



INGENIERÍA DEL SOFTWARE I

Tema 3

Requisitos

Univ. Cantabria – Fac. de Ciencias

Francisco Ruiz



Objetivos

- Conocer la naturaleza de los requisitos software.
- Conocer el proceso de desarrollo de requisitos y sus principales actividades: elicitación, análisis, especificación y validación de requisitos.
- Aprender a especificar requisitos aplicando el estándar IEEE 830.
- Comprender la importancia de una correcta gestión de requisitos.



Contenido

- Introducción
 - Definiciones
 - Características
 - Dificultades
- Niveles
 - Usuario
 - Dominio
 - Sistema
- Tipos
 - Producto vs Proceso
 - Funcionales vs No Funcionales
 - Niveles vs Tipos
- Proceso
 - Características y Objetivos
 - Actores
 - Desarrollo vs Gestión
- Elicitación
 - Fuentes
 - Técnicas de Captura
- Análisis
 - Clasificación
 - Modelado Conceptual
 - Localización
 - Negociación
- Especificación
 - Documento
 - Estándar IEEE 830
- Validación
 - Revisiones
 - Prototipado
- Gestión de Requisitos
 - Atributos
 - Trazabilidad



Bibliografía

- Básica
 - IEEE Computer Society (2004)
 - SWEBOK - Guide to the Software Engineering Body of Knowledge, 2004 Version.
 - Capítulo 2.
 - <http://www.swebok.org/>
 - Caps. 6 y 7 del libro de Sommerville (2005).
 - IEEE Std 830 (1998)
 - IEEE Recommended Practice for Software Requirements Specifications.
 - Disponible versión en español.
- Complementaria
 - Cap. 7 del libro de Pressman (2005).
 - Cap. 4 del libro de Pfleeger (2002).



Definiciones

- **Requisito:**
 - Propiedad que debe ser exhibida por un software para resolver un problema particular (SWEBOK).
 - Condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado.
- **Especificación de Requisitos Software (ERS, SRS):**
 - Documento formal de los Requisitos del Sistema
 - *SRS – Software Requirements Specification*



Definiciones

- **Ingeniería de Requisitos:**
 - “Conjunto de actividades para descubrir, documentar y mantener un conjunto de requisitos”
 - Establecer los servicios que el cliente requiere de un sistema y las restricciones bajo las cuales opera y es desarrollado.
- **Proceso de Ingeniería de Requisitos:**
 - “Conjunto estructurado de actividades de cuya ejecución se obtiene, valida y mantiene el documento de requisitos del sistema”
- **Gestión de Requisitos:**
 - Actividad para gestionar los cambios en los requisitos de un sistema



Características

- Los requisitos son una etapa clave en el ciclo de vida:
 - Su coste es alrededor de 10-15% del coste total del proyecto.
 - Un error en los requisitos puede ser hasta 100 veces más costoso que un error en el código.
 - Una equivocación en la etapa de requisitos se arrastra en las demás fases del ciclo de vida
 - Los procesos/sistemas complejos implican miles de requisitos
 - Necesidad de gestión y soporte automatizado



Características

- Los requisitos de un software suelen ser una **combinación compleja de los requisitos de diferentes personas** en diferentes niveles de una organización **y del entorno** en el cual operará el software.
- Es fundamental que un requisito sea **verificable**.
- Otros atributos que les caracterizan son:
 - Prioridad
 - Identificador único.
 - x.
- Los requisitos deben ser lo más **claros y no ambiguos** que se pueda, y **cuantificables** (si es posible).



Dificultades

- **Imprecisión**

- Es fuente de problemas.
- Los requisitos ambiguos pueden ser interpretados de diferentes formas por desarrolladores y usuarios.

- **Ejemplo:**

- El software incluirá **visores adecuados**
- Intención del usuario: deberá incluir un visor especial para cada tipo de documento manejado.
- Interpretación del desarrollador: incluir un visor de texto que muestre el contenido de los documentos.



Dificultades

- En principio, los requisitos deben ser:

- **Completos**

- Deben incluir descripciones de todos los servicios requeridos.

- **Consistentes**

- No deben existir conflictos o contradicciones en las descripciones.

- En la práctica este objetivo es **imposible de cumplir al 100%** en un documento de requisitos de complejidad media o grande.



Dificultades

- Es frecuente que no esté clara la **frontera entre Requisitos y Diseño**.
- En principio, los requisitos indican lo que el sistema debe hacer y el diseño describe cómo lo hace.
- En la práctica, ambas etapas son **inseparables** porque
 - La arquitectura del sistema puede ser diseñada para estructurar los requisitos,
 - El sistema interacciona con otros sistemas que generan requisitos de diseño del software
 - El uso de un diseño específico puede ser un requisito de dominio.

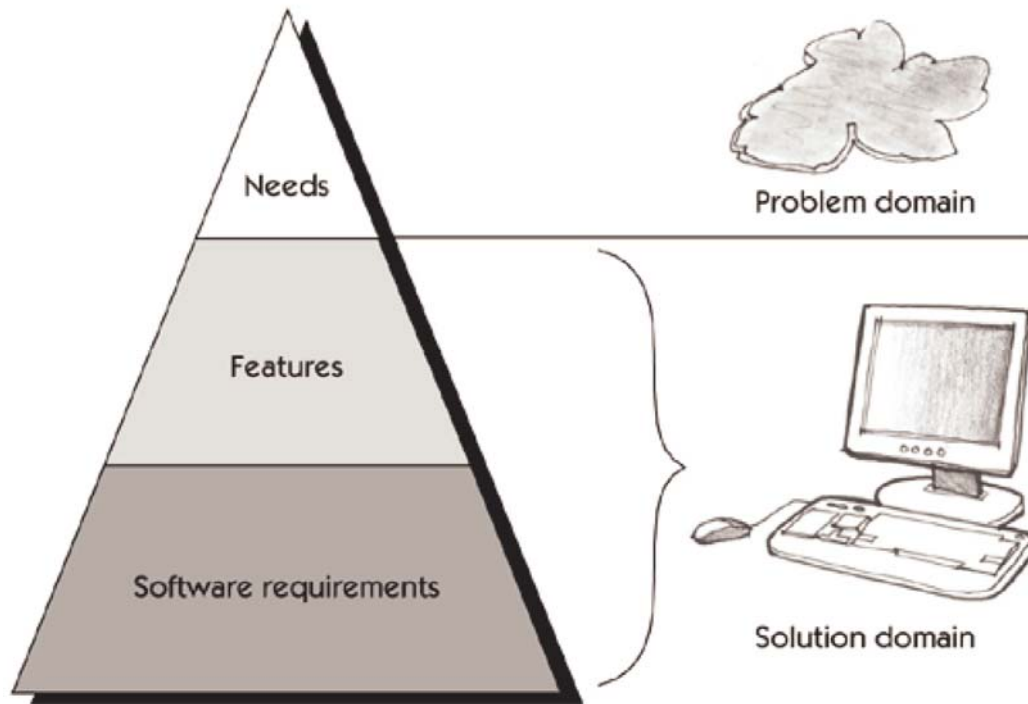


Niveles

- Los requisitos se pueden definir a distintos **niveles** de abstracción o detalle:
 - **Usuario** (*needs*)
 - **Sistema** (*features*)
 - **Software** (*SW equirements*)



Niveles



Francisco Ruiz - IS1

3.15



Niveles

- **De Usuario**
 - Declaraciones en lenguaje natural y, quizás, tablas y diagramas, de los servicios que el sistema provee y sus restricciones operacionales.
- **De Sistema**
 - Un documento estructurado con las descripciones detalladas de las funciones, servicios y restricciones operacionales del sistema.
 - Define lo que deberá ser implementado.
 - Puede ser parte de un contrato con el cliente.
- **De Software**
 - Declaraciones detalladas de diseño e implementación del software.

Francisco Ruiz - IS1

3.16



Niveles

- EJEMPLOS

- De Usuario

- El software debe proveer un medio de representar y acceder a los ficheros externos creados por otras herramientas.

- De Sistema

- El usuario debe poder elegir el tipo de fichero externo.
- Cada tipo de fichero externo debe poder tener asociada una herramienta externa para editarlo y mostrarlo.
- Cada tipo de fichero externo tiene un icono asociado.

- De Software

- Los iconos de los tipos de fichero se guardan en archivos JPG.



Niveles

- Cada nivel está dirigido a una clase diferente de **"lector"**:

- **Usuario** → Gestores clientes, Usuarios Finales, Ingenieros Clientes, Gestores Contratistas, Arquitectos del Sistema

- **Sistema** → Usuarios Finales, Ingenieros Clientes, Arquitectos del Sistema, Desarrolladores SW

- **Software** → Ingenieros Clientes (quizás), Arquitectos del Sistema, Desarrolladores SW



Niveles – Usuario

- Los **Requisitos de Usuario** describen requisitos de manera comprensible por los usuarios sin conocimientos técnicos detallados.
- Se definen mediante **lenguaje natural**, tablas y diagramas.
- **Problemas** con el lenguaje natural:
 - **Falta de claridad**: es difícil conjuntar precisión y facilidad de lectura.
 - **Confusión** entre requisitos: Requisitos funcionales y no funcionales tienden a estar mezclados.
 - **Amalgama** de requisitos: Varios requisitos diferentes pueden ser expresados juntos.



Niveles – Usuario

- Los **Requisitos de Usuario** pueden responder a varios orígenes:
 - Dominio del problema (**Requisitos de Dominio**)
 - Intereses de la organización (**Requisitos de Negocio u Organizacionales**)
 - Necesidades de los usuarios finales del software.



Niveles - Dominio

- Los **Requisitos de Dominio** son requisitos de usuario que se derivan del dominio de aplicación.
 - Describen características y necesidades propias del dominio (comunes a todas las organizaciones de ese sector).
- Ejemplos del Dominio de Bibliotecas:
 - El interfaz de usuario será común para acceder a todas las bases de datos electrónicas y estará basado en la norma XYZ.
 - Dependiendo de las restricciones de copyright, los documentos podrán ser impresos o sólo visualizados por los usuarios.



