
**Universidad Autónoma del Estado de Hidalgo**

**Instituto de Ciencias Básicas e Ingenierías**

Monografía:
# Ingeniería de Requerimientos

Que para obtener el título de:
**LICENCIADO EN SISTEMAS COMPUTACIONALES**

Presenta:
**Ma. de Lourdes Pérez Huebe**

Asesor:
**Lic. en C. Luis Islas Hernández**

Pachuca, 2005

**AGRADECIMIENTOS**

***A DIOS***

Gracias dios por darme la oportunidad de vivir, por darme una familia tan maravillosa y gracias por darme la luz de cada mañana.

***A MIS PADRES Y HERMANOS***

Doña Lulú, Don Sabas, mil gracias por darme su educación, su apoyo, cariño y amor, yo creo que esto es la base para lograr lo que hasta ahora soy, de verdad mil gracias por confiar en mi en todo momento.

Itzel, Luis.....Carnalitos, muchas gracias por soportar mi mal humor, por compartir sus risas, sus bromas y por estar conmigo cuando los necesito.

***AMIGOS***

Gracias a cada uno de ustedes por estar conmigo en la buenas y en la malas, Luís como así le gusta que le llamemos los cuates gracias por su tiempo, dedicación y colaboración para concluir esto, que ya es una realidad, gracias por estar cuando lo necesite, Licenciada gracias por su apoyo y su amistad, Linda y Reyna muchas gracias por estar a mi lado en todo momento.

A la Universidad Autónoma del Estado de Hidalgo y profesores por mi formación.

Muchísimas gracias a todas la persona que confiaron en mi, esto es una parte de mi gratitud hacia ustedes.

MIL MIL GRACIAS

SINCERAMENTE LULU.

**ÍNDICE**

# I. INTRODUCCIÓN

La Especificación de Requerimientos que definen los servicios que un Sistema debe proveer y establecer los límites y restricciones en las operaciones del mismo. En la actualidad, el conjunto de procesos y métodos que tienen por objetivo capturar y formalizar estos requerimientos se ha venido a denominar Ingeniería de Requerimientos **(IR).**

La Ingeniería de Requerimientos es un campo muy activo dentro de la Informática, y en particular dentro de la Ingeniería del Software, y se dirige a unas actividades esenciales en el trabajo diario de las organizaciones de desarrollo de software. Se ha demostrado mediante varios estudios experimentales que la Ingeniería de Requerimientos es crítica respecto del éxito o fracaso de numerosos proyectos informáticos y su mala gestión tiene una gran incidencia probada en relación con el desbordamiento de costos o el incumplimiento de plazos de finalización. En esta área de la informática, como se menciona al inicio, se están desarrollando gran cantidad de métodos, técnicas, herramientas, y estándares, que en muchas ocasiones no son conocidos por parte de los profesionales del sector.

En la presente monografía se dan a conocer las especificaciones básicas de la ingeniería de requerimientos, dividida en cuatro capítulos, a continuación se especificaran.

- En el capítulo uno, se habla de la definición de la Ingeniería de Requerimientos, y de la importancia que tiene un requerimiento, sus formas en las que puede dividirse, sus características y las dificultades que tiene el definirlo.

- En el capítulo dos, se desglosa la importancia de la Ingeniería de Requerimientos, cuales son las personas involucradas, los puntos a considerar durante la Ingeniería y sus actividades que realiza.

- En el capítulo tres, se desglosan las técnicas utilizadas en la Ingeniería de Requerimientos, se hace una comparación de estas técnicas y además sus ventajas y desventajas de cada una de ellas.

- En el capítulo cuatro, se mostrará como se administran los requisitos dentro del espectro de gestión.

# II. OBJETIVO GENERAL

Incorporar los conceptos, técnicas y metodologías de la Ingeniería de Requerimientos aplicado a obtener información en diferentes niveles de la organización o funcionalidad que se desee estudiar, Así como entender la importancia de: el análisis y especificación de requisitos, los factores humanos, y de la documentación en el desarrollo de un software.

# III. OBJETIVOS ESPECÍFICOS

- Resaltar la importancia que tiene la Ingeniería de Requerimientos dentro del ciclo de desarrollo del software.
- Dar a conocer las diferentes alternativas que existen para identificar requerimientos.
- Ayudar a comprender la diferencia que existe entre las distintas técnicas utilizadas en la Ingeniería de Requerimientos
- Minimizar las dudas que se tiene sobre los casos de uso.

# IV JUSTIFICACIÓN

En la actualidad muchos desarrollos de aplicaciones fracasan por que no se hace un análisis correcto sobre la determinación de requerimientos que tiene el usuario para darle solución a su problema de información.

Es por esto que el presente trabajo aborda el tema; La ingeniería de Requerimientos que facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución sin ambigüedad, validando la especificación y gestionando loºs requisitos para que se transformen en sistema operacional.

# CAPÍTULO 1

# INGENIERÍA DE REQUERIMIENTOS.

## *1.1La Ingeniería de Requerimientos*

La Ingeniería de Requerimientos, se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto determinado [16]. El uso del término "ingeniería" implica que se deben utilizar técnicas sistemáticas y repetibles para asegurar que los requerimientos del sistema estén completos y sean consistentes y relevantes.

Normalmente, un tema de la Ingeniería de Software tiene diferentes significados. De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el glosario de la IEEE[1] [6].

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en los puntos anteriores (1) ó (2).

[1] INSTITUTO DE INGENIEROS ELECTRICISTAS Y ELECTRONICOS

## 1.2 *Definición de Ingeniería de Requerimientos.*

### ¿Para qué un Proceso de Ingeniería de Requerimientos?

El Proceso de Ingeniería de Requerimientos es un conjunto estructurado de actividades, mediante las cuales obtenemos, validamos y mantenemos el documento de especificación de requerimientos.

Las actividades del proceso incluyen la extracción de requerimientos, el análisis, la negociación y la validación.

No existe un proceso único que sea válido de aplicar en todas las organizaciones. Cada organización debe desarrollar su propio proceso de acuerdo al tipo de producto que se esté desarrollando, a la cultura organizacional, y al nivel de experiencia y habilidad de las personas involucradas en la ingeniería de requerimientos.

### Ingeniería de Requerimientos

El proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema llamado Ingeniería de Requerimientos [16]

> *La meta de la Ingeniería de Requerimientos es entregar una especificación de requisitos de software correcta y completa.*

### 1.2.1 Algunas definiciones para Ingeniería de Requerimientos según distintos autores.

- “Ingeniería de Requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema” [2]
- “Ingeniería de Requerimientos es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones” [8]
- “Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos” [1]
- “Ingeniería de requerimientos es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto” [23]

Dadas las definiciones anteriores, se aporta que la Ingeniería de Requerimientos es una sucesión de pasos que ayudan a obtener la definición clara, consistente y compacta de las especificaciones correctas que precisan el comportamiento de la información que requiere el usuario para su sistema.

### *1.3 Importancia de los requerimientos*

“La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con gente, máquinas y otros sistemas. Ninguna otra parte del trabajo afecta tanto el sistema l. Ninguna es tan difícil de corregir más adelante... Entonces, la tarea más importante que el ingeniero de software hace para el cliente es la extracción iterativa y el refinamiento de los requerimientos del producto." [3]

Los requerimientos se deben descubrir antes de empezar a construir un producto, y que puede ser algo que el producto debe hacer o una cualidad que el producto debe tener. Un requerimiento existe ya sea porque el tipo de producto demanda ciertas funciones o cualidades, o porque el cliente quiere que ese requerimiento sea parte del producto final. Así que si no se tienen los requerimientos correctos, no se puede diseñar o construir el producto correcto y, consecuentemente, el producto no permitirá a los usuarios finales realizar su trabajo.

### *1.4 División de los Requerimientos.*

Los requerimientos se pueden dividir en funcionales y no funcionales [8], a continuación se definen:

- Los requerimientos funcionales

Definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas, especifican los servicios que debe proporcionar la aplicación (por ejemplo, "la aplicación debe calcular el valor del portafolio de inversión del usuario") [5]

- Los requerimientos no funcionales

Tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez[2] del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, es el *cómo*, *cuándo* y *cuánto* del *qué*. [5]

[2] ES EL GRADO EN QUE EMTREGA RESULTADOS UN PROGRAMA

### *1.5 Características de los Requerimientos.*

Las características de los requerimientos son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de atributos tanto individualmente como en grupo [19]

**Características más importantes:**

**Necesario:** Si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

**Conciso:** Si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlos en un futuro.

**Completo:** Si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

**Consistente:** Si no es contradictorio con otro requerimiento.

**No ambiguo:** Cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

### *1.6 Dificultades para definir los requerimientos [12]*

- Los requerimientos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje es ambiguo)
- Existen muchos tipos de requerimientos y diferentes niveles de detalle.
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.
- Nunca son iguales. Algunos son más difíciles, más riesgosos, más importantes o más estables que otros.
- Los requerimientos están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso.
- Cada requerimiento tiene propiedades únicas y abarcan áreas funcionales específicas.
- Un requerimiento puede cambiar a lo largo del ciclo de desarrollo.

