Software: instrucciones, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación. Elemento lógico, se desarrolla y no se desgasta. El producto de software tiene problemas con los cambios.   
Hay softwares de tipo genéricos que son los sistemas aislados producidos por organizaciones desarrolladoras y que se venden en un mercado abierto (cada vez hay más software genérico que se adapta al cliente a posteriori), tambien existen los personalizados que son aquellos requeridos por un cliente particular.  
El software libre es (Richard Stallman):

1. Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito para siempre.
2. Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al codigo fuente.
3. Libertad de distribución, de modo que nos permita colaborar con vecinos y amigos.
4. Libertad para mejorar el programa y publicar las mejoras. Tambien exige el codigo fuente.

Clasificación de software:

* De sistemas.
* De aplicación.
* Científico y de ingeniería.
* Integrado.
* De linea de productos.
* Aplicaciones web/móviles.
* De IA.

A medida que se implementan nuevas tecnologías, parte del trabajo es integrar los sistemas tradicionales con los nuevos para asegurar un contexto útil.  
Debemos estar conscientes de que al integrar tecnologías se ven afectados todos los tipos de usuarios y sistemas.  
Ingeniería de software: Disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema incluyendo la evolución de este, luego que se comienza a ejecutar. Es el uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento del software.

Participantes en el desarrollo de software

* Gerentes ejecutivos.
* Gerente del proyecto.
* Profesionales especializados.
* Clientes.
* Usuarios finales.

Historia

Años 60-80: Programación modular, con ideas de acoplamiento y cohesión. Surge una enfoque mas formal, la programación estructurada.  
Surge el desarrollo en cascada y hay ideas de prototipacion.

Años 80-90: Surgen los primeros problemas de calidad con el surgimiento de la globalización y el sistemas distribuido.  
Con las ideas de POO, surgen nuevos lenguajes y metodología estructurada pasa a competir con el desarrollo orientado a objetos. Surge la ingeniería basada en componentes.

Años 90: Surge internet y los cambios.  
La integración continua con desarrollo incremental e iterativo se convierte en norma.  
Los patrones influyen en la generación del desarrollo de software.  
Surgen las bases de codigo abierto.  
Los dispositivos móviles aparecen en escena y el mundo cambia nuevamente.  
Metodologías agiles.

Actualidad: Big data e IA.  
Las ideas de IA estan hace años, lo que cambia es la disponibilidad de datos, lo que hace viable los enfoques estadísticos y las redes neuronales.

Ingeniero de software:

El ingeniero debe dominar los aspectos técnicos, aprender habilidades requeridas para entender el problema, diseñar solución y desarrollarla, etc.  
La IS se desarrolla en un marco económico, social y legal.

Al iniciar un proyecto. La primera actividad es saber lo que quiere el usuario, como lo quiere, cuando y porque.  
Al hablar de necesidades, en términos técnicos, estamos hablando de requerimientos.

Fuentes de requerimientos:

* Documentación.
* Stakeholders.
* Especificación de sistemas similares.

Stakeholder: son aquellos grupos o personas que se verán afectados por el sistema, directa o indirectamente.

Entre los Stakeholders se encuentran:

* Usuarios finales.
* Ingenieros.
* Gerentes.
* Expertos del dominio.
* Diferentes visiones.

Existen tres tipos genéricos de puntos de vista:

* Interactuadores: representan a las personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema de algún modo.
* Indirecto: representan a los Stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo.
* Dominio: representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

Elicitación de requerimientos

Un requerimiento (o requisito) es una característica del sistema o una descripción de algo que el sistema es capaz hacer con el objeto de satisfacer el propósito del sistema.  
Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo.  
Condición o capacidad que debe satisfacer o poseer un sistema o una componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.  
Representación documentada de una condición o capacidad.  
El proceso de adquirir (eliciting) (sonsacar) todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema.

Objetivos:

* Conocer el dominio del problema para poder comunicarse con clientes y usuarios y entender sus necesidades.
* Conocer el sistema actual (manual o informatizado).
* Identificar las necesidades, tanto explicitas como implícitas, de clientes y usuarios y sus expectativas sobre el sistema a desarrollar.

Requerimiento = requisito.

Técnicas de elicitación

Recopilación de información

**Métodos discretos**

1. Muestreo de la documentación, los formularios y los datos existentes.
2. Investigación y visitas al lugar.
3. Observación del ambiente de trabajo.

**Métodos interactivos**

1. Cuestionarios.
2. Entrevistas.
3. Planeación conjunta de requerimientos (JRP o JAD).
4. Lluvia de ideas.

Métodos interactivos

La base es hablar con las personas en la organización y escuchar para comprender.

*Cuestionarios*

Documento que permite al analista recabar información y opiniones de los encuestados.

Sirve para:

* Recolectar hechos de un gran número de personas.
* Detectar un sentimiento generalizado.
* Detectar problemas entre usuarios.
* Cuantificar respuestas.

Ventajas:

* Respuesta rápida.
* Económicos.
* Anónimos.
* Estructurados de facil análisis.

Desventajas:

* Numero bajo de respuestas.
* No responde a todas las preguntas.
* Preguntas rígidas.
* No se puede realizar un análisis corporal.
* No se pueden aclarar las respuestas incompletas.
* Difíciles de preparar.

Tipos de preguntas

Abiertas: son las que dejan abiertas todas las posibles opciones de respuestas.

Cerradas: delimitan o cierran las opciones de respuestas disponibles.

Cuando usar cuestionarios

* Las personas estan distribuidas geográficamente.
* Muchas personas involucradas.
* Queremos obtener información general.
* Queremos identificar problemas generales.

Entrevista

Técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara.  
Es una conversación con un propósito especificado, que se basa en un formato de preguntas y respuestas en general.  
Conocer opiniones y sentimientos del entrevistado.

Ventajas

* El entrevistado se siente incluido en el proyecto.
* Es posible obtener una retroalimentación del encuestado.
* Es posible adaptar las preguntas de acuerdo al entrevistado.
* Información no verbal observando las acciones y expresiones del entrevistado.

Desventajas

* Costosas.
* Tiempo y recursos humanos.
* Las entrevistas dependen en gran parte de las habilidades del entrevistador.
* No aplicable a distancia.

Tipos de entrevistas

Estructuradas (cerradas)

* El encuestador tiene un conjunto especifico de preguntas para hacerle al entrevistado.
* Se dirige al usuario sobre un requerimiento puntual.
* No permite adquirir un amplio conocimiento del dominio.

No estructuradas (abiertas)

* El encuestador lleva un tema en general.
* Sin preparación de preguntas específicas.
* Iniciar con preguntas que no dependen del contexto, para conocer el problema, la gente involucrada, etc.

Tipos de preguntas

*Abiertas*: permite al encuestado responder de cualquier manera.  
Ventajas

* Revelan nueva linea de preguntas.
* Hacen más interesante la entrevista.
* Permiten espontaneidad.

Desventajas

* Pueden dar detalles irrelevantes.
* Se puede perder el control de la entrevista.
* Parece que el entrevistador no tiene los objetivos claros.

*Cerradas:* las respuestas son directas, cortas o de selección especifica.

Ventajas

* Ahorran tiempo.
* Se mantiene facil el control de la entrevista.
* Se consiguen datos relevantes.

Desventajas

* Pueden aburrir al encuestado.
* No se obtienen detalles.

*Sondeo:* permite obtener más detalle sobre un tema puntual.

Entrevistas, la organización de una entrevista se puede dar en forma piramidal (inductivo) que es de preguntas cerradas a abiertas, en forma de embudo (deductivo) que es de preguntas abiertas a cerradas o diamante que es una combinación de ambas que comienza en pirámide y termina en embudo.

Preparación previa (Kendall).

Leer los antecedentes: poner atención en el lenguaje. Buscar un vocabulario en común. Imprescindible para poder entender al entrevistado.

Establecer los objetivos de la entrevista: usando los antecedentes. Los directivos suelen proporcionar una visión general, mientras que los futuros usuarios una más detallada.

Seleccionar los entrevistados: se debe minimizar el número de entrevistas. Los entrevistados deben conocer con antelación el objetivo de la entrevista y las preguntas que se le van a hacer.

Planificación de la entrevista y preparación del entrevistado: establecer fecha, hora, lugar y duración de cada entrevista de acuerdo con el entrevistado. Selección de tipo de preguntas a usar y su estructura.

Como conducir la entrevista

* Selección del entrevistado.
  + Según el requerimiento a analizar.
  + Conocer sus fortalezas, prejuicios y motivaciones.
    - Armar la entrevista en base a las características de la persona.
  + Hacer una cita.
  + Establecer una duración.
  + Informar al entrevistado el tema a tratar antes de la reunión.
  + Definir un guion de entrevista.
  + Se deben evitar preguntas sesgadas, con intención o críticas.
  + Usar lenguaje claro y conciso.
  + No incluir opinión como parte de la pregunta.
  + Evitar realizar preguntas largas y complejas.

*Planeación conjunta de requerimientos (JRP)*

Proceso mediante el cual se conducen reuniones de grupo altamente estructurados con el propósito de analizar problemas y definir requerimientos.

* Requiere de extenso entrenamiento.
* Reduce el tiempo de exploración de requisitos.
* Amplia participación de los integrantes.
* Se trabaja sobre lo que se va generando.
* Alguna bibliografía la menciona como JAD (Join Application Design).

Ventajas

* Ahorra tiempo.
* Usuarios involucrados.
* Desarrollos creativos.

Desventajas

* Es dificil de organizar los horarios de los involucrados.
* Es complejo encontrar un grupo de participantes integrados y organizados.

Lluvia de ideas

* Técnica para generar ideas al alentar a los participantes para que ofrezcan tantas ideas como sea poco en un corto tiempo sin ningún análisis hasta que se hayan agotado las ideas.
* Se promueve el desarrollo de ideas creativas para obtener soluciones.
* Se realizan reuniones del equipo involucrado en la resolución del problema, conducidas por el director.

Los principios en que se basa esta técnica son:

* Cuantas más ideas se sugieren, mejores resultados se conseguirán.
* La producción de ideas en grupos puede ser más efectiva que la individual.
* Las ideas de una persona hacer que aparezcan otras por “contagio”.
* A veces las mejores ideas aparecen tarde.
* Es mejor elegir sobre una variedad de soluciones.

Incluye una serie de fases de aplicación:

* Descubrir hechos, producir ideas, descubrir soluciones.
* Clave para resolver la falta de consenso entre usuarios.
* Es útil combinarlo con la toma de decisiones.
* Ayuda a entender el dominio del problema.
* Encara la dificultad del usuario para transmitir.
* Ayuda a entender: al usuario y al analista.

**Métodos discretos**

Los métodos discretos son menos perturbadores que otras formas de averiguar los requerimientos.  
Se consideran insuficientes para recopilar información cuando se utilizan por si solos, por lo que deben utilizarse junto con uno o varios de los métodos.  
Utilizar diferentes métodos para acercarse a la organización es una práctica inteligente mediante la cual podrá formarse un panorama más completo de los requerimientos.

* Muestreo de la documentación, los formularios y los datos existentes.
* Investigación y visitas al sitio.
* Observación del ambiente de trabajo.

**Muestro de la documentación, los formularios y los datos existentes**

* Recolección de hechos a partir de la documentación existente.
  + ¿Qué tipo de documentos pueden enseñar algo acerca del sistema?
    - Organigrama.
    - Memos, notas internas, minutas, registros contables.
    - Solicitudes de proyectos de sistemas de información anteriores.
  + Permiten conocer el historial qe origina el proyecto.
* Recolección de hechos a partir de la documentación existente.
  + Documentos que describen la funcionalidad del negocio que está siendo analizada.
    - Declaración de la misión y plan estratégico de la organización.
    - Objetivos formales del departamento en cuestión.
    - Políticas, restricciones y procedimientos operativos.
    - Bases de datos.
    - Sistemas en funcionamiento.
  + Documentación de sistemas anteriores.
    - Diagramas.
    - Diccionarios o repositorios de proyecto.
    - Documentos de diseño.
    - Manuales de operación y/o entrenamiento.

**Investigación y visitas al sitio**

* Investigar el dominio.
* Patrones de soluciones (mismo problema en otra organización).
* Revistas especializadas.
* Buscar problemas similares en internet.
* Consultar otras organizaciones.

**Observación del ambiente de trabajo**

* El analista se convierte en observador de las personas y actividades con el objeto de aprender acerca del sistema.
* Lineamientos de la observación:
  + Determinar quién y cuándo será observado.
  + Obtener el permiso de la persona y explicar el porqué será observado.
  + Mantener el perfil bajo.
  + Tomar nota de lo observado.
  + Revisar las notas con la persona apropiada.
  + No interrumpir a la persona en su trabajo.
* Ventajas.
  + Datos confiables.
  + El analista puede ver exactamente lo que se hace (tareas difíciles de explicar con palabras).
  + Análisis de disposiciones físicas, transito, iluminación y ruido.
  + Económica en comparación de otras técnicas.
* Desventajas.
  + La gente se siente incómoda siendo observada.
  + Algunas actividades del sistema pueden ser realizadas en horarios incomodos.
  + Las tareas estan sujetas a interrupciones.
  + Tener en cuenta que la persona observada puede estar realizando las tareas de la forma “correcta” y no como lo hace habitualmente.

**Especificación de requerimientos**

Objetivos

* Permitir que los desarrolladores expliquen cómo han entendido lo que el cliente pretende del sistema.
* Indicar a los diseñadores que funcionalidad y características va a tener el sistema resultante.
* Indicar al equipo de pruebas que demostraciones llevar a cabo para convencer al cliente de que el sistema que se le entrega es lo que habia pedido.

Propiedad de los requerimientos

* Necesario: su omisión provoca una deficiencia.
* Conciso: facil de leer y entender.
* Completo: no necesita ampliarse.
* Consistente: no contradictorio con otro.
* No ambiguo: tiene una sola implementación.
* Verificable: puede testearse a través de inspecciones, pruebas, etc.
* Documento de definición de requerimientos: listado completo de todas las cosas que el cliente espera que haga el sistema propuesto.
* Documento de especificación de requerimientos: definición en términos técnicos.
* Documento de especificación de requerimientos de software: el objetivo es brindar una colección de buenas prácticas para escribir especificaciones de requerimiento de software (SRS) y, además, se describen los contenidos y las cualidades de una buena especificación de requerimientos.

Aspectos básicos de una especificación de requerimientos.

* Funcionalidad.
  + ¿Qué debe hacer el software?
* Interfaces externas.
  + ¿Cómo interactuara el software con el medio externo (gente, hardware, otro software)?
* Rendimiento.
  + Velocidad, disponibilidad, tiempo de respuesta, etc.
* Atributos.
  + Portabilidad, seguridad, mantenibilidad, eficiencia.
* Restricciones de diseño.
  + Estándares requeridos, lenguaje, límite de recursos, etc.

**Técnicas de especificación de requerimientos**

* Estáticas.
  + Se describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con otros. No describe como las relaciones cambian con el tiempo.
  + Cuando el tiempo no es un factor mayor en la operación del sistema, es una descripción útil y adecuada.

Ejemplos: referencia indirecta, relaciones de recurrencia, definición axiomática, expresiones regulares, abstracciones de datos, entre otras.

* Dinámicas.
  + Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo.
  + Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar de estado.

Ejemplo: tablas de decisión, diagramas de transición de estados, tablas de transición de estados, diagramas de persianas, diagramas de transición extendidos, redes de Petri, entre otras.

**Historia de usuario**

* Una historia de usuario es una descripción corta y simple de un requerimiento de un sistema, que se describe en lenguaje común del usuario y desde su perspectiva.
* Son utilizadas en las metodologías agiles (ejemplo: XP, SCRUM) para la especificación de requerimientos.
  + Acompañadas de las discusiones con los usuarios y las pruebas de validación.
* **Debe ser limitada**, esta debería poder escribirse sobra una nota adhesiva pequeña.
* Son **una forma rápida de administrar los requisitos** de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos.
* Permitir **responder rápidamente a los requerimientos cambiantes**.
* Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de **discutirlas con los clientes**.
* Generalmente se espera que la **estimación de tiempo** de cada historia de usuario se situe entre unas **10 horas y un par de semanas**.
* Estimaciones **mayores a dos semanas** son indicativo de que la historia es muy compleja y debe ser **dividida en varias historias**.

Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas.  
**¿Quién se beneficia?  
¿Qué se quiere?  
¿Cuál es el beneficio?**

Esquema:  
Como **rol** quiero **algo** para **beneficio**.

Como **cliente** quiero **suscribirme por medio del sitio web** para **obtener un nuevo plan de TV por cable**.

**Independientes unas de otras:** de ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.  
**Negociables:** la historia en sí misma no es lo suficientemente explicita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y este debe dejarse explicito bajo la forma de pruebas de validación.  
**Valoradas por los clientes o usuarios:** los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.  
**Estimables:** Un resultado de la discusión de una historia de usuario de la estimación del tiempo que tomara completarla. Esto permite estimar el tiempo total del proyecto.  
**Pequeñas:** las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.  
**Verificables:** las historias de usuarios cubren requerimientos funciones, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

*Criterios de aceptación*

* Un criterio de aceptación es el criterio por el cual se define si una historia de usuario fue desarrollada según la expectativa del Product Manager/Owner y se si puede dar por hecha.
* Deben ser definidos durante la etapa inicial antes de la codificación, acompañan a la historia de usuario, porque complementan la historia de usuario y ayudan al equipo de desarrollo a entender mejor como se espera que el producto se comporte.
* Los criterios de aceptación son utilizados para expresar el resultado de las conversaciones del cliente con el desarrollador. El cliente debería ser quien las escriba más que los desarrolladores.
* Representan el inicio de la definición del cómo. No estan diseñados para ser tan detallados como una especificación de diseño tradicional.
* Si una historia de usuario tiene más de 4 criterios de aceptación, debe evaluarse subdividir la historia.
* Puede añadirse un numero de escenario para identificar al criterio, asociado a la historia de usuario en cuestión.

**Plantilla** *Plantilla*

ID: identificador univoco de la historia expresado como texto generalmente de la forma <verbo><sustantivo>.

Título: Descripción de la historia de a forma: **Como** <rol> **quiero** <algo> **para** <beneficio>.

Reglas de negocio: Conjunto de reglas, normas, políticas, etc. que condicionan el modo de operación.

*Criterios de aceptación*

Escenario 1: título del criterio.

Dado <un contexto actual>,  
Cuando <ocurre un evento>,  
Entonces <garantizo uno o más resultados>.

Escenario 2: título del criterio.

Dado <un contexto actual>,  
Cuando <ocurre un evento>,  
Entonces <garantizo uno o más resultados>.

Escenario N: título del criterio.

Dado <un contexto actual>,  
Cuando <ocurre un evento>,  
Entonces <garantizo uno o más resultados>.

ID: Matricular persona  
Título: Como persona quiero matricularme al instituto para poder hacer los cursos.  
Reglas de negocio:

* Un DNI no puede estar registrado dos veces con diferentes matriculas.
* El pago debe realizarse con tarjeta de crédito.

Escenario 1: Matriculación exitosa.  
Dado que el DNI 222 no se encuentra matriculado y las condiciones son las adecuadas para el pago exitoso.  
Cuando la persona ingresa “Juan Pérez”, DNI 222, direccion 7 #123 y presiona “Matricularse” matricula a la persona y genera su número de matrícula.

Escenario 2: Matriculación fallida por matricula existente.

Dado que el DNI 222 se encuentra matriculada.  
Cuando la persona ingresa “Juan Pérez”, DNI 222 y direccion 7 #123 y presiona “Matricularse”.  
Entonces el sistema informa que la persona ya se encuentra matriculada.

Escenario 3: Matriculación fallida por error de pago.  
Dado que el DNI 123 no se encuentra matriculado y las condiciones no son las adecuadas para un pago exitoso.  
Cuando la persona ingresa “Pedro Gonzales”, DNI 123 y direccion 6 #234 y presiona “Matricularse”.  
Entonces el sistema redirige al usuario al pago de matrícula con tarjeta de crédito, espera respuesta e informa que el pago no ha sido correcto por lo que no se puedo matricular a la persona.

**Beneficios**

* Al ser muy corta, esta representa requisitos del modelo de negocio que pueden implementarse rápidamente.
* Necesitan poco mantenimiento.
* Mantienen una relacion cerca al cliente.
* Permite dividir los proyectos en pequeñas entregas.
* Permite estimar fácilmente el esfuerzo de desarrollo.
* Es ideal para proyectos con requisitos volátiles o no muy claros.

**Limitaciones**

* Sin criterios de aceptación pueden quedar abiertas a distintas interpretaciones haciendo difícil utilizarlas como base para un contrato.
* Se requiere un contacto permanente con el cliente durante el proyecto lo cual puede ser difícil o costoso.
* Podría resultar difícil escalar a proyectos grandes.
* Requiere desarrolladores muy competentes.

**Épica**

* Se denomina épica a un conjunto de HU que se agrupan por algún denominador común.
* La épica representa un objetivo alcanzable que nace de la necesidad del cliente.
* Es un objetivo al que nos aproximamos y que esperamos alcanzar algún día.
* La épica no es una funcionalidad.

Como VP de mercadeo y ventas quiero revisar el desempeño histórico de las ventas para identificar las regiones geográficas y productos de mejor desempeño.

Esta épica se puede subdividir en:

Como VP de mercadeo quiero seleccionar el periodo de tiempo en el cual realizare la revisión de ventas para analizar el desempeño.

Como VP de mercadeo quiero clasificar la información de ventas por región geográfica y productos para obtener la mejor zona de ventas.

* Las épicas suelen abarcar varios equipos de desarrollo.
* Recogen normalmente muchas historias de usuario.
* Los clientes determinan si eliminan o añaden historias dentro de cada épica.
* Una épica sirve para estructurar los temas e iniciativas (objetivos.
* Las épicas tambien sirven para dar flexibilidad y agilidad al proyecto.

**Requerimientos – Tipos**

*Requerimientos funcionales*

* Describen una interacción entre el sistema y su ambiente. Como debe comportarse el sistema ante determinado estimulo.
* Describen lo que el sistema debe hacer, o incluso como no debe comportarse.
* Describen con detalle la funcionalidad del mismo.
* Son independientes de la implementación de la solución.
* Se pueden expresar de distintas formas.

*Requerimientos no funcionales*

* Describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema.
* Requerimientos del producto.
  + Especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad)
* Requerimientos funcionales.
  + Se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares).
* Requerimientos externos.
  + Interoperabilidad, legales, privacidad, seguridad, éticos.

**Ingenieria de requerimientos**

Es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones.  
La ingeniería de requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizara el sistema.

* Tambien es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos.
* “Ingenieria de requerimientos” es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es tambien el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto.

Importancia

* Permite gestionar las necesidades del proyecto en forma estructurada.
* Mejora la capacidad de predecir cronogramas de proyectos.
* Disminuye los costos y retrasos del proyecto.
* Mejora la calidad del software.
* Mejora la comunicación entre equipos.
* Evita rechazos de usuarios finales.

**Estudio de viabilidad**

A partir de la descripción resumida del sistema se elabora un informe que recomienda la conveniencia o no de realizar el proceso de desarrollo.

Responde a las siguientes preguntas:

* ¿El sistema contribuye a los objetivos generales de la organización? (Si no contribuye, entonces no tiene un valor real en el negocio).
* ¿El sistema puede implementar con la tecnología actual?
* ¿El sistema puede implementar las restricciones de costo y tiempo?
* ¿El sistema puede integrarse a otros que existen en la organización?
* Una vez que se ha recopilado toda la información necesaria para contestar las preguntas anteriores se debería hablar con las fuentes de información para responder nuevas preguntas y luego se redacta el informe, donde debería hacerse una recomendación sobre si debe continuar o no el desarrollo.

**Validación de requerimientos**

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario.  
Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar el sistema.  
Es importante, porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde.

*Validación:* Al final del desarrollo evaluar el software para asegurar que el software cumple los requerimientos.

*Verificación:* El software cumple con los requerimientos correctamente.

*¿Es suficiente validar despues del desarrollo del software?*

* La evidencia estadística dice que NO.
* Cuanto más tarde se detecta, más cuesta corregir (Boehm).
* Bola de nieve de defectos.
* Validar en la fase de especificación de requerimientos puede ayudar a evitar costosas correcciones despues del desarrollo.

*¿Contra qué se verifican los requerimientos?*

* No existen los “requerimientos de los requerimientos”.
* No puede probarse formalmente que un Modelo de Requerimientos es correcto. Puede alcanzarse una convicción de que la solución especificada en el modelo de requerimientos es el correcto para el usuario.

*Comprenden*

* Verificaciones de la validez (para todos los usuarios).
* Verificaciones de consistencia (sin contradicciones).
* Verificaciones de completitud (todos los requerimientos).
* Verificaciones de realismo (se pueden implementar).
* Verificabilidad (se puede diseñar conjunto de pruebas).

*Técnicas de validación*

Pueden ser manuales o técnicas

* Revisiones de requerimientos (formales o informales)
  + Informales: los desarrolladores deben tratar los requerimientos con tantos Stakeholders como esa posible.
  + Formal: El equipo de desarrollo debe conducir al cliente, explicándole las implicaciones de cada requerimiento.
* Antes de una revisión formal, es conveniente realizar una revisión informal.
* Construcción de prototipos.
* Generación de casos de prueba.

**Casos de uso – CU**

Proceso de modelado de las “funcionalidades” del sistema en termino de los eventos que interactúan entre los usuarios del sistema.  
Tiene sus orígenes en el modelado orientado a objetos (Jacobson 1992) pero su eficiencia en modelado de requerimientos hizo que se independice de la técnica de diseño utilizada, siendo aplicable a cualquier metodología de desarrollo.  
El uso de CU facilita y alienta la participación de los usuarios.

* Herramienta para capturar requerimientos funcionales.
* Descompone el alcance del sistema en piezas más manejables.
* Medio de comunicación con los usuarios.
* Utiliza lenguaje común y fácil de entender por las partes.
* Permite estimar el alcance del proyecto y el esfuerzo a realizar.
* Define una línea base para la definición de los planes de prueba.
* Define una línea base para toda la documentación del sistema.
* Proporciona una herramienta para el seguimiento de los requisitos.

*Componentes*

Diagrama de casos de uso

* Ilustra las interacciones entre el sistema y los actores.

Escenarios (narración del CU)

* Descripción de la interacción entre el actor y el sistema para realizar la funcionalidad.

En el escenario se describen:

* La interacción del escenario.
* Eventos alternativos.

*Caso de uso (ovalo)*

Representa un objetivo (funcionalidad) individual del sistema y describe la secuencia de actividades y de interacciones para alcanzarlo.  
Para que el CU sea considerado un requerimiento debe estar acompañado de su respectivo escenario.

*Actores (personita)*

Un actor inicia una actividad (CU) en el sistema.  
Representa un papel desempeñado por un usuario que interactúa (rol).  
Puede ser una persona, sistema externo o dispositivo externo que dispare un evento (reloj).

*Diagrama*

Relaciones

* Asociados.

Relacion entre un actor y un CU en el que interactúan entre si.

* Extensiones (extends).

Un CU extiende la funcionalidad de otro CU.  
Un CU puede tener muchos CU extensiones.  
Los CU extensiones solo son iniciados por un CU.

* Uso o inclusión (uses).

Reduce la redundancia entre dos o más CU al combinar los pasos comunes de los CU.

* Herencia.

Relacion entre actores donde un actor hereda las funcionalidades de uno o varios actores.

Pasos

* Identificar los actores.
  + ¿Dónde buscar actores potenciales?
    - Documentación o manuales existentes.
    - Minutas de reunión.
    - Documentos de requerimiento.
  + Responder a:
    - ¿Quién o que proporciona las entradas al sistema?
    - ¿Quién o que recibe las salidas del sistema?
    - ¿Se requieren interfaces con otros sistemas?
    - ¿Quién mantendrá la información del sistema?
  + Deberán nombrarse con un sustantivo o frase sustantiva.
  + Identificar los CU para los requerimientos
  + Responder a:
    - ¿Cuáles son las principales tareas del actor?
    - ¿Qué información necesita el actor del sistema?
    - ¿Qué información proporciona el actor al sistema?
    - ¿Necesita el sistema información al actor de eventos o cambios ocurridos?
    - ¿Necesita el actor informar al sistema de evento o cambios ocurridos?
  + Construir el diagrama.
  + Realizar los escenarios.
  + Un CU representar una funcionalidad concreta.
  + La descripción de los pasos en los escenarios debe contener más de un paso, para representar la interacción entre los componentes.
  + El uso de condicionales en el curso normal, es limitado a la invocación de excepciones, ya que este flujo representa la ejecución del caso sin alteraciones.
  + Las pre-condiciones no deben representarse en los cursos alternativos, ya que al ser una pre-condición no va a ocurrir.
  + Los “uses” deben ser accedidos por lo menos desde dos CU.
* Identificar los CU para los requerimientos.
* Construir el diagrama.
* Realizar los escenarios.

Un sitio web brinda información acerca de los artículos periodísticos más destacados de la semana. La información puede ser accedida por usuarios registrados o anónimos. A los usuarios registrados se les permite leer y/o descargar los artículos. Si el artículo tiene categoría “exclusiva” la descarga del artículo tendrá un costo. El pago es mediante tarjeta de crédito.  
A los usuarios anónimos sólo se les permite leer los artículos.  
Un usuario anónimo puede registrarse y pasar a ser un usuario registrado, para lo cual debe completar los datos personales, ingresar el número de tarjeta de crédito a la que se cargará el monto mensual del abono.   
Los usuarios registrados pueden modificar sus datos personales.

Naranja actores y verde casos de uso.

* Identificar los usuarios:
  + Usuario anónimo.
  + Usuario registrado.
  + Servidor externo (banco).
* Identificar casos de uso:
  + Leer artículo.
  + Descargar artículo.
  + Registrar usuario.
  + Modificar datos personales.
  + Iniciar sesión.
  + Cerrar sesión.
  + Verificar tarjeta.

**Técnicas de especificación de requerimientos**

Estáticas:

* Se describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con otros. No describe como las relaciones cambian con el tiempo.
* Cuando el tiempo no es un factor mayor en la operación del sistema, es una descripción útil y adecuada.
* Ejemplos: Referencia indirecta, Relaciones de recurrencia, Definición axiomática, Expresiones regulares, Abstracciones de datos, entre otras.

Dinámicas:

* Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo.
* Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado.
* Ejemplos: Tablas de decisión, diagramas de transición de estados, Tablas de transición de estados, Diagramas de persianas, Diagramas de transición extendidos, Redes de Petri, entre otras.

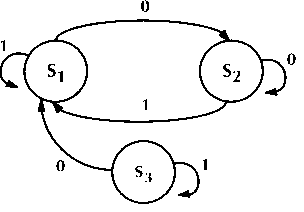
**Diagrama de transición de estado**

Máquinas de Estado Finito:

* Describe al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos).

f(Si, Cj) = Sk

* Al estar en el estado Si, la ocurrencia de la condición Cj hace que el sistema cambie al estado Sk.

f(S1, 0) = S2

f(S1, 1) = S1

f(S2, 0) = S2

f(S2, 1) = S1

f(S3, 0) = S1

f(S3, 1) = S3

*Máquinas de Estado Finito*

El DVD pasa por diferentes estados, dependiendo de lo que “escucha” del control remoto.



f(A, ) = B

f(B, ) = A

f(B, ) = C

f(C, ) = B

f(C, ) = A

A: esperando

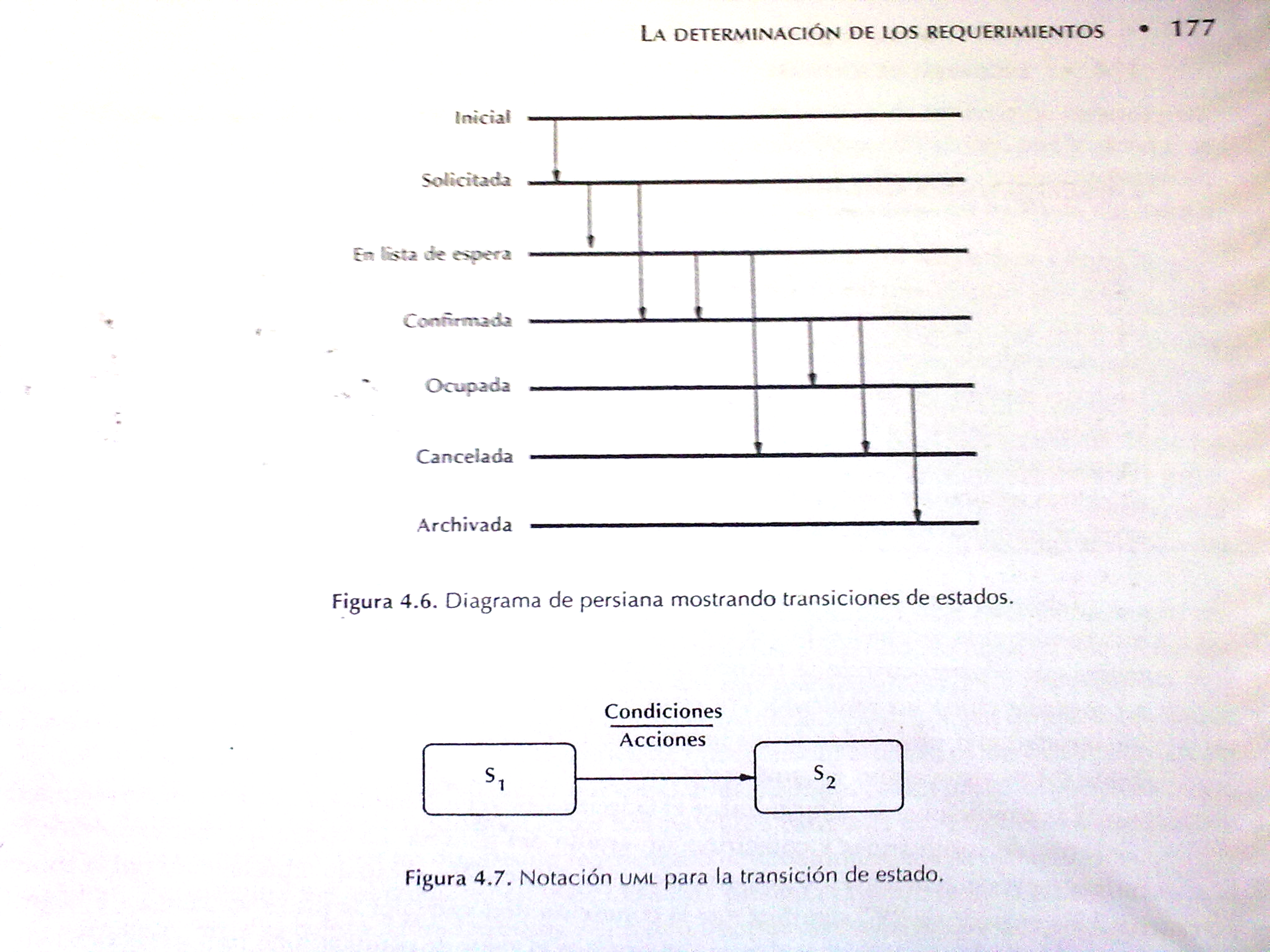
B: mostrando

C: pausa

Definición formal

* Formalmente, un autómata finito (AF) puede ser descrito como una 5-tupla (S,Σ,T,s,A) donde:
* Σ es un alfabeto.
* S un conjunto de estados.
* T es la función de transición.
* s es el estado inicial.
* A es un conjunto de estados de aceptación o finales.

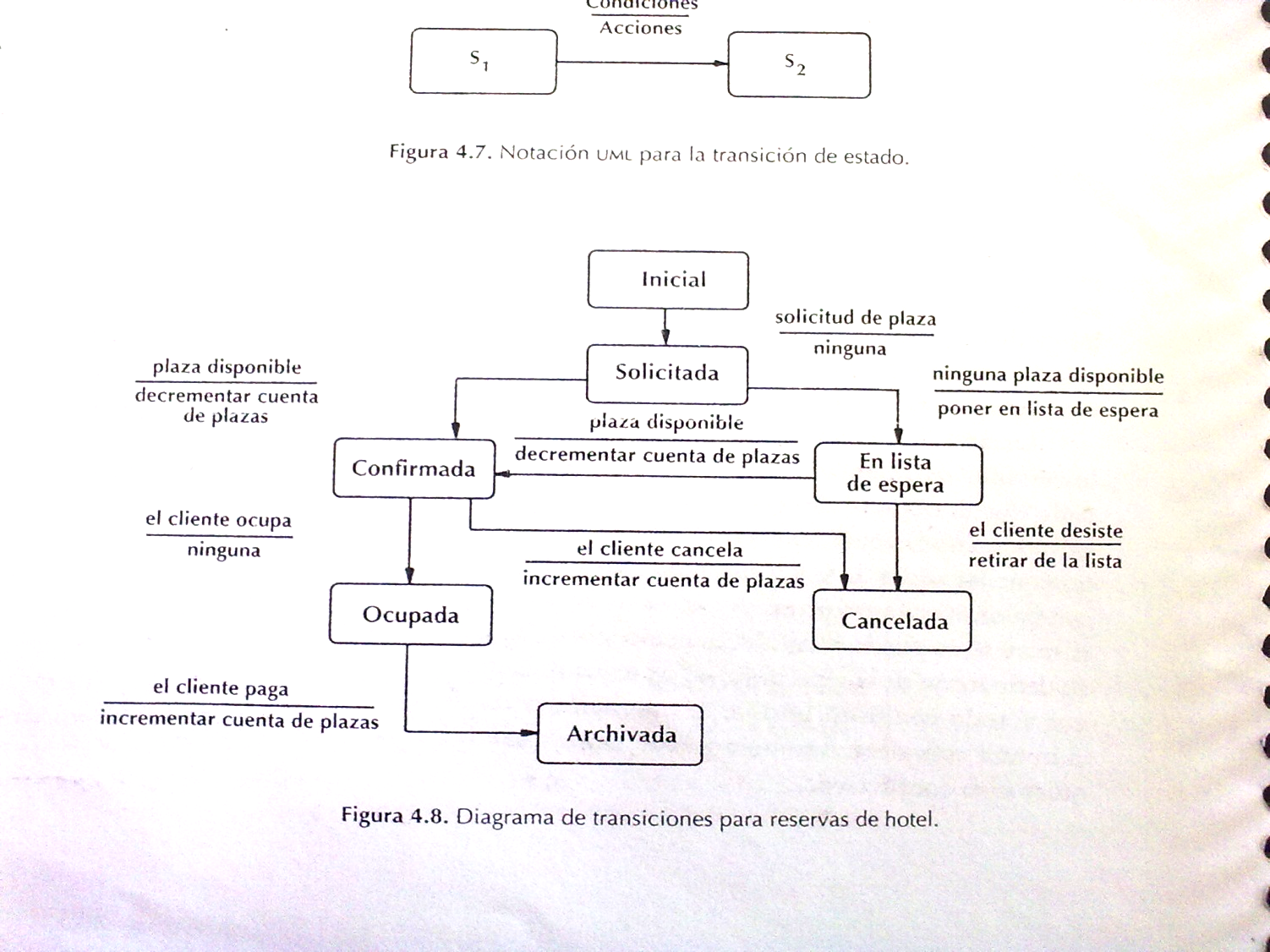
Representación en gráfico de persiana



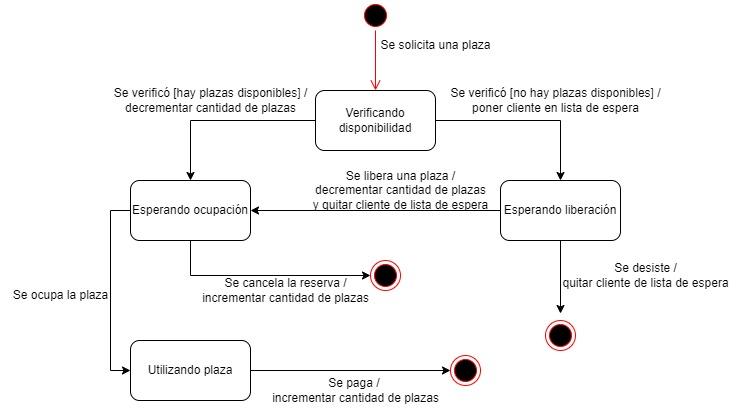
Transiciones

Estados

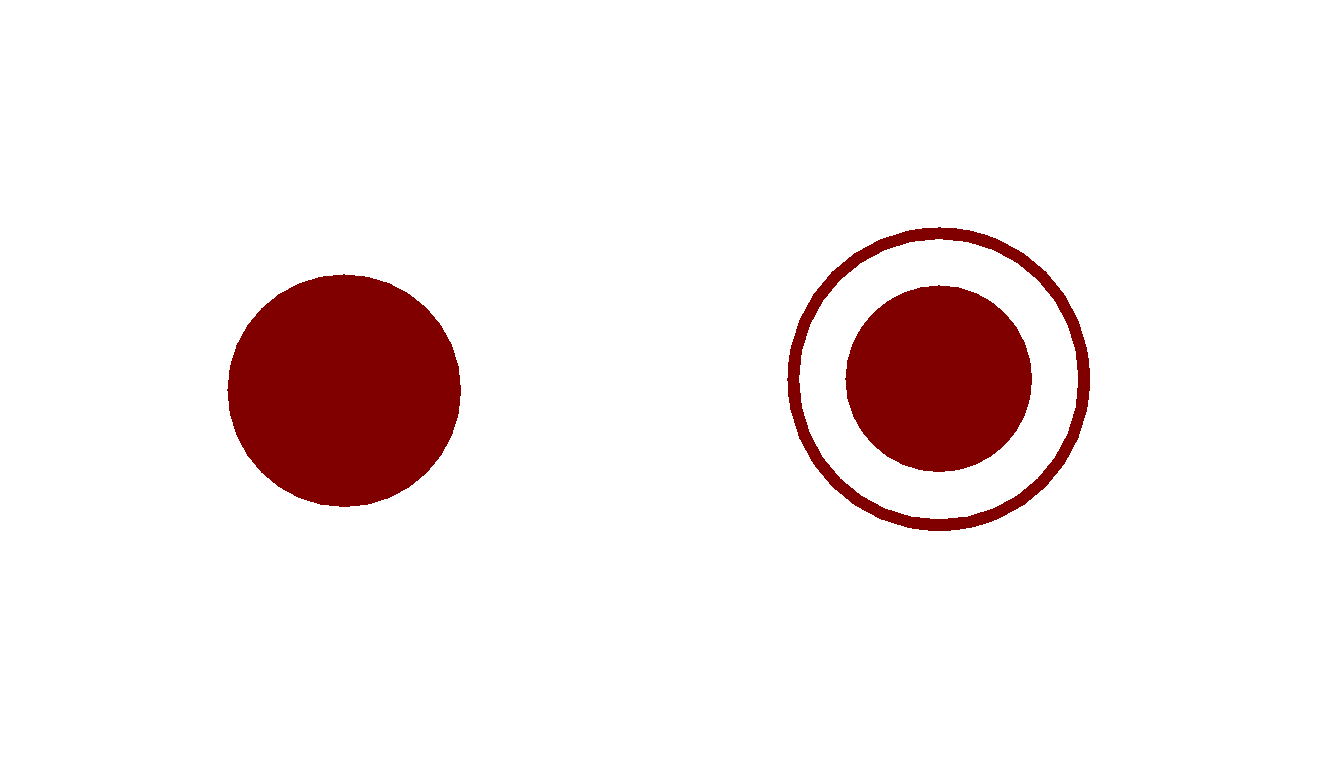
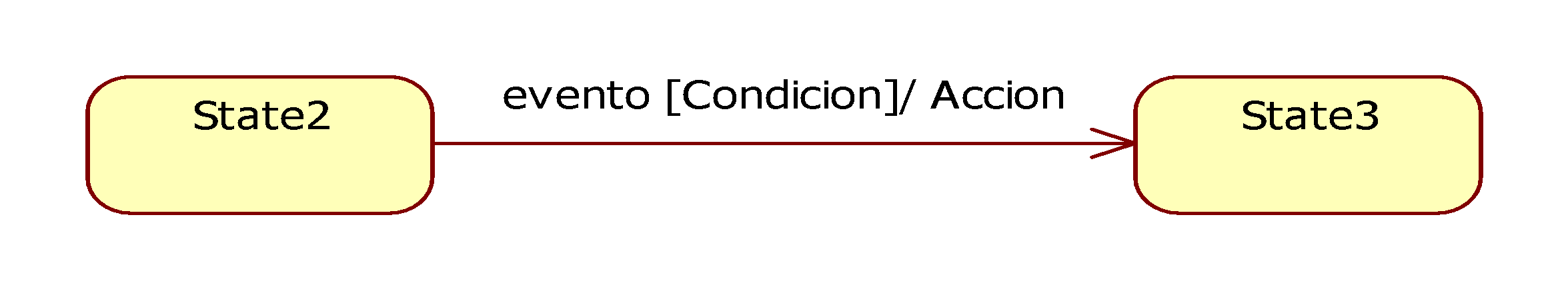
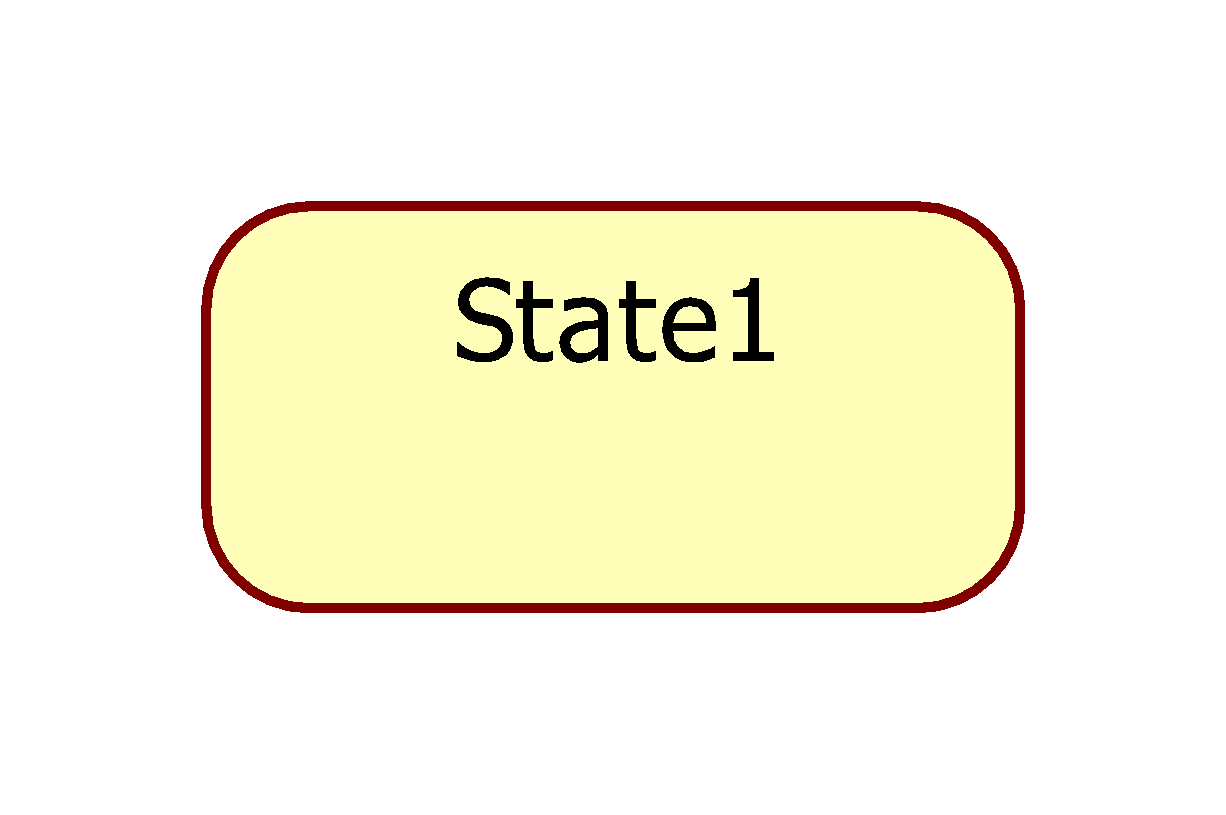
Representación en máquina de estado finito

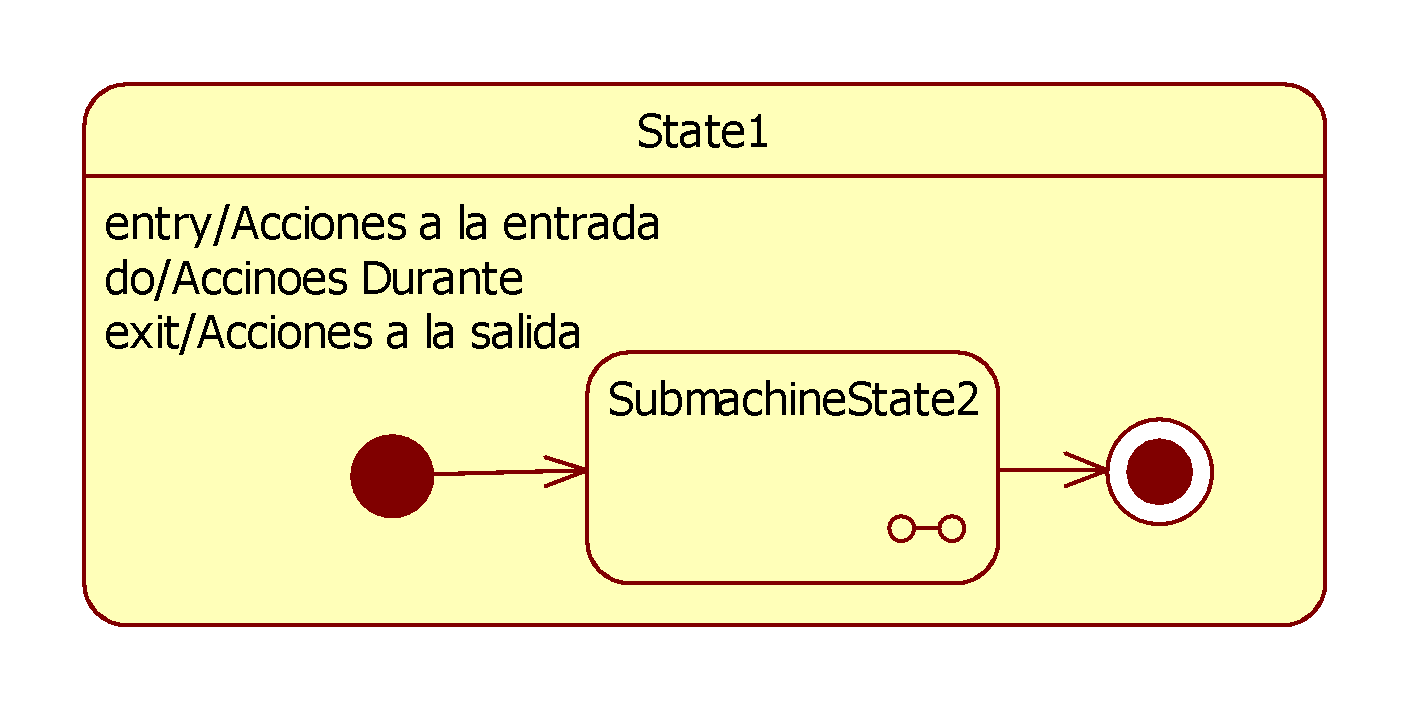
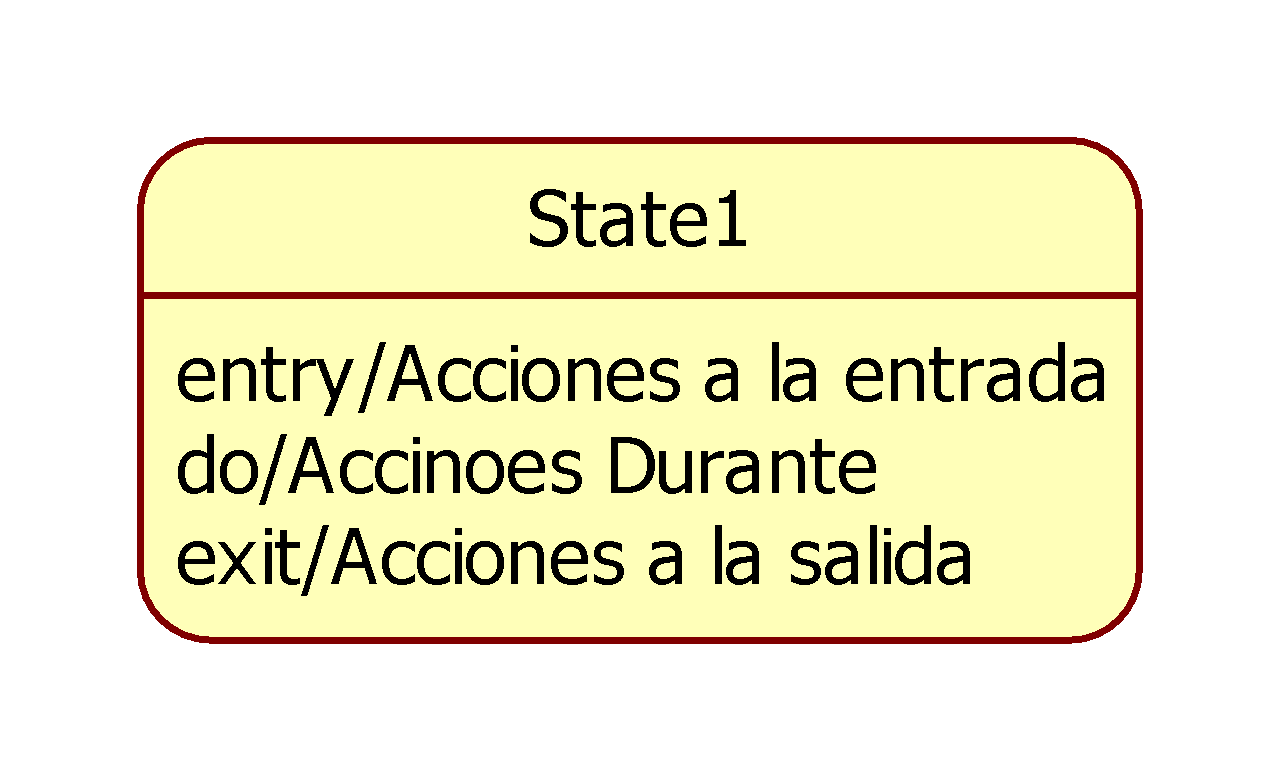


Representación en Diagrama de Transición de Estados



Notación UML Diagrama de Transición de Estado (DTE)





Estado básico.

Estado inicial/final.

Estado extendido.

Estado compuesto o sub-estado.

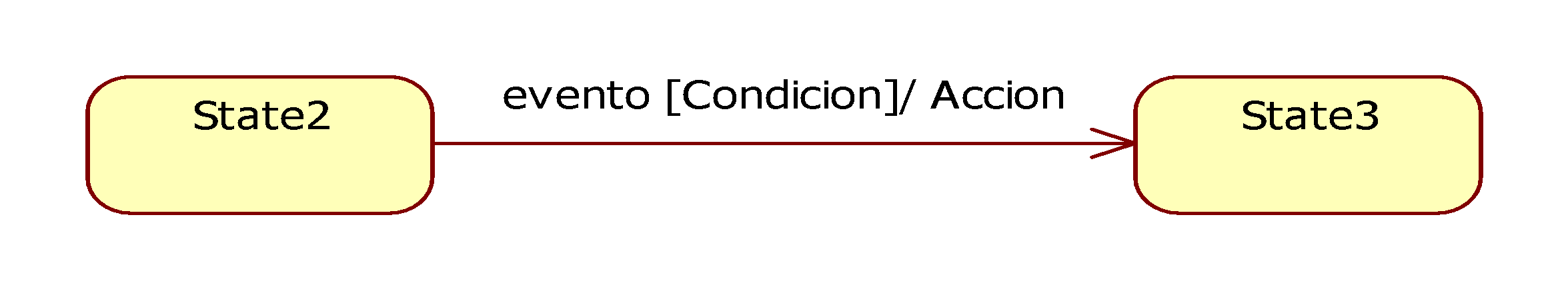
**Evento**

Es un **suceso significativo** que debe tenerse en cuenta, que influye en el comportamiento y evolución del sistema.

Tiene lugar en un punto del tiempo y carece de duración respecto a la granularidad temporal del sistema.

No tiene sentido preguntarse por lo que sucede mientras está teniendo lugar el evento.

Las transiciones se producen como consecuencia de eventos. Pueden o no tener un procesamiento asociado.



Evento obligatorio.

Condicional: opcional, puede haber transiciones sin condición.

Acción: opcional, puede haber transiciones sin acciones.

Construccion de un DTE:

1. Identificar los estados.
2. Si hay un estado complejo se puede explotar.
3. Identificar el estado inicial.
   1. Desde el estado inicial, se identifican los cambios de estado con flechas.
4. Se analizan las condiciones y las acciones para pasar de un estado a otro.
5. Se verifica la consistencia:
   1. Se han definido todos los estados.
   2. Se pueden alcanzar todos los estados.
   3. Se pueden salir de todos los estados.
   4. En cada estado, el sistema responde a todas las condiciones posibles (normales y anormales).

Se requiere modelar el comportamiento de un sistema de autenticación simple para una aplicación móvil. El sistema debe seguir las siguientes reglas:

Cuando se abre la aplicación por primera vez, el sistema muestra una "Pantalla de Inicio". Desde allí el usuario puede presionar el botón "Ingresar". Esto lo lleva al "Formulario de Ingreso".

Proceso de autenticación: en el "Formulario de Ingreso", si el usuario ingresa las credenciales correctas y presiona "Autenticar", el sistema activa la sesión del usuario. Si las credenciales son incorrectas, el sistema muestra un mensaje de error y permanece en el "Formulario de Ingreso".

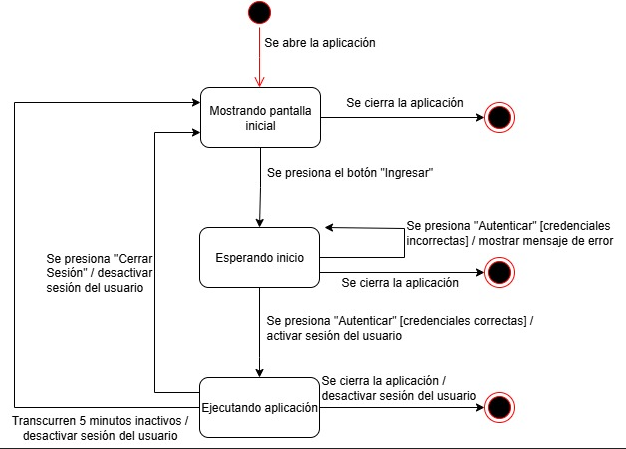
Una vez iniciada la sesión, el usuario puede presionar el botón "Cerrar Sesión", lo que lo devuelve a la "Pantalla de Inicio".

Inactividad: después de 5 minutos de inactividad, la sesión del usuario expira automáticamente, y el sistema regresa a la "Pantalla de Inicio".

Si la aplicación se cierra, el sistema finaliza, independientemente del estado en el que se encuentre

Funciones

* Inicialmente (al abrir la app) visualiza la pantalla inicial.
* Visualizar formulario de ingreso.
* Ingresar datos de autenticación (correctos o incorrectos).
* Presionar botón cerrar sesión.
* Dejar la aplicación inactiva X tiempo.
* Finaliza si la aplicación se cierra.



Ventajas del DTE:

* **Mejora la comprensión**: ofrece un mapa visual que hace fácil entender cómo un sistema reacciona a diferentes eventos.
* **Previene errores:** ayuda a encontrar fallos de lógica y caminos que no deberían existir, todo esto en la etapa de diseño, lo que ahorra tiempo y esfuerzo en la codificación.
* **Valida el diseño:** confirma que todos los requisitos del sistema han sido considerados y que el comportamiento es coherente.
* **Facilita las pruebas**: proporciona un plan para crear pruebas, asegurando que se validan todas las interacciones posibles del sistema.

Inconvenientes del DTE:

* **Complejidad:** pueden volverse muy complejos y difíciles de manejar en sistemas con una gran cantidad de estados y transiciones.
* **Enfoque en un solo objeto:** un DTE típicamente modela el comportamiento de un único objeto o componente. Para entender el sistema completo, se necesita una colección de DTEs.
* **No modela la concurrencia:** los DTEs básicos no son ideales para modelar sistemas concurrentes, donde varios estados pueden cambiar al mismo tiempo.

¿Cuándo usar DTE?

* **Sistemas de tiempo real**: cuando el software está impulsado por eventos externos, como un sistema de control para una máquina o un dispositivo.
* **Modelado del comportamiento de la interfaz de usuario:** para aplicaciones con interfaces de usuario complejas, un DTE puede mostrar cómo la pantalla o un componente de la interfaz cambia en respuesta a las acciones del usuario.
* **Sistemas con un número finito de estados**: cuando el componente de software puede estar en un número limitado de estados bien definidos (por ejemplo, una válvula que puede estar "abierta" o "cerrada").
* **Diseño de algoritmos de control:** en el diseño de software para sistemas de control, donde el estado actual del sistema determina su próximo comportamiento.

**Redes de Petri**

Utilizadas para especificar sistemas de tiempo real en los que son necesarios representar aspectos de concurrencia.

Los sistemas concurrentes se diseñan para permitir la ejecución simultánea de componentes de programación, llamadas tareas o procesos, en varios procesadores o intercalados en un solo procesador.

Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas (pueden operar a distintas velocidades, deben prevenir la modificación de datos compartidos o condiciones de bloqueo).  
Pueden realizarse varias tareas en paralelo, pero son ejecutados en un orden impredecible.

Éstas NO son secuenciales.

Eventos o acciones: Los eventos se representan como transiciones (T).

Estados o condiciones: Los estados se representan como lugares o sitios (P).

Caso más simple:

f(EstadoA, Evento) -> EstadoS

Se requieren varios eventos para pasar de un estado a otro. Los eventos NO ocurren en un orden determinado.

f(EstadoA, Even1,Even2...EvenN)->EstadoS

Se requieren varios eventos para habilitar el paso del estado a otros varios estados que se ejecutan en paralelo.

f(EstadoA, Even1,Even2...EvenN)-> Estado1, Estado2…, EstadoN

Una estructura de Red de Petri es una 4-upla

C= (P, T, I, O)

Lugares: P= {P1, P2, Pm}  
Transiciones: T= {T1, T2, Tn}  
Funcion de entrada: I:T->P  
Funcion de salida: O:T->P

Los arcos indican a través de una flecha la relación entre sitios y transiciones y viceversa.

A los lugares se les asignan tokens (fichas) que se representan mediante un número o puntos dentro del sitio. Esta asignación de tokens a lugares constituye la marcación.

Luego de una marcación inicial se puede simular la ejecución de la red. El número de tokens asignados a un sitio es ilimitado.

Funciones de entrada salida:



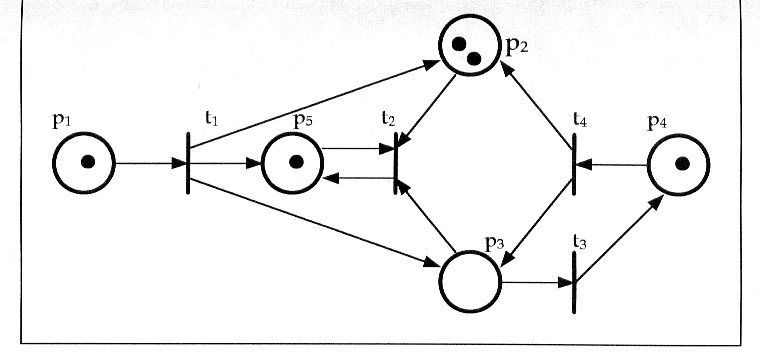
I: T->P O: T->P

I(t1) ={P1} O(t1) = {P2, P3, P5}

I(t2) = {P2, P3, P5} O(t2) ={P5}

I(t3) ={P3} O(t3) ={P4}

I(t4) ={P4} O(t4) = {P2, P3}

Marcación inicial: 

M (P1, P2, P3, P4, P5) = (1,2,0,1,1)

El conjunto de tokens asociado a cada estado sirve para manejar la coordinación de eventos y estados.

Una vez que ocurre un evento, un token puede “viajar” de uno de los estados a otro.

Las reglas de disparo provocan que los tokens “viajen” de un lugar a otro cuando se cumplen las condiciones adecuadas.

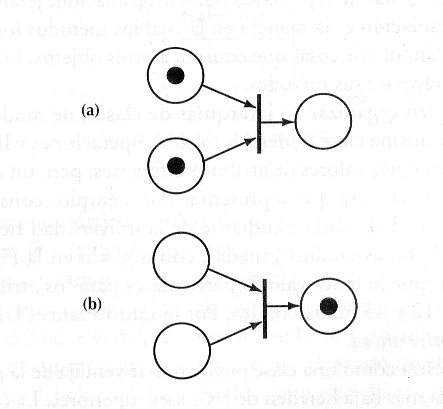
La ejecución es controlada por el número y distribución de los tokens.

La ejecución de una Red de Petri se realiza disparando transiciones habilitadas.

Una transición está habilitada cuando cada lugar de entrada tiene al menos tantos tokens como arcos hacia la transición.

Disparar una transición habilitada implica remover tokens de los lugares de entrada y distribuir tokens en los lugares de salida (teniendo en cuenta la cantidad de arcos que llegan y la cantidad de arcos que salen de la transición).

La transición está habilitada



La transición no está habilitada

La ocurrencia de los eventos (transiciones) depende del estado del sistema.

Una condición puede ser V (con token) o F (sin token)

La ocurrencia de un evento está sujeta a que se den ciertas condiciones (pre) y al ocurrir el evento causa que se hagan verdaderas las post-condiciones.

Las RP son asincrónicas y el orden en que ocurren los eventos es uno de los permitidos

* La ejecución es NO DETERMINÍSTICA

Se acepta que el disparo de una transición es instantáneo.

P0

P1

P2

P3

P4

P5

P6

P7

T0

T1

T2

T3

T 4

Sincronización:

Para que varios procesos colaboren en la solución de un problema es necesario que compartan información y recursos, pero esto debe ser controlado para asegurar la integridad y correcta operación del sistema.

Expresión de exclusión mutua:

P0

P1

P2

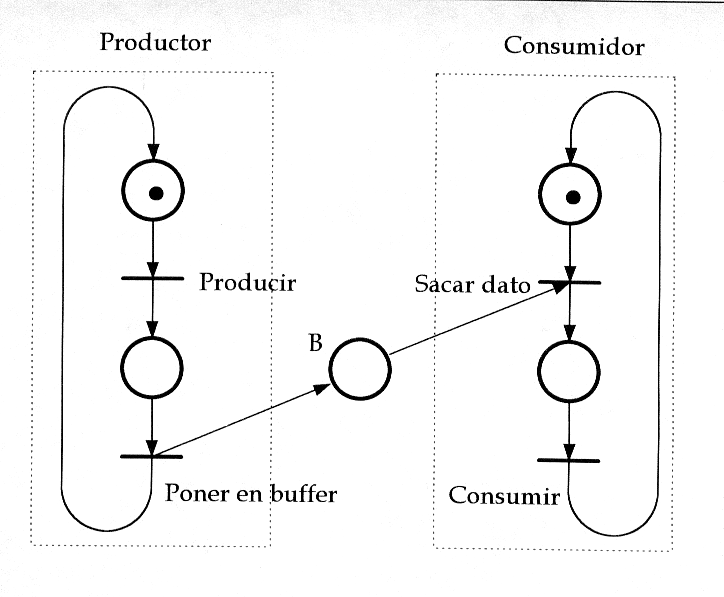
P3

P4

T0

T1

Productor - Consumidor



Condición de bloqueo:

P0

P1

P2

P3

T1

T2

Es importante desarrollar modelos de los sistemas de eventos discretos para estudiarlos y comprender su comportamiento.

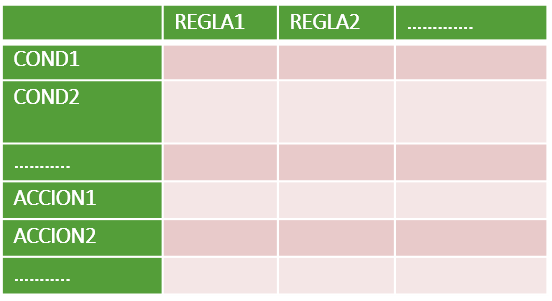
Existen herramientas computacionales que permiten analizar este tipo de sistemas, las cuales están basadas en análisis estadísticos y ofrecen soluciones con ciertos grados de incertidumbre.

Por otro lado, las RP pueden ser aplicadas para la modelación de sistemas de eventos discretos, las cuales ofrecen una forma de representación gráfica y matemática de los sistemas modelados.

La formalidad matemática de la RP proporciona herramientas de análisis para analizar los posibles estados a los que el sistema modelado pudiera alcanzar.

**Tablas de decisiones**

* Tablas de Decisión
  + - **Es una herramienta que permite presentar de forma concisa las reglas lógicas que hay que utilizar para decidir acciones a ejecutar en función de las condiciones y la lógica de decisión de un problema específico**.
* Describe el sistema como un conjunto de:
  + - Posibles CONDICIONES satisfechas por el sistema en un momento dado
    - REGLAS para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y ACCIONES a ser tomadas como un resultado.
    - Construiremos las tablas con:
      * condiciones simples y acciones simples
      * Las condiciones toman sólo valores Verdadero o Falso
      * Hay 2N Reglas donde N es la cantidad de condiciones



¿Cómo se llena la tabla?

A partir de un enunciado se debe:

* 1. Identificar las condiciones y las acciones.
  2. Completar la tabla teniendo en cuenta:
     1. Si hay condiciones que son opuestas, debe colocarse una de ellas porque por la negativa se “obtendrá” la otra. (Si son n condiciones excluyentes, colocar n-1 en la tabla).
     2. Las condiciones deben ser atómicas.
  3. Se construyen las reglas

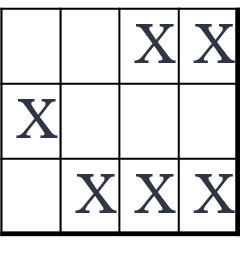
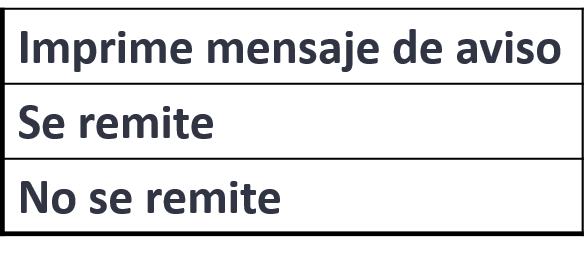
Modelizar el problema de remisión de mercadería con las siguientes consideraciones:

Si el comprador no es cliente se imprime un mensaje de aviso y no se remite.

Si no hay stock y el comprador es cliente no se remite.

Si hay stock y el comprador es cliente se remite.



**Especificaciones completas**

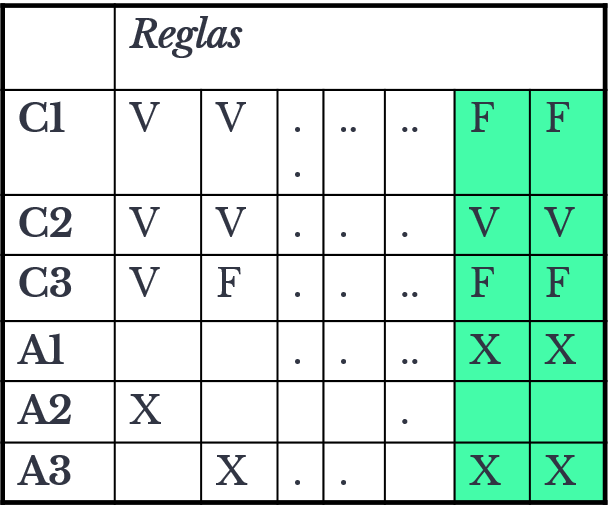
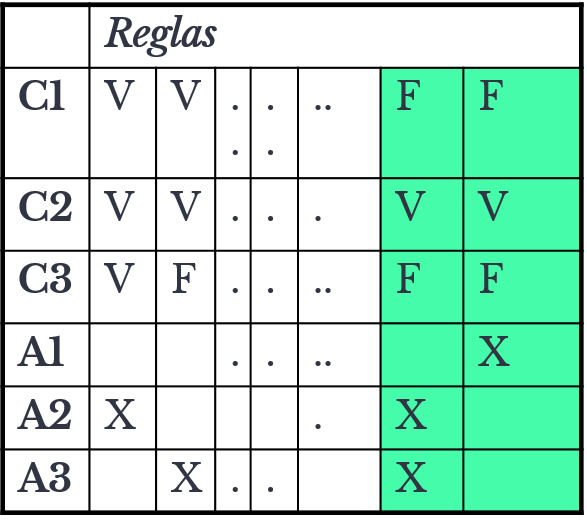
Aquellas que determinan acciones (una o varias) para todas las reglas posibles.

**Especificaciones redundantes**

Aquellas que marcan para reglas que determinan las mismas condiciones acciones iguales.

**Especificaciones contradictorias**

Aquellas que especifican para reglas que determinan las mismas condiciones acciones distintas.



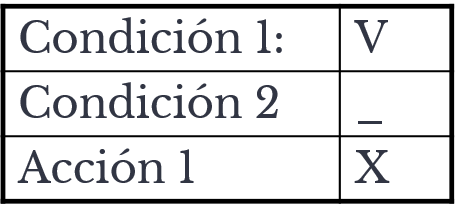
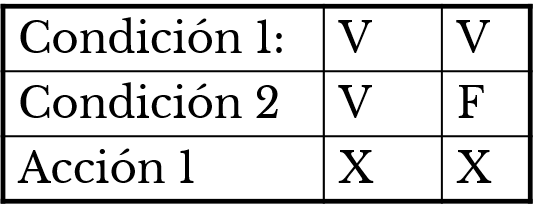
**Redundante**

**Contradictoria**

Reducción de Complejidad (Redundancia)

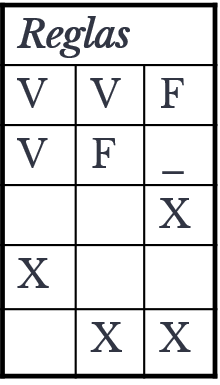
Combine las reglas en donde sea evidente que una alternativa no representa una diferencia en el resultado.

El guion [—] significa que la condición 2 puede ser V o F, y que aun así se realizará la acción.

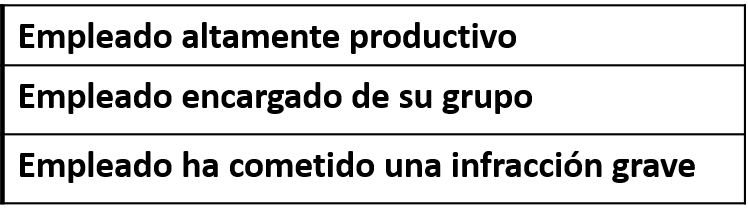


Reducción de Complejidad (Redundancia)

Álgebra de Boole

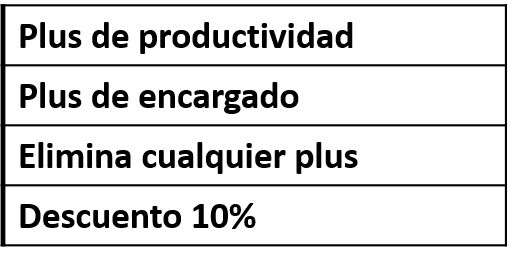


Identificar las condiciones y las acciones.

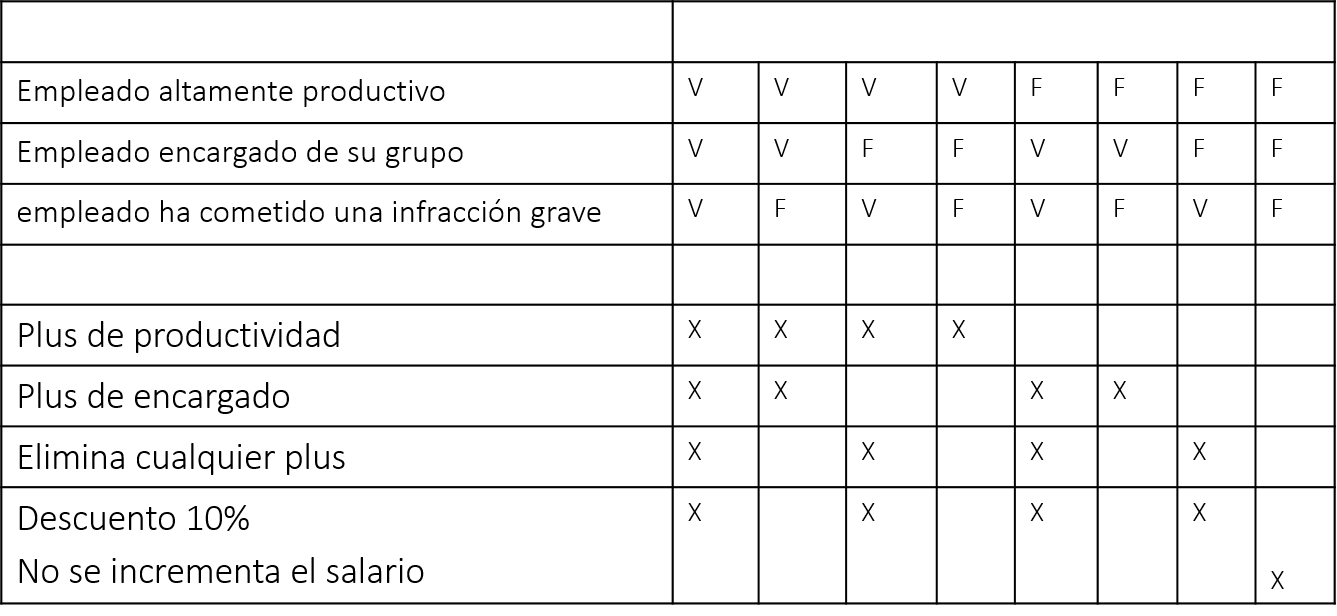


**Condiciones**

**Acciones**



Completar la tabla



**Análisis estructurado**

* Para entender los requerimientos, se debe poder reconocer además cómo se mueven los datos, los procesos o transformaciones que sufren dichos datos y sus resultados.
* La elicitación proporciona una descripción verbal del sistema, una descripción visual puede consolidar la información.
* La técnica de análisis estructurado permite lograr una representación gráfica que permite lograr una comprensión más profunda del sistema a construir y comunicar a los usuarios lo comprendido.
* La notación no especifica aspectos físicos de implementación.
* Hace énfasis en el procesamiento o la transformación de datos conforme estos pasan por distintos procesos.

**Diagrama de Flujo de Datos (DFD)**

* Es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por “conductos” y almacenamientos de datos.
* Representa la transformación de entradas a salidas y es también llamado diagrama de burbujas.
* Es una herramienta comúnmente utilizada por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja.

Se utiliza un rectángulo para representar una ***entidad externa****,* esto es, un elemento del sistema (por ejemplo, un elemento hardware, una persona, otro programa) u otro sistema que produce información para ser transformada por el software, o recibe información producida por el software.

Un círculo (también llamado burbuja) representa un ***proceso*** o *transformación* que es aplicado a los datos (o al control) y los modifica.

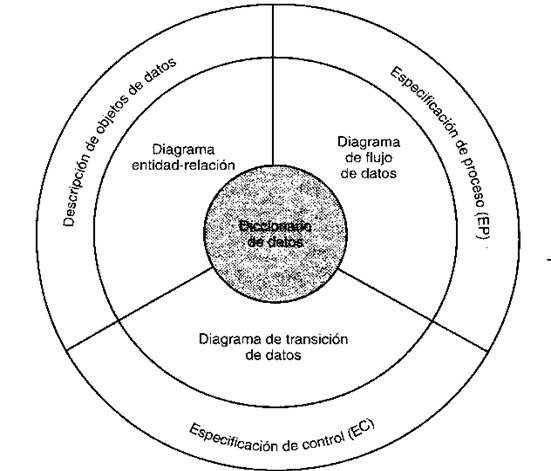
Una flecha representa un ***flujo*** para uno o más ***elementos de datos*** (objetos de dato).

Un rectángulo abierto (lado izquierdo y derecho) que representa un **almacén de datos.**

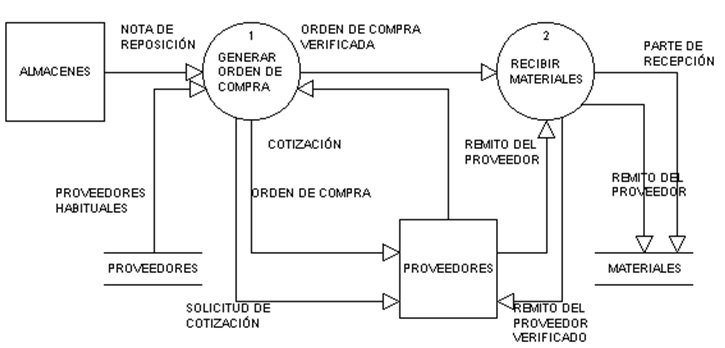
Listado organizado de todos los datos pertinentes al sistema. Es una definición sin ambigüedad de los datos y elementos del sistema

Características

* + - Permite revisar consistencia
    - Representa el contenido de la información
    - Define el significado de los flujos y los almacenes
    - Un Dato debe contener
      * Tipo
      * Nombre
      * Descripción







Flujos de datos

Proceso

Almacén

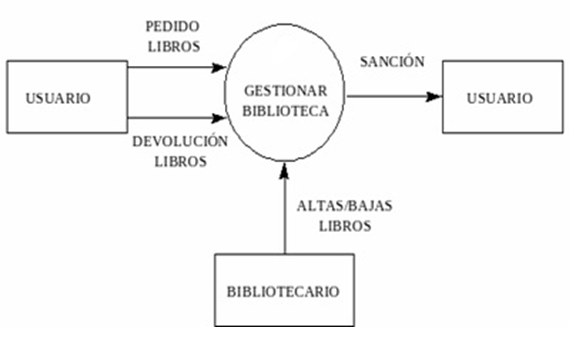
Entidad Externa

Se debe visualizar desde una perspectiva jerárquica de arriba hacia abajo.

Pasos :

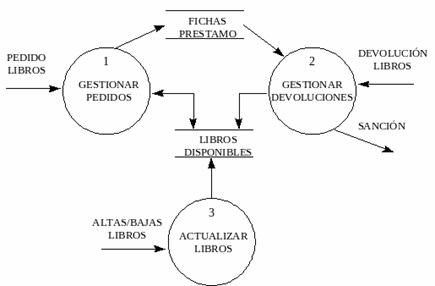
1. Redactar la lista de actividades (eventos) de la organización para determinar:
   * Entidades externas
   * Flujos de datos
   * Procesos
   * Almacenes de datos
2. Crear un diagrama de contexto que muestre las entidades externas y los flujos de datos desde y hacia el sistema.
3. Dibujar el Diagrama 0 (siguiente nivel), con procesos generales y los almacenes correspondientes

Dibujar un diagrama hijo por cada uno de los procesos del Diagrama 0



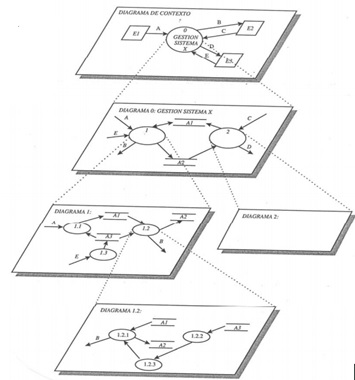
Se muestra un panorama global que muestre las entradas básicas y las salidas

Es el nivel más alto en un DFD y contiene un solo proceso que representa a todo el sistema



Es la ampliación del Diagrama de contexto.

Las entradas y salidas del Diagrama de contexto permanecen, sin embargo, se amplía para incluir hasta 9 procesos (como máximo) y mostrar los almacenes de datos y nuevos flujos.



Cada proceso se puede a su vez ampliar para crear un diagrama hijo más detallado.

Las entradas y salidas del proceso padre permanecen, sin embargo, pueden aparecer nuevos almacenes de datos y nuevos flujos.

**Nivelación de un DFD**

A partir del DFD preliminar se realizan nivelaciones

**Ascendentes**

Agrupa las burbujas con algún criterio

**Descendentes**

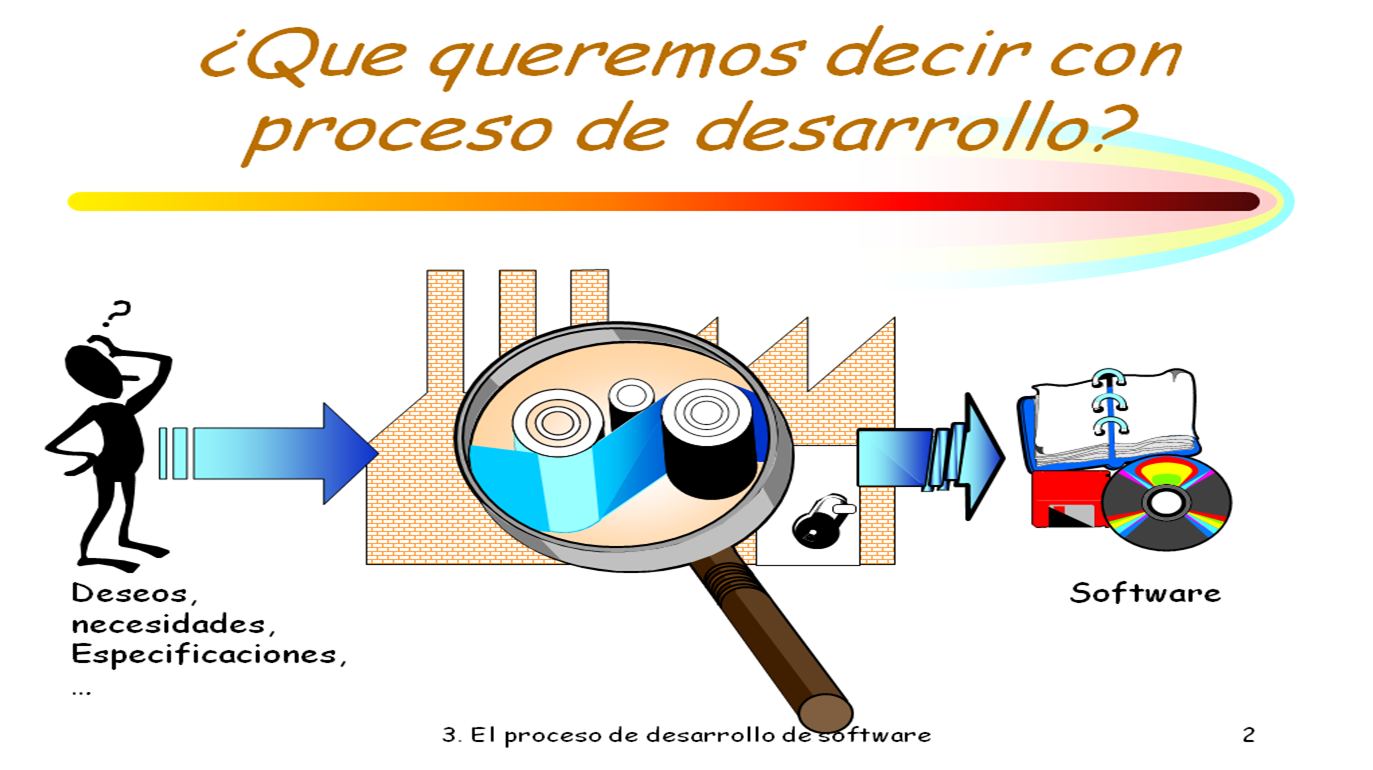
Descompone las burbujas funcionalmente

**Procesos**

* Cuando proveemos un servicio o creamos un producto, siempre seguimos una secuencia de pasos para realizar un conjunto de tareas.
* Las tareas son realizadas usualmente en el mismo orden.
* Por ejemplo, no se puede cocinar una torta antes de que todos los ingredientes sean mezclados.
* Se puede pensar que un “conjunto ordenado de tareas” como un proceso.

**¿Qué es un proceso de software?**

Es un conjunto de actividades y resultados asociados que producen un producto de software.



Actividades fundamentales de los procesos:

* + Especificación del software
  + Desarrollo del software
  + Validación del software
  + Evolución del software

**Especificación del software**

* Consiste en el proceso de comprender y definir qué servicios se requieren del sistema, así como la identificación de las restricciones sobre la operación y desarrollo del sistema.

También llamada *Ingeniería de Requerimientos.*

**Desarrollo del software**

Corresponde al proceso de convertir una especificación del sistema en un sistema ejecutable.

Incluye los procesos de diseño y programación.

Se crea una descripción de la estructura del software que se va a implementar, los modelos y estructuras de datos, las interfaces, etc.

**Validación del software**

Se realiza para mostrar que un sistema cumple tanto con sus especificaciones como con las expectativas del cliente.

La prueba del sistema con datos de prueba simulados es una de las formas de validación.

Pero tambien incluye inspecciones y revisiones en distintas etapas.

**Evolución del software**

El mantenimiento es una actividad para tener en cuenta en el proceso de desarrollo de software. Eso implica también cambios y mejoras.

**¿Qué es un modelo de proceso de software?**

* Es una representación simplificada de un proceso de software que presenta una visión de ese proceso.
* Estos modelos pueden incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.
* Marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso

**Modelos de Proceso**

Características

* Establece todas las actividades.
* Utiliza recursos, está sujeto a restricciones y genera productos intermedios y finales.
* Puede estar compuesto por subprocesos.
* Cada actividad tiene entradas y salidas definidas.
* Las actividades se organizan en una secuencia.
* Existen principios que orientan sobre las metas de cada actividad.
* Las restricciones pueden aplicarse a una actividad, recurso o producto.

**Ciclo de vida**

Proceso que implica la construcción de un producto

**Ciclo de vida del Software**

Describe la vida del producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento

**Modelos de proceso de software**

Es una representación abstracta de un proceso del software

**Términos Equivalentes**

**Modelo de proceso**

**Paradigma de software**

**Ciclo de vida del software**

Modelos **prescriptivos**

Prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de la ingeniería del software, tareas, aseguramiento de la calidad y mecanismos de control.

Cada modelo de proceso prescribe también un “flujo de trabajo”, es decir de qué forma los elementos del proceso se interrelacionan entre sí.

Modelos **descriptivos**

Descripción en la forma en que se realizan en la realidad.

**Ambos modelos deberían ser iguales**

**Modelos tradicionales**

Formados por un conjunto de fases o actividades en las que e no tienen en cuenta la naturaleza evolutiva del software

* Clásico, lineal, secuencial o en cascada
* Modelo en V
* Basado en prototipos

**Modelos evolutivos**

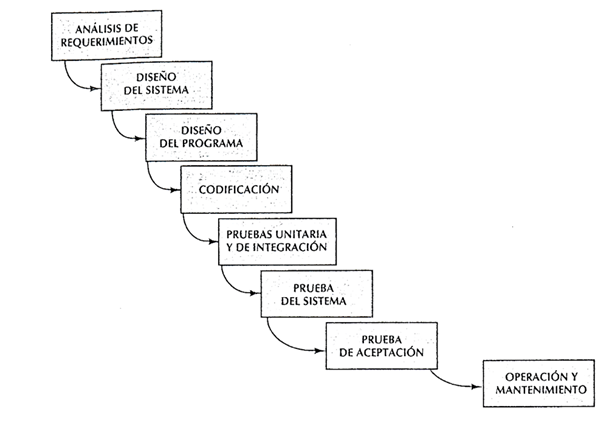
Son modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo

* En espiral
* Evolutivo
* Incremental

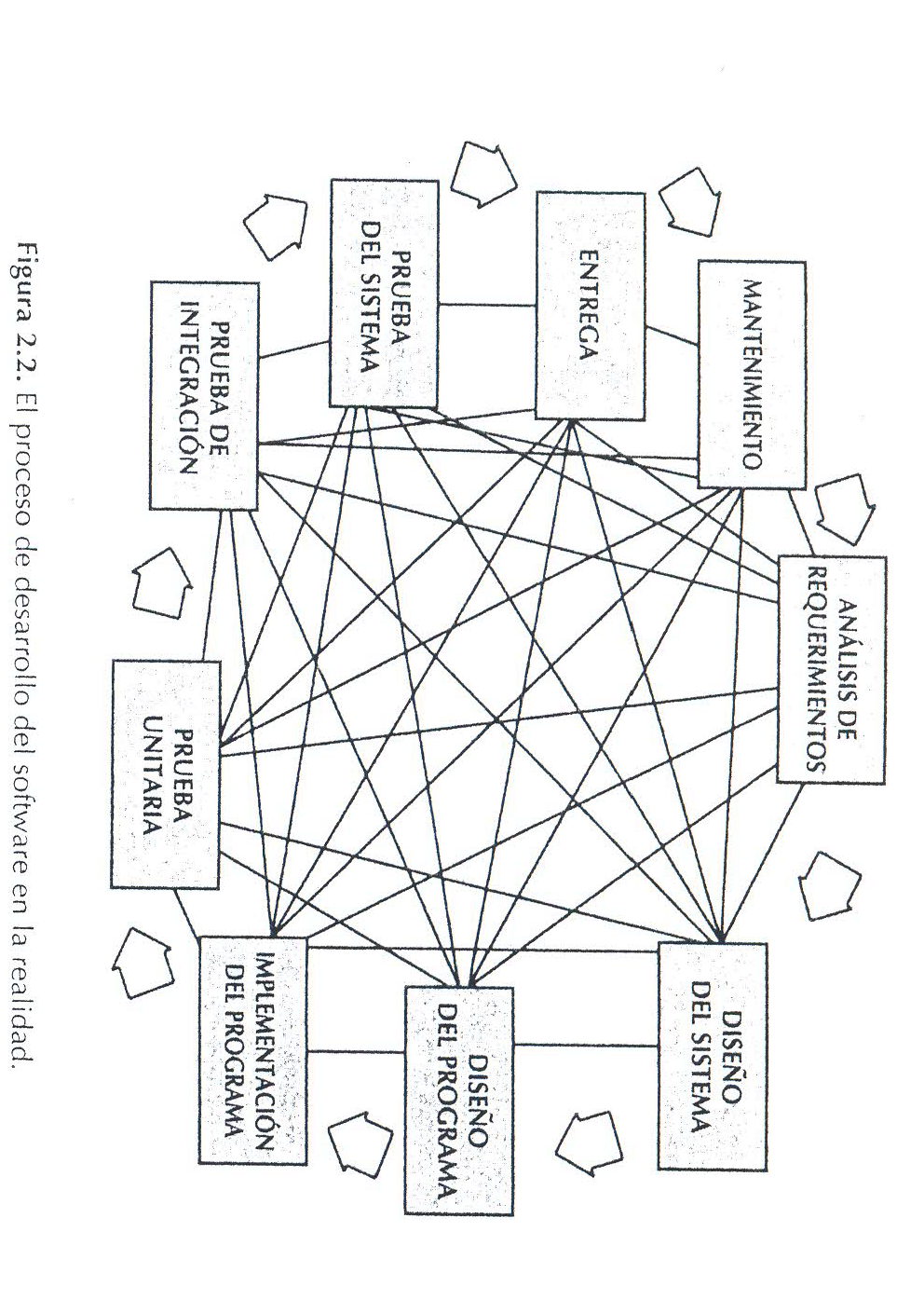
**Procesos ágiles**

**Modelo en cascada**

* Las etapas se representan cayendo en cascada
* Cada etapa de desarrollo se debe completar antes que comience la siguiente
* Útil para diagramar lo que se necesita hacer
* Su simplicidad hace que sea fácil explicarlo a los clientes



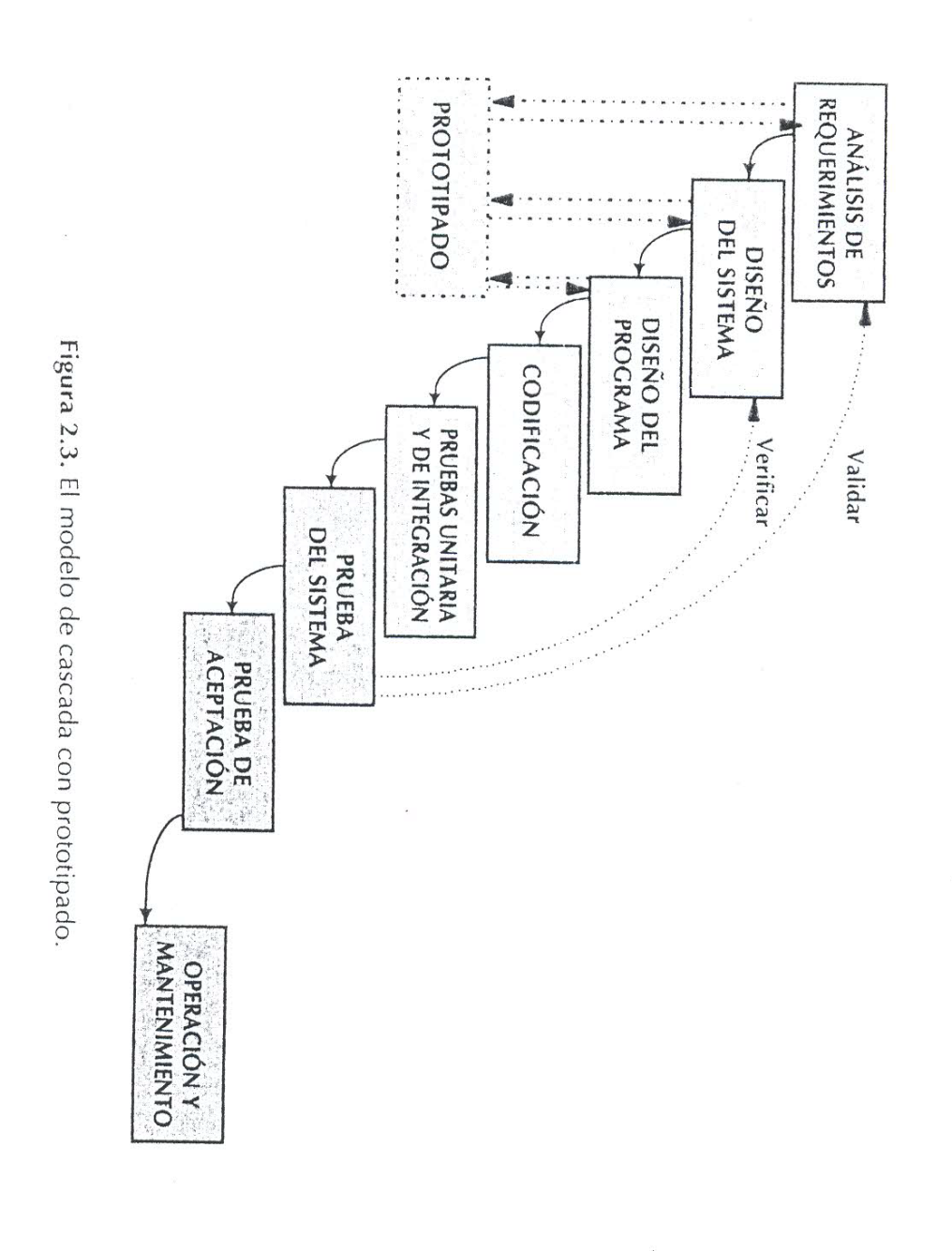
Modelo de la realidad (sin control entre las etapas)



Dificultades:

