

Ingeniería de Requisitos

Temario

- Definiciones
- Requisitos Funcionales y No Funcionales
- Tipos de Requisitos
- Ingeniería de Requisitos
 - Proceso de los Requisitos
 - Obtención de Requisitos - Técnicas
 - Modelado del Sistema - Técnicas
 - Especificación de Requisitos - Documentos de Requisitos
 - Validación – Técnicas
 - Administración de los Requisitos
 - Administración del cambio
 - Medición

Bibliografía

- Pfleeger: Capítulo 4
- Ingeniería de SW - Sommerville (7ma. edición): Capítulos
 - 6 – Requisitos del software
 - 7 – Procesos de la Ing. de Requisitos
 - 10 – Especificación formal
- IEEE Recommended Practice for Software Requirements Specifications – Std 830-1998.
- Casos de uso:
 - Larman: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)
 - Artículos en la página:
 - Capítulo 6 del libro de Craig Larman: Applying UML and Patterns – 2nd Edition
 - Capítulo 19 del libro de Alistair Cockbourn: Writing Effective Use Cases
 - Is the clock an actor? Artículo de la revista Rational Edge
- UML:
 - <http://www.uml.org/>
 - Artículos en la página:
 - Introducción a UML - Artículo de la revista Rational Edge
 - Diagramas de Actividad - Artículo de la revista Rational Edge

Definiciones

- Requisitos:
 - Descripción de los servicios que debe brindar un sistema y sus restricciones.
- Ingeniería de Requisitos
 - Proceso de descubrir, analizar, documentar y verificar esos servicios y restricciones.

Definiciones

- Sistema
 - Incluye *hardware, software, firmware, personas, información, técnicas, servicios, y otros elementos de soporte*
- Requisitos del Sistema
 - Son los requisitos para el sistema entero
- Requisitos del Software
 - Se refieren solo al SW

Requisitos vs. Diseño

- Requisitos definen el Qué (el problema) del sistema
- El Diseño define el Cómo (la solución)

Estadísticas de proyectos de software

TRADITIONAL RESOLUTION FOR ALL PROJECTS

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|------------|------|------|------|------|------|
| SUCCESSFUL | 39% | 37% | 41% | 36% | 36% |
| CHALLENGED | 39% | 46% | 40% | 47% | 45% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

The Traditional resolution of all software projects from FY2011–2015 within the new CHAOS database.

Estadísticas de proyectos de software

| PROJECT SIZE BY CHAOS RESOLUTION | | | | |
|----------------------------------|------------|------------|--------|-------|
| | SUCCESSFUL | CHALLENGED | FAILED | TOTAL |
| Grand | 6% | 51% | 43% | 100% |
| Large | 11% | 59% | 30% | 100% |
| Medium | 12% | 62% | 26% | 100% |
| Moderate | 24% | 64% | 12% | 100% |
| Small | 61% | 32% | 7% | 100% |

The size of software projects by the Modern Resolution definition from FY2011–2015 within the new CHAOS database.

The Standish Group - Chaos Report 2015 - https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Estadísticas de proyectos de software

CHAOS RESOLUTION BY PROJECT TYPE

| PROJECT TYPE | SUCCESSFUL | CHALLENGED | FAILED |
|--|------------|------------|--------|
| Developed from scratch using traditional languages and methods | 22% | 61% | 17% |
| Developed from scratch using modern methodologies | 23% | 54% | 23% |
| Developed some components and purchased others | 24% | 59% | 17% |
| Purchased components and assembled the application | 25% | 59% | 16% |
| Purchased application and modified | 42% | 37% | 21% |
| Purchased application and performed no modifications | 57% | 28% | 15% |
| Modernization | 53% | 38% | 9% |
| Other | 28% | 47% | 25% |

The resolution of all software projects by project type from FY2011–2015 within the new CHAOS database.

The Standish Group - Chaos Report 2015 - https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Estadísticas de proyectos de software

| CHAOS RESOLUTION BY AGILE VERSUS WATERFALL | | | | |
|--|-----------|------------|------------|--------|
| SIZE | METHOD | SUCCESSFUL | CHALLENGED | FAILED |
| All Size Projects | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| Large Size Projects | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Medium Size Projects | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Small Size Projects | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

The Standish Group - Chaos Report 2015 - https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Estadísticas de proyectos de software

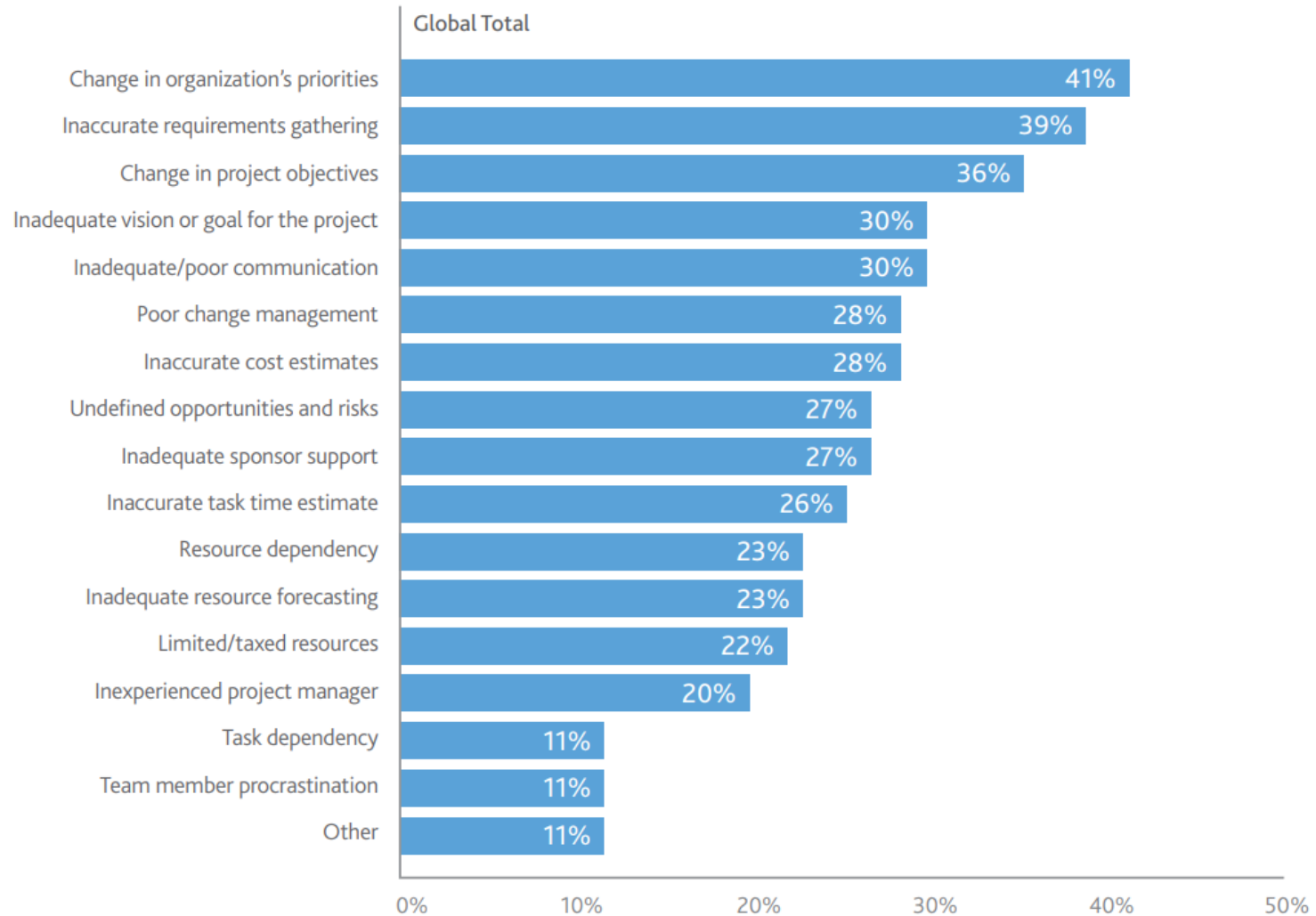
CHAOS FACTORS OF SUCCESS

| FACTORS OF SUCCESS | POINTS | INVESTMENT |
|------------------------------|--------|------------|
| Executive Sponsorship | 15 | 15% |
| Emotional Maturity | 15 | 15% |
| User Involvement | 15 | 15% |
| Optimization | 15 | 15% |
| Skilled Resources | 10 | 10% |
| Standard Architecture | 8 | 8% |
| Agile Process | 7 | 7% |
| Modest Execution | 6 | 6% |
| Project Management Expertise | 5 | 5% |
| Clear Business Objectives | 4 | 4% |

The 2015 Factors of Success. This chart reflects our opinion of the importance of each attribute and our recommendation of the amount of effort and investment that should be considered to improve project success.

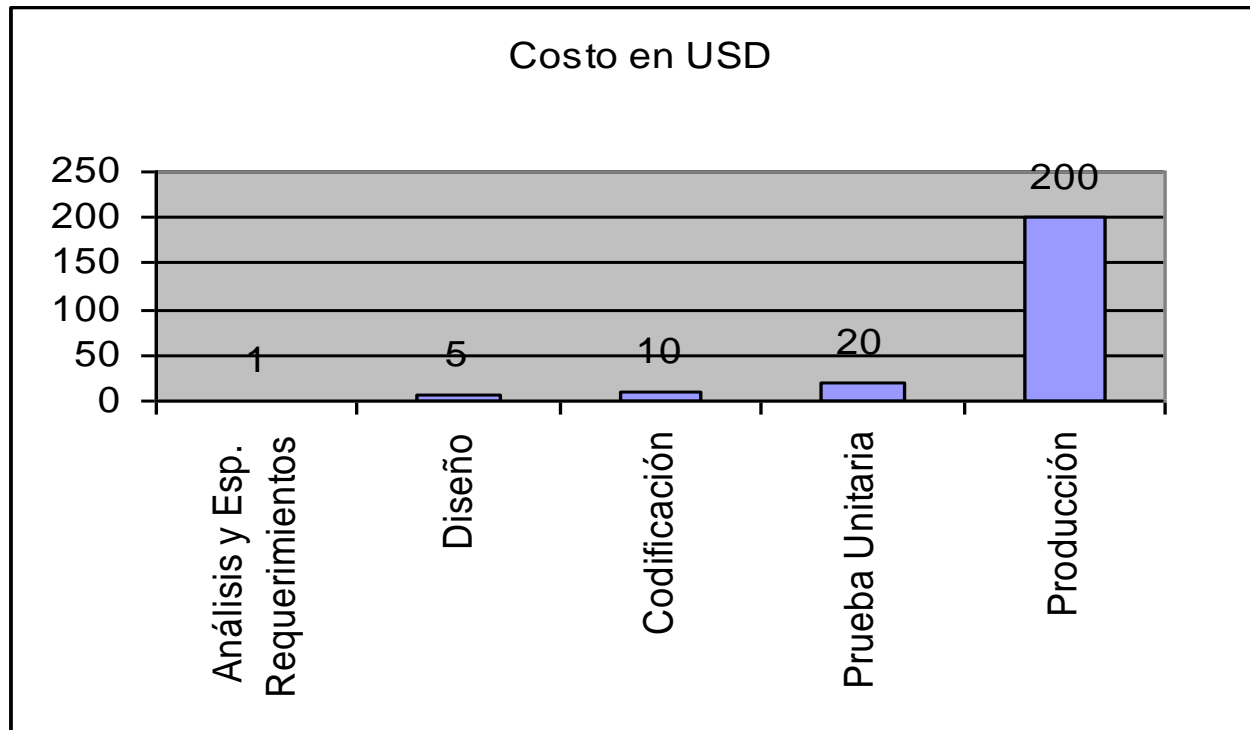
The Standish Group - Chaos Report 2015 - https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Estadísticas de proyectos de software



Costos de Errores en los Requisitos

- Costo de corregir un error en los requisitos (Boehm-Papaccio, 1988)





La solicitud del usuario



Lo que entendió el líder del proyecto



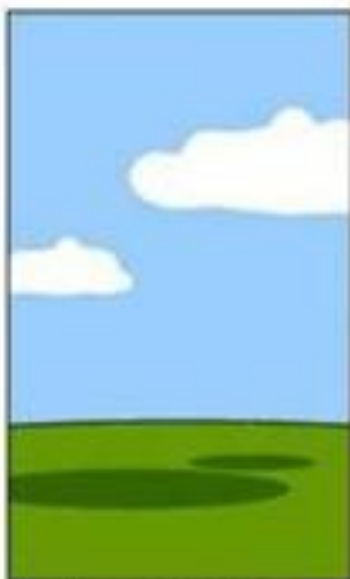
El diseño del analista de sistemas



El enfoque del programador



La recomendación del consultor externo



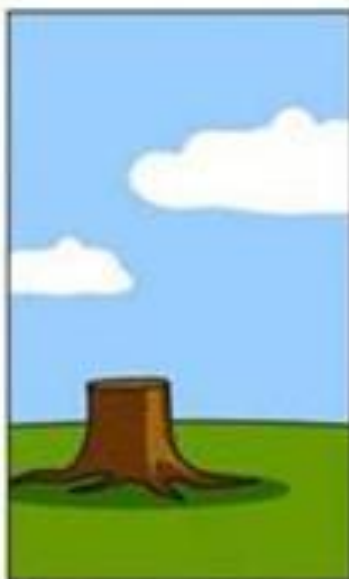
La documentación del proyecto



La implantación en producción



El presupuesto del proyecto



El soporte operativo



Lo que el usuario realmente necesitaba

Requisitos Funcionales y No Funcionales

- Funcionales:
 - Servicios o funciones que proveerá el sistema
 - Describen la interacción entre el sistema y su entorno
 - **Ejemplos:**
 - Se deben ingresar cédula, nombre y teléfono de cada cliente
 - Se quiere un listado de los clientes por zona
- No-funcional:
 - Restricciones a los servicios o funciones ofrecidos por el sistema
 - Describen restricciones que limitan las elecciones para construir una solución
 - **Ejemplos:**
 - Las consultas deben resolverse en menos de 3 segundos
 - El lenguaje de programación debe ser Java

Requisitos No Funcionales

- Del Producto: Especifican restricciones al comportamiento del producto
 - Ejemplos: desempeño, confiabilidad, portabilidad, usabilidad
- De la Organización: Se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador
 - Ejemplos: estándares, lenguajes de programación, método de diseño
- Externos: Se derivan de factores externos, como:
 - Interoperabilidad: con otros sistemas
 - Legislativos: privacidad, seguridad
 - Éticos: dependen del contexto, las personas, etc

Requisitos - Tipos (1)

Al describir requisitos se deben tener en cuenta los siguientes aspectos:

- Ubicación y Entorno Físicos
 - dónde, uno o varios, restricciones ambientales
- Interfaces
 - Entrada de 1 o + sistemas, Salida a 1 o + sistemas, restricciones de formato, soporte
- Usuarios y Factores Humanos
 - capacidad de cada tipo de usuario, tipo de entrenamiento, facilidad de uso, posibilidad de mal uso
- Funcionalidad y Restricciones asociadas
 - qué debe hacer, cuándo, modos de operación, cómo y cuándo se puede modificar el sistema, restricciones de velocidad, tiempo de respuesta, capacidad de proceso

Requisitos - Tipos (2)

- Documentación
 - cuánto, formato, para quién
- Datos
 - formatos E/S, frecuencia, fuentes, destinos, calidad requerida, precisión en cálculos, flujo en el sistema
- Recursos
 - materiales, personal y otros para construir, usar y mantener el sistema, habilidades de los desarrolladores, necesidades de espacio y ambientales, calendario prescrito, limitaciones en presupuesto

Requisitos - Tipos (3)

- Seguridad
 - control de acceso a las funciones/datos, aislamiento de los programas, respaldos-frecuencia, disponibilidad-, seguridad física
- Aseguramiento de la Calidad
 - Confiabilidad – tiempo medio entre fallas, robustez, tolerancia a fallas
 - Disponibilidad - tiempo para estar operativo luego de falla-mantenimiento estando activo- tiempo máximo de no disponibilidad
 - Mantenibilidad
 - Seguridad
 - Portabilidad

