CIBERTEC
Empieza por ti

Análisis y Diseño de Sistemas I Teoría

ÍNDICE

Anexo I: Plantillas de la disciplina modelado de negocio

Anexo II: Plantillas de la disciplina captura de requisitos

Glosario

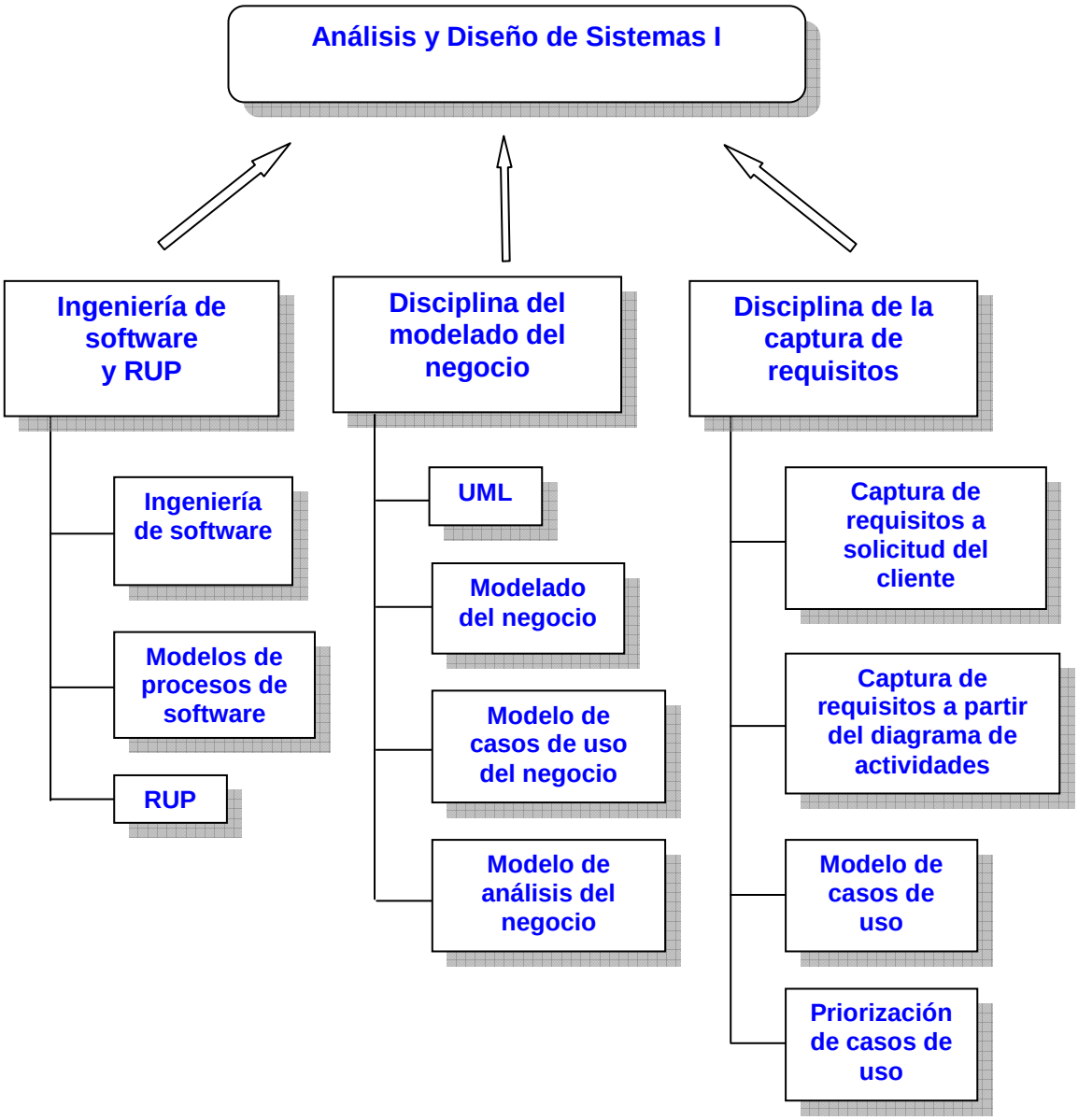
PRESENTACIÓN

Análisis y Diseño de Sistemas I pertenece a la línea formativa y se dicta en las carreras de Computación e Informática, Administración y Sistemas, Redes y Comunicaciones. El curso imparte conocimientos relacionados con el proceso de Ingeniería de Software Orientado a Objetos que permite a los alumnos utilizar una metodología y el Lenguaje de Modelamiento Unificado para desarrollar un software de calidad.

El manual para el curso ha sido diseñado bajo la modalidad de unidades de aprendizaje, las que se desarrollan durante semanas determinadas. En cada una de ellas, hallará los logros, que debe alcanzar al final de la unidad; el tema tratado, el cual será ampliamente desarrollado; y los contenidos, que debe desarrollar, es decir, los subtemas. Por último, encontrará las actividades que deberá desarrollar en cada sesión, que le permitirán reforzar lo aprendido en la clase.

El curso es eminentemente práctico: consiste en un taller de desarrollo de proyectos de software. En primer lugar, se inicia con la comprensión de la Ingeniería de Software y el proceso de desarrollo de software RUP. Continúa con la presentación del modelamiento visual y el lenguaje de modelamiento unificado UML. Luego, se desarrolla la disciplina del modelado del negocio. Por último, se concluye con el desarrollo de la disciplina de la captura de requisitos.

RED DE CONTENIDOS



UNIDAD DE APRENDIZAJE

1

INGENIERÍA DE SOFTWARE Y RUP

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno a través de un trabajo práctico describe las características, ventajas y desventajas de los modelos de proceso de software y la importancia de emplear la metodología RUP para modelar el ciclo de vida del desarrollo de un software.

TEMARIO

1. Ingeniería de Software
2. Modelos de proceso de software
3. Rational Unified Process (RUP)

ACTIVIDADES PROPUESTAS

1. Los alumnos elaboran un cuadro comparativo de ventajas y desventajas de los modelos de procesos de software.

2. Los alumnos identifican el modelo de proceso de software que podría utilizarse para desarrollar los sistemas propuestos.

1. INGENIERÍA DE SOFTWARE

Según el *DRAE*[1], ***ingeniería*** es el "Estudio y aplicación, por especialistas, de las diversas ramas de la tecnología.", también es la "Actividad profesional del ingeniero" y el término ***ingeniero*** lo define como "Persona que profesa la ingeniería o alguna de sus ramas."; mientras que, en el diccionario *El pequeño Larouse*, ***ingeniería*** es el "Conjunto de conocimientos y técnicas científicos aplicados a la invención, perfeccionamiento y utilización de la técnica industrial en todas sus dimensiones".

Por otro lado, en el DRAE encontramos que ***software*** es el "Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora."

Uniendo todas estas definiciones, podemos establecer que la Ingeniería de Software es el "Uso o aplicación de conocimientos y técnicas científicos para crear programas de computadora".

El término ingeniería del software se utilizó por primera vez en la *Software Engineering Conference* (SEC) de la OTAN celebrada en Garmisch, Alemania, en 1968, donde se discutía sobre la "crisis del software" de la época. Esta crisis era consecuencia de la informalidad que reinaba en el proceso del desarrollo de software de esos años, especialmente en grandes proyectos.

La Ingeniería de Software puede ser definida de múltiples maneras. Es por ello que existen muchas definiciones expuesta por autores acreditados que comenzaron en su momento a utilizar el término, entre ellos Bauer, Boehm, Zelkovitz y Sommerville y otras dadas por organismos internacionales profesionales de prestigio tales como IEEE[2] o ACM[3]. Más adelante la definición fue incluyendo el término de calidad, mejorando así la definición de la Ingeniería de Software.

Se ha elegido la definición utilizada por Roger Pressman, quién indica que la Ingeniería de Software es una tecnología multicapa. Como muestra la figura 1.1, cualquier enfoque de ingeniería, incluida Ingeniería del Software como lo indica el autor, debe apoyarse sobre un compromiso de organización de calidad. La calidad, según indica, es la concordancia del software producido con los requisitos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requisitos implícitos no establecidos formalmente, que desea el usuario.

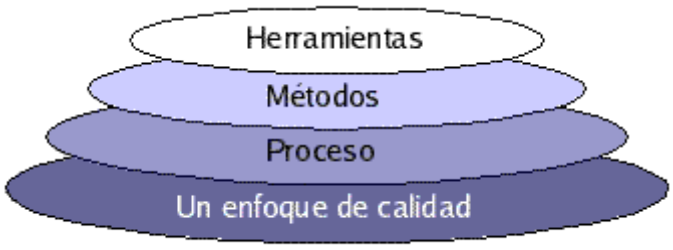


Figura 1.1. Capas de la Ingeniería de software

[1] DRAE, Diccionario de la Real Academia Española de la Lengua.
[2] IEEE, Institute of Electrical and Electronics Engineers.
[3] ACM, Association for Computing Machinery.

El fundamento de la Ingeniería del Software es la capa de proceso. Este proceso es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la Ingeniería del Software. El proceso define un marco de trabajo para un conjunto de áreas clave de proceso que se deben establecer para la entrega efectiva de la tecnología de la Ingeniería del Software. Las áreas claves del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.

Los métodos de la Ingeniería del Software indican «cómo» construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Estos métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

Las herramientas de la Ingeniería del Software proporcionan un enfoque automático o semiautomático para el proceso y para los métodos. Cuando se integran herramientas para que la información creada por una herramienta la pueda utilizar otra, se establece un sistema de soporte para el desarrollo del software llamado Ingeniería del software asistida por computadora (CASE).

Luego de describir cada capa, se puede afirmar que el objetivo de la Ingeniería de Software es lograr productos de software de calidad (tanto en su forma final como durante su elaboración), mediante un proceso apoyado por métodos y herramientas.

1.1. Elementos claves de la Ingeniería de Software

La Ingeniería de Software incluye los siguientes elementos clave: paradigmas, estrategias, métodos o técnicas, herramientas y procesos, los que a continuación se detallan.

1.1.1. Paradigmas

Un paradigma representa un enfoque particular o filosofía para la construcción de software. Cada uno tiene ventajas y desventajas, es por ello que hay situaciones donde un paradigma resulta más apropiado que otro.

1.1.2. Estrategias

Se denominan estrategias para el desarrollo de software a las distintas maneras en que se ordena la ejecución de las actividades requeridas para producir software.

1.1.3. Métodos o técnicas

Los métodos o técnicas indican cómo construir el software técnicamente y abarcan un amplio espectro de actividades que incluyen planificación y estimación de proyectos, análisis de requisitos y del sistema de software, diseño de la arquitectura de software, codificación, documentación, prueba y mantenimiento.

1.1.4. Herramientas

Son instrumentos que suministran un soporte automático o semiautomático para el proceso y para los métodos. Cuando se integran las herramientas de forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte de desarrollo del software, llamado ingeniería del software asistida por computadora (CASE, en inglés). CASE combina software, hardware y bases de datos sobre la Ingeniería del software (una estructura de datos que contenga la información relevante sobre el análisis, diseño, codificación y prueba) para crear un entorno de ingeniería del software análogo al diseño/ingeniería asistido por computadora, CAD/CAE para el hardware.

1.1.5. Procesos

Los procesos son la combinación de estrategias, métodos y herramientas que, en forma conjunta, dan un resultado particular. Los procesos indicarán qué herramientas deberán utilizarse y cuándo se aplican determinados métodos o técnicas. Definen la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que aseguren la calidad y las mejores prácticas que permiten a los gestores a evaluar los progresos. Concretamente, el proceso de desarrollo de software define quién va a hacer qué, cuándo hacerlo y cómo alcanzar un cierto objetivo.

1.2. Las fases genéricas de un proceso de software

Con independencia del área de aplicación, tamaño o complejidad del proyecto, el trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas:

- Fase de definición
- Fase de desarrollo
- Fase de mantenimiento

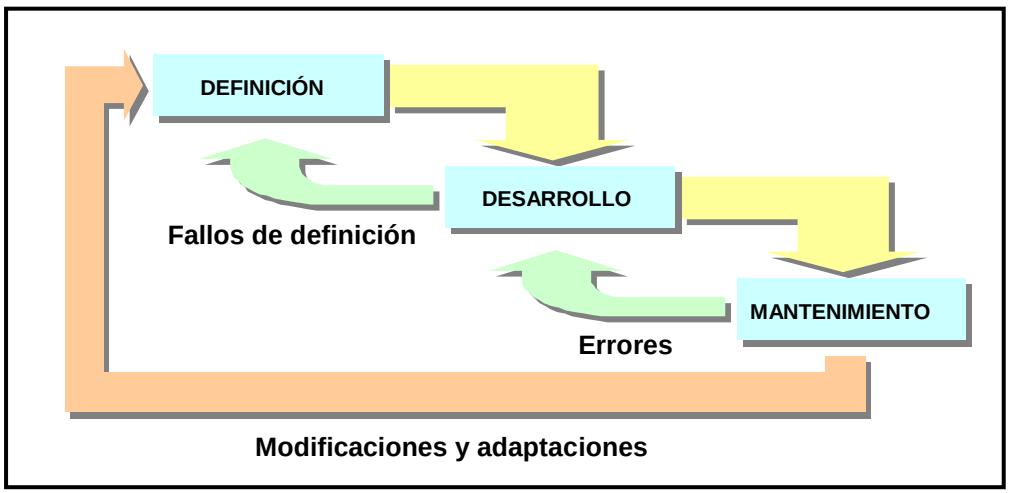


Figura 1.2. Fases genéricas de un proceso de software

1.2.1. Fase de definición

Se centra sobre el **qué**. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. Por tanto, han de identificarse los requisitos clave del sistema y del software. Aunque los métodos aplicados durante la fase de definición variarán dependiendo del paradigma de ingeniería del software (o combinación de paradigmas) que se aplique, de alguna manera tendrán lugar tres tareas principales:

- La ingeniería de sistemas o de información
- Planificación del proyecto del software
- Análisis de requisitos

1.2.2. Fase de desarrollo

Se centra en el **cómo**. Es decir, durante el desarrollo, un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba. Los métodos aplicados durante la fase de desarrollo variarán, aunque las tres tareas específicas técnicas deberían ocurrir siempre:

- El diseño del software
- La generación de código
- Las pruebas del software

1.2.3. Fase de mantenimiento

Se centra en el **cambio** que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. Durante la fase de mantenimiento, se encuentran cuatro tipos de cambios:

- Correctivo. Para corregir los defectos
- Adaptativo. Para acomodarlo a los cambios de su entorno externo (modificaciones en la legislación, CPU, SO, las reglas de negocio, etc.)
- Perfectivo. Para agregar otras funciones adicionales que van a producir beneficios
- Preventivo. También llamado reingeniería del software. Estos cambios se realizan con la finalidad de que se puedan corregir, adaptar y mejorar más fácilmente los programas.

Además de estas actividades de mantenimiento, los usuarios de software requieren un mantenimiento continuo. Los asistentes técnicos a distancia, teléfonos de ayuda y sitios Web de aplicaciones específicas se implementan frecuentemente como parte de la fase de mantenimiento.

1.2.4. Actividades de protección

Las fases descritas en esta visión general de la ingeniería del software se complementan con un número de actividades protectoras. Entre las actividades típicas de esta categoría se incluyen:

- Seguimiento y control del proyecto de software
- Revisiones técnicas formales
- Garantía de calidad del software
- Gestión de configuración del software
- Preparación y producción de documentos
- Gestión de reutilización
- Mediciones
- Gestión de riesgos

2. MODELOS DE PROCESOS DE SOFTWARE

Para resolver los problemas reales de una industria, un ingeniero del software o un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos, herramientas y fases genéricas descritos en los puntos anteriores. Esta estrategia a menudo se llama *Modelo de Proceso o Paradigma de Ingeniería del Software*. Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren. Existen muchos modelos de procesos para la ingeniería de software, entre ellos:

- Modelo lineal secuencial
- Modelo de construcción de prototipos
- Modelo DRA (Desarrollo Rápido de Aplicaciones)
- Modelos evolutivos de procesos de software:
  - El modelo incremental
  - El modelo espiral
  - El modelo de desarrollo concurrente
- Desarrollo basado en componentes

2.1. Modelo lineal secuencial

Llamado también ciclo de vida básico o modelo en cascada, el cual propone un enfoque sistemático, secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. Es ideal para proyectos pequeños, rígidos, y donde se especifiquen muy bien los requisitos.

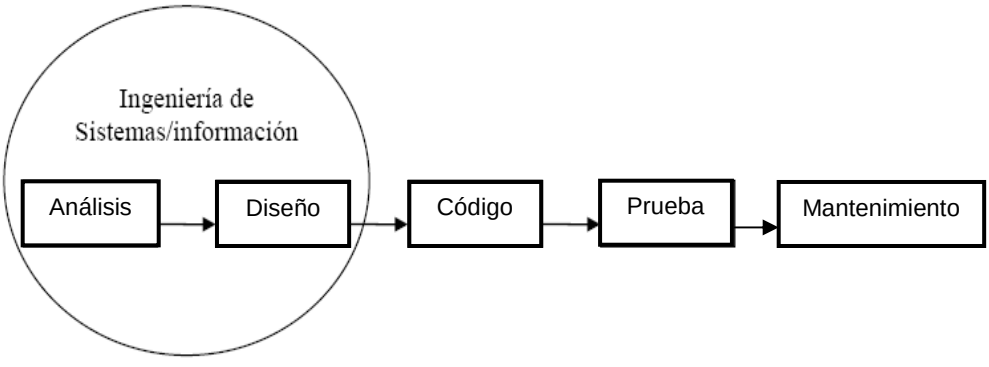


Figura 1.3. Modelo lineal secuencial

Existen algunos problemas que ocurren al utilizar este modelo:

- Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo, pues traslapan las etapas.
- A menudo, es difícil que el cliente exponga, explícitamente, todos los requisitos. El interesado debería exponer los requisitos en la etapa inicial, pero, en realidad, él lo hace a través de todo el proceso lo cual complica las cosas.
- El cliente debe tener paciencia. La primera versión del software llega al final del proceso. A veces, el afán del cliente hace que la aplicación final no cumpla con los requisitos

2.2. Modelo de construcción de prototipos

Este modelo comienza con la recolección de requisitos, el desarrollador y el cliente definen los objetivos globales para el software, originándose un diseño rápido que se centra en una representación de esos aspectos del software que sean visibles para el usuario/cliente. De este diseño surge la construcción de un prototipo y este es evaluado por el cliente/usuario. La interacción ocurre cuando el prototipo satisface las necesidades del cliente.

Con este modelo se reduce el riesgo de construir productos que no satisfagan las necesidades del usuario. Por otro lado, reduce costos y aumenta la probabilidad de éxito. Pero el problema es que el cliente se sienta decepcionado por no permitirle usar la primera versión del prototipo o que el desarrollador se sienta tentado en aumentar el prototipo para construir el sistema final sin tener en cuenta cuestiones de calidad.

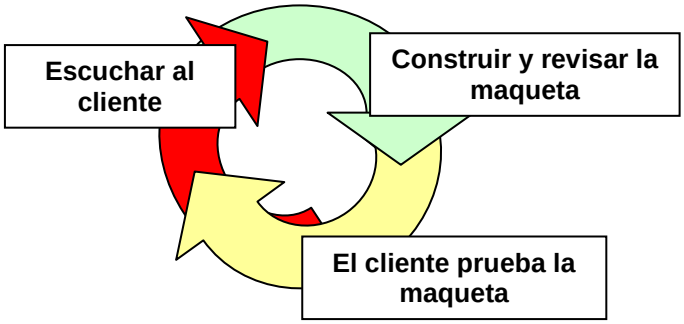


Figura 1.4. Modelo de construcción de prototipos

2.3. Modelo DRA

Es una adaptación a alta velocidad del modelo lineal secuencial, en el que se logra el desarrollo rápido, con un ciclo extremadamente corto, de proyectos grandes.

La velocidad es lograda gracias a un enfoque de construcción basado en el componente y al empleo de Técnicas de Cuarta Generación (T4G), así como a la posibilidad de modularización del sistema (cada una de las funciones puede ser afrontada por un equipo separado que trabaja en paralelo, y finalmente ser integradas en un solo producto).

Si no existe el compromiso en tiempo entre clientes y desarrolladores o si los riesgos técnicos son altos, los proyectos DRA fracasan. Por otro lado, para proyectos grandes, se requiere de un gran número de personas como para poder crear un número de equipos paralelos suficiente.

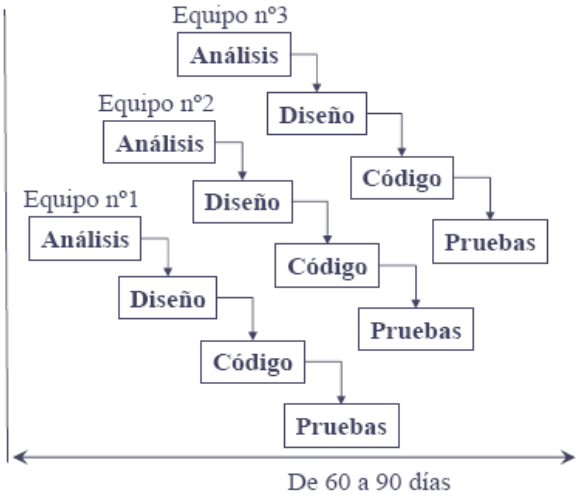


Figura 1.5. Modelo de construcción de prototipos

2.4. Modelo incremental

Este modelo combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos; viene a suplir el problema de no poder retroceder en las fases de desarrollo del software.

El primer incremento es un producto esencial, se afrontan requisitos básicos, pero muchas funciones suplementarias quedan sin extraer. El cliente utiliza el producto central y como resultado de utilización o evaluación, se desarrolla un plan para el incremento siguiente, este plan afronta la modificación del producto central para lograr satisfacer al cliente, la entrega de funciones y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

Este modelo es apropiado para proyectos de larga duración que no consuman muchos recursos y, como el producto va desarrollándose incrementalmente, se puede financiar el proyecto por partes.

Debido a la interacción con usuarios finales cuando es necesaria la retroalimentación hacia el grupo de desarrollo, este proceso puede exigir demasiado tiempo, agregándose un costo extra a la compañía, pues mientras estos usuarios evalúan el software dejan de ser productivos para la compañía.

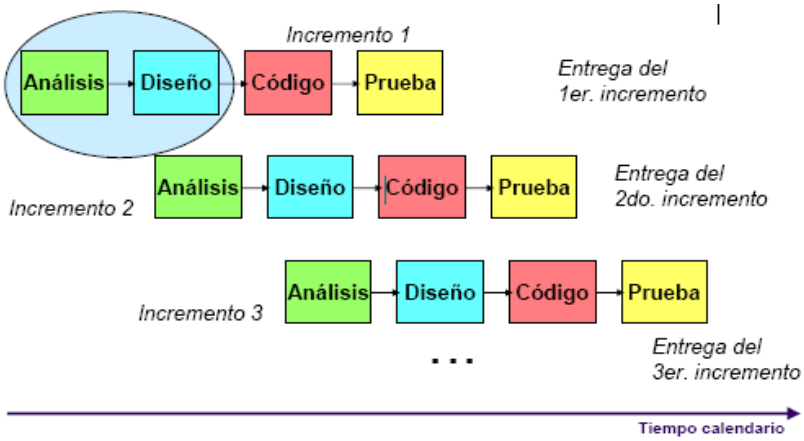


Figura 1.6. Modelo incremental

2.5. Modelo espiral

Es un modelo de proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo; en las últimas iteraciones, se producen versiones cada vez más completas de ingeniería del sistema. Este modelo se divide en un número de actividades estructurales o regiones de tareas, como comunicación con el cliente, planificación, análisis de riesgos, ingeniería, construcción y adaptación, evaluación del cliente.

Debido a su complejidad, no se aconseja utilizarlo en pequeños sistemas. Por otro lado, resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable.

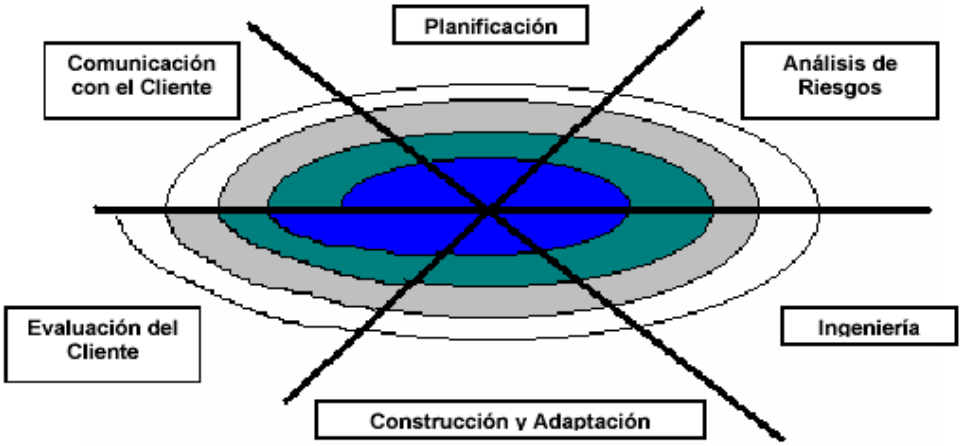


Figura 1.7. Modelo incremental.

2.6. Modelo de desarrollo concurrente

Llamado, algunas veces, ingeniería concurrente. Es un modelo de tipo de red donde todas las personas actúan simultáneamente. Provee una meta descripción del proceso de software. Mientras que en el espiraln la principal contribución es que las actividades del software ocurran repetidamente, en el concurrente, es la capacidad de describir las múltiples actividades del software que ocurren simultáneamente.

El modelo de proceso concurrente define una serie de acontecimientos que dispararán transiciones de estado a estado para cada una de las actividades.

Dado que los requisitos cambian, es muy probable que, una vez haya comenzado la fase de diseño, sea necesario incorporar cambios. En estos casos, no se debe detener el diseño, sino que se debe continuar, “si es posible”, al mismo tiempo que se modifican los requisitos. Por lo tanto, en este modelo, diversas actividades pueden estar ocurriendo concurrentemente.

La siguiente figura proporciona una representación esquemática de una actividad dentro del modelo de proceso concurrente. Por ejemplo, al principio del proyecto, la actividad de comunicación con el cliente (no mostrada en la figura) ha finalizado su primera interacción y existe en el estado de **cambios en espera**. La actividad de análisis (que existía en el estado **ninguno,**

mientras que comenzaba la comunicación inicial con el cliente) ahora hace una transición al estado **bajo desarrollo**. Sin embargo, si el cliente indica que se deben hacer cambios en requisitos, la actividad análisis cambia del estado **bajo desarrollo** al estado **cambios en espera**.

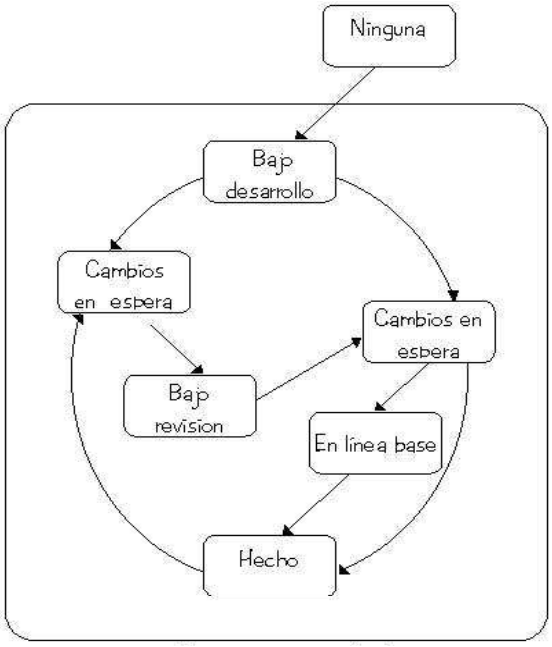


Figura 1.8. Modelo de desarrollo concurrente

2.7. Desarrollo basado en componentes

Es un modelo fuertemente orientado a la reutilización y trabaja sobre la base de tecnologías orientado a objetos. Este modelo consta de 4 fases ilustradas en la figura 1.9. A continuación, se describe cada fase:

- Análisis de componentes: Se determina qué componentes pueden ser utilizados para el sistema en cuestión. Casi siempre, hay que hacer ajustes para adecuarlos.
- Modificación de requisitos: Se adaptan (en lo posible) los requisitos para concordar con los componentes de la etapa anterior. Si no se puede realizar modificaciones en los requisitos, hay que seguir buscando componentes más adecuados (fase 1).
- Diseño del sistema con reutilización: Se diseña o reutiliza el marco de trabajo para el sistema. Se debe tener en cuenta los componentes localizados en la fase 2 para diseñar o determinar este marco.
- Desarrollo e integración: El software que no puede comprarse, se desarrolla. Se integran los componentes y subsistemas. La integración es parte del desarrollo en lugar de una actividad separada.

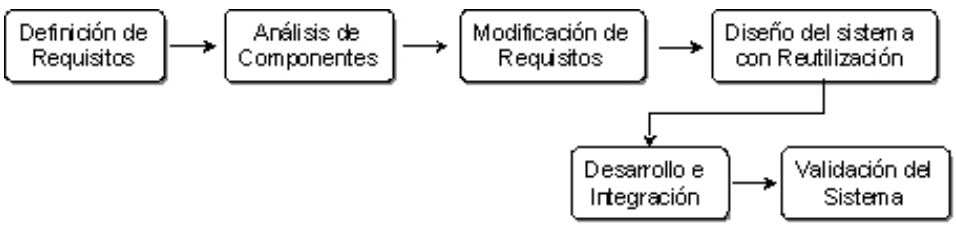


Figura 1.9. Desarrollo basado en componentes

3. RATIONAL UNIFIED PROCESS (RUP)

RUP es un proceso o marco de trabajo para el desarrollo de un proyecto de software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Presenta tres características esenciales:

- ***Dirigido por casos de uso***: Orientan el proyecto a la importancia para el usuario y lo que éste quiere.

- ***Centrado en la arquitectura***: Relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.

- ***Iterativo e incremental***: Divide el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada.

Como filosofía RUP maneja seis principios claves:

- ***Adaptación del proceso***. El proceso deberá adaptarse a las características propias de la organización. El tamaño del mismo, así como las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

- ***Balancear prioridades***. Los requisitos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

- ***Colaboración entre equipos***. El desarrollo de software no lo hace una única persona, sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.

- ***Demostrar valor iterativamente***. Los proyectos se entregan, aunque sea de un modo interno, en iteraciones. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como, también, los riesgos involucrados.

- ***Elevar el nivel de abstracción***. Este principio dominante motiva el uso de conceptos reutilizables, tales como patrón del software, lenguajes 4GL o esquemas (*frameworks*), por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
- ***Enfocarse en la calidad***. El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

Por otro lado, RUP describe cómo aplicar efectivamente enfoques comprobados comercialmente para el desarrollo de software. Estos enfoques son llamados "Mejores Prácticas" o “Best Practices”, en su denominación inglesa, pues son utilizados en la industria por organizaciones exitosas.

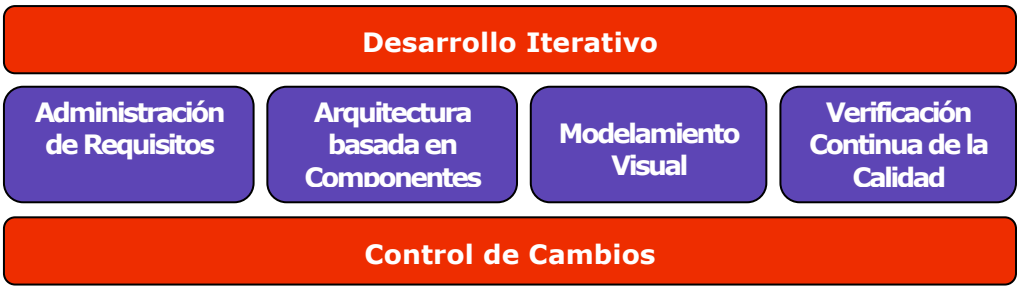


Figura 1.10. RUP – Mejores prácticas

- ***Desarrollo iterativo***

En función de la cada vez mayor complejidad solicitada para los sistemas de software, ya no es posible trabajar secuencialmente, es decir, definir primero el problema completo; luego, diseñar toda la solución, construir el software y, finalmente, testear el producto. Es necesario un enfoque iterativo que permita una comprensión creciente del problema a través de refinamientos sucesivos, llegando a una solución efectiva luego de múltiples iteraciones acotadas en complejidad.

RUP utiliza y soporta este enfoque iterativo e incremental que ayuda a atacar los riesgos mediante la producción de entregables ejecutables progresivos y frecuentes que permiten la opinión e involucramiento del usuario.

A través de las iteraciones que generan entregables ejecutables, se logra detectar, en forma temprana, los desajustes e inconsistencias entre los requisitos, el diseño, el desarrollo y la implementación del sistema, manteniendo al team de desarrollo focalizado en producir resultados.

- ***Administración de requisitos***

Los requisitos son las condiciones o capacidades que el sistema debe conformar. La administración de requisitos es un enfoque sistemático para hallar, documentar, organizar y monitorear los requisitos cambiantes de un sistema.

La administración de requisitos permite:

a) que las comunicaciones estén basadas en requisitos claramente definidos;
b) que los requisitos puedan ser priorizados, filtrados y monitoreados;
c) que sea posible realizar evaluaciones objetivas de funcionalidad y *performance*;
d) que las inconsistencias se detecten fácilmente.

RUP describe como:

a) Obtener, organizar y documentar la funcionalidad y restricciones requeridas;
b) Documentar y monitorear las alternativas y decisiones.

Las nociones de casos de uso y de escenarios utilizadas en RUP han demostrado ser una manera excelente de capturar los requisitos funcionales

y asegurarse que dirigen el diseño, la implementación y la prueba del sistema, logrando así que el sistema satisfaga las necesidades del usuario.

- ***Arquitectura basada en componentes***

El proceso de software debe focalizarse en el desarrollo temprano de una arquitectura robusta ejecutable, antes de comprometer recursos para el desarrollo en gran escala. RUP describe cómo diseñar una arquitectura flexible, que se acomode a los cambios, comprensible intuitivamente y promueve una más efectiva reutilización de software. Soporta el desarrollo de software basado en componentes: módulos no triviales que completan una función clara. RUP provee un enfoque sistemático para definir una arquitectura utilizando componentes nuevos y preexistentes.

- ***Modelamiento visual***

RUP muestra cómo representar el software visualmente para capturar la estructura y comportamiento de arquitecturas y componentes. Las abstracciones visuales ayudan a comunicar diferentes aspectos del software; comprender los requisitos, ver cómo los elementos del sistema se relacionan entre sí, mantener la consistencia entre diseño e implementación y promover una comunicación precisa. El estándar UML (Lenguaje de Modelado Unificado), creado por *Rational Software*, es el cimiento para un modelamiento visual exitosa.

- ***Verificación continua de la calidad***

Es necesario evaluar la calidad de un sistema respecto de sus requisitos de funcionalidad, confiabilidad y performance. La actividad fundamental es el testeo (*testing*), que permite encontrar las fallas antes de la puesta en producción. RUP asiste en el planeamiento, diseño, implementación, ejecución y evaluación de todos estos tipos de testeo (*testing*).

El aseguramiento de la calidad se construye dentro del proceso, en todas las actividades, involucrando a todos los participantes, utilizando medidas y criterios objetivos, permitiendo así detectar e identificar los defectos en forma temprana.

- ***Control de cambios***

La capacidad de administrar los cambios es esencial en ambientes en los cuales el cambio es inevitable. RUP describe como controlar, rastrear y monitorear los cambios para permitir un desarrollo iterativo exitoso. Es también una guía para establecer espacios de trabajo seguros para cada desarrollador, suministrando el aislamiento de los cambios hechos en otros espacios de trabajo y controlando los cambios de todos los elementos de software (modelos, código, documentos, etc.). Describe cómo automatizar la integración y administrar la conformación de entregables.

3.1 Estructura de RUP

RUP es un proceso que puede describirse en dos dimensiones, tal como se muestra en la figura 1.11, a lo largo de dos ejes:

- El eje horizontal representa tiempo y muestra el aspecto dinámico del proceso expresado en términos de ciclos, fases, iteraciones, y metas.
- El eje vertical representa el aspecto estático del proceso; está descrito en términos de actividades, artefactos, trabajadores o roles y flujos de trabajo.

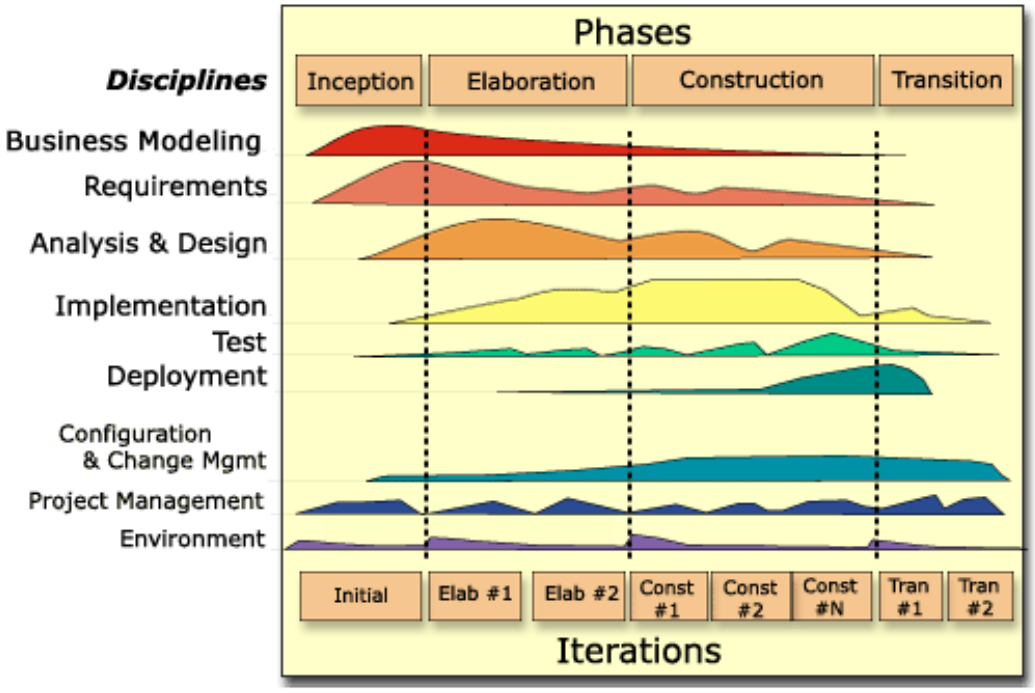


Figura 1.11. Estructura de RUP

3.2 Fases

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se desarrolla en mayor o menor proporción los distintos flujos de trabajo:

- ***Inicio:*** En esta primera fase se define el alcance y objetivos del proyecto.
- ***Elaboración:*** Se desarrolla el plan del proyecto, la especificación de características y la arquitectura base del sistema.
- ***Construcción:*** Esta fase se concentra en la elaboración, de un producto totalmente operativo y eficiente y el manual de usuario.
- ***Transición:*** Es la fase en la cual se instala el producto en el cliente y se entrena a los usuarios.