Niveles - Dominio

- Las principales problemas con los **Requisitos de Dominio** son:
 - **Comprensibilidad:**
 - Los requisitos utilizan el lenguaje y vocabulario habitual en el dominio de aplicación.
 - Los ingenieros software que desarrollan el sistema no conocen dicho lenguaje.
 - **Sobreentendidos:**
 - Los expertos en el área del dominio la conocen tan bien que inconscientemente no hacen explícitos todos los requisitos del dominio.



Niveles - Sistema

- Los **Requisitos de Sistema** son especificaciones más detalladas que los requisitos de usuario de las funciones, servicios y restricciones del sistema.
- Son la base para el diseño del sistema.
- Pueden ser incorporados en el contrato.
- Pueden ser definidos o ilustrados usando modelos del sistema (conceptuales, etc.).



Tipos

- De manera ortogonal a los niveles anteriores, según su naturaleza, los requisitos software se pueden clasificar en base a dos criterios
 - Producto vs Proceso
 - Funcionales vs No Funcionales



Tipos – Producto vs Proceso

- **De Producto**

- Son los requisitos propiamente en el software a desarrollar.
- Ejemplo:
 - *El software verificará que un estudiante ha superado todos los pre-requisitos antes de dejarle matricularse en una asignatura.*

- **De Proceso**

- Esencialmente son restricciones sobre la manera en que se desarrolla el software.
- Ejemplos:
 - *El software se escribirá en Ada.*
 - *Se utilizará METRICA 3 como metodología.*



Tipos – Funcionales vs No Funcionales

- Este es el criterio de clasificación más habitual
- **Funcionales:** [capacidades]
 - Describen las funciones que lleva a cabo el software, cómo debe reaccionar ante ciertas entradas y cómo debe comportarse en situaciones particulares.
- **No Funcionales:** [restricciones, requisitos de calidad]
 - Restricciones sobre las funciones o servicios ofrecidos por el sistema.
- **De Dominio:**
 - Proviene del dominio de aplicación del sistema, reflejando sus características.



Tipos – Funcionales vs No Funcionales

- **Funcionales**

- Describen la funcionalidad o servicios del software.
- Dependen de:
 - El tipo de software
 - Las expectativas de los usuarios, y
 - EL tipo de sistema donde el software se usará.

- **Los Requisitos Funcionales**

- **De Usuario** son descripciones de alto nivel de lo que el sistema debe hacer, mientras que los
- **De Sistema** describen los servicios con más detalle.



Tipos – Funcionales vs No Funcionales

- **Ejemplos de Requisitos Funcionales**

- El usuario debe ser capaz de buscar entre todo el conjunto de bases de datos o de seleccionar un subconjunto de ellas.
- Cada pedido tiene un identificador único (pedido_id).



Tipos – Funcionales vs No Funcionales

● Requisitos No Funcionales

- Definen cualidades o atributos globales del sistema, o
- Establecen restricciones sobre el producto desarrollado, el proceso de desarrollo o externas

- No están generalmente relacionados con la funcionalidad del sistema

- **No hay una distinción clara** entre requisitos funcionales y no funcionales
- Expresar un requisito como funcional o no funcional depende del nivel de detalle requerido en el documento de requisitos



Tipos – Funcionales vs No Funcionales

● Ejemplos:

- “El sistema asegurará que los datos son protegidos de accesos no autorizados”
 - Generalmente sería considerado como un requisito **no funcional** porque el requisito no se refiere a una funcionalidad específica del sistema

- “El sistema incluirá un procedimiento de autorización de usuario en el que los usuarios se identifican mediante un nombre de usuario y una contraseña. Sólo los usuarios autorizados pueden acceder a los datos del sistema”
 - El requisito especifica con más detalle una función que debe ser incorporada en el sistema → Requisito **Funcional**



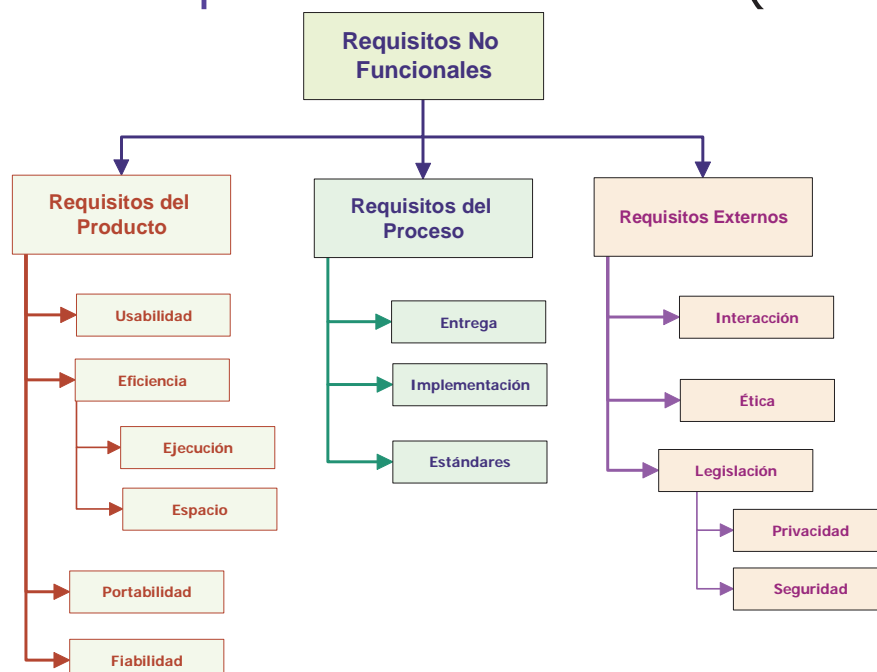
Tipos – Funcionales vs No Funcionales

- **Subtipos de Requisitos No Funcionales (IEEE 830-1998):**
 - Interfaces Externas
 - De Rendimiento
 - Lógicos de Base de Datos
 - Restricciones de Diseño
 - Cumplimiento de Estándares
 - De Atributos del Software:
 - De Fiabilidad
 - De Disponibilidad
 - De Seguridad
 - De Mantenibilidad
 - De Portabilidad



Tipos – Funcionales vs No Funcionales

- **Subtipos de Requisitos No Funcionales (Sommerville, 2005):**





Tipos – Funcionales vs No Funcionales

- **Subtipos de Requisitos No Funcionales**
(Sommerville, 2005):

- **De Producto**

- Especifican comportamiento del producto (velocidad de ejecución, fiabilidad, etc.)

- **Organizacionales**

- Son una consecuencia de las políticas y procedimientos de la organización (estándares usados, requisitos de implementación, etc.).

- **Externos**

- Están relacionados con factores externos al sistema y su proceso de desarrollo (requisitos de interoperabilidad, normas legales, etc.).



Tipos – Funcionales vs No Funcionales

- **Ejemplos de Requisitos No Funcionales**

- **De Producto**

- El interfaz de usuario será implementado como HTML simple sin frames ni applets Java.

- **Organizacionales**

- El proceso de desarrollo y los entregables serán conformes a lo establecido en METRICA v3.

- **Externos**

- El sistema no registrará ninguna información personal de los clientes salvo su nombre y número de referencia respetando lo establecido en la LOPD.



Tipos – Funcionales vs No Funcionales

● Requisitos No Funcionales de Producto

- Especifican Restricciones en la Ejecución del Sistema
- Parte de estos requisitos se pueden formular de forma precisa, de forma que puedan ser fácilmente cuantificables:
 - Rendimiento
 - Capacidad
- Otros son más difíciles de cuantificar y por lo tanto se expresan generalmente de modo informal
 - Usabilidad



Tipos – Funcionales vs No Funcionales

● Requisitos No Funcionales de Producto

● EJEMPLOS

- **Fiabilidad:**
 - "El servicio A del Sistema debe tener una disponibilidad de 99 %"
- **Rendimiento:**
 - "El Sistema Y procesará un mínimo de 8 transacciones por segundo"
- **Espacio:**
 - "El código ejecutable del Sistema Z estará limitado a 512 Kb"
- **Portabilidad:**
 - "El Sistema se desarrollará para las plataformas PC y Macintosh"
- **Seguridad:**
 - "El Sistema encriptará todas las comunicaciones externas usando el algoritmo RSA"



Tipos – Funcionales vs No Funcionales

● Requisitos No Funcionales de Producto

■ Conflictos

- Un requisito de rendimiento puede estar en conflicto con requisitos de seguridad o fiabilidad
- Un requisito de utilización de espacio puede estar en contradicción con un requisito que exige el uso de un compilador que no genera un código muy compacto

■ Los conflictos se resuelven en función de:

- El Nivel de Importancia de cada Requisito
- Consecuencias sobre el cambio en el Requisito
- Los objetivos de negocio



Tipos – Funcionales vs No Funcionales

● Requisitos No Funcionales de Proceso

■ Son **restricciones** en el **proceso** de desarrollo del Sistema

- Estándares y Métodos de Desarrollo
- Herramientas a usar
- Producción de Informes

■ Ejemplos:

- El proceso de desarrollo debe ser conforme con ISO 9003
- El Sistema debe desarrollarse usando la Herramienta Visual Paradigm
- Los informes de Gestión sobre el esfuerzo dedicado a cada componente se deben generar cada dos semanas



Tipos – Funcionales vs No Funcionales

- **Requisitos No Funcionales Externos**

- Están relacionados con el **Entorno**
- Se pueden aplicar a Producto y Proceso
- Ejemplos:
 - Sistema Médico:
 - “El responsable de la protección de datos de la organización debe certificar que todos los datos se mantienen de acuerdo a la legislación vigente”
 - Sistema Protección de Trenes:
 - “El tiempo requerido para detener completamente el tren se basa en la siguiente ecuación de deceleración: $\gamma_{\text{tren}} = \gamma_{\text{control}} + \gamma_{\text{gradiente}}$ ”