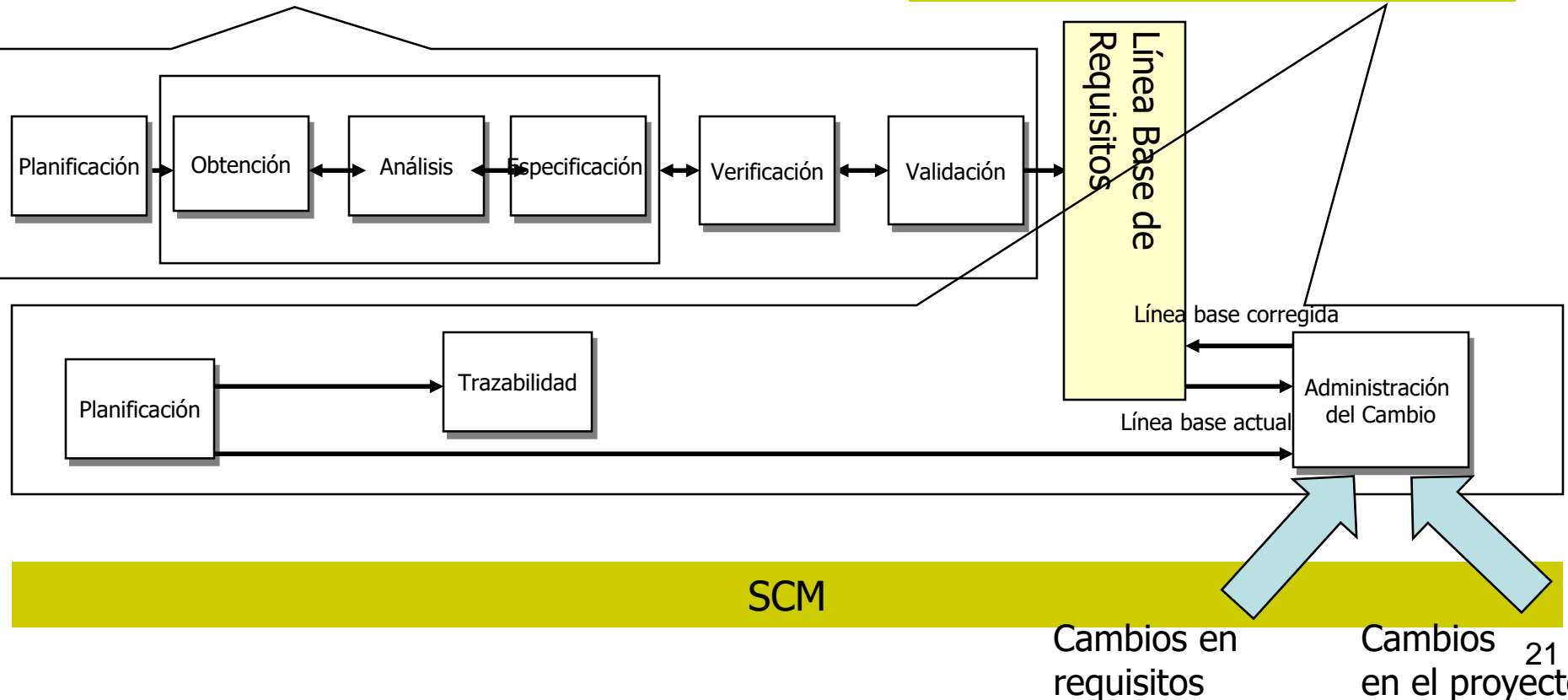
Ingeniería de Requisitos

Ingeniería de Requisitos

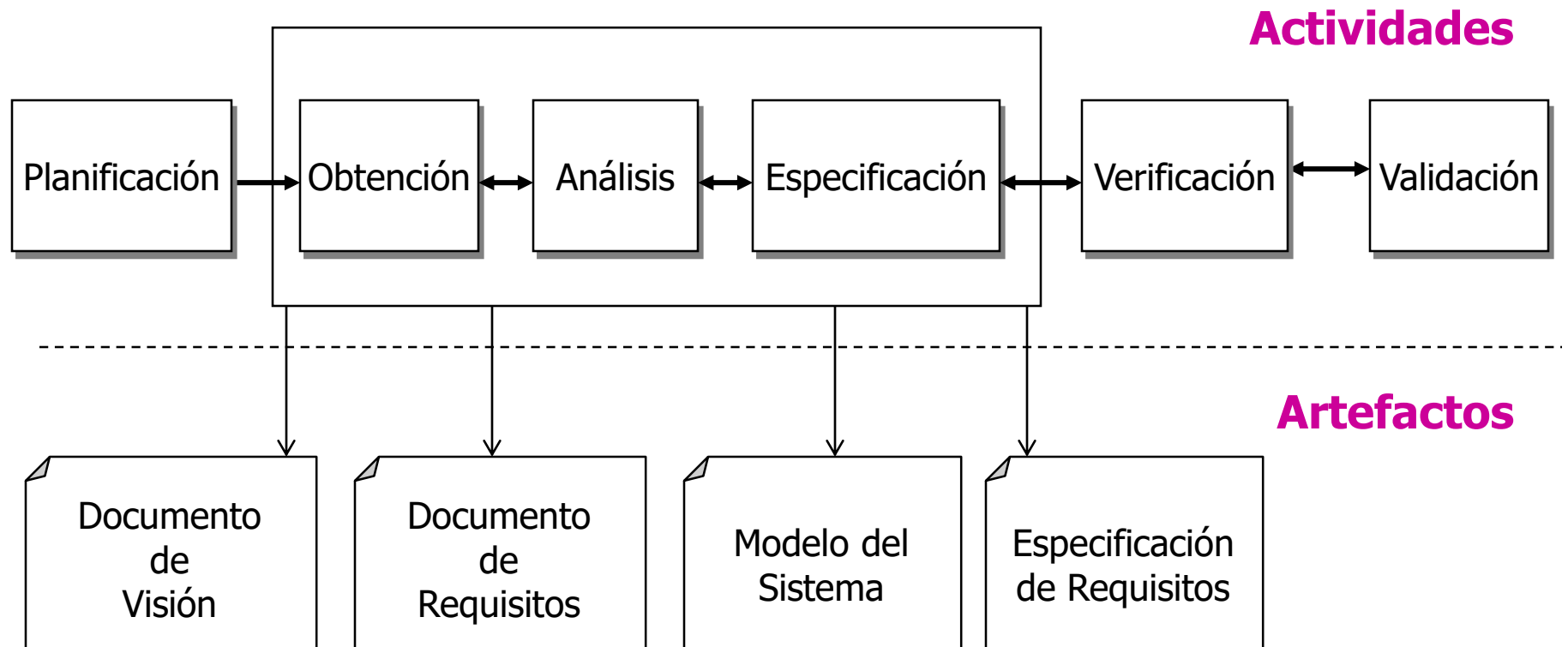
Ingeniería de Requisitos

Proceso de los Requisitos

Administración de los Requisitos



Proceso de Requisitos

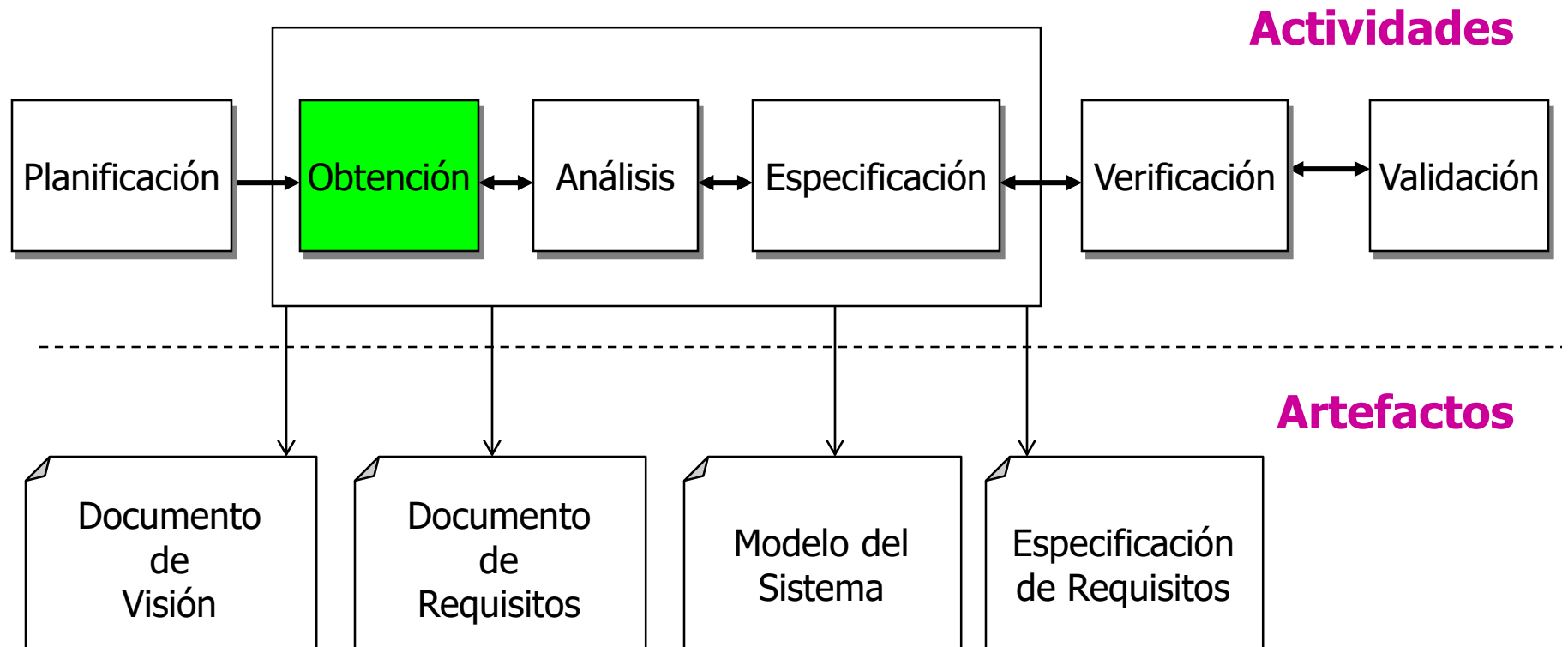


Participantes en el Proceso de Requisitos

- Cliente y Usuarios
 - Requisitos adecuados a sus necesidades
- Diseñadores
 - Comprenderlos para lograr diseño que los satisfaga
- Supervisores del Contrato, sugieren:
 - Hitos de Control, cronogramas
- Gerentes del Negocio, entienden:
 - Impacto en la Organización
- Verificadores
 - Comprenderlos para poder verificar si el sistema los satisface

Obtención de Requisitos

Proceso de Requisitos



Requisitos - Fuentes

- Necesidades del cliente, usuario, otros interesados
- Modelos del dominio
- Revisar la situación actual
- Organización actual y sistemas
- Versión actual del sistema
- Desarrolladores de versión anterior
- Documentos existentes (antecedentes)
- Sistemas análogos ya existentes (antecedentes)

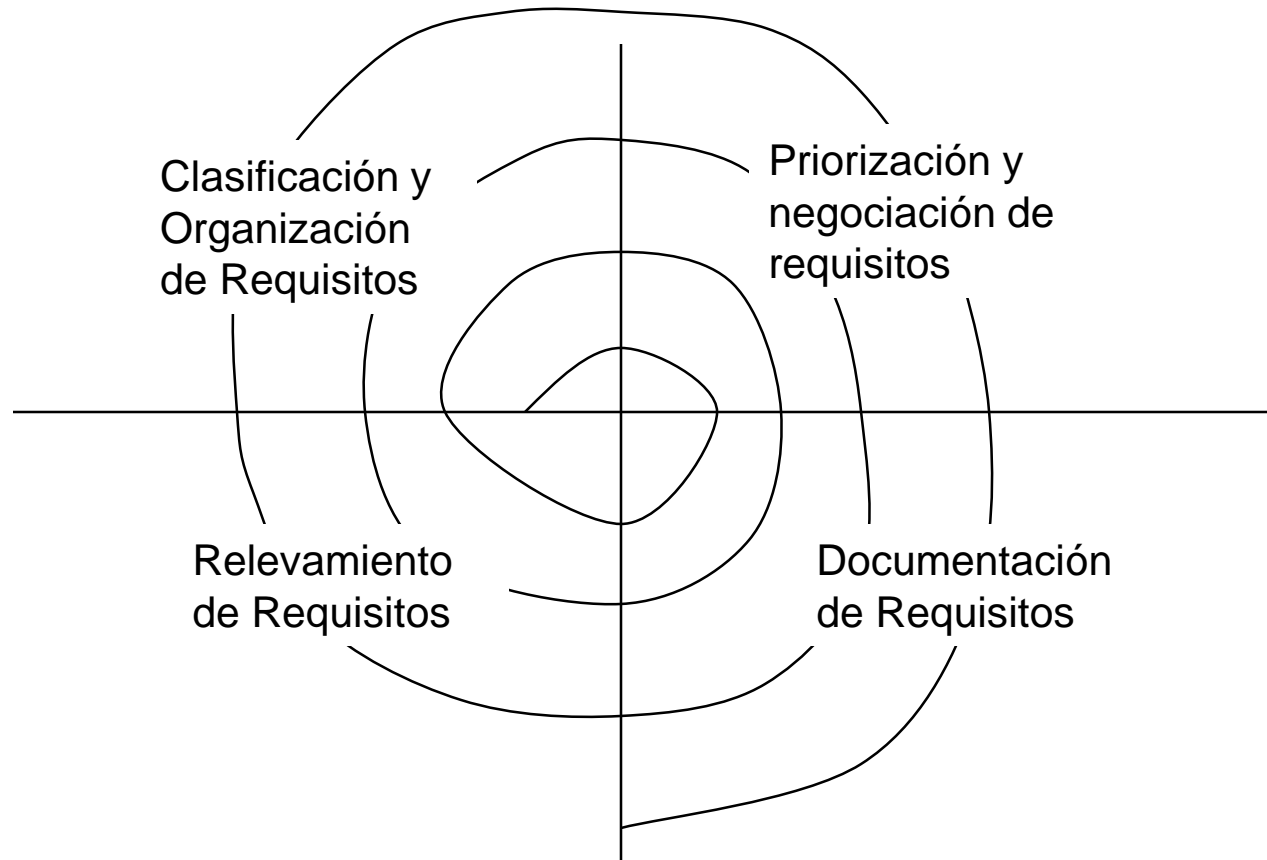
Obtención & Análisis de Requisitos

- Se trabaja en conjunto con los usuarios y clientes
- Problemas comunes:
 - No saben lo que quieren del sistema, sólo en términos generales, no conocen el costo de sus peticiones
 - Los requisitos están en sus términos y con conocimiento implícito de su propio trabajo
 - Distintos usuarios tienen distintos requisitos, se deben encontrar todas las fuentes
 - Influyen factores políticos
 - La prioridad que se da a los requisitos varía con el tiempo
 - Aparecen nuevos requisitos

Brecha en la Comunicación (Scharer '90)

| Según <u>desarrolladores</u> , los <u>usuarios</u> ... | Según <u>usuarios</u> , los <u>desarrolladores</u> ... |
|---|--|
| no saben lo que quieren | no captan las necesidades operativas |
| no pueden articular lo que quieren | ponen excesivo énfasis en aspectos meramente técnicos |
| muchas necesidades por motivos políticos | pretenden indicarnos cómo hacer nuestro trabajo |
| quieren todo ya | no son capaces de traducir necesidades claramente establecidas en un sistema |
| son incapaces de definir prioridades entre sus necesidades | siempre dicen que no |
| rehúsan asumir responsabilidades por el sistema | siempre están pasados del presupuesto |
| incapaces de dar un enunciado utilizable de sus necesidades | siempre están atrasados |
| no están comprometidos con los proyectos de desarrollo | nos exigen tiempo y esfuerzo aún a costa de las obligaciones esenciales |
| no aceptan soluciones de compromiso | establecen estándares no realistas para la definición de requisitos |
| no pueden mantener el cronograma | son incapaces de responder rápidamente a cambios en las necesidades |

Obtención & Análisis de Requisitos (Modelo Genérico)



Qué relevar

- Requerimientos del negocio:
 - Objetivos del negocio de alto nivel. Responden a la pregunta: ¿cómo será el mundo mejor para una determinada comunidad?
 - Categorías:
 - beneficios financieros
 - mejora de las operaciones de negocio
 - posicionamiento estratégico / competitivo
 - adoptar una nueva ley
 - nivelado / desarrollo tecnológico.
 - product vision statement
 - beneficios para los stakeholders
 - descripción de la funcionalidades en alto nivel
 - prioridades del proyecto
 - limitaciones del producto
- **Documento de visión.** Sirve para establecer una visión común con el cliente y para difusión del proyecto.

Qué relevar

- Reglas de negocio. - Existen independientemente del software. Aplican aunque se haga manualmente.
 - políticas de la organización
 - estándares
 - algoritmos
 - leyes y regulaciones
- Requisitos funcionales
- Requisitos no funcionales:
 - atributos de Q
 - interfaces externas
 - restricciones
- Criterios de éxito. Por ejemplo:
 - que puedan desarrollar bien tareas significativas
 - manejo de errores
 - que satisfaga expectativas de calidad

Obtención de Requisitos – Técnicas

- Investigar antecedentes
- Entrevistas individuales/grupales
- Encuestas/Cuestionarios
- Tormenta de ideas
- Workshop
- Casos de Uso
- Observación/Participación
- Prototipado

Investigar antecedentes

- Estudio, muestreo, visitas,...
- Buena forma de comenzar un proyecto
- Interna: estructura de la organización, políticas y procedimientos, formularios e informes, documentación de sistemas
- Externa: publicaciones de la industria y comercio, Encuentros profesionales, visitas, literatura y presentaciones de vendedores

■ Ventajas

- Ahorra tiempo de otros
- Prepara para otros enfoques
- Puede llevarse a cabo fuera de la organización

■ Desventajas

- Perspectiva limitada
- Desactualizado
- Demasiado genérico

Entrevista Individual / Grupal

- Usar para:
 - Entender el problema de negocio
 - Entender el ambiente de operación
 - Evitar omisión de requisitos
 - Mejorar las relaciones con el cliente
- Ventajas
 - Orientado a las personas
 - Interactivo/flexible
 - Rico
- Desventajas
 - Costoso
 - Depende de las habilidades interpersonales
- Pasos para las Entrevistas
 - Seleccionar participantes
 - Aprender tanto como sea posible de antemano
 - Preparar la entrevista
 - Utilizar un patrón de estructura
 - Conducirla
 - Apertura, desarrollo, conclusión
 - Enviar un memo con resultado
 - Seguimiento

Entrevista – Patrón para conducirla

- Datos de las Personas: usuarios, interesados, disparador del proyecto
 - ¿Qué trabajo realizan? ¿Para quién?
 - ¿Qué interfiere con su trabajo?
 - ¿Qué cosas hacen su trabajo mas fácil o mas difícil?
- Datos: entradas y salidas clave, datos ya existentes
 - Listar las entradas y salidas
 - ¿Cuál es el problema? ¿Cómo se resuelve ahora? ¿Como le gustaría que se resolviera?
- Procesos: propósito, objetivos y metas
 - ¿Quién necesita la aplicación?
 - ¿Cuántos usuarios la van a usar y de qué tipo?
- Ubicaciones: lugares involucrados, contexto de los usuarios
 - Entorno de los usuarios, computadoras, plataformas
 - Aplicaciones relevantes existentes
 - Experiencia de los usuarios con este tipo de aplicación, expectativas de tiempo de entrenamiento

Entrevista – Patrón para conducirla (2)

- Evaluar confiabilidad, desempeño y soporte necesario:
 - ¿Cuáles son las expectativas respecto a la confiabilidad?
 - ¿Y respecto a la performance?
 - ¿Qué tipo de mantenimiento se espera?
 - ¿Qué nivel de control y seguridad?
 - ¿Qué requisitos de instalación existen?, ¿cómo se distribuye el software? , ¿debe ser empaquetado?
- Otros
 - ¿Existen requisitos legales, regulatorios u otros estándares que deban ser tenidos en cuenta?
- Factores críticos de éxito:
 - ¿Qué se considera una buena solución?
- Tener en cuenta:
 - Si el entrevistado comienza a hablar sobre los problemas existentes, no cortarlo con una próxima pregunta
 - Luego de la entrevista y mientras los datos aún están en mente, resumir los principales req. (aprox. 3) de este entrevistado

Encuesta / Cuestionario

- No substituye la entrevista
 - Antes de usar el enfoque:
 - Determinar la información que se precisa
 - Determinar el enfoque más adecuado:
 - Abierto, cerrado, combinado
 - Múltiple opción, valor en escala, orden relativo
 - Desarrollar cuestionario
 - Probarlo con perfil típico
 - Analizar resultado de las pruebas
 - Su principal uso es para validar asunciones y obtener datos estadísticos sobre preferencias
- | | |
|---|--|
| <ul style="list-style-type: none">■ Ventajas<ul style="list-style-type: none">■ Economía de escala■ Conveniente para quien contesta■ Respuestas anónimas | <ul style="list-style-type: none">■ Desventajas<ul style="list-style-type: none">■ Menos rico■ Problemas por no-respuesta■ Esfuerzo de desarrollo |
|---|--|

Observación / Participación

- Poco utilizado...
- Antes de usarlo
 - Determinar información necesaria
 - Comunicar a los involucrados
 - Considerar períodos normales y atípicos
 - Planificar las anotaciones
- Ventajas
 - Confiable
 - Muy rico
 - Desarrolla empatía
- Desventajas
 - Efecto Hawthorne
 - Cuidado con generalizar demasiado (sesgo particular/local)

Tormenta de Ideas (Brainstorming)

- Objetivo: Lograr consenso sobre los requisitos
- Ayuda a la participación de todos los involucrados
- Permite pensar en otras ideas
- Un secretario saca notas de todo lo discutido
- Reglas
 - No se permite criticar ni debatir
 - Dejar volar la imaginación
 - Generar tantas ideas como sea posible
 - Mutar y combinar ideas

Tormenta de Ideas – Fase de Generación

- Los principales stakeholders se juntan en un cuarto.
- Se explican las reglas.
- Se establece el objetivo:
 - ¿Qué características esperan en el producto?
 - ¿Qué servicios esperan que provea?

Los objetivos permiten decidir cuando terminar.
- Se pide que cada participante escriba sus ideas, luego las ideas son leídas para que otros piensen en ideas relacionadas y de esa forma las ideas mutan y se combinan.