3.3 Flujos de trabajo

Los flujos de trabajo son también conocidos como disciplinas. Consisten en una **secuencia de actividades** que producen un resultado observable (artefacto) a cargo de algún miembro del proyecto (rol).

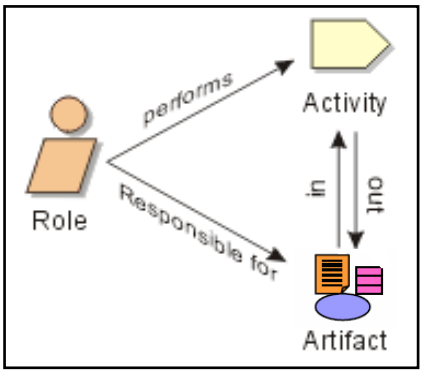


Figura 1.12. Elementos de RUP que describen el flujo de trabajo

Existen dos grupos de flujos de trabajo: de proceso y de apoyo, los cuales que se describirán a continuación.

3.3.1. Flujos de trabajo del proceso

Orientados al desarrollo del software. Comprende:

- ***Modelado del negocio:*** Describe la estructura y la dinámica de la organización donde se va a implantar el sistema que construyamos.
- ***Captura de requisitos:*** Establece exactamente lo que tiene que hacer el sistema, para ello, se extrae los requisitos utilizando diferentes métodos.
- ***Análisis y diseño:*** Traduce los requisitos a una especificación que describe cómo implementar el sistema creando, para ello, diferentes vistas arquitectónicas.
- ***Implementación:*** Tiene en cuenta el desarrollo de software, las pruebas unitarias y la integración.
- ***Pruebas:*** Describe la ejecución de pruebas y las métricas para rastreo de defectos.
- ***Despliegue o implantación*****:** Incluye actividades relacionadas con la entrega de la aplicación.

3.3.2. Flujos de trabajo de apoyo o de soporte

Orientados a la gestión del proyecto. Éstos son:

- ***Configuración y control de cambios:*** Mantiene la integridad de todos los artefactos que se crean en el proyecto. También mantiene información del proceso evolutivo que se ha seguido.
- ***Gestión del proyecto:*** Es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.
- ***Entorno:*** Cubre la infraestructura necesaria para desarrollar un sistema.

3.4 Roles en RUP

Un rol define el comportamiento y responsabilidades de un individuo o de un grupo de individuos trabajando juntos como un equipo. Un miembro del equipo de proyecto cumple, normalmente, muchos roles. Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos. Existen muchos roles específicos dentro de los roles genéricos RUP, tales como:

Analistas:

- Analista de procesos de negocio
- Diseñador del negocio
- Analista de sistema
- Especificador de requisitos

Desarrolladores:

- Arquitecto de software
- Diseñador
- Diseñador de interfaz de usuario
- Diseñador de cápsulas
- Diseñador de base de datos
- Implementador
- Integrador

Gestores:

- Jefe de proyecto
- Jefe de control de cambios
- Jefe de configuración
- Jefe de pruebas
- Jefe de despliegue
- Ingeniero de procesos
- Revisor de gestión del proyecto
- Gestor de pruebas

Apoyo:

- Documentador técnico
- Administrador de sistema
- Especialista en herramientas
- Desarrollador de cursos
- Artista gráfico

Especialista en pruebas:

- Especialista en Pruebas
- Analista de pruebas
- Diseñador de pruebas

Otros roles:

- Stakeholders
- Revisor
- Coordinador de revisiones
- Revisor técnico

A continuación, se describirá los roles específicos dentro del rol Analista:

3.4.1. Analista de procesos de negocio

Conduce y coordina el caso de uso del negocio que modela delimitando la organización que es modelada. El analista del proceso de negocio es **responsable de la arquitectura del negocio**, pues **establece qué actores del negocio y casos de uso del negocio existen y como trabajan entre ellos**.

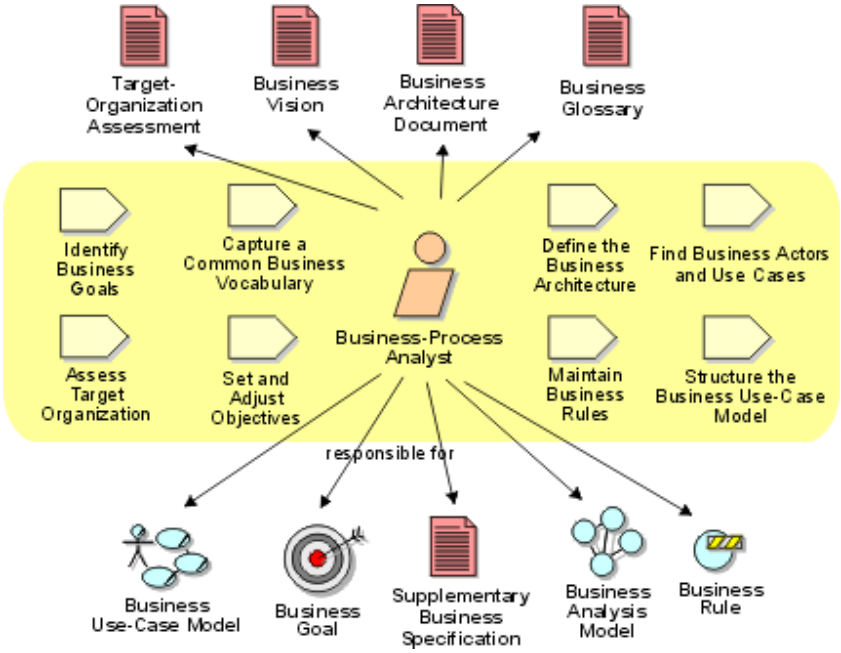


Figura 1.13. Analista de procesos de negocio

3.4.2. Diseñador del negocio

Detalla la especificación de una parte de la organización describiendo el flujo de trabajo de uno o varios casos de uso del negocio. Este rol **especifica los trabajadores del negocio y las entidades de negocio necesarios para realizar un caso de uso del negocio** y distribuye el comportamiento del caso de uso del negocio a éstos. Asimismo define las responsabilidades, las operaciones, las cualidades, y las relaciones de uno o varios trabajadores del negocio y entidades de negocio.

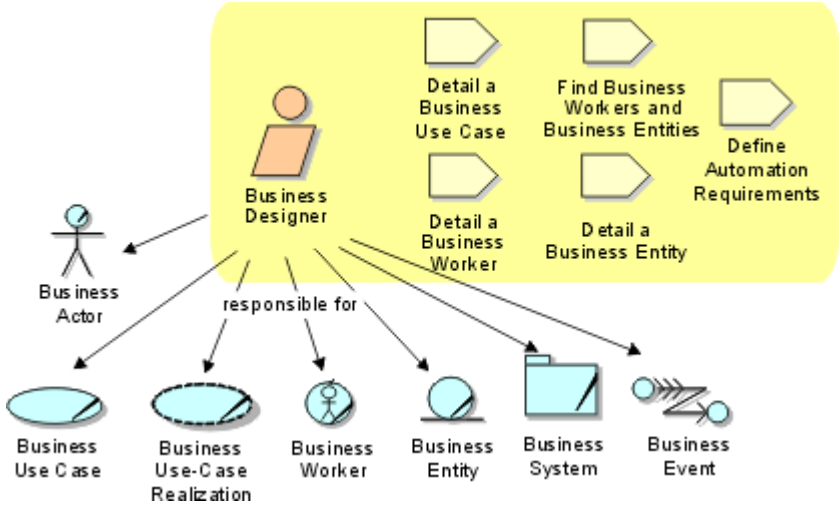


Figura 1.14. Diseñador del negocio

3.4.3. Analista de sistema

Conduce y coordina los requerimientos y los casos de uso modelando y delimitando la funcionalidad del sistema y delimitando el sistema; por ejemplo, **estableciendo qué actores y casos de uso existen y como se relacionan**.

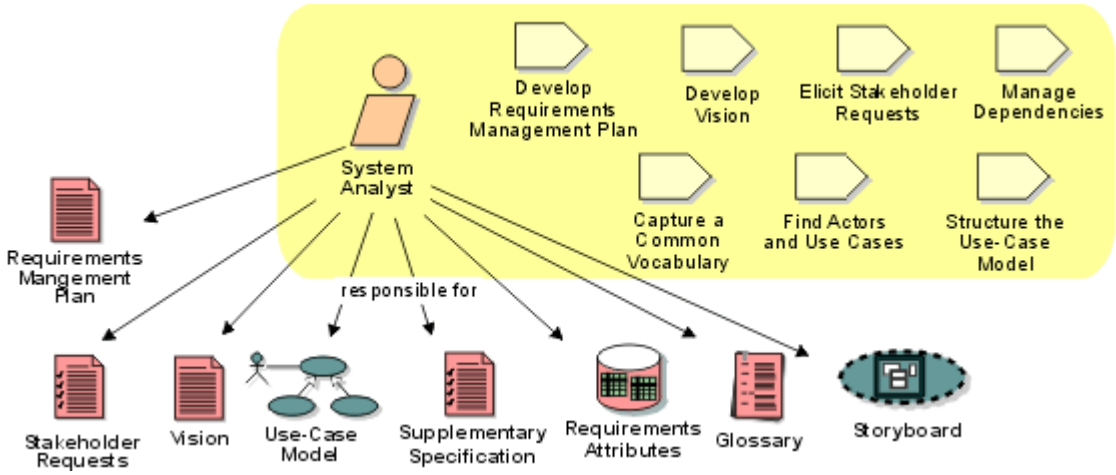


Figura 1.15. Analista de sistema

3.4.4. Especificador de requisitos

Detalla la especificación de una parte de la funcionalidad del sistema (caso de uso) **describiendo el aspecto de los requisitos de uno o varios casos de uso y otros requisitos de soporte del software en el ERS** (Especificación de Requisitos del Software). También, puede ser responsable de un paquete de casos de uso y mantiene la integridad de ese paquete. Se recomienda que el especificador de los requisitos responsable de un paquete de casos de uso sea también responsable de sus casos de uso y actores contenidos.

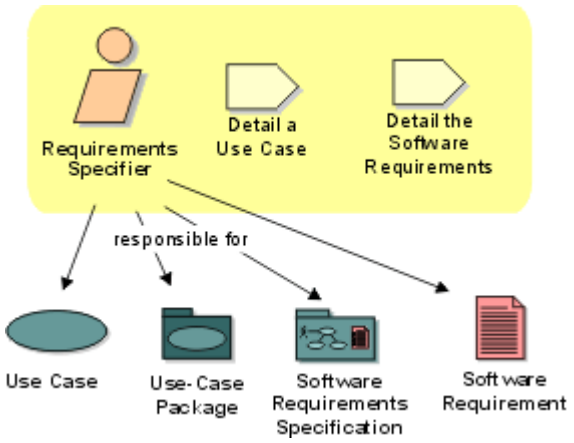


Figura 1.16. Especificador de requisitos

ACTIVIDAD PROPUESTA

1. Prepare un cuadro comparativo de las ventajas y desventajas de los modelos de proceso de software.

2. Sugiera el modelo de proceso del software que podría utilizarse para gestionar el desarrollo de los siguientes sistemas dando algunas razones basadas en el tipo de sistema a desarrollar:

    - Un sistema de control antibloqueo de frenos de un automóvil.
    - Un sistema de realidad virtual para ayudar al mantenimiento del software.
    - Un sistema de contabilidad universitaria que reemplace el existente.
    - Un sistema interactivo que permita a los pasajeros encontrar los horarios de los trenes a partir de las terminales instaladas en las estaciones.

Resumen

- La ingeniería de software tiene muchas definiciones. Según Roger Pressman, es una tecnología multicapa que reúne cuatro elementos: un enfoque de calidad, proceso, métodos y herramientas.

- Un proceso de desarrollo de software tiene tres fases genéricas: definición (se centra sobre el qué), desarrollo (se centra en el cómo) y mantenimiento (se centra en el cambio). Además, existe una categoría de actividades, que complementan a las fases genéricas: Actividades de protección.

- Existen muchos modelos de procesos de software, conocidos también como paradigmas de Ingeniería de software, que son estrategias de desarrollo que acompañan al proceso, métodos, herramientas y fases genéricas y son seleccionadas de acuerdo a la naturaleza del proyecto en general.

- Cada modelo de desarrollo de software incluye los requisitos del sistema como entrada y el producto utilizado por los usuarios como salida.

- RUP es un marco de trabajo para el desarrollo de software, el cual se describe en dos dimensiones, el horizontal es de aspecto dinámico (ciclos, fases, iteraciones y metas) y el vertical cubre el aspecto estático (actividades, artefactos, trabajadores o roles y flujos de trabajo).

- RUP reúne las mejores prácticas de organizaciones exitosas: desarrollo iterativo, administración de requisitos, arquitectura basada en componentes, modelamiento visual, verificación continua de la calidad y control de cambios.

- Si desea saber más acerca de estos temas, puede consultar los siguientes libros.

  - "Ingeniería del Software. Un enfoque práctico" de Roger S. Pressman

    En los primeros capítulos, el autor desarrolla temas de Ingeniería de Software. En capítulos posteriores, permite al lector seguir a un equipo de proyecto (ficticio) conforme planifica y diseña un sistema basado en computadora.

  - "EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE" de Ivar Jacobson, Grady Booch y James Rumbaugh.

    Encontrará una amplia revisión sobre el proceso unificado de desarrollo de software, poniendo especial énfasis en el modelo práctico con UML.

  - "Rational Unified Process. Best Practices for Software Development Teams" Rational Software White Paper.

    Un Material desarrollado por Rational Software Corporation sobre RUP.

UNIDAD DE APRENDIZAJE

2

DISCIPLINA DEL MODELADO DEL NEGOCIO

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno resuelve un caso propuesto por el profesor acerca del modelado de negocio, el cual está conformado por el modelo de casos de uso del negocio, en el que identifica los objetivos, casos de uso y actores del negocio, y realiza el diagrama general de casos de uso del negocio, mientras que para el modelo de análisis del negocio, a los trabajadores y entidades, diagrama las clases y actividades del negocio.

TEMARIO

1. Lenguaje Unificado de Modelado (UML)
2. Modelado del negocio
3. Modelo de casos de uso del negocio
4. Modelo de análisis del negocio
5. Casos de estudio

ACTIVIDADES PROPUESTAS

1. Los alumnos indican los elementos más importantes de los diagramas propuestos.

2. Los alumnos desarrollan el modelado de negocio de un proceso de negocio, el cual contiene:
   - El modelo de casos de uso de negocio
   - El modelo de análisis del negocio

1. LENGUAJE UNIFICADO DE MODELADO (UML)

1.1. El modelamiento visual

El modelamiento visual es el modelado de una aplicación usando notaciones gráficas. Pero ¿qué tan importante es construir el modelo de una aplicación? En este sentido, comúnmente, se hace la comparación hacia la arquitectura tradicional en la construcción de casas. Aún cuando la construcción que se planee hacer sea una casa sencilla, el resultado será más satisfactorio si se cuenta con todo un respaldo en un correcto diseño.

Booch compara la construcción de software con la construcción de una casa para una mascota, de una casa para una familia y de un gran edificio. En el primer caso no será tan evidente la falta de un buen diseño, o al menos nuestra mascota no se quejará demasiado. En el segundo caso, es humanamente posible hacer la construcción de una casa sin los planos adecuados, pero la casa resultante seguramente tendrá varias carencias, o en el peor de los casos, posiblemente no resistirá ciertas condiciones extremas, como un temblor. En el caso del edificio, definitivamente, sería un grave error comenzar la construcción sin los estudios y planos adecuados.

En el caso del software, es curioso como muchos proyectos del tamaño de un rascacielos, son construidos como si se tratara de la casa de un perro. El modelado visual del software ayuda a capturar las partes esenciales de un sistema:

- Se utiliza para capturar los procesos de negocios desde la perspectiva del usuario.
- El modelado visual se utiliza para analizar y diseñar una aplicación, distinguiendo entre los dominios del negocio y los dominios de la computadora.
- Ayuda a reducir la complejidad.
- El modelado visual se realiza de manera independiente al lenguaje de implementación.
- Promueve la reutilización de componentes.

Un modelo se considera como útil si presenta las siguientes características:

- Preciso: Describen correctamente el sistema a ser construido.
- Consistente: Los distintos puntos de vista no expresan cosas que entren en conflicto entre ellas.
- Fácil de comunicárselo a otros. Mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.
- Fácil de cambiar
- Legible: Todas las representaciones se realizan de la manera más simple como sea posible.

1.2. El Lenguaje de Modelamiento Unificado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para visualizar, especificar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre tales sistemas. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en acercamiento estándar. UML incluye conceptos semánticos, notación y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos.

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo para entender y controlar las dependencias entre paquetes y para gestionar las versiones de las unidades de modelo en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implantación y para elementos de tiempo de ejecución.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. UML no es un lenguaje altamente formal pensando para probar teoremas. Hay varios lenguajes de ese tipo, pero no son fáciles de entender ni de usar para la mayoría de los propósitos. UML es un lenguaje de modelado de propósito general. Para dominios especializados, tales como la composición de IUG, diseño de circuitos VLSI, o inteligencia artificial basada en reglas, podría ser mas apropiada una herramienta especializada con un lenguaje especial. UML es un lenguaje de modelado discreto. No se creó para modelar sistemas continuos como los basados en ingeniería y física. UML quiere ser un lenguaje de modelado universal, de propósito general, para sistemas discretos, tales como los compuestos por software, *firmware* o lógica digital.

1.2.1. Breve Historia de UML

Los lenguajes de modelado de objetos aparecieron en la década de 1980, y llegaron a ser unos 50 a mediados de la década de 1990. Unos pocos métodos empezaron a ganar importancia, ente ellos el método de Grady Booch (BM), el método OOSE de Ivar Jacobson, y el método OMT de James Rumbaugh.

El esfuerzo para la definición de UML comenzó en Octubre de 1994 cuando Rumbaugh se unió a Booch en Rational Software Corporation (empresa donde trabajaba Booch). Unificaron sus métodos de modelado y elaboraron el borrador de la versión 0.8 de UML. En 1995, Jacobson también se incorporó a Rational, incorporando así OOSE a UML. En 1996, se publicó la versión 0.9 de UM. Las tres personalidades que dieron el origen a UML son conocidas como “los tres amigos”.

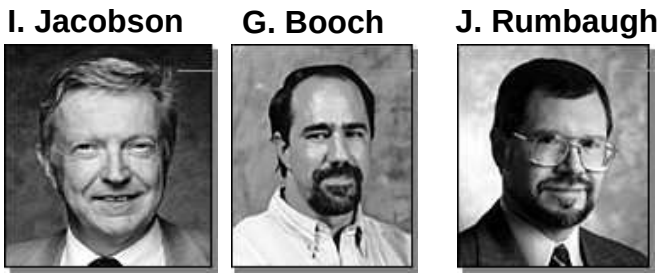


Figura 2.1. Los tres amigos

Las organizaciones que contribuyeron a la definición de 1.0 de UML fueron Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments y Unisys. UML 1.0 se ofreció para su estandarización al OMG en enero de 1997.

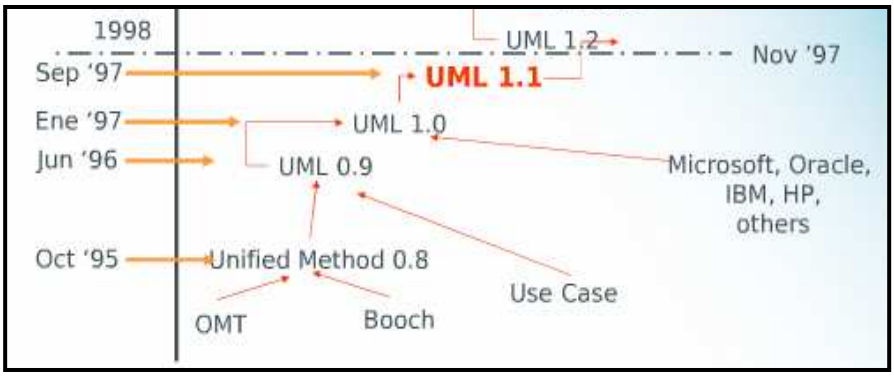


Figura 2.2. Historia de UML

Luego de varios años y varias modificaciones, OMG adoptó la versión oficial de UML 2.0 en el año 2005. Los documentos de la especificación de UML se encuentran en la página web de OMG. La versión actual es 2.2.

Unified Modeling Language™ (UML®)

OMG Formally Released Versions Of UML

Version	Release date
2.2	February 2009
2.1.2	November 2007
2.1.1	August 2007
Please note that version 2.1 was never released as a formal specification	
2.0	July 2005
1.5	March 2003
1.4	September 2001
1.3	March 2000

Figura 2.3. Lanzamientos del UML

¿Qué es la OMG?

La OMG (Object Management Group) es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software, como, por ejemplo, IBM, Apple Computer, Sun Microsystems Inc y Hewlett-Packard. Esta asociación se encarga de la definición y mantenimiento de estándares para aplicaciones de la industria de la computación. Otro de los estándares definidos por la OMG, además del UML, es CORBA, el cual permite interoperabilidad multiplataforma a nivel de objetos de negocio.

1.2.2. Objetivos de UML

Los objetivos de UML son muchos, pero se pueden sintetizar en las siguientes:

- Visualizar: UML permite expresar de una forma gráfica, un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados, se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.2.3. Especificaciones fundamentales de UML 2.0

En las versiones previas del UML, se hacía un fuerte hincapié en que UML no era un lenguaje de programación. Un modelo creado mediante UML no podía ejecutarse. En el UML 2.0, esta asunción cambió de manera drástica y se modificó el lenguaje, de manera tal que permitiera capturar mucho más comportamiento. De esta forma, **se permitió la creación de herramientas que soporten la automatización y generación de código ejecutable**, a partir de modelos UML.

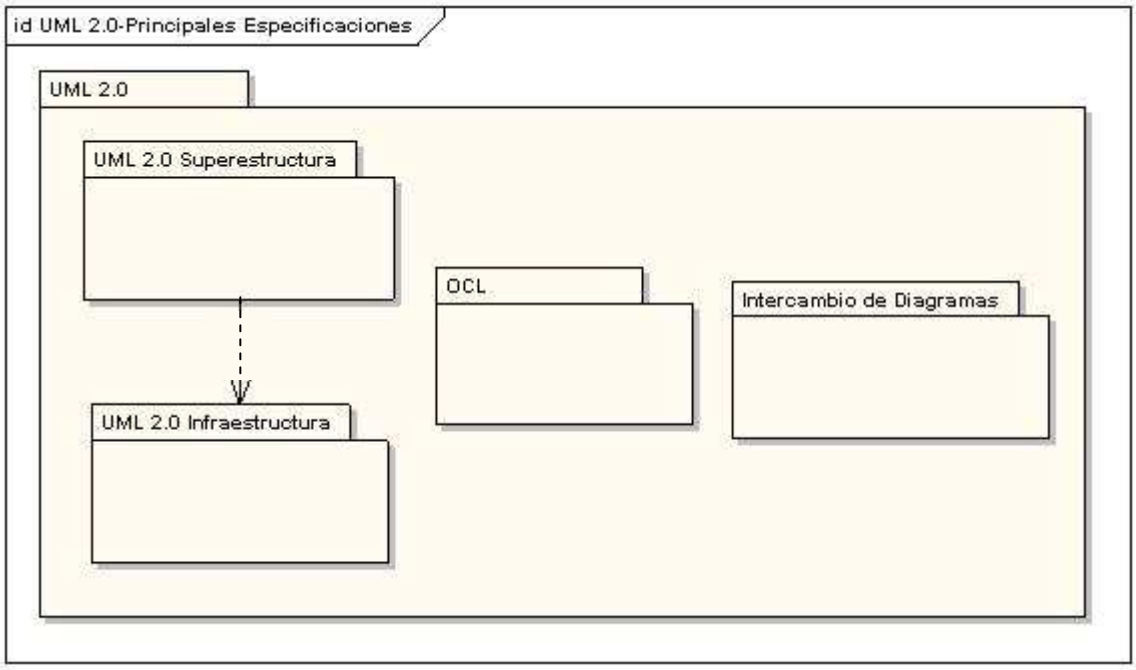


Figura 2.4. Especificaciones principales de UML 2.0

Para lograr los objetivos de UML enunciados en el punto anterior, varios aspectos del lenguaje fueron reestructurados y/o modificados. La especificación se separó en cuatro especificaciones bien definidas, tal como se muestra en la Figura 2.2. Es interesante destacar que el UML 2.0 puede definirse a sí mismo. Es decir, su estructura y organización es modelable utilizando el propio UML 2.0. De esta manera, se da un ejemplo de utilización del UML en un dominio distinto al del desarrollo de software. En este caso, cada paquete del diagrama representa cada una de las cuatro especificaciones que componen el lenguaje.

Veamos, a continuación, cada una de las principales especificaciones que componen UML 2.0.

1.2.3.1. Superestructura

La superestructura del UML es la definición formal de los elementos del UML Es aquí dónde se definen los diagramas y los elementos que los componen. Esta definición sola contiene más de 640 páginas. La superestructura es utilizada por los desarrolladores de aplicación. Es aquella sobre la que hablan los libros y la que la mayoría conoce de versiones anteriores del UML.

Se encuentra dividida en niveles. Estos niveles se conocen como:

- Básico (L1): Contiene los elementos básicos del UML 2.0, entre ellos, diagramas de clases, diagramas de actividades, diagramas de interacciones, y diagramas de casos de uso

- Intermedio (L2): Contiene los siguientes diagramas: diagramas de estado, perfiles, diagramas de componentes y diagramas de despliegue.
- Completo (L3): Representa la especificación del UML 2.0 completa, como, por ejemplo, las acciones, características avanzadas y PowerTypes, entre otros.

Es importante destacar que basta con que una herramienta implemente el nivel de conformidad básico (L1), para que se considere UML 2.0 compatible. Por eso, es normal ver una disparidad de características (features) bastante amplia entre dos herramientas distintas, aunque éstas sean UML 2.0 compatibles.

1.2.3.2. Infraestructura

En la Infraestructura del UML se definen los conceptos centrales y de más bajo nivel. La Infraestructura es un meta-modelo (un modelo de modelos) y, mediante la misma, se modela el resto del UML. Generalmente, la infraestructura no es utilizada por usuarios finales del UML, pero provee la piedra fundamental sobre la cual la Superestructura es definida. La Infraestructura brinda también varios mecanismos de extensión, que hacen del UML un lenguaje configurable. Para los usuarios normales del UML, basta con saber si la infraestructura existe y cuáles son sus objetivos.

1.2.3.3. OCL

OCL son siglas en inglés que significan: Object Constraint Language y que en castellano se traducen como: Lenguaje de Restricciones de Objetos. El OCL define un lenguaje simple, para escribir restricciones y expresiones sobre elementos de un modelo. El OCL suele ser útil cuando se está especificando un dominio particular mediante el UML y es necesario restringir los valores permitidos para los objetos del dominio. El OCL brinda la posibilidad de definir invariantes, precondiciones, poscondiciones y restricciones en los elementos de un diagrama. Fue incorporado a UML en la versión 1.1. Originalmente, fue especificado por IBM y es un ejemplo más de las muchas herramientas agregadas a UML.

1.2.3.4. Intercambio de diagramas

La especificación para el intercambio de diagramas fue escrita para facilitar una manera de compartir modelos, realizados mediante UML, entre diferentes herramientas de modelado.

En versiones anteriores del UML, se utilizaba un Schema XML para capturar los elementos utilizados en el diagrama, pero este Schema no decía nada acerca de la manera en que el modelo debía graficarse. Para solucionar este problema, la nueva especificación para el intercambio de diagramas fue

desarrollada utilizando un nuevo Schema XML, que permite construir una representación SVG (Scalable Vector Graphics).

Esta especificación es denominada con las siglas XMI, que en inglés significa XML Metadata Interchange; y en castellano se traduce como XML de Intercambio de Metadata (datos que representan datos). Típicamente esta especificación es solamente utilizada por quienes desarrollan herramientas de modelado UML.

1.2.4. Diagramas UML 2.0

En UML 2.0, hay 13 tipos diferentes de diagramas, como se muestra en la figura 2.3. En la versión 2.2, se agregó el Diagrama de perfiles (figura 2.4).

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de perfiles (A partir de UML 2.2)
- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (A partir de UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los diagramas de comportamiento enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de máquina de estados

Los diagramas de interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de comunicación
- Diagrama de tiempos (A partir de UML 2.0)
- Diagrama de descripción de la interacción (A partir de UML 2.0)

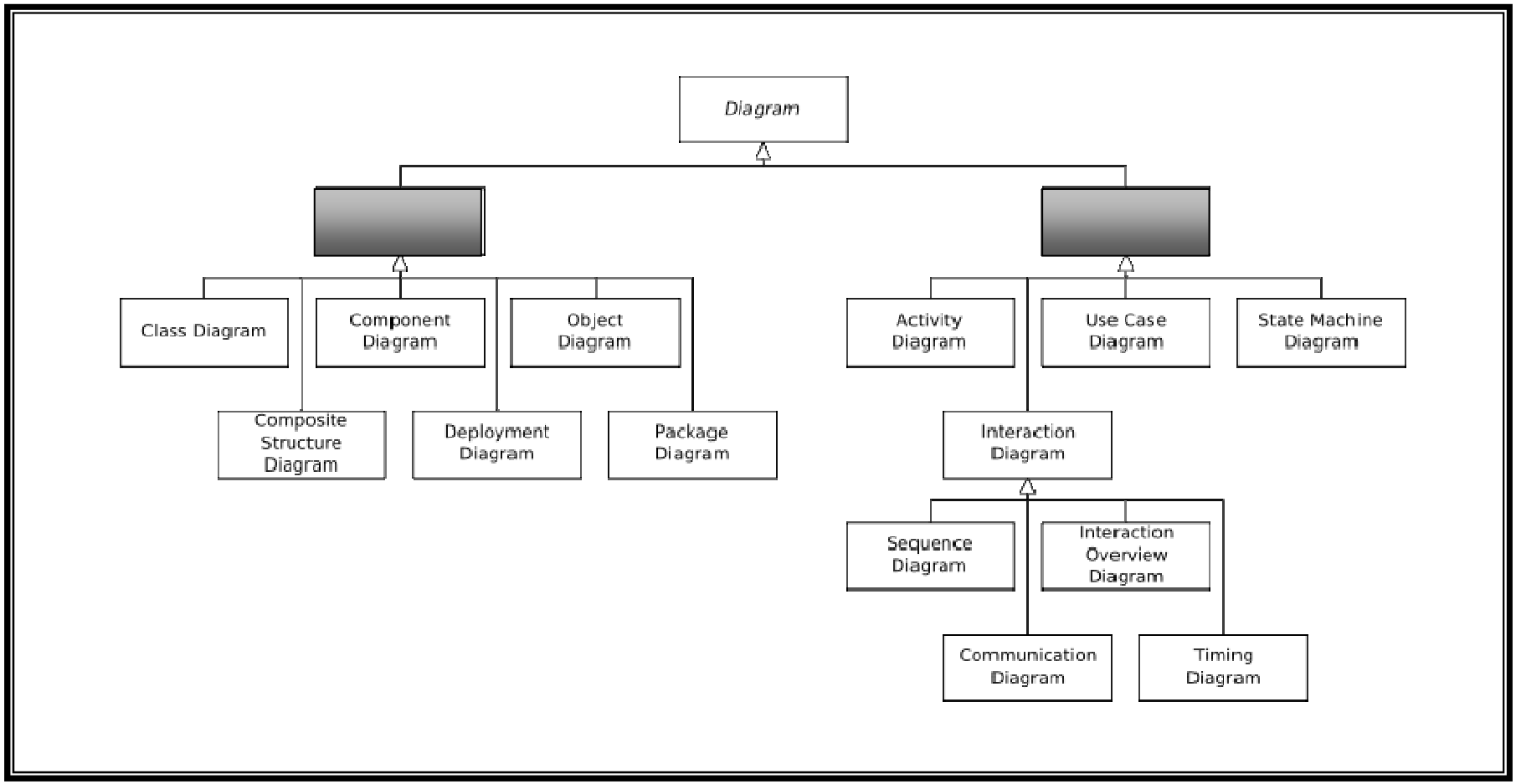


Figura 2.3. Jerarquía de los diagramas UML 2.0

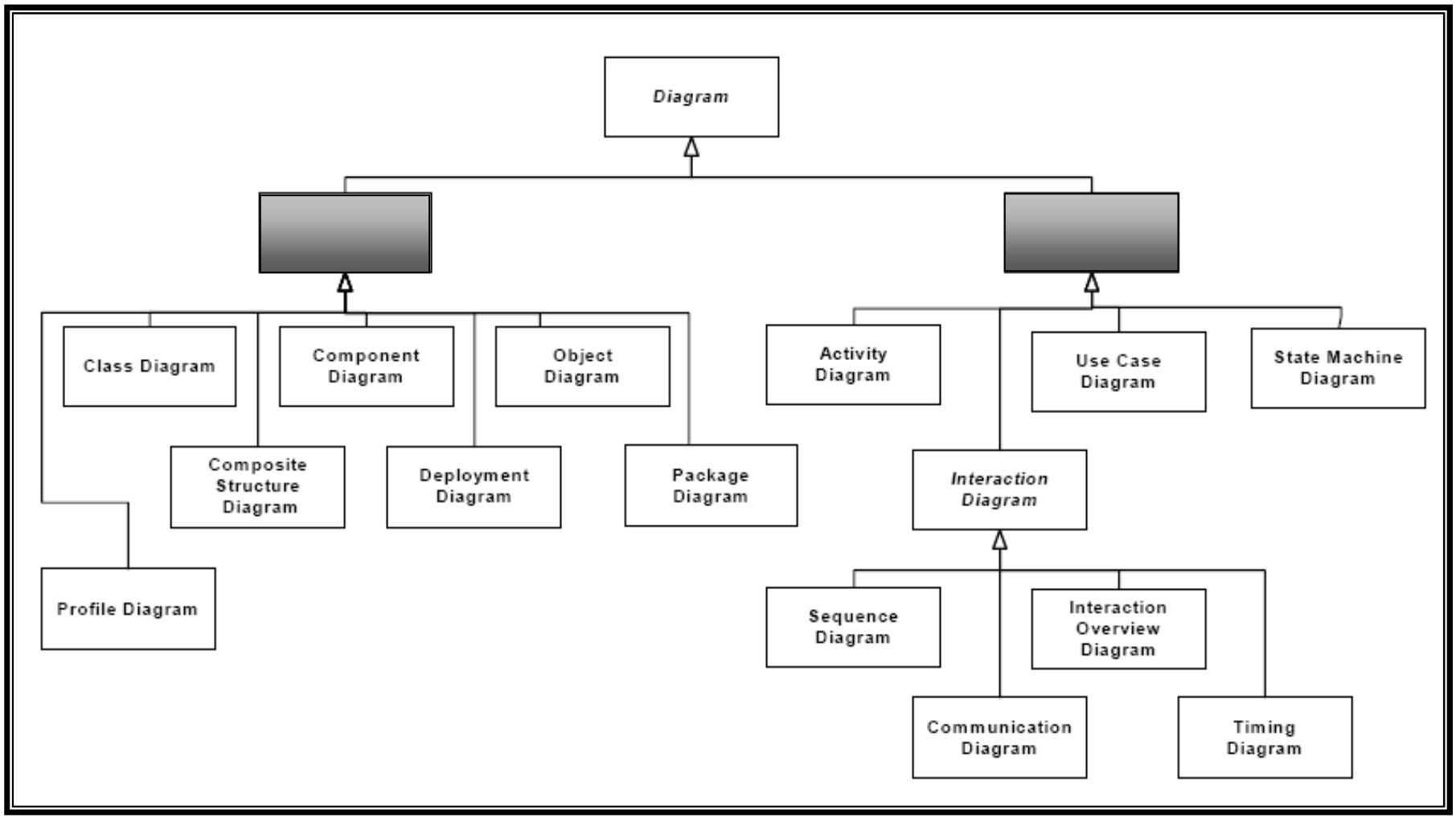


Figura 2.4. Jerarquía de los diagramas UML 2.2

1.2.5. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

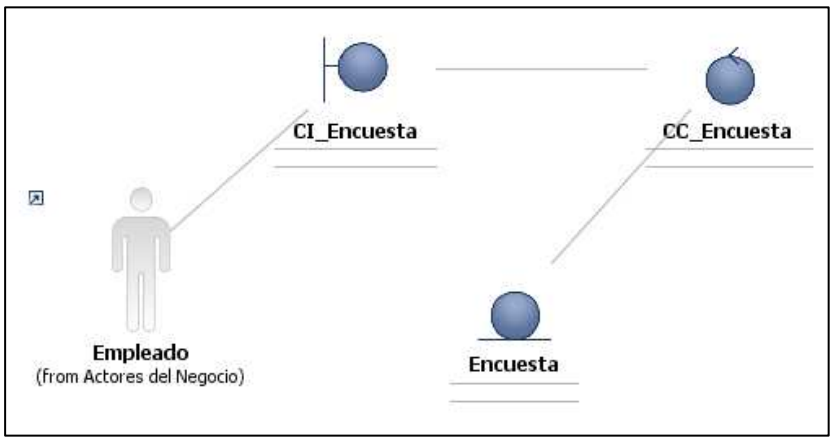


Figura 2.5. Diagrama de clases de análisis

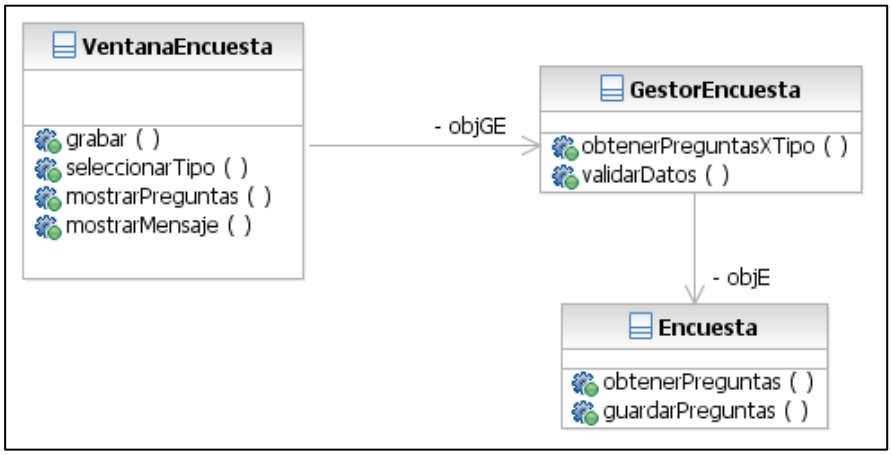


Figura 2.6. Diagrama de clases de diseño

1.2.6. Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los componentes, sólo

aparecen tipos de componentes, ya que las instancias específicas de cada tipo se encuentran en el diagrama de despliegue.