Tipos – Funcionales vs No Funcionales

- Los Requisitos **No Funcionales** suelen ser **difíciles de expresar** porque
 - Hay restricciones relacionadas con soluciones de diseño que son desconocidas en la etapa de requisitos
 - Hay restricciones altamente subjetivas que sólo se pueden determinar mediante evaluaciones empíricas complejas
 - Tienen a estar relacionados con uno o más requisitos funcionales
 - Tienen al conflicto o las contradicciones entre ellos
 - No existen recomendaciones y reglas “universalmente aceptadas” para determinar cuando se satisfacen de forma óptima ciertos requisitos no funcionales



Tipos – Funcionales vs No Funcionales

- Los Requisitos **No Funcionales** se suelen extraer de **restricciones de los interesados** (*stakeholders*).
 - Ejemplos:
 - Objetivos de Negocio Críticos
 - Características esenciales del Sistema (ej: Seguridad)
 - Seguridad Física, Rendimiento, Mantenibilidad, etc..
 - Estos intereses o preocupaciones suelen estar definidos de forma "vaga":



Tipos – Funcionales vs No Funcionales

- Los Requisitos **No Funcionales** son especialmente importantes en **Sistemas Críticos**
 - Es decir, aquellos cuyos fallos causan un daño significativo de tipo económico, físico o humano a las organizaciones o personas.
 - Ejemplos:
 - Negocio: Sistema reserva aerolínea
 - Misión: Sistema de control de órbita de un satélite
 - Seguridad Física o Humana: Sistema de Control Central Nuclear, Sistema médico de control de radiación para tratamiento de Cáncer
- Los principales RNF en estos sistemas son:
 - Fiabilidad, Rendimiento, Seguridad, Usabilidad, Seguridad Física



Tipos – De Sistema vs de Software

• De Sistema

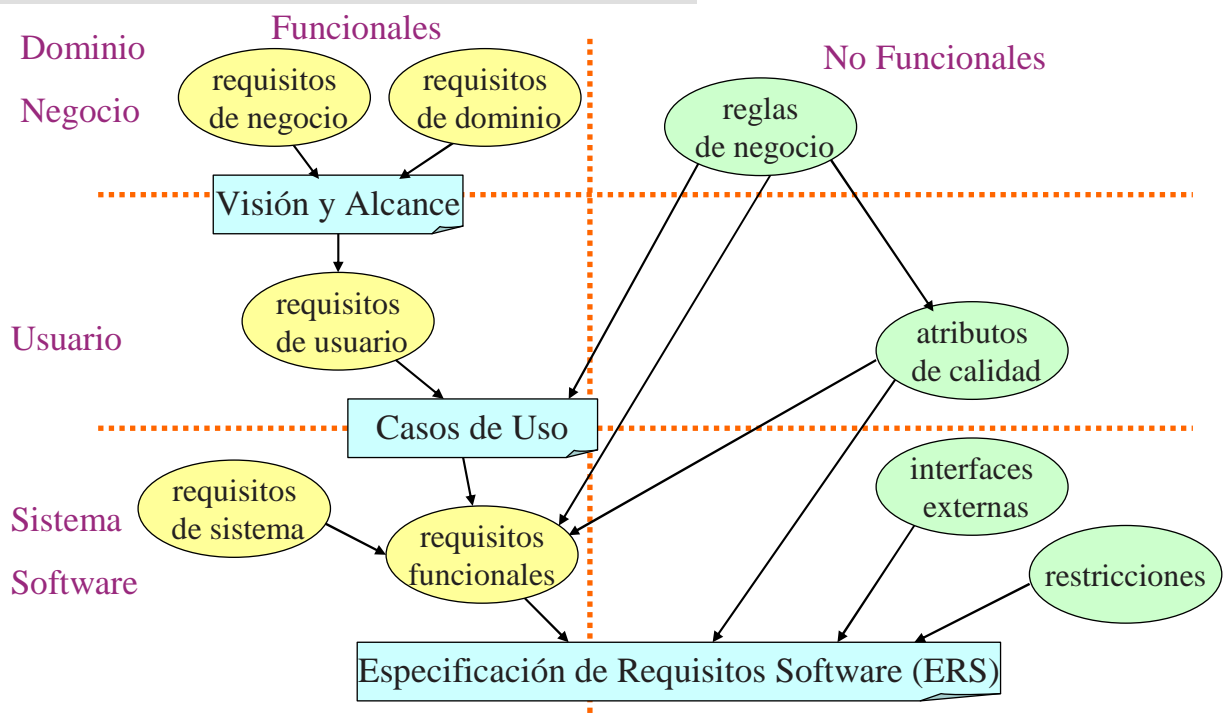
- Requisitos para el sistema en su conjunto.
- Un **Sistema** engloba, entre otros, elementos de tipo hardware, software, firmware, personas, información, técnicas, facilidades y servicios.
- Estos requisitos engloban a los requisitos de diferentes interesados: usuarios, clientes, autoridades, etc.

• De Software

- Solo se refieren al software, pero en un sistema completo se derivan de los requisitos del sistema.



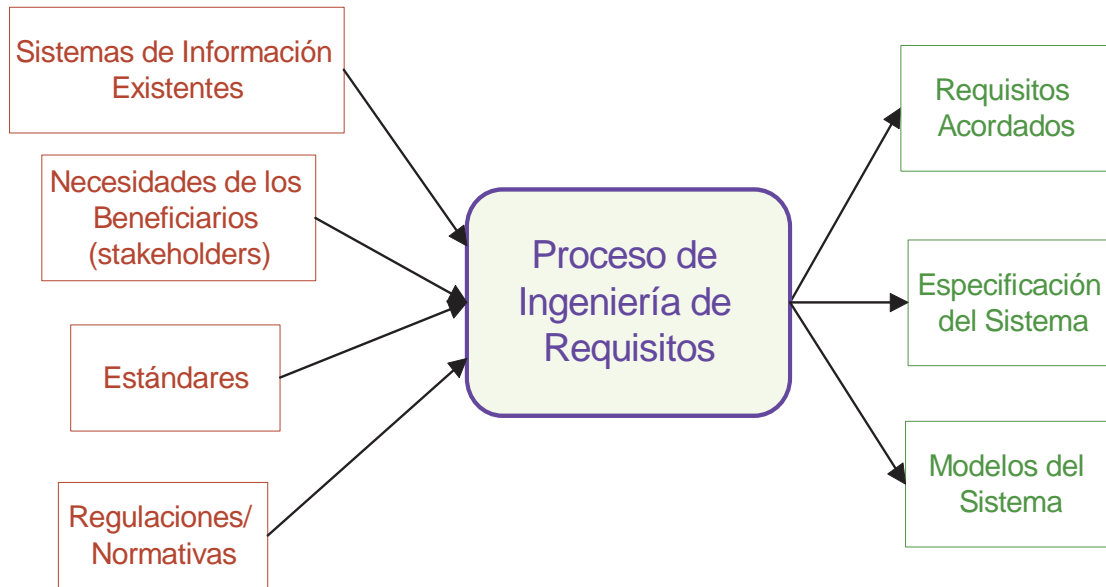
Tipos – Niveles vs Tipos



Niveles, tipos y documentos de requisitos



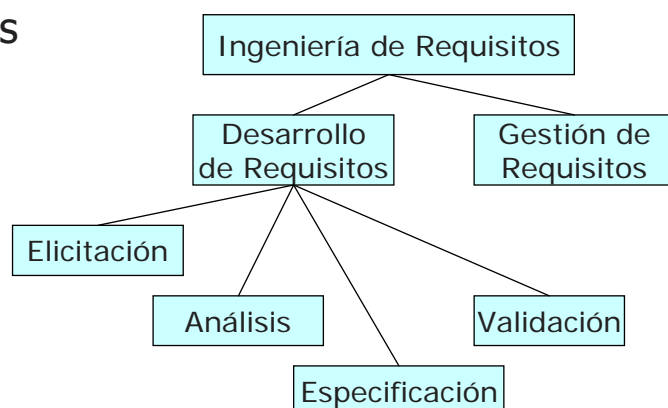
Proceso – Características y Objetivos



Proceso – Características y Objetivos

- **Cómo se hace varia** en función de la metodología seguida, el dominio de aplicación, las personas participantes y la organización.
- Pero de forma general, se distinguen las siguientes **actividades**:

- Desarrollo de Requisitos
 - Elicitación
 - Análisis
 - Especificación
 - Validación
- Gestión de Requisitos



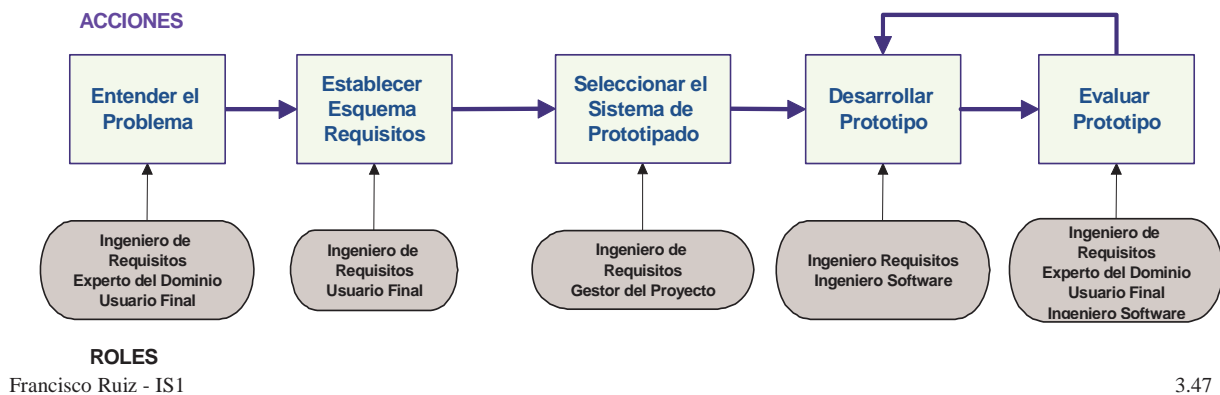


Proceso – Actores

- **Factor Humano:**

- El Proceso de Ingeniería de Requisitos está dominado por factores humanos, sociales y organizacionales
 - Implican diferentes *stakeholders* de diferentes procedencias y perfiles (técnicos, no técnicos de distintas disciplinas) y con distintos objetivos