# CAPÍTULO 2

# PRINCIPALES ACTIVIDADES DE LA INGENIERÍA DE REQUERIMIENTOS

### *2.1 Importancia de la Ingeniería de Requerimientos*

Los principales beneficios que se obtienen de la Ingeniería de Requerimientos son [13, [7]]:

- Permite gestionar las necesidades del proyecto en forma estructurada: Cada actividad de la IR consiste de una serie de pasos organizados y bien definidos.
- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados: La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- Disminuye los costos y retrasos del proyecto: Muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro.
- Mejora la calidad del software: La calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc..
- Mejora la comunicación entre equipos: La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores, si este consenso no ocurre, el proyecto no será exitoso.

- Evita rechazos de usuarios finales: La ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

### *2.2 Personal involucrado en la Ingeniería de Requerimientos*

Son muchas las personas involucradas en el desarrollo de los requerimientos de un sistema. Es importante saber que cada una de esas personas tienen diversos intereses y juegan roles específicos dentro de la planificación del proyecto [18]; el conocimiento de cada papel desempeñado, asegura que se involucren a las personas correctas en las diferentes fases del ciclo de vida, y en las diferentes actividades de la IR.

No conocer estos intereses puede ocasionar una comunicación poco efectiva entre clientes y desarrolladores, que a la vez traería impactos negativos tanto en tiempo como en presupuesto. Los papeles o roles más importantes pueden clasificarse en:

**Usuario final:** Son las personas que utilizarán el sistema desarrollado. Ellos están relacionados con la disponibilidad y la fiabilidad del sistema; están familiarizados con los procesos específicos que debe realizar el software, dentro de los parámetros de su ambiente laboral. Serán quienes utilicen las interfaces y los manuales de usuario.

**Usuario Líder:** Son los individuos que comprenden el ambiente del sistema o el dominio del problema en donde será empleado el software desarrollado. Ellos proporcionan al equipo técnico los detalles y requerimientos de las interfaces del sistema.

**Personal de Mantenimiento:** Para proyectos que requieran un mantenimiento eventual, estas personas son las responsables de la administración de cambios, de la implementación y resolución de anomalías. Su trabajo consiste en revisar y mejorar los procesos del producto ya finalizado.

**Analistas y programadores:** Son los responsables del desarrollo del producto en sí; ellos interactúan directamente con el cliente.

**Personal de pruebas:** Se encargan de elaborar y ejecutar el plan de pruebas para asegurar que las condiciones presentadas por el sistema son las adecuadas. Son quienes van a validar si los requerimientos satisfacen las necesidades del cliente.

**Otras personas que pueden estar involucradas, dependiendo de la magnitud del proyecto, pueden ser:**

- ***Administradores de proyecto*:** Responsable de administrar los procesos internos del desarrollo de sistemas, para este cargo es necesario contar con habilidades de administración para evadir las diferentes situaciones que se presenten y además garantizar el cumplimiento de los objetivos dentro de los tiempos estipulados. Estas habilidades van desde la definición del proyecto, hasta la administración de las medidas de avance del mismo.
- ***Documentadores**:* Es el encargado de evidenciar la información generada a lo largo del desarrollo del producto hasta el lanzamiento, algunas de los informes que debe de realizar son la prototipificación[3](especificaciones, etc.), recopilación de información (análisis), creación de un documento a partir de las notas recopiladas (borradores), validación del manual (pruebas de uso) y distribución (por ejemplo. documentación impresa en papel). Cada etapa del proceso afecta a todas las demás etapas y debiera ser creada en conjunto con todas las otras etapas de desarrollo.
- **Diseñadores de base de datos**: Es el encargado de desglosar el problema en diferentes subsistemas, identificar la concurrencia inherente al problema, asignar los subsistemas a los procesadores y tareas, seleccionar los almacenes de datos, manejar el acceso a recursos globales, implementar el control del software,

[3]Especificaciones de la información del ejemplar que se está elaborando

## *2.3 Puntos a considerar durante la Ingeniería de Requerimientos.*

### 2.3.1 Objetivos del negocio y ambiente de trabajo

Aunque los objetivos no estén definidos frecuentemente en términos generales, son usados para descomponer el trabajo en tareas específicas [17]. En ciertas situaciones IR se enfoca en la descripción de las tareas y en el análisis de sistemas. Esta información proporciona la base para especificar el sistema, que será construido; aunque frecuentemente se añadan al sistema tareas que no encajan con el ambiente de trabajo planificado.

El nuevo sistema cambiará el ambiente de trabajo, sin embargo, es muy difícil anticipar los efectos actuales sobre la organización. Los cambios no ocurren solamente cuando un nuevo software es implementado y puesto en producción; también suceden cuando cambia el ambiente que lo rodea (nuevas soluciones a problemas, nuevo equipo para instalar, etc.). La necesidad de cambio es sustentada por el enorme costo de mantenimiento; aunque existen diversas razones que dificultan el mantenimiento del software, la falta de atención a la IR es la principal.

Frecuentemente la especificación inicial es también la especificación final, lo que obstaculiza la comunicación y el proceso de aprendizaje de las personas involucradas; esta es una de las razones por las cuales existen sistemas inadecuados.

### 2.3.2 Punto de vista de los clientes

Muchos sistemas tienen diferentes tipos de clientes. Cada grupo de clientes tiene necesidades diferentes y distintos requerimientos tienen diversos grados de importancia para ellos. Por otro lado, escasas veces tenemos que los clientes son los mismos usuarios; trayendo como consecuencia que estos soliciten procesos que causan conflictos con los solicitados por el usuario.

Diferentes puntos de vista también pueden tener consecuencias negativas, tales como redundantes, inconsistentes y ambiguos.

El tamaño y complejidad de los requerimientos ocasiona desentendimiento, dificultad para enfocarse en un solo aspecto a la vez y dificultad para visualizar relaciones existentes entre requerimientos.

### 2.3.3 Barreras de comunicación

La Ingeniería de Requerimientos depende de una intensa comunicación entre clientes y analistas de requerimientos; sin embargo, existen problemas que no pueden ser resueltos mediante la comunicación.

Para remediar esto, se deben abordar nuevas técnicas operacionales que ayuden a superar estas barreras y así ganar experiencia dentro del marco del sistema propuesto.

### 2.3.4 Evolución e integración del sistema

Pocos sistemas son construidos desde cero. En la práctica, los proyectos se derivan de sistemas ya existentes. Por lo tanto, los analistas de requerimientos deben comprender esos sistemas, que por lo general son una integración de componentes de varios proveedores.

Para encontrar una solución a problemas de este tipo, es muy importante hacer planeamientos entre los requerimientos y la fase de diseño; esto minimizará la cantidad de fallas directas en el código.

### 2.3.5 Documentación de requerimientos

Los documentos de Ingeniería de Requerimientos son largos. La mayoría están compuestos de cientos o miles de páginas; cada página contiene muchos detalles que pueden tener efectos profundos en el resto del sistema.

Normalmente, las personas se encuentran con dificultades para comprender documentos de este tamaño, sobre todo si lo leen cuidadosamente. Es casi imposible leer un documento de especificación de gran tamaño, pues difícilmente una persona puede memorizar los detalles del documento. Esto causa problemas y errores que no son detectados hasta después de haberse construido el sistema.

## *2.4 Actividades de la Ingeniería de Requerimientos*

En el proceso de IR son esenciales diversas actividades. En un proceso de Ingeniería de Requerimientos efectivo, estas actividades son aplicadas de manera continua y en orden variado.