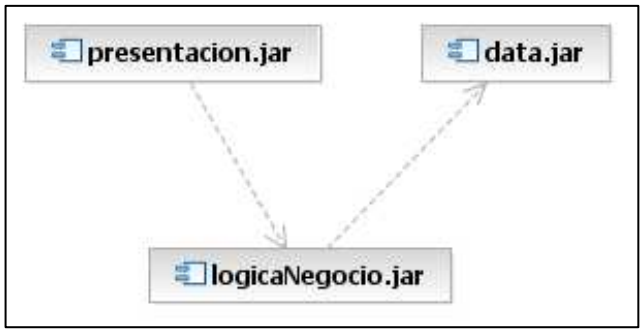


Figura 2.7. Diagrama de componentes

1.2.7. Diagrama de objetos

Se puede considerar un caso especial de un diagrama de clases en el que se muestran instancias específicas de clases (objetos) en un momento particular del sistema.

Los diagramas de objetos utilizan un subconjunto de los elementos de un diagrama de clase. Los diagramas de objetos no muestran la multiplicidad ni los roles, aunque su notación es similar a los diagramas de clase.

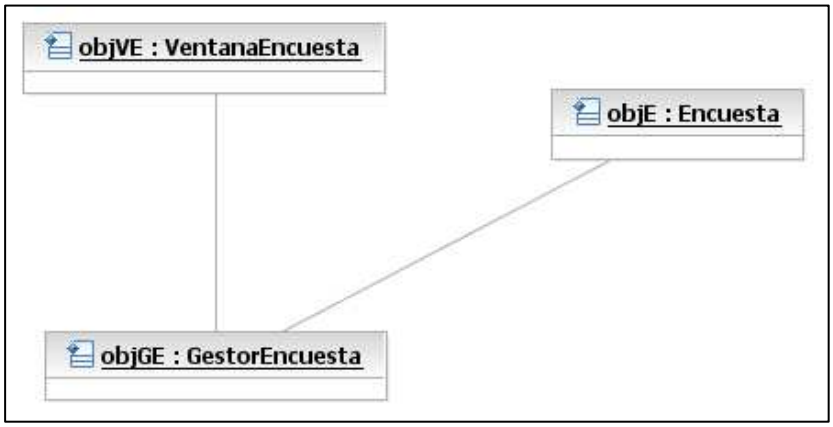


Figura 2.8. Diagrama de Objetos

1.2.8. Diagrama de estructura compuesta

Este tipo de diagrama fue específicamente **diseñado para la representación de patrones de diseño**, y es una de las modificaciones de mayor impacto dentro de UML 2.0. Esta modificación al UML hace que ahora todos los Clasificadores puedan tener una estructura compuesta. Mediante una composición de estructuras, el comportamiento de las instancias de otros Clasificadores contenidos (estructura interna) en un Clasificador determinado, puede especificarse como Colaboraciones. Los conceptos principales para describir la estructura interna son: Partes, Puertos y Conectores.

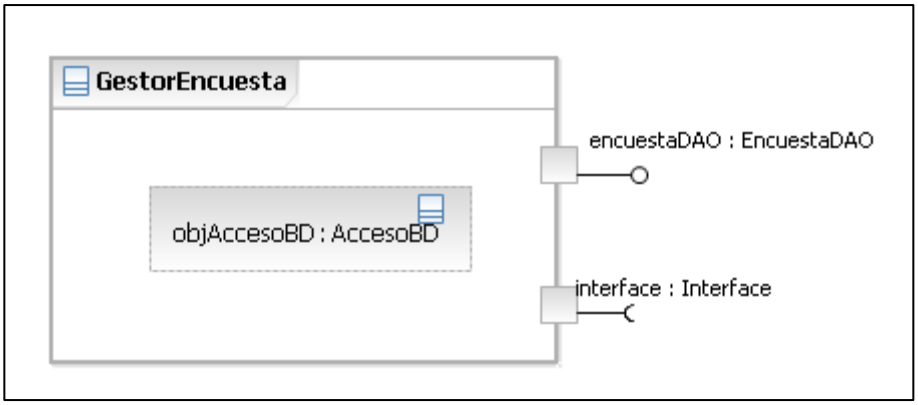


Figura 2.9. Diagrama de estructura compuesta

1.2.9. Diagrama de despliegue

Describen la configuración del entorno de máquinas y redes sobre el que se distribuyen componentes y procesos del sistema.

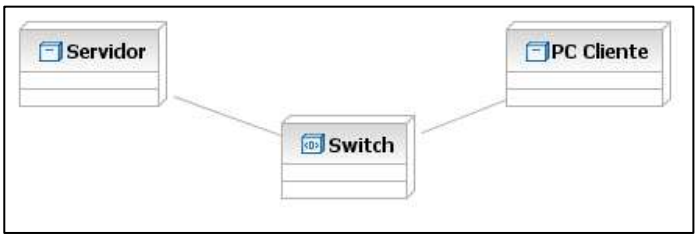


Figura 2.10. Diagrama de despliegue

1.2.10. Diagrama de paquetes

Este tipo de diagrama se usa para dividir el modelo en contenedores lógicos (paquetes) y describen las interacciones entre ellos a un nivel más alto. Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas/capas agrupando elementos de modelado. Cada paquete se corresponde a un submodelo (subsistema) del modelo (sistema), se pueden anidar paquetes. Por último, puede crearse relaciones de dependencia. Esto se da cuando un componente de un paquete necesita un componente de otro paquete.

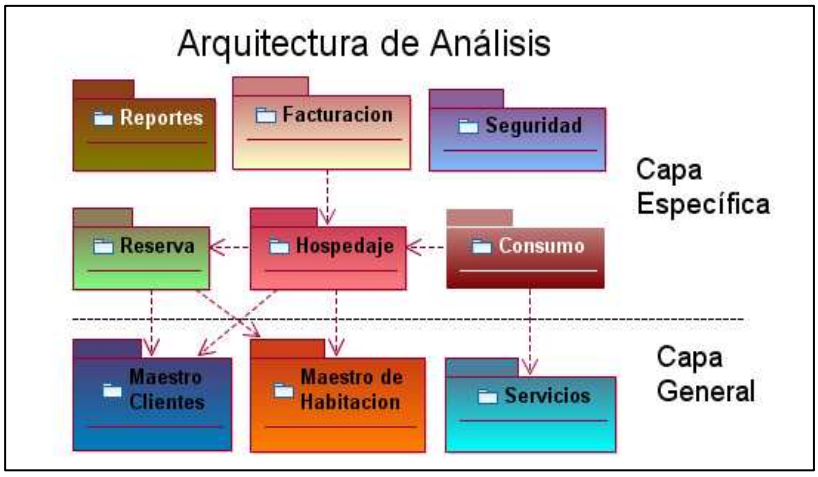


Figura 2.11. Diagrama de paquetes

1.2.11. Diagrama de actividades

Muestra un flujo ordenado de actividades. Los diagramas de actividades tienen un amplio número de usos: desde definir un flujo de programa básico hasta capturar los puntos de decisión y acciones dentro de cualquier proceso generalizado.

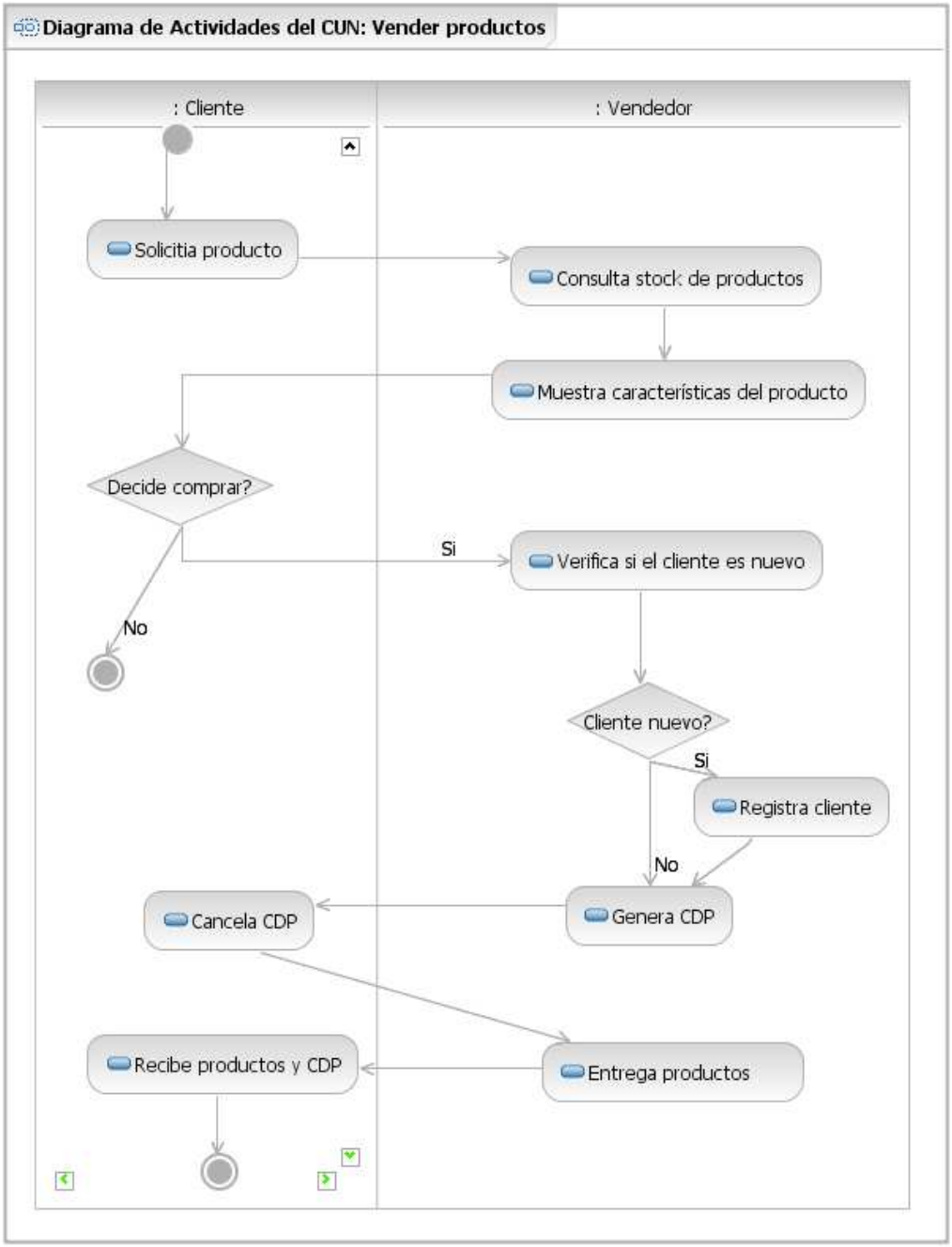


Figura 2.12. Diagrama de actividades

1.2.12. Diagrama de casos de uso

El diagrama de casos de uso permite realizar la especificación del alcance funcional del producto software que se construye y de los actores, entes que interactúan con el producto software, que requieren los diferentes casos de usos.

Los diagramas de casos de uso son usados para representar los procesos de negocio de la organización objetivo y las funcionalidades que representan la arquitectura del sistema por cada proceso de negocio.

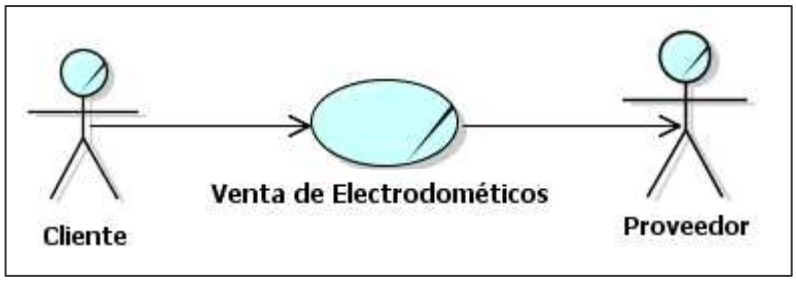


Figura 2.13. Diagrama de casos de uso del negocio

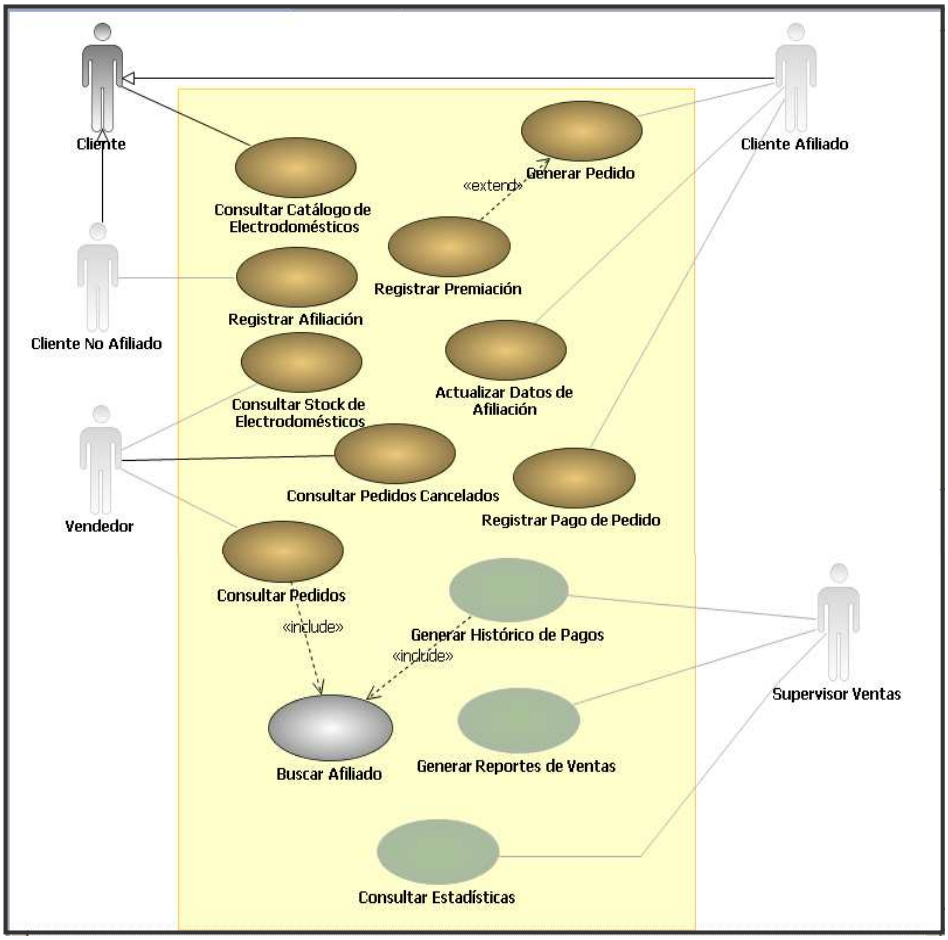


Figura 2.14. Diagrama de casos de uso

1.2.13. Diagrama de máquina de estados

Típicamente, este diagrama se utiliza para representar todos los posibles estados que los objetos de una clase puedan tener. Los diagramas de estado **no se hacen para todas las clases, es sólo para aquellas que tengan un número de estados bien definidos** y en donde el comportamiento de la clase es afectado y cambiado por los distintos estados.

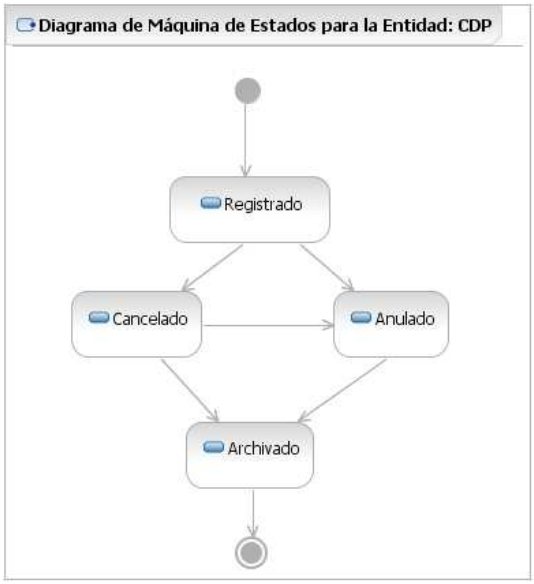


Figura 2.15. Diagrama de máquina de estados

1.2.14. Diagrama de secuencia

Muestra una secuencia ordenada de mensajes pasadas entre los objetos usando una línea de tiempo vertical.

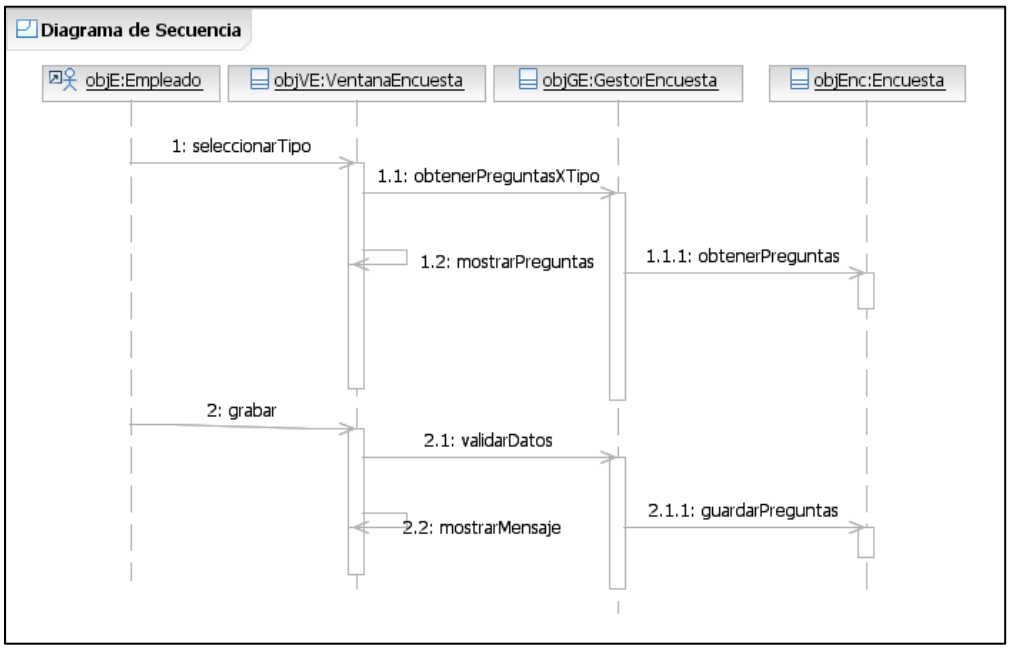


Figura 2.16. Diagrama de secuencia

1.2.15. Diagrama de comunicación

Antes, era conocida como diagrama de colaboración. Muestra la red y la secuencia de mensajes de comunicaciones entre objetos en tiempo de ejecución durante una instancia de colaboración.

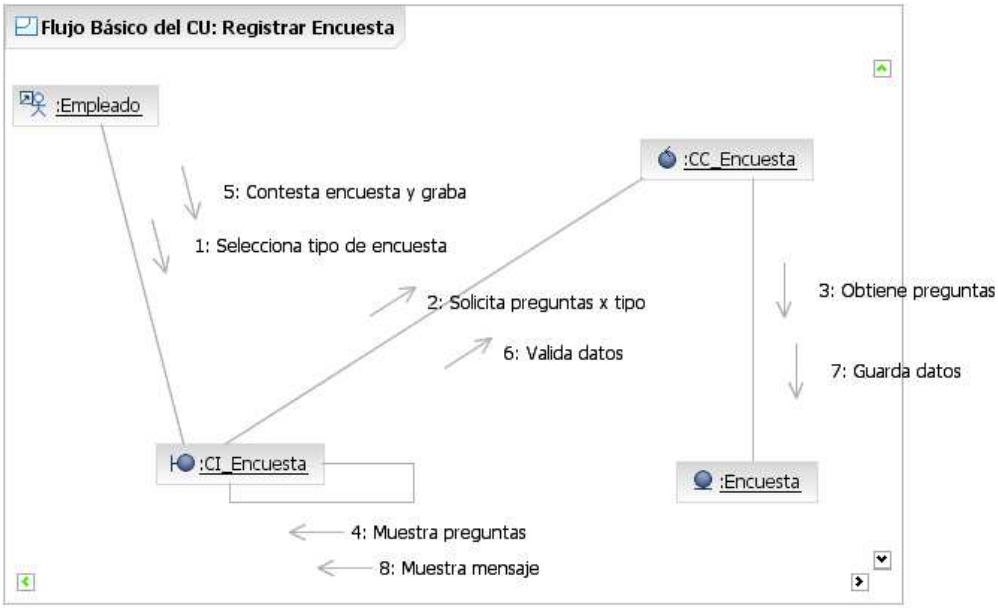


Figura 2.17. Diagrama de comunicación

1.2.16. Diagrama de tiempo

El diagrama de Tiempo define el comportamiento de los diferentes objetos con una escala de tiempo. Provee una representación visual de los objetos cambiando de estado e interactuando a lo largo del tiempo.

Puede usar diagramas de tiempos para definir componentes de software dirigidos por hardware o embebidos; por ejemplo, aquellos usados en un sistema de inyección de combustible, un controlador de microondas. También, puede usar diagramas de tiempo para especificar procesos de negocio dirigidos por tiempo.

La especificación del UML del OMG (*UML 2.0 Superstructure*, *p. 450*) establece que:

"Los Diagramas de Tiempos se usan para mostrar las interacciones cuando el propósito primario del diagrama es razonar acerca del tiempo. Los diagramas de Tiempos enfocan sobre las condiciones de cambio con y entre las Líneas de Vida a lo largo de un eje lineal de tiempo. Los diagramas de Tiempos describen el comportamiento de los clasificadores individuales y de las interacciones de los clasificadores, enfocando la atención sobre el tiempo de la ocurrencia de los eventos causando los cambios en las condiciones de modelado de la Líneas de Vida."

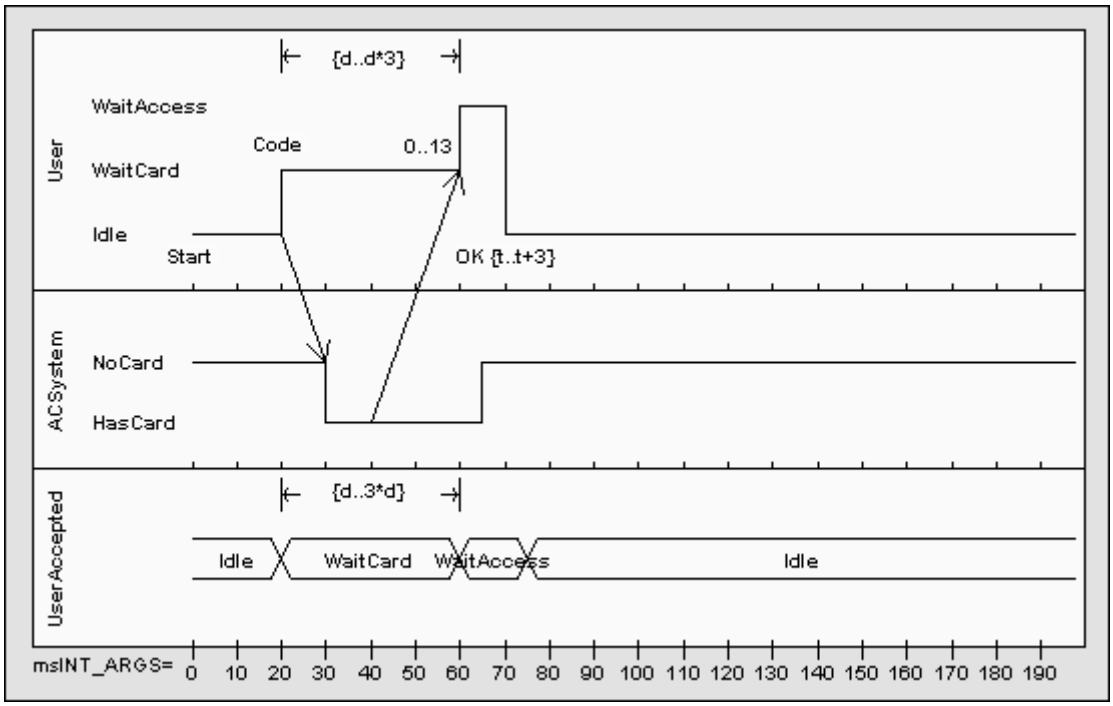


Figura 2.18. Diagrama de tiempo

1.2.17. Diagrama de descripción de la interacción

Es un diagrama que muestra cómo interactúan varios diagramas de interacciones (por ejemplo, de secuencias). Este tipo de diagramas es muy útil para mostrar de qué manera distintos escenarios se combinan.

Son una variante de los diagramas de actividades, la mayor parte de la notación es similar, al igual que el proceso de construcción del diagrama. Los puntos de decisión, bifurcación, unión, puntos de inicio y final son los mismos. En lugar de actividades, se usan elementos rectangulares. Existen dos tipos de estos elementos:

Los elementos de interacción muestran un diagrama de interacción en línea, el cual puede ser un diagrama de secuencias, comunicaciones, de tiempos o de descripción de las interacciones.

Los elementos de ocurrencia de interacción son referenciados a un diagrama de interacción existente. Ellos son visualmente representados por un marco, con **ref** en el espacio del título del marco. El nombre del diagrama está indicado en el contenido del marco.

En el ejemplo de la figura 2.19, se muestra la interacción de un cliente con un cajero ATM separado en cuatro fragmentos:

- Secuencia de login: Esta secuencia pedirá un usuario y una clave a un cliente. Se asume que la clave y usuario ingresados son válidos.

- Secuencia de seleccionar una operación: Las operaciones permitidas para este cajero ATM son cancelar o extraer dinero.
- Si cancela, se ejecutará la secuencia de deslogueo del cliente. Luego, finalizará la operación.
- Si se selecciona extraer dinero, se ejecutará la secuencia de dicha operación. Luego, finalizará la operación.

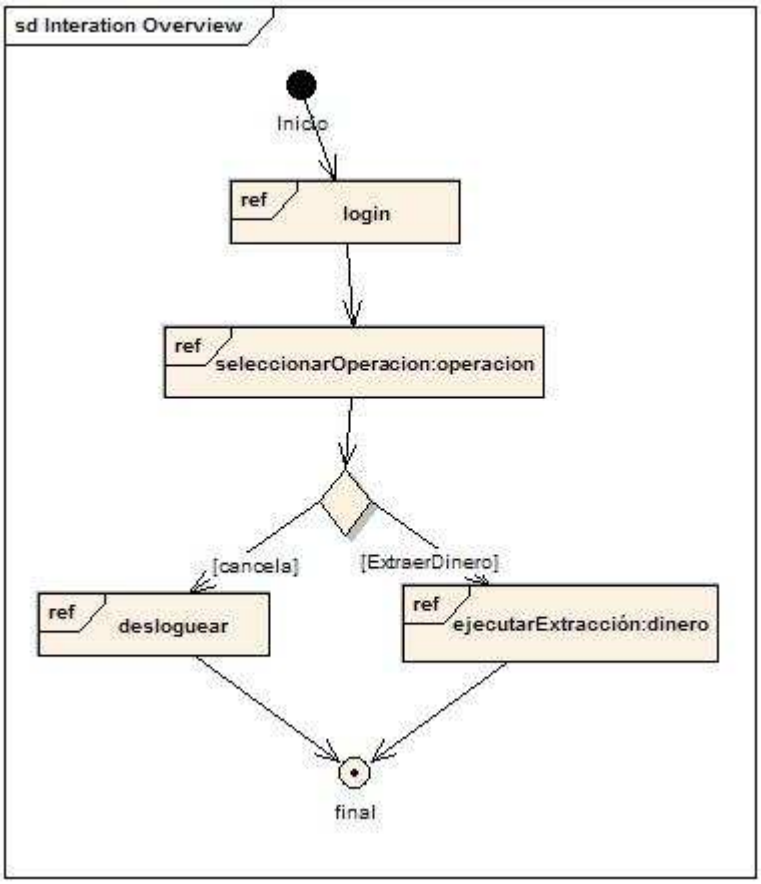


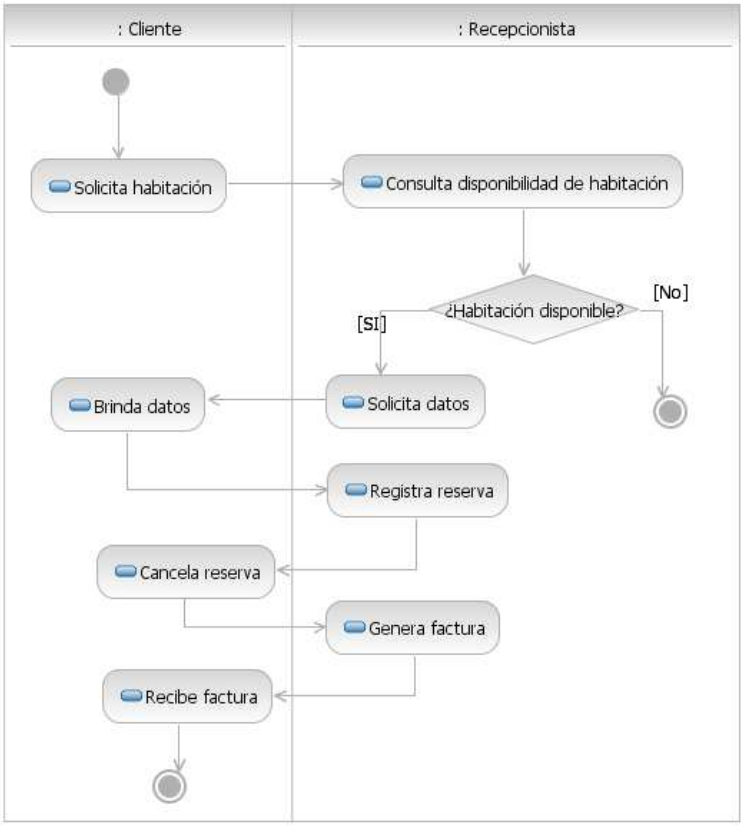
Figura 2.19. Diagrama de descripción de la interacción

ACTIVIDADES PROPUESTAS

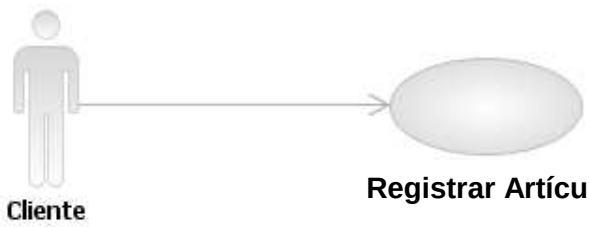
Indique los nombres de los diagramas y de sus elementos que se muestran a continuación.

1. **Diagrama** _____

2. **Diagrama** _____



3. **Diagrama** _____



Resumen

- El Modelado visual es el modelado de una aplicación usando notaciones gráficas. Debe ser preciso, consistente, fácil de comunicar, fácil de cambiar y legible.

- El Lenguaje de Modelamiento Unificado (UML) es un lenguaje de modelado visual que se usa para visualizar, especificar, construir y documentar artefactos de un sistema de software. Luego de varios años y varias modificaciones, OMG adoptó la versión oficial de UML 2.0 a principios del año 2005.

- Son cuatro las especificaciones fundamentales de UML 2.0 que permiten lograr sus objetivos: supraestructura, infraestructura, OCL e intercambio de diagramas.

- En UML 2.0 hay 13 tipos diferentes de diagramas, las cuales están distribuidas jerárquicamente en dos categorías:
  - Diagramas de estructura: el diagrama de clases, el diagrama de componentes, el diagrama de objetos, el diagrama de estructura compuesta, el diagrama de despliegue y el diagrama de paquetes
  - Diagramas de comportamiento: el diagrama de actividades, .el diagrama de casos de uso, el diagrama de máquina de estados y una subcategoría de diagramas de interacción
    - ✓ Diagramas de interacción: el diagrama de secuencia, el diagrama de comunicación, el diagrama de tiempos y el diagrama de descripción de la interacción

- Si desea saber más acerca de estos temas, puede consultar el siguiente libro.

  - "EL LENGUAJE UNIFICADO DE MODELADO. UML 2.0" de Ivar Jacobson, Grady Booch y James Rumbaugh.

    El libro que permite conocer, de forma rápida, las nuevas características de UML e ilustra su aplicación a problemas de modelado complejos en una variedad de dominios de aplicación.

- Además, puede consultar las siguientes páginas:

  - http://www.epidataconsulting.cl/site/files/UML%2020%20-%20CODE.pdf
    White Paper sobre UML 2.0

  - http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=31
    Aquí encontrará información sobre las nuevas características de los diagramas UML 2.0

  - http://www.omg.org/
    Aquí hallará información sobre la OMG.

2. MODELADO DEL NEGOCIO

Es una disciplina opcional. La necesidad de esta disciplina surge ante el hecho de que muchos de los productos software que se desarrollan automatizan algunos o todos los procesos existentes en un negocio, y es necesario estudiar las implicaciones de los cambios producidos por la adopción de estos productos. Hay que entender cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para ello, se hace un estudio en el dominio del negocio y en el dominio del software.

Así, los objetivos de esta disciplina son los siguientes:

- Entender los problemas actuales en la organización objetivo para identificar los aspectos a mejorar;
- Estudiar el impacto que pueden producir los cambios a nivel organizativo;
- Asegurar que los clientes, usuarios finales, desarrolladores y otros involucrados tienen una visión común de la organización considerada;
- Obtener los requisitos del sistema software que den soporte a la organización objetivo;
- Entender como el sistema software encaja en la organización.

Por lo tanto, el Modelo del Negocio proporciona una vista estática de la estructura de la organización y una vista dinámica de los procesos dentro de la organización.

Los creadores de RUP señalan que el modelo de negocio está soportado por dos artefactos principales:

- Modelo de casos de uso del negocio
- Modelo de análisis del negocio

El **modelo de casos de uso de negocio** describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente. Por otro lado, el **modelo de análisis del negocio** es un modelo interno a un negocio, que describe cómo cada caso de uso de negocio es llevado a cabo por un grupo de trabajadores que utilizan entidades del negocio.

El conjunto completo de artefactos del modelo de negocio, mostrado en la figura 2.20, captura y presenta el contexto del sistema, y sirven como entrada y referencia para la definición de los requisitos del sistema.

2.1 ¿Cuándo será necesario hacer el modelado de negocio?

- Cuando el grupo de trabajo es nuevo en la organización.
- Cuando la organización a enfrentado un reciente proceso de reingeniería de negocios.
- Cuando la organización esta planificando un proceso de reingeniería de negocios.
- Cuando el software que se va a construir será utilizado por una parte importante de la organización.
- Cuando existen flujos de trabajo complejos dentro de la organización que no están documentados.
- Cuando se es un consultor en una organización en la cuál no se ha trabajado antes.

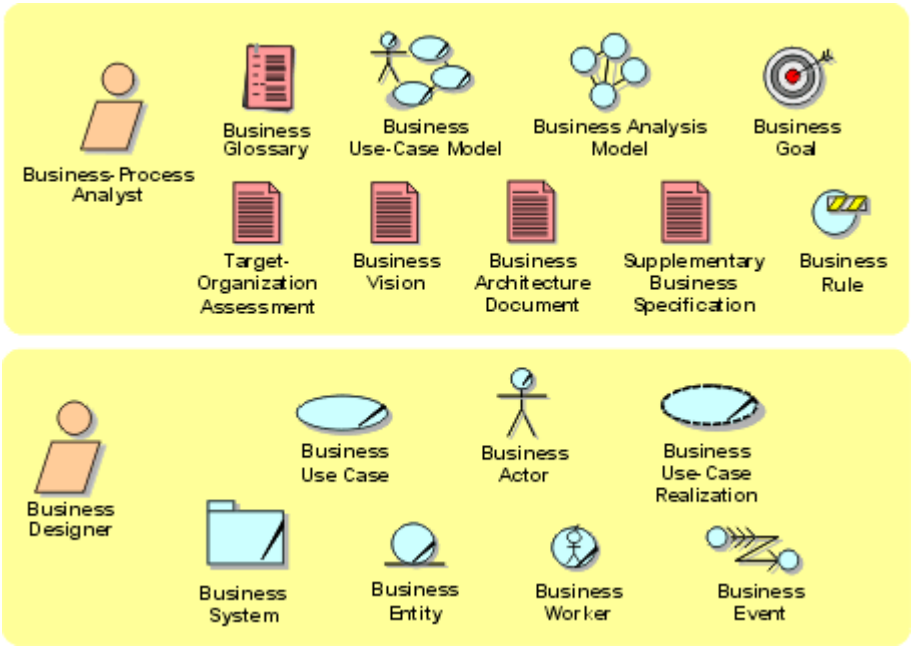


Figura 2.20. Artefactos del modelado de negocio.