- **Ejemplo de Modelo de Roles:**



Proceso – Actores

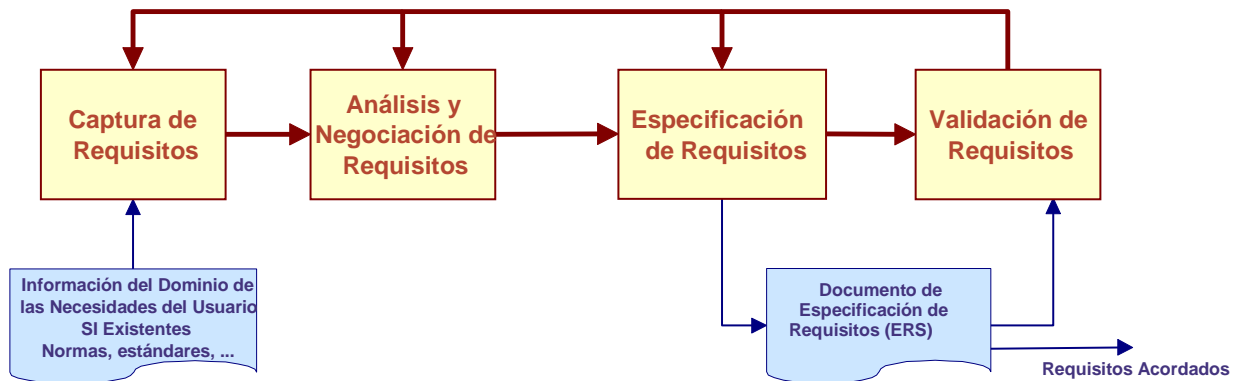
- **Principales interesados (*stakeholders*):**

- **Usuarios:** los que utilizan el sistema.
- **Clientes:** Los propietarios del software o que representan el mercado de destino.
- **Analistas de mercado:** Identifican lo que el mercado demanda.
- **Reguladores:** Aseguran el cumplimiento con las normas y leyes establecidas.
- **Ingenieros de software**



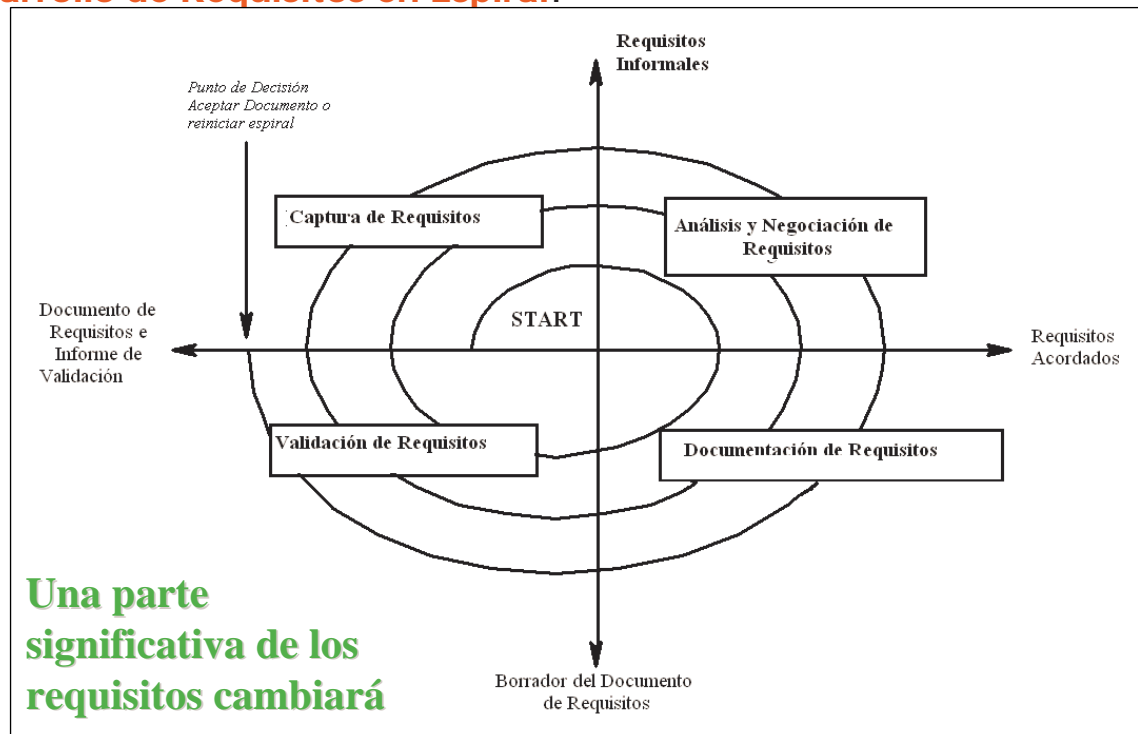
Proceso – Desarrollo vs Gestión

- **Desarrollo de Requisitos:**



Proceso – Desarrollo vs Gestión

- **Desarrollo de Requisitos en Espiral:**





Proceso – Desarrollo vs Gestión

- **Desarrollo vs Gestión de Requisitos:**



Elicitación

- Es la primera actividad a realizar.
- Para llevarla a cabo el ingeniero debe aprender:
 - De donde vienen los requisitos, y
 - Cómo puede recopilarlos.
- Es una actividad fundamentalmente humana, en la que se identifican los interesados (stakeholders) y se establecen relaciones entre el equipo de desarrollo y los clientes.



Elicitación

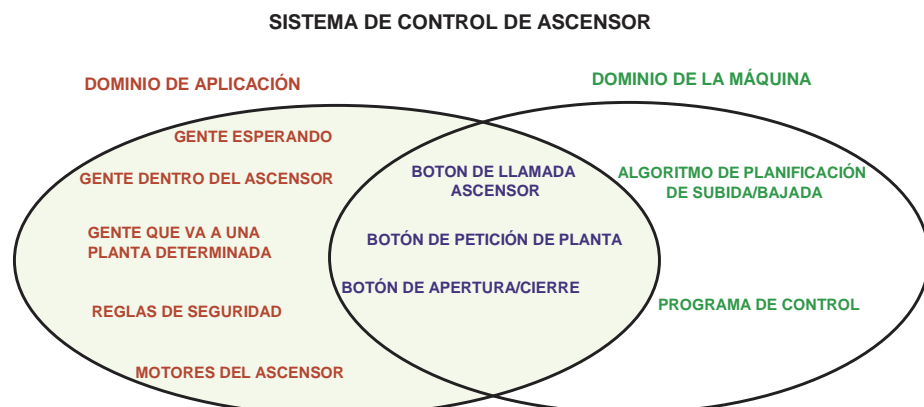
- También llamada **Captura / Descubrimiento de Requisitos**
- **Supone el Entendimiento de:**
 - **El Problema**
 - Los detalles del problema específico del cliente donde se aplicará el sistema
 - **El Negocio**
 - Cómo los sistemas interactúan y contribuyen a las metas del negocio
 - **Las Necesidades y Restricciones de los Beneficiarios del Sistema**
 - Necesidades específicas de la gente que requiere soporte del sistema para su trabajo



Elicitación



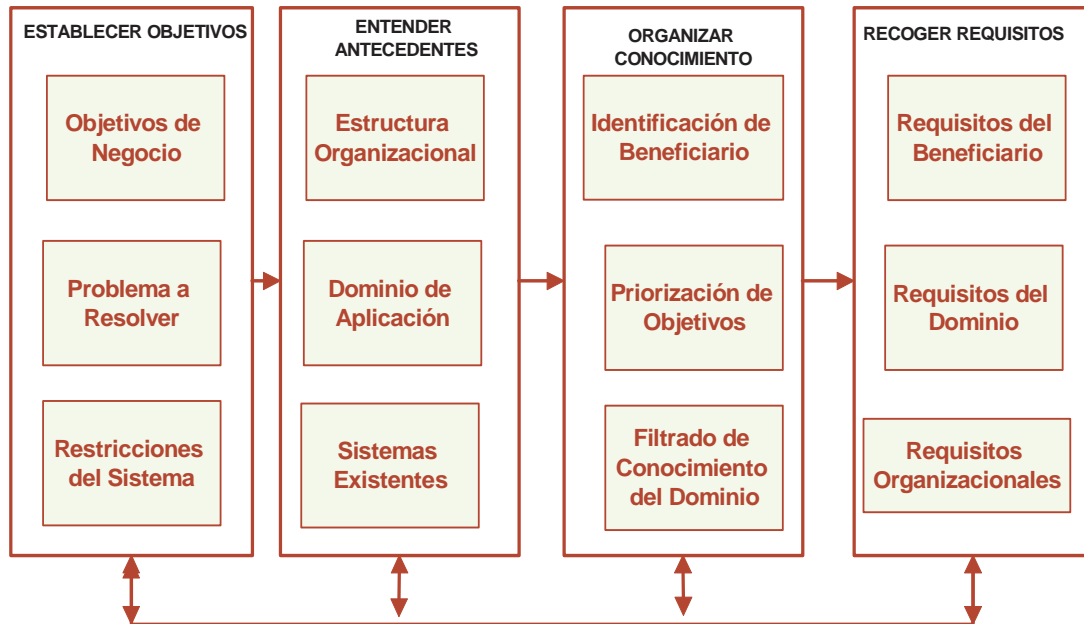
EJEMPLO:





Elicitación

- Tipos de esfuerzos / Tareas que se realizan:



Elicitación - Fuentes

- Los requisitos se originan a partir de:
 - **Objetivos** (intereses de negocio, factores críticos de éxito): proveen la motivación para realizar el software, pero suelen ser ambiguos.
 - **Conocimiento del Dominio:** Permite al ingeniero inferir conocimiento tácito que los *stakeholders* no articulan.
 - **Interesados** (stakeholders): El ingeniero necesita identificar, representar y gestionar los puntos de vista de los diferentes tipos de interesados.
 - **Entorno operacional:** el entorno en el que se ejecutará el software.
 - **Entorno organizacional:** El ingeniero debe ser sensible a la estructura, cultura y políticas de la organización, así como a los procesos de negocio a los que dará soporte el software.