Dependiendo del tamaño del proyecto y del modelo de proceso de software utilizado para el ciclo de desarrollo, las actividades de la IR varían tanto en número como en nombres, la Tabla 1 muestra algunos ejemplos de las actividades identificadas para cada proceso.

| **MODELO** | Oliver and Steiner 1996 | EIA / IS-632 | IEEE Std 1220-1994 | CMM nivel Repetitivo (2) | RUP |
|---|---|---|---|---|---|
| **Actividades** | Evaluar la información disponible | Análisis de requerimientos | Análisis de Requerimientos | Identificación de requerimientos | Análisis del Problema |
| | Definir métricas efectivas | Análisis funcional | Estudio de los requerimientos | Identificación de restricciones del sistema a desarrollar | Comprender las necesidades de los involucrados |
| | Crear un modelo del comportamiento del sistema | Síntesis | Validación de requerimientos | Análisis de los requerimientos | Definir el sistema |
| | Crear un modelo de los objetos | Análisis y control del sistema | Análisis funcional | Representación de los requerimientos | Analizar el alcance del proyecto |
| | Ejecutar el análisis | | Evaluación y estudio de funciones | Comunicación de los requerimientos | Modificar la definición del sistema |
| | Crear un plan secuencial de construcción y pruebas | | Verificación de funciones | Validación de requerimientos | Administrar los cambios de requerimientos |

Tabla 1. Actividades de la IR para diferentes modelos de procesos de Ingeniería de Software

A pesar de las diferentes interpretaciones que cada desarrollador tenga sobre el conjunto de actividades mostradas en la tabla anterior, podemos identificar y extraer cinco actividades principales que son:

- Análisis del Problema
- Evaluación y Negociación
- Especificación
- Validación
- Evolución

A continuación se explicará cada una de ellas [11], presentándolas en el orden en que debe estar para un nuevo proyecto.

### 2.4.1 Análisis del problema

El objetivo de está actividad es entender la verdadera necesidad del negocios.

Antes de describir que pasos deben cumplirse en esta actividad, se debe tener una definición clara del término “Problema”.

Un problema puede ser definido como la diferencia entre las cosas como se perciben y las cosas como se desean. Aquí vemos nuevamente la importancia que tiene una buena comunicación entre desarrolladores y clientes; de esta comunicación con el cliente depende que entendamos sus necesidades.

A través de la definición de problema, podemos ver entonces que la actividad de ***Análisis del Problema*** tiene por objetivo que se comprendan los problemas del negocio, se evalúen las necesidades iniciales de todos los involucrados en el proyecto y que se proponga una solución de alto nivel para resolverlo.

Durante el análisis del problema, se realizan una serie de pasos para garantizar un acuerdo entre los involucrados, basados en los problemas reales del negocio.

## Estos pasos son los siguientes:

- **Comprender el problema que se está resolviendo:** Es importante determinar quién tiene el problema realmente, considerar dicho problema desde una variedad de perspectivas y explorar muchas soluciones desde diferentes puntos de vista.

  Las soluciones iniciales, deben ser definidas tomando en cuanta tanto la perspectiva técnica como la del negocio.

- **Construir un vocabulario común:** Debe confeccionarse un glosario en dónde se definan todos los términos que tengan significados comunes (sinónimos) y que serán utilizados durante el proyecto.

  La creación de un glosario es sumamente beneficiosa ya que reduce los términos ambiguos desde el principio, ahorra tiempo, asegura que todos los participantes de una reunión están en la misma página, además de ser reutilizable en otros proyectos.

- **Identificar a los afectados por el sistema:** Identificar a todos los afectados evita que existan sorpresas a medida que avanza el proyecto. Las necesidades de cada afectado, son discutidas y sometidas a debate durante de ingeniería de requerimientos, aunque esto no garantiza que vaya a estar disponible toda la información necesaria para especificar un sistema adecuado.

  Para saber quiénes son las personas, departamentos, organizaciones internas o externas que se verán afectadas por el sistema, debemos realizar algunas preguntas.

  - ¿Quién usará el sistema que se va a construir?
  - ¿Quién desarrollará el sistema?
  - ¿Quién probará el sistema?
  - ¿Quién documentará el sistema?
  - ¿Quién dará soporte al sistema?
  - ¿Quién dará mantenimiento al sistema?
  - ¿Quién mercadeará, venderá, y/o distribuirá el sistema?
  - ¿Quién se beneficiará por el retorno de inversión del sistema?

  Como vemos, debe conocerse la opinión de todo aquél que de una u otra forma está involucrado con el sistema, ya sea directa o indirectamente.

- **Definir los límites y restricciones del sistema:** Este punto es importante pues debemos saber lo que se está construyendo, y lo que no se está construyendo, para así entender la estrategia del producto a corto y largo plazo. Debe determinarse cualquier restricción ambiental, presupuestaria, de tiempo, técnica y de factibilidad que limite el sistema que se va a construir.

### 2.4.2 Evaluación y negociación de los requerimientos

La diversa gama de fuentes de las cuales provienen los requerimientos, hacen necesaria una evaluación de los mismos antes de definir si son adecuados para el cliente. El término "adecuado" significa que ha sido percibido a un nivel aceptable de riesgo tomando en cuenta las factibilidades técnicas y económicas, a la vez que se buscan resultados completos, correctos y sin ambigüedades. En esta etapa se pretende limitar las expectativas del cliente apropiadamente, tomando como referencia los niveles de abstracción y descomposición de cada problema presentado.

***Los principales pasos de esta actividad son:***

- **Descubrir problemas potenciales:** En este paso se asegura que todas las características antes descritas estén presentes en cada uno de los requerimientos, es decir, se identifican aquellos requerimientos ambiguos, incompletos, inconsistentes.
- **Clasificar los requerimientos:** En este paso se busca identificar la importancia que tiene un requerimiento en términos de implementación. A esta característica se le conoce como prioridad y debe ser usada para establecer la secuencia en que ocurrirán las actividades de diseño y prueba de cada requisito. La prioridad de cada requerimiento dependerá de las necesidades que tenga el negocio.

  En base a la prioridad, cada requerimiento puede ser clasificado como mandatario, deseables o innecesarios.

  Un requerimiento es mandatario si afecta una operación crítica del negocio. Si existe algún proceso que se quiera incluir para mejorar los procesos actuales, estamos ante un requerimiento deseable; y si se trata de un requerimiento informativo o que puede esperar para fases posteriores, el requerimiento es catalogado como innecesario.

Una vez hecha esta categorización de los requerimientos, se puede tomar como estrategia general el incluir los mandatarios, discutir los deseables y descartar los innecesarios. Antes de decidir la inclusión de un requerimiento, también debe analizarse su costo, complejidad, y una cantidad de otros factores. Por ejemplo, si un requerimiento fuera trivial de implementar, puede ser una buena idea incluirlo por más que éste sea sólo deseable.

- **Evaluar factibilidades y riesgos:** Involucra la evaluación de factibilidades técnicas (¿pueden implementarse los requerimientos con la tecnología actual?); factibilidades operacionales (¿puede ser el sistema utilizado sin alterar el organigrama actual?); factibilidades económicas (¿ha sido aprobado por los clientes el presupuesto?).

En la actividad de evaluación y negociación, se incrementa la comunicación entre el equipo de desarrollo y los afectados. Para que los requerimientos puedan ser comunicados de manera efectiva, hay una serie de consideraciones que deben tenerse en cuenta; entre las principales están:

- Documentar todos los requerimientos a un nivel de detalle apropiado.
- Mostrar todos los requerimientos a los involucrados en el sistema.
- Analizar el impacto que causen los cambios a requerimientos antes de aceptarlos.

- Establecer las relaciones entre requerimientos que indiquen dependencias.
- Negociar con flexibilidad para que exista un beneficio mutuo. Enfocarse en intereses y no en posiciones

### 2.4.3 Especificación

En esta fase se documentan los requerimientos acordados con el cliente, en un nivel apropiado de detalle.

En la práctica, esta etapa se va realizando conjuntamente con el análisis, pero podríamos decir que la Especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML.

### 2.4.4. Validación

La validación es la etapa final de la IR. Su objetivo es verificar todos los requerimientos que aparecen en el documento especificado para asegurarse que representan una descripción, por lo menos, aceptable del sistema que se debe implementar. Esto implica verificar que los requerimientos sean consistentes y que estén completos. La validación representa un punto de control interno y externo; interno, porque se debe verificar internamente lo que se está haciendo, y externo, porque se debe validar con el cliente. Preferentemente, el documento de requerimientos obtenido en la etapa anterior sólo debería incluir los requerimientos que son aceptables para los usuarios. Pero es inevitable que durante la validación

se descubran algunos problemas relacionados con los usuarios, y esto se debe corregir antes de aprobarse el documento final de requerimientos.

En definitiva, la validación de especificaciones realmente significa asegurarse de que el documento de requerimientos represente una descripción clara del sistema, y es una verificación final de que los requerimientos cubren las necesidades de los usuarios. Esta etapa puede confundirse con la de análisis, pero la diferencia es clara: mientras que en el análisis se trabaja sobre el boceto del documento de requerimientos, en la validación se utiliza el documento final, lo que equivale a decir, los requerimientos "depurados".