2.2 ¿Cuándo no será necesario hacer el modelado de negocio?

- Cuando se tiene un conocimiento de la estructura de la organización, de las metas, de la visión y de los clientes/usuarios.
- Cuando el software a construir será usado por una pequeña parte de la organización, y no tiene un efecto en el resto del negocio.
- Cuando los flujos de trabajo de la organización están bien documentados.
- Cuando el tiempo no lo permita, no todos los procesos tienen el tiempo necesario para completar un análisis de negocio.

2.3 Actividades para realizar un modelado de negocio

Según RUP, el modelado de negocio comprende las siguientes actividades: (Ver figura 2.21)

- Determinar la situación de la organización;
- Describir el actual negocio;
- Identificar los procesos de negocio;
- Refinar las definiciones de los procesos de negocio;
- Diseñar las realizaciones de los procesos de negocio;
- Refinar roles y responsabilidades;
- Explorar procesos automatizados;
- Desarrollar un modelado de dominio.

En este apartado, trataremos la ejecución de actividades relevantes que permiten obtener los artefactos principales del modelo de negocio.

Los pasos que contemplaremos para obtener el Modelo de casos de uso del negocio son:

- Determinar la situación de la organización;
- Identificar los procesos de negocio;
- Refinar las definiciones de los procesos de negocio;

Por último, las actividades que ejecutaremos para obtener el modelo de análisis del negocio es:

- Diseñar las realizaciones de los procesos de negocio
- Refinar los roles y responsabilidades

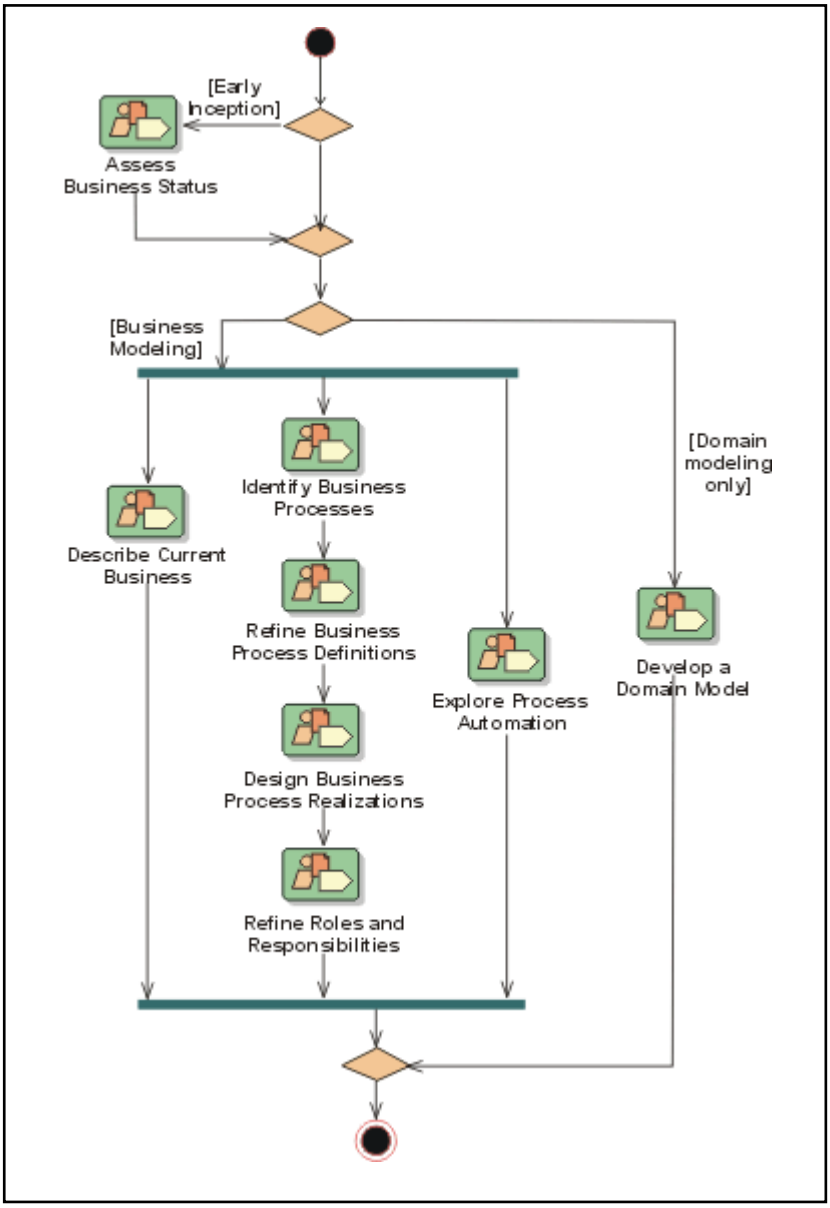


Figura 2.21. El modelado de negocio

3. MODELO DE CASOS DE USO DEL NEGOCIO

3.1 Determinar la situación de la organización

El objetivo es reconocer el negocio en estudio para delimitarlo. Las actividades que se llevan a cabo son:

a) Identificar la misión y visión de la organización y/o áreas de estudio que correspondan y plasmarlo en el documento Visión del negocio.
b) Identificar los objetivos del negocio, y documentarlos en los objetivos del negocio. Estos objetivos son determinados por los stakeholders y responsables del negocio y serán usados para validar los casos de uso del negocio.
c) Identificar las reglas del negocio y luego plasmarlas en las reglas del negocio.
d) Elaborar una lista de términos y definiciones usados comúnmente en el glosario del negocio.

Se ha preferido reunir los documentos anteriormente mencionados en el artefacto ***Situación del negocio***.

3.2 Identificar los procesos de negocio

Requiere haber identificado los objetivos del negocio. El equipo de trabajo debe tener claras las fronteras del negocio que está describiendo. Para ello, debe identificar y priorizar los casos de uso del negocio y los actores de negocio involucrados.

Para mostrar la interacción entre actores de negocio y casos de uso de negocio, se crea un ***Diagrama general de casos de uso de negocio***.

Por cada caso de uso del negocio, se realiza una ***Especificación de caso de uso del negocio***. En este documento, se indica una descripción breve del proceso de negocio.

Para describir los actores del negocio y su asociación con los casos de uso de negocio encontrados, se utiliza el artefacto ***Actores del negocio***.

3.3 Refinar las definiciones de los procesos de negocio

Consiste en:

a) Detallar la definición de los casos de uso del negocio en su ***ECUN;***
b) Describir cómo los casos de uso del negocio soportan los objetivos de negocio;
c) Verificar que los casos de uso del negocio representan correctamente cómo el negocio es conducido.

Aquí se refina la ***especificación de caso de uso del negocio***, describiendo paso a paso las actividades que se desarrollan en el proceso de negocio.

4. MODELO DE ANÁLISIS DEL NEGOCIO

4.1 Diseñar las realizaciones de los procesos de negocio

Consiste en identificar todos los roles, productos, entregables del negocio y describir cómo el proceso del negocio será llevado a cabo por los trabajadores y las entidades dentro del negocio.

El documento que plasma la descripción breve de trabajadores del negocio y cómo ellos manipulan las entidades del negocio es ***Trabajadores del negocio***.

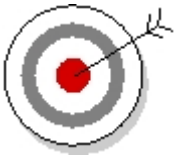
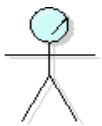
Además se crea el artefacto ***Entidades del Negocio*** para describir las entidades y especificar, mediante diagramas de estado, sus estados.

Para la realización de cada proceso del negocio se crea un ***diagrama de clases de negocio*** y un ***diagrama de actividades de negocio***.

Al finalizar esta actividad, se completará cada ***especificación de caso de uso del negocio*** generado en el modelo de casos de uso de negocio, agregando al final de cada documento, los diagramas de clases y actividades correspondientes.

4.2 Refinar roles y responsabilidades

Consiste en detallar más los documentos ***trabajadores del negocio*** y ***entidades del negocio.***

Artefacto	Descripción
 Situación del Negocio	Documento que contiene la visión del negocio, un glosario de términos del negocio, los objetivos del negocio y reglas del negocio.
 Objetivos del Negocio	Es un requisito que debe ser satisfecho por el negocio. Describe el valor deseado de una medida en particular a futuro, y se utiliza para planear y administrar las actividades del negocio. El objetivo debe ser claro, mesurable, alcanzable, realista y sensible al tiempo. Se permite la relación de dependencia entre objetivos del negocio y la de soporte de un caso de uso del negocio.
 Casos de Uso del Negocio	Define un conjunto de acciones que el negocio lleva a cabo y provee resultados de valor a quienes interactúan con el. Describe un proceso de negocio desde un punto de vista externo que percibe algún tipo de valor. Definen los límites de la organización.
 Actor del Negocio	Representa un rol que algo o alguien externo desempeña en relación con el negocio. Puede ser asociado a uno ó más casos de uso del negocio.

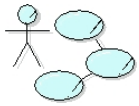
 Modelo de Casos de Uso del Negocio	Representa la vista externa del negocio. Modelo que describe la dirección e intención del negocio. La dirección es provista por los objetivos del negocio. Mientras que la intención es expresada por los diagramas que permiten ver cómo interactuar con el entorno.
 Actores del Negocio	Documento que contiene información de los actores del negocio identificados en el modelo de casos de uso del negocio.
 Especificación de Caso de Uso del Negocio	Documento que contiene las características de un proceso de negocio. Se realiza una especificación por cada caso de uso de negocio.

Tabla 2.1. Artefactos del modelo de casos de uso del negocio

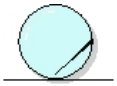
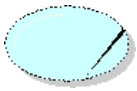
Artefacto	Descripción
 Trabajadores del Negocio	Un trabajador del negocio es un rol interno al negocio. Colabora con trabajadores de otro sector, es notificado de acontecimientos del negocio y manipula entidades de negocio para realizar sus responsabilidades.
 Entidades del Negocio	Ente significativo y persistente manipulado por actores del negocio y trabajadores del negocio. Hay dos tipos de entidades: - Informativos (documentos) - Persistentes (fichas de datos)
 Realización de Caso de Uso del Negocio	Colección de diagramas que muestra cómo los actores y/o trabajadores del negocio y entidades del negocio llevan a cabo el caso de uso del negocio. Generalmente, se utilizan diagramas de clases y diagramas de actividades para realizar el detalle de cada proceso de negocio.
 Modelo de Análisis del Negocio	Representa la vista interna del negocio. Es un modelo que describe la realización de los casos de uso del negocio. Es una abstracción de cómo los trabajadores del negocio y las entidades de negocio se relacionan y de cómo colaboran para realizar los casos del uso del negocio.
 Trabajadores del Negocio	Documento que contiene información de los trabajadores del negocio identificados en el modelo de análisis del negocio.
 Entidades del Negocio	Documento que contiene información de las entidades del negocio identificadas en el modelo de análisis del negocio.

Tabla 2.2. Artefactos del modelo de análisis del negocio

En el ANEXO 1, se muestran las plantillas de los documentos que se crean en la disciplina del modelado de negocio en el siguiente orden:

- Situación del negocio
- Actores del negocio
- Especificación de caso de uso del negocio
- Trabajadores del negocio
- Entidades del negocio

Resumen

- El Modelado del negocio es una técnica para comprender los procesos de negocio de la organización objetivo.
- El modelo de casos de uso del negocio es un modelo elaborado bajo una perspectiva externa del negocio.
- El modelo de análisis del negocio detalla cómo el proceso de negocio es implementado internamente.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  http://www-128.ibm.com/developerworks/rational/library/5167.html

  Aquí hallará una descripción de los elementos del modelado de negocio.

5. CASOS DE ESTUDIO N°1

5.1 CASO DESARROLLADO

Según el flujo de trabajo propuesto, desarrolle el modelado de negocio, el cual debe comprender:

1) Actores de negocio
2) Casos de uso de negocio
3) Diagrama de casos de uso de negocio
4) Realizaciones de casos de uso de negocio
   a) Diagrama de clases del negocio
   b) Diagrama de actividades del negocio

ECUN: Retiro y cambio de curso

Flujo de trabajo

1. Flujo básico

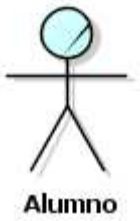
1.1. El alumno solicita al asistente SA realizar un retiro o cambio de horario de un curso.
1.2. El asistente SA verifica si el alumno está matriculado en un determinado curso.
1.3. Si está matriculado en un determinado curso, el asistente SA verifica si la solicitud está dentro de la fecha límite de presentación.
1.4. Si la solicitud está dentro de la fecha límite, el asistente SA verifica si corresponde a un retiro o a un cambio de horario.
1.5. Si es retiro de un curso
   1.5.1. El asistente SA registra retiro de alumno en documento Acción Académica.
   1.5.2. El asistente SA actualiza estado de alumno a retirado.
1.6. Si es un cambio de horario
   1.6.1. El asistente SA muestra los horarios con vacantes disponibles.
   1.6.2. El alumno selecciona el horario del curso.
   1.6.3. El asistente SA actualiza horario de alumno.
   1.6.4. El asistente SA registra cambio de horario en documento Acción Académica.
1.7. El asistente SA actualiza los datos referentes a la matrícula del alumno y finaliza el proceso.

2. Flujos alternativos

2.1. En el punto 1.2, si el alumno no está matriculado, es rechazado.
2.2. En el punto 1.3, si la solicitud ha pasado la fecha límite, ésta es rechazada.

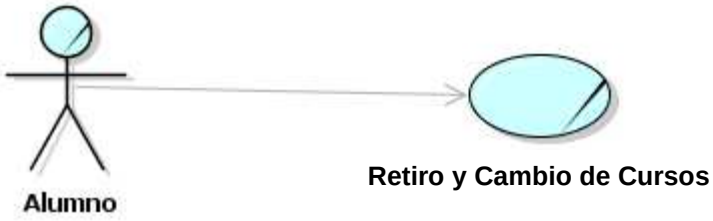
SOLUCIÓN DEL CASO

1) Actores de negocio

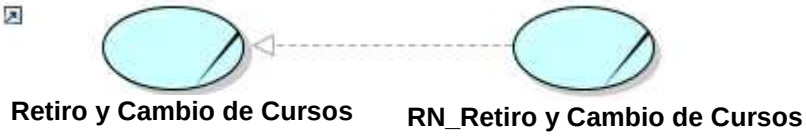


2) Casos de uso de negocio

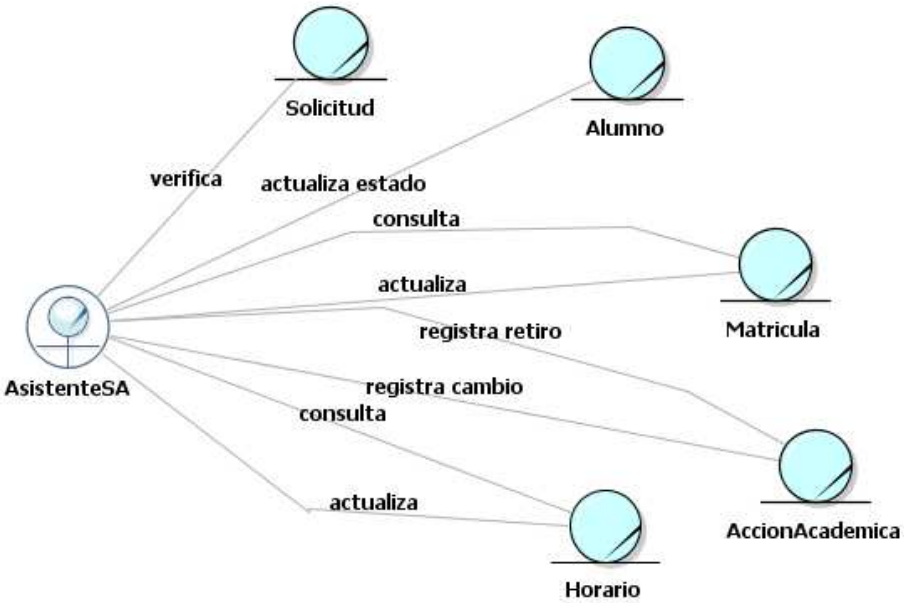
3) Diagrama de casos de uso de negocio



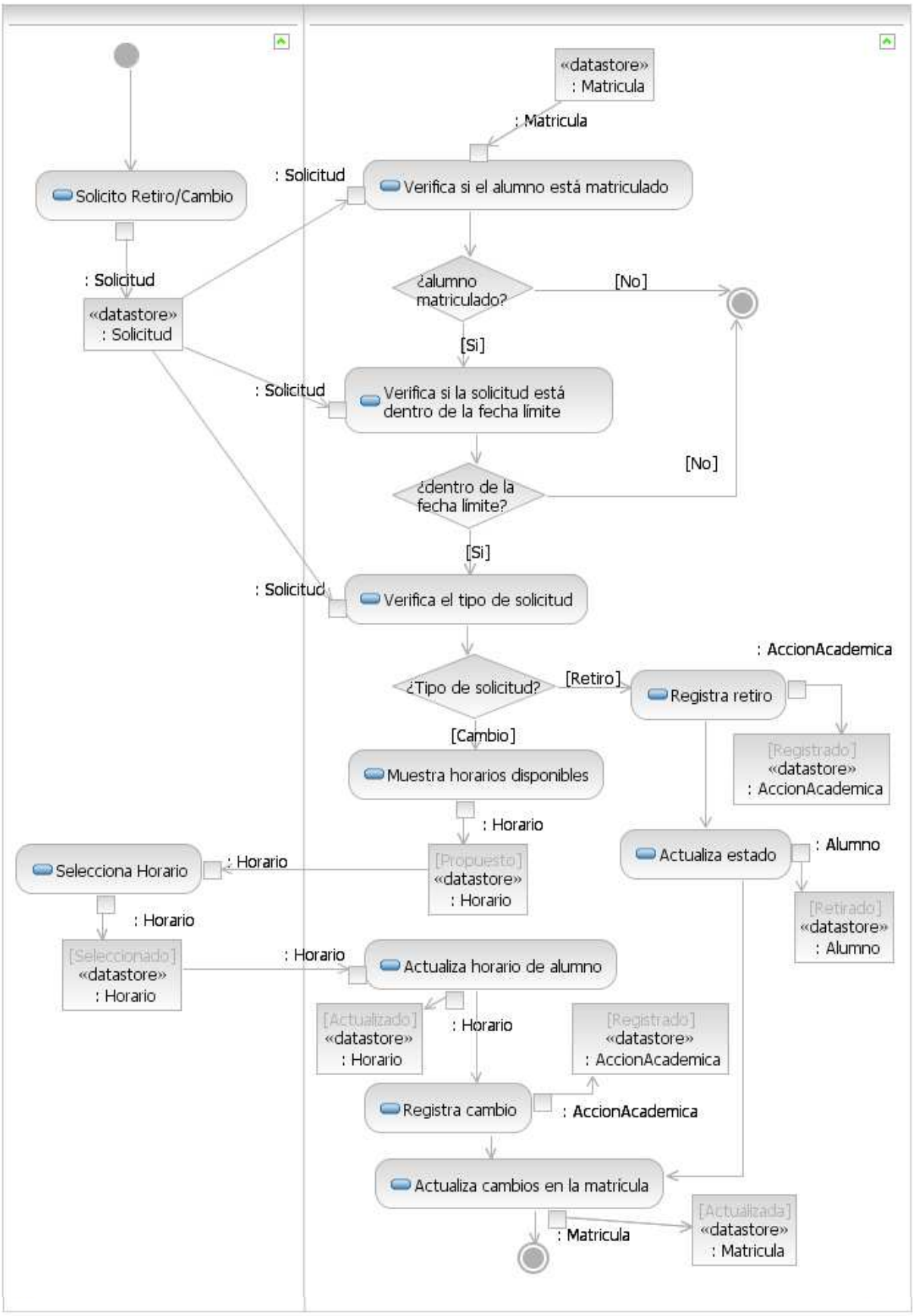
4) Realizaciones de casos de uso de negocio



a) Diagrama de clases del negocio



b) Diagrama de actividades del negocio



5.2 CASO PROPUESTO

Según el flujo de trabajo propuesto, desarrolle el Modelo de Negocio, el cual debe comprender:

1) Actores de negocio
2) Casos de uso de negocio
3) Diagrama de casos de uso de negocio
4) Realizaciones de casos de uso de negocio
   c) Diagrama de clases del negocio
   d) Diagrama de actividades del negocio

ECUN: Atención de servicio de exposiciones de arte

Flujo de trabajo

1. Flujo básico

1.1. El artista solicita el servicio de la galería para una exposición.
1.2. El anfitrión solicita los datos personales del artista.
1.3. El artista entrega sus datos personales al anfitrión.
1.4. El anfitrión busca si los datos del Artista están registrados previamente en la galería.
1.5. El anfitrión solicita información de las obras de arte al artista.
1.6. El artista entrega la información de las obras al anfitrión.
1.7. El anfitrión registra la información de las obras de arte.
1.8. El anfitrión busca información sobre las técnicas que maneja la galería en el sistema LogiSis.
1.9. El sistema LogiSis entrega información sobre las técnicas que maneja la galería.
1.10. El anfitrión recibe la información sobre las técnicas y determina si la galería maneja las técnicas de las obras de arte.
1.11. El anfitrión llena la solicitud de servicio.
1.12. El anfitrión archiva la solicitud de servicio y entrega una copia al artista.
1.13. El artista recibe la copia de la solicitud de servicio y finaliza el proceso.

2. Flujos alternativos

Los datos personales del artista no están registrados previamente.

2.1. En el punto 1.4, si la galería no cuenta con los datos personales del artista.

2.1.1. El anfitrión registra los datos personales del artista.

2.1.2. El caso de uso continúa en el punto 1.5 del flujo básico.

La galería no domina las técnicas de alguna obra.

2.2. En el punto 1.10, si la galería no domina la técnica de alguna obra.

2.2.1. El anfitrión elabora el documento de rechazo de pedido.

2.2.2. El anfitrión archiva el documento de rechazo de pedido y entrega una copia al artista.

2.2.3. El artista recibe la copia del documento de rechazo de pedido y finaliza el proceso.

6. CASOS DE ESTUDIO N°2

Para los siguientes casos propuestos elabore:

1) El diagrama de casos de uso del negocio Vs. objetivos del negocio
2) El diagrama general de casos de uso de negocio

CASO: "CIBER PARK"

"Ciber Park" es una empresa dedicada al alquiler de juegos mecánicos, brinda servicios para empresas o personas que deseen organizar fiestas infantiles con juegos mecánicos.

Para atender los servicios de alquiler, la secretaria de la empresa consulta el catálogo de juegos mecánicos y le indica al cliente las características de cada una de ellas. Una vez que el cliente elige los juegos, la secretaria verifica disponibilidad de dichos juegos. Si el cliente es nuevo, la secretaria le pide al cliente que llene una hoja de datos para registrarlo como tal, de lo contrario la secretaria procede a elaborar el contrato. En el contrato se especifica los juegos y el lugar donde desea que se instalen dichos juegos. Si después de evaluar el contrato, el cliente está conforme, lo firma y la secretaria genera una orden de pago; el cliente realiza el pago correspondiente y la secretaria registra el alquiler de juegos.

Por otro lado, la empresa controla la instalación de los juegos mecánicos en el lugar que el cliente indicó. Para ello, el jefe de personal asigna el trabajo de instalación a uno o más técnicos, dependiendo de los juegos que se deben instalar. Al final, ellos son los encargados de realizar un informe técnico de la instalación de los juegos en el local del evento.

CASO: CONSULTORIA DE ABOGADOS SAC

"Consultoría de Abogados SAC" es un estudio de abogados que brinda asesoría legal efectiva para todo tipo de área, sea penal, civil, o del tipo familia.

La secretaria se encarga de atender a los clientes. Si el cliente es nuevo, la secretaria le informará la tarifa por hora cobrada por los doctores (abogados); si el cliente acepta, la secretaria le pedirá sus datos personales (nombres, apellidos, DNI, dirección, teléfono, etc.) para registrarlo en una hoja de Excel. Luego, la secretaria indica al cliente que pase a la sala de espera para entrevistarse con el Abogado Jefe.

La secretaria informa al abogado jefe disponible que atienda el caso del cliente. El abogado jefe se entrevista con el cliente, quién expone su caso. Luego, el abogado jefe, le explica las posibles soluciones y le pide que regrese dentro de un plazo máximo que puede ser de 3 días o un mes, según sea el caso, pues, en ese lapso el estudio realiza un minucioso análisis del caso para elaborar la mejor estrategia.

Luego del plazo dado, el cliente regresa al estudio para entrevistarse con los abogados de menor rango a cargo del abogado jefe. Ellos explican al cliente un análisis de riesgos, donde se le indican los riesgos a tomar; asimismo indica el presupuesto aproximado para el seguimiento del caso. Si el cliente acepta, se genera un contrato del servicio.

Por otro lado, el estudio de abogados necesita un preciso y exhaustivo seguimiento de los casos que se trabajan, el cual es registrado en un expediente de caso generado después de que el cliente firma el contrato. Este seguimiento incluye registro de los documentos generados en cada actividad del proceso, estrategias y esquemas desarrollados en cada entrevista con el cliente realizado por los abogados de menor rango.

7. CASOS DE ESTUDIO N°3

Para los siguientes flujos de trabajo propuestos elabore la realización del negocio, el cual comprende:

1) El diagrama de clases del negocio
3) El diagrama de actividades del negocio

ECUN: Atención de pacientes

Flujo de trabajo

1. Flujo básico

2.1. El paciente solicita su atención de cita en el consultorio que le corresponde.
2.2. La asistente de especialidad le solicita el carné de atención al paciente.
2.3. La asistente de especialidad verifica la fecha y hora de la cita para que le permita su ingreso.
2.4. Si la fecha y hora es correcto, la asistente pide al paciente que pase al consultorio.
2.5. El médico de especialidad verifica el historial clínico del paciente para revisar lo que ha diagnosticado el médico general.
2.6. El médico evalúa al paciente y le indica el tratamiento que debe seguir
2.7. El médico actualiza el historial clínico del paciente y lo entrega a su asistente.
2.8. El asistente de especialidad actualiza el carné de atención del paciente con la próxima fecha de la cita.
2.9. El asistente de especialidad entrega el carné de atención al paciente.
2.10. El asistente de especialidad elabora y entrega el presupuesto del tratamiento al paciente y finaliza el proceso.

2. Flujo alternativo

1.1. En el punto 1.3, si el asistente de especialidad verifica que el horario no es el correcto, entonces informa al paciente del horario que le corresponde y finaliza el proceso.

ECUN: Venta de pasajes

Flujo de trabajo

1. Flujo básico

1.1. El cliente se acerca a ventanilla y pregunta los horarios que hay de salida para su destino deseado.

1.2. La asistente de ventas informa sobre los horarios y tipos de servicios que tienen cada uno de ellos.

1.3. El cliente selecciona uno de los horarios y servicios ofrecidos, indicando el número de pasajes deseados.

1.4. La asistente de ventas procede a reservar y confirmar los pasajes requeridos.

1.5. La asistente de ventas pregunta sobre algún tipo de preferencia y/o restricción de los pasajeros en la alimentación a brindarse durante el servicio.

1.6. El cliente indica preferencias y/o restricciones en la alimentación de cada uno de los pasajeros para los que esta adquiriendo sus boletos.

1.7. La asistente de ventas procede a registrar las preferencias y/o restricciones indicadas.

1.8. La asistente de ventas informa el monto a pagar y pregunta modalidad de pago que prefiere el cliente.

1.9. El cliente indica modalidad de pago deseada y procede a realizar el pago.

1.10. La asistente de ventas procede a emitir los pasajes solicitados.

1.11. El cliente recibe los pasajes con la conformidad de los mismos y finaliza el proceso.

2. Flujo alternativo

2.1. En el punto 1.4, si no existe la cantidad suficiente de pasajes según lo solicitado, la asistente de ventas informará de este hecho y solicitará la selección de otro horario y/o tipo de servicio. De no escoger ninguno de ellos, termina la atención; de lo contrario, se continúa con el paso 1.5.

2.2. En el punto 1.9, si la forma de cancelación es con tarjeta de crédito o de débito, el asistente de ventas solicita al cliente pasar su tarjeta a través de un terminal de banco. Luego, el asistente entrega el voucher generado por el terminal. El flujo continúa en el paso 1.10.

UNIDAD DE APRENDIZAJE

3

DISCIPLINA DE LA CAPTURA DE REQUISITOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, los alumnos, trabajando en equipo, elaboran y sustentan su proyecto final sobre el modelado del negocio y la captura de requisitos, en el que identifica el modelo de casos de uso del negocio, el modelo de análisis del negocio, y el modelo de casos de uso con sus respectivos artefactos, aplicando la metodología RUP, el lenguaje de modelado UML y la herramienta IBM Rational Software Architect.

TEMARIO

1. Captura de requisitos
2. Captura de requisitos a solicitud del cliente
3. Captura de requisitos a partir del Diagrama de actividades del negocio. Matriz de actividades y requisitos

ACTIVIDADES PROPUESTAS

1. Los alumnos clasifican los requisitos de una lista propuesta según las categorías descritas por el modelo FURPS+.
2. Los alumnos identifican los requisitos funcionales a partir de un diagrama de actividades del negocio.

1. CAPTURA DE REQUISITOS

El esfuerzo principal en esta disciplina es desarrollar un modelo del sistema que se va a construir. La utilización de los casos de uso es una forma adecuada de crear ese modelo. Esto es debido a que los requisitos funcionales se estructuran de forma natural mediante casos de uso.

Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo. Un caso de uso puede contener uno o más requisitos funcionales.

El modelo de casos de uso es construido a través de un proceso iterativo durante el cual las discusiones entre los desarrolladores del sistema y los clientes (y/o usuarios finales) llevan a una especificación de requisitos en la que todos estén de acuerdo.

Así, los propósitos de la disciplina Captura de requisitos son:

- Establecer y mantener los acuerdos con los clientes y otros interesados (stakeholders) sobre lo que el sistema debe hacer;
- Proporcionar a los desarrolladores un mejor entendimiento de los requisitos del sistema;
- Definir las fronteras del sistema;
- Proveer la base para planificar las iteraciones;
- Proporcionar la base para estimar los costos y tiempos del desarrollo del sistema;
- Definir las interfaces de usuario con el sistema, enfocado a las necesidades y objetivos de los usuarios.

1.1 Artefactos de la captura de requisitos

El conjunto completo de artefactos de la captura de requisitos, mostrado en la figura 3.1, sirven como entrada y referencia para el análisis, diseño, implementación y pruebas del sistema.

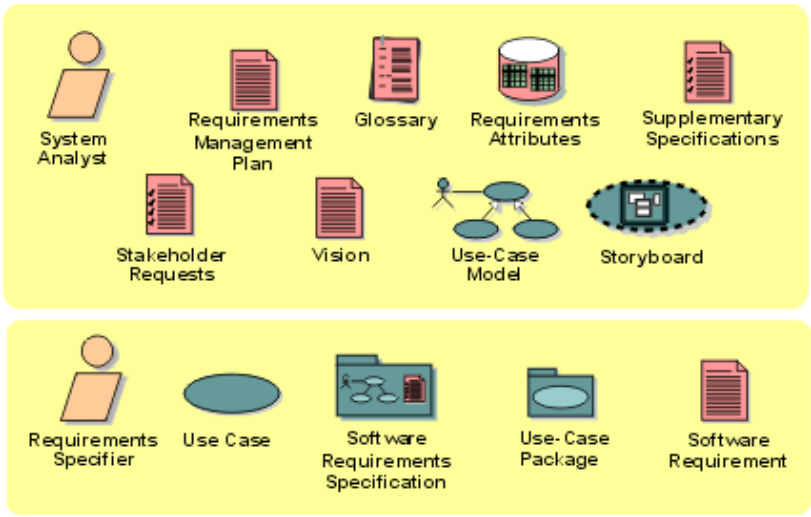


Figura 3.1. Artefactos de la captura de requisitos.

La propuesta del curso, para una solución de mediana envergadura, es crear los artefactos proporcionados en la tabla 3.1.

Artefacto	Descripción

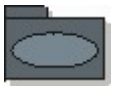
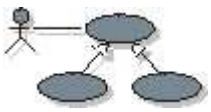
 Visión	Documento que define la opinión de los stakeholders del producto que se desarrollará, especificada en términos de necesidades y características claves de los stakeholders. Contiene un esquema de los requisitos previstos, el cual proporciona la base contractual para los requisitos técnicos más detallados.
 Especificación de Requisitos de Software	La especificación de requisitos de software es un documento que enfoca la organización completa de los requisitos del proyecto. Comúnmente conocido como SRS por sus iniciales en inglés. Contiene la lista de requisitos funcionales y no funcionales.
 Paquetes de Casos de Uso	Es una colección de casos de uso, de actores, de relaciones, de diagramas, y de otros paquetes, de ser necesario; es utilizado para estructurar el modelo de casos de uso dividiéndolo en piezas más pequeñas.
 Caso de Uso	Es una funcionalidad específica del sistema con identidad propia, el cual define una secuencia de acciones que el sistema realiza para un actor en particular. Un caso de uso contiene uno o más requisitos funcionales.
 Actor	Representa un rol (humano, hardware o software) externo al sistema con el que se establece intercambio directo de información. Puede ser asociado a uno ó más casos de uso.
 Modelo de Casos de Uso	Es un modelo que captura los requisitos funcionales de los usuarios a un alto nivel y establece la estructura fundamental del sistema. Es un input esencial para las actividades en análisis, diseño y pruebas.
 Actor	Es un documento que contiene información de los actores identificados en el modelo de casos de uso.
 Especificación de Caso de Uso	Documento que contiene las características de un caso de uso. Contiene, primordialmente, una descripción del flujo de eventos que describen la interacción entre los actores y el sistema. La especificación, también, contiene otra información, tal como precondiciones, poscondiciones, requisitos especiales y prototipos. Se realiza una especificación por caso de uso.
 Especificación Suplementaria	Documento que especifica los requisitos funcionales que no son traducidos a casos de uso y los requisitos no funcionales.

Tabla 3.1. Artefactos del modelo de casos de uso.

1.2 Actividades para realizar la captura de requisitos

Según RUP, la captura de requisitos comprende las siguientes actividades:

- Analizar el problema
- Entender las necesidades de stakeholders
- Definir el sistema
- Administrar el alcance del sistema
- Refinar la definición del sistema
- Administrar cambios de requisitos

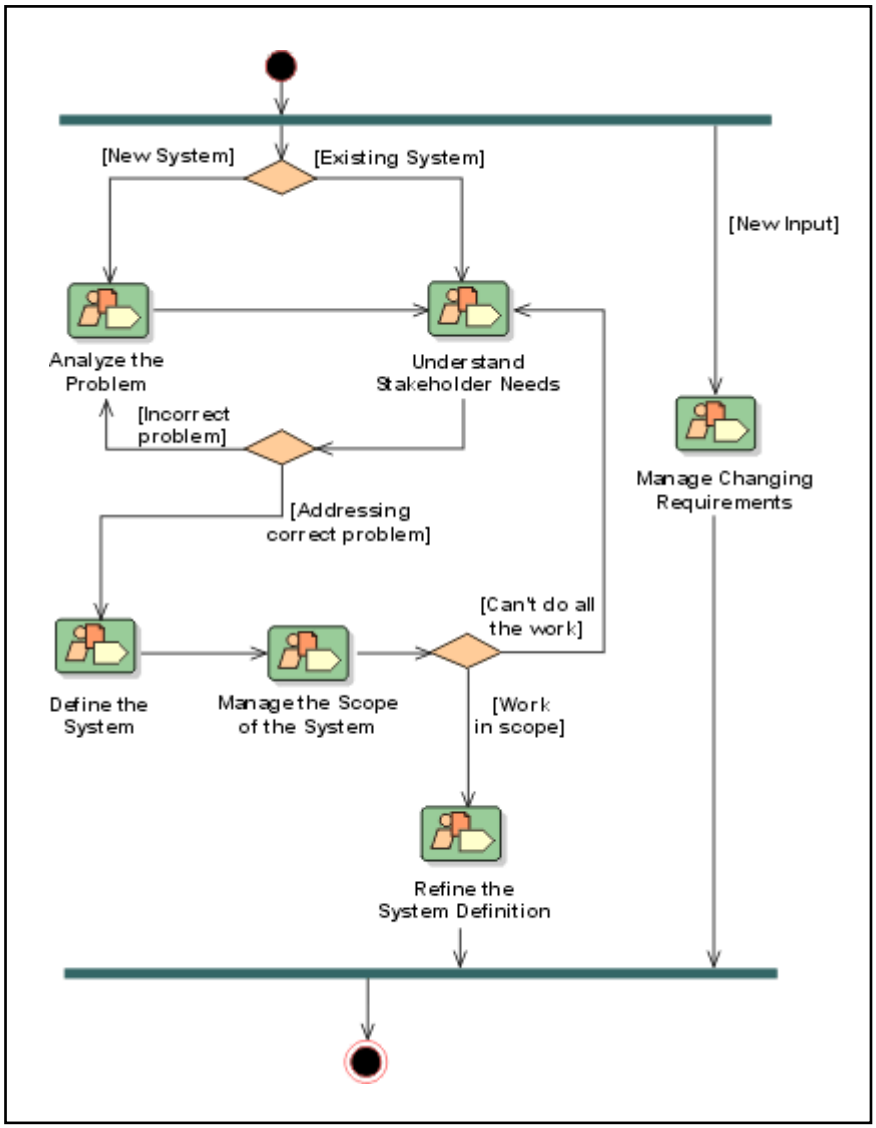


Figura 3.2. La captura de requisitos.