Elicitación - Técnicas

- Las **Técnicas de Captura para Requisitos** sirven para
 - Conseguir que los interesados humanos articulen sus requisitos, y
 - Recopilar el conocimiento sobre los requisitos del sistema.
- Este Conocimiento debe estar estructurado:
 - Partición → Agregando conocimiento relacionado
 - Abstracción → Reconociendo generalidades
 - Proyección → Organizándolo de acuerdo a una perspectiva

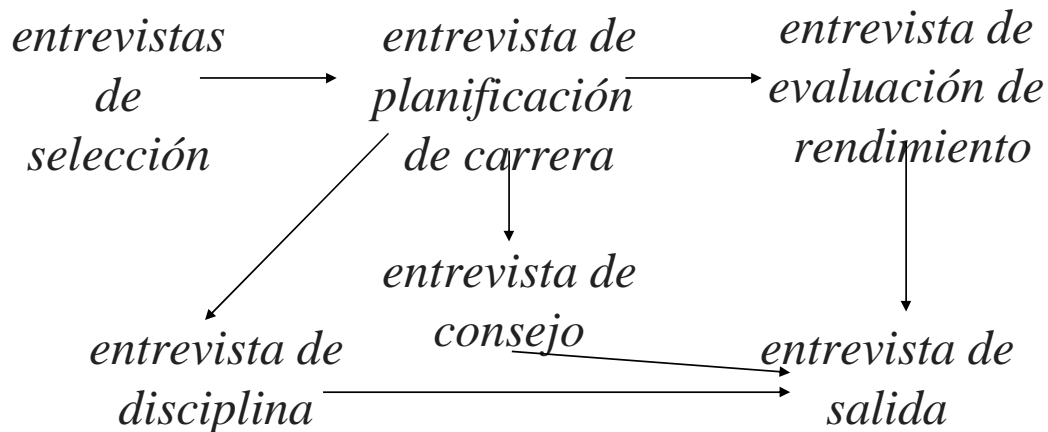


Elicitación - Técnicas

- Las principales **Técnicas para Captura de Requisitos** son:
 - Entrevistas
 - Escenarios
 - Observación
 - Reutilización de Requisitos
 - Prototipado

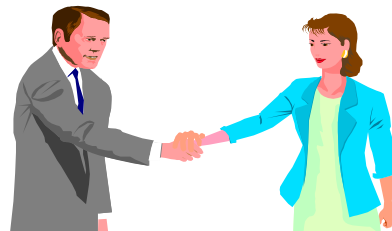


- **Entrevista:** Intento sistemático de recoger información de otra persona a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada.



- El **entrevistado** puede presentar:

- Pasividad, inhibición
- No aceptación
- Rechazo
- Agresividad



“relación asimétrica, dinámica y única”

- El **entrevistador** debe poseer:

- Ciertas cualidades personales
- Conocimiento de técnicas
- Actitud adecuada
- Experiencia práctica



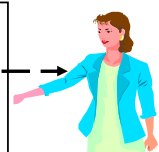
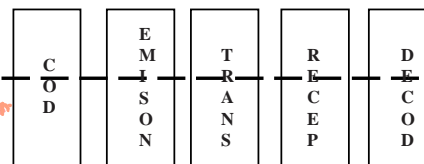
• **Cualidades personales de un buen entrevistador:**

- Saber observar y escuchar
- Poseer madurez
- Ser objetivo e imparcial
- No ser autoritario
- Capacidad de empatía
- Comprensión
- Ser cordial y accesible
- Respetar la intimidad
- Ser sincero, paciente, sereno
- Ser prudente



• **Problemas de las Entrevistas:**

- Discrepancia de objetivos
- Barreras de comunicación
 - Oír lo que queremos
 - Pasar por alto ideas contrarias
 - Prejuicios sobre el emisor
 - Diferente significado de las palabras
 - Comunicación no verbal
 - Emociones
 - Ruido
 - Distancia
- Mantenimiento de la motivación



■ **Cómo eliminar las barreras**

- Adaptarse al mundo del receptor
- Utilizar el diálogo
- Servirse de la comunicación directa
- Insistir varias veces
- Utilizar lenguaje sencillo y directo
- Utilizar vías distintas
- Reducir las distancias



- Factores a considerar para **detectar problemas durante las entrevistas**:
 - Comunicación no verbal
 - Contacto corporal.
 - Proximidad física. Orientación. Postura
 - Ademanes. Cabeza. Expresión facial. Ojos.
 - Apariencia.
 - Aspectos del lenguaje.
 - "Escuchar y responder"
 - Vocabulario
 - Expresión verbal



- **Fases** de una **entrevista**:
 - Preparación
 - Conducción/Realización
 - Análisis
- **Preparación**:
 - Investigar la situación
 - Identificar las personas a entrevistar
 - Preparación del objetivo y contenido
 - Planificar lugar y hora





● **Conducción/Realización:**

- Apertura
- Desarrollo
 - Métodos directos:
 - Preguntas abiertas, directas, cerradas, sí/no
 - Sondeo
 - Métodos indirectos:
 - Utilizar palabras y frases apropiadas
 - Asentir y dar muestras de escuchar
 - Repetir las respuestas dadas; resumir
 - Reflejar ideas, sentimientos
 - Pausas
 - Tiempos: entrevistado 80%
- Terminación

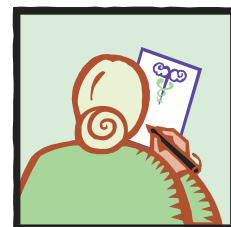


Cumplir las reglas del protocolo



● **Análisis:**

- Pasar a limpio las notas
- Reorganizar la información y contrastarla
- Evaluar la entrevista → Mejora de aspectos
 - Conclusiones
 - Fracaso
 - Entrevista deficiente
 - Mala relación
 - Acusaciones
 - Éxito
 - Planificación
 - Flexibilidad
 - Personalidad
 - “No existe la entrevista ideal”

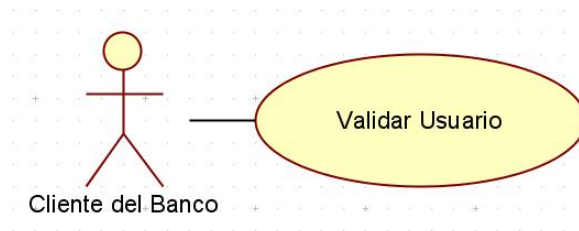




- Los **Escenarios** son ejemplos de la vida real de cómo puede ser usado un sistema.
- Deben incluir
 - Una descripción de la situación de partida;
 - Una descripción del flujo normal de eventos;
 - Una descripción de lo que puede ir mal;
 - Información sobre otras actividades concurrentes;
 - Una descripción del estado cuando el escenario acaba.
- EL tipo de escenario más usado son los **Casos de Uso**.



- **ESCENARIO**
 - **Ejemplo:**



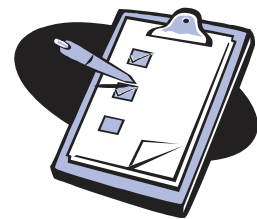
- El **Sistema** pide al **Cliente** un número de identificación personal (PIN)
- El **Cliente** introduce el PIN a través del teclado y acepta la entrada
- El **Sistema** comprueba si el PIN es válido
- El **Sistema** acepta la entrada y así finaliza el escenario



- Los **Prototipos** son una herramienta para clarificar requisitos poco claros.
- Existe un amplio rango de técnicas de prototipado para requisitos:
 - Desde diseños de pantallas dibujados en papel,
 - Hasta versiones beta de prueba de productos software complejos.
- A la vez que para elicitar-clarificar requisitos, sirven también en su validación.



- **Prototipo:**
 - "No sé exactamente que quiero, pero lo sabré cuando lo vea".
 - **Situaciones de Utilidad:**
 - Área de aplicación no definida
 - Coste alto de rechazo de la aplicación
 - Necesidad de evaluación del impacto del sistema
 - **Razones para emplearlo:**
 - Prototipado de la interfaz de usuario
 - Modelos de rendimiento
 - Prototipado evolutivo
 - Calidad del prototipo: Ser construido más rápidamente que la aplicación.





- Las **Reuniones** sirven para intentar alcanzar un **efecto acumulativo** o sinérgico de manera que un grupo de personas puede profundizar más y mejor en sus requisitos que trabajando de forma individual.
- **Ventajas:**
 - Enfrentar al grupo de interesados con las contradicciones y conflictos en los requisitos que han propuesto.
- **Peligros:**
 - Preponderancia de la opinión de los poderosos, aún sea errónea.



- Mediante **Observaciones** los ingenieros software pueden aprender sobre las tareas de los usuarios haciendo ellos mismos una inmersión en el entorno real de los usuarios y observando cómo cada usuario interactúa con su software y con otros usuarios.
- **Desventajas:**
 - Coste relativamente alto.
- **Ventajas:**
 - Ilustran el porqué de que muchos usuarios no pueden describir fácilmente sus requisitos: sus tareas y procesos de negocio son demasiado complejos y sutiles.