# CAPÍTULO 3

# TÉCNICAS UTILIZADAS

# EN LA

# INGENIERÍA

# DE REQUERIMIENTOS

### *3.1 Técnicas utilizadas en las actividades de IR*

Existen varias técnicas para la IR [11], sin embargo, se mencionarán algunas de las más importantes. Cada técnica puede aplicarse en una o más actividades de la IR; en la práctica, la técnica más apropiada para cada actividad dependerá del proyecto que esté desarrollándose.

### *3.2 Entrevistas y Cuestionarios*

Las entrevistas y cuestionarios se emplean para reunir información proveniente de personas o de grupos. Durante la entrevista, el analista conversa con el encuestado; el cuestionario consiste en una serie de preguntas relacionadas con varios aspectos de un sistema.

Por lo común, 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.

Las preguntas que deben realizarse en esta técnica, deben ser preguntas de alto nivel y abstractas que pueden realizarse al inicio del proyecto para obtener información sobre aspectos globales del problema del usuario y soluciones potenciales.

Con frecuencia, se utilizan preguntas abiertas para descubrir sentimientos, opiniones y experiencias generales, o para explorar un proceso o problema. Este tipo de preguntas son siempre apropiadas, además que ayudan a entender la perspectiva del afectado y no están influenciadas por el conocimiento de la solución.

Las preguntas pueden ser enfocadas a un elemento del sistema, tales como usuarios, procesos, etc. El siguiente ejemplo muestra algunos tipos de preguntas abiertas.

**Del Usuario**

- ¿Quién es el cliente?
- ¿Quién es el usuario?
- ¿Son sus necesidades diferentes?
- ¿Cuáles son sus habilidades, capacidades, ambiente?

**Del Proceso**

- ¿Cuál es la razón por la que se quiere resolver este problema?
- ¿Cuál es el valor de una solución exitosa?
- ¿Cómo usted resuelve el problema actualmente?
- ¿Qué retrasos ocurren o pueden ocurrir?

**Del Producto**

- ¿Qué problemas podría causar este producto en el negocio?
- ¿En qué ambiente se usará el producto?
- ¿Cuáles son sus expectativas para los conceptos fácil de usar, confiable, rendimiento?
- ¿Qué obstáculos afectan la eficiencia del sistema?

El éxito de esta técnica combinada, depende de la habilidad del entrevistador y de su preparación para la misma. Los analistas necesitan ser sensibles las dificultades que algunos entrevistados crean durante la entrevista y saber cómo tratar con problemas potenciales. Asimismo, necesitan considerar no sólo la información que adquieren a través del cuestionario y la entrevista, sino también, su significancia.

### *3.3 Lluvia de ideas (Brainstorm)* [3]

Este método comenzó en el ámbito de las empresas, aplicándose a temas tan variados como la productividad, la necesidad de encontrar nuevas ideas y soluciones para los productos del mercado, encontrar nuevos métodos que desarrollen el pensamiento creativo a todos los niveles, etc.

Pero pronto se extendió a otros ámbitos, incluyendo el mundo de desarrollo de sistemas; básicamente se busca que los involucrados en un proyecto desarrollen su creatividad, promoviendo la introducción de los principios creáticos.

A esta técnica se le conoce también como torbellino de ideas, tormenta de ideas, desencadenamiento de ideas, movilización verbal, bombardeo de ideas, sacudidas de cerebros, promoción de ideas, tormenta cerebral, avalancha de ideas, tempestad en el cerebro y tempestad de ideas, entre otras.

### 3.3.1 Principios de la lluvia de ideas [24]

- Aplazar el juicio y no realizar críticas, hasta que no se agoten las ideas, ya que actuaría como un inhibidor. Se debe crear una atmósfera de trabajo en la que nadie se sienta amenazado.
- Cuantas más ideas se sugieren, mejores resultados se conseguirán: “la cantidad produce la calidad”. Las mejores ideas aparecen tarde en el periodo de producción de ideas, será más fácil que se encuentren las soluciones y sé tendrá más variedad sobre la que hay que elegir.
- La producción de ideas en grupos puede ser más efectiva que la individual.
- Tampoco se debe olvidar que durante las sesiones, las ideas de una persona, serán asociadas de manera distinta por cada miembro, y hará que aparezcan otras por contacto.

### 3.3.2 Integrantes que conforman en una lluvia de ideas[24]

- **El Director:** es la figura principal y el encargado de dirigir la sesión. Debe ser un experto en pensamiento creador. Su función es formular claramente el problema y que todos se familiaricen con él. Cuando lo haga, debe estimular ideas y hacer que se rompa el hielo en el grupo. Es el encargado de que se cumplan las normas, no permitiendo las críticas. Debe permanecer callado e intervenir cuando se corte la afluencia de ideas, por lo que le será útil llevar ya un listado de ideas. Debe hacer que todos participen y den ideas. Además, es la persona que da concede la palabra y da por finalizada la sesión. Posteriormente, clasificará las ideas de la lista que le proporciona el secretario.

- **El secretario:** registra por escrito las ideas según van surgiendo. Las enumera, las reproduce fielmente, las redacta y se asegura que todos están de acuerdo con lo escrito. Por último realizará una lista de ideas.

- **Los participantes:** pueden ser habituales o invitados; cualquier involucrado en el proyecto entra en esta categoría. Su función es producir ideas. Conviene que entre ellos no halla diferencias jerárquicas.

Las personas que componen el grupo deben estar motivadas para solucionar el problema, y con un ambiente que propicie la participación de todos. Pueden sentirse confiados y con la sensación de que pueden hablar sin que se produzcan críticas. Todas las ideas en principio deben tener el mismo valor, pues cualquiera de ellas puede ser la clave para la solución. Es necesario prestar mucha atención a las frases que pueden coartar la producción de ideas.

Además durante la celebración no deben asistir espectadores.

Se deben evitar todos los bloqueos que paralizan la idea: como son los hábitos o ideas preconcebidas, el desánimo o falta de confianza en si mismo, el temor y la timidez.

**Las fases de la aplicación en el Brainstorm son:**

*Descubrir hechos*

Al menos con un día de antelación, el director comunica por escrito a los miembros del grupo sobre los temas a tratar. El director explica los principios de la Tormenta de ideas e insiste en la importancia de tenerlos en cuenta.

La sesión comienza con una ambientación de unos 10 minutos, tratando un tema sencillo y no comprometido. Es una fase especialmente importante para los miembros sin experiencia. Se determina el problema, delimitándolo, precisándolo y clarificándolo.

A continuación se debe plantear el problema, recogiendo las experiencias que se poseen o consultando documentación. Cuando es complejo, conviene dividirlo en partes. Aquí es importante la utilización del análisis, desmenuzando el problema en pequeñas partes para conectar lo nuevo y lo desconocido.

Producir ideas es la fase de tormenta de ideas propiamente dicha, se van aplicando alternativas. Se busca producir una gran cantidad de ideas, aplicando los principios que hemos visto. Además, es útil cuando se ha trabajado mucho, alejarse del problema, pues es un buen momento para que se produzcan asociaciones. Muchas de las nuevas ideas serán ideas antiguas, mejoradas o combinadas con varias ya conocidas.

Al final de la reunión, el director da las gracias a los asistentes y les ruega que no abandonen el problema, ya que al día siguiente se le pedirá una lista de ideas que les puedan haber surgido. Se incorporan las ideas surgidas después de la reunión.

***Descubrir soluciones***

Se elabora una lista definitiva de ideas, para seleccionar las más interesantes. La selección se realiza desechando las ideas que no tienen valor y se estudia si son válidas las que se consideran interesantes. Lo mejor es establecer una lista de criterios de conveniencia para cada idea.

Se seleccionan las ideas más útiles y si es necesario se ponderarán. Pueden realizarlo los mismos miembros del grupo o crear otros para esta tarea; la clasificación debe hacerse por categorías (tarea que corresponde al director). Se presentan las ideas de forma atractiva, haciendo uso de soportes visuales.

***Prototipos***

Los prototipos permiten al desarrollador crear un modelo del software que debe ser construido.

Al igual que todos los enfoques al proceso de desarrollo del software, el prototipado comienza con la captura de requerimientos. Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requerimientos que son conocidos, y señalan áreas en las que será necesaria la profundización en las definiciones. Luego de esto, tiene lugar un "diseño rápido". El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles al usuario (por ejemplo, entradas y formatos de las salidas). El diseño rápido lleva la construcción de un prototipo.