1.2.1 Analizar el problema

El documento **visión** es el principal artefacto en el cual el análisis del problema es documentado.

Para determinar el alcance inicial del proyecto, los límites del sistema deben ser definidos. El analista de sistema identifica usuarios y sistemas, representado por actores, los cuales interactúan con el

sistema. En este caso, el analista crea el modelo de casos de uso que contendrá sólo los actores.

1.2.2 Entender las necesidades del stakeholder

El artefacto principal es un documento refinado de la **visión**. También, los requisitos son discutidos y expresados en términos de casos de uso y actores. Los requisitos no funcionales, que no son representados en el modelo de casos de uso deberán ser documentados en especificaciones suplementarias.

El analista se relaciona con los stakeholders utilizando técnicas para capturar requisitos, tales como las entrevistas si se encuentra en las primeras iteraciones de esta disciplina y prototipos si se encuentra en las últimas iteraciones.

Los stakeholders son un grupo de personas cuyas necesidades deben ser satisfechas por el proyecto. El papel puede ser desempeñado por cualquier persona que es (o será potencialmente) afectado por los resultados del proyecto. Por lo tanto, son fuentes de requisitos, por ejemplo, usuarios finales del sistema, gerentes, accionistas, reguladores quiénes certifican la aceptabilidad del sistema.

1.2.3 Definir el sistema

En definir el sistema, se enfoca en identificar a los actores y los casos de uso completamente para obtener un modelo de casos de uso refinado y expandir los requisitos no funcionales definidos en los documentos de especificaciones suplementarias.

1.2.4 Administrar el alcance del sistema

El alcance del proyecto es definido por el conjunto de requisitos definidos para éste. La clave para manejar un proyecto exitoso es administrar el alcance del proyecto cumpliendo con los recursos disponibles tales como el tiempo, la gente y el dinero. La priorización los casos de uso, desarrollado por el arquitecto de software, permite planificar el proyecto.

1.2.5 Refinar la definición del sistema

El resultado de este flujo de trabajo del RUP es una comprensión más profunda de la funcionalidad del sistema expresada en casos de uso detallados y documentos de especificaciones suplementarias detallados. Si es necesario, una especificación de requisitos de software formal puede ser desarrollada, además de los documentos detallados de casos de uso y especificaciones suplementarias.

1.2.6 Administrar los cambios de requisitos

Los cambios a los requisitos impactan los modelos producidos en la disciplina de análisis y diseño, el modelo de pruebas creado en la disciplina de pruebas y el material de soporte al usuario final de la disciplina de despliegue. Las relaciones de trazabilidad son establecidas para identificar las relaciones entre los requisitos y otros artefactos. Las relaciones de trazabilidad son la clave para entender el impacto del cambio de los requisitos.

2. REQUISITOS

Un requisito se define como una **condición o capacidad** a la que debe ajustarse el sistema que se construye para satisfacer un contrato, norma, especificación u otro documento formalmente impuesto.

El proceso de recopilar, analizar y verificar las necesidades del cliente o usuario para un sistema es llamado ingeniería de requisitos. La meta de la Ingeniería de requisitos (IR) es entregar una especificación de requisitos de software correcta y completa.

Algunos otros conceptos de Ingeniería de requisitos son:

Según Pressman “Ingeniería de Requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software”.

Por otro lado, Sommerville define que “La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema”.

En síntesis, el proceso de ingeniería de requisitos se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requisitos para un producto de software determinado, donde es muy importante tomar en cuenta que el aporte de la IR vendrá a ayudar a determinar la viabilidad de llevar a cabo el software (si es factible llevarlo a cabo o no), pasando posteriormente por un subproceso de obtención y análisis de requisitos, su especificación formal, para finalizar con el subproceso de validación donde se verifica que los requisitos realmente definen el sistema que quiere el cliente.

2.1 Tipos de requisitos

Existen dos tipos de requisitos: requisitos funcionales y requisitos no funcionales.

2.1.1 Requisitos funcionales

Son lo que los usuarios requieren que el sistema haga. Son usados para expresar el comportamiento de un sistema, especificando las condiciones de entrada y salida que el sistema debe cumplir. Los casos de uso son usados para establecer lo que el sistema debe hacer. Un estudio profundo del área de estudio usando casos de uso permite conocer las necesidades de los usuarios. Estos requisitos pueden establecerse más claramente usando prototipos.

2.1.2 Requisitos no funcionales

Son restricciones que especifican propiedades del sistema, tales como facilidad de uso, restricciones del entorno o de implementación, rendimiento, dependencias de plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad y escalabilidad.

El incumplimiento de un requerimiento no funcional puede significar que el sistema entero sea inutilizable. Por ejemplo, si un sistema de

contabilidad no cumple sus requisitos de fiabilidad, no se certificará como seguro para el funcionamiento; si un sistema de control de tiempo real no cumple sus requisitos de rendimiento, las funciones de control no funcionarían correctamente.

2.2 Requisitos FURPS+

Este es un tipo de clasificación de requisitos especificado en la documentación de RUP. Se utiliza el acrónimo FURPS (por las siglas en inglés) para describir las principales categorías de requisitos:

- Funcionalidad (Functionality)
- Facilidad de uso (Usability)
- Confiabilidad (Reliability)
- Rendimiento (Performance)
- Soporte (Supportability)

El símbolo "+" en **FURPS+** hace referencia a que se deben incluir otros requisitos, tales como:

- Restricciones de diseño
- Requisitos de implementación
- Requisitos de interfaz
- Requisitos físicos

2.2.1 Funcionales

Los requisitos funcionales deben incluir:

- Conjunto de características
- Capacidades
- Seguridad

Por ejemplo, para un Sistema de Ventas:

R1: Mostrar descripción y precio de productos
R2: Registrar venta de productos
R3: Reducir stock cuando se realiza la venta
R4: Identificar al cajero utilizando un usuario y una clave

2.2.2 Facilidad de uso

Deben incluir subcategorías tales como:

- Factores humanos
- Estéticos
- Consistencia de interfaz de usuario
- Ayuda en línea o “context-sensitive”
- Asistentes (“wizards”)
- Documentación de usuario
- Materiales de capacitación/entrenamiento

Por ejemplo:

R1: El sistema deberá proporcionar ayudas en línea para orientar en el uso de las interfaces.
R2: Maximizar eficiencia mediante la navegación con teclado.
R3: El sistema debe tener interfaces gráficas de administración y de operación en idioma español y en ambiente 100% Web, para permitir su utilización a través de navegadores de Internet

2.2.3 Confiabilidad

- Frecuencia de fallos
- Capacidad de recuperación a fallos
- Posibilidades de predicción del programa
- Precisión
- Tiempo medio de fallos

Por ejemplo:

R1: El sistema debe registrar los pagos a crédito autorizados que se hagan a las cuentas por cobrar en un plazo de 24 horas, aun cuando se produzcan fallas de energía o del equipo.

R2: La cuenta del usuario se bloqueará por un lapso de 30 minutos luego de 4 intentos fallidos para evitar vulnerabilidades en la seguridad del sistema.

2.2.4 Rendimiento

Condiciones impuestas a requisitos funcionales, tales como:

- Velocidad
- Eficiencia
- Disponibilidad
- Tiempo de respuesta
- Tiempo de recuperación
- Uso de recursos

Por ejemplo:

R1: El tiempo máximo para mostrar el reporte de cuentas por cobrar mediante un histograma es de 20 segundos

R2: El sistema debe estar disponible al 100% o muy cercano a esta disponibilidad durante el horario hábil laboral de la empresa a nivel nacional, es decir, de lunes a viernes de 8:00 a.m. a 5:00 p.m., con excepción de los días festivos.

2.2.5 Soporte

Es la capacidad que tiene el software de ser modificado fácilmente para adecuar mejoras o cambios. Incluye:

- Adaptabilidad
- Facilidad de mantenimiento
- Compatibilidad
- Configurabilidad
- Facilidad de instalación
- Internacionalización

Por ejemplo:

R1: El sistema debe operar de manera independiente del navegador que se utilice (Microsoft Internet Explorer 6.0 o superior, Netscape 6.0 o superior, Mozilla FireFox).

R2: El sistema deberá estar orientado a que las actualizaciones sólo se hagan en el sitio del servidor.

2.2.6 Restricciones de diseño

Especifican o restringen el diseño de un sistema. Por ejemplo:

R1: El sistema deberá considerar, en su arquitectura, un modelo tres capas, donde se definen tres componentes lógicos de manera independiente: servicios de presentación o interfaz de usuario, servicios de funcionalidad y servicios de datos.

2.2.7 Requisitos de implementación

Especifica restricciones de codificación o de construcción del sistema:

- Estándares requeridos
- Lenguajes de implementación
- Políticas para la integridad de bases de datos
- Límite de recursos
- Ambientes de operación

Por ejemplo:

R1: El sistema debe desarrollarse con el lenguaje JAVA V1.6.

2.2.8 Requisitos de interfaz

Especifica:

- Elemento externo con el que el sistema debe interactuar
- Restricciones o formatos, tiempos u otros factores usados en tales interacciones

Por ejemplo:

R1: El sistema deberá proporcionar, para los diferentes reportes solicitados, salidas en documentos electrónicos (Word, Excel o Acrobat Reader).

R2: En una visión preliminar de impresión se consideraría que todos los textos estarán relacionados con un visor de PDF's, las estadísticas y resultados de consultas estarán relacionados con Excel 2003.

2.2.9 Requisitos físicos

Especifican características físicas que el sistema debe poseer; por ejemplo, material, forma, tamaño y peso. Pueden especificarse los requisitos de hardware.

Por ejemplo:

R1: Para que un cliente de la aplicación pueda ejecutar procesos, en línea, considerados en el sistema el punto de acceso deberá cumplir con los siguientes requisitos mínimos.

- Procesador 1.0 GHz.
- Memoria 128 MB.
- Disco duro 10 GB.
- Sistema Operativo Windows XP o Linux.
- Navegador internet Explorer 6.0 o posterior, Mozilla Firefox 2.X, Netscape Navigator 6.X o posterior con plug ins para Macromedia Flash y Java.
- Conexión a Internet, mínimo 56Kbps

2.3 Técnicas para capturar requisitos

Existen varias técnicas para capturar requisitos de usuarios, de las cuales examinaremos algunas.

2.3.1 Entrevistas

Utilizada para reunir información proveniente de una persona o de un grupo de personas. Generalmente, los encuestados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto. En algunos casos, son gerentes o empleados que proporcionan datos para el sistema propuesto o que serán afectados por él.

El éxito de esta técnica, depende de la habilidad del entrevistador para crear un clima de confianza y de su preparación para la misma. Después de la entrevista, se debe analizar la información obtenida y construir algunos requisitos candidatos.

Los puntos esenciales de esta técnica se anotan a continuación:

- Dos entrevistadores: Conviene que dos personas realicen la entrevista para mejorar la gestión del tiempo, pues uno conduce la entrevista y el otro supervisa la interacción y toma notas.
- Formule tanto preguntas abiertas como cerradas. Las preguntas abiertas no presuponen ninguna respuesta particular y animan al entrevistado a hablar sobre el problema, mientras que las preguntas cerradas presentan un intervalo específico de respuesta. Ejemplos:
  - Pregunta abierta: "¿Quién utiliza el sistema?"
  - Pregunta cerrada: "¿Utiliza usted el sistema?"
- No invente una solución, pues puede pensar que tiene una muy buena idea de lo que necesitan los grupos de decisión, pero debe mantener esta preconcepción a un lado durante la entrevista. Ésta es la única forma de averiguar lo que realmente necesita.
- Escuche. Ésta es la única forma en la que averiguará qué quieren los grupos de decisión, por lo tanto déjeles tiempo para hablar. Permítales hablar sobre una pregunta y que la exploren de su propia forma. Si busca respuestas específicas, es posible que invente una solución y formule preguntas cerradas basándose en esa invención.
- No adivine los pensamientos. Ésta es una habilidad humana muy importante, ya que es la base de la empatía. Sin embargo, no es recomendable cuando trata de obtener requisitos, pues puede acabar especificando lo que usted necesita en lugar de lo que necesitan los grupos de decisión.

2.3.2 Cuestionarios

Los cuestionarios pueden ser un suplemento de utilidad para las entrevistas. Son excelentes para conseguir respuestas a preguntas cerradas. Puede descubrir preguntas claves a partir de las entrevistas e incorporar éstas en un cuestionario que puede distribuir a una audiencia más amplia. Esto le puede ayudar a validar su entendimiento de los requisitos.

Por el tipo de preguntas que contiene, existen dos tipos de cuestionarios: abiertos y cerrados.

- Abiertos: No presuponen ninguna respuesta particular y animan al entrevistado a hablar sobre el problema para obtener opiniones y/o referencias.
- Cerrados: Limitan las respuestas posibles a través de un estilo cuidadoso en la pregunta.

Los tipos de respuestas de un cuestionario cerrado podrían ser los siguientes:

- **SI/NO**

¿Cree que se cometen muchos errores en la codificación de los números de cuenta en las facturas?

SI ☐

NO ☐

- **DE ACUERDO/EN DESACUERDO**

¿Se cometen muchos errores al codificar os números de cuenta en las facturas?

DE ACUERDO ☐

EN DESACUERDO ☐

- **ESCALAS**

¿Se cometen muchos errores al codificar los números de cuenta en las facturas?

TOTALMENTE DE ACUERDO ☐

DE ACUERDO ☐

NO ESTOY SEGURO ☐

EN DESACUERDO ☐

TOTALMENTE EN DESACUERDO ☐

- **NÚMERO**

De cada 100 facturas que se procesan, ¿cuántas tienen errores? Anote el número: _____

- **RANGO**

De cada 100 facturas que se procesan, ¿cuántas tienen errores?	0 - 5	
	6 - 10	
	11 - 15	
	16 - 20	
	21 - 25	
	26 - 30	
	31 - 40	
	41 - 50	
	Más de 50	

- **SELECCIÓN DE RESPUESTAS LIMITADAS**

Cuando se presentan errores en la codificación de los números de cuenta en las facturas, ¿cual es la causa mas frecuente? (Anote el número de la respuesta apropiada. También, la segunda razón mas común y la menos común).

1....
2....
3....

Los buenos cuestionarios se deben diseñar previamente. Un pensamiento cuidadoso, acompañado de una prueba previa, tanto del formato como de las preguntas, son la base de una recopilación de datos significativa a través de cuestionarios.

Pautas que le ayudarán a confeccionar un buen cuestionario:

1. Determine qué datos necesitan recopilarse y qué personas son las más calificadas para proporcionarlos. Si otros grupos pueden proporcionar datos variantes y mayor visión, identifíquelos también.

2. Seleccione el tipo de cuestionario (abierto o cerrado). Reconozca cuáles pueden ser más útiles, si contienen una sección de respuestas cerradas y otras de respuestas abiertas.

3. Desarrolle un Grupo de preguntas para incluirlas en el cuestionario. Las preguntas extras que son intencionalmente redundantes, pueden ser útiles al asegurar respuestas consistentes por parte de quien responda.

4. Examine el cuestionario para encontrarle fallas y defectos, como:
   a. Interrogantes innecesarias
   b. Preguntas que puedan ser mal interpretadas debido a su enfoque o forma de escritura
   c. Preguntas que el sujeto no pueda responder
   d. Preguntas que están escritas de forma que se escogerá la respuesta preferida
   e. Preguntas que se interpretaran en forma diferente dependiendo del marco de referencia de cada entrevistado
   f. Preguntas que no proporcionan opciones adecuadas de respuesta
   g. Un ordenamiento no adecuado de las preguntas y respuestas

5. Pruébelo previamente en un grupo pequeño para detectar otros problemas posibles.

6. Analice la respuesta del grupo de prueba para asegurar que el análisis de los datos que se busca se puede llevar a cabo con los datos recopilados. Si los datos no revelan algo que el analista no conoce, el cuestionario puede no ser necesario.

7. Realice cambios finales de edición e imprímalo en una forma legible.

8. Distribuya el cuestionario. Cuando sea posible, anote el nombre de cada persona.

2.3.3 Lluvia de ideas (Brainstorm)

Este es un modelo que se usa para **generar ideas**. La intención en su aplicación es la de generar la máxima cantidad posible de requisitos para el sistema. No hay que detenerse en pensar si la idea es o no del todo utilizable. La intención de este ejercicio es generar, en una primera instancia, muchas ideas.

Las reglas básicas a seguir son:

- Los participantes deben pertenecer a distintas disciplinas y, preferentemente, deben tener mucha experiencia. Esto trae como consecuencia la obtención de una cantidad mayor de ideas creativas.
- Conviene suspender el juicio crítico y se debe permitir la evolución de cada una de las ideas, porque sino se crea un ambiente hostil que no alienta la generación de ideas.
- Por más locas o salvajes que parezcan algunas ideas, no se las debe descartar, porque, luego de maduradas, probablemente se tornen en un requerimiento sumamente útil.
- A veces, ocurre que una idea resulta en otra idea y, otras veces, podemos relacionar varias ideas para generar una nueva.
- Escriba las ideas sin censura.

2.3.4 Prototipos

Durante la actividad de extracción de requisitos, puede ocurrir que algunos requisitos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requisitos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final.

Entonces, para validar los requisitos hallados, se construyen prototipos. Estos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiéndonos conseguir una importante retroalimentación en cuanto a si el sistema diseñado sobre la base de los requisitos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva.

El desarrollo del prototipo comienza con la captura de requisitos. Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requisitos que son conocidos, y señalan áreas en las que será necesaria profundizar las definiciones. Luego de esto, tiene lugar un “diseño rápido”, el cual se centra en una representación de aquellos aspectos del software que serán visibles al usuario (entradas y formatos de las salidas), llevando a la construcción de un prototipo.

2.4 Herramientas automatizadas para administrar requisitos

En el desarrollo de software se cuenta con una ventaja proporcionada por las herramientas CASE (Ingeniería del Software Asistida por Computadora). Existe un grupo para el campo de la ingeniería de requisitos, las cuales se concentran en capturar requisitos, administrarlos y producir una especificación de requisitos. Las herramientas más utilizadas para este propósito son *Rational RequisitePro* de IBM y *DOORS* de Telelogic (compañía de IBM).

2.4.1 RequisitePro

La herramienta de Rational RequisitePro es una solución poderosa y muy fácil de usar, que promueve una mejor comunicación, mejora la colaboración de los equipos y reduce el riesgo de los proyectos.

La solución de IBM Rational RequisitePro promueve la herramienta ampliamente usada y conocida de Microsoft® Word para facilitar la captura de requisitos. Aunque útiles para la captura de requisitos, los documentos no son un entorno óptimo para priorizar y organizar la información, actividades que se realizan mejor usando una base de datos. Vinculando documentos de requisitos a una base de datos, el producto RequisitePro une lo mejor de ambos mundos.

2.4.2 DOORS

Telelogic DOORS®, la familia de soluciones para la Gestión de Requisitos, mejora la calidad optimizando la comunicación y la colaboración, y promoviendo la conformidad con las normas y estándares y la validación mediante las capacidades siguientes:

- Interfaces intuitivas que promueven la adopción de la Gestión de Requisitos
- Escalabilidad para cualquier tamaño de proyecto con cualquier número de usuarios
- Una matriz de trazabilidad de requisitos fácil de usar, actualizada y flexible
- El soporte más completo para el registro, estructuración, gestión y análisis de requisitos y su trazabilidad
- Integración sin precedentes con otras soluciones de Telelogic y herramientas de terceros para tener una mejor visión de los requisitos y controlar su trazabilidad a lo largo del ciclo de vida del proceso de desarrollo

3. CAPTURA DE REQUISITOS A SOLICITUD DEL CLIENTE

Si el equipo de desarrollo tiene un conocimiento de la estructura de la organización, de las metas, de la visión y de los clientes/usuarios o si sólo se está añadiendo una nueva característica a un sistema existente, entonces, RUP no recomienda que se empiece con un modelado del negocio. En ese caso, RUP recomienda que se empiece con la captura de requisitos.

Esta actividad consiste en identificar todas las necesidades de stakeholders. Como se explicó anteriormente, el término stakeholder se utiliza para referirse a cualquier persona o grupo que está interesado por los resultados del proyecto.

Obtener y comprender los requisitos de los stakeholders es difícil por varias razones:

- Los stakeholders, a menudo, no conocen lo que desean obtener del sistema informático, excepto en términos muy generales; puede resultarles difícil expresar lo que quieren que haga el sistema o pueden hacer demandas irreales debido a que no conocen el costo de sus peticiones.
- Los stakeholders expresan los requisitos distintos con sus propios términos de forma natural y con un conocimiento implícito de su trabajo.

- Diferentes requisitos de diferentes stakeholders tienen concordancia y algunos generan conflictos.

En la figura 3.3, se muestra un modelo general para obtener y analizar requisitos. Con cada vuelta del ciclo de este modelo, la comprensión de los requisitos por parte del analista mejorará.

Cada equipo de desarrollo tendrá su propia versión o instancia de este modelo dependiendo de la habilidad del personal, el tipo de sistema a desarrollar y los estándares utilizados.

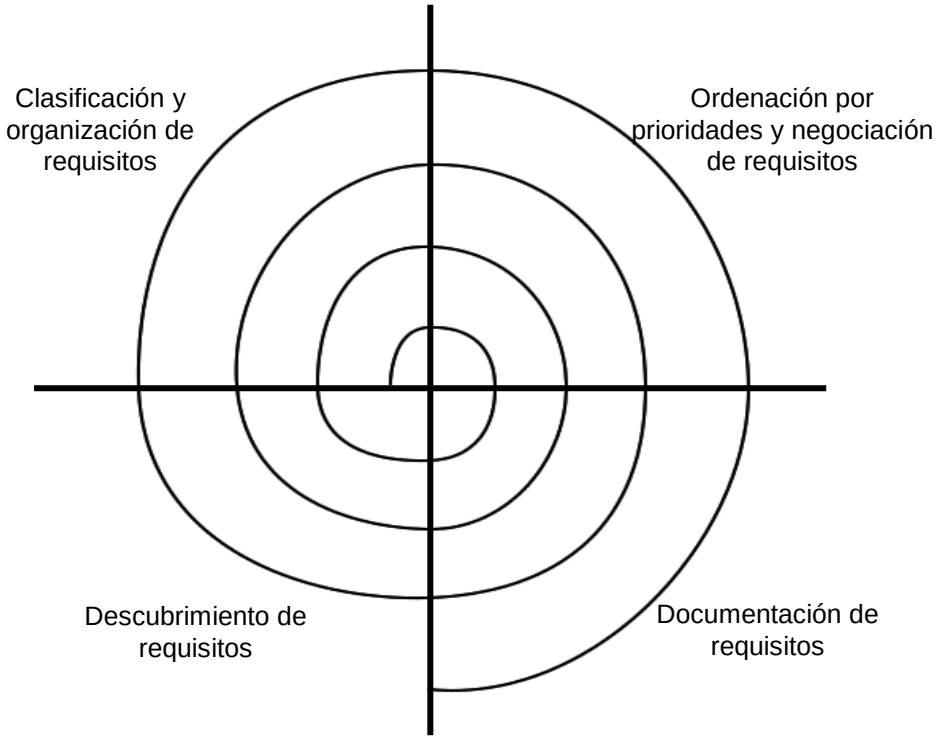


Figura 3.3. El proceso de obtención y análisis de requisitos.

Los pasos a seguir son:

1. Descubrimiento de requisitos. Es el proceso de interactuar con los stakeholders del sistema para recopilar sus requisitos. Para ello, se utilizan técnicas de captura de requisitos, como las entrevistas y prototipos.
2. Clasificación y organización de requisitos. En esta actividad, el analista toma la recopilación no estructurada de requisitos, grupos relacionados de requisitos y los organiza en grupos coherentes.
3. Ordenación por prioridades y negociación de requisitos. Inevitablemente, cuando existen muchos stakeholders involucrados, los requisitos entrarán en conflicto. Esta actividad se refiere a ordenar según las prioridades de requisitos, y a encontrar y resolver los requisitos en conflicto a través de la negociación.
4. Documentación de requisitos. Se documentan los requisitos y se continúa en la siguiente vuelta de la espiral. Se pueden producir documentos de requisitos formales o informales, como, por ejemplo, modelo de casos de uso para requisitos funcionales y especificación de requisitos de software que contiene todos los tipos de requisitos (funcionales y no funcionales) del sistema.

4. CAPTURA DE REQUISITOS A PARTIR DEL DIAGRAMA DE ACTIVIDADES DEL NEGOCIO

Mediante la utilización del modelo del negocio como entrada, el analista emplea una técnica sistemática para crear un modelo de casos de uso tentativo. Para ello, construirá un diagrama de casos de uso tomando como punto de partida los diagramas de actividades de los casos de uso del negocio.

En primer lugar, se obtienen los requisitos funcionales a partir de las actividades candidatas a ser informatizadas. Luego, con estos requisitos, se crean los casos de uso. Las actividades que no serán soportadas por el sistema no les corresponderán un caso de uso. Los actores se identificarán a partir de los roles (trabajadores o actores del negocio) que realizan las actividades del negocio a informatizar.

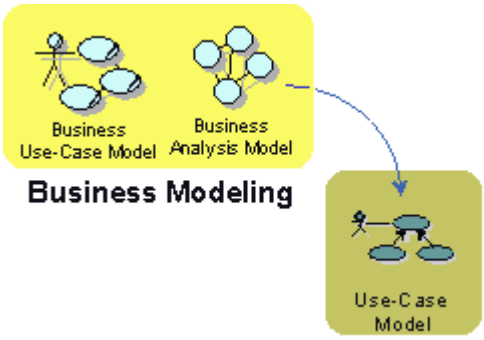


Figura 3.4 Del modelo del negocio al modelo de casos de uso

Es importante documentar el seguimiento de estos elementos: actividades a informatizar, requisitos funcionales y casos de uso; más aún, si se trata de seguir requisitos funcionales de casos de uso, el cual es un proceso complicado por el hecho de que existe una relación muchos a muchos entre ellos. **Un caso de uso puede tratar muchos requisitos funcionales y un requerimiento funcional puede estar presente en varios casos de uso diferentes**.

Afortunadamente, existen herramientas de ingeniería de requisitos, como el RequisitePro y DOORS. Pero si no tiene ningún soporte de herramienta de modelado, tiene que hacer el trabajo manualmente. Un buen enfoque es crear una matriz denominada Matriz de actividades Vs. requisitos. Estas matrices se crean a menudo en hojas de cálculo (Excel). La plantilla se proporciona en la Tabla 3.2.

Matriz de Actividades Vs. REquisitos del Sistema <Nombre del Sistema>						
Proceso de Negocio	**Actividad del Negocio**	**Responsable del Negocio**	**Requisito**		**Caso de Uso**	**Actores**
Proceso 1			R01		CUS01	
			R02		CUS02	
Proceso 2			R03		CUS03	
			R04		CUS04	
			R05		CUS05	
			R06		CUS06	

Tabla 3.2. Plantilla de matriz de actividades y requisitos.

Una matriz de actividades Vs. requisitos es una herramienta manual utilizada para obtener requisitos funcionales a partir de actividades del negocio que se van a informatizar. Una vez identificado los requisitos funcionales, se crean los casos de uso. Por otro lado, los actores son creados a partir de los responsables de las actividades del negocio que se tienen en la matriz.

Examinemos un ejemplo para capturar requisitos a partir de un diagrama de actividades del negocio para un sistema de créditos XYZ. En la figura 3.5, se muestra el diagrama de actividades del caso de uso de negocio "otorgamiento de Créditos"; las actividades sombreadas se identificaron para ser informatizadas.

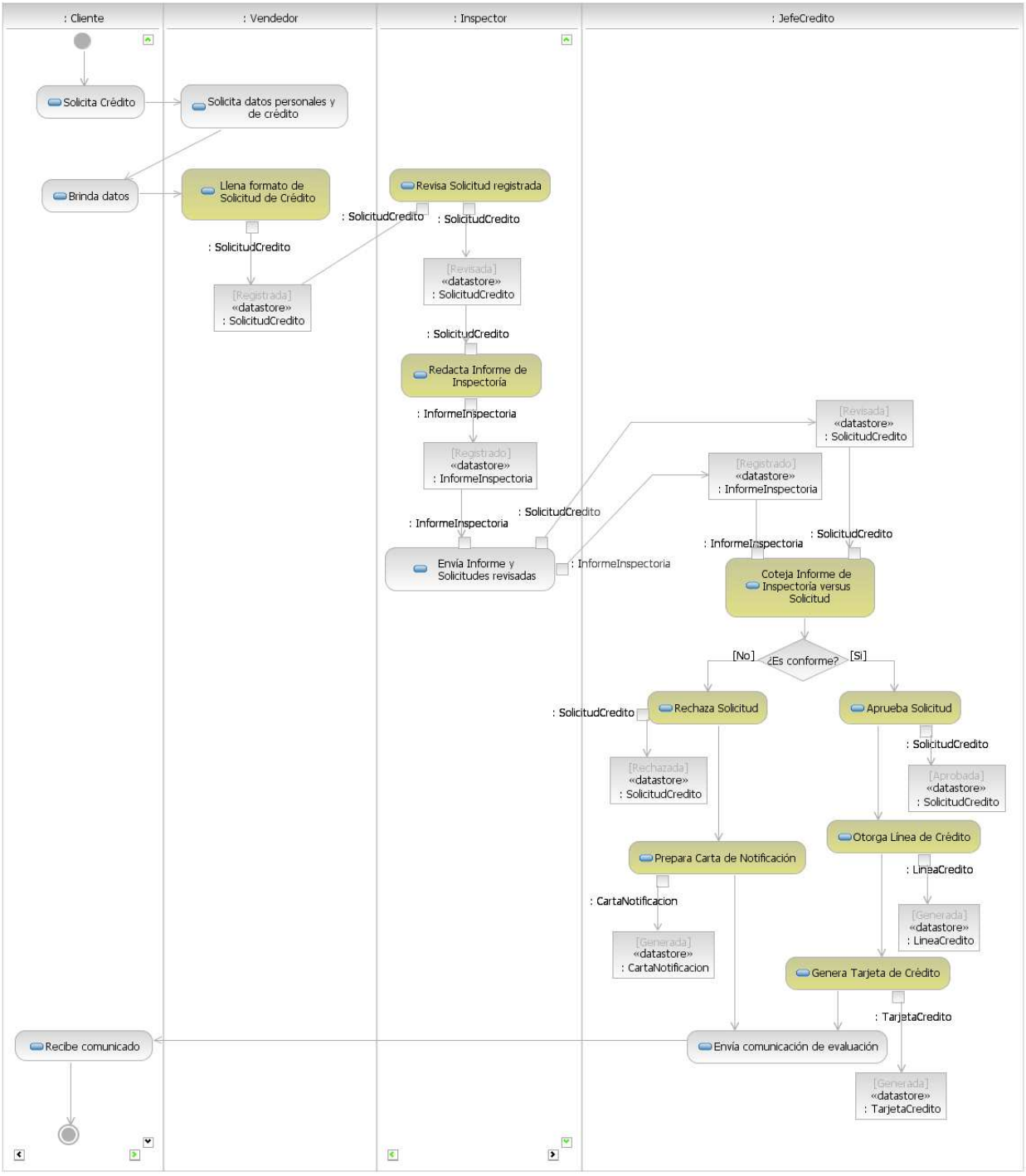


Figura 3.5. Diagrama de actividades del negocio del sistema de créditos XYZ.

La matriz de actividades Vs. requisitos para el sistema de créditos XYZ es el siguiente:

Matriz de Actividades Vs. Requisitos Funcionales del Sistema de Créditos XYZ

Proceso de Negocio	Actividad del Negocio (Las que se automatizan)	Responsable del Negocio	Requisito (Capacidad o Condición)		Caso de uso (Funcionalidad)	Actores (Roles de Usuario)
Otorgamiento de Créditos	Llena formato de Solicitud de Crédito	Vendedor	R01	Registrar Solicitud de Crédito	Registrar Solicitud de Crédito	Vendedor
	Revisa Solicitud registrada	Inspector	R02	Consultar Solicitudes Registradas	Consultar Solicitudes por Estado	Visualizador de Solicitudes
	Redacta Informe de Inspectoría		R03	Registrar Inspectoría	Registrar Inspectoría	Inspector
	Coteja Informe de Inspectoría versus Solicitud	Jefe de Créditos	R04	Consultar Solicitudes Verificadas	Consultar Solicitudes por Estado	Visualizador de Solicitudes
	Rechaza Solicitud		R05	Rechazar Solicitudes	Evaluar Crédito	Jefe de Créditos
	Prepara Carta de Notificación		R06	Generar Cartas de Rechazo		
	Aprueba Solicitud		R07	Aprobar Solicitudes		
	Otorga Línea de Crédito		R08	Generar Línea de Crédito		
	Genera Tarjeta de Crédito		R09	Generar Tarjeta Física de Crédito	Generar Tarjeta de Crédito	

Tabla 3.3. Matriz de actividades y requisitos del sistema de créditos XYZ.

Observaciones:

- Note que, para el requerimiento R02 y R04, se crea el mismo caso de uso, esto es, debido a que ambos requisitos dan lugar al criterio de búsqueda del **CU Consultar Solicitudes por Estado**.
- Por otro lado, fue necesario crear el actor **Visualizador de Solicitudes** para el Inspector y Jefe de Créditos, pues dos actores no pueden iniciar el mismo caso de uso.
- Por último, los requisitos R05, R06, R07 y R08 dieron lugar al caso de uso Evaluar Crédito porque serán definidos como **subflujos en la Especificación del CU Evaluar Crédito.**

Luego, el diagrama de casos de uso inicial resultante será el siguiente:

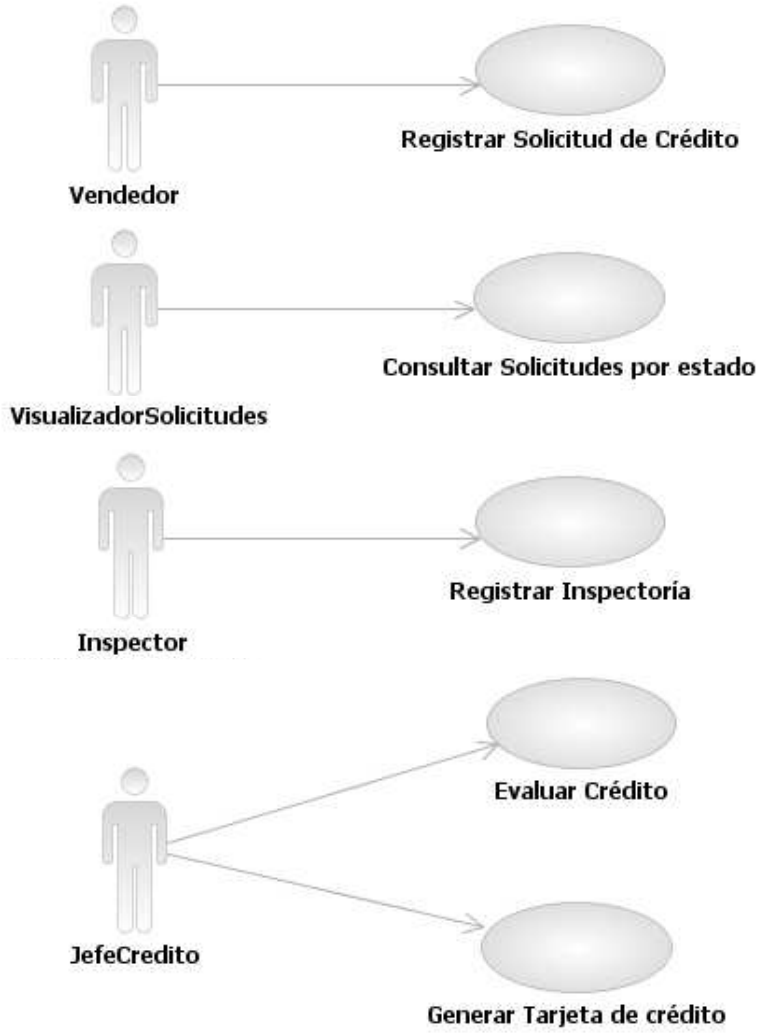


Figura 3.6. Diagrama de casos de uso del sistema de créditos XYZ.

Este diagrama de casos de uso debe ser refinado. Nuevos casos de uso serán detectados al describir cada caso de uso obtenido; para ello, se utiliza el documento *Especificación de caso de uso*.

Por último, los nuevos casos de uso serán agregados en un diagrama denominado Diagrama General de Casos de Uso, consiguiendo de esta manera, la versión final del *modelo de casos de uso*.

ACTIVIDADES PROPUESTAS

1. De la lista, clasifique los requisitos según las categorías de FURPS+.

R01. El sistema deberá garantizar que su despliegue se pueda realizar, ya sea en el lado del servidor o del cliente, sobre plataforma hardware de 32 bits o de 64 bits sin que esto afecte el rendimiento del mismo.

R02. El sistema debe contar con un Manual Técnico de Referencia para la Aplicación, el cual estará orientado a profesionales capacitados en aspectos técnicos del área de sistemas.

R03. La clave de los usuarios considerará las siguientes políticas:
- Expirar según parametrización del sistema
- Tener mínimo una longitud de 8 caracteres
- Contener letras y números.
- No puede contener espacios.
- No pueden repetirse las 3 últimas contraseñas.
- No contendrá el nombre o apellido de la persona dueña del usuario.

R04. El código fuente del sistema deberá cumplir con el estándar de codificación definido en el documento “Estándares y Nomenclaturas de Código Fuente”.

R05. Utilizar el idioma castellano para los mensajes y textos en la Interfaz.

R06. El sistema será utilizado por clientes de todo el mundo. Adicionalmente, la Organización Pro-Turismo exige que, para anunciar servicios en su portal, éstos deben estar provistos en español, inglés y portugués. Estos tres idiomas deben ser soportados por el producto desarrollado.