Análisis

- El **Análisis de Requisitos** se refiere al esfuerzo de analizar los requisitos para
 - Detectar y resolver conflictos entre requisitos
 - Descubrir los límites del software y cómo debe interactuar con su entorno
 - Elaborar los requisitos de sistema de cara a la posterior obtención de los requisitos software
- Para ello se recomienda realizar **4 tareas** con los requisitos
 - Clasificación
 - Modelado Conceptual
 - Localización
 - Negociación



Análisis

- **Técnicas para Análisis de Requisitos**
 - **Listas de Comprobación (CheckList)**
 - Diseño Prematuro
 - Requisitos Combinados
 - Requisitos Innecesarios
 - Uso de HW no estándar
 - Conformidad con Objetivos de Negocio
 - Ambigüedad
 - Realismo (viabilidad)
 - Facilidad de Prueba
 - **Matriz de Interacción**

Requirement	R1	R2	R3	R4	R5	R6
R1	0	0	1000	0	1	1
R2	0	0	0	0	0	0
R3	1000	0	0	1000	0	1000
R4	0	0	1000	0	1	1
R5	1	0	0	1	0	0
R6	1	0	1000	1	0	0



Análisis - Clasificación

- Según la naturaleza del proyecto, los requisitos se pueden **clasificar** por varios criterios:
 - Funcionales vs No Funcionales
 - Origen
 - Requisito de alto nivel vs establecido directamente sobre el software
 - De producto vs de Proceso.
 - Prioridad
 - Alcance
 - Componentes o piezas software afectados.
 - Algunos no funcionales tienen alcance global.
 - Volátiles vs Estables.



Análisis – Modelado Conceptual

- Como en muchas otras situaciones en ingeniería del software, el **modelado conceptual** de los requisitos busca la comprensión del problema antes de iniciar el diseño de la solución.
- Principales Técnicas de Modelado de Requisitos:
 - Flujos de Datos y de Control (DFDs)
 - Modelos de Estados (máquinas de estados)
 - Diagramas de Eventos
 - Diagramas de Interacción de usuarios
 - Modelos de Objetos y de Datos

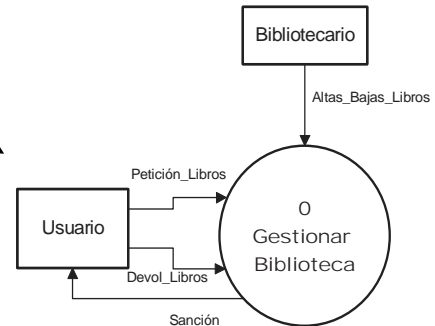


Análisis – Modelado Conceptual

- **Técnicas de Modelado Conceptual de Requisitos**

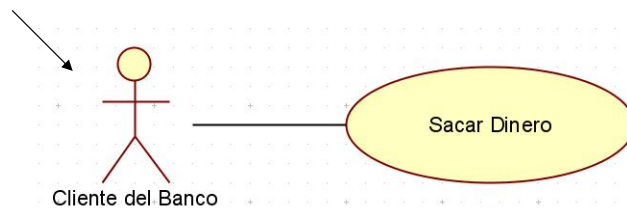
- Modelos de **Flujo de Datos (DFD)**

- Entidad / Interrelación



- Orientación a Objetos

- Escenarios (**Casos de Uso**)



Análisis – Localización

- La **localización** de requisitos consiste en **asignar a componentes** la responsabilidad de satisfacer los requisitos.
- Para ello es necesario realizar antes el **diseño arquitectural** del software, ya que de él se derivan los componentes que tendrá el sistema.
- Una vez asignados requisitos a componentes es posible realizar un análisis más detallado de cada componente para descubrir nuevos requisitos sobre cómo debe interactuar con los demás componentes.



Análisis – Negociación

- También llamado **resolución de conflictos**.
- Consiste en **resolver los problemas**
 - Cuando dos interesados requieren características mutuamente incompatibles,
 - Entre requisitos y recursos,
 - Entre requisitos funcionales y no funcionales.
- A ser posible, las decisiones deben estar basadas en el consenso.
- También puede tener que llevarse a cabo durante la actividad de validación.



Análisis – Negociación

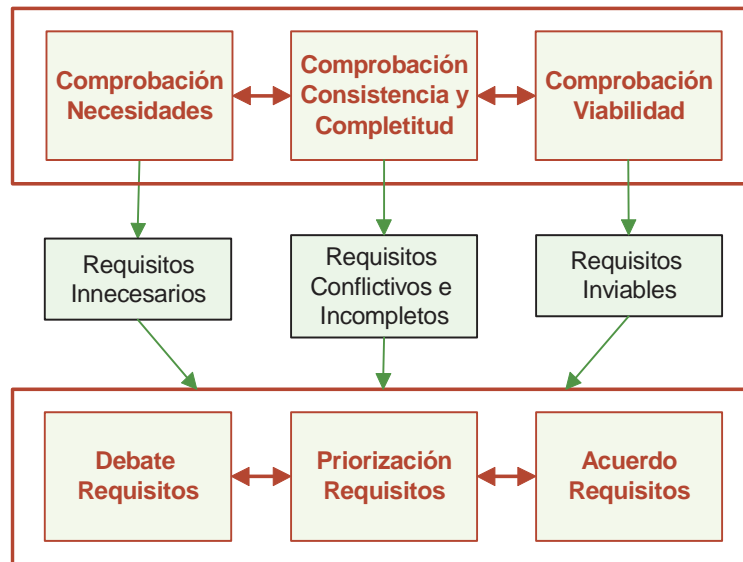
- **Negociación de Requisitos**
 - Los conflictos son prácticamente inevitables cuando intervienen diversos interesados
 - Conflicto \neq Fallo
- Para resolver un conflicto se recomienda seguir tres pasos:
 1. Información. Se explica la naturaleza de los problemas asociados con un requisito
 2. Debate. Cada beneficiario expone cómo se podría resolver el problema
 3. Resolución. Se llega a un acuerdo sobre las acciones que afectan a cada requisito (borrado, modificación o captura de más detalles)



Análisis – Negociación

- La negociación es el paso final en el análisis de los requisitos innecesarios, conflictivos, incompletos o inviables

ANÁLISIS DE REQUISITOS



NEGOCIACIÓN DE REQUISITOS



Especificación

- La **Especificación de Requisitos Software** se refiere a la producción de un documento
 - Que puede ser revisado, evaluado y aprobado.

Las palabras se las lleva el viento

- En proyectos que no incluyen sólo software (sistemas informáticos), se distinguen otros dos documentos previos:
 - **Definición del Sistema.** Define los requisitos de los usuarios desde la perspectiva del dominio de aplicación.
 - **Especificación de Requisitos del Sistema.** Incluye los requisitos del sistema, de los cuales se pueden derivar los requisitos software.



Especificación - Documento

- **Especificación de Requisitos Software ERS**
 - **Software Requirements Specification SRS**
 - **Especificación:** Documento que define, de forma completa, precisa y verificable, los requisitos, el diseño, el comportamiento u otras características de un sistema o componente de un sistema.
 - **Software:** Conjunto de programas, procedimientos y documentación asociada a la operación de un sistema informático.



Especificación - Documento

- **Especificación de Requisitos Software ERS**
 - **Software Requirements Specification SRS**
- Este **documento** es la base para el acuerdo entre clientes y desarrolladores o suministradores sobre lo que hará el producto.
 - Es de gran ayuda para una evaluación rigurosa de los requisitos antes del comienzo del diseño, reduciendo los esfuerzos posteriores de rediseño.
 - Suele estar escrito en lenguaje natural, acompañado de descripciones formales o semi-formales.
 - Las notaciones deben permitir describir los requisitos tan preciso como sea posible.



Especificación - Documento

- Implicaciones:
 - Describir correctamente todos los requisitos del software
 - No describir ningún detalle del diseño del software
- **Características de una buena ERS (IEEE 830):**
 - No ambigua
 - Completa
 - Fácil de verificar
 - Consistente
 - Clasificada por orden de importancia o estabilidad
 - Fácil de modificar
 - Fácil para identificar el origen de cada requisito (traza)
 - Fácil de utilizar durante las fases



Especificación - Documento

- **Características de una buena ERS (IEEE 830):**
 - **No ambigua**
 - Un requisito ambiguo se presta a distintas interpretaciones.
 - Cada característica del producto final debe ser descrita utilizando un término único.
 - Si un término tiene distintos significados en distintos contextos se debe incluir un glosario
 - **Completa:**
 - Incluye todos los requisitos significativos del software
 - Define la respuesta del software a todas las posibles clases de datos de entrada y en todas las posibles situaciones
 - Está conforme con cualquier estándar de especificación que se deba cumplir.
 - Están etiquetadas y referenciadas en el texto todas las figuras, tablas y diagramas.



Especificación - Documento

- **Características** de una buena ERS (IEEE 830):
 - **Fácil de Verificar**
 - Existe algún procedimiento finito y efectivo en coste para que una persona o máquina compruebe que el SW satisface cada requisito
 - **Consistente**
 - Los Requisitos no entran en conflicto =>
 - Describen el mismo objeto o elemento pero con distintos términos
 - Se contradicen en:
 - Las características del objeto
 - Aspectos lógicos o temporales entre acciones
 - **Clasificada por orden de importancia o estabilidad**
 - Facilita la gestión a los desarrolladores



Especificación - Documento

- **Características** de una buena ERS (IEEE 830):
 - **Fácil de Modificar**
 - Cualquier cambio se puede realizar fácil, de forma completa y consistente
 - Implica una organización coherente y manejable
 - Tabla de Contenidos, Índice y Referencias Cruzadas
 - Es fundamental que la ERS sea No Redundante
 - **Facilidad de Traza (Trazabilidad)**
 - Debe facilitar las referencias con otros productos del ciclo de vida
 - Referencias hacia atrás
 - Referencias hacia delante



Especificación - Documento

- **Características** de una buena ERS (IEEE 830):
 - **Facilidad de Utilización** en Explotación y Mantenimiento
 - Se deben tener en cuenta las necesidades de Mantenimiento
 - El personal que no ha estado relacionado con el desarrollo del producto software se encarga del mantenimiento
 - Gran parte de los conocimientos y de la información necesaria para el mantenimiento se dan por supuestos en la organización del desarrollo



Especificación - Documento

- **Lectores/Usuarios** de la ERS
 - **Clientes.** Para comprobar que satisface sus necesidades y para hacer cambios a los requisitos.
 - **Gestores.** Para preparar una oferta por le sistema y planificar el proceso de desarrollo.
 - **Ingenieros de Desarrollo.** Para comprender el sistema a desarrollar.
 - **Ingenieros de Pruebas.** Para elaborar pruebas de validación del sistema.
 - **Ingenieros de Mantenimiento.** Para facilitar la comprensión del sistema y las relaciones entre sus partes.



