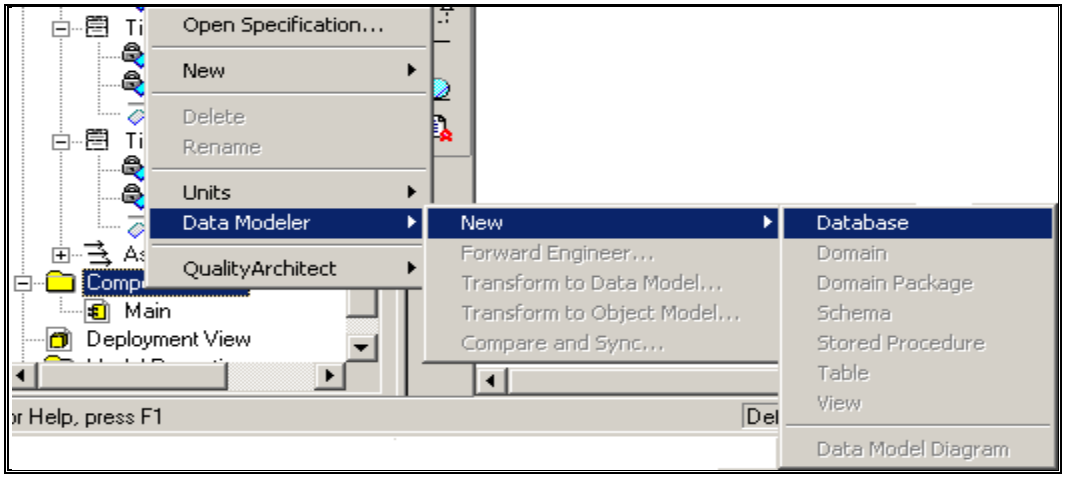


Figura N° 180, Creando una nueva base de datos en el paquete de componentes

PASO N° 05

A continuación se debe seleccionar el gestor de base de datos

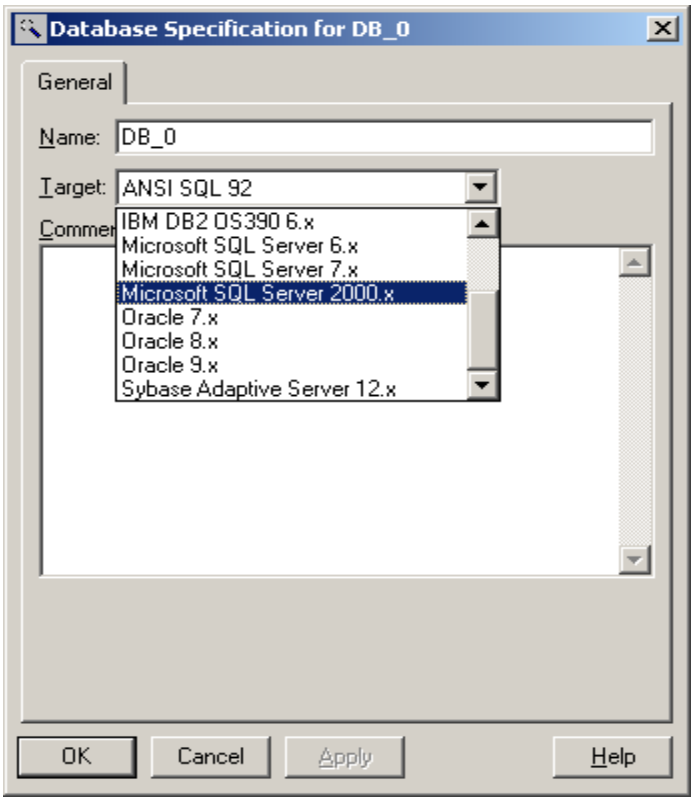


Figura N° 181, En esta figura se puede seleccionar el gestor de base de datos.

PASO N° 06

Se debe considerar un espacio de tabla virtual para completar el proceso de creación del componente.