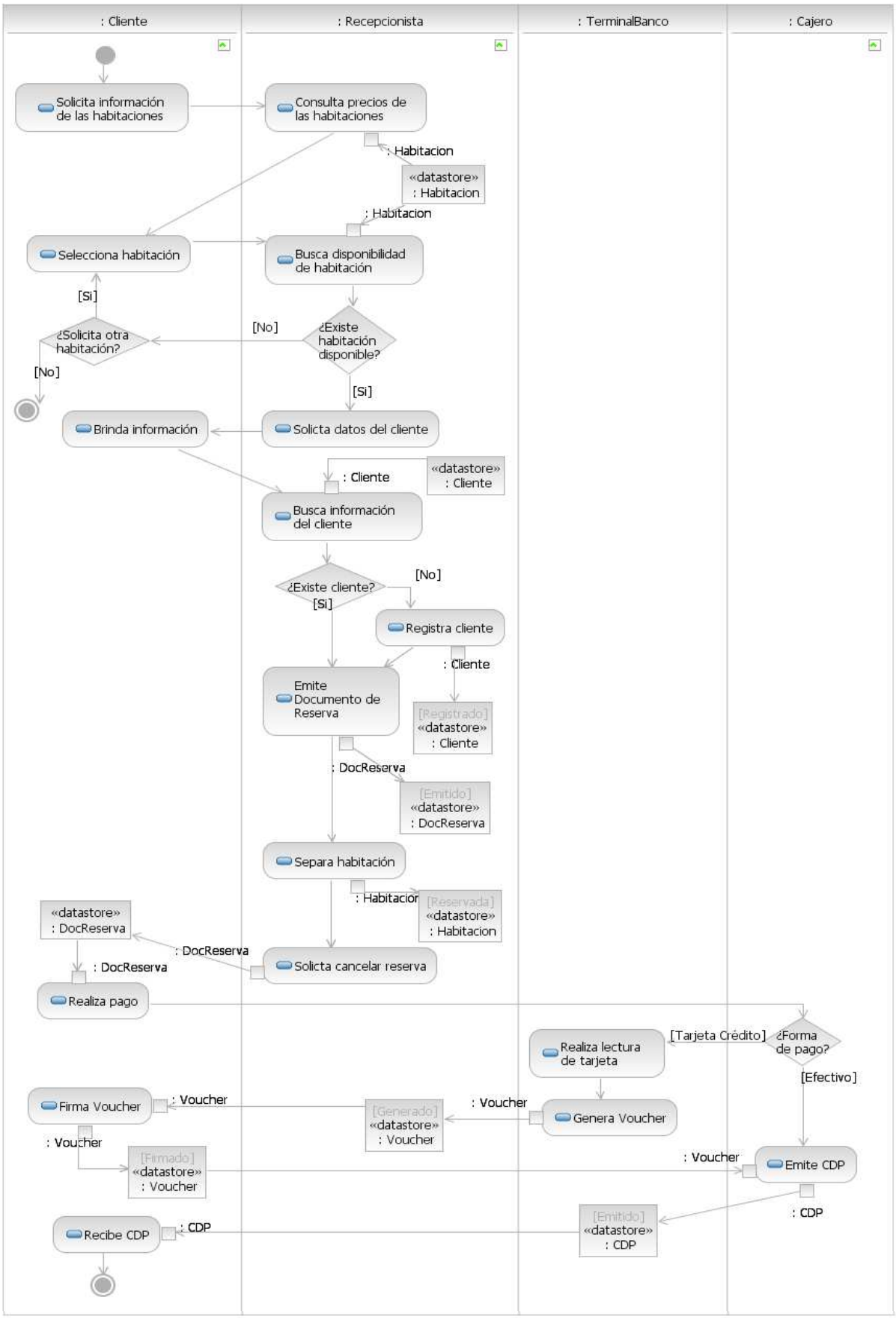
R07. El sistema deberá permitir la creación, modificación, activación, desactivación y autorización de los roles de usuarios definidos.

R08. El sistema deberá prever contingencias que pueden afectar la prestación estable y permanente del servicio.
La siguiente es la lista de las contingencias que se deben tener en cuenta y se pueden considerar críticas:
- Sobrecarga del sistema por volumen de usuarios
- Caída del sistema por sobrecarga de procesos
- Caída del sistema por sobrecarga de transacciones
- Caída del sistema por volumen de datos excedido en la base de datos

R09. El sistema deberá contar con el manual de FAQ (Frequent Asked Questions), en línea e impreso, que es un resumen con las respuestas a las preguntas más frecuentes que se hacen los usuarios.

R10. El sistema debe considerar, en su arquitectura, el patrón de diseño MVC.

2. A partir del diagrama de actividades del negocio del proceso “Reserva de Habitaciones” para un Sistema de Reservaciones de un Hotel realice la matriz de actividades Vs. requisitos y el diagrama de casos de uso (Inicial).



Resumen

- En la captura de requisitos se describe las condiciones o capacidades que el sistema debe cumplir.
- La identificación de los requisitos funcionales llevará a la proyección de las funciones del sistema (casos de uso).
- La descripción de los requisitos no funcionales facilitarán la construcción de la plataforma del sistema.
- Los stakeholders son un grupo de interés cuyas necesidades deben ser satisfechas por el proyecto, por ejemplo, usuarios finales del sistema, gerentes, accionistas, reguladores quiénes certifican la aceptabilidad del sistema, etc.
- El Modelo de obtención y análisis de requisitos se utiliza para capturar requisitos a partir de los stakeholders.
- La matriz de actividades Vs. requisitos es una técnica utilizada para documentar la trazabilidad de actividades Vs. requisitos funcionales y de requisitos funcionales Vs. casos de uso.
- Si desea saber más acerca de estos temas, puede consultar los siguientes libros.

  - "Ingeniería del Software" de Ian Sommerville

    En el capítulo 6, encontrará información sobre requisitos del sistema; y en el capítulo 7, se profundiza el tema de la ingeniería de requisitos.

  - "UML 2" de Jim Arlow e Ila Neustadt

    En el capítulo 3, encontrará información sobre la captura de requisitos.

5. MODELO DE CASOS DE USO

El modelo de casos de uso captura los requisitos funcionales de los usuarios a un alto nivel y establece la estructura fundamental del sistema. Este modelo utiliza actores y casos de uso. Los actores son los roles que los usuarios pueden representar y los casos de uso representan lo que los usuarios deberían poder hacer con el sistema.

Este modelo no incluye los requisitos no funcionales ni los requisitos funcionales internos. No usa descomposición funcional. El modelo resultante es fácil de entender y debe ser verificado por los usuarios antes de empezar a construir el sistema.

El modelo de casos de uso es empleado no solamente para capturar requisitos de nuevos sistemas; también, es utilizado cuando nuevas generaciones de sistemas son desarrolladas. Cuando una nueva versión de un sistema es desarrollada, las nuevas funcionalidades son agregadas al modelo de casos de uso existente, insertando nuevos actores y casos de uso y modificando las especificaciones de los actuales casos de uso.

El modelo de casos de uso se construye mediante los siguientes pasos:

- Encontrar actores y casos de uso
- Relacionar actores y casos de uso (Diagrama de caso de uso inicial)
- Priorizar casos de uso
- Detallar un caso de uso y especificar prototipo de interfaz de usuario en la ECU
- Estructurar el modelo de caso de uso (Diagrama de caso de uso final)

Estos pasos no tienen por qué ser ejecutados en ningún orden en particular y a menudo se hacen simultáneamente. De hecho, podemos trabajar por múltiples vías que producen un resultado final equivalente. Podemos, por ejemplo, comenzar encontrando algunos casos de uso (la actividad de encontrar actores y casos de uso), después diseñar las interfaces de usuario (la actividad de crear prototipo de interfaz de usuario), para darnos cuenta de que necesitamos añadir un nuevo caso de uso (así que retrocederemos a la actividad de encontrar actores y casos de uso, rompiendo la secuencia estricta marcada), y así sucesivamente.

Los resultados de la primera iteración a través de este flujo de trabajo consisten en una primera versión del modelo de casos de uso. Los resultados de cualquier iteración subsiguiente consistirán, entonces, en nuevas versiones de artefactos involucrados en la creación del modelo de casos de uso. Hay que recordar que todos los artefactos se contemplan y mejoran incrementalmente a través de iteraciones del proyecto.

Las distintas actividades en el modelado de caso de uso adoptan formas diferentes en diferentes fases del proyecto. Por ejemplo, cuando un analista ejecuta la actividad de encontrar actores y casos de uso durante la fase de inicio, identificará muchos actores y casos de uso nuevos. Pero cuando la actividad se realiza durante la fase de construcción, el analista hará cambios secundarios en la lista de actores y casos de uso, tales como la creación de un diagrama de casos de uso que describe mejor el modelo de casos de uso desde una perspectiva en particular.

5.1. Encontrar actores

Éste es uno de los primeros pasos para definir qué o quiénes interactuarán con el sistema. Antes de indicar cómo se identifican los actores empezaremos por definirlo.

5.1.1. Actor

Un actor representa un rol (humano, hardware o software) externo al sistema con el que se establece intercambio directo de información.

Hay una diferencia entre actor y usuario. Usuario es el que utiliza el sistema, mientras que el actor representa un cierto rol que un usuario puede desempeñar. Es decir que los actores definen los roles que pueden representar los usuarios.

A continuación, daremos algunos ejemplos para tener claro lo que constituye un actor.
Muchos usuarios pueden desempeñar el mismo rol. Por ejemplo, la figura 3.7 muestra a dos usuarios, Ivar y Mark, que cumplen el rol de operador de una máquina de reciclaje. Cuando ellos usan la máquina de reciclaje cada uno es representado por una instancia del actor operador.

Sin embargo, en algunas situaciones, una sola persona desempeña el rol representado por el actor. Por ejemplo, puede haber sólo una persona desempeñando el papel de administrador del sistema para un sistema bastante pequeño.

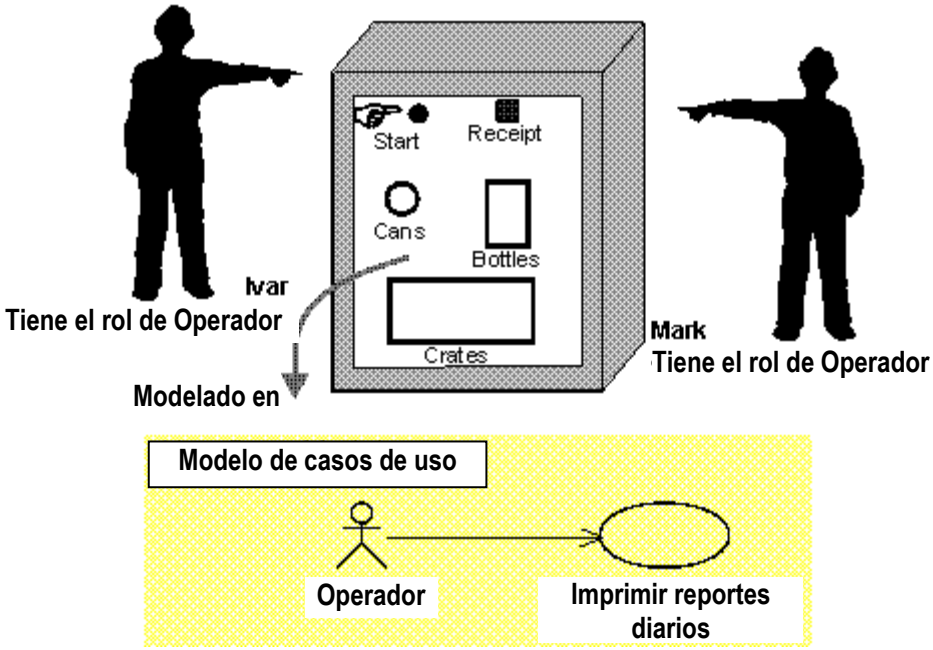


Figura 3.7. Muchos usuarios representados por un actor.

También, podemos encontrar que el mismo usuario puede ser representado por varios actores, esto es porque la misma persona puede desempeñar roles diferentes.

5.1.2. Cómo identificar actores

Para identificar actores debe responder las siguientes preguntas:

- ¿Qué actores del negocio o trabajadores del negocio son responsables de las actividades a informatizar?
- ¿Qué grupos de usuarios necesitan ayuda del sistema para llevar a cabo sus tareas?
- ¿Qué grupos de usuarios son fundamentales para ejecutar las funciones principales del sistema?
- ¿Qué grupos de usuarios son los que llevarán a cabo funciones secundarias como mantenimiento o administración?
- ¿El sistema a desarrollar interactuará con algún hardware o sistema de software?

Cualquier individuo, grupo o fenómeno que cumpla con una o más de estas categorías es candidato para ser un actor. La figura 3.8 muestra el entorno del sistema del cual se encontrarán categorías de actores.

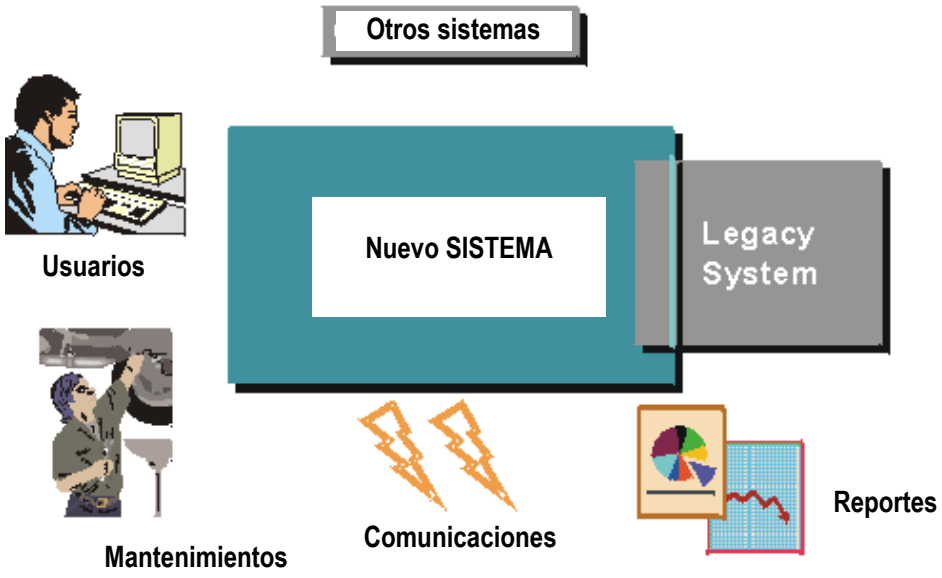


Figura 3.8. Entorno del sistema.

Para determinar si se tienen los actores correctos, se puede elegir dos o tres personas que usarán el sistema y, a continuación, ver si el conjunto de actores obtenido es suficiente para cubrir sus necesidades.

Por último, se completa la identificación de actores al obtener todos los casos de uso del sistema que conlleva a crear otros actores o eliminar a algunos que existen. Este proceso de refinamiento se verá en el próximo capítulo “Estructurar el modelo de casos de uso”.

5.1.3. Definición de las fronteras del sistema

Encontrar a los actores significa también definir las fronteras del sistema, que ayuda a comprender el propósito y el alcance del sistema. **Sólo aquellos que se comunican directamente con el sistema son actores**. Por ejemplo:

En un sistema de reservas de líneas aéreas, ¿quiénes podrían ser actores? Esto depende del tipo de sistema que se construye. Si está desarrollando el sistema de reservaciones para un agente de viajes, el actor será el agente de viaje. El pasajero no interactúa con el sistema directamente, entonces no será un actor.

Figura 3.9. Actores de un sistema de reservaciones I.

En cambio, si está desarrollando un sistema de reservaciones para que los pasajeros se puedan conectar a través de Internet, el pasajero, ahora sí, interactuará con el sistema y se convertirá en actor.

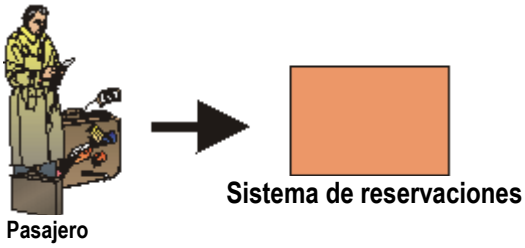


Figura 3.10. Actores de un sistema de reservaciones II.

5.1.4. El tiempo como actor

Aunque claramente el tiempo es una entidad externa al sistema (por lo que cumple con la primera mitad de la definición de un actor) difícilmente podremos pensar que el tiempo se encuentra interesado en una funcionalidad de nuestro sistema.

En este caso, es el sistema quien va a tener interés en el tiempo, debido a que deben efectuar operaciones automáticas en determinados momentos; y siendo esto un requisito funcional obvio, resulta de interés desarrollar alguna forma de capturar dicho requisito en el modelo de casos de uso. La técnica es introducir al actor "Tiempo", quien está asociado a casos de uso que capturan una funcionalidad automática. Un ejemplo de esto sería un respaldo automático del sistema que se ejecuta todas las noches o la generación automática de reporte de ventas diario, tal como se ilustra en la figura 3.11.

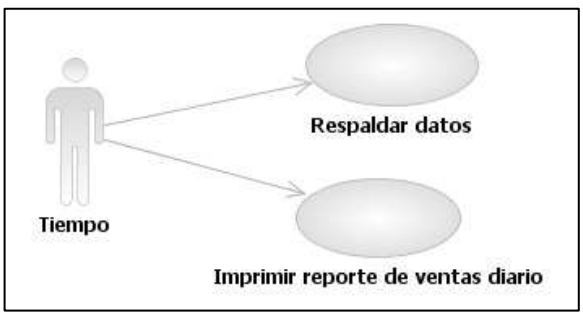


Figura 3.11. Tiempo como actor.

5.1.5. Sugerencias para identificar adecuadamente a los actores del sistema

- Son roles (humanos, software o hardware), no personas con nombres propios.
- No siempre está asociado con el nombre de un cargo en la planilla de la organización objetivo.
- El nombre no debe representar áreas, departamentos o partes de una organización sino roles de ejecución.
- Cada actor debe estar asociado con al menos un caso de uso del sistema.
- Si no participa en ningún proceso, debe ser eliminado del modelo.

5.1.6. Breve Descripción de actores

La breve descripción del actor debe incluir información sobre:

- ¿A qué o a quién representa el actor?
- ¿Por qué el actor es necesario?
- ¿Qué intereses tiene el actor en el sistema?

La breve descripción debe ser realizada en pocas líneas tanto en el modelo de casos de uso como en el documento actores del sistema.

5.2. Encontrar casos de uso

Cuando su primer esbozo de los actores se completa, el siguiente paso es identificar los casos de uso del sistema. Los primeros casos de uso son muy preliminares, y que sin duda tiene que cambiar un par de veces hasta que sean estables. Si el sistema de visión o de los requisitos son deficientes, o si el sistema de análisis es vaga, la funcionalidad del sistema será poco clara. Por lo tanto, usted debe preguntarse constantemente si ha encontrado los correctos casos de uso. Además, usted debe estar preparado para agregar, eliminar, combinar y dividir los casos de uso antes de llegar a una versión final. Usted recibirá una mejor comprensión de los casos de uso una vez que se han descrito en detalle.

5.2.1. Caso de uso

Un caso de uso es un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para los actores. De manera más precisa, un caso de uso especifica una secuencia de acciones que el sistema ejecuta interactuando con sus actores e incluyendo alternativas dentro de la secuencia. Las acciones pueden involucrar la comunicación con un determinado número de actores así como también realizar cálculos y trabajos dentro del sistema.

Las características de un caso de uso son:

- Siempre es iniciado por un actor. El actor, directa o indirectamente, ordena al sistema que se ejecute el caso de uso.
- Provee valor para un actor, es decir, un caso de uso debe entregar algún tipo de valor tangible para el actor.
- Es completo. Un caso de uso no está completo hasta que el valor final se produzca.

5.2.2. Cómo identificar casos de uso

La mejor manera de encontrar casos de uso es considerar lo que cada actor requiere del sistema. Recuerde que el sistema existe sólo para sus usuarios, y por lo tanto debe partir de las necesidades de los usuarios.

Si cuenta con un modelado de negocio, los diagramas de actividades del modelo de análisis de negocio serán utilizados para empezar a identificar casos de uso.

Las respuestas a las siguientes preguntas ayudarán a encontrar casos de uso:

- ¿Cuáles son las actividades del negocio objetos de automatización?
- ¿Cuáles son las tareas que el actor desea que el sistema desarrolle?
- ¿El actor crea, almacena, cambia, elimina o consulta datos en el sistema?
- ¿El actor necesita informar al sistema cambios generados en el entorno circundante al sistema?
- ¿El actor necesita ser informado sobre la ocurrencia de situaciones externas al sistema?

5.2.3. Sugerencias pasa identificar adecuadamente a los casos de uso

- Son procesos o funcionalidades del sistema que, en la mayoría de los casos, corresponden con opciones de ejecución.
- Deben estar asociados a, por lo menos, un actor del sistema u otro caso de uso del sistema.
- Representan la generalidad del comportamiento del proceso y no una instancia o escenario específico o caso muy particular del sistema.

5.2.4. Breve Descripción de casos de uso

La breve descripción del caso de uso debe reflejar su propósito, resumiendo las acciones que ejecuta el caso de uso al interactuar con los actores. Esta descripción se realiza, en primer lugar, con algunas frases en el modelo de casos de uso y, más adelante, con una descripción paso a paso de lo que el sistema necesita hacer cuando interactúa con sus actores, en la especificación de caso de uso.

5.3. Crear el diagrama de casos de uso

El diagrama de casos de uso muestra cómo se relacionan los casos de uso con los actores. Pueden diseñarse varios diagramas de casos de uso, cada uno, creado en un paquete que contiene un conjunto de casos de uso. **La organización por paquetes puede ser de acuerdo a cada proceso de negocio**.

Empiece dibujándolos en la esquina superior izquierda. Los casos de uso deben dibujarse en el orden en que normalmente los usará el actor. Opcionalmente, los casos de uso mostrados en el diagrama se pueden encerrar dentro de un rectángulo que representa los límites del sistema.

El primer diagrama de casos de uso es un esbozo de la comunicación que existe entre un actor y un caso de uso. Más adelante, se diseña una versión final agregando relaciones entre los casos de uso (*include*, *extend* o generalización).

La figura 3.12 muestra un esbozo del diagrama de casos de uso de una máquina de reciclado. El rectángulo contiene los casos de uso que constituyen el comportamiento del sistema.

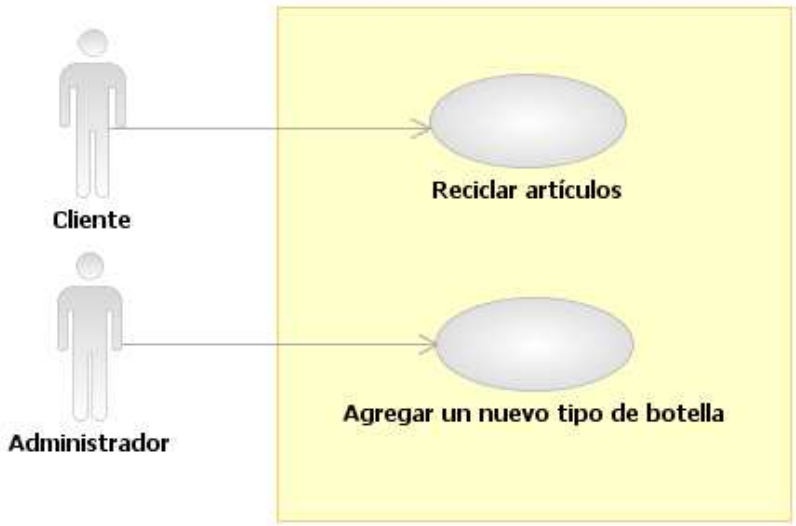


Figura 3.12. Diagrama de casos de uso para una máquina de reciclado (inicial).

6. CASOS DE ESTUDIO N°1

Identifique a los actores y casos de uso en los siguientes casos propuestos:

CASO: Perú TOURS

La agencia de viajes Perú TOURS requiere de un sistema web para que sus clientes no socios se afilien, llenando un formulario con sus datos personales. Una vez que el cliente se registre como socio, puede solicitar paquetes turísticos registrando en un formulario el destino, número de personas a viajar, fecha, hora y ciudad de partida, fecha y hora de regreso.

El agente receptivo es el responsable de elaborar las cotizaciones por paquete turístico solicitado para que, posteriormente, el socio lo consulte. Si el socio está de acuerdo con alguna de las cotizaciones presentadas en el sistema, la selecciona y registra su aprobación. De lo contrario, debe tener la opción de registrar alguna observación de la cotización que le interesa.

El sistema, también, debe permitir al socio realizar el pago de la cotización aceptada. Para ello, el sistema se conecta con el sistema bancario para realizar la transferencia.

Por otro lado, el gerente de la agencia y el agente receptivo requieren consultar cotizaciones canceladas, aceptadas u observadas por rango de fechas. Y uno de los procesos automáticos que debe realizar el sistema es que cada fin de mes se genere un backup de la base de datos.

Resumen

- El modelo de casos de uso captura los requisitos funcionales del sistema.
- Un actor representa un rol (humano, hardware o software) externo al sistema y que interactúa con ella. Se introduce al actor “Tiempo” para asociarlos a casos de uso que capturen funcionalidades automáticas.
- Es importante tener clara la diferencia entre usuario y actor. Un actor es una clase de rol, mientras que un usuario es una persona que, cuando usa el sistema, asume un rol. En este sentido, un usuario puede acceder al sistema como distintos actores.
- Un caso de uso es un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para los actores.
- Todo sistema de software ofrece a su entorno (aquellos que lo usan) una serie de servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando decimos “alguien o algo” hacemos referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y software.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  http://rup.hops-fp6.org/process/modguide/md_ucmod.htm
  Aquí encontrará información detallada sobre el modelo de casos de uso.

  http://rup.hops-fp6.org/process/modguide/md_uc.htm
  Aquí encontrará información detallada sobre casos de uso.

  http://rup.hops-fp6.org/process/modguide/md_actor.htm
  Aquí encontrará información detallada sobre actores.

7. ESTRUCTURAR EL MODELO DE CASOS DE USO

Esta actividad se centra en relacionar los casos de uso y los actores del sistema, e identificar sus comportamientos opcionales y excepcionales. Se establece las inclusiones, extensiones y generalizaciones entre casos de uso, y las generalizaciones entre actores.

Existen tres razones para realizar la estructuración del modelo de casos de uso:

- Hacer que los casos de uso sean fáciles de entender
- Extraer el comportamiento común encontrado en varios casos de uso
- Hacer que el modelo de casos de uso sea fácil de mantener

La ejecución de cada caso de uso incluye la comunicación con uno o más actores. Una instancia de un caso de uso siempre es iniciada por un actor pidiendo al sistema hacer algo. Esto implica que cada caso de uso debe tener la asociación de comunicación con los actores. La razón de esta norma es hacer cumplir el sistema para ofrecer sólo la funcionalidad que los usuarios necesitan, y nada más. Si se encuentran casos de uso que nadie pide significa que algo está mal en el modelo de caso de uso o en los requisitos.

Sin embargo, **hay algunas excepciones a esta regla**:

- Si un caso de uso es abstracto (no se instancia por separado, sino en el contexto de otro caso de uso), su comportamiento no puede incluir la interacción con algún actor. En ese caso, el caso de uso abstracto no tendrá ninguna asociación de comunicación con actores.
- Un caso de uso hijo en una relación de generalización no necesita tener un actor asociado a él si el caso de uso padre describe la comunicación con el actor.
- Un caso de uso base en una relación include no necesita tener un actor asociado a él si el caso de uso incluido describe la comunicación con el actor.
- Un caso de uso puede ser iniciado de acuerdo con un horario (por ejemplo, una vez a la semana o una vez al día), lo que significa que el reloj del sistema es el iniciador. El reloj del sistema es interno al sistema y el caso de uso no es iniciado por un actor, sino por un evento interno del sistema. Si ninguna interacción de actor se produce en el caso de uso, éste no tendrá ninguna asociación con un actor. Sin embargo, se puede utilizar un actor ficticio "Tiempo" para mostrar cómo el caso de uso es iniciado en el diagrama de casos de uso.

1.1 Objetivos

Los objetivos de esta actividad son:

- Extraer descripciones de funcionalidad (de casos de uso) generales y compartidas que pueden ser utilizadas por casos de uso más específicos (generalización)
- Extraer descripciones de funcionalidad (de casos de uso) adicionales u opcionales que pueden extender casos de uso más específicos (relaciones de extensión)
- Extraer descripciones de funcionalidad (de casos de uso) adicionales e incondicionales incluidas en la ejecución de casos de uso específicos (relaciones de inclusión)

1.3 Tipos de relaciones

Existen tres tipos de relaciones entre casos de uso: include, extend y de generalización. Entre actores se puede utilizar sólo la relación de generalización.

1.3.1 Relación *include*

Una relación *include* se define como la utilización de los pasos de un caso de uso como parte de la secuencia de otro caso de uso al que se llamará caso de uso base. El caso de uso incluido nunca se encuentra aislado, **su uso es obligatorio,** es instanciado sólo como parte de algún caso de uso base que lo incluye.

Esta relación se usa para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte y que será incluido por un caso de uso base.

Su representación gráfica es la siguiente:

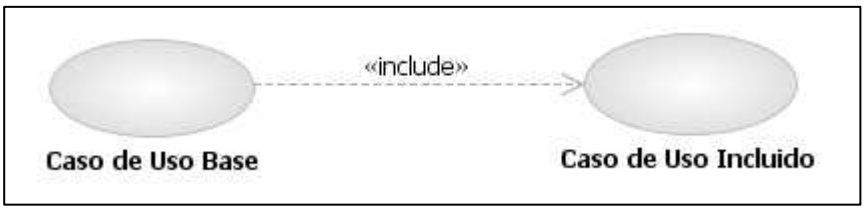


Figura 3.13. Relación <<*include*>> entre casos de uso.

Para entender la ejecución de un caso de uso incluido, analice la figura 3.14. Puede observar que el comportamiento del caso de uso incluido se inserta en un punto del caso de uso base. Cuando una instancia de caso de uso, el cual sigue la descripción de un caso de uso base, llega a un punto en donde la relación *include* es definida, seguirá la descripción del caso de uso incluido. Una vez que la inclusión se lleva a cabo, la instancia del caso de uso regresará al caso de uso base, en el punto donde se detuvo.

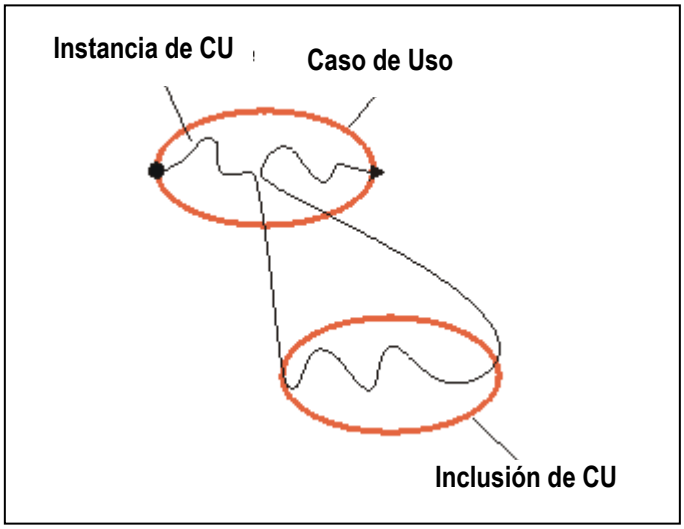


Figura 3.14. Ejecución de la inclusión.

1.3.2 Relación *extend*

Una relación *extend* se define como la agregación de pasos a la secuencia del caso de uso original, que pasará a conocerse como caso de uso base. Esta extensión se realiza en puntos indicados, llamados puntos de extensión, de manera específica dentro de la secuencia del caso de uso base. El caso de uso puede estar aislado pero, en algunas condiciones, su comportamiento puede extenderse con el comportamiento de otro caso de uso.

Esta relación se utiliza para modelar la parte de un caso de uso que el usuario puede ver **como comportamiento opcional del sistema**. De esta forma, se separa el comportamiento opcional del obligatorio.

Su representación gráfica es la siguiente:

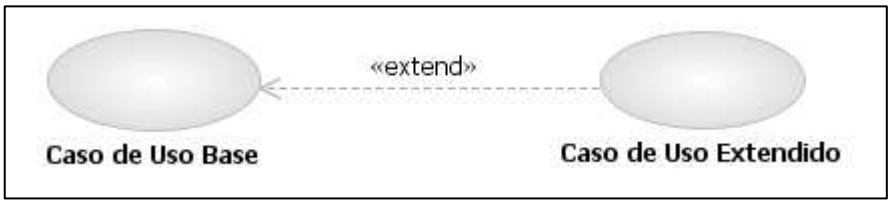


Figura 3.15. Relación <<*extend*>> entre casos de uso.

La figura 3.16 ilustra la ejecución de un caso extendido. Note que cuando una instancia del caso de uso base, llega a un lugar en donde un punto de extensión se ha definido, la condición en la correspondiente relación *extend* es evaluada. Si la condición es verdadera, la instancia del caso de uso seguirá la extensión; de lo contrario, la extensión no se ejecuta.

Una vez que la instancia de caso de uso ha realizado la extensión, la instancia de caso de uso reanuda su ejecución al caso de uso base, en el punto donde se detuvo.

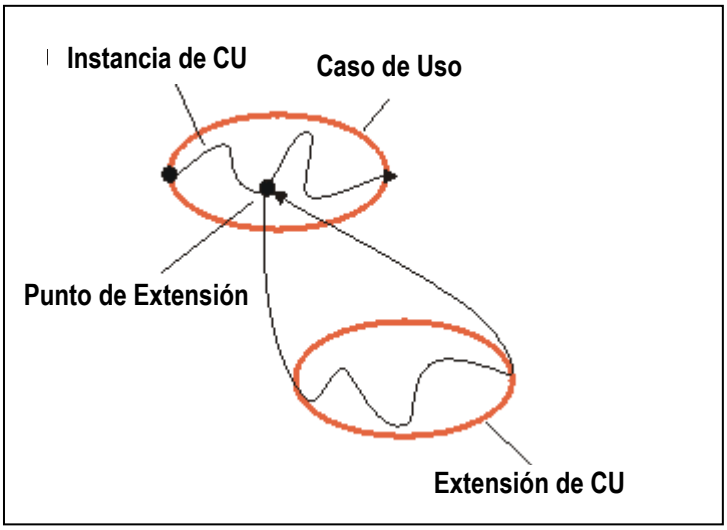


Figura 3.16. Ejecución de la extensión.

1.3.2 Relación de generalización

La generalización entre casos de uso es como la generalización entre clases. En este caso, significa que el caso de uso hijo hereda el comportamiento y el significado del caso de uso padre; el hijo puede añadir o redefinir el comportamiento del padre. La relación de generalización puede representarse también entre actores.

Su representación gráfica es la siguiente:

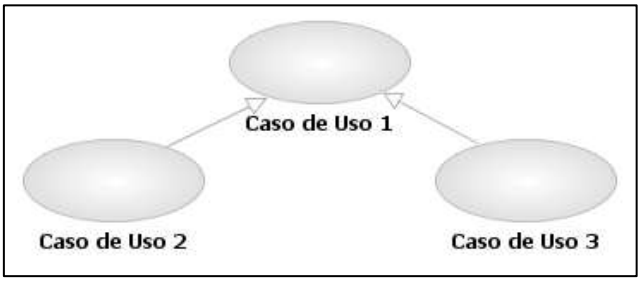


Figura 3.17. Relación de generalización entre casos de uso.

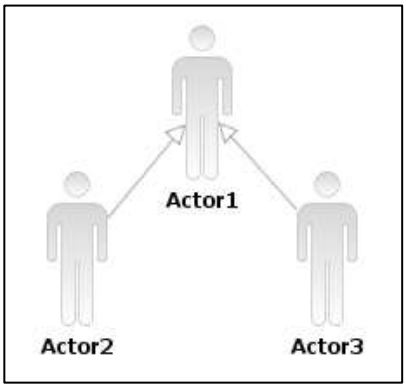


Figura 3.18. Relación de generalización entre actores.

Una instancia de caso de uso ejecutada por el caso de uso hijo seguirá el flujo de eventos descritos por el caso de uso padre insertando comportamiento adicional y modificando el comportamiento, tal como se define en el flujo de eventos del caso de uso hijo.

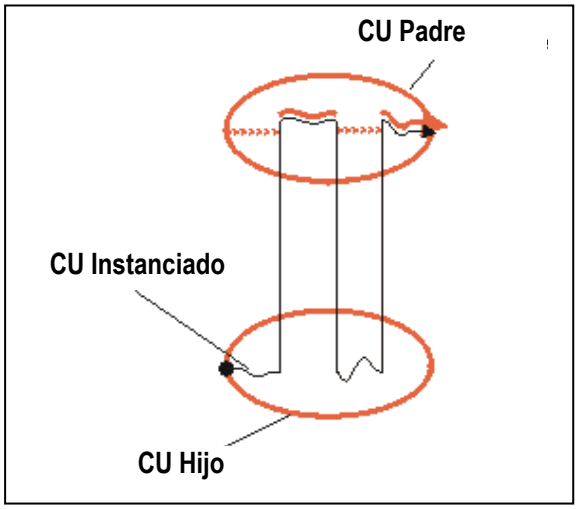


Figura 3.19. Ejecución de la generalización.

1.4 Casos de uso abstractos y concretos

Se dice que un caso de uso es abstracto sólo si se instancia en el contexto de otro caso de uso, es decir, dependen de otro caso de uso para instanciarse puesto que no existe un actor que lo active.

Un caso de uso es concreto si es iniciado por un actor y constituye un completo flujo de eventos. "Completo" significa que una instancia del caso de uso lleva a cabo toda la operación solicitada por el actor.