Tormenta de Ideas – Fase de Reducción

- El secretario lee cada idea y pregunta si es válida
 - Si hay cualquier desacuerdo, la idea se queda
- Agrupamiento de ideas
 - Nombrar los grupos
- Definición
 - Se escribe una breve descripción de lo que la idea significa para la persona que la escribió
 - Ayuda a tener un entendimiento común del requerimiento
 - Lleva unos minutos por idea
- Priorización (opcional)
 - Test de los \$100: Cada persona tiene dinero para comprar ideas, se ordena según ideas más compradas
 - Solo se puede hacer una vez
 - Se debe limitar la cantidad a gastar en 1 sola idea

Sesiones de Trabajo (Workshop)

- Ámbito para las tormentas de ideas
- Preparación
 - Venderlo a los posibles miembros de la reunión
 - Asegurarse que asisten los stakeholders correctos
 - Estructurar la invitación, el lugar, etc.
 - Enviar material previo a la reunión
 - Doc de requisitos
 - Entrevistas, defectos de los sistemas existentes, etc.
 - Asegurarse de enviar lo necesario, sin exagerar
 - Organizar la Agenda
 - Introducción
 - Tormenta de ideas – generación
 - Tormenta de ideas – reducción
 - Priorización
 - Resumen

Casos de Uso

- Formato simple y estructurado donde los usuarios y desarrolladores pueden trabajar juntos
- No son de gran ayuda para identificar aspectos no funcionales
- Mientras se definen los casos de uso, puede ser un buen momento para definir pantallas u otros objetos con los que el usuario interactúa
- Pueden ser usados en el diseño y en el testing del sistema

■ Usarlo

- Cuando el sistema está orientado a la funcionalidad, con varios tipos de usuarios
- Cuando la implementación se va a hacer OO y con UML

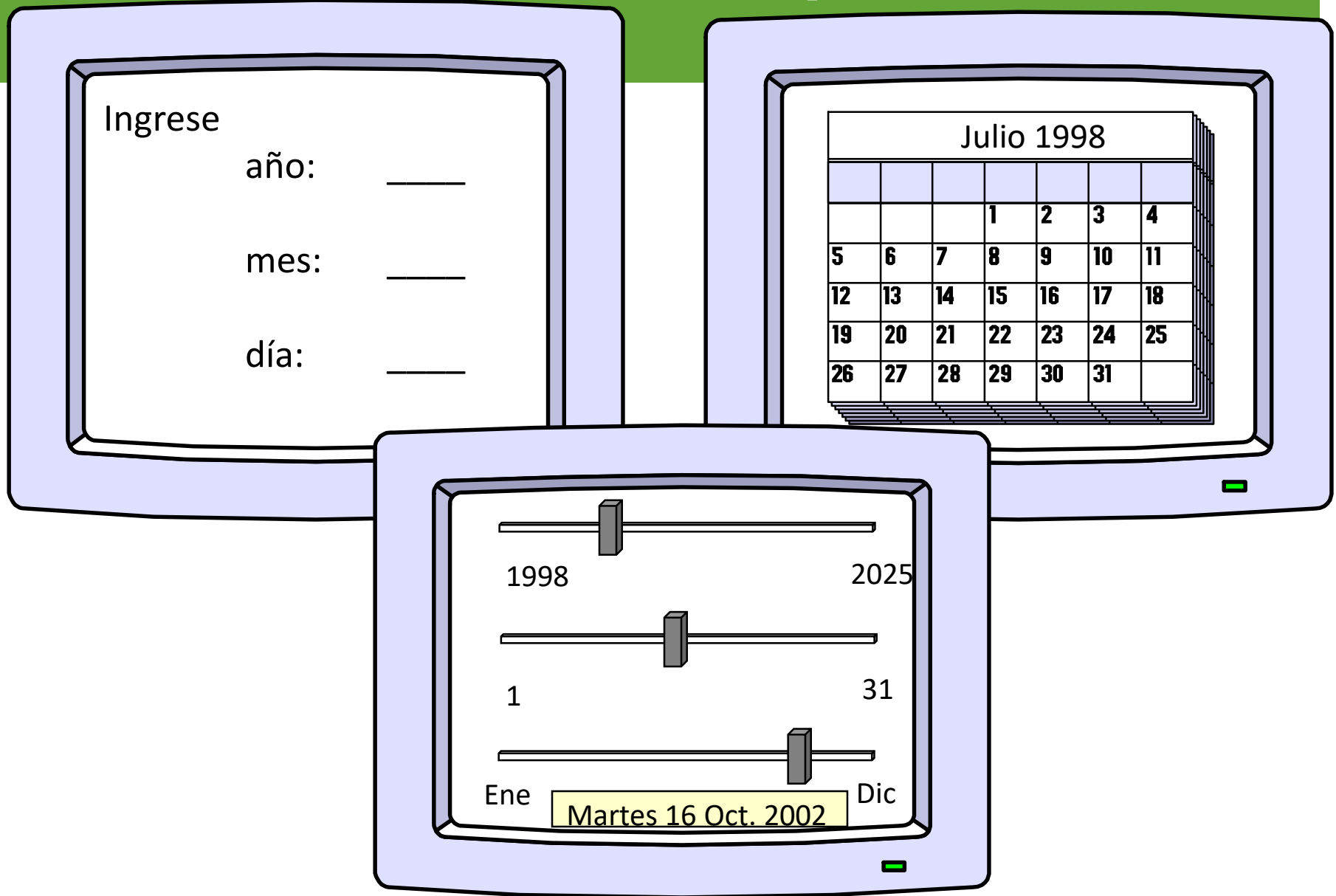
■ No son la mejor elección:

- Sistemas sin usuarios y con pocas interfaces
- Sistemas dominados primariamente por requisitos no funcionales y restricciones de diseño

Prototipado

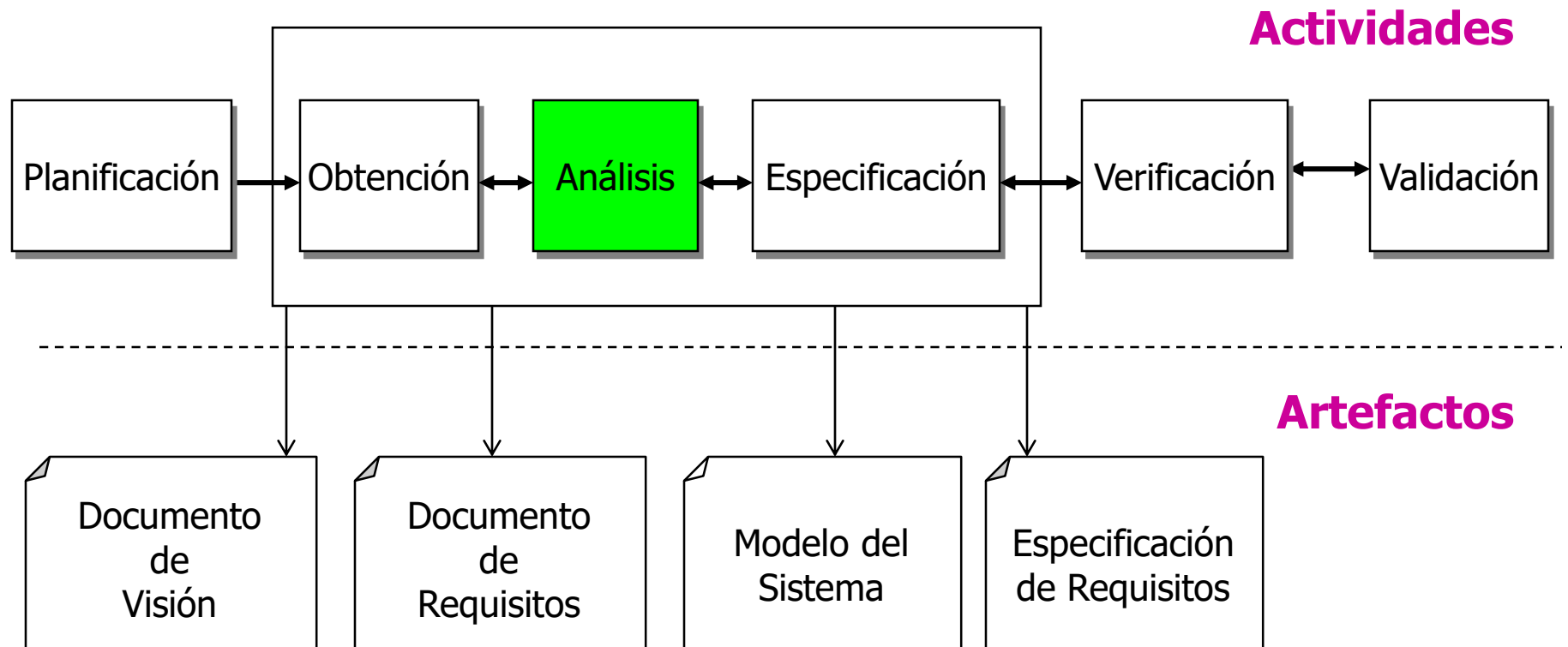
- Implementación parcial, permite a los desarrolladores y usuarios:
 - entender mejor los requisitos
 - cuáles son necesarios, deseables
 - acotar riesgos
- **Prototipo desechable**: El propósito es solo establecer que algo se puede hacer, luego se parte de cero en la construcción, quedando el conocimiento aprendido
- **Prototipo evolutivo**: Es implementado sobre la arquitectura del producto final, el sistema final se obtiene de evolucionar el prototipo
- Aspectos para los que es frecuente construir prototipos:
 - Apariencia y percepción de la interfaz de usuario
 - Arquitectura (riesgos tecnológicos, tiempos de respuesta)
 - Otros aspectos riesgosos

Mismos datos, pero...



Análisis de Requisitos

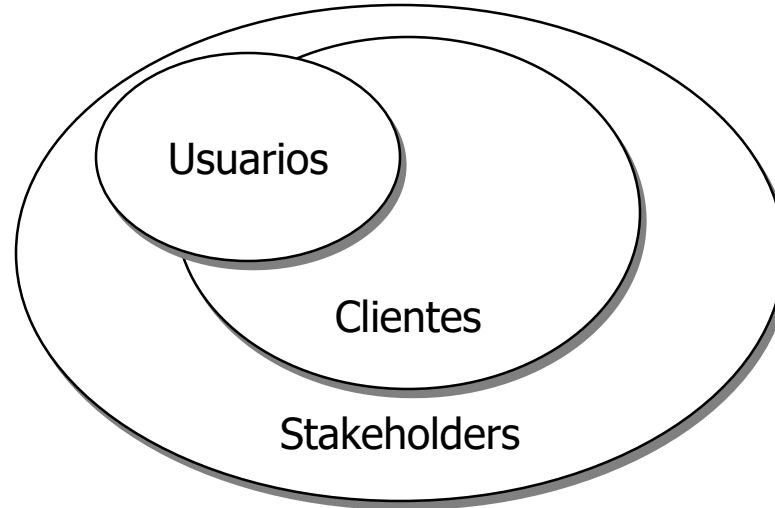
Proceso de Requisitos



Análisis de Requisitos

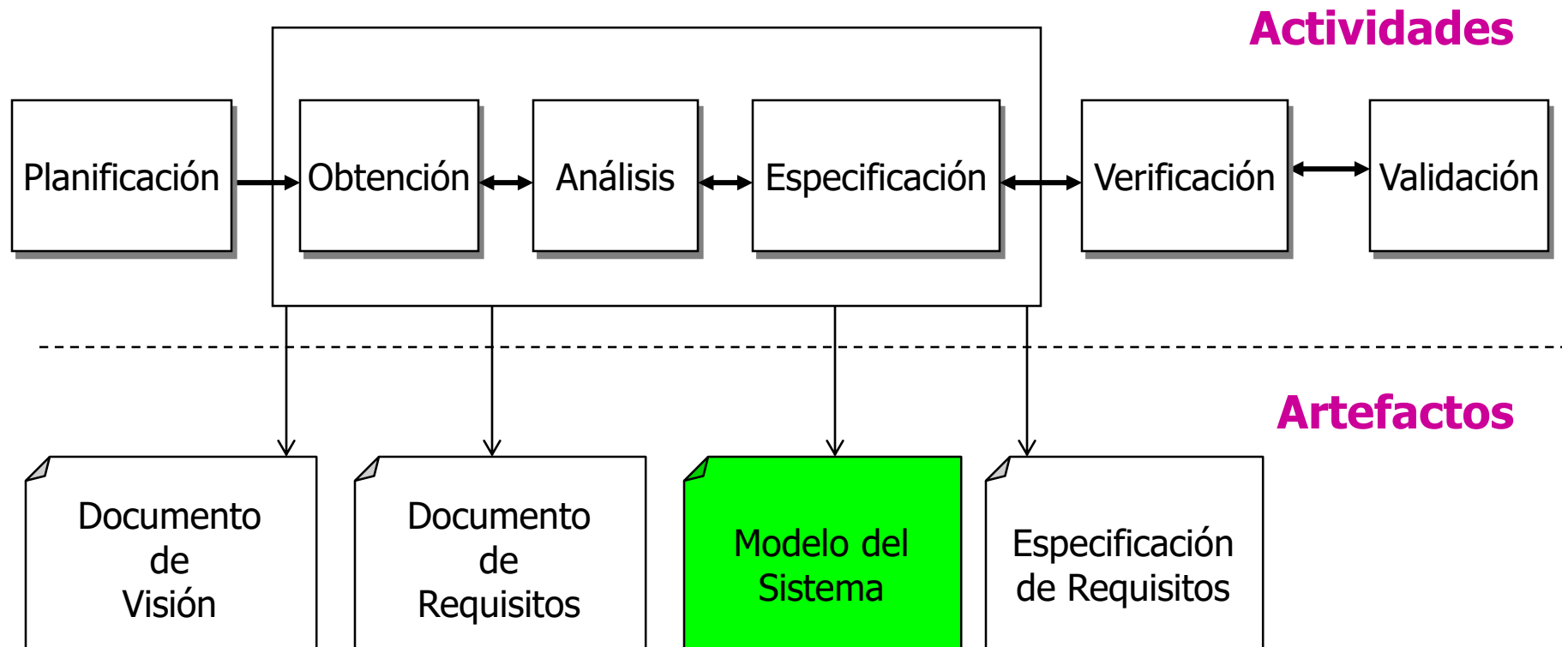
- Analizar stakeholders / clientes / usuarios
- Crear vistas
- Detallar
- Negociar prioridades
- Buscar reqs que faltan
- Evaluar factibilidad técnica - Prototipos
- Evaluar riesgos de requerimientos – En el Plan

Stakeholders / Clientes / Usuarios



- Clientes:
 - Definir responsable de:
 - resolución de conflictos
 - validación
 - Planificar reuniones de revisión de avance con el responsable.
 - Definir proceso de resolución de conflictos pe. en alcance.
- Usuarios:
 - dividirlos en clases
 - definir representantes
 - definir prototipos
 - acordar responsabilidades y estrategias de colaboración con representantes

Proceso de Requisitos



Modelos o Vistas del Sistema

- Glosario
- Modelos gráficos:
 - Modelo conceptual
 - Diagramas de estado – para entidades complejas que pasen por distintos estados.
 - Diagramas de Flujo de Datos (DFD)
- Prototipos de interfaz gráfica. – Definir docs del prototipo (reqs, diseño, CP)
- Casos de Prueba
- Tablas de Decisión
- Redes de Petri
- Casos de Uso

Diagramas

- BPM
- UML
 - Diagramas de Casos de Uso
 - Diagramas de Actividad
 - Diagramas de Estado
 - Modelo de dominio (Diagrama de Clases)
 - Modelo Conceptual



Tablas de Decisión

- Descripción dinámica
- Conjunto de **condiciones posibles** en un cierto instante
- **Estados** donde se verifica una combinación determinada de las condiciones
- **Acciones** a tomar

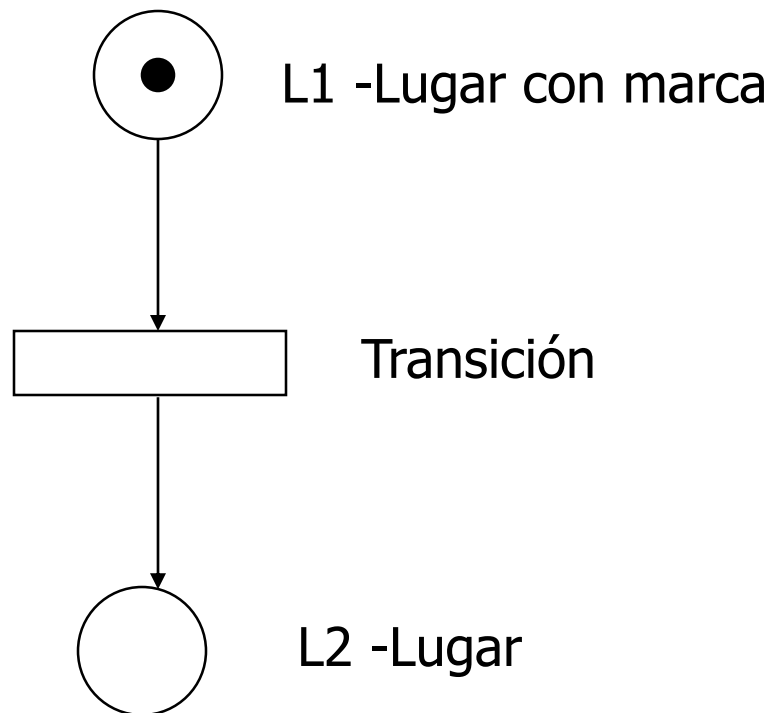
| | | Estados | | | | |
|-------------|-----------------------|---------|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Condiciones | Importe > 1000 | F | F | V | V | V |
| | Buenos Antecedentes | V | F | V | V | F |
| | Ya operó antes | - | - | V | F | - |
| Acciones | Autorizar Crédito | X | | X | | |
| | Analizar antecedentes | | X | | X | X |

- $\text{Nro estados} = 2^{\text{nro condiciones}} \Rightarrow$ tablas extensas

Redes de Petri

- Descripción dinámica
- Permiten describir:
 - Concurrencia
 - Sincronización
- Grafo dirigido con dos tipos de nodos:
 - Lugares  y Transiciones 
 - Arcos sólo pueden unir lugares con transiciones y transiciones con lugares

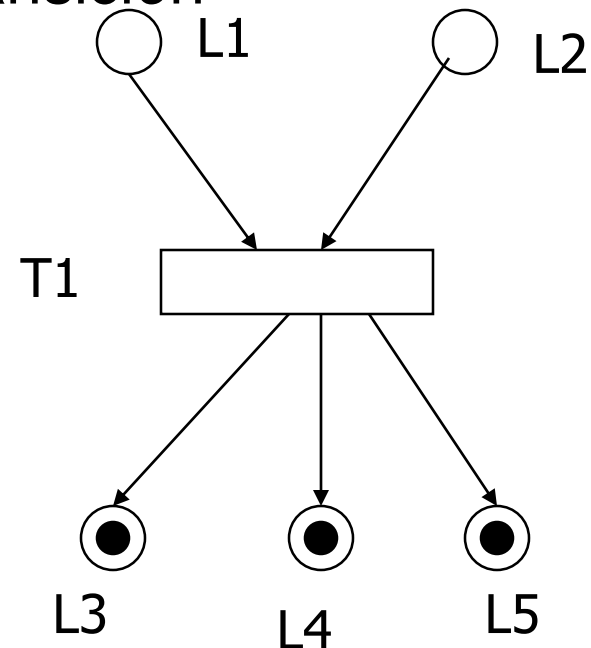
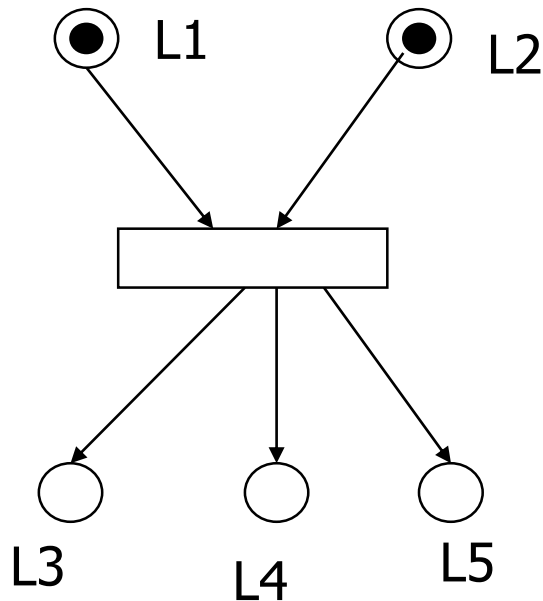
Redes de Petri (2)