El prototipo es evaluado por el cliente y el usuario, y utilizado para refinar los requerimientos del software a ser desarrollado. Un proceso de iteración tiene lugar a medida que el prototipo es “puesto a punto" para satisfacer las necesidades del cliente y permitiendo al mismo tiempo una mejor comprensión del problema por parte del desarrollador.

## Existen principalmente dos tipos de prototipos:

- **Prototipo rápido (o concept prototipe)**:El prototipado rápido es un mecanismo para lograr la validación pre-compromiso. Se utiliza para validar requerimientos en una etapa previa al diseño específico. En este sentido, el prototipo puede ser visto como una aceptación tácita de que los requerimientos no son totalmente conocidos o entendidos antes del diseño y la implementación. El prototipo rápido puede ser usado como un medio para explorar nuevos requerimientos y así ayudar a "controlar" su constante evolución.

- **Prototipo evolutivo:** Desde una perspectiva diferente, todo el ciclo de vida de un producto puede ser visto como una serie incremental de detallados prototipos acumulativos. Tradicionalmente, el ciclo de vida está dividido en dos fases distintas: desarrollo y mantenimiento.

  La experiencia ha demostrado que esta distinción es arbitraria y va en contra de la realidad ya que la mayor parte del costo del software ocurre después de que el producto se ha entregado.

El punto de vista evolutivo del ciclo de vida del software considera a la primera entrega como un prototipo inicial en el campo. Modificaciones y mejoras subsecuentes resultan en nuevas entregas de prototipos más maduros.

Este proceso continúa hasta que se haya desarrollado el producto final. La adopción de esta óptica elimina la distinción arbitraria entre desarrollo y mantenimiento, resultando en un importante cambio de mentalidad que afecta las estrategias para la estimación de costos, enfoques de desarrollo y adquisición de productos.

### *3.4 Proceso de Análisis Jerárquico (AHP)*

Esta técnica [25] tiene por objetivo resolver problemas cuantitativos, para facilitar el pensamiento analítico y las métricas. Consiste en una serie de pasos, a saber:

- Encontrar los requerimientos que van a ser priorizados.
- Combinar los requerimientos en las filas y columnas de la matriz n x n de AHP.
- Hacer algunas comparaciones de los requerimientos en la matriz .
- Sumar las columnas.
- Normalizar la suma de las filas.
- Calcular los promedios

Estos pasos pueden aplicarse fácilmente a una cantidad pequeña de requerimientos, sin embargo, para un volumen grande, esta técnica no es la más adecuada.

## *3.5 Administración de Requerimientos con Casos de Uso*

### 3.5.1 Definición de Casos de Uso

Los casos de uso [25] son una técnica para especificar el comportamiento de un sistema: "Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios."

Todo sistema de software ofrece a su entorno una serie de servicios. Un caso de uso [19] es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando se dice "alguien o algo" se hace referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y software.

Por ejemplo, un sistema de ventas, si pretende tener éxito, debe ofrecer un servicio para ingresar un nuevo pedido de un cliente. Cuando un usuario accede a este servicio, podemos decir que está "ejecutando" el caso de uso ingresando pedido.

Los Casos de Uso fueron introducidos por Jacobson en 1992 [8]. Sin embargo, la idea de especificar un sistema a partir de su interacción con el entorno es original de McMenamin y Palmer, dos precursores del análisis estructurado [22], que en 1984 definieron un concepto muy parecido al del caso de uso: el evento.

Para McMenamin y Palmer, un evento es algo que ocurre fuera de los límites del sistema, ante lo cual el sistema debe responder. Siguiendo con el ejemplo anterior, nuestro sistema de ventas tendrá un evento "Cliente hace Pedido". En este caso el sistema deberá responder al estimulo que recibe.

Sin embargo, existen algunas diferencias entre los casos de uso y los eventos
Las principales son [10]:

- Los eventos se centran en describir qué hace el sistema cuando el evento ocurre, mientras que los casos de uso se centran en describir cómo es el diálogo entre el usuario y el sistema.
- Los eventos son "atómicos": se recibe una entrada, la procesa, y se genera una salida, mientras que los casos de uso se prolongan a lo largo del tiempo.

- Para los eventos, lo importante es qué datos ingresan al sistema o salen de él cuando ocurre el evento (estos datos se llaman datos esenciales), mientras que para los casos de uso la importancia del detalle sobre la información que se intercambia es secundaria. Según esta técnica, ya habrá tiempo más adelante en el desarrollo del sistema para ocuparse de este tema.

Los casos de uso combinan el concepto de evento del análisis estructurado con otra técnica de especificación de requerimientos poco difundida: aquella que dice que una buena forma de expresar los requerimientos de un sistema es escribir su manual de usuario antes de construirlo. Esta técnica, si bien ganó pocos adeptos, se basa en un concepto muy interesante: al definir requerimientos, es importante describir al sistema desde el punto de vista de aquél que lo va a usar, y no desde el punto de vista del que lo va a construir. De esta forma, es más fácil validar que los requerimientos documentados son los verdaderos requerimientos de los usuarios, ya que éstos comprenderán fácilmente la forma en la que están expresados.

### 3.5.2 Los Casos de Uso y UML

A partir de la publicación del libro de Jacobson, gran parte de los más reconocidos especialistas en métodos Orientados a Objetos coincidieron en considerar a los casos de uso como una excelente forma de especificar el comportamiento externo de un sistema [15].

De esta forma, la notación de los casos de uso fue incorporada al lenguaje estándar de modelado UML (Unified Modeling Language) propuesto por Ivar Jacobson, James Rumbaugh y Grady Booch [25], tres de los precursores de las metodologías de Análisis y Diseño Orientado a Objetos, y avalado por las principales empresas que desarrollan software en el mundo [22]. UML va en camino de convertirse en un estándar para modelado de sistemas de software de amplia difusión.

A pesar de ser considerada una técnica de Análisis Orientado a Objetos [15], es importante destacar que los casos de uso poco tienen que ver con entender a un sistema como un conjunto de objetos que interactúan, que es la premisa básica del análisis orientado a objetos "clásico". En este sentido, el éxito de los casos de uso no hace más que dar la razón al análisis estructurado, que propone que la mejor forma de empezar a entender un sistema es a partir de los servicios o funciones que ofrece a su entorno, independientemente de los objetos que interactúan dentro del sistema para proveerlos.

Como era de esperar, es probable que en el futuro los métodos de análisis y diseño que prevalezcan hayan adoptado las principales ventajas de todos los métodos disponibles en la actualidad (estructurados, métodos formales, métodos orientados a objetos, etc.).

## 3.5.3 Definiciones y Notaciones [15]

### 3.5.3.1 Actores

Un actor es una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema que estamos construyendo de la misma forma. Por ejemplo, para una empresa que recibe pedidos en forma telefónica, todos los operadores que reciban pedidos y los ingresen en un sistema de ventas, si pueden hacer las mismas cosas con el sistema, son considerados un único actor: "Empleado de Ventas".

Los actores son externos al sistema que se va a desarrollar. Por lo tanto, al identificar actores se está empezando a delimitar el sistema, y a definir su alcance. Esto debe ser el primer objetivo de todo analista, ya que un proyecto sin alcance definido nunca podrá alcanzar sus objetivos.

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. De esta forma, un usuario puede acceder al sistema como distintos actores. La forma más simple de entender esto es pensar en perfiles de usuario de

un sistema operativo. Una misma persona puede acceder al sistema con distintos perfiles, que le permiten hacer cosas distintas. Los perfiles son en este caso equivalentes a los actores.

Otro sistema que interactúa con el que se está construyendo también es un actor. Los actores se representan con dibujos simplificados de personas, llamados en inglés "stick man" (hombres de palo). (Fig. 1)

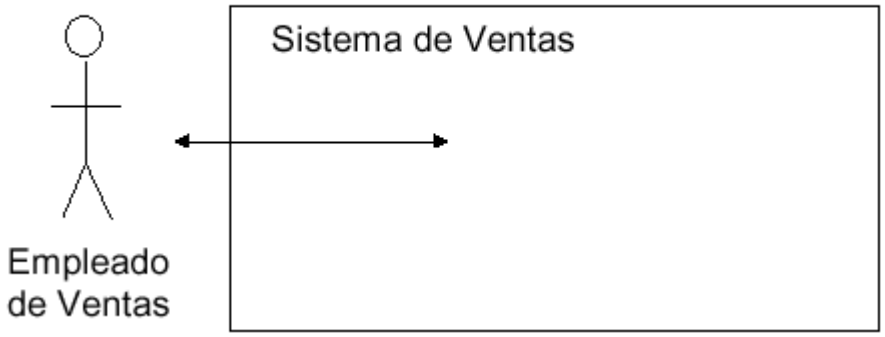


**Fig. 1 El empleado de ventas es un actor del sistema de ventas.**

Si bien en UML los actores siempre se representan con "hombres de palo", a veces resulta útil representar a otros sistemas con alguna representación más clara(Fig2).