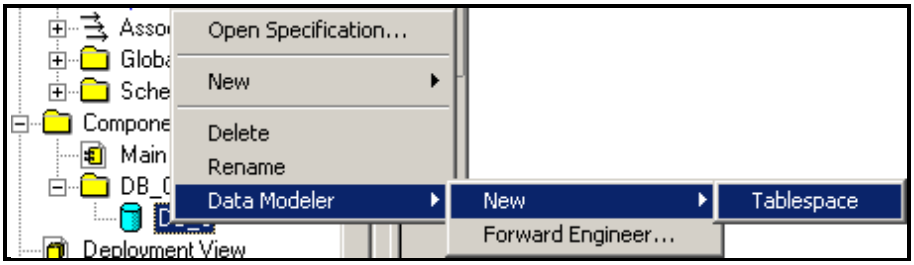


Figura N° 182, la ventana muestra el proceso de creación de un espacio de tabla virtual.

PASO N° 07

Estableciendo el espacio de tabla, la cantidad seleccionada es por defecto. El espacio de tabla, permite crear un espacio virtual en la memoria, esta condición es necesaria en el proceso de transformación del modelo de objetos al modelo de datos.

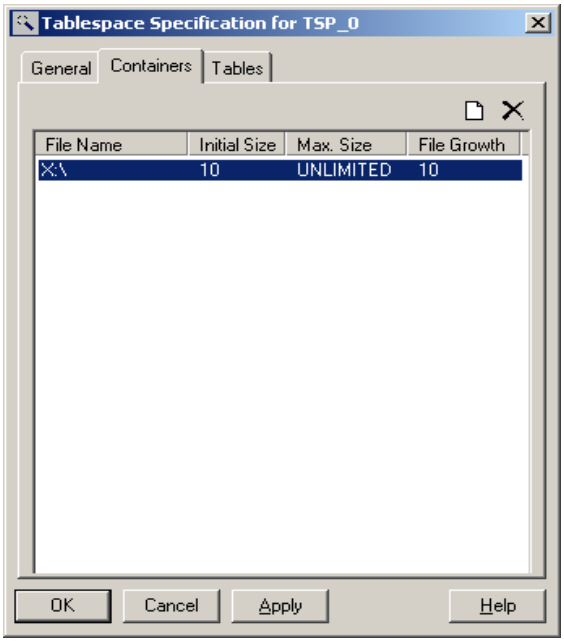


Figura N° 183, la ventana muestra el espacio de tabla.

PASO N° 08

Seleccionar el paquete que contiene a las clases del modelo lógico.