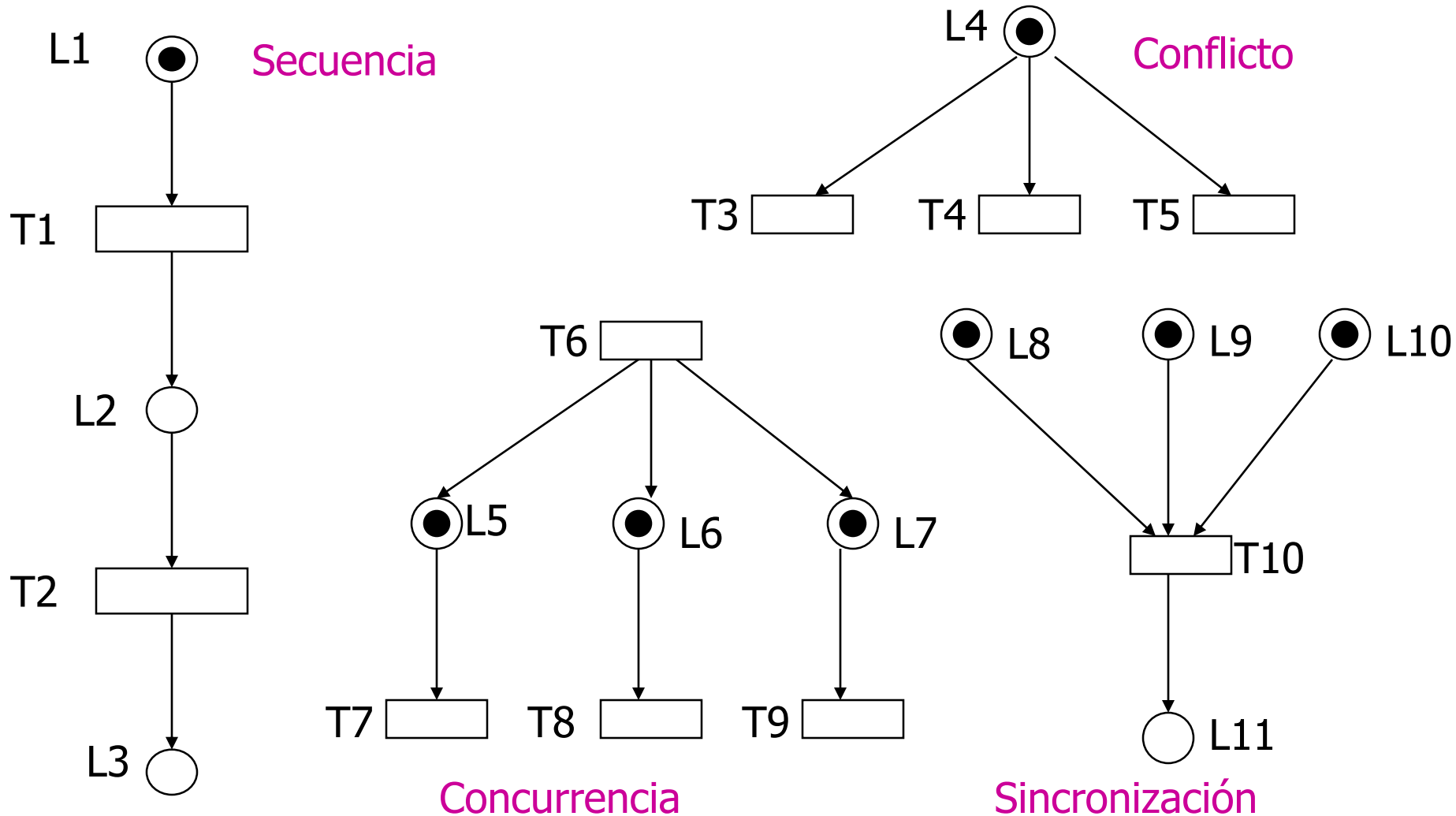
- El estado inicial de una red de Petri se le llama marca, está dado por los Tokens (marcas) iniciales
- Significado:
 - Transiciones: Modelan eventos o acciones
 - Lugares con marca: Cumplimiento de una condición
 - Transición activada: Ocurrencia del evento o ejecución de la acción

Redes de Petri (3)

- Transición habilitada: Existe al menos un token en cada uno de sus lugares de entrada
- Al activarse una transición, los tokens que activaron la transición desaparecen de los lugares de entrada y se generan tokens en los lugares de salida de la transición
- Estado pronto para activar la transición

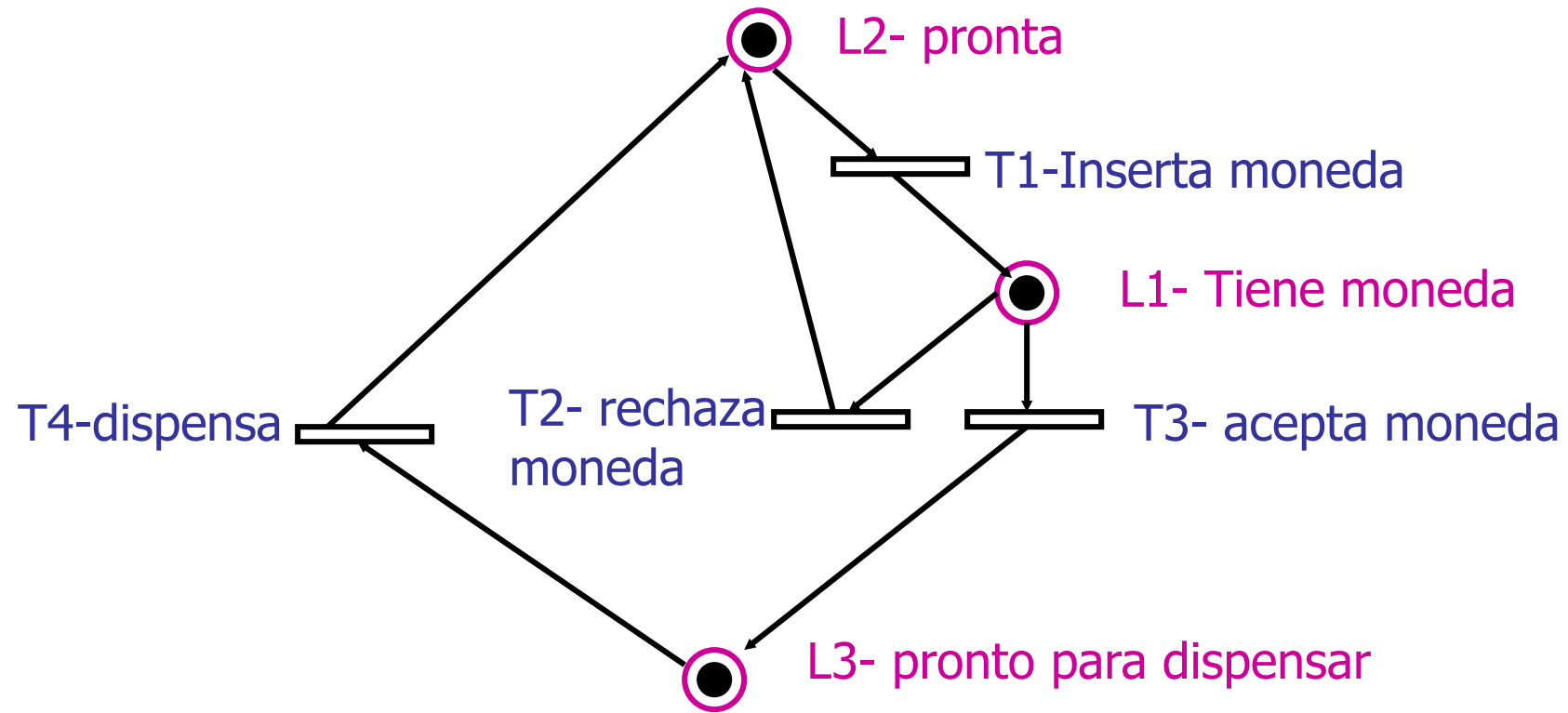


Redes de Petri – Algunas Primitivas



Redes de Petri –

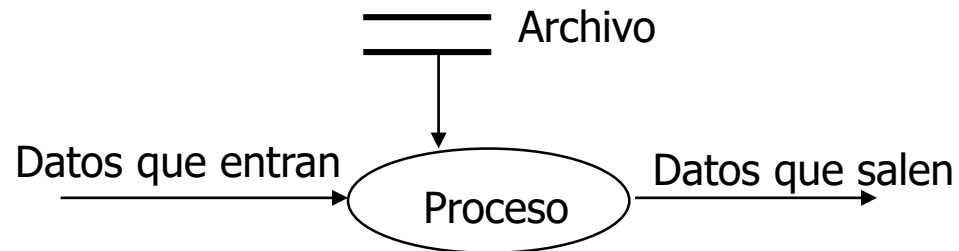
Ejemplo: Máquina Dispensadora



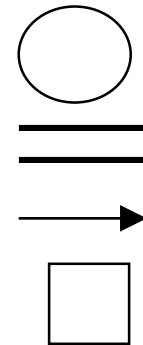
Se dispararon las transiciones: t1,t3,t4
Otra secuencia posible seria t1,t2

Diagrama de Flujo de Datos (DFD)

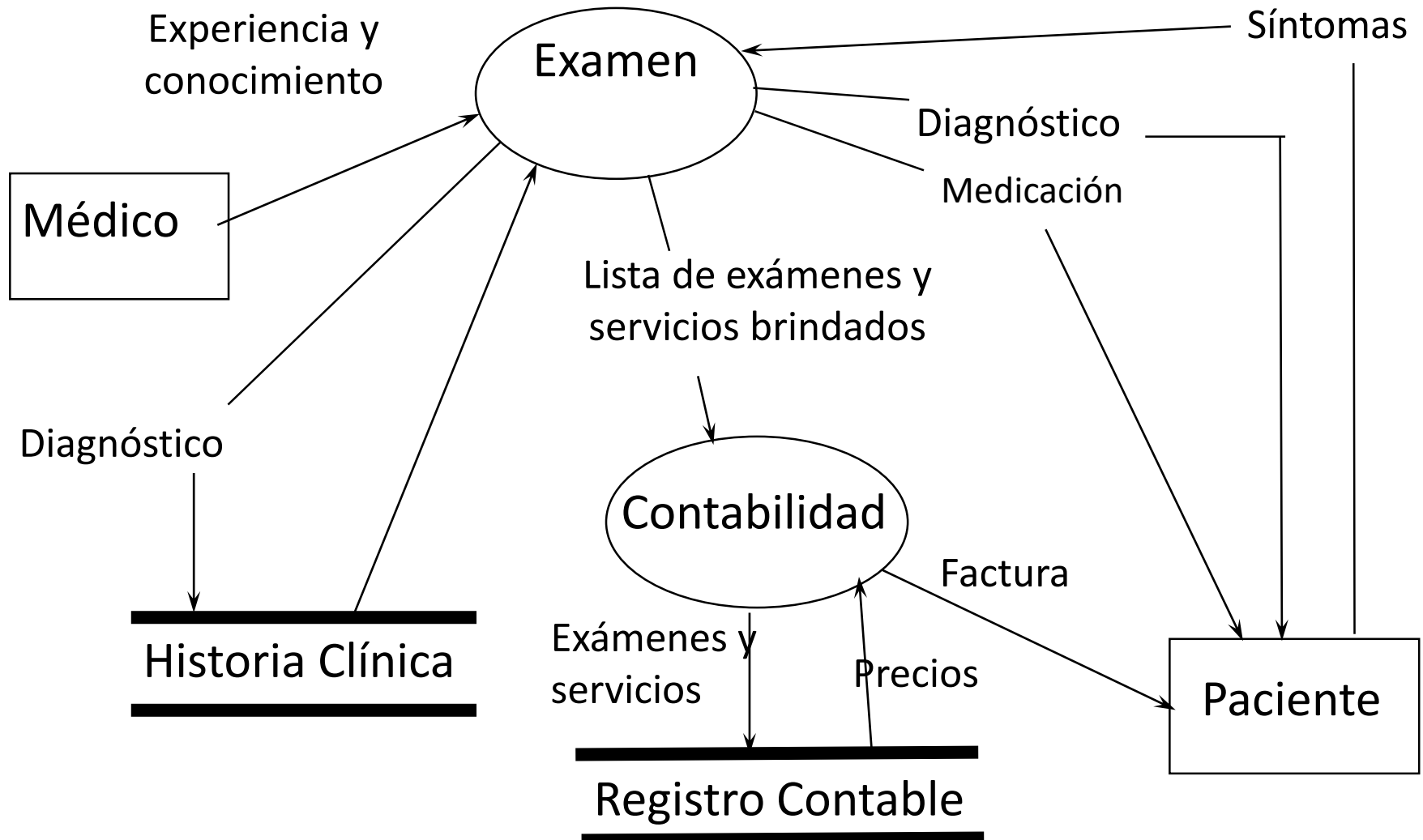
- Descripción dinámica
- Proviene de Metodología de Análisis y Diseño Estructurado
 - Fin de la década del 70
 - Usados en versión original de OMT (Rumbaugh 91), no incorporados a UML
 - Antes de los Casos de Uso era una de las formas más usadas para describir un sistema



- Elementos
 - Proceso del sistema que recibe datos y genera otros
 - Archivo de datos (almacenamiento)
 - Flujo de Datos
 - Entidad Externa al sistema a modelar (actor)



DFD - Ejemplo



DFD

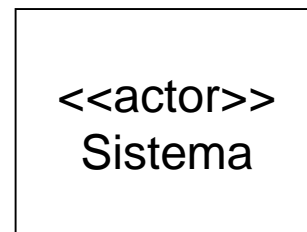
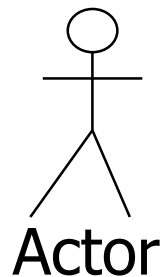
- Permite visualizar cómo fluye la información por el sistema
 - está asociado a una realización particular del sistema
- El diagrama no es suficiente para precisar el comportamiento:
 - por un flujo que entra a un proceso desde un archivo,
¿fluye un registro o todo el archivo?
 - No estipula sincronización, un flujo llega a una entidad externa y otro sale ¿Están relacionados? ¿Uno es respuesta del otro?
- No permite definir procesos (bucles, condiciones, etc.)
- Se complementa con un diccionario de datos que describe:
 - estructura de los flujos y otros detalles
 - los procesos (lenguaje natural estructurado) con lo que el comportamiento queda determinado
- Permite distintos niveles de abstracción, anidando DFDs
- A menudo sistemas legados están documentados con DFD

Casos de Uso

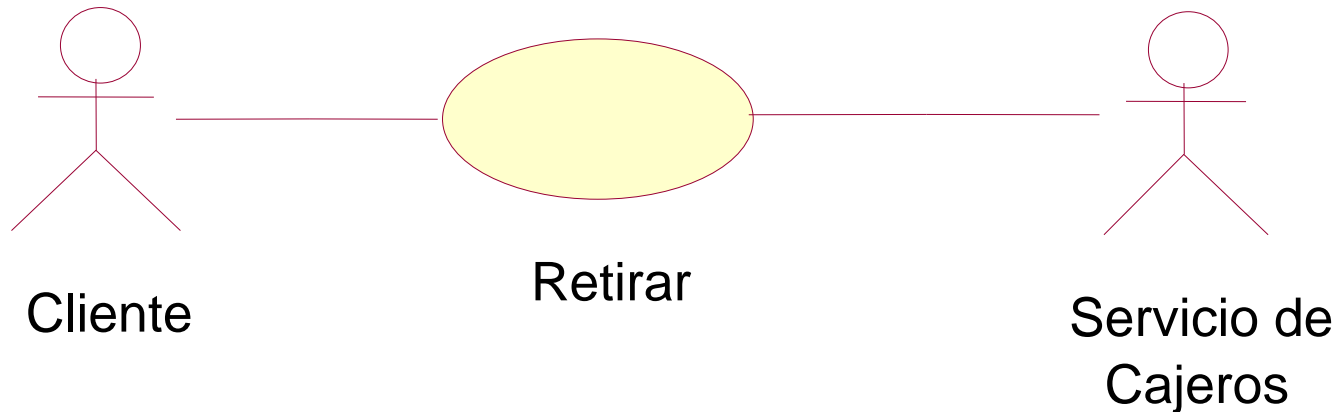
- Técnica para entender y describir requisitos
- Los casos de uso son requisitos, describen requisitos funcionales
- Pone el acento en el uso del producto
- Describen como el sistema debe comportarse desde el punto de vista del usuario
- Casos de Uso como caja negra: Especifican que es lo que el sistema debe hacer sin especificar cómo debe hacerlo
- Se describen mediante documentos de texto
- Introducido por Ivar Jacobson (1992)

Actor

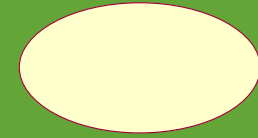
- Entidad externa que interactúa con el sistema (persona identificada por un rol o sistema externo)
- Actor principal: Sus objetivos son cumplidos al realizar el caso de uso
- Los actores son externos al sistema que vamos a desarrollar.
- Al identificar actores estamos delimitando el sistema
- Usuario: persona que cuando usa el sistema, asume un rol.



Cajero Automático - Ejemplo



- **Actor principal:** Cliente
- **Actores:** Servicio de Cajeros
- **Caso de Uso:** Retirar
- **Descripción:** Un cliente de un banco retira dinero de una cuenta a través del cajero automático utilizando una tarjeta bancaria, el Servicio de Cajeros verifica que el PIN sea válido y que el monto de la cuenta sea suficiente para realizar el retiro



- **Escenario:**
 - Secuencia de acciones e interacciones entre los actores y el sistema, dando un resultado de valor observable para un actor particular
 - También se conoce como instancia de caso de uso
 - Es una forma particular de usar el sistema, un camino a través de un caso de uso.
- **Caso de uso:** conjunto de escenarios posibles que puede encarar un actor (o varios) con el sistema para el logro de cierto objetivo.
- “Un resultado observable de valor” se basa en entregar sistemas que hagan lo que las personas realmente necesitan.

Caso de Uso: Retirar

Flujo principal:

1. Cliente inserta una tarjeta bancaria en el lector del CA.
2. El CA lee el código de la tarjeta y verifica que es correcto
3. El CA pide el código de PIN de 4 dígitos
4. EL Cliente ingresa el PIN
5. El CA envía código de Tarjeta y PIN al SC
6. El SC verifica que el PIN sea correcto y contesta: OK
7. El CA despliega las distintas alternativas disponibles: retiro, depósito, consulta
8. El Cliente elige Retiro
9. El CA pide cuenta y monto
10. El Cliente los ingresa
11. CA envía código de Tarjeta, PIN, cuenta y monto al SC
12. El SC contesta: OK
13. El CA dispensa el dinero
14. El CA devuelve la tarjeta
15. El CA imprime el recibo

Caso de Uso : Retirar

Flujo principal: (otra forma)

| Cliente | Sistema | Servicio de Cajeros |
|--|---|--------------------------------------|
| 1. Inserta una tarjeta bancaria en el lector del CA. | | |
| | 2. Lee el código de la tarjeta y verifica que es correcto | |
| | 3 Pide el código de PIN de 4 dígitos | |
| 4 Ingresa el PIN | | |
| | 5 – Envía Id. De tarjeta y PIN | |
| | | 6 – Verifica que el PIN sea correcto |
| | 7- Despliega las distintas alternativas disponibles | |
| 8- Elige la opción: Retiro | | |
| | 9. Pide cuenta y monto | |
| 10- Ingresa cuenta y monto | | |
| | 11. Envía al SC el Id. Tarjeta, PIN, cuenta y monto | |
| | | 12 Contesta: Continuar (OK) |
| | 13 Dispensa el dinero | |
| | 14 Devuelve la tarjeta | |
| | 15 Imprime recibo | 67 |

Casos de Uso

- Forma de encontrarlos: Mirar cada uno de los actores del sistema y preguntarse que es lo que buscan cuando usan el sistema.
- Recomendación: Identificarlos con un verbo.
- Cada caso de uso modela partes de la dinámica.
- Diagrama de Casos de Uso – descripción estática.
- Los casos de uso son independientes del método de diseño que se utilice, y por lo tanto del método de programación, no son parte del análisis OO, pero son una excelente entrada para ello.
- Los casos de uso pueden dirigir el proceso de desarrollo. Guían el diseño, la implementación y la prueba del sistema.