Por lo tanto, podemos afirmar que:

- Los casos de uso activados por un actor son concretos.
- Los casos de uso incluidos o extendidos que únicamente pueden instanciarse cuando son llamados por el caso de uso base son casos de uso abstractos
- En el caso de la generalización, generalmente el caso de uso padre será abstracto debido a que no están definidos completamente, pues los casos de uso hijos contienen las funciones específicas requeridas por los actores y serán los concretos.

La figura 3.20 ilustra ejemplos de casos de uso abstractos y concretos en un Diagrama de casos de uso estructurado. Es opcional escribir con letra cursiva el nombre del caso de uso abstracto.

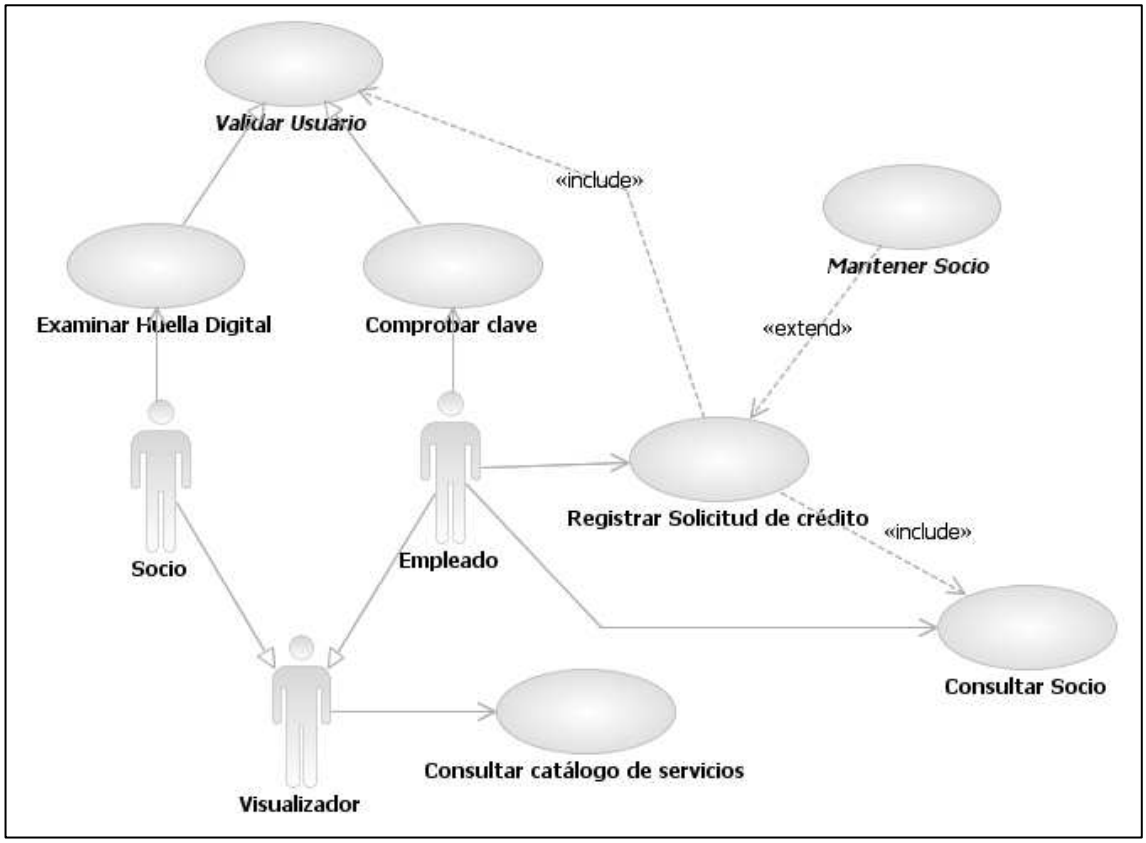


Figura 3.20. Diagrama de casos de uso estructurado.

ACTIVIDADES PROPUESTAS

Elabore el diagrama de casos de uso estructurado del siguiente caso.

CASO: Sistema de Medio de Pagos (SISMEPA)

La entidad financiera “ABC” a través de Internet permitirá que sus clientes (compradores y vendedores) puedan efectuar sus transacciones utilizando cuentas de la entidad para el proceso de sus pagos.

Los compradores enviarán sus pedidos de productos o servicios y los vendedores confirmarán el pedido y enviarán sus facturas mediante el sistema a implementar.

El pago de la operación se podrá hacer a través de las cuentas que los clientes tienen con “ABC”. El comprador utilizará la opción “Planificar Pagos” para planificar los pagos de la factura.

Por otro lado, el sistema efectuará el pago el día de las fechas de vencimiento con cargo a la cuenta del comprador. Si no hay saldo suficiente para ejecutar la transacción, el sistema notifica al comprador.

El SISMEPA se comunicara con el Sistema de Administración de Cuentas, para validar las transacciones al momento de pagar factura y efectuar el pago.

Los pasos para pagar una factura son:
1. El comprador ya ha recibido una factura de parte del vendedor con los productos y/o servicios solicitados.
2. El Comprador verifica si la factura a pagar corresponde al pedido efectuado.
3. El Comprador programa el pago de la factura y se genera una orden de pago con las fechas de vencimiento y las cuentas para la transferencia.
4. El comprador puede rechazar la factura si no concuerda con su pedido.

Resumen

- El caso de uso incluido se ejecuta de manera obligatoria por el caso de uso base para que cumpla con su objetivo.
- El caso de uso extendido nace de la excepción o condición de un caso de uso base.
- La generalización de casos de uso se da cuando dos o más casos de uso base poseen un comportamiento y estructura común.
- La generalización de casos de uso se da cuando dos o más actores requieren la misma funcionalidad del sistema.
- Un caso de uso es abstracto cuando no es iniciado directamente por un actor.
- Un caso de uso es concreto cuando es iniciado directamente por un actor.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  - http://rup.hops-fp6.org/process/modguide/md_incl.htm
    Aquí encontrará información detallada sobre la relación *include.*

  - http://rup.hops-fp6.org/process/modguide/md_exrel.htm
    Aquí encontrará información detallada sobre la relación *extend.*

  - http://rup.hops-fp6.org/process/modguide/md_ucgen.htm
    Aquí encontrará información detallada sobre la relación de *generalización* entre casos de uso.

  - http://rup.hops-fp6.org/process/modguide/md_actgn.htm
    Aquí encontrará información detallada sobre la relación de *generalización* entre actores.

8. DETALLAR UN CASO DE USO

No existe un documento estándar para una especificación de caso de uso. Sin embargo, una plantilla para una especificación sencilla de caso de uso utilizada comúnmente contiene la siguiente información:

- Nombre del caso de uso
- Breve descripción
- Actores implicados en el caso de uso
- Flujo de eventos. Incluye el flujo básico, subflujos y flujos alternativos
- Precondiciones
- Poscondiciones
- Puntos de extensión
- Requisitos especiales
- Prototipos

La plantilla de la especificación de caso de uso se muestra en el anexo 2.

1.1 Nombre del caso de uso

El nombre del caso de uso debe empezar con un verbo en infinitivo seguido de un sustantivo que plasme la funcionalidad del caso de uso. Veamos algunos casos:

- Para el mantenimiento de datos maestros, los cuales poseen subflujos como agregar, modificar, eliminar, etc.

  `Mantener <Nombre de la información que mantiene>`

  Por ejemplo: “Mantener Productos”, “Mantener Cliente”.

- Para el tratamiento de documentos legales, formales o de transacciones. Para tener el control adecuado de los perfiles de los usuarios y niveles de seguridad se suelen crear varios casos de uso que manipulan este tipo de documento.

  En caso de agregar:

  `Registrar/Generar <Nombre del documento formal>`

  Por ejemplo: “Generar Factura”, “Generar Contrato”.

  En caso de modificar o eliminar dependerá del documento y de cómo es tratado en la organización. Por ejemplo:

  Para eliminar una factura se crearía el caso de uso “Anular Factura” que registra el motivo de la anulación y que cambia el estado de la factura a anulada y para modificar una factura se creará el caso de uso “Generar Nota de Crédito”, ya que legalmente una factura no se puede modificar sin un documento que sustente el cambio.

- Para el tratamiento de la búsqueda de información, se utiliza Buscar en caso de datos maestros; y Consultar, en datos transaccionales

  `Buscar/Consultar <Información a buscar>`

  Por ejemplo: “Buscar Productos”, “Consultar Historial de Notas”.

- Para el tratamiento de la verificación de la información, la cual retorna un valor de verdadero o falso dependiendo de si encontró o no la información.

  Verificar/Validar <información a verificar>

  Por ejemplo: “Verificar Existencia de Producto”, “Validar Usuario”.

- Para el tratamiento de documentos informales o de uso interno, el cual incluye las opciones de mantenimiento en un sólo caso de uso.

  Gestionar/Administrar <Nombre del documento informal>

  Por ejemplo: “Administrar Cotización”, “Gestionar Nota de Pedido”.

  Es necesario aclarar que si uno de los documentos informales originó un documento formal ya no se puede modificar o anular. Por ejemplo, una cotización que se aprueba y genera una factura ya no podría modificarse o anularse.

1.2 Breve descripción

Debería ser un solo párrafo que resuma el objetivo del caso de uso. Además de los requerimientos funcionales que cumple.

1.3 Actores

Desde el punto de vista de un caso de uso específico, existen dos tipos de actores:

- Actores primarios o principales: Activan el caso de uso.
- Actores secundarios: Interactúan con el caso de uso después de haberse activado.

1.4 Flujo de eventos

Es una secuencia enumerada de pasos que describe la interacción del actor con el sistema. Incluye: Flujo básico, subflujos y flujos alternativos.

1.4.1 Flujo básico

Es el flujo principal del caso de uso y presenta las siguientes reglas:

a) El primer paso

Empieza por el actor primario haciendo algo para activar el caso de uso. Así:

1. El caso de uso se inicia cuando <actor> <función>

Por ejemplo:

1. El caso de uso se inicia cuando la recepcionista selecciona la opción “Generar Reserva” en la interfaz del menú principal.

Si el tiempo es el actor, se empieza así:

1. El caso de uso se inicia cuando es el fin de semana.

Si el caso de uso es abstracto, comienza así:

1. El caso de uso se inicia cuando es invocado por otro caso de uso base.

b) Se centra en el qué, no en el cómo
Mantenga los detalles de diseño fuera del caso de uso.

Por ejemplo, el siguiente paso es incorrecto.

5. El cliente pulsa el botón Aceptar.

La mejor forma de expresar ese paso es la siguiente:

4. El cliente selecciona "Aceptar Pedido".

c) Referencia a un caso de uso incluido
Para especificar la invocación a un caso de uso incluido se utiliza la siguiente expresión:

El sistema ***Incluye el CU Buscar Habitación***.

Por ejemplo:

7. La recepcionista solicita "Buscar Habitaciones" disponibles.
8. El sistema ***Incluye el CU Buscar Habitación***.

d) Ramificación dentro de un flujo
Para indicar una ramificación en el flujo se utiliza la palabra Si. La condición sujeta puede llevar a un conjunto de sub-acciones (desviaciones simples) o a un subflujo (desviaciones complejas).

El siguiente ejemplo utiliza ramificaciones.

5. Si la recepcionista elige un cliente
   a. Si selecciona "Modificar", ver el subflujo Modificar Cliente
   b. Si selecciona "Eliminar", ver el subflujo Eliminar Cliente.

e) Repetición dentro de un flujo
Para indicar la repetición de un conjunto de acciones se utiliza al final de la acción la siguiente expresión:

Si <actor> <función>, repite los pasos del <X_1> al <X_2>.

Por ejemplo:

...
7. La recepcionista solicita "Buscar Habitaciones" disponibles.
8. El sistema ***Incluye el CU Buscar Habitación***.
9. El sistema muestra las habitaciones disponibles.
10. La recepcionista ingresa la cantidad de personas para la habitación seleccionada.
11. El sistema valida la cantidad de personas ingresada.
12. El sistema calcula y muestra el subtotal del precio a pagar y el monto total.

13. Si la recepcionista quiere seleccionar otra habitación, repite los pasos del 7 al 12.

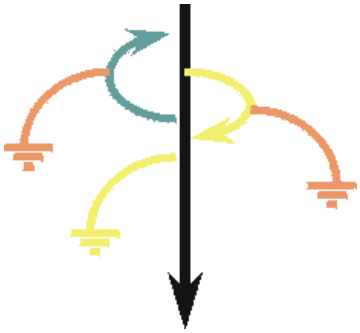
...

1.4.2 Subflujos

Es opcional en un caso de uso. Pueden presentarse varios subflujos y cada uno de ellos sigue las mismas reglas del flujo básico.

1.4.3 Flujos alternativos

Son rutas de acceso alternativas a través del caso de uso que capturan errores, ramificaciones e interrupciones en el flujo principal. En la figura 3.21, se ilustran los caminos posibles de una instancia de caso de uso (escenario).



3.21. Caminos del flujo de eventos.

A continuación, se muestra dos flujos alternativos para el caso de uso "Generar Orden de Reparación".

<Automóvil no Registrado>

En el punto 8, si el sistema verifica que el automóvil no está registrado muestra el MSG "Automóvil no registrado", la secretaria puede ir a "Registrar Automóvil" y continuar con el paso 9.

<Cancelar>

Si la secretaria solicita "Cancelar" antes de grabar la orden de reparación, el sistema cierra la interfaz y el caso de uso finaliza.

1.5 Precondiciones

Restringen el estado del sistema antes de que el caso de uso pueda empezar. Se describen en tiempo pasado. Si un caso de uso no tiene ninguna precondición, se debería escribir "Ninguna". Ejemplo:

1. El recepcionista se logeó en el sistema.
2. Lista disponible de Clientes.
3. Lista disponible de Habitaciones.

1.6 Poscondiciones

Restringen el estado del sistema después de que el caso de uso se ha ejecutado. Se describen en tiempo futuro. Si un caso de uso no tiene ninguna poscondición, se debería escribir "Ninguna". Ejemplo:

1. En el sistema quedará registrada la reserva con su detalle.
2. Las habitaciones seleccionadas se actualizarán al estado "Reservadas".

1.7 Puntos de extensión

Se utiliza para hacer referencia a un caso de uso extendido. Pueden existir varios puntos de extensión. Por ejemplo:

> En el paso 5, el sistema extiende al caso de uso Mantener Clientes – Flujo básico "Agregar cliente".

1.8 Requisitos especiales

En esta sección se especifican los requisitos no funcionales asociados a este caso de uso. A continuación. se muestra un requerimiento físico para el caso de uso "Generar Factura":

> Contar con formato especial para imprimir las facturas, con el logo de la empresa.

1.9 Prototipos

En esta sección, se muestran las interfaces gráficas de usuario diseñadas para el caso de uso. No es relevante mostrar las interfaces de los mensajes de advertencias o de error.

ACTIVIDADES PROPUESTAS

1. Utilice la información que se muestra abajo para elaborar la especificación de caso de uso (ECU), para el caso de uso “Alquilar Vehículos”. Debe incluir todas las partes de una ECU; asuma posibles subflujos, flujos alternativos, casos de uso incluidos y/o extendidos y un diseño de prototipo que concuerde con su ECU.

   Información para la ECU
   - Actor: Recepcionista
   - Pre-requisito: actor logeado al sistema
   - Propósito: registrar en el sistema el(los) vehículo(s) como “Alquilado” y generar una factura al cliente.
   - Datos del cliente: los que crea que son necesarios
   - Datos del (los) vehículo(s): los que crea que son necesarios
   - Reglas del negocio de RENTA CAR:
     - El cliente debe estar afiliado con un número de contrato con la empresa RENTA CAR para efectuarle el alquiler.
     - El cliente no puede tener facturas con multas (por morosidad), tiene que pagarlas antes del nuevo alquiler.
     - En el alquiler de vehículos se debe especificar la fecha de entrega. Si el cliente no entrega a tiempo, se le emitirá facturas por morosidad.
     - El formato especial para la impresión de facturas debe tener el logo de la empresa RENTA CAR.

2. Elabore la especificación de caso de uso para un caso de uso de su proyecto.

Resumen

- La especificación del caso de uso incluye:
  - Un nombre de caso de uso
  - Una breve descripción
  - Lista de actores
  - Flujo de eventos, el cual contiene el flujo principal, subflujos y flujos alternativos.
  - Precondiciones
  - Poscondiciones
  - Puntos de extensión
  - Requisitos especiales
  - Prototipos

- Los escenarios son a los casos de uso lo que objetos a las clases; es decir un escenario es básicamente una instancia de un caso de uso.

- Es importante aclarar que un caso de uso describe *qué* hace un sistema, pero no especifica *cómo* lo hace. Es por ello, que debe evitarse escribir detalles de diseño en los flujos de eventos.

- La referencia a un caso de uso incluido se realiza dentro del *Flujo principal y Sub flujos*, mientras que la referencia a un caso de uso extendido se especifica en la sección *Puntos de extensión.*

- Si desea saber más acerca de estos temas, puede consultar los siguientes libros.
  - "UML 2" de Jim Arlow e Ila Neustadt

    En el capítulo 4, sección 4.5, encontrará información sobre especificaciones de casos de uso.

9. PRIORIZACIÓN DE CASOS DE USO

Es la actividad de arquitectura y planificación de proyecto el cual consiste en clasificar los casos de uso según su importancia para establecer el orden de realización de los mismos. En este sentido, los casos de uso con significado arquitectónico se identifican y se priorizan. Una vez que se han priorizado los casos de uso, se puede decidir el orden de desarrollo del sistema.

Se establecen períodos, ciclos o iteraciones de trabajo para desarrollar la realización de los casos de uso. Se distribuyen los casos de uso en cada ciclo de trabajo; los casos de uso primarios deben realizarse en primer orden y, luego, los secundarios. Los casos de uso opcionales se deben dejar para el final de la realización.

1. Objetivos

El propósito de la priorización de los USE CASE es identificar los casos de uso primarios para la presente etapa de desarrollo del proyecto. Según estos criterios, se determinan los casos de uso críticos para especificarlos en esta etapa del proyecto.

2. Alcance

La priorización permitirá darle la debida atención (y con mas tiempo) a los USE CASE mas complejos e importante.

3. Priorización

Distingue a los USE CASE críticos o primarios de los secundarios. Más adelante, se especifica el criterio utilizado para determinar cuáles son primarios y cuáles son secundarios.

3.1. Nivel crítico (o primario)

Agrupa a los USE CASE que tienen que ver con las funciones básicas del sistema.

3.2. Nivel de baja importancia (o secundario)

Agrupa a los USE CASE que tienen que ver con las funciones de soporte del sistema y que no representan mayor riesgo para el proyecto.

4. Factores tomados en cuenta en la priorización

Se tomaron en cuenta pesos (que representan porcentaje) por cada factor que afecta a cada USE CASE. Los valores que pueden tomar los factores están en la escala del 1 al 10 (1: menor importancia; 10: mayor importancia). Se considerarán primarios a aquellos USE CASE que tengan un puntaje mayor a 6.5, ya que esto significa que superan el 65% de prioridad en el sistema (PONDERACIÓN).

- ***Importancia en el proceso del negocio*:** indica la relevancia que tiene la funcionalidad con el proceso de negocio. Una alta puntuación revela que las transacciones de la empresa se apoyan considerablemente en la funcionalidad que tiene este USE CASE. Su ponderación es 0.4.
- ***Complejidad de desarrollo*:** Indica la dificultad que se percibe del USE CASE, en cuanto a las tareas de análisis, diseño, implementación, pruebas e integración del mismo. Su ponderación es 0.3.
- ***Riesgo asociado*:** Indica la relación que se percibe entre el USE CASE y la lista de riesgos. Un alto valor en este factor indica que el caso de uso tiene

bastantes riesgos o riesgos de alto valor asociados. Los USE CASE con alto valor en este factor pueden ser considerados primarios debido a que deben ser enfrentados en las etapas iniciales. Su ponderación es 0.2.

- ***Impacto de los requerimientos no funcionales*:** Indica como afectan los requerimientos no funcionales (usability, reliability, performance, supportability) al proceso del negocio y en qué manera el USE CASE se vería comprometido. . Su ponderación es 0.1.

EJEMPLO DE PRIORIZACION DE LOS CASOS DE USO "LACTEOS LA LUZ"

I. A continuación se muestra los Subsistemas de "Lácteos La Luz", de acuerdo a sus objetivos y tareas.

SUBSISTEMAS

- Servicios al cliente
- Gestión de ventas
- Tareas del despachador
- Tareas ejecutivas.

A. Servicios al cliente

1. Registrar cliente
2. Elaborar pedido
3. Rastrear pedido
4. Consultar cuenta
5. Acusar recibo / reclamo

	Importancia en el proceso del negocio	Complejidad de desarrollo	Riesgo asociado	Impacto de requerimientos no funcionales	Total
Registrar cliente	10	6	9	9	8.5
Elaborar pedido	9	7	7	9	8
Rastrear pedido	6	8	5	8	6.75
Consultar cuenta	9	8	6	9	8
Acusar recibo /reclamo	5	5	3	7	5

B. Gestión de ventas

1. Aceptar / Rechazar pedido
2. Facturar pedidos
3. Actualizar cuenta
4. Consolidar pedido
5. Ordenar producción

	Importancia en el proceso del negocio	Complejidad de desarrollo	Riesgo asociado	Impacto de requerimientos no funcionales	Total
Aceptar /Rechazar pedido	8	4	5	3	5
Facturar pedidos	9	7	8	9	8.25
Actualizar cuenta	9	7	9	9	8.5
Consolidar pedido	10	6	8	6	7.5
Ordenar producción	6	8	7	3	6

C. Tareas del despachador

1. Configurar despachos
2. Rastrear pedido
3. Configurar embalaje
4. Configurar ruta
5. Acusar recibo / reclamo
6. Devolver mercancía

	Importancia en el proceso del negocio	Complejidad de desarrollo	Riesgo asociado	Impacto de requerimientos no funcionales	Total
Configurar despachos	9	6	6	7	7
Rastrear pedido	7	6	7	6	6.5
Configurar embalaje	8	8	7	7	7.5
Configurar ruta	7	6	8	6	6.75
Acusar recibo / reclamo	4	5	7	6	5.5
Devolver mercancía	4	7	5	5	5.25

D. Tareas Ejecutivas

1. Obtener información de productos
2. Evaluar el desempeño de productos
3. Generar informe

	Importancia en el proceso del negocio	Complejidad de desarrollo	Riesgo Asociado	Impacto de requerimientos no funcionales	Total
Obtener información de productos	8	7	7	6	7
Evaluar el desempeño de productos	9	8	7	7	7.75
Generar informe	7	8	7	7	7.25

II. Luego de haber priorizado cada subsistema, se agrupa por iteraciones, esta agrupación consiste en tomar los 3 CU más importantes del subsistema (con mayor ponderación). Estás iteraciones deberán ser desarrolladas en la fase de construcción del proceso del sistema.

A. Servicios al cliente

- **Iteración 1**

Registrar cliente		8.5
Consultar cuenta		8
Elaborar pedido		8

- **Iteración 2**

Rastrear pedido		6.75
Acusar Recibo / Reclamo		5

B. Gestión de ventas

- **Iteración 1**

Actualizar cuenta		8.5
Facturar pedidos		8.2
Consolidar pedido		7.5

- **Iteración 2**

Ordenar roducción	6	
Aceptar / Rechazar Pedido	5	

C. Tareas del despachador

- **Iteración 1**

Configurar embalaje	7.5	
Configurar despacho	7	
Configurar ruta		6.75

- **Iteración 2**

Rastrear pedido		6.5
Devolver mercancía	5.25	
Acusar Recibo / Reclamo		5.5

D. Tareas ejecutivas

- **Iteración 1**

Evaluar desempeño del producto	7.75	
Generar informe		7.25
Obtener información de productos	7	

Nota.- Requerimientos primarios serán aquellos que presenten un puntaje mayor a 6.5.

ACTIVIDADES PROPUESTAS

1. A partir de la matriz de actividades y requisitos de su proyecto, priorice los casos de uso.

Resumen

- La clasificación de los casos de uso, según su importancia, permiten establecer la prioridad de realización. Existen tres tipos: primarios, secundarios y opcionales.

- Se establecen períodos, ciclos o iteraciones de trabajo para desarrollar la realización de los casos de uso. En cada ciclo de trabajo, se distribuyen los casos de uso, comenzando por los casos de uso primarios y terminando con los opcionales.

CASOS PROPUESTOS

Elabore el diagrama de casos de uso estructurado para los siguientes casos.

CASO 1: SOFT CORPORATION

La empresa Soft Corporation cuenta con un área llamada Soporte y Sistemas. En los últimos años, con el crecimiento de la empresa, ha aumentado también el número de usuarios, lo que ha llevado a comprar más equipos. Los requisitos de atención y resolución de problemas también han aumentado considerablemente. Es por ello que el jefe del área ha pedido desarrollar un sistema para organizar mejor las tareas del área y optimizar el uso de los recursos.

El sistema debe permitir que tanto los técnicos como el personal de sistemas, incluso el jefe del área puedan registrar las incidencias hechas por los usuarios de la empresa. Para ello, el usuario debe indicar el código de su equipo y el problema que presenta, ya sea vía email o por teléfono. Además, los datos que son necesarios para dicho registro son el nombre del usuario (responsable del equipo), la fecha y hora en que se registra el problema y el nombre de la persona que ha registrado la incidencia.

El jefe del área se encargará de asignar las incidencias a cada técnico para que se haga responsable de solucionarlo. Cada técnico tendrá un límite de atención. Es por ello, que para la asignación de responsables, es necesario verificar la disponibilidad de los técnicos.

Por otro lado, los técnicos tendrán que consultar qué tareas tienen asignadas y dirigirse al área del usuario para atender el problema. Puede darse el caso de que el problema no sea muy grave y lo solucione allí mismo (en la oficina del usuario). Caso contrario, el técnico tendrá que llevarse el equipo a su área para hacer el cambio de alguna pieza del equipo. En este caso, el técnico debe solicitar al área de Logística que le envíe el repuesto que necesita la máquina. Esto puede tardar varios días. El problema, entonces, va a pasar por dos estados: Pendiente y Solucionado.

Cuando se soluciona el problema, el técnico registrará el informe técnico al sistema. Los técnicos también pueden ingresar a una opción de diccionario de fallas que les permitirá registrar y consultar las soluciones de determinado problema.

Adicionalmente, el jefe del área debe tener una opción para mantenerse informado del estado de las PC de usuarios que han tenido problemas complicados y de cuántos equipos han arreglado los técnicos diariamente. Por último, cualquier miembro del área debe tener la opción de consultar el historial de un equipo para verificar si es necesario o no la compra de uno nuevo.

CASO 2: TALLER DE AUTOMÓVILES

Un taller de servicio técnico de automóviles requiere un sistema para controlar la atención de problemas presentados en automóviles.

Cuando un cliente solicita los servicios del taller, la recepcionista registra una OST (Orden de Servicio Técnico). Para ello, la recepcionista verifica si el taller cuenta previamente con la información del cliente; en caso de no tenerlo, lo registra en ese preciso momento en el sistema. La información del cliente está compuesta por el número de DNI, nombre completo, dirección de residencia, sexo y teléfono de contacto. Adicionalmente, en la OST se ingresan las características del automóvil a reparar, como: placa, marca y modelo. La recepcionista procede a completar los datos de la orden que contiene la fecha y hora en que se llena la misma y la falla del automóvil. La orden se registra con el estado “Pendiente”. Luego la recepcionista le entrega la OST impresa al cliente y otra copia al técnico supervisor.

Para comenzar el proceso de reparación de equipos, el técnico supervisor procede a revisar el automóvil para realizar un diagnóstico del problema. El técnico supervisor consulta del sistema la bitácora de problemas, el cual presenta la solución respectiva. En caso de no estar registrada, lo registra al momento de registrar el informe técnico.

Al finalizar la reparación el técnico supervisor registra el informe. El automóvil puede ser reparado por más de un técnico. Para ello, el técnico supervisor previamente consulta la OST y luego ingresa el detalle de la solución. Adicionalmente, ingresa el nombre de los técnicos y el trabajo que realizó cada uno de ellos en el automóvil. Este registro actualiza el estado de la OST a “Atendida”. En caso de que el problema presentado sea nuevo, se registra en una bitácora de problemas técnicos.

CASOS PROPUESTOS

Elabore el diagrama de casos de uso estructurado para los siguientes casos.

CASO 1: DYNAMIC RESTAURANT

"DYNAMIC RESTAURANT" es un restaurant campestre que requiere de un sistema web para el control de atención a sus clientes que tengan las siguientes opciones:

Los socios desde el sistema, podrán registrar una reservación, indicando la cantidad de personas a asistir (adultos, niños, bebés), el tipo de evento, fecha y hora de asistencia. Si el cliente no está afiliado, en el sistema contará con la opción para registrarse. Una vez afiliado, el socio puede actualizar sus datos en cualquier momento desde el sistema.

Al momento de registrar una reserva, si el socio llegó a solicitar 6 reservas, el sistema le mostrará un formulario de premiación, en el cual ingresará y seleccionará algunos datos para ganarse un viaje de ida y vuelta para dos personas durantes 4 días. El dato a ingresar es el nombre completo de la persona que lo acompañará y los datos a seleccionar son el destino, el hotel y si desea o no un guía en el tours.

Las reservas son consultadas por el administrador del local, para realizar los preparativos. Mientras, al finalizar el mes, los registros de premiación son consultados por el jefe de marketing para realizar el sorteo.

Cuando el socio se acerca a caja, el cajero le genera un comprobante de pago por el consumo. El detalle del consumo debe indicar la cantidad de entradas, platos principales, postres y bebidas que las personas han consumido. Este registro actualiza la reserva a "Atendida".

Por otro lado, se requiere que, en el sistema, permita al administrador del local y al gerente general consultar las reservas atendidas y consultar los platos más solicitados por rango de fechas.

CASO 2: GAMES PARTY

"Games Party" es una empresa dedicada a prestar servicio de alquiler de juegos mecánicos y electrónicos para empresas o personas que deseen organizar fiestas infantiles. La empresa requiere de nuestros servicios para desarrollarle un sistema que le permita controlar la atención a sus clientes.

Para atender los servicios de alquiler, la recepcionista de la empresa consulta el catálogo de juegos mecánicos y le indica al cliente las características de cada una de ellas. Una vez que el cliente elige los juegos, la secretaria registra un contrato, para ello verifica la disponibilidad de dichos juegos. Si hay juegos disponibles los agrega en el contrato, mostrándose el monto total a pagar. Luego, procede a consultar los datos del cliente. Si el cliente es nuevo, la secretaria lo registra en ese preciso momento. Adicionalmente, la secretaria especifica el lugar y fecha donde se desea que se instalen dichos juegos. Luego la secretaria imprime el contrato y se lo entrega al cliente para que lo firme.

El cliente se acerca a caja para realizar el pago respectivo. El cajero consulta el contrato para registrar el alquiler de juegos, luego imprime el documento y se lo entrega al cliente sellado.

Por otro lado, la empresa controla la instalación de los juegos mecánicos en el lugar que el cliente indicó. Para ello, el técnico supervisor asigna el trabajo de instalación a uno o más técnicos, dependiendo de los juegos que se deben instalar. Previamente, consulta los registros de alquiler.

Por último, el técnico supervisor registra un informe técnico de la instalación de los juegos en el local del evento por cada registro de alquiler, indicando el código, nombre y apellidos de los técnicos que realizaron el trabajo. Este registro actualiza el estado del registro de alquiler a "Instalada".

ANEXO

1

PLANTILLAS DE DOCUMENTOS DE LA DISCIPLINA DEL MODELADO DE NEGOCIO

CONTENIDO

- Situación del negocio
- Actores del negocio
- Especificación de caso de uso del negocio
- Trabajadores del negocio
- Entidades del negocio

<Nombre del Sistema>
Situación del Negocio

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
01/08/2004	1.0	Inicial	Rosa Pérez

Posicionamiento - Visión del negocio

1. *Misión de la empresa.*

2. *Misión del área de negocio.*

3. *Oportunidad de negocio.*

<Brevemente describa las oportunidades de negocio a ser aprovechadas mediante su proyecto de Análisis y Diseño de Sistemas. Considere que estas oportunidades mencionadas deberían permitir mejorar los ingresos de la Empresa.>

4. *Descripción del problema*

<Provea una descripción resumida sobre los problemas a ser resueltos por su proyecto. Utilice el siguiente formato:>

El problema de	Cuellos de botella al momento de registrar a los alumnos en la prematricula y en la matrícula. Gran cantidad de reportes obtenidos del Sistema de Horarios para revisar y chequear manualmente las disponibilidades de horarios y pre-requisitos.
Afecta a	Alumno Personal Registros Académicos.
El impacto del cual es	Disminución del número de inscritos. Incumplimiento con los tiempos establecidos por el Jefe de Registros Académicos.
Una solución exitosa sería	Desarrollar un sistema de información que permita a los alumnos y docentes acceder vía web. Interconectar el Sistema de Información de Cursos de Extensión al Sistema de Horarios.

Glosario de términos del negocio

1. **CI:** Centro de Información de la institución académica.

2. **OOP:** Oficina de Oportunidades Laborales. Destinada a vincular a los alumnos y egresados del instituto con las empresas que requieren profesionales para puestos de prácticas y empleo.

3.

4.

Objetivos del negocio

1. Complementar la formación académica de los alumnos mediante el dictado de cursos de extensión a precios económicos.

2.

3.

4.

Reglas del negocio

1. Un alumno no puede registrarse en más de dos cursos.

2. El número acumulado de inasistencias por curso debe ser menor a 3.

3.

4.

<Nombre del Sistema>

Especificación de Caso de Uso de Negocio: Retiro y Cambio de Cursos

ECUN001

Versión 2.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
01/08/2004	1.0	Inicial	Rosa Pérez
15/08/2004	2.0	Revisión del procedimiento	Carlos Chu

Especificación de caso de uso de negocio: Retiro y cambio de cursos

1. Introducción

Propósito

Recolectar, analizar y describir las actividades que se realizan en el proceso gestionar el retiro y cambio de cursos.

Alcance

El presente documento se aplica a la descripción del proceso gestionar el retiro y cambio de cursos.

Definiciones, acrónimos y abreviaturas

Asistente SA: Asistente de Secretaría Académica.

Referencias

No existen documentos de referencias.

Resumen del documento

Este documento está dividido en 5 secciones básicas: Breve descripción del proceso, objetivo que satisface, flujos de trabajo, categoría a la que pertenece y gestor del proceso.

2. Retiro y cambio de cursos

2.1. Breve descripción

En este proceso se contemplan los pasos para gestionar el retiro y cambio de cursos. Este proceso brinda apoyo a los alumnos que trabajan y a quienes presentan problemas económicos.

3. Objetivos

- Minimizar en un 60% el tiempo de atención al alumno.

4. Flujo de trabajo

4.1. Flujo básico

4.1.1. El alumno solicita al Asistente SA realizar un retiro o cambio de horario de un curso.

4.1.2. El Asistente SA verifica si el alumno está matriculado en un determinado curso.

4.1.3. Si está matriculado en un determinado curso, el Asistente SA verifica si la solicitud está dentro de la fecha límite de presentación.

4.1.4. Si la solicitud está dentro de la fecha límite, el Asistente SA verifica si corresponde a un retiro o a un cambio de horario.

4.1.5. Si es retiro de un curso

- El Asistente SA registra retiro.

4.1.6. Si es un cambio de horario

- El Asistente SA muestra los horarios con vacantes disponibles.
- El alumno selecciona el horario del curso.

4.1.7. El Asistente SA actualiza los datos referentes a la matrícula del alumno y finaliza el proceso.

4.2. Flujos alternativos

4.2.1. En el punto 4.1.3 si el alumno no está matriculado es rechazado.

4.2.2. En el punto 4.1.4 si la solicitud ha pasado la fecha límite, esta es rechazada y finaliza el proceso.

5. Categoría

<Se coloca si es básica o de apoyo>

Apoyo.

6. Gestor del proceso

Jefe de Secretaría Académica.

<Luego, en las siguientes páginas agregar el diagrama de actividades y diagrama de clases de negocio.>

<Nombre del Sistema>
Actores del Negocio

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
01/08/2004	1.0	Inicial	Rosa Pérez

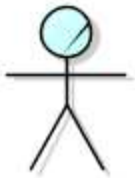
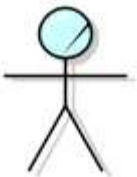
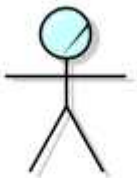
Actores del Negocio

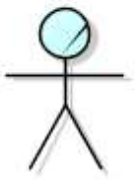
1. Descripción breve

<Una breve descripción sobre los actores del negocio que se presentan >

2. Características

<Escribir las características de cada actor de negocio.>

	Beneficiado con los cursos de extensión que le permitirán complementar su formación académica.
	Profesional a quien se le contratará para el dictado de los cursos de extensión.
	Sistema académico de horarios, matricula y alumnos de la carrera.



3. Relaciones

<Explicar las relaciones de generalización.>

4. Diagramas

<Diagrama general de caso de uso de negocio.>

<Nombre del Sistema>
Trabajadores del Negocio

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
01/08/2004	1.0	Inicial	Rosa Pérez

Trabajadores del negocio

1. Descripción breve

<Una breve descripción sobre los trabajadores del negocio que se presentan.>

2. Responsabilidades

<Escribir Las responsabilidades del trabajador del negocio, incluyendo sus nombres y descripciones breves.>

	Responsable del registro de trámites y gestiones documentarias del instituto.