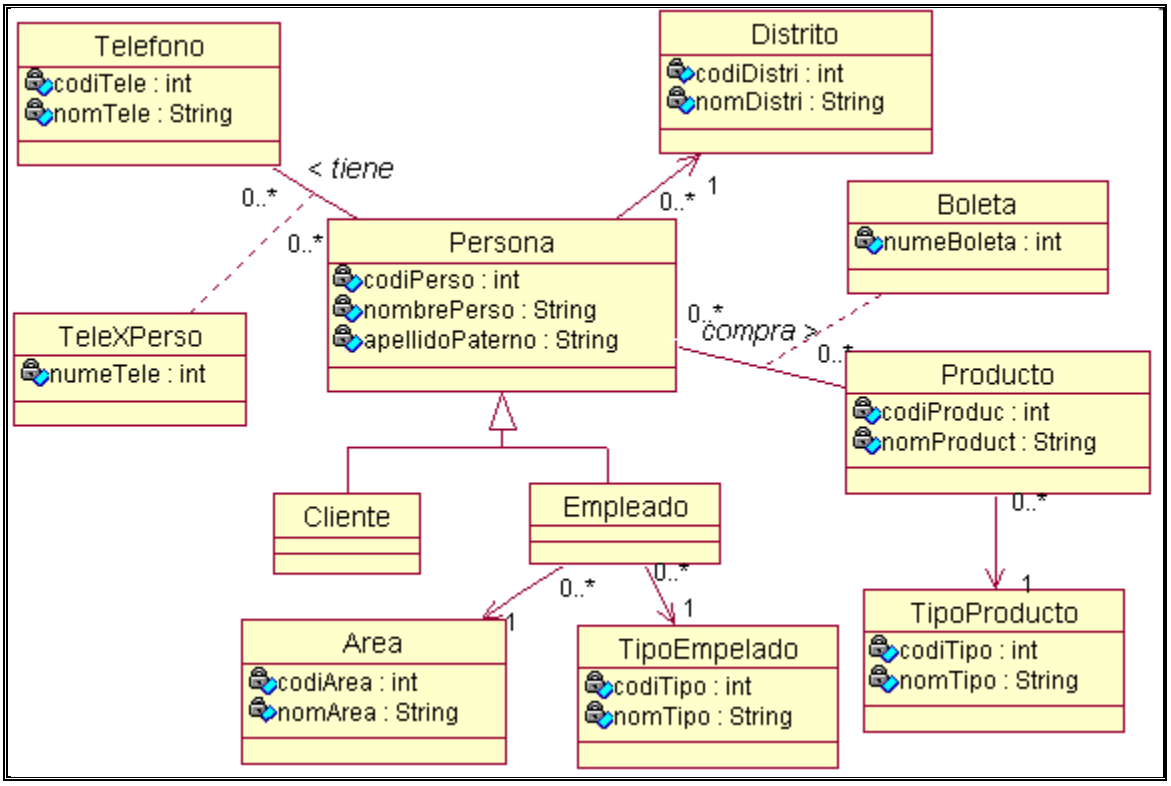


Figura N° 184, Modelo lógico. Nótese la característica de los campos, se considera llave primaria y tipo de dato en cada uno de los campos.

PASO N° 09

Seleccionar la opción "Transform to Data Model…"

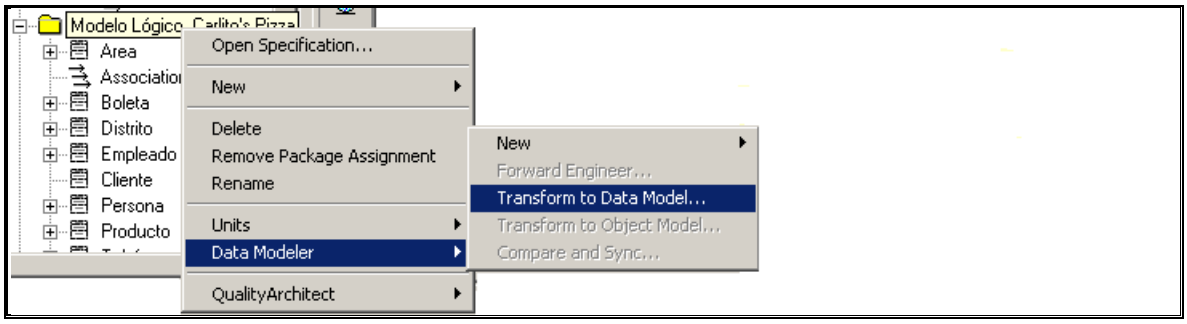


Figura N° 185, Proceso de transformación del modelo de objetos al modelo de datos.