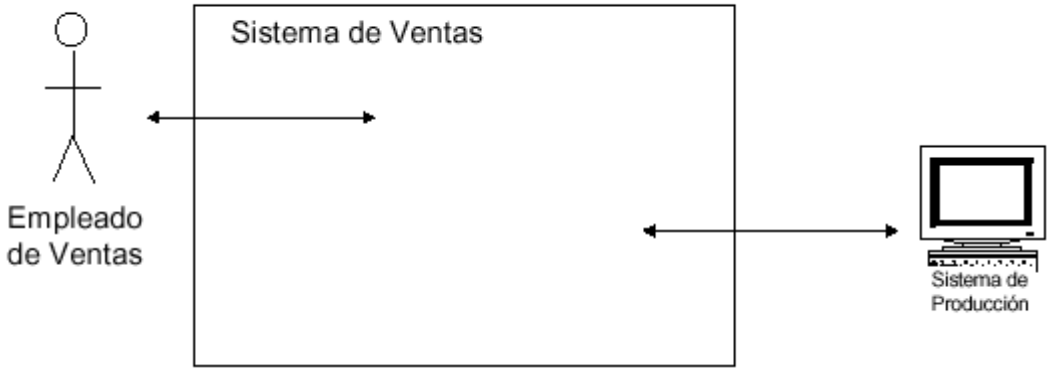

**Fig. 2 Cuando el actor es otro sistema se puede cambiar la notación.**

Las flechas, que existían en la propuesta original de Jacobson, pero que desaparecieron del modelo semántico de UML, pueden usarse para indicar el flujo de información entre el sistema y el actor. Si la flecha apunta desde el actor hacia el sistema, esto indica que el actor está ingresando información en el sistema. Si la flecha apunta desde el sistema hacia el actor, el sistema está generando información para el actor.

Identificar a los actores es el primer paso para usar la técnica de casos de uso.

Es común que los distintos actores coincidan con distintas áreas de la empresa en la que se implementará el sistema, o con jerarquías dentro de la organización (empleado, supervisor y gerente son distintos actores, si realizan tareas distintas).

Todos los actores participan de los casos de uso. Ahora bien, es lógico que existan intersecciones entre lo que hacen los distintos actores. Por ejemplo, un supervisor puede autorizar pedidos, pero también puede ingresarlos.

### 3.5.3.2 Casos de Uso

Como mencionamos anteriormente, un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios.

Un caso de uso es iniciado por un actor. A partir de ese momento, ese actor, junto con otros actores, intercambia datos o control con el sistema, participando de ese caso de uso.

El nombre de un caso de uso se expresa con un verbo en gerundio, seguido generalmente por el principal objeto o entidad del sistema que es afectado por el caso. Gráficamente, los casos de uso se representan con un óvalo, con el nombre del caso en su interior (Fig. 3).

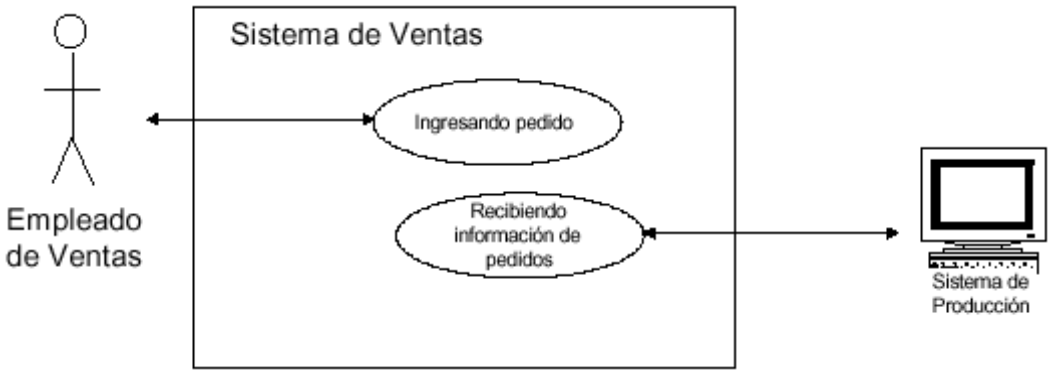


**Fig. 3 Los casos de uso se representan gráficamente con óvalos.**

Es importante notar que el nombre del caso siempre está expresado desde el punto de vista del actor y no desde el punto de vista del sistema. Por eso el segundo caso de uso se llama "Recibiendo información de pedidos" y no "Generando información de pedidos".

Los casos de uso tienen las siguientes características:

- Están expresados desde el punto de vista del actor.
- Se documentan con texto informal.

- Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción.
- Son iniciados por un único actor.
- Están acotados al uso de una determinada funcionalidad del sistema, claramente diferenciada.

El último punto es tal vez el más difícil de definir. Uno podría, después de todo, decir que todo sistema tiene un único caso de uso denominado "Usando el Sistema". Sin embargo, la especificación resultante sería de poca utilidad para entenderlo; sería como implementar un gran sistema escribiendo un único programa.

### 3.5.3.3 Extensiones (Extends)

Muchas veces, la funcionalidad de un caso de uso incluye un conjunto de pasos que ocurren sólo en algunas oportunidades. Suponga que se está especificando un sistema en el cual los clientes pueden ingresar pedidos interactivamente, y que dentro de la funcionalidad del ingreso de pedidos el usuario puede solicitar al sistema que le haga una presentación sobre los nuevos productos disponibles, sus características y sus precios. En este caso, hay una excepción dentro del caso de uso "Ingresando Pedido". La excepción consiste en interrumpir el caso de uso y pasar a ejecutar el caso de uso "Revisando Presentación de Nuevos Productos". En este caso se debe decir que el caso de uso "Revisando Presentación de Nuevos Productos", extiende el caso de uso "Ingresando Pedido" y se representa por una línea de trazos desde el caso que ‘extiende a’ al caso que es ‘extendido’ (Fig. 4).

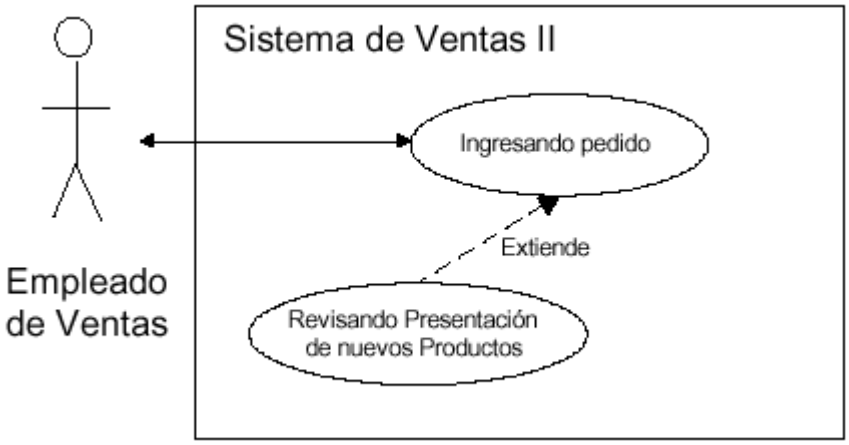


**Fig. 4 Una Relación de Extensión entre dos casos de uso.**

Las extensiones tienen las siguientes características:

- Representan una parte de la funcionalidad del caso que no siempre ocurre.
- Son un caso de uso en sí mismas.
- No necesariamente provienen de un error o excepción.

### 3.5.3.4 Usos (Uses)

Es común que la misma funcionalidad del sistema sea accedida a partir de varios casos de uso. Por ejemplo, la funcionalidad de buscar un producto puede ser accedida desde el ingreso de pedidos, desde las consultas de productos, o desde los reportes de ventas por producto.

¿Cómo hago para no repetir el texto de esta funcionalidad en todos los casos de uso que la acceden? La respuesta es simple: sacando esta funcionalidad a un nuevo caso de uso, que es usado por los casos de los cuales fue extraída. Este tipo de relaciones se llama “relaciones de uso”, y se representa por una línea punteada desde el caso ‘usa a’, al caso que es ‘usado’.

Se Dice, por ejemplo, que el caso de uso "Obteniendo reporte de ventas por producto", usa al caso de uso "Buscando producto" (Fig. 5).