Especificación – Estándar IEEE 830

- **IEEE Std 830 (1998)**
 - **IEEE Recommended Practice for Software Requirements Specifications.**
 - Disponible versión en español.
- Establece una estructura recomendable para el documento ERS.
 - Introducción
 - Descripción General
 - Requisitos Específicos
 - Apéndices
 - Índice



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:

1. Introducción
1.1. Objetivo
1.2. Ámbito
1.3. Definiciones, Siglas y Abreviaturas
1.4. Referencias
1.5. Visión Global
2. Descripción general
2.1. Perspectiva del producto
2.2. Funciones del producto
2.3. Características del usuario
2.4. Limitaciones generales
2.5. Supuestos y dependencias
3. Requisitos específicos
Apéndices
Índice

3. Requisitos específicos
3.1. Requisitos funcionales
3.1.1. Requisito funcional 1
3.1.1.1. Introducción
3.1.1.2. Entradas
3.1.1.3. Procedimiento
3.1.1.4. Salidas
3.1.2. Requisito funcional 2
.....
3.1.n. Requisito funcional n
3.2. Requisito de Interfaz externa
3.2.1. Interfaces de usuario
3.2.2. Interfaces hardware
3.2.3. Interfaces software
3.2.4. Interfaces de comunicaciones
3.3. Requisitos de ejecución
3.4. Restricciones de diseño
3.4.1. Acatamiento de estándares
3.4.2. Limitaciones hardware
3.5. Atributos de calidad
3.5.1. Seguridad
3.5.2. Mantenimiento
3.6. Otros requisitos
3.6.1. Base de datos
3.6.2. Operaciones
3.6.3. Adaptación de situación



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:
 - **1. Introducción:**
 - 1.1 Objetivo:
 - Propósito y Audiencia de la ERS
 - 1.2 Ámbito
 - ¿Qué hace y qué no hace el producto SW?
 - 1.3 Definiciones, siglas, y abreviaturas
 - En forma de apéndices o referencias a otros documentos
 - 1.4 Referencias
 - Lista completa de todas las referencias de los documentos en otra parte de la ERS
 - 1.5 Visión Global
 - Cómo se organiza el resto de la ERS



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:
 - **2. Descripción General:**
 - 2.1 Perspectiva del Producto
 - Relación con otros Productos SW del Sistema
 - Interfaces Sistema; Usuario; HW; SW; Comunicaciones ..
 - 2.2 Funciones del Producto
 - ¿Qué hace y qué no hace el producto SW?
 - 2.3 Características del Usuario
 - Bagaje, Formación Académica, Experiencia, Especialización Técnica
 - 2.4 Restricciones Generales
 - Regulaciones, Limitaciones HW, Interfaces, ...
 - 2.5 Asunciones y Dependencias
 - Factores que pueden afectar a los requisitos especificados



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:
 - 3. **Requisitos Específicos:**
 - Contiene todos los requisitos software.
 - Para cada requisito, se debe incluir:
 - Identificar único
 - Descripción de cada entrada (el estímulo) en el sistema,
 - Cada salida (la contestación) del sistema, y
 - Todas las funciones realizadas por el sistema en la salida a una entrada o en el apoyo de la salida. Esta es la parte más grande y más importante del SRS, los principios siguientes aplican:
 - Los requisitos se pueden organizar y ordenar de varias maneras (ver anexo A del estándar).



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:
 - 3. **Requisitos Específicos:** **EJEMPLO**

Requisito: 1.1 Dar de Alta Tarjeta	
Entrada	Los datos del cliente (nombre, apellidos, dirección, teléfono y e-mail)
Proceso	Generar un nuevo identificador de tarjeta. Añadir en el almacén TARJETAS una entrada con este identificador, la fecha de alta (fecha actual), los puntos iniciales de la tarjeta (0) y los datos del cliente.
Salida	Una entrada nueva en el almacén TARJETAS y la tarjeta física que se entrega al cliente.
Error	El usuario deberá introducir obligatoriamente el nombre y los apellidos del cliente. De no hacerlo, el sistema emitirá el mensaje de error "Imposible dar alta tarjeta: no se han proporcionado el nombre y los apellidos del cliente". Asimismo, para poder ponerse en contacto con el cliente, será obligatorio introducir la dirección y el teléfono del cliente. De no hacerse, el sistema mostrará el mensaje de error "Imposible dar alta tarjeta: no se han proporcionado la dirección y el teléfono del cliente".



Especificación – Estándar IEEE 830

- **Estructura** para la ERS:
 - **3. Requisitos Específicos:** según modo (anexo A)
 - 3.1 Interfaces Externas
 - Descripción detallada de las entradas y salidas del Sistema SW
 - Complementa las descripciones de Interfaz de los apartados anteriores
 - 3.2 Funciones (requisitos funcionales)
 - 3.3 Requisitos de Rendimiento/Ejecución
 - Estáticos y Dinámicos
 - 3.4 Restricciones de Diseño
 - Impuestas por otros estándares (formato informes; convenciones de nombrado elementos; etc..)
 - Limitaciones del HW
 - 3.5 Atributos de Calidad del Software
 - Fiabilidad, Disponibilidad, Mantenibilidad, Seguridad,
 - 3.6 Otros Requisitos

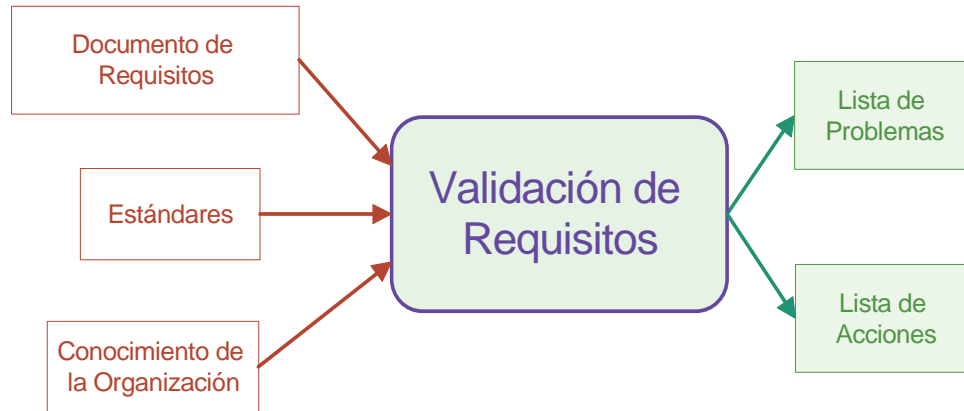


Validación

- Los requisitos deben ser validados para
 - Asegurar que el ingeniero software los ha comprendido
 - El documento de ERS es conforme a los estándares establecidos, comprensible, consistente y completo.
- Se trata de asegurar que el documento ERS define **el software adecuado**, es decir, **el que espera el usuario**.
- Las principales técnicas de validación:
 - Revisiones
 - Prototipado
 - Validación de Modelos *[otro tema]*
 - Pruebas de Aceptación *[otro tema]*



Validación



Análisis

- ¿Tenemos los requisitos correctos?

Validación

- ¿Tenemos los requisitos descritos correctamente?



Validación

- **Comprobaciones** a realizar:
 - **Validez.** El sistema provee las funciones que soportan mejor las necesidades del cliente.
 - **Consistencia.** No existen conflictos entre los requisitos.
 - **Compleitud.** Están incluidas todas las funciones requeridas por el usuario.
 - **Realismo.** Los requisitos pueden ser implementados con el presupuesto y tecnologías disponibles.
 - **Verificable.** Los requisitos pueden ser verificados.



Validación - Revisiones

- **Técnicas de Validación de Requisitos:**
 - **Revisiones** [inspecciones]
 - Un grupo de personas lee, analiza, discute el documento de ERS y las formas de solucionar los problemas detectados.
 - Deben participar representantes del cliente y los usuarios.
 - Norma: IEEE Std 1028-1997, *IEEE Standard for Software Reviews*

 - Las inspecciones pueden ser formales (documento concluido) o informales.



Validación - Revisiones

- **Técnicas de Validación de Requisitos:**
 - **Revisiones** [revisiones]
 - Se pueden usar **Listas de Comprobación:**
 - **Unicidad**
 - ¿Está cada requisito identificado de forma unívoca?
 - **Comprensibilidad**
 - ¿Se describen en el glosario los términos especializados o técnicos?
 - ¿Es cada requisito autocontenido o se tienen que examinar otros requisitos para comprenderlo?
 - ¿Hay contradicciones?
 - ¿Están agrupados los requisitos relacionados?
 - **Trazabilidad**
 - ¿Está establecido adecuadamente el origen de cada requisito?
 - **Adaptabilidad**
 - ¿Puede cambiarse un requisito sin impactar demasiado a los otros?