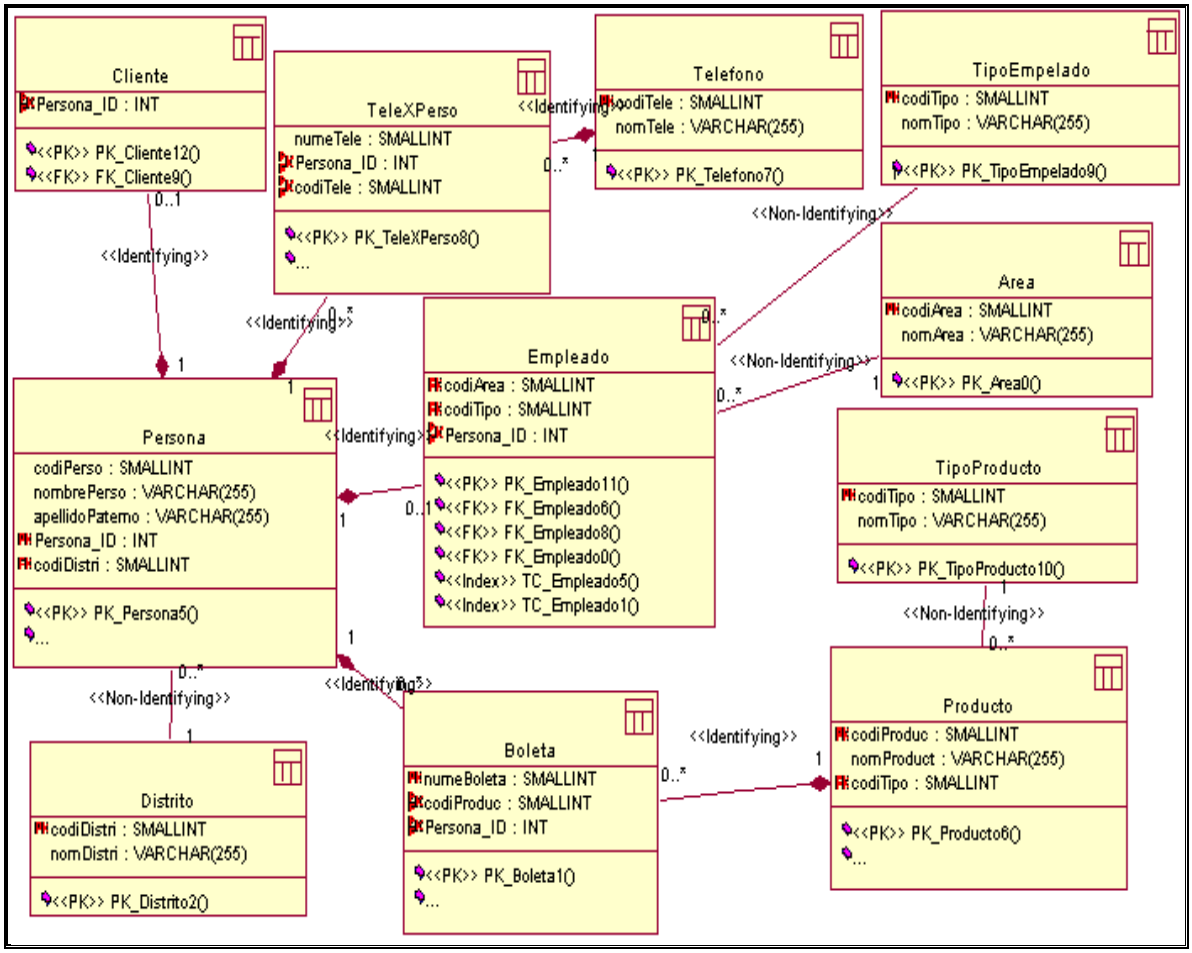


Figura N° 186, Modelo de datos del caso Carlitos's Pizza. Nótese la estructura de las llaves primarias de cada uno de las tablas.

7.6.4.2. GENERACIÓN DE CÓDIGO SQL A PARTIR DE UN MODELO DE DATOS

El proceso de generación de código se realiza con la opción: Data Modeler/Forward Engineer

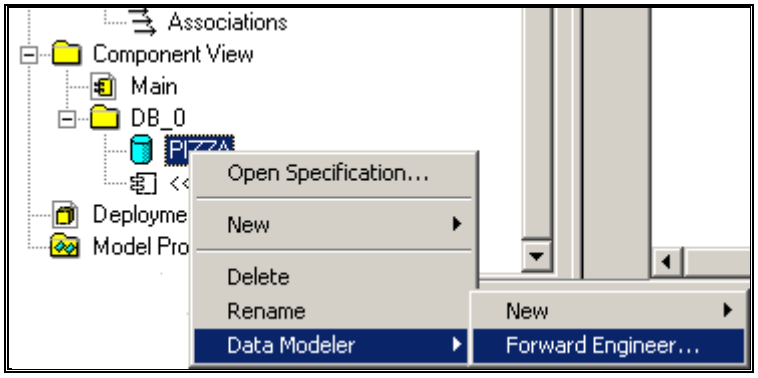


Figura N° 187, La Ventana anterior muestra la opción del case de modado Rational Rose, el cual permite la generación de código SQL.

Figura N° 188, Ventana de ayuda para la generación de código (tablas de la base de datos PIZZA)en el gestor de base de datos SQL/SERVER