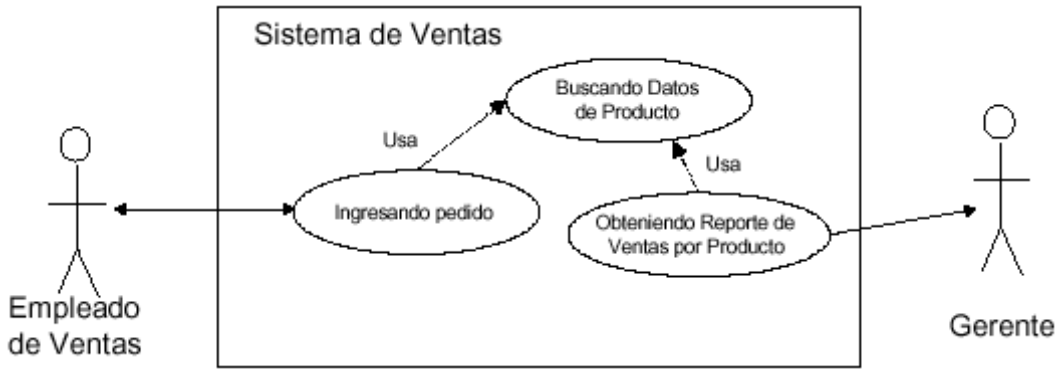


**Fig. 5. Relaciones de Uso entre Casos de Uso.**

Este concepto no es novedoso, es simplemente el concepto de la subrutina o subprograma usado en un nivel más alto de abstracción.

### 3.5.3.5 Abstracción

Al identificar relaciones de uso y extensión, puede ser que se extraigan casos de uso que son accedidos por varios actores. Por ejemplo, el caso de uso "buscando datos de producto" es accedido por muchos actores (el empleado de ventas que ingresa un pedido, el gerente que quiere obtener estadísticas por producto, el supervisor que quiere consultar la información de algún producto, etc.).

Ahora bien, como el caso de uso nunca se ejecuta fuera del contexto de otro caso de uso, se dice que es un caso de uso abstracto.

Le llamaremos abstracto porque no es implementable por sí mismo: sólo tiene sentido como parte de otros casos.

De la misma forma, el actor que participa de este caso de uso, que reúne características comunes a todos los actores de los casos de uso que lo usan, es un actor abstracto. En el ejemplo, se puede decir que se cuenta con un actor abstracto "Buscador de Datos de Producto". Los actores abstractos, entonces, son necesarios para no dejar sin actores a los casos de uso abstractos (Fig. 6).

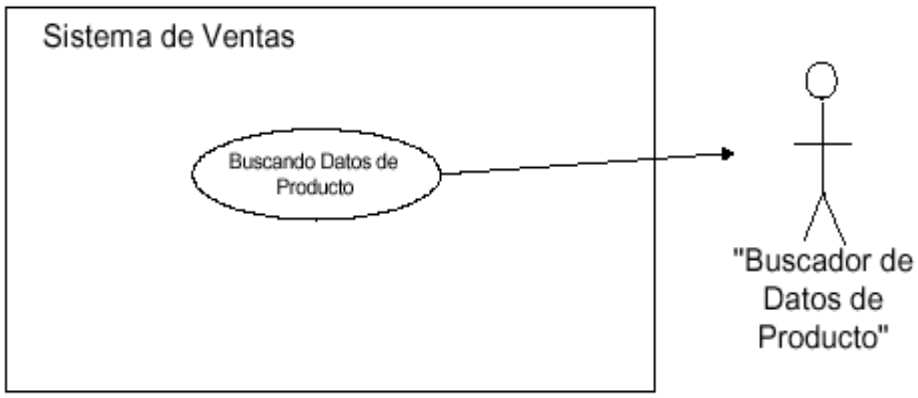


**Fig. 6 El nombre de los Actores abstractos suele no ser feliz.**

### 3.5.3.6 Herencia

La duda ahora es cómo relacionar este actor abstracto con los actores concretos: los que sí existen en la realidad y ejecutan casos de uso concretos, como "ingresando pedido" y "obteniendo estadísticas de ventas".

Para esto se debe usar el concepto de herencia, uno de los conceptos básicos de la orientación a objetos. Como todos los actores concretos también ejecutan el caso "buscando datos de producto", a través de la relación de uso, se dice que los actores concretos heredan al actor abstracto (Fig. 7).

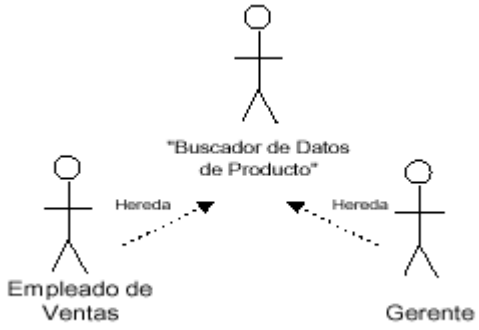


**Fig. 7 Herencia de Actores.**

La relación de herencia no necesariamente implica la existencia de un caso abstracto. Puede ocurrir que un actor ejecute todos los casos que ejecutan otro actor, y algunos más. En este sistema, el supervisor de ventas puede hacer todo lo que hace el empleado de ventas, pero además puede autorizar pedidos.

En este caso, el Supervisor de Ventas hereda al Empleado de Ventas, aunque el Empleado de Ventas no sea un actor abstracto. De esta forma, toda la funcionalidad que está habilitada para el Empleado de Ventas también lo está para el Supervisor.

### *3.6 Ventajas y Desventajas de las Técnicas de la Ingeniería de requerimientos.*

La tabla 2 muestra las ventajas y desventajas encontradas de las técnicas y herramientas utilizadas en la Ingeniería de los Requerimientos.

| TÉCNICA | VENTAJAS | DESVENTAJAS |
|---|---|---|
| **Entrevistas Y Cuestionarios** | • Mediante ellas se obtiene una gran cantidad de información correcta a través del usuario.<br>• Pueden ser usadas para obtener un pantallazo del dominio del problema.<br>• Son flexibles.<br>• Permiten combinarse con otras técnicas. | • La información obtenida al principio puede ser redundante o incompleta.<br>• Si el volumen de información manejado es alto, requiere mucha organización de parte del analista, así como la habilidad para tratar y comprender el comportamiento de todos los involucrados. |
| **Lluvia de Ideas** | • Los diferentes puntos de vista y las confusiones en cuento a terminología, son aclaradas por expertos.<br>• Ayuda a desarrollar ideas unificadas basadas en la experiencia de un experto. | • Es necesaria una buena compenetración del grupo participante. |
| **Análisis Jerárquico** | • Permite determinar el grado de importancia de cada requerimiento.<br>• Ayuda a identificar conflictos en los requerimientos.<br>• Muestra el orden en que deben ser implementados los requerimientos. | • Debe construirse un estándar claro de evaluación, que incluya la participación del cliente. |

| | | |
|---|---|---|
| **Casos de Uso** | • Representan los requerimientos desde el punto de vista del usuario.<br>• Permiten representar más de un rol para cada afectado.<br>• Identifica requerimientos estancados, dentro de un conjunto de requerimientos. | • En sistemas grandes, toma mucho tiempo definir todos los casos de uso.<br>• El análisis de calidad depende de la calidad con que se haya hecho la descripción inicial. |

**Tabla 2. Donde se comparan estas técnicas de la IR.**

### *3.7 Comparación entre algunas de las técnicas:* [11]

**Entrevistas vs. Casos de Uso:** Un alto porcentaje de la información recolectada durante una entrevista, puede ser usada para construir casos de uso. Mediante esto, el equipo de desarrollo puede entender mejor el ambiente de trabajo de los involucrados. Cuando el analista sienta que tiene dificultades para entender una tarea, pueden recurrir al uso de un cuestionario y mostrar los detalles recavados en un caso de uso. De hecho, durante las entrevistas cualquier usuario puede utilizar diagramas de casos de uso para explicar su entorno de trabajo.

**Entrevistas vs. Lluvia de Ideas:** Muchas de las ideas planteadas en el grupo, provienen información recopilada en entrevistas o cuestionarios previos. Realmente la lluvia de ideas trata de encontrar las dificultades que existen para la comprensión de términos y conceptos por parte de los participantes; de esta forma se llega a un consenso.