Casos de Uso - Conceptos

- **Precondiciones:** Establece que cosas deben ser siempre verdaderas antes de comenzar un caso de uso. No se verifican dentro del caso de uso ya que se asume que son verdaderas dentro de él.
- **Poscondiciones:** Establece que cosas ocurren al completar el caso de uso.
- **Flujo principal:** Describe el escenario del caso de uso de mayor interés para el actor. Típicamente no incluye condiciones ni bifurcaciones.
- **Flujos alternativos:** Son todos los otros escenarios; son bifurcaciones en el flujo principal.
- **Requisitos Especiales:** Son los requisitos no funcionales, atributos de calidad o restricciones específicas relacionadas con el caso de uso.

Caso de Uso : Retirar

Flujos Alternativos :

2A. La tarjeta no es válida

1. El CA devuelve la tarjeta con el mensaje “tarjeta no válida”
2. Fin CU

6A. PIN inválido y menos de 3 intentos

El Cliente puede realizar tres intentos para ingresar el PIN válido. Sino, el CA retiene la tarjeta.

1. El SC contesta indicando PIN inválido
2. El CA muestra el mensaje “PIN incorrecto” y sigue en punto 3

6B. PIN inválido y 3 intentos

El CA debe retener la tarjeta

1. El SC contesta indicando PIN inválido
2. El CA muestra el mensaje “Se le retiene la tarjeta”
3. Fin CU

9A. El CA no tiene dinero

1. La opción “Retiro” en esta situación no es una alternativa posible, y el CA despliega la advertencia: “Sin dinero”.
2. Fin CU

Caso de Uso : Retirar

Flujos Alternativos (cont.):

11A. Monto insuficiente para el cajero

El monto indicado por el cliente no puede obtenerse a partir de los billetes de que dispone el CA

- 1 El CA despliega el mensaje “No se cuenta con ese monto en este cajero”
- 2 Vuelve a 9.

12A. No hay suficiente saldo en la cuenta.

1. CA despliega mensaje “Su saldo no permite extraer ese monto”
2. El CA devuelve la tarjeta
3. Fin CU

12B. No hay contacto con el Servicio de Cajeros (SC)

1. CA despliega el mensaje “sin conexión a la red de cajeros”
- 2 . El CA devuelve la tarjeta
3. Fin CU

12C. Enlace con el computador central se cae durante la transacción

Hay que asegurar que el SC considera sólo los retiros efectivamente realizados

14A. El dinero no es retirado de la bandeja.

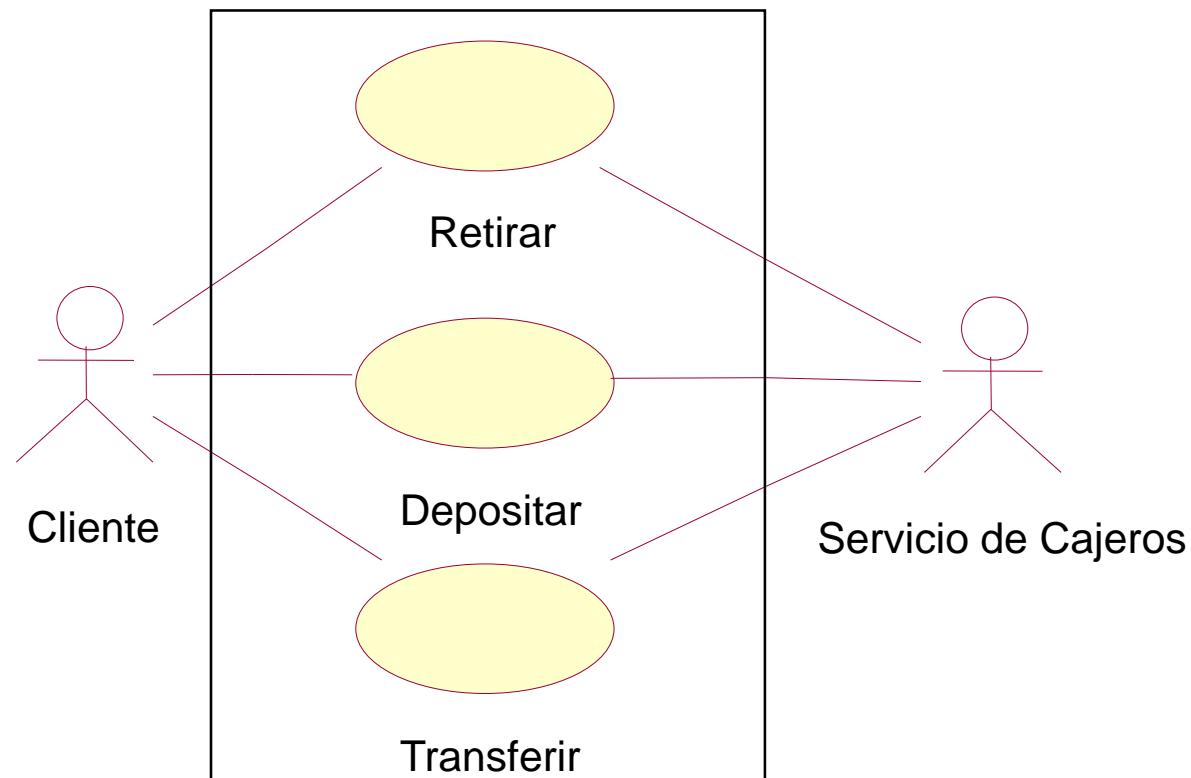
1. Si después de YY segundos el dinero está todavía en la bandeja, el CA lo recupera y lo deja en el depósito de dinero usado
1. Sigue en 14

14B. La tarjeta se tranca al intentar devolverla.

1. CA trata de devolverla durante xx segundos.
2. Si en ese tiempo no puede devolverla, CA avisa a mantenimiento
3. Fin CU

Diagrama de Casos de Uso

- UML provee notación para los casos de uso para ilustrar los actores, los casos de uso y las relaciones entre ellos
- Permite realizar un Diagrama del Contexto del Sistema
- Muestra los bordes del sistema

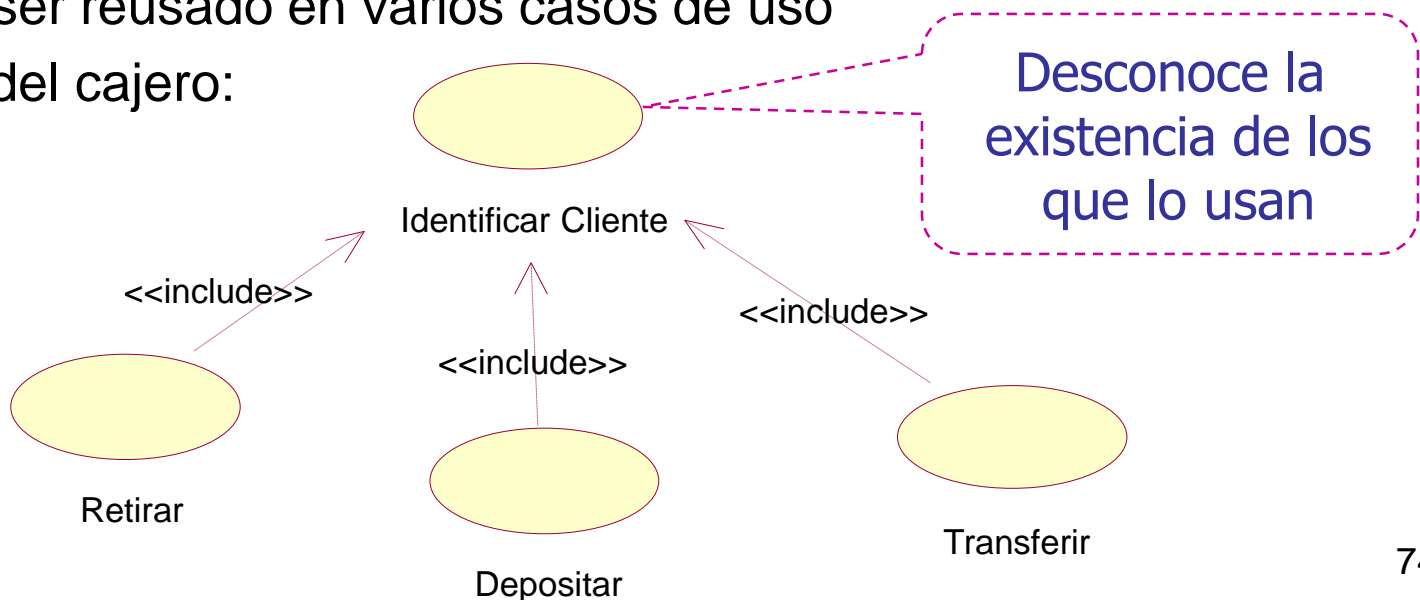


Construcción del Modelo - Pasos

- Definir frontera
- Identificar Actores
- Para cada Actor, identificar qué cosas quiere hacer
 - cada uno va a determinar un caso de uso
 - darle un nombre
- Dado un caso de uso
 - Identificar si participan otros actores
 - Describirlo brevemente de forma narrativa, centrándose en el flujo principal (distintas variantes de presentación y contenido)
- Una vez definido el conjunto de casos de uso relevante:
 - Refinarlos incluyendo condiciones especiales
 - Identificar casos de uso comunes y particulares (“incluye” y “extiende”), generalización

Relaciones entre CU – Include

- Escenarios comunes a más de un caso de uso
- El caso de uso incluido no depende del caso de uso base
- Cuando una instancia del caso de uso «llega al lugar» donde el comportamiento de otro caso de uso debe ser incluido, ejecuta todo el comportamiento descrito por el caso de uso incluido y luego continúa de acuerdo a su caso de uso original.
- El caso de uso incluido representa comportamiento encapsulado que puede ser reusado en varios casos de uso
- En el caso del cajero:



Caso de Uso: Retirar

Flujo principal:

1. **Incluye** el caso de uso: Identificar Cliente
2. El CA despliega las distintas alternativas disponibles: retiro, depósito, consulta
3. El Cliente elige Retiro
4. El CA pide cuenta y monto
5. El Cliente los ingresa
6. CA envía código de Tarjeta, PIN, cuenta y monto al SC
7. El SC contesta: OK
8. El CA dispensa el dinero
9. El CA devuelve la tarjeta
10. El CA imprime el recibo

Caso de Uso: Identificar Cliente

Descripción Breve:

Verifica que la tarjeta y el PIN sean válidos

Flujo Principal:

1. Cliente inserta una tarjeta bancaria en el lector del CA.
2. El CA lee el código de la tarjeta y verifica que es correcto
3. El CA pide el código de PIN de 4 dígitos
4. EL Cliente ingresa el PIN
5. El CA envía código de Tarjeta y PIN al SC
6. El SC verifica que el PIN sea correcto y contesta: OK

Flujos Alternativos:

2A. La tarjeta no es válida

1. El CA devuelve la tarjeta con el mensaje “tarjeta no válida”
2. Fin CU

6A. PIN inválido y menos de 3 intentos

El Cliente puede realizar tres intentos para ingresar el PIN válido. Sino, el CA retiene la tarjeta.

1. El SC contesta indicando PIN inválido
2. El CA muestra el mensaje “PIN incorrecto”
3. Sigue en punto 3

6B. PIN inválido y 3 intentos

El CA debe retener la tarjeta

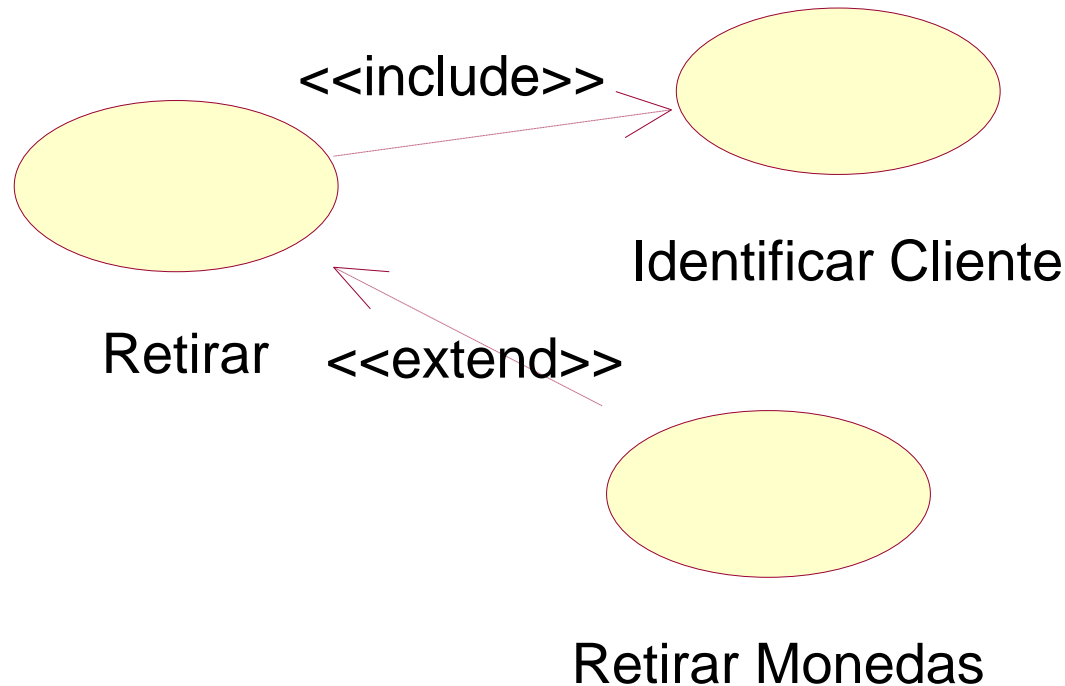
1. El SC contesta indicando PIN inválido
2. El CA muestra el mensaje “Se le retiene la tarjeta”
3. Fin CU

Relaciones entre CU – Extend

- Es un fragmento de un caso de uso, que agrega comportamiento a otro caso de uso.
- Se usan para explicar escenarios que sería complejo presentar como flujo alternativo, o que se desea destacar.
- Representan una parte de la funcionalidad del caso que no siempre ocurre (condicional).
- Se ejecuta solo si la condición se cumple.
- El caso de uso extendido referencia a su caso de uso base.
- Punto de extensión: Punto dentro del caso de uso, donde se puede insertar comportamiento adicional.
- Al terminar el caso de uso extendido, se vuelve al caso de uso base, en la sentencia siguiente al punto de extensión.

Extend - Ejemplo

- El cliente puede querer retirar monedas además de billetes



Caso de Uso : Retirar

Flujo principal:

1. **Incluye** el caso de uso: Identificar Cliente
2. El CA despliega las distintas alternativas disponibles: retiro, depósito, consulta
3. El Cliente elige Retiro
4. El CA pide cuenta y monto
5. El Cliente los ingresa
6. CA envía código de Tarjeta, PIN, cuenta y monto al SC
7. El SC contesta: OK
8. El Cliente pide dispensar el dinero
9. El CA dispensa el dinero
10. El CA devuelve la tarjeta
11. El CA imprime el recibo

Puntos de Extensión:

Retiro de Monedas: En el punto 8 del flujo principal

Caso de Uso: Retirar Monedas

Descripción Breve: El cliente opcionalmente puede querer retirar monedas

Punto de extensión
indicado por un nombre

Flujo Principal:

Extensión de **Retirar** en el punto **Retirar Monedas**, el cliente también puede elegir “monedas”, en ese caso:

1. El Cliente elige retirar monedas, especificando tipos de monedas y la cantidad de rollos para cada uno.
2. El CA calcula el importe a retirar para cada moneda y el total y lo muestra
3. El Cliente confirma

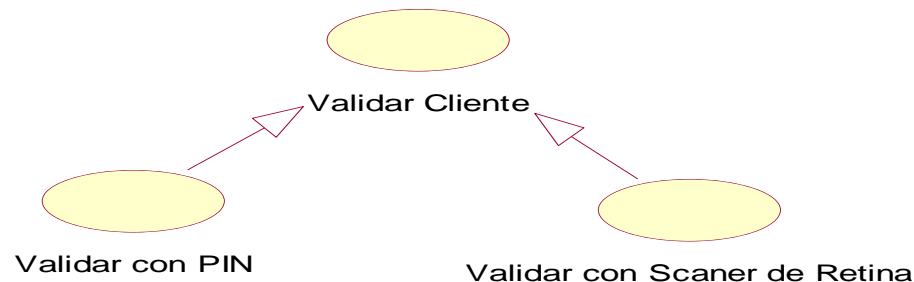
Flujos Alternativos:

3A El cliente puede querer cambiar la selección, se vuelve a 1

G1 El cliente cancela el retiro de monedas. Fin CU Retirar Monedas

Relaciones entre CU – Generalización

- Algunas veces existe más de un escenario principal para un caso de uso
- Se puede crear un caso de uso abstracto, crear un caso de uso para cada escenario principal y que estos hereden del caso abstracto
- El caso de uso hijo hereda los escenarios, puntos de extensión y relaciones definidos en el caso de uso padre
- El caso de uso hijo puede definir nuevas operaciones, como también redefinir o enriquecer con nuevas secuencias de acciones operaciones ya existentes en el caso de uso padre



Actividades

- Encontrar actores y casos de uso
- Priorizar los casos de uso
- Detallar un caso de uso
- Estructurar el modelo de casos de uso

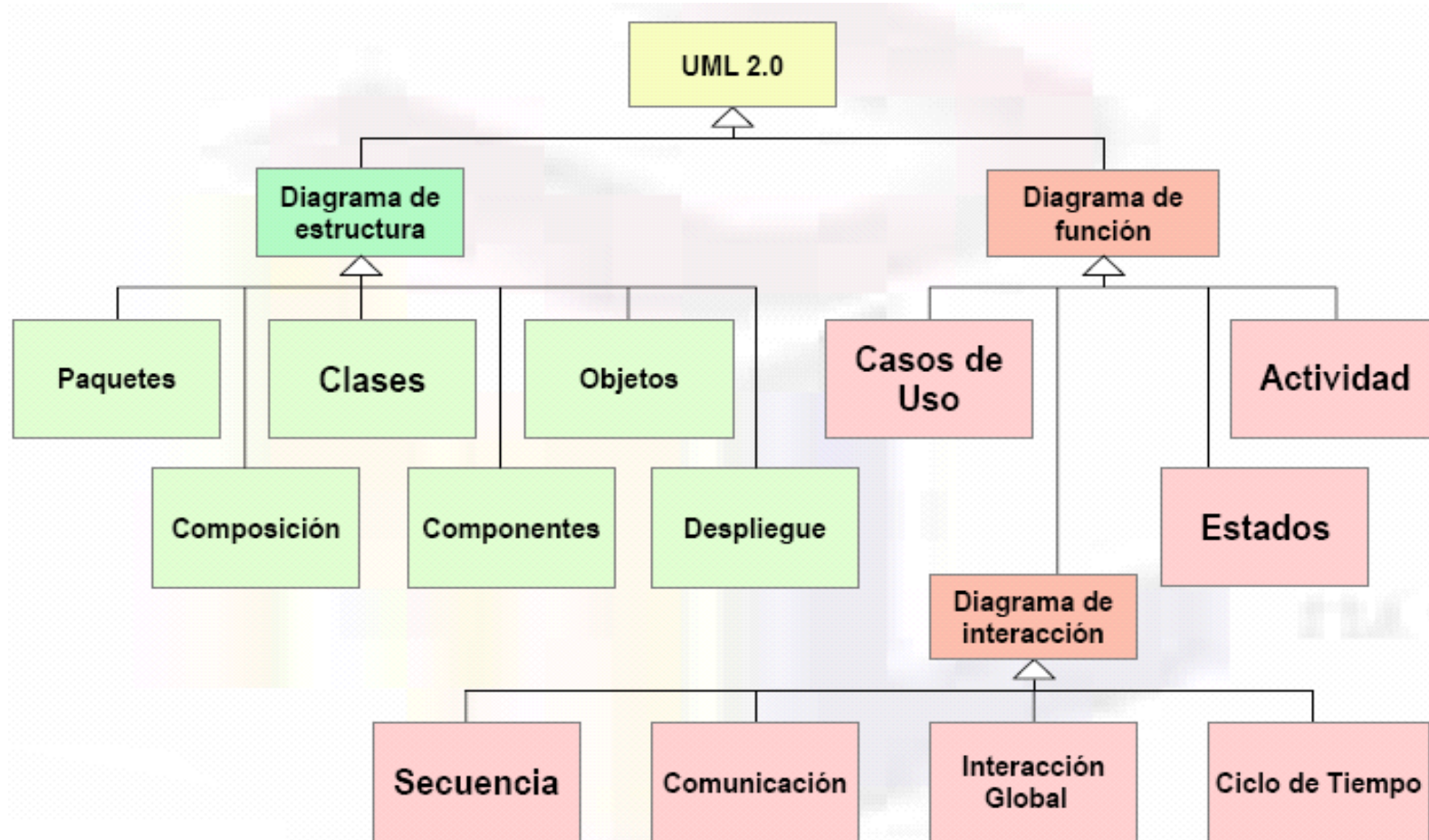
Casos de Uso - Nivel de detalle

¿A qué nivel se deben expresar los CU en el análisis de requerimientos?