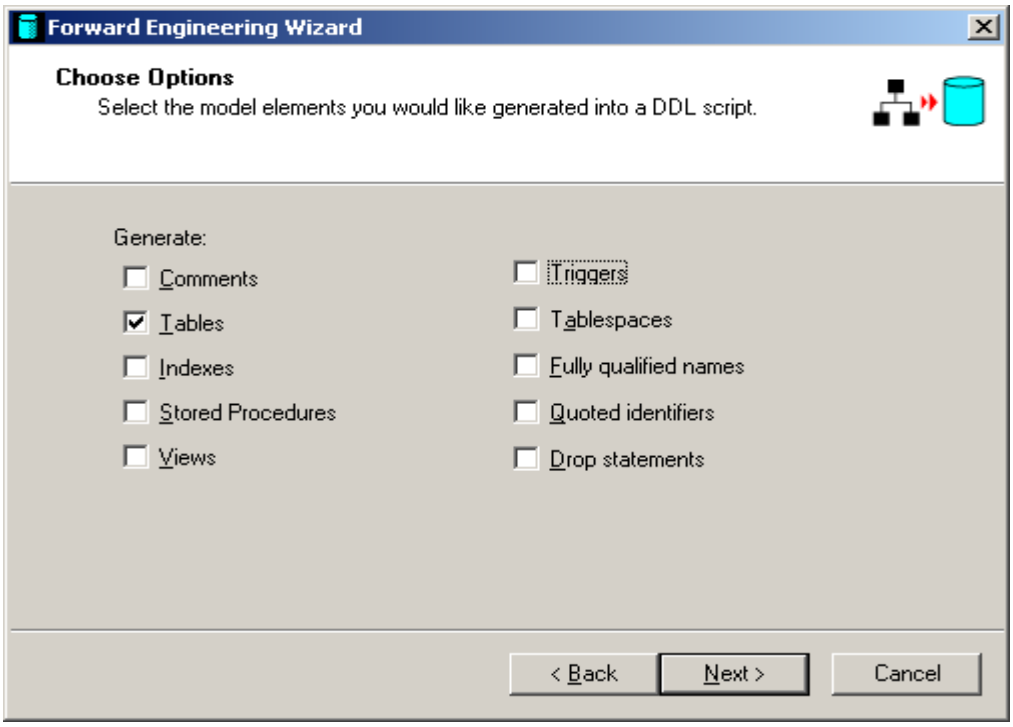


Figura N° 189, En esta ventana se selecciona los elementos del modelo que se desea generar

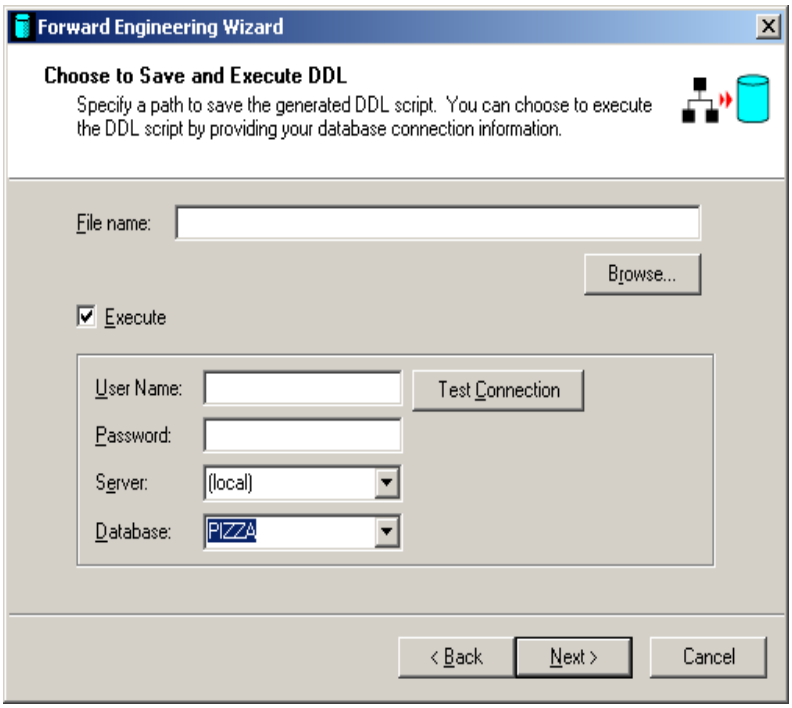


Figura N° 190, En esta ventana se nombra el servidor y la base de datos donde se crearán las tablas

Figura N° 191, Mensaje de satisfacción en el proceso de creación de tablas referido a la base de datos PIZZA