**Casos de Uso vs. Lluvia de Ideas:** La lista de ideas proveniente del brainstorm puede ser representada gráficamente mediante casos de uso. La tabla 3 muestra las técnicas que pueden ser utilizadas en las diferentes actividades de la IR.

| | **Análisis del Problema** | **Evaluación y negociación** | **Especificación de Requisitos** | **Validación** | **Evolución** |
|---|---|---|---|---|---|
| **Entrevistas y Cuestionarios** | X | | | | X |
| **Lluvia de Ideas** | | X | | | X |
| **Prototipos** | | | | X | |
| **Análisis Jerárquico** | | X | | | X |
| **Casos de Uso** | X | | X | | X |

Tabla 3. Actividades en dónde pueden ser utilizadas las técnicas de la IR.

# CAPÍTULO 4

# VALIDACIÓN Y GESTIÓN DE REQUISITOS

# 4.1 VALIDACIÓN DE REQUISITOS

El resultado del trabajo realizado es una consecuencia de la ingeniería de requisitos (especificación del sistema e información relacionada) y es evaluada su calidad en la fase de validación. La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto.

El primer mecanismo para la validación de los requisitos es la revisión técnica formal. El equipo de revisión incluye ingenieros del sistema, clientes, usuarios, y otros intervinientes que examinan la especificación del sistema buscando errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta, inconsistencias (es un problema importante), requisitos contradictorios o requisitos imposibles o inalcanzables.

## 4.2 GESTIÒN DE REQUISITOS.

Es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento. Muchas de estas actividades son idénticas a las técnicas de gestión de configuración del software.

Como en la Gestión de Configuración del Software (GCS), la gestión la gestión de requisitos comienza con la actividad de identificación. A cada requisito se le asigna un único identificador , que puede tomar la forma:

<tipo de requisito><requisito n,o>

| | | |
|---|---|---|
| **RF- <id del requisito>** | <**nombre del requisito funcional**> | |
| **Versión** | <numero de versión y fecha> | |
| **Autores** | <autor> | |
| **Fuentes** | <fuente de la versión actual> | |
| **Objetivos asociados** | <nombre del objetivo> | |
| **Descripción** | El sistema deberá comportarse tal como se describe en el siguiente caso de uso { concreto cuando <evento de activación> , abstracto durante la realización de los casos de uso <lista de casos de uso>} | |
| **Precondición** | <precondición del caso de uso> | |
| **Secuencia Normal** | **Paso** | **Acción** |
| | 1 | {El <actor> , El sistema} <acción realizada por el actor o sistema>, se realiza el caso de uso < caso de uso RF-x> |
| | 2 | Si <condición>, {el <actor> , el sistema} <acción realizada por el actor o sistema>>, se realiza el caso de uso < caso de uso RF-x> |
| | 3 | |
| | 4 | |
| | 5 | |
| | 6 | |
| | n | |
| **Postcondición** | <postcondición del caso de uso> | |
| **Excepciones** | **Paso** | **Acción** |
| | 1 | Si <condición de excepción>,{el <actor> , el sistema} }<acción realizada por el actor o sistema>>, se realiza el caso de uso < caso de uso RF-x>, a continuación este caso de uso {continua, aborta} |
| | 2 | |
| | 3 | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 1 | n segundos |
| | 2 | n segundos |
| **Frecuencia esperada** | <nº de veces> veces / <unidad de tiempo> | |
| **Importancia** | {sin importancia, importante, vital} | |
| **Urgencia** | {puede esperar, hay presión, inmediatamente} | |
| **Comentarios** | <comentarios adicionales> | |

# CONCLUSIONES

A pesar de la importancia que tiene la Ingeniería de Requerimientos, ha costado mucho que se le preste la atención adecuada a esta actividad. Aún quedan muchos desafíos que deben ser mejorados, tales como la integración de requerimientos funcionales y no funcionales, la evaluación de especificaciones alternativas, entre otras.

Cada actividad y técnica de la IR utilizada individualmente, dará diferentes soluciones para diferentes proyectos, incluyendo aquellos casos en los que el dominio y el área del problema son el mismo. Por esta razón, se considera que no existe un modelo de proceso ideal para la IR; encontrar el método o la técnica perfecta es una ilusión, pues cada método y técnica ofrece diferentes soluciones ante un problema.

En esta investigación se presentaron una serie de actividades y técnicas, que no pertenecen a un modelo de proceso en sí, sino, que son una alternativa al material publicado por diferentes autores y que, se consideraron como los más importantes.

Se debe recordar que la Ingeniería de Requerimientos es una actividad que involucra a clientes, usuarios, equipo de desarrollo, administradores de proyectos, etc.; por lo tanto, el proceso de IR no depende solamente de la forma en cómo se

percibe el problema, sino también, del nivel de experiencia que tengan los involucrados.

Es importante tomarse el tiempo necesario para conocer a los clientes y usuarios, así como su ambiente de trabajo. Esto, también ayuda a establecer una buena relación de trabajo y comunicación entre el equipo de desarrollo y los clientes. Es realmente necesario que los clientes y usuarios participen en la definición de sus requerimientos, pues ellos son los que deciden el destino en el proyecto, deciden si les gusta o no y además financian el proyecto.

En cuanto a la investigación realizada de la técnica de Casos de Uso para la Ingeniería de Requerimientos, puede decirse que estos son independientes del método de diseño que se utilice, y por lo tanto, del método de programación.

Luego de documentar los requerimientos de un sistema con casos de uso, se puede diseñar un sistema "estructurado" (manteniendo una separación entre datos y funciones), o un sistema Orientado a Objetos, sin que la técnica sea de mayor o menor utilidad en alguno de los dos casos. Esto da más flexibilidad al método, y probablemente contribuya a su éxito.

# REFERENCIAS BIBLIOGRÁFICAS

**[1]** "Advancing Software Engineering Project Technical Report RPT – 071", University of California at Irvine Departament of Information and Computer Sicence, Jun 1987.

**[2]** Boehm, "Software Engineering", IEEE Transactions on Computers Dic. 1976.

**[3]** Brainstorm, Alex Osborne, 1941

**[4]** Federick P. Brook, "Essence and Accident in Software Engineering". USA IEEE Computer, 1987.

**[5]** Ian Sommerville. "Software Engineering", Addison-Wesley, 1992.

**[6]** IEEE "Task Force on Requirements Engineering".

**[7]** Eric j Barudre, "Ingenieria de Software, Una perspective Orientada a Objetos", University Alfaomega, Boston.

**[8]** IT – Starts: "Developers Guide", National Computing Center, Manchester EE. UU. 1989.

**[9]** Ivar Jacobson, James Rumbaugh y Grady Booch, "El Lenguaje Unificado de Modelado", Addison Wesley, Madrid 1999.

**[10]** Jacobson, I. *et*. *al*. 1992. "Object-Oriented Software Engineering; A Use Case Driven Aproach", ACM Press. Adison-Wesley Publishing. Co. U.S.A.

**[11]** Kotonya, G. and Sommerville, I., "Requirements Engineering, Processes and Techniques", USA 1998.

**[12]** Kotonya, G. and Sommerville, I., *Viewpoints for requirements definition*, Software Engineering Journal, November 1992.

**[13]** Macaulay Linda A., “Requirements Engineering”, Verlag Berlin Heidelberg, New York 1996.

**[14]** McMenamin, S.M, J.F. Palmer (1984), “Essential Systems Analysis”, New York: Yourdon Press.

**[15]** Meilir Page-Jones, “Fundamentals of Object-Oriented Design in UML”, Addison Wesley Reading, Massachusetts, 2000

**[16]** Ortas, A. “Aproximación a la Ingeniería de Requerimientos”, Uruguay, Universidad ORT Uruguay, 2001.

**[17]** Pressman Roger S., **“**Software Engineering: A Practitioner's Approach**”,** 5th edition, McGraw-Hill 2001.

**[18]** Raghavan S., G. Zelesnik y G. Ford., “Notes on Requirements Elicitation”, Educational Materials CMU/SEI-94-EM-10, Software Egineering Institute, Carnegie Mellon University, 1994.

**[19]** Senn, James A. “Análisis y Diseño de Sistemas de Información”. Segunda Edición. McGraw Hill. 1992.

**[20]** Barbara Romero “Técnicas de apoyo a la toma de decisiones en la Administración Pública” Instituto Nacional de Administración Pública, Madrid, 1984.

**[21]** “Unified Modeling Language Specification”, v1.3. Object Management Group (OMG), 1999.

**[22]** Rainer Burkhardt, “UML: Unified Modeling Language”, Addison-Wesley, 1997.

# REFERENCIAS DE INTERNET

**[23]** http://www-306.ibm.com/software/rational/oifferings/reqanalysis.html

**[24]**http://brainstorming.co.uk/tutorials/brainstormingprinciples.html

**[25]**http://www.brainstorming.co.uk/tutorials/preparingforbrainstorming.html