3. Relaciones

<Se refiere a las relaciones en las que está envuelto el trabajador del negocio. Aquí se incluyen:

- *Las asociaciones, sus nombres, sus breves descripciones sobre las relaciones con las entidades de negocio.*
- *Las generalizaciones, sus breves descripciones y superclases.>*

4. Operaciones

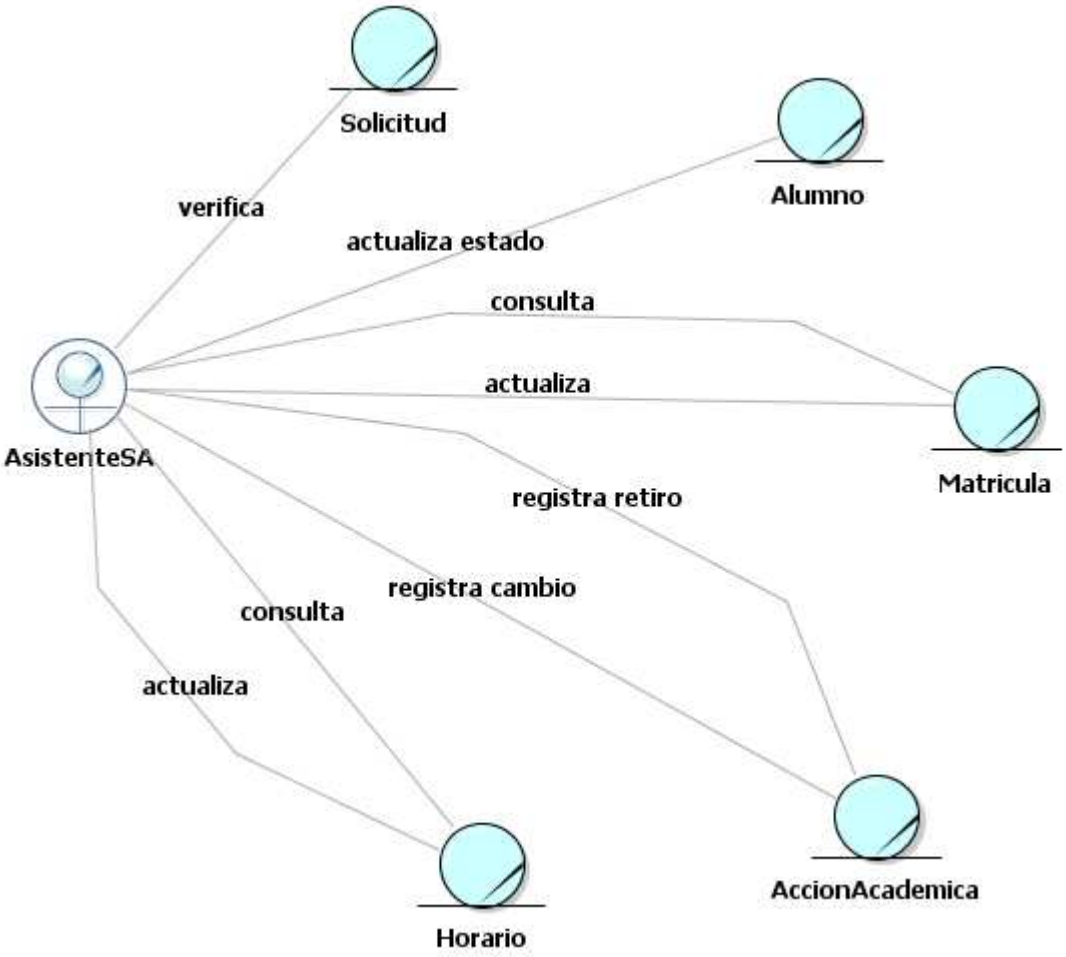
<Se refiere a las operaciones del trabajador del negocio. Se incluyen los nombres de las operaciones y breves descripciones.>

5. Atributos

<Aquí se describe las características de los trabajadores del negocio, se incluye su nombre, tipos y breves descripciones.>

6. Diagramas

<Diagrama de clases de negocio por cada caso de uso de negocio.>



<Nombre del Sistema>
Entidades del Negocio

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
01/08/2004	1.0	Inicial	Rosa Pérez

Entidades del negocio

1. Descripción breve

<Una breve descripción sobre las entidades del negocio que se presentan >

2. Responsabilidades

<*Aquí se describe el rol de la clase entidad en el negocio y su ciclo de vida desde su creación hasta su eliminación.>*

 **AccionAcademica**	Esta entidad representa la información manejada por los trámites: retiro de cursos, cambio de cursos, traslado interno o externo y reincorporación.
 **Alumno**	En esta entidad, se registra los datos del alumno.
 **Horario**	En esta entidad, se registran los días, horas de inicio y fin en el que se dictarán los cursos. Existen dos tipos de horarios: los horarios tentativos y los horarios definitivos.
 **Matricula**	En esta entidad, se registran las inscripciones de un alumno en uno o más cursos.
 **Solicitud**	Esta entidad, permite al alumno solicitar diversas acciones como el retiro, el cambio, la reincorporación o el traslado. Esta solicitud sólo se puede realizar en fechas definidas.

3. Relaciones

<Se refiere a las relaciones en las que está envuelta la clase entidad de negocio.>

4. Operaciones

<Se refiere a las operaciones que manipulan las entidades del negocio.>

5. Atributos

<Aquí se describe los atributos de las entidades del negocio, se incluye su nombre, tipos y breves descripciones.>

6. Diagramas

<Se agrega el diagrama de estados por cada entidad del negocio.>

Diagrama de estados de horario

ANEXO

2

PLANTILLAS DE DOCUMENTOS DE LA DISCIPLINA DE LA CAPTURA DE REQUISITOS

CONTENIDO

- Visión
- Especificación de requisitos de software
- Modelo de casos de uso
- Actores
- Especificaciones de casos de uso
- Matriz de Actividades Vs. Requisitos
- Lista de priorización

<Nombre del proyecto>
Visión

Versión 1.0

Historial de revisiones

Fecha	Versión	Descripción	Autor

Visión

1. Introducción

Propósito

[Breve descripción del propósito del presente documento, como puede ser servir de soporte a la especificación de las características software y de los atributos de las mismas, por ejemplo. También, reflejar si el sistema que se modela está dividido en otros subsistemas o bien el propósito general de la empresa]

Alcance

[Definición del alcance del presente documento, es decir, todo ámbito del que recoge características o detalles]

Definiciones, acrónimos, y abreviaciones

RUP: Son las siglas de Rational Unified Process. Se trata de una metodología para describir el proceso de desarrollo de software.

Referencias

- Glosario.
- Plan de desarrollo de software.
- RUP (Rational Unified Process).
- Diagrama de casos de uso.

2. Posicionamiento

Oportunidad de negocio

[Ventajas que obtendrá la empresa al implantar el sistema informático]

Sentencia que define el problema

El problema de	
afecta a	
El impacto asociado es	
una adecuada solución sería	

Sentencia que define la posición del Producto

Para	
Quiénes	
El nombre del producto	
Qué	
No como	
Nuestro producto	

3. Descripción de stakeholders (Participantes en el Proyecto) y usuarios

[Para proveer de una forma efectiva productos y servicios que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requisitos. También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos. No describe sus requisitos específicos, ya que éstos se capturan mediante otro artefacto. En lugar de esto, proporciona la justificación de por qué estos requisitos son necesarios.]

Resumen de stakeholders

[Hay varios stakeholders con un interés en el desarrollo. Presente una lista de estos stakeholders]

Nombre	Descripción	Responsabilidades
[Nombre del Stakeholder]	*[Descripción breve del Stakeholder]*	*[Resumen de las responsabilidades principales de los stakeholders relacionadas con el sistema bajo desarrollo. Por ejemplo, un stakeholder:* - *asegura que el sistema será mantenible* - *asegura que el producto tiene demanda en el mercado* - *monitorea el desarrollo del proyecto* - *aprueba los avances]*

Resumen de usuarios

Nombre	Descripción	Stakeholder
[Nombre de un usuario del sistema]	*[Descripción de responsabilidades del usuario]*	*Si el usuario no está directamente representado en el negocio, identifique qué stakeholders representa los intereses de este usuario]*
[Nombre de otro usuario del sistema]	*[Descripción de responsabilidades del usuario]*	*Si el usuario no está directamente representado en el negocio, identifique qué stakeholders representa los intereses de este usuario]*

Entorno de usuario

[Descripción del entorno de trabajo del usuario, características de los PC's a utilizar, sistemas operativos, etc.]

Perfil de los stakeholders

[Describa cada stakeholder en el sistema rellenando la tabla siguiente para cada stakeholder. Recuerde que los tipos de stakeholder pueden ser usuarios, departamentos o diseñadores técnicos. Un perfil completo cubriría los temas siguientes para cada tipo de stakeholder.]

3.4.1 <Nombre del Stakeholder>

Representante	*[nombre del stakeholder]*
Descripción	*[Descripción breve del tipo de stakeholder]*
Tipo	*[Califique la experticia o área de experticia del stakeholder, sus conocimientos y grado de instrucción. Puede ser un guru, experto, usuario, usuario casual, etc.]*
Responsabilidades	*[Lista de las principales responsabilidades de los stakeholders en relación con el sistema, es decir, teniendo en cuenta sus intereses en el sistema.]*
Criterio de éxito	*[¿Qué identifica como éxito en el proyecto para este stakholder?]*
Grado de participación	*[¿Cúales roles jugará este stakeholder dentro del proceso?]*
Comentarios	*[Cualquier comentario adicional]*

Perfiles de usuario

3.5.1 <Nombre de un usuario>

Representante	*[Nombre(s) de la persona que representa este usuario]*
Descripción	*[Descripción breve del tipo de usuario]*
Tipo	*[Califique la experticia o área de experticia del usuario, sus conocimientos y grado de instrucción. Puede ser un guru, experto, usuario, usuario casual, etc.]*
Responsabilidades	*[Lista de las principales responsabilidades dl usuario en relación con el sistema, es decir, teniendo en cuenta sus intereses en el sistema.]*
Criterio de éxito	*[¿Qué identifica como éxito en el proyecto para este usuario?]*
Grado de participación	*[¿Cuáles roles jugará este usuario dentro del proceso?]*
Comentarios	*[Problemas que interfieren con el éxito y cualquier otra información pertinente. Incluirían elementos que pueden hacen el trabajo del usuario más fácil o más difícil.]*

4. Descripción global del producto

4.1 Perspectiva del producto

[Ámbito de aplicación del sistema y expectativas del mismo]

4.2 Resumen de características

A continuación, se mostrará un listado con los beneficios que obtendrá el cliente a partir del producto:

Beneficio del cliente	Características que lo apoyan

4.3 Suposiciones y dependencias

[Todas las suposiciones y dependencias deben ser definidas por el cliente]

4.4 Costo y precio

[El costo y precio del sistema con todas las características software son decisión entre cliente y empresa de desarrollo software]

5. Descripción global del producto

5.1 <Una característica principal de software>

[Descripción de una característica software, ámbito y propiedades de la misma]

5.2 <Otra característica principal de software>

[Descripción de una característica software, ámbito y propiedades de la misma]

5.2.1 <Una subcaracterística software>

[Descripción de una característica software que deriva de una característica software jerárquicamente superior, ámbito y propiedades de la misma]

6. Restricciones

[A definir por el cliente]

7. Precedencia y prioridad

[A definir por el cliente]

8. Otros requisitos del producto

[A definir por el cliente]

Estándares aplicables

[A definir por el cliente]

Requisitos de sistema
[A definir por el cliente]

Requisitos de desempeño
[A definir por el cliente]

Requisitos de entorno
[A definir por el cliente]

9. Requisitos de documentación

9.1 Manual de usuario
[A definir por el cliente]

9.2 Ayuda en línea
[A definir por el cliente]

9.3 Guías de instalación, configuración, y fichero léame
[A definir por el cliente]

10. Atributos de características

Número y nombre de la característica	Estado	Beneficio	Esfuerzo	Riesgo	Estabilidad	Asignación
5.1 <Una característica>	Propuesta: *[Sí / No]* Aprobada: *[Sí / No]* Incorporada: *[Sí / No]*	*[A definir por el cliente]*	[Alto / Medio / Bajo]	*[A definir por el cliente]*	*[A definir por el cliente]*	[Personal asignado al desarrollo de esta característica]
5.2 <Otra característica>	Propuesta: *[Sí / No]* Aprobada: *[Sí / No]* Incorporada: *[Sí / No]*	*[A definir por el cliente]*	[Alto / Medio / Bajo]	*[A definir por el cliente]*	*[A definir por el cliente]*	[Personal asignado al desarrollo de esta característica]
5.2.1 <Una sub-característica>	Propuesta: *[Sí / No]* Aprobada: *[Sí / No]* Incorporada: *[Sí / No]*	*[A definir por el cliente]*	[Alto / Medio / Bajo]	*[A definir por el cliente]*	*[A definir por el cliente]*	[Personal asignado al desarrollo de esta característica]
5.2.2 <Otra sub-característica>	Propuesta: *[Sí / No]* Aprobada: *[Sí / No]* Incorporada: *[Sí / No]*	*[A definir por el cliente]*	[Alto / Medio / Bajo]	*[A definir por el cliente]*	*[A definir por el cliente]*	[Personal asignado al desarrollo de esta característica]
5.3 <Otra característica>	Propuesta: *[Sí / No]* Aprobada: *[Sí / No]* Incorporada: *[Sí / No]*	*[A definir por el cliente]*	[Alto / Medio / Bajo]	*[A definir por el cliente]*	*[A definir por el cliente]*	[Personal asignado al desarrollo de esta característica]

<Nombre del proyecto>
Especificación de Requisitos de Software

Versión 1.0

Historial de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Especificación de requisitos de software

Esta sección contiene la descripción de los requerimientos de software con nivel de detalle suficiente para que los analistas y diseñadores definan el sistema para satisfacerlos y que los testadores prueben que el sistema los satisface.

1. Funcionalidad

El sistema debe:

1.1. Asociados a los casos de uso del sistema.

1.1.1. Registrar las solicitudes de servicio de exposición
1.1.2. Registrar información del artista
1.1.3. Registrar información de obras
1.1.4. Elaborar el documento de Rechazo de Pedido
1.1.5. Registrar resultados de la evaluación artística
1.1.6. Aprobar o rechazar obras artísticamente
1.1.7. Elaborar el documento de Rechazo de Obras
1.1.8. Registrar resultados de la evaluación económica
1.1.9. Registrar precio y ganancia de las obras
1.1.10. Elaborar el documento de Venta de Obras
1.1.11. Elaborar reporte de solicitudes mensuales
1.1.12. Elaborar reporte de obras rechazadas por la galería
1.1.13. Elaborar reporte de obras en venta de la galería
1.1.14. Registrar usuarios del sistema
1.1.15. Registrar perfiles de usuarios del sistema
1.1.16. Cambiar contraseña del usuario
1.1.17. Realizar login al sistema
1.1.18. Administrar copias de seguridad

1.2. Asociados a aspectos generales.

1.2.1. Obligar al usuario a que el cambio de contraseña sea cada 60 días
1.2.2. Incluir un mecanismo que permita su actualización automática sin la intervención del usuario
1.2.3. Mantener un registro de los errores. Para cada error el sistema debe registrar: el código del error, una descripción del error, la fecha y la hora del error
1.2.4. Permitir que los reportes sean exportados a Hoja de Cálculo con formato Microsoft Excel 2000

2. Usabilidad

2.1. El sistema permitirá a los usuarios el registro de la solicitud de servicio como promedio en 30 segundos.
2.2. El sistema permitirá a los usuarios de la galería realizar búsquedas sin entrenamiento previo.
2.3. El aspecto de la interfaz gráfica del sistema facilitará su empleo a usuarios con conocimientos mínimos en informática sin entrenamiento especializado más allá del uso de un web browser.
2.4. El sistema se ajustará a los estándares CUA (Common User Access) de IBM.
2.5. En caso de error del usuario el sistema informará claramente el mensaje del error y la solución.
2.6. El lenguaje empleado en la interfaz gráfica del sistema respetará los términos usados en el negocio.

3. Confiabilidad

3.1. El sistema debe estar disponible 24x7x365 días al año.
3.2. El tiempo promedio entre fallas (MTBF) del sistema será de 30 días
3.3. La duración promedio de una reparación (MTTR) del sistema no debe ser mayor de 20 minutos.
3.4. El sistema estará disponible al 95 por ciento entre las 8:00 AM y las 6:00 PM.
3.5. El sistema cumplirá los procedimientos definidos en el reglamento de la institución (GA-2006JX).

4. Rendimiento

4.1. Durante el proceso de evaluación de las solicitudes de servicio, el sistema permitirá el acceso concurrente de 100 especialistas de arte y de ventas como promedio.
4.2. Durante el proceso de solicitud de servicio, el sistema permitirá el acceso concurrente de 70 anfitriones como promedio.
4.3. El sistema almacenará la información de hasta 5000 artistas y 40 000 obras de arte.
4.4. El tiempo de respuesta promedio del sistema para las operaciones involucradas en el proceso de evaluación es de 5 segundos.
4.5. El 95 por ciento de las transacciones del sistema no deben exceder los 5 segundos.
4.6. El sistema debe soportar un promedio de 50 transacciones por minuto.

5. Soporte

5.1. El cliente Web del sistema soportará los navegadores Microsoft Internet Explorer 6.0 o superior y FireFox 1.5 o superior para Linux y para Windows.
5.2. El sistema será compatible con Windows 2000 profesional y Windows XP.
5.3. El sistema permitirá a un usuario su instalación sin entrenamiento previo.
5.4. El tiempo máximo para corregir una falla será una semana.
5.5. El sistema debe incluir un mecanismo que permita su actualización automática sin la intervención del usuario.

6. Consideraciones de diseño

6.1. El sistema debe alinearse con los sistemas de la Corporación y tener una disposición acorde con la información mostrada en dichos sistemas.
6.2. El sistema debe operar en cualquier computador personal con procesador Pentium IV o superior, 256 Mb de memoria RAM y disco duro de 20 Gb.
6.3. La tecnología a utilizar será ser J2EE sobre plataforma AS400El sistema deberá ser compatible con la maquina virtual de java 1.1 o superior.
6.4. El sistema deberá tener seguridad de encriptamiento sobre las imágenes
6.5. La arquitectura lógica deberá considerarse en tres capas.
6.6. El motor de base de datos deberá ser DB2 de IBM, como principal; y MySQL, como secundario.

7. Documentación de usuario en línea y sistema de ayuda

No aplica a este proyecto.

8. Componentes adquiridos

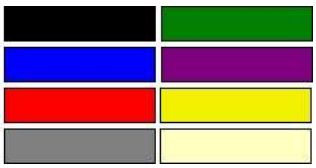
No aplica a este proyecto.

9. Interfaces

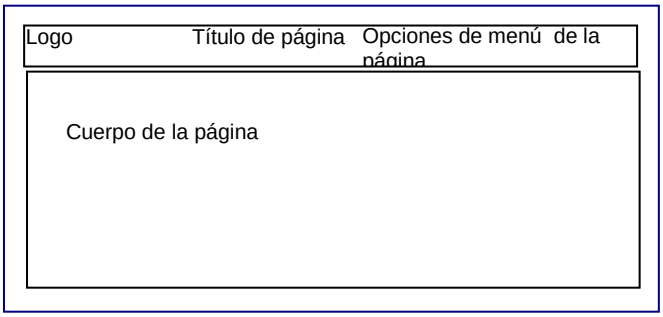
9.1. Interfaces de usuario

9.1.1. El diseño de la interfaz gráfica del sistema se alineará al estándar definido en la empresa para las aplicaciones Web.

9.1.2. Las interfaces de usuario estarán basadas en un diseño web en el que predominará los colores institucionales de ABC S.A. según la imagen adjunta.



9.1.3. El logotipo estará siempre presente en la parte superior izquierda de todas las interfaces.
9.1.4. El tipo de letra general será verdana de tamaño general de 3 para web.
9.1.5. El ancho de la página se limita a un tamaño de pantalla de 640x480 píxel sin scroll horizontal.
9.1.6. Las barras de scroll se activarán una vez que el texto sobrepase este límite.
9.1.7. En ancho de la pantalla, deberá estar limitado a 600 píxel con un 20% de espacio superior horizontal que ocupa todo el ancho de la pantalla para el logo en la parte izquierda y, sobre la derecha, las opciones que se pudieran presentar. El 80% restante corresponde al cuerpo de la pantalla.



9.1.8. Los gráficos que se presenten en las interfaces, tendrán un peso no mayor a los 100 kb.
9.1.9. Los reportes mostrarán el logotipo y nombre de la empresa ABC S.A.

9.2. Interfaces de hardware

No aplica a este proyecto.

9.3. Interfaces de software

9.3.1. El sistema se conectará con el sistema existente WEBNews para recibir el contenido de las noticias.

9.4. Interfaces de comunicaciones

No aplica a este proyecto.

10. Licenciamiento

No aplica a este proyecto.

11. Requerimientos legales, Copyright y otros

No aplica a este proyecto.

12. Estándares aplicables

No aplica a este proyecto.

<Nombre del proyecto>
Modelo de Casos de Uso

Versión 1.0

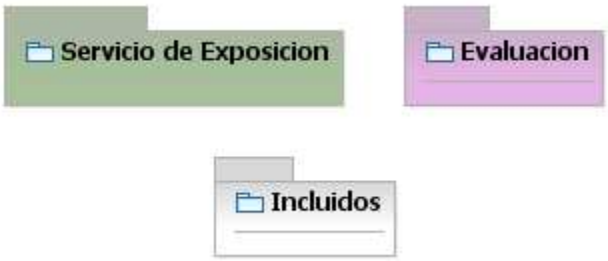
Historial de revisiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

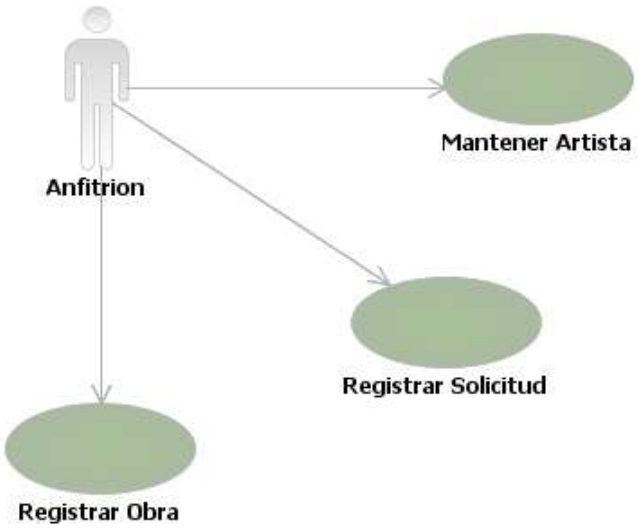
Modelo de casos de uso

1. Diagrama de actores

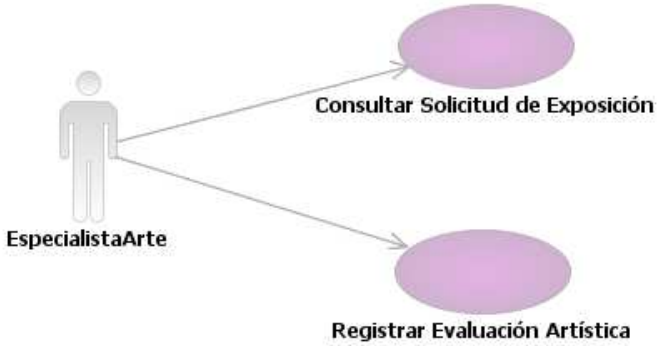
2. Diagrama de paquetes o módulos del sistema



3. Diagrama de casos de uso del paquete “Servicio de Exposición”

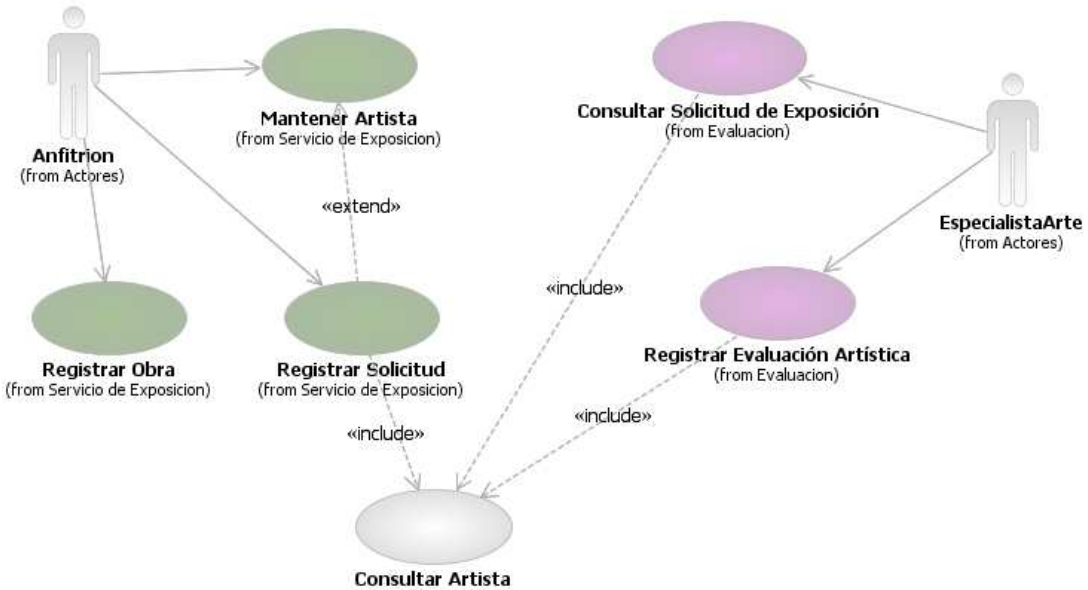


4. Diagrama de casos de uso del paquete “Evaluación”



5. Incluidos

6. Diagrama general de casos de uso



<Nombre del proyecto>
Actores

Versión 1.0

Historial de revisiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Actores

1. Anfitrión

Encargado de registro de la solicitud de servicio de exposición de arte solicitada por un artista, de la información de las obras asociadas a la solicitud, de la información de los artistas que realizan las solicitudes, así como elaborar el documento de Rechazo de Pedido.

2. Especialista de arte

Encargado del registro del resultado de la evaluación artística de las obras. Decide las obras que tienen la calidad necesaria para ser presentadas en la exposición y elabora el documento de Rechazo de Obras para el resto.

Especificación de caso de uso: Generar Reserva

Versión 0.3

Historial de revisiones

Fecha	Versión	Descripción	Autor
18/06/2006	0.1	Versión inicial	cnavarro
22/10/2007	0.2	Versión revisada	cnavarro
25/10/2007	0.3	Versión revisada	lpalacios
20/05/2008	0.4	Versión revisada	gguzman

Especificación de caso de uso: Generar Reserva

1. Breve descripción

El caso de uso permite a la recepcionista de un Hotel generar una reserva de habitación(es).Además de saber en que estados se encuentran: reservado, ocupado o disponible.

2. Actor(es)

Recepcionista

3. Flujo de eventos

3.1. Flujo Básico

1. El Caso de uso se inicia cuando la recepcionista selecciona la opción "Generar Reserva" en la interfaz del menú principal.
2. El sistema muestra la interfase RESERVA con los siguientes datos:
   Datos del cliente: Código y Nombres y Apellidos.
   Datos de la reserva: fecha de la reserva y cantidad de días a hospedarse.
   Datos de las habitaciones: número de habitación, tipo, categoría, capacidad, monto por día, sub total y monto total.
   Además incluye las opciones: **Buscar Cliente, Buscar Habitaciones, Agregar, Eliminar, Grabar Reserva y Salir.**
3. La recepcionista selecciona "Buscar Cliente".
4. El sistema ***Incluye el CU Buscar Cliente*.**
5. El sistema muestra los datos del cliente.
6. La recepcionista ingresa la fecha de reserva y cantidad de días.
7. La recepcionista solicita "Buscar Habitaciones" disponibles.
8. El sistema ***Incluye el CU Buscar Habitación.***
9. El sistema muestra datos de habitación disponible.
10. La recepcionista ingresa la cantidad de personas para la habitación seleccionada y selecciona Agregar.
11. El sistema valida la cantidad de personas ingresada.
12. El sistema calcula y muestra subtotal y monto total a pagar.
13. Si la recepcionista quiere seleccionar otra habitación, se repite del paso 7 al 12.
14. La recepcionista selecciona "Grabar Reserva".
15. El sistema autogenera un número de reserva.
16. El sistema graba la reserva con su detalle y registra la disponibilidad de la(s) habitación(es) en estado "Reservado" en las fechas solicitadas.

17. El sistema muestra el número de reserva, imprime el documento de reserva y muestra el MSG “Reserva generada con el Nro. 99999”.
18. La recepcionista cierra la interfaz RESERVA y regresa a la interfaz del menú principal del sistema y finaliza el caso de uso.

3.2. Subflujos

Ninguno.

3.3. Flujos alternativos

<Cliente no existe>

En el paso 5, si el sistema detecta que el cliente no existe, muestra el MSG: “Cliente no existe” y ofrecerá la posibilidad de registrar al nuevo cliente.

<Habitaciones no disponibles>

En el paso 9, si el sistema detecta que no hay habitaciones disponibles, muestra el MSG: “No hay habitaciones disponibles” y finaliza el caso de uso.

<Cantidad de Personas incorrecta>

En el paso 12, si la recepcionista ingresó una cantidad mayor a la capacidad de la habitación seleccionada, el sistema muestra el MSG “Ingrese una cantidad de personas menor o igual a la capacidad de la habitación” y continúa con el paso 10.

<Eliminar habitación>

Si la recepcionista solicita “Eliminar” una habitación seleccionada, antes de grabar, el sistema lo borra del detalle y el caso de uso continúa en el paso 13.

4. Precondiciones

4.1. El recepcionista está identificado en el sistema.
4.2. Lista de clientes disponibles
4.3. Lista de habitaciones disponibles

5. Poscondiciones

5.1. En el sistema, queda registrado la reserva con su detalle.
5.2. Las disponibilidades de habitaciones seleccionadas se registran en estado “Reservadas”.

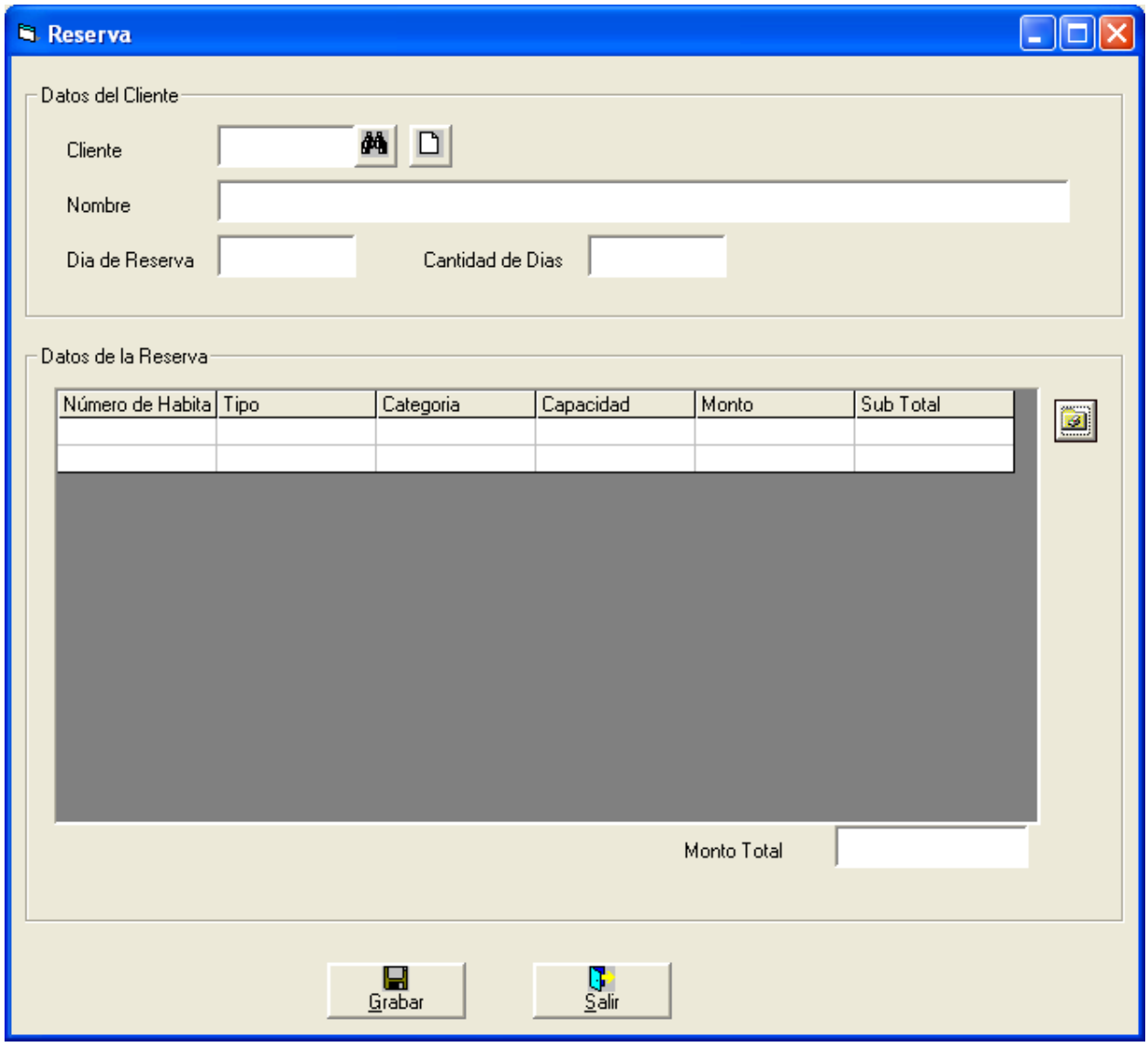
6. Puntos de extensión

En el paso 5, el sistema extiende al caso de uso Mantener Clientes – Flujo básico “Agregar Cliente”.

7. Requisitos especiales

Formato especial para los documentos de reserva, con el logo del hotel.

8. Prototipos



Matriz de Actividades Vs. Requisitos del sistema <Nombre del sistema>

Matriz de Actividades Vs. Requisitos del sistema <Nombre del sistema>							
Proceso de Negocio	**Actividad del Negocio**	**Responsable del Negocio**	**Requisito**		**Caso de Uso**		**Actores**
Proceso 1			R01		CUS01		
			R02		CUS02		
Proceso 2			R03		CUS03		
			R04		CUS04		
			R05		CUS05		
			R06		CUS06		

PRIORIZACION DE CASOS DE USO DEL <NOMBRE DEL SISTEMA>

ACTOR	*CASO DE USO*	0,4 IMPORTANCIA	0,3 COMPLEJIDAD	0,2 RIESGO	0,1 IMPACTO RNF	TOTAL

Glosario

Abstracción
Características esenciales de una entidad que la distingue de otros tipos de entidades. Define una frontera desde la perspectiva del observador.

Artefacto
Pieza discreta de información que es utilizada o producida por un proceso de desarrollo de software.

Caso de uso abstracto
Un caso de uso es abstracto sólo si se instancia en el contexto de otro caso de uso, es decir, dependen de otro caso de uso para instanciarse puesto que no existe un actor que lo active.

Caso de uso concreto
Un caso de uso es concreto si es iniciado por un actor y constituye un completo flujo de eventos. "Completo" significa que una instancia del caso de uso lleva a cabo toda la operación solicitada por el actor.

Condición de guardia
Condición que se debe satisfacer para permitir que se dispare una transición asociada. Es utilizado en Diagrama de Actividades después de un control de decisión.

Diagrama
Representación gráfica de un conjunto de elementos, representado en la mayoría de casos como un grafo conexo de nodos (elementos) y arcos (relaciones).

Diagrama de actividades
Diagrama que muestra el flujo de control datos entre actividades. Cubren la vista dinámica de un sistema.

Diagrama de casos de uso
Diagrama que representa procesos de negocio o funcionalidades del sistema y externos.

Diagrama de clases
Muestra un conjunto de clases y sus relaciones.

Diagrama de componentes
Muestra la organización y las dependencias entre un conjunto de componentes (elementos de implementación) del sistema.

Diagrama de comunicación
Diagrama de interacción que resalta la organización estructural de objetos que envían y reciben mensajes.

Diagrama de despliegue
Muestra la configuración en tiempo de ejecución de los nodos de procesamiento y dispositivos que componen una red.

Diagrama de estados
Representa los estados potenciales de los objetos y las transiciones entre esos estados.

Diagrama de objetos
Muestra un conjunto de objetos y enlaces en un momento dado.

Diagrama de secuencia
Diagrama de interacción que resalta la secuencia temporal de los mensajes entre objetos.

Elemento
Constituyente atómico de un modelo.

Escenario
Secuencia específica de acciones que ilustra un comportamiento.

Especificación
Descripción textual de la sintaxis y la semántica de un bloque de construcción específico; descripción declarativa de lo que algo es o hace.

Estereotipo
Extensión del vocabulario de UML que permite crear nuevos bloques de construcción derivados a partir de los existentes pero específicos a un problema concreto.

Ingeniería de Software
Rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas a los proyectos de desarrollo o mantenimiento de software de calidad.

Instancia
Manifestación concreta de un bloque de construcción de UML.

Modelo de proceso de software
Abstracción de un proceso real de desarrollo de software.

Notación
Sistema de signos convencionales que se adoptan para expresar un conjunto de conceptos sobre el sistema de software por desarrollar.

OMG Object Management Group
Consorcio del cual forman parte las empresas más importantes que se dedican al desarrollo de software.

Proceso de software
Conjunto de etapas cuyo objetivo es obtener un software de calidad. Conocido también como ciclo de vida del software.

Refinamiento
Relación que representa una especificación más completa de algo que ya ha sido especificado a cierto nivel de detalle.

Requisito
Característica, propiedad o comportamiento deseado de un sistema.

RUP *Rational Unified Process*
Proceso Unificado de Rational, metodología del proceso de ingeniería de *software* que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo.

Stakeholder
Personas u organizaciones que están directa o indirectamente envueltas en la elaboración o tomas de decisiones claves acerca de la funcionalidad y propiedades del Sistema. Asimismo, son fuentes de requisitos del sistema.

UML *Unified Modeling Language*
Lenguaje Unificado de Modelado, notación estándar para el modelado de sistemas *Software*.

Vista
Proyección de un modelo, que se ve desde una perspectiva o un punto de vista dado, y que omite entidades que no son relevantes desde esa perspectiva.

Vista dinámica
Aspecto de un sistema que destaca su comportamiento.

Vista estática
Aspecto de un sistema que destaca su estructura.