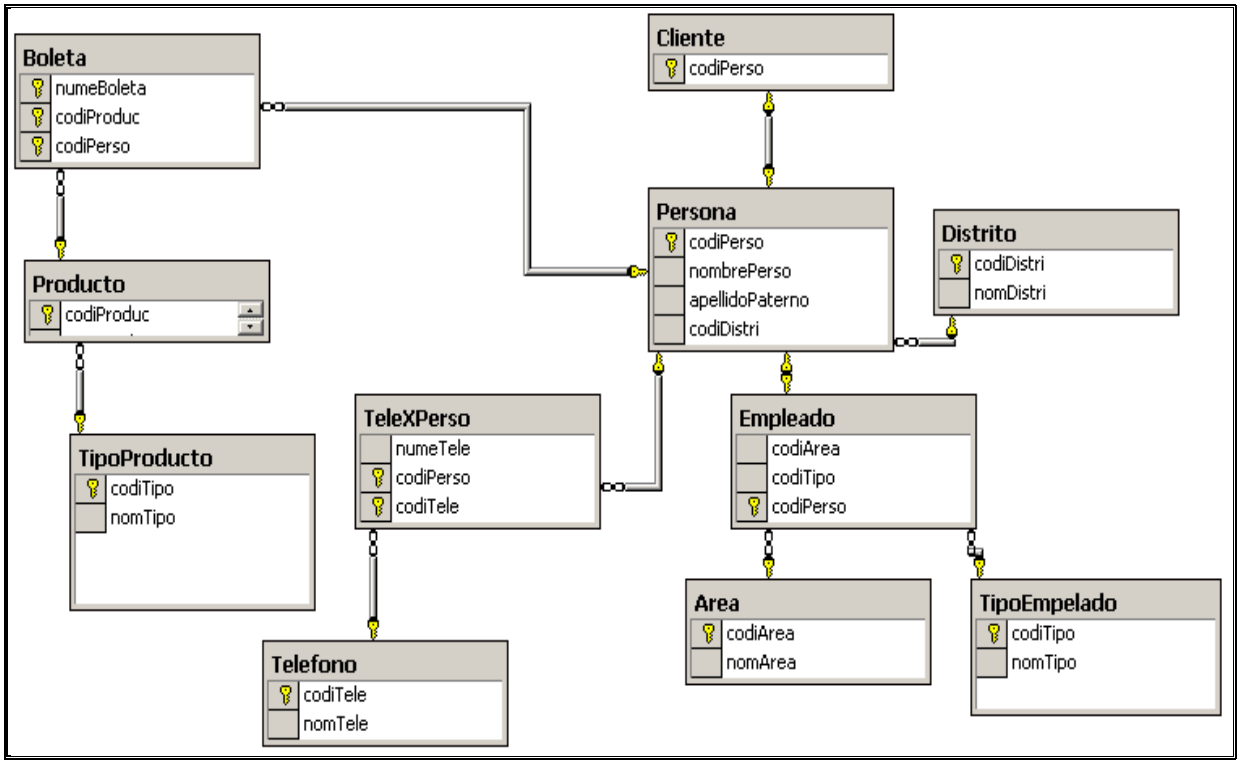


Figura N° 192, Diagrama realizado en el gestor de base de datos SQL/SERVER donde se muestra todas las tablas generadas automáticamente en el gestor ya mencionado.

7.7. DESARROLLO DEL ANÁLISIS Y DISEÑO DE LA BASE DE DATOS APLICADO AL CASO EMPRESA DE TRANSPORTES "TECSA"

7.7.1. MODELO CONCEPTUAL

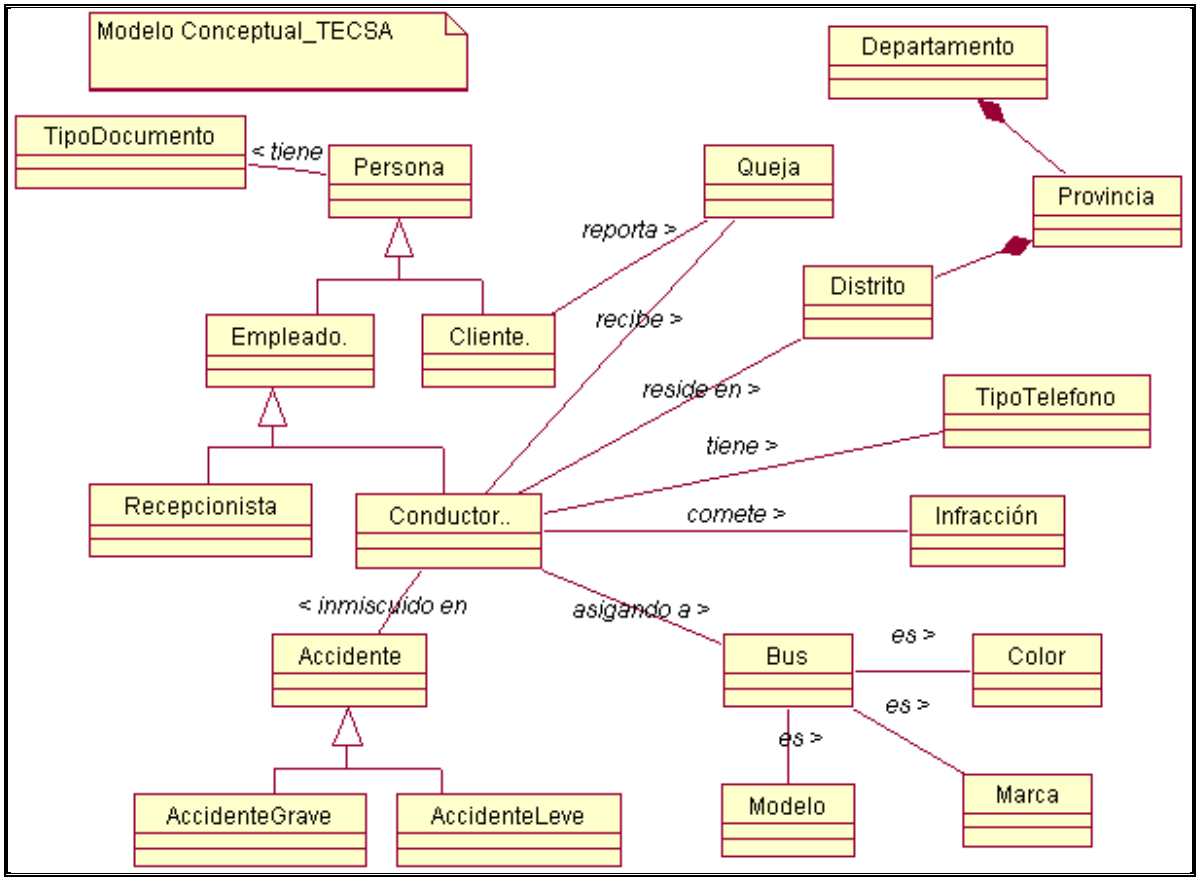


Figura N° 193, Modelo Conceptual aplicado al caso empresa de transportes TECSA.

7.7.2. MODELO LÓGICO

7.7.3. ORDEN DE CREACIÓN DE TABLAS

7.7.4. MODELO FÍSICO

7.7.6. MODELO DE DATOS (DATA MODELER)

7.8. RESUMEN DEL CAPÍTULO

En este capítulo revisamos todos los modelos necesarios para la construcción de la base de datos basado en el Proceso Unificado.

En análisis inicia desde un punto de vista básico dado por el *modelo conceptual* donde se muestran a las clases relevantes y a los principales tipos de relaciones entre clases.

En base al modelo conceptual se realizó el *modelo lógico* el cual es una propuesta técnica, destinado a eliminar la presencia de datos nulos y redundantes en la base de datos, por lo tanto la propuesta orientada a objetos ya está normalizada.

El siguiente modelo tratado lleva por nombre *modelo físico*, el cual fue construido en base a los criterios establecidos en el modelo lógico, este modelo es la propuesta física de la base de datos, en el cual se consideran los campos, tipos de dato, longitud y la estructura completa de las llaves primarias y secundarias.

El capítulo también mostró una propuesta de equivalencias entre la propuesta de objetos y la propuesta de datos; a través de un proceso de conversión existente en el case de modelado Rational Rose 2005, conocido como "*data modeler*".

Estimado lector, no olvide que el proyecto desarrollado al 100% se encuentra en el CD que acompaña al presente texto, nuevamente lo invito a revisar el mencionado CD.

7.9. EVALUACIÒN DE COMPETENCIAS DEL CAPÍTULO

- Realizar un cuadro comparativo entre los modelos conceptual y lógico.
- Detallar los significados de la relación "asociación unidireccional" en la base de datos.
- Realizar un ejemplo de aplicación del atributo de enlace.