- Enfocarse en CU al nivel de Proceso de Negocio Elemental (**elementary business process (EBP)**):
Una tarea ejecutada por una persona en un lugar en un determinado momento, en respuesta a un evento del negocio, que agrega valor de negocio medible y deja los datos en estado consistente.
- Que cumpla un objetivo del usuario.
- Error común: definir muchos CU a nivel demasiado bajo: por cada paso o subtarea dentro de un EBP.
- Excepciones: Pe. casos de uso incluidos

- Unified Modeling Language
- Objetivo: Proveer un lenguaje común que puede ser usado para el desarrollo de software
- Lenguaje que permite:
 - Visualizar: La comunicación es a través de gráficos
 - Especificar: construyendo modelos para el análisis, diseño, implementación
 - Construir: Permite la generación de código a partir de un modelo UML, y la construcción de un modelo a partir del código (ingeniería reversa)
 - Documentar: Permite la documentación completa de todo el sistema
- Aprobado como estándar por la OMG en 1997
- Actualmente se encuentra en la versión 2.1.2 (nov 2007)

Diagramas en UML



Tipos de Diagramas

- **Modelo Estático**

- Construye y documenta los aspectos estáticos de un sistema.
- Refleja la estructura básica y estable de un sistema software.
- Crea una representación de los principales elementos del dominio del problema

- **Modelo Dinámico**

- Crea los diagramas que muestran el comportamiento de un sistema
- Para requisitos se utilizan los siguientes diagramas:
 - Diagrama de Casos de Uso
 - Diagrama de Clases (Modelo Conceptual)
 - Diagrama de Actividad
 - Diagramas de Estados

Diagrama de Casos de Uso

- Permite visualizar en una forma compacta los casos de uso del sistema y que actores participen en cada caso de uso
- Presenta las relaciones que existen entre los casos de uso
- Muestra los límites del sistema
- Visión estática de los Casos de Uso de un sistema
- Consta de los siguientes elementos:
 - Actor
 - Caso de Uso
 - Relaciones
 - Include
 - Extend
 - Generalización

Diagrama de Casos de Uso - Ejemplo

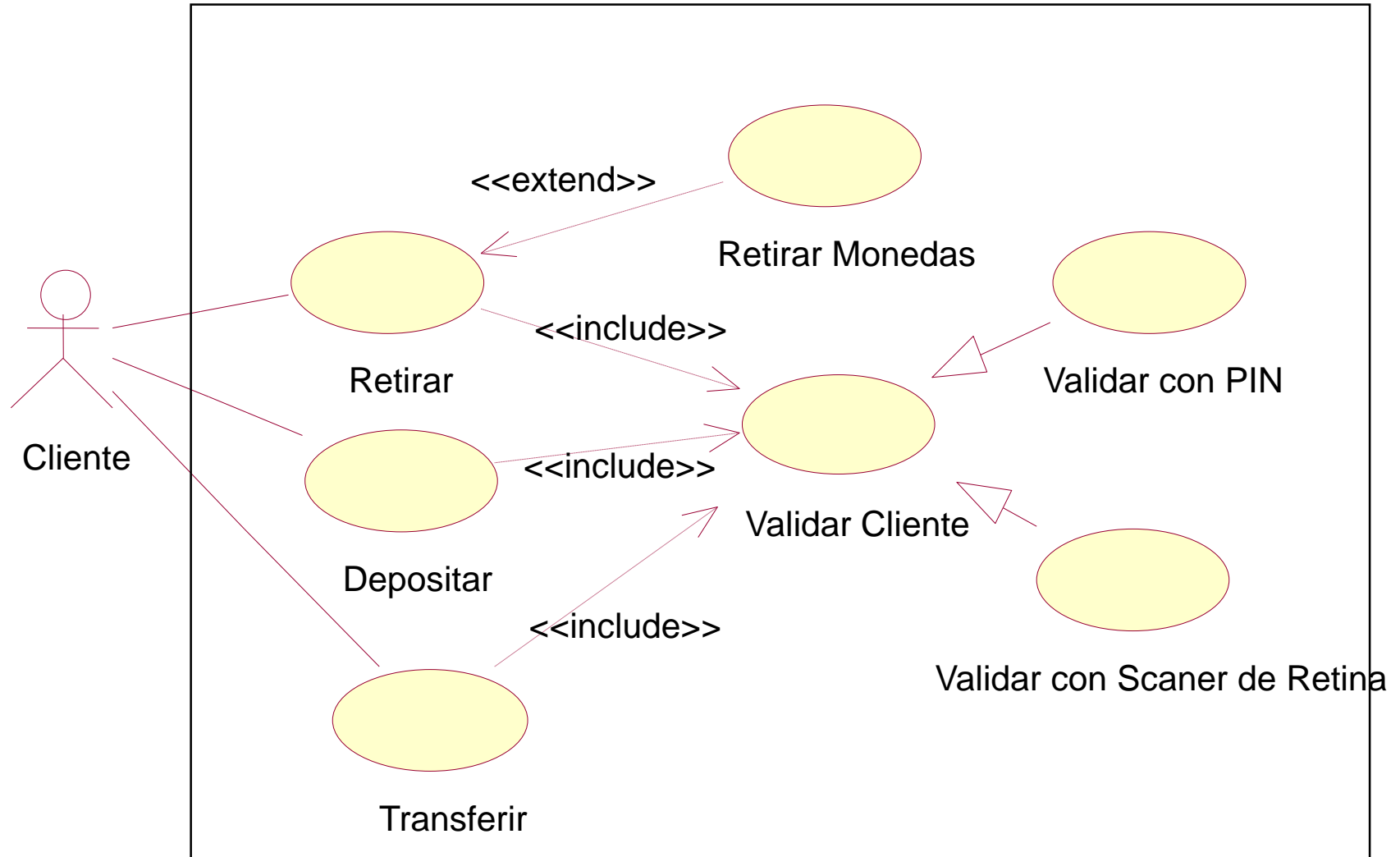


Diagrama de Clases

| |
|--------------|
| Nombre Clase |
| Atributos |
| |

- Muestra las clases e interfaces que componen el sistema y las relaciones que existen entre ellas
- Muestra aspectos estáticos
- Clase: conjunto de objetos que comparten:
 - Atributos
 - Operaciones
 - Relaciones
 - Semántica
- **Modelo de Dominio (Conceptual):** ayudan a entender los conceptos del dominio del problema y el vocabulario del mismo. Se excluyen detalles referentes a la implementación o al lenguaje de programación.
- **Diagramas de clases de implementación:** muestran todos los métodos y atributos necesarios para implementar cada clase. Es un diagrama dependiente de la implementación y del lenguaje.

Modelo del Dominio (Conceptual)

- Permite describir las entidades que conforman el dominio, sus relaciones y atributos
- Se representan los conceptos del dominio
- Muestra aspectos estáticos

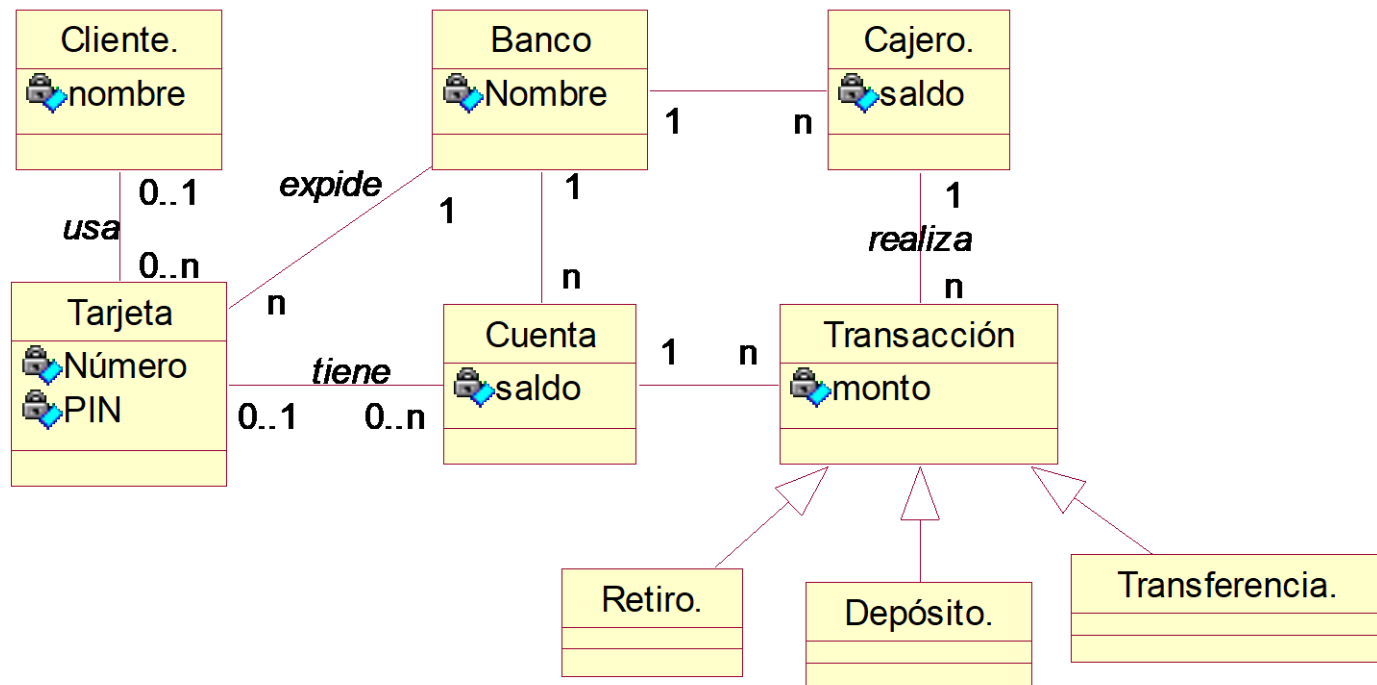
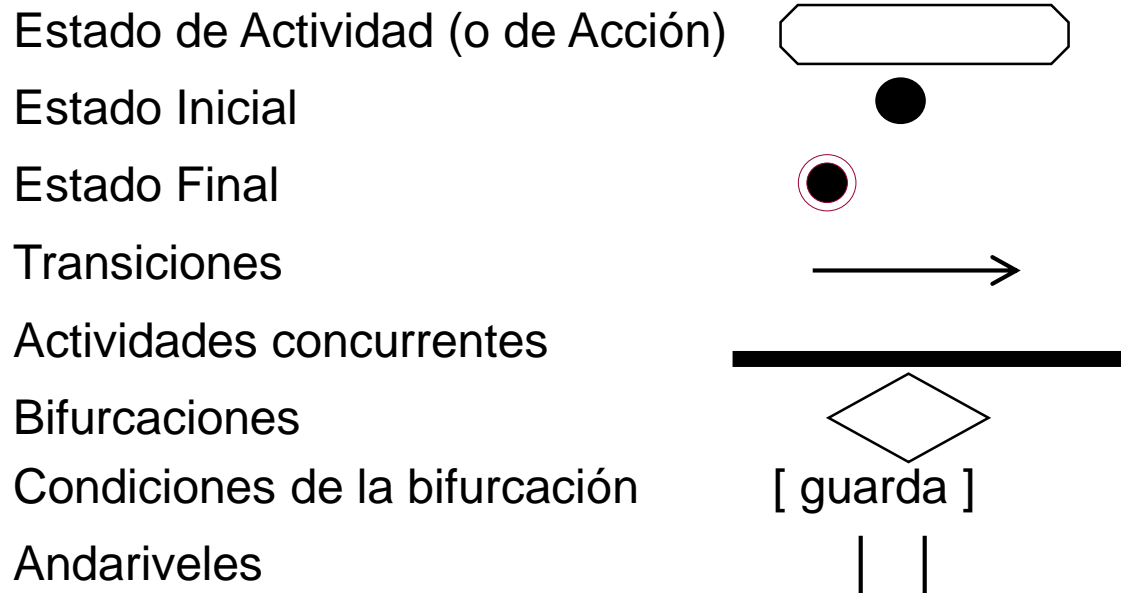


Diagrama de Actividad

- Se construye para modelar el flujo del control (workflow)
- Elementos:



- Permite modelar el flujo del trabajo
 - En un sistema
 - En una organización

Diagrama de Actividad - Ejemplo

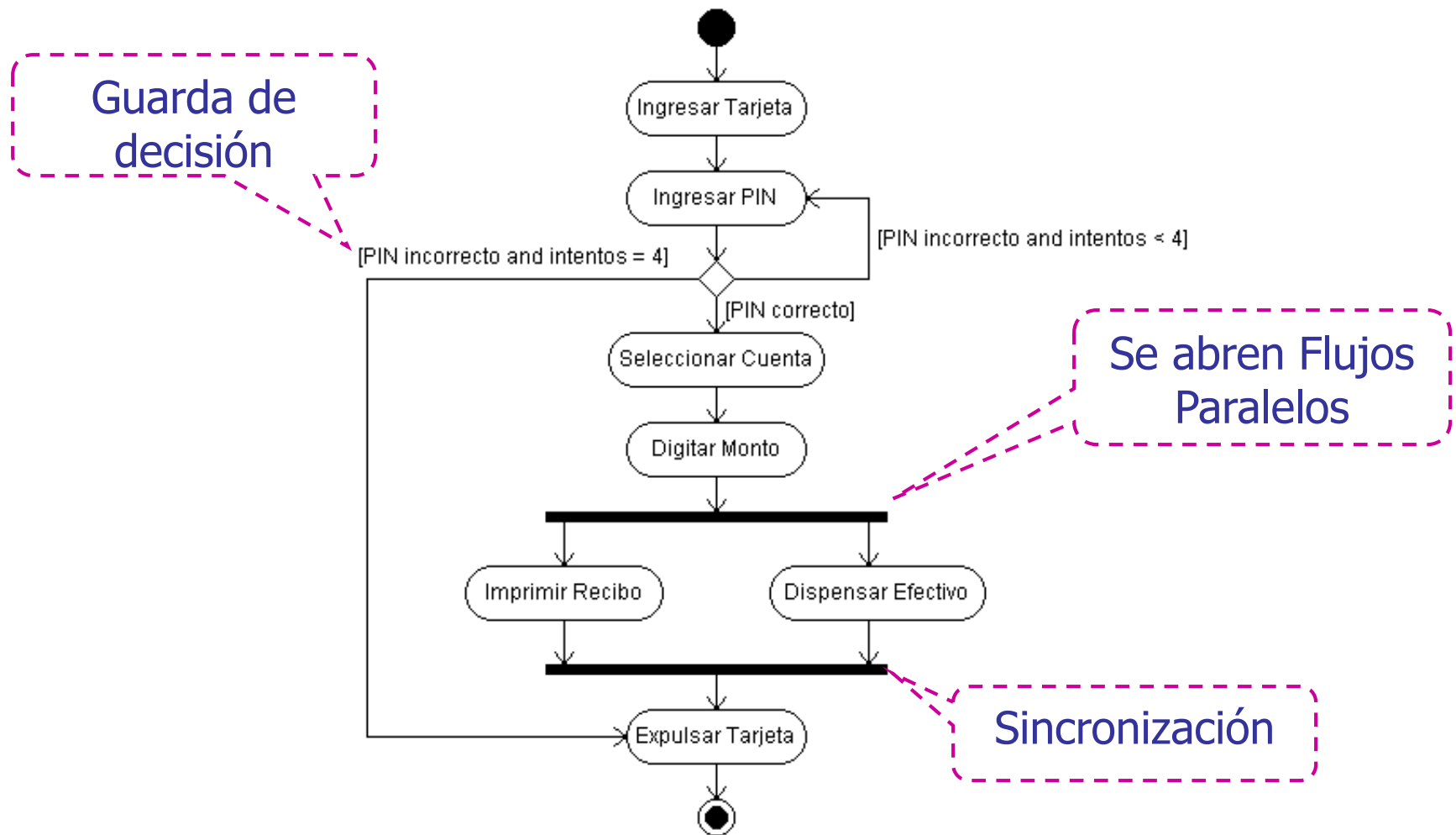


Diagrama de Estados

- Muestra el comportamiento de un objeto representando los estados en que se puede encontrar y los eventos que le hace pasar de uno a otro.
- Se utiliza para:
 - Modelar el estado interno de una entidad
 - Modelar el estado de un caso de uso
- Da una vista dinámica del sistema
- Permite:
 - Anidamiento: un estado con subestados
 - Estados paralelos: reduce el nro. de estados necesarios en el modelo
 - Condiciones de bifurcación

Diagrama de Estados – Ejemplo 1

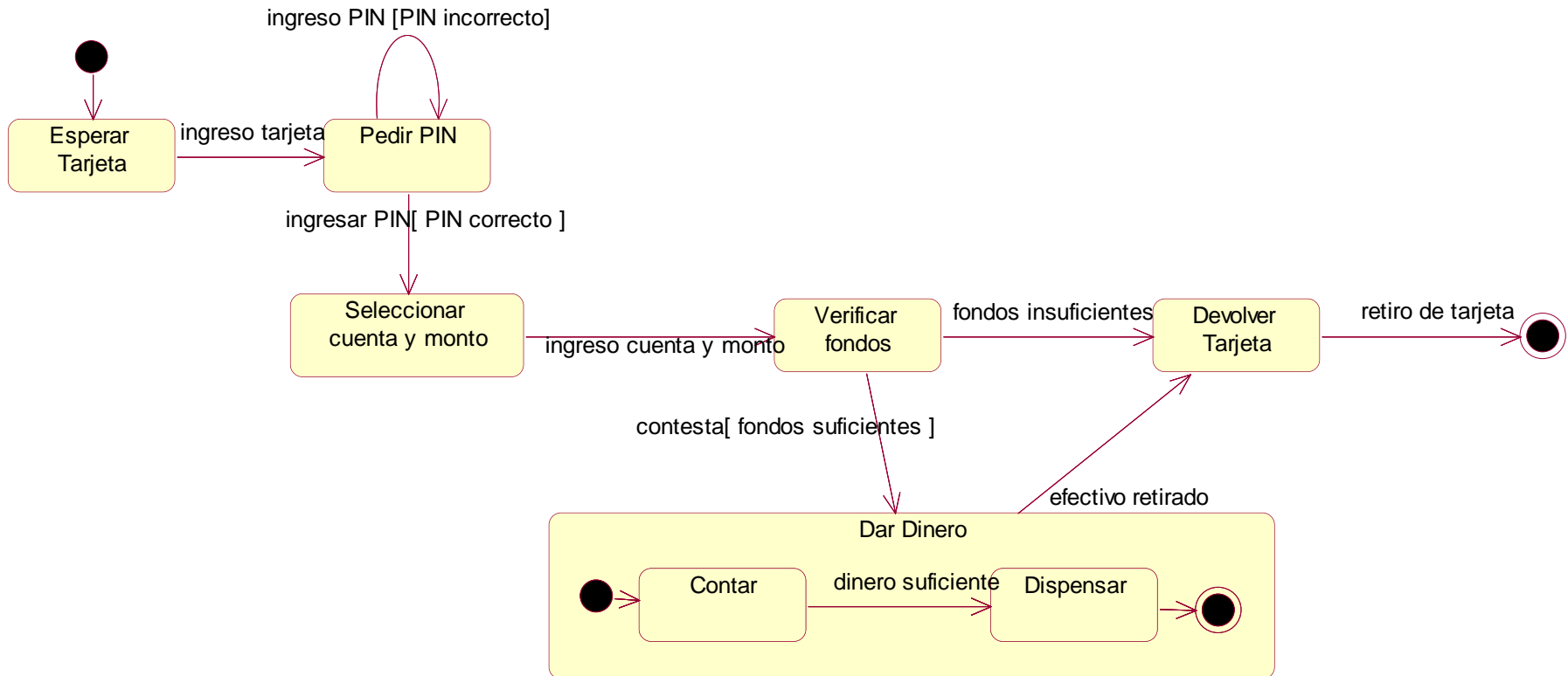
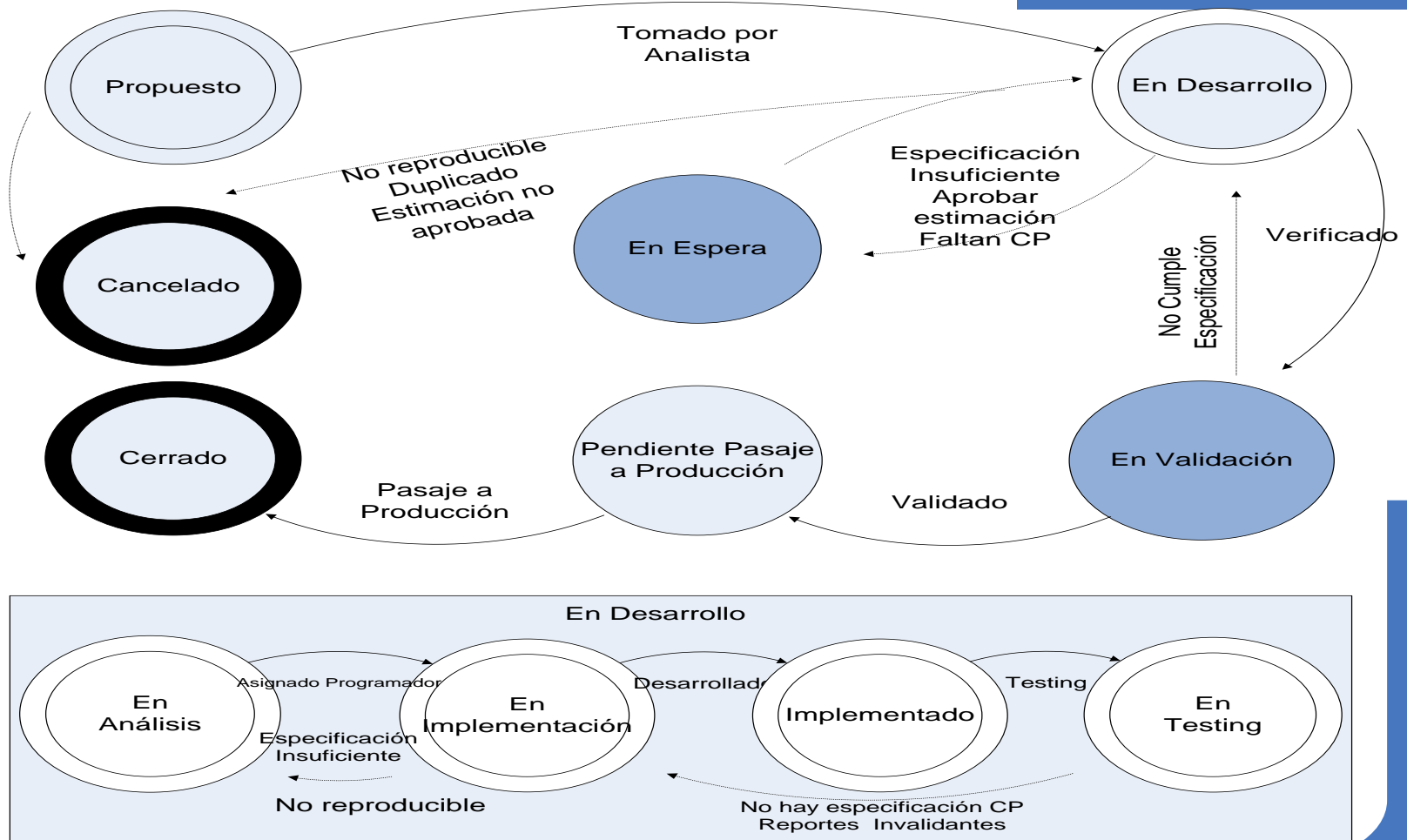


Diagrama de Estados –Ejemplo 2

Ciclo de Vida de Incidentes

Versión 0.8

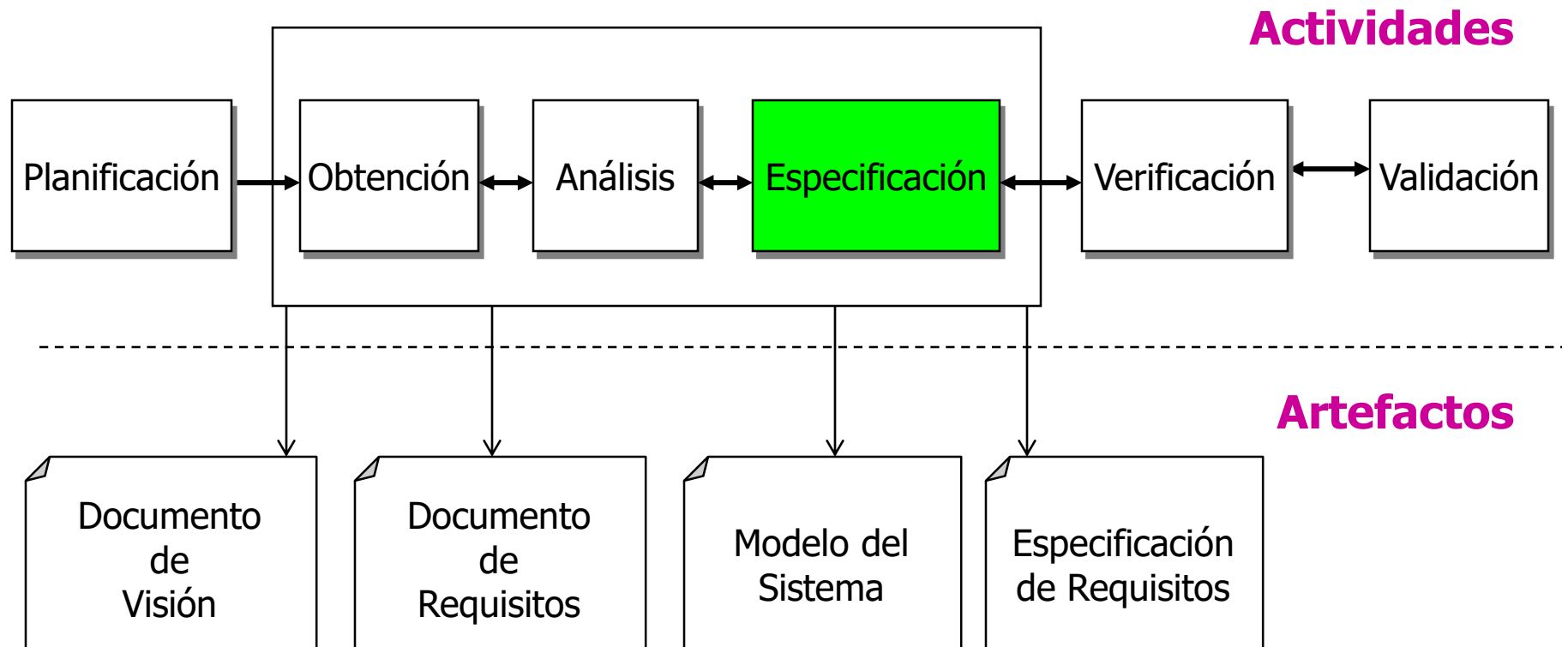


Elección de una Técnica para Modelar Requisitos

- No existe un único enfoque aplicable a todos los sistemas, depende de cada proyecto
- Puede ser necesario combinar varios enfoques

Especificación de Requisitos

Proceso de Requisitos



Lenguajes de Notación

- Lenguaje Natural
 - Comprensible para el Cliente/Usuario
 - Ambiguo (glosario)
 - Poca legibilidad (plantilla, formateo del texto)
 - Difícil de tratar (Verificar correctitud, consistencia, completitud)
- Notaciones Especiales (más formales)
 - Poca o ninguna ambigüedad
 - Facilita tratamiento
 - Necesidad de entrenamiento en la notación
 - Dificultades de comprensión por Cliente/Usuario

Notaciones Especiales

- Gráficas vs. Basadas en texto
- Estáticas vs. Dinámicas
- Descripciones Estáticas
 - Se especifican entidades y sus atributos, los requisitos se pueden ver como las relaciones entre las entidades.
 - No describe como cambian las relaciones con el tiempo
- Descripciones Dinámicas
 - Especifican estados y las transiciones entre estados en el tiempo

Documentación de requisitos

- Qué documentar:
 - lo que hace el sistema actual
 - lo que el cliente pide
 - lo que el sistema va a hacer
 - criterios de aceptación
 - criterios de verificación
- Recomendaciones:
 - agrupar por temas
 - formular los reqs como reqs positivos y no negativos
 - expresarlos en voz activa y no pasiva
 - indicar si se está documentando solo lo que va en el alcance o todo lo que se pidió.
 - representar reqs. con múltiples vistas (ejemplo de los ciegos y el elefante).

Documentos de Requisitos

- **Definición de Requisitos:** lista completa de lo que el cliente espera que el sistema haga, escrita de forma que el cliente la pueda entender
 1. Se debe proveer un medio para acceder a archivos externos creados por otras herramientas
- **Especificación de Requisitos (SRS):** reformula la definición en términos técnicos para que los diseñadores puedan comenzar el diseño
 - 1.1 Se proveerá al usuario los recursos para definir el tipo de archivo externo
 - 1.2 Cada tipo de archivo tendrá una herramienta asociada y un ícono que lo identifica
 - 1.3 Cuando el usuario seleccione el ícono que representa un archivo externo, el efecto es aplicar la herramienta asociada con ese tipo de archivo al archivo seleccionado

Documentos de Requisitos (2)

- Usar un mismo documento: Entendimiento común entre Cliente, usuario, analistas, desarrolladores
- Usar dos documentos: Se debe aplicar Gestión de Configuración: necesaria para asegurar la correspondencia entre ambos (si existen por separado)
- Permite seguir la pista y correspondencia entre:
 - Definición de Requisitos
 - Especificación de Requisitos
 - Módulos de Diseño
 - Código que implementa los módulos
 - Pruebas para verificar la funcionalidad
 - Documentos que describen el sistema

Documento Definición de Requisitos

- Registrar los requisitos en los términos del cliente
 - 1. Delinear el propósito general del sistema: Incluir referencias a otros sistemas, glosario y abreviaciones
 - 2. Describir el contexto y objetivos del desarrollo del sistema
 - 3. Delinear visión global del sistema: Incluir restricciones generales
 - 4. Definir en detalle las características del sistema propuesto, definir la frontera del sistema e interfaces.
 - 5. Discutir el ambiente en el que el sistema va a operar (hardware, comunicaciones, personal).

Características de una Buena Especificación SRS (IEEE 830)

- **Correcta / Válida:** Todos los req. son requeridos en el sistema.
 - No existe herramienta que asegure esto.
 - Validado por el cliente (que efectivamente refleje sus necesidades).
 - Revisar que sea consistente contra otros documentos existentes (pe. especificación de reqs. del sistema).
- **No Ambigua:** Todo req tiene una única interpretación.
 - Incluir glosario.
 - No ambigua para quienes lo crearon y para quienes lo usan.
- **Completa:** Incluye:
 - Todos los requisitos asociados con funcionalidad, desempeño, restricciones de diseño, atributos o interfaces externas.
 - Definición de respuestas del sw a todo posible datos de entrada (válidos o inválidos) en toda clase de situaciones realizables.
 - No hay referencias sin definir en la especificación.
 - La frase “a determinar” indica SRS no completa. Ocasionalmente necesaria; describir:
 - condiciones que causan que no se sepa aún.
 - qué se debe hacer para determinar lo que falta, quién y cuándo.

Características de una Buena Especificación SRS (IEEE 830)

- Consistente internamente: Los requisitos no son contradictorios entre sí. Probables conflictos:
 - entre características de entidades. Pe. color de las luces, formatos distintos
 - conflicto lógico o temporal entre dos acciones. Pe. multiplicar o sumar; en forma simultánea o consecutiva.
 - diferentes términos para describir el mismo objeto.
- Ordenados por grado de importancia y/o estabilidad – identificador.
 - Importancia: esencial / deseado
 - Estabilidad: cantidad de cambios esperados
 - Necesidad: esencial / condicional / opcional
 - Esencial (condiciona aceptación del sw)
 - Condicional (valor agregado)
 - Opcional (puede o no valer la pena; se aceptan propuestas alternativas)

Características de una Buena Especificación SRS (IEEE 830)

- Verificable: Un requerimiento es verificable si existe un proceso finito de costo accesible para determinar que el sistema lo cumple
 - Usar términos concretos y cantidades medibles.
 - Preparar pruebas para demostrar que se cumplen. Si no se puede, eliminar o revisar el requisito.
- Modificables: Su estructura y estilo son tales que cualquier cambio en los requisitos puede ser hecho fácilmente en forma completa y consistente.
 - Organización coherente y fácil de usar (tablas, índices, refs. cruzadas)
 - No redundante.
 - Ventajas de redundancia: lo hace más legible.
 - Desventajas: difícil de mantener
 - Si la uso: referencias cruzadas
 - Expresar cada req. separadamente.

Características de una Buena Especificación SRS (IEEE 830)

- Trazables: El origen de cada requerimiento es claro, y es posible seguirle la pista en futuros desarrollos o mejora de la documentación
 - Trazabilidad hacia atrás: en versiones previas
 - Trazabilidad hacia adelante: documentos posteriores:
 - requiere IDENTIFICADOR ÚNICO.
-
- Realistas / Factibles
 - Ej.: tiempo de respuesta local=remoto
 - Ej.: El cliente quiere adelantarse a la tecnología
 - Entendibles: Tanto por los usuarios como por los desarrolladores

Características de una Buena Especificación SRS (IEEE 830)

- Ejemplos ambigüedad:
 - “La página web finalmente obtenida deberá permitir que toda su funcionalidad esté disponible utilizando los principales navegadores del mercado, como Explorer, Firefox, Opera y Chrome.”
 - “La solución deberá ser multiplataforma, encontrándose certificada para plataformas Linux, AIX y Windows.”
 - “Dada la naturaleza de este sistema de información, es requisito indispensable que la solución trabaje en alta disponibilidad.”

Características de una Buena Especificación SRS (IEEE 830)

- Ejemplos ambigüedad:
 - “La página web finalmente obtenida deberá permitir que toda su funcionalidad esté disponible utilizando los principales navegadores del mercado, como Explorer, Firefox, Opera y Chrome.”
 - ¿Qué versión de Firefox o de los otros navegadores?
 - “La solución deberá ser multiplataforma, encontrándose certificada para plataformas Linux, AIX y Windows.”
 - “Dada la naturaleza de este sistema de información, es requisito indispensable que la solución trabaje en alta disponibilidad.”

Características de una Buena Especificación SRS (IEEE 830)

- Ejemplos ambigüedad:

- “La página web finalmente obtenida deberá permitir que toda su funcionalidad esté disponible utilizando los principales navegadores del mercado, como Explorer, Firefox, Opera y Chrome.”
- “La solución deberá ser multiplataforma, encontrándose certificada para plataformas Linux, AIX y Windows.”
 - ¿Qué versión de “plataformas” Linux y demás? Porque de unas versiones a otras hay enorme diferencia.
- “Dada la naturaleza de este sistema de información, es requisito indispensable que la solución trabaje en alta disponibilidad.”

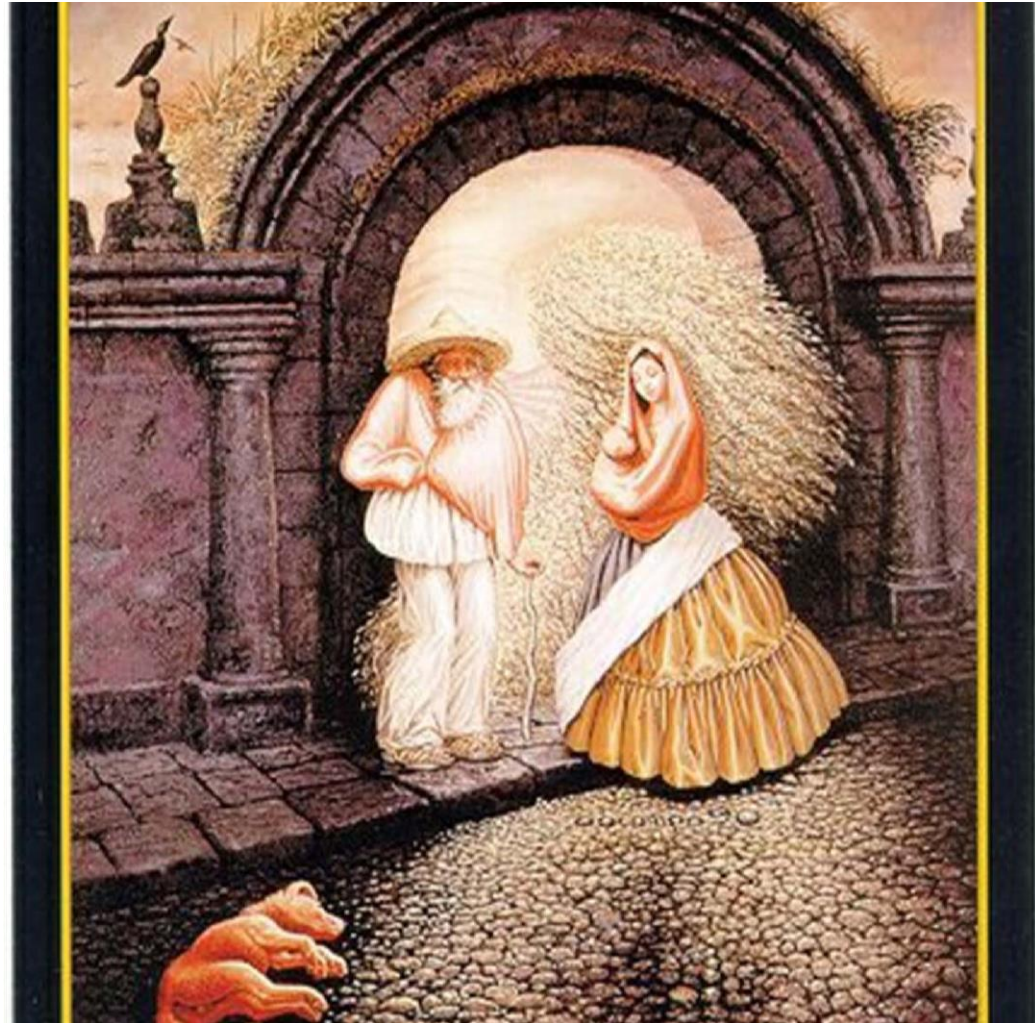
Características de una Buena Especificación SRS (IEEE 830)

- Ejemplos ambigüedad:

- “La página web finalmente obtenida deberá permitir que toda su funcionalidad esté disponible utilizando los principales navegadores del mercado, como Explorer, Firefox, Opera y Chrome.”
- “La solución deberá ser multiplataforma, encontrándose certificada para plataformas Linux, AIX y Windows.”
- “Dada la naturaleza de este sistema de información, es requisito indispensable que la solución trabaje en alta disponibilidad.”
 - ¿Alta disponibilidad? ¿en qué porcentaje de tiempo de funcionamiento / año? ¿un 90%? ¿98%? La diferencia es tremendamente enorme.

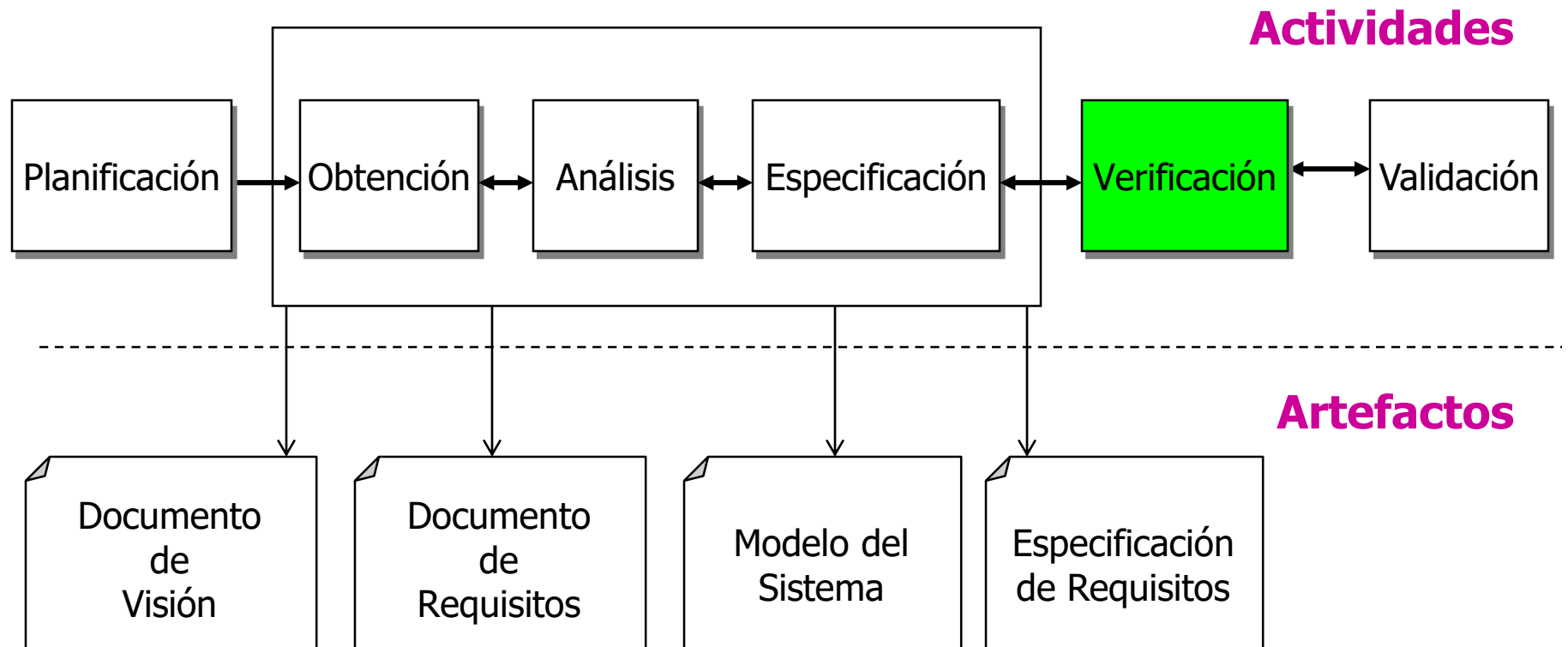
Características de una Buena Especificación SRS (IEEE 830)

- Dificultades para comprender los requerimientos:



Verificación de Requisitos

Proceso de Requisitos



Verificación de Requisitos

- Se verifica en el documento de requisitos:
 - Consistencia: que no haya contradicciones
 - Completitud: que no falte nada. Chequear por:
 - Omisiones. Hacer árboles de decisión para ver que estén todas las opciones detalladas.
 - Límites. Más claro con tabla, ahí se ve que no falta ninguno.
 - Necesidad
 - Ambigüedades
 - Realismo o Factibilidad: que se puedan implementar con la tecnología, presupuesto y calendario existentes.
 - Verificabilidad: que se pueda diseñar conjunto de pruebas para demostrar que el sistema cumple esos requisitos. Cuidado con adjetivos y adverbios.
 - Comprensibilidad: que los usuarios finales lo entiendan
 - Adaptabilidad: que el requisito se pueda cambiar sin afectar a otros.
 - Trazabilidad: que esté establecido el origen. Incluye verificar trazabilidad entre la especific. y el doc. de requisitos.

Verificación de Requisitos NO Funcionales

- Son difíciles de verificar
- Se deben expresar de manera cuantitativa utilizando métricas que se puedan probar de forma objetiva (esto es IDEAL)

| Propiedad | Medida |
|------------------|---------------------------------|
| Rapidez | Transacciones por seg |
| Tamaño | KB |
| Fiabilidad | Tiempo promedio entre fallas |
| Portabilidad | Número de sistemas, especificar |
| Facilidad de uso | Tiempo de capacitación |

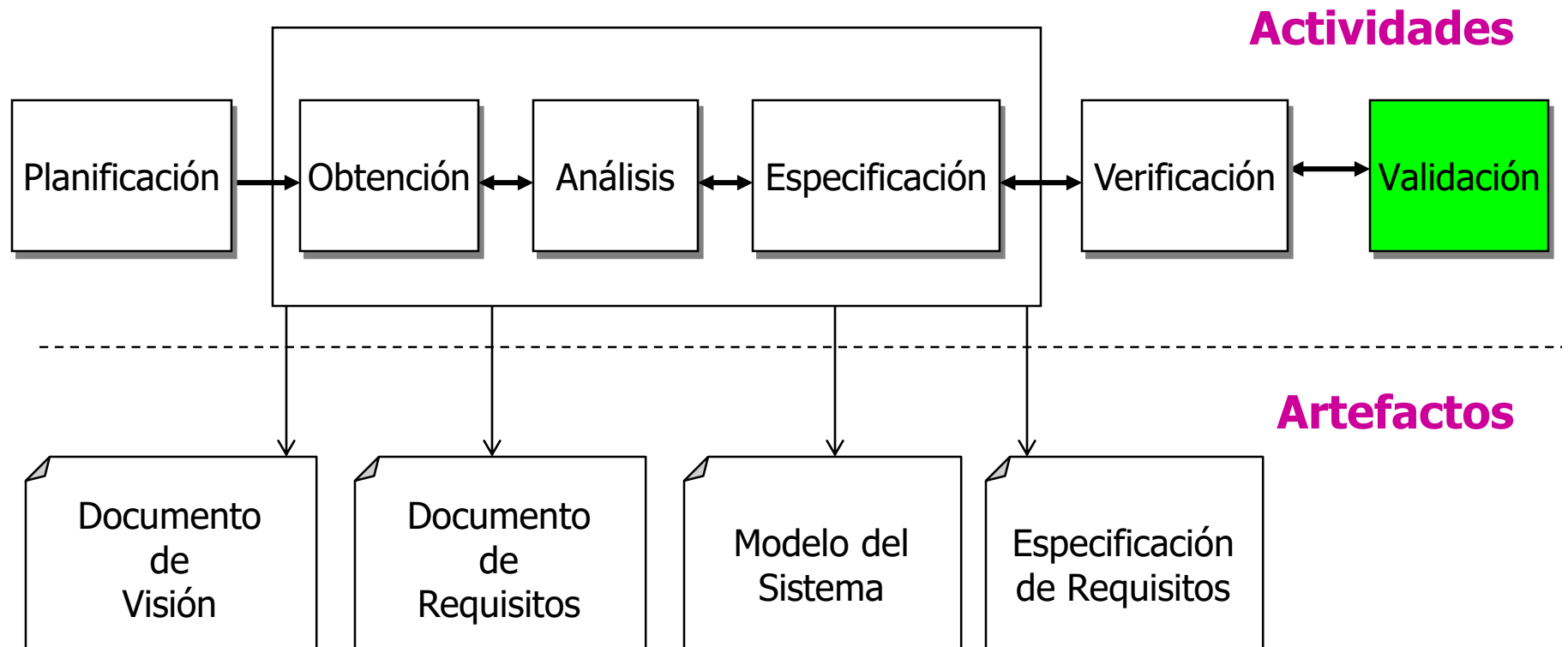
- Para los usuarios es difícil especificarlos en forma cuantitativa

Verificación de requisitos

- Definir quiénes, cómo, cuándo.
- Cómo:
 - revisiones formales (en grupo)
 - revisiones por pares
 - listas de comprobación

Validación de Requisitos

Proceso de Requisitos



Validación de Requisitos

- Proceso por el cual se determina si la especificación es consistente con las necesidades del cliente
- Se verifica en el documento de requisitos:
 - Validez: que el usuario valide qué es lo que quiere
- Planificar quién (qué stakeholder) va a validar qué artefacto cómo (técnica).
- Ejecutar
- Registrar – Reporte de validación / Firma

Validación de Requisitos

- Técnicas de Validación:
 - Manuales
 - Lectura, Referencias cruzadas Manuales
 - Instancias de validación formal: Entrevistas, Revisiones.
 - Listas de Comprobación
 - Modelos Manuales para verificar funciones y relaciones
 - Escenarios
 - Generación de Casos de Prueba
 - Pruebas Matemáticas: Si se usó un lenguaje formal, por ejemplo Z
 - Automatizadas
 - Referencias cruzadas automáticas: Si los requisitos se expresan formalmente, las herramientas CASE verifican su consistencia
 - Ejecución de Modelos
 - Construcción de Prototipos

Revisión de Requisitos

- Proceso manual. Se revisa el documento de requisitos buscando anomalías y omisiones:
 - Revisiones informales: discusión informal
 - Revisiones formales: se hace una “recorrida” del doc de req con el cliente, explicando implicancias de cada requisito.
- Participan representantes:
 - del cliente: operadores, quienes realicen entradas, utilicen salidas, y sus gerentes
 - del equipo de desarrollo: analistas de requisitos, diseñadores, encargados de pruebas y gestión de configuración
- Incluye:
 - revisar objetivos del sistema
 - evaluar alineamiento de requisitos con los objetivos (necesidad)
 - revisar el ambiente de operación y las interfaces con otros sistemas
 - funciones completas, restricciones realistas
 - evaluar riesgos
 - considerar:
 - pruebas del sistema
 - cambios en los requisitos en el proyecto, su verificación y validación

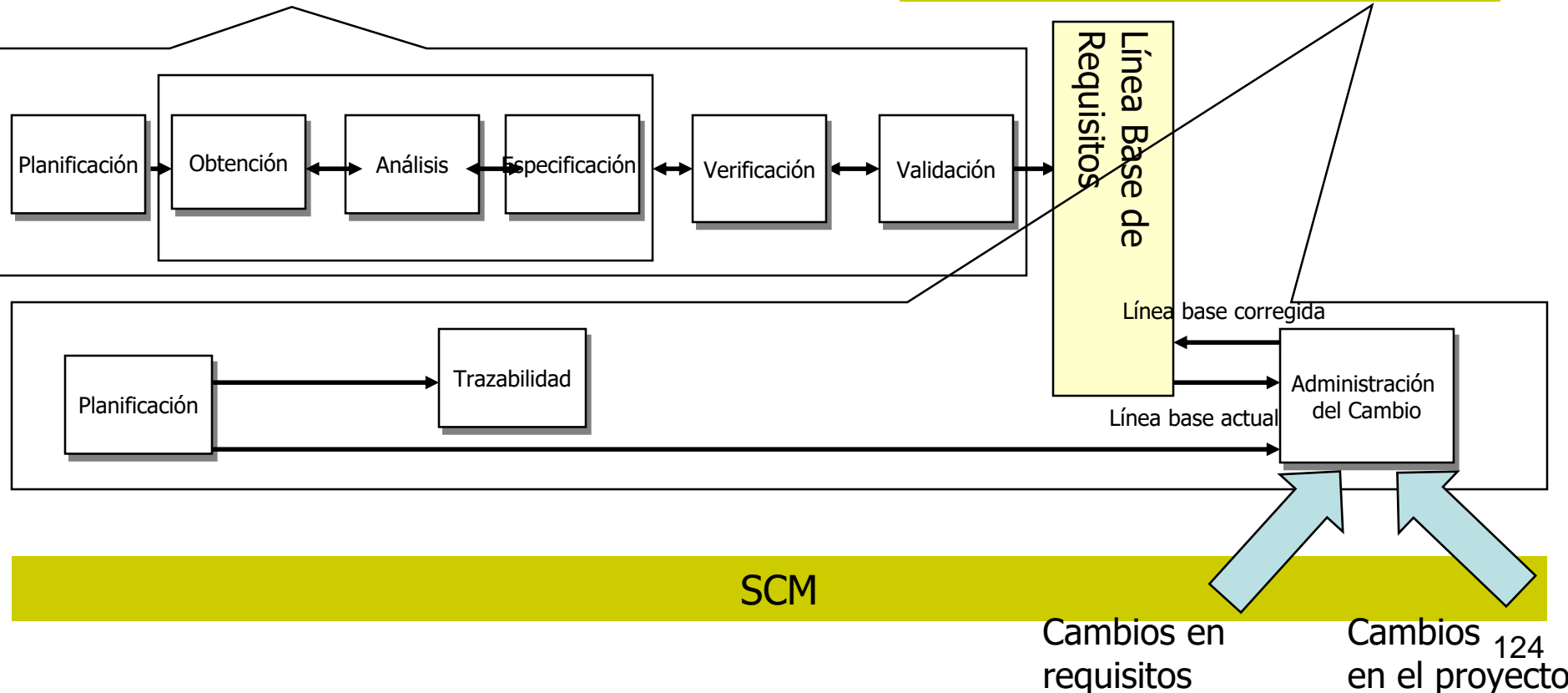
¿Cómo asegurar que la reunión es efectiva? Moderador, secretario y responsables por acciones

Ingeniería de Requisitos

Ingeniería de Requisitos

Proceso de los Requisitos

Administración de los Requisitos



Administración de los Requisitos

- Los requisitos cambian, debido a:
 - Muchos usuarios
 - Quienes pagan por el sistema y los usuarios no son las mismas personas
 - Cambios en el negocio
 - Cambios en la tecnología
- Proceso de comprender y controlar los cambios en los requisitos del sistema
- Se hace en paralelo con el Proceso de Requisitos.
- Tres etapas:
 - Planificación: Se realiza al comenzar el análisis de requisitos
 - Administración del cambio: Comienza una vez que se tiene una primera versión del documento de requisitos
 - Trazabilidad: Se mantiene a lo largo del proceso de requisitos

Planificación

- Muchas actividades son tomadas de las técnicas de SCM
Se debe decidir sobre:
 - **Identificación de Requisitos:** Cada requisito debe identificarse en forma única, para poder ser referenciado por otros.
 - Ejemplo:
 - <Tipo> <nro> donde Tipo: F – Funcional, D- Datos, etc.
 - Ejemplo: F12
 - **Proceso de Administración del Cambio:** Actividades que evalúan el impacto y costo del cambio
 - **Políticas de Trazabilidad:** Definen qué relaciones entre reqs. y el diseño se deben registrar y cómo se van a mantener.
 - **Herramientas CASE:** De soporte para:
 - Almacenar los requisitos
 - Administrar el cambio
 - Administrar de la trazabilidad

Trazabilidad

- Información de rastreo que se debe mantener
 - **La fuente**: Quién propuso el requerimiento y porqué
 - **Requisitos dependientes**: Vincula los requisitos dependientes entre sí, se usa para el análisis del cambio
 - Trazabilidad **entre artefactos** distintos, qué versión se corresponde con cuál. Pe.:
 - Rastreo reqs - CU
 - **Rastreo al diseño**: Vincula el req con los módulos de diseño que lo implementan
- Uso de matrices de trazabilidad

Administración del Cambio

- El cambio va a ocurrir.
- Objetivos del control de cambios:
 - Manejar el cambio y asegurar que el proyecto incorpora los cambios correctos por las razones correctas.
 - Anticipar y acomodar los cambios para producir la mínima interrupción y costo.
- Si los reqs cambian mucho dp de LB →
 - relevamiento incompleto/inefectivo
 - o acuerdo prematuro

Administración del Cambio

- Cuando se propone un cambio, debe evaluarse el impacto.
- Etapas:
 1. Especificación del cambio
 2. Evaluar impacto - Análisis del cambio y costo:
 - Se usa la información del rastreo
 - Se calcula el costo en términos de modificaciones a:
 - Docs de requisitos
 - Diseño e implementación
 3. Discutir costo con cliente.
 4. Implementar el cambio: se modifican los artefactos necesario
- Siguiendo estos pasos se logra
 - Todos los cambios se tratan en forma consistente
 - Los cambios a los docs de requisitos se hacen en forma controlada

Procedimiento de control de cambios

- Establecer procedimiento de control de cambios:
 - quién - Comité de Control de Cambios (CCC)
 - documentar:
 - integración del CCC
 - alcance de autoridad
 - procedimientos operativos (pe. evaluar impacto)
 - proceso de toma de decisiones

Gestión de la Configuración de los Requisitos

- Tiene que haber un responsable
- Control de versiones: Definir:
 - Ítems de configuración
 - Procedimientos
- Línea Base. Definición:
 - Conjunto de especificaciones y /o productos que han sido revisados formalmente y acordados, que sirven de base para desarrollo futuro, y que solo pueden ser cambiados a través de procedimientos formales de control de cambios.

Línea Base de Requisitos

- LB de reqs, arranca cuando se decide que son suficientemente buenos como para arrancar diseño y construcción.
- Sobre LB planifico cronograma y costo.
- Asociada a la liberación de un producto. Debo poder recomponer la liberación.
- Definir:
 - qué artefactos van en Línea Base
 - cuándo entran

Medir y Evaluar Requisitos

- Medir características de los requisitos para obtener detalles
 - Proceso de los Requisitos
 - Calidad de los Requisitos
- Las mediciones van a estar relacionadas con:
 - Producto (de los requisitos)
 - tamaño, calidad, atributos técnicos,
 - Proceso
 - actividades,...
 - Recursos
 - personas, equipos, tiempo, dinero,...

Medir y Evaluar Requisitos

- Medir
 - # Requisitos
 - Entrada para estimación del producto
 - # Cambios introducidos
 - Requisitos Agregados, Modificados, Desechados en el tiempo
 - Estabilidad
 - # Requisitos por tipo de requisitos
 - Permite luego ver en qué parte se encuentra el cambio
 - # Requisitos validados
- Tamaño del producto y del proyecto (ej.:PF, LoC)
 - planificar

Muchas gracias