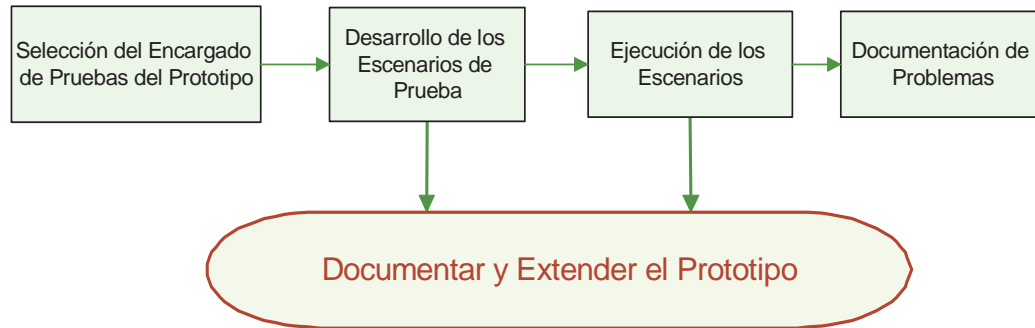


Validación - Prototipado

- **Técnicas de Validación de Requisitos:**

- **Prototipos:**

- Si se desarrolló durante la Elicitación se sigue con su desarrollo en esta fase



- Escribir el **Manual de Usuario** fuerza a un análisis detallado de los requisitos y así revelarse nuevos problemas



Gestión de Requisitos

- **Los requisitos cambian:**

- Cambio en la estrategia o prioridades del negocio
- Cambios tecnológicos
- Cambios en leyes o regulaciones

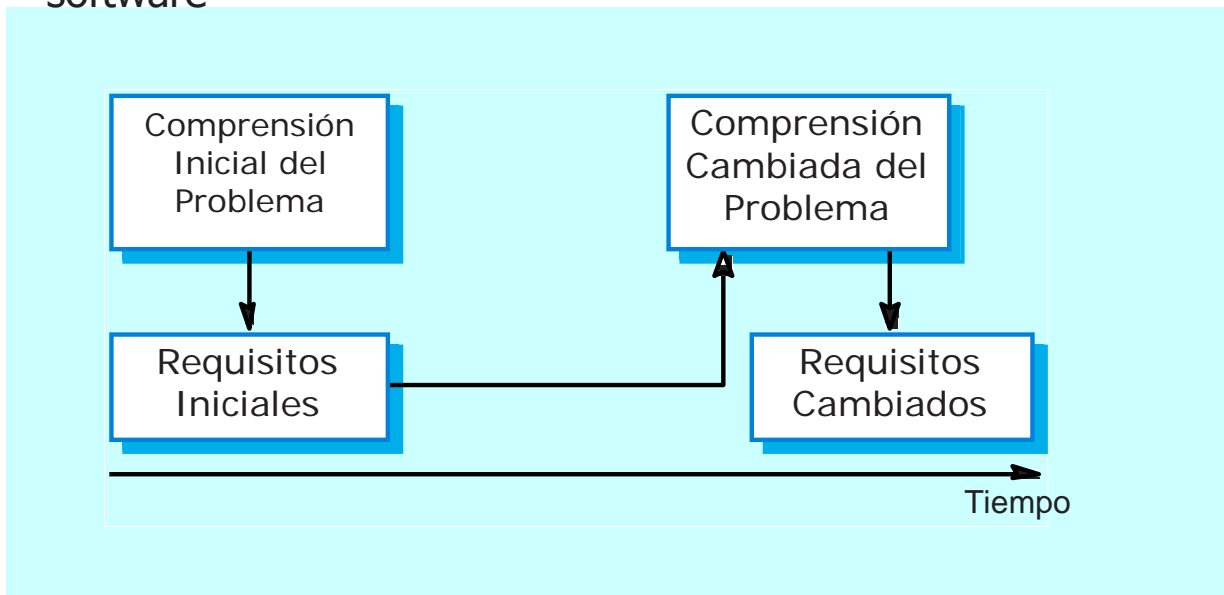
- La **Gestión de Requisitos** consiste en gestionar

- Los **cambios** en los requisitos acordados
- Las **relaciones** entre requisitos
- Las **dependencias** entre el **documento** ERS y otros documentos



Gestión de Requisitos

- Más del 50% de los requisitos suelen cambiar durante la elaboración del ERS y después, durante el desarrollo del software



Gestión de Requisitos

- La **Gestión de Requisitos** implica la recolección, almacenamiento y mantenimiento de grandes cantidades de información
- Para ello existen herramientas CASE con:
 - Base de Datos para almacenar requisitos
 - Facilidades de análisis y generación de documentos
 - Facilidades de gestión de cambios
 - Facilidades de Trazabilidad



Gestión de Requisitos

- Algunos requisitos cambian más que otros
 - **Requisitos Estables**
 - Esencia del Sistema y su Dominio de Aplicación
 - **Requisitos Volátiles**
 - Específicos del Sistema en un entorno particular para un cliente particular
 - Las **causas del cambio** son variadas:
 - Corrección de errores y problemas en los requisitos
 - Conocimiento creciente del cliente/usuario
 - Problemas técnicos, de calendario o costes
 - Cambio en las prioridades del cliente
 - Cambios en el entorno
 - Cambios organizacionales
 - Requisitos emergentes durante el diseño / implementación
 - Asunciones erróneas
 - Dependencias de otros equipos y procesos



Gestión de Requisitos

- Es importante poder identificar y almacenar los requisitos
 - **Identificación de Requisitos**
 - Asignación no ambigua de identificadores
 - Renumeración dinámica, Identificación de Registro de BD, Identificación simbólica
 - **Almacenamiento de Requisitos**
 - Se debe facilitar acceso rápido y en relación con otros sistemas
 - Documentos o BD

SYS_MODELS
Model: MODEL Description: TEXT Next: MODEL NULL

REQUIREMENT
Identifier: TEXT Statement: TEXT GRAPHIC Date_entered: DATE Date_changed: DATE Sources: SOURCE_LIST Rationale: REQ_RATIONALE Status: STATUS Dependents: REQ_LIST Is_dependent_on: REQ_LIST Model_links: SYS_MODELS Comments: TEXT

SOURCE_LIST
People: TEXT Documents: TEXT Reqs: REQ_LIST

REQ_LIST
Req: REQUIREMENT Description: TEXT Next: REQUIREMENT NULL

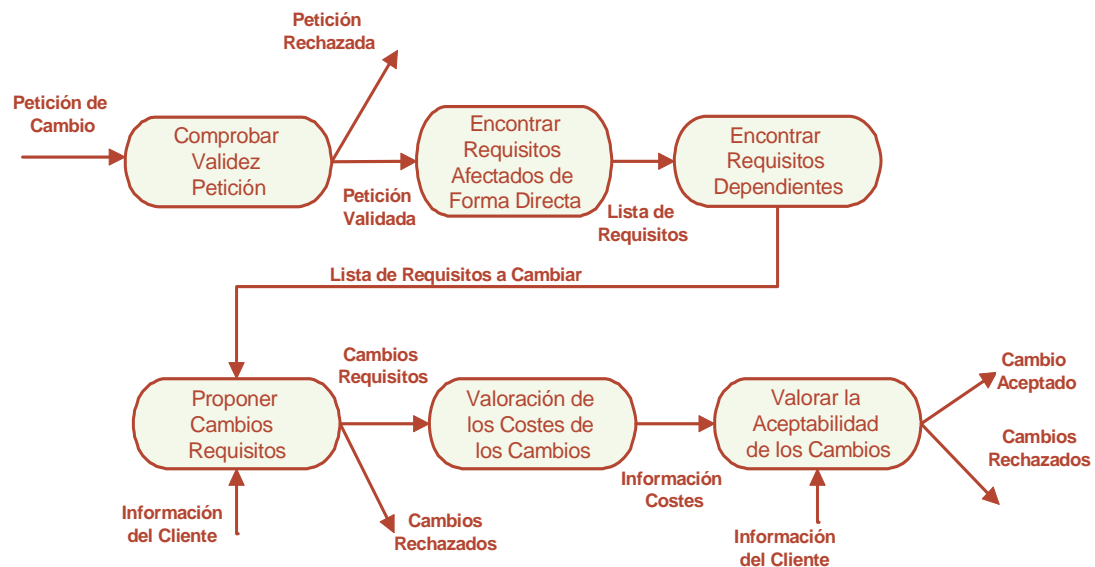
REQ_RATIONALE
Rationale: TEXT Diagrams: GRAPHIC Photos: PICTURE



Gestión de Requisitos

● Gestión de los Cambios:

- Identificación del Cambio / Análisis del Cambio / Implementación del Cambio



Gestión de Requisitos - Atributos

- Para una correcta gestión de los requisitos es necesario registrar diversos **atributos** para cada uno:
 - ID único
 - Especificación
 - Clasificaciones según criterios establecidos
 - Método de verificación
 - Plan de aceptación
 - Resumen
 - Origen
 - Historial de Cambios



Gestión de Requisitos - Trazabilidad

- La **Trazabilidad de los Requisitos** consiste en poder recuperar el origen o predecir los efectos de los requisitos.
 - Es fundamental para analizar el impacto de un cambio en los requisitos.
- La trazabilidad de un requisito tiene dos direcciones:
 - Hacia atrás** para conocer los requisitos de usuario e interesados que lo motivaron.
 - Hacia delante** para identificar los requisitos y entidades de diseño que lo satisfacen.



Gestión de Requisitos - Trazabilidad

- Información para **Trazabilidad de Requisitos**
 - Backward-from Fuentes \leftarrow R
 - Enlace de Requisitos con sus fuentes
 - Forward-from R \rightarrow D-I
 - Enlace Requisitos con Diseño e Implementación
 - Backward-to R \leftarrow D-I
 - Enlace Diseño e Implementación con Requisitos
 - Forward-to Fuentes \rightarrow R
 - Enlaza documentos previos con los Requisitos
 - Formatos:
 - Matriz de Trazabilidad (arriba)
 - Lista de Trazabilidad (abajo)

Depends-on

	R1	R2	R3	R4	R5	R6
R1			*	*		
R2					*	*
R3				*	*	
R4		*				
R5						*
R6						

Requisito	Depende de
R1	R3, R4
R2	R5, R6
R3	R4, R5
R4	R2
R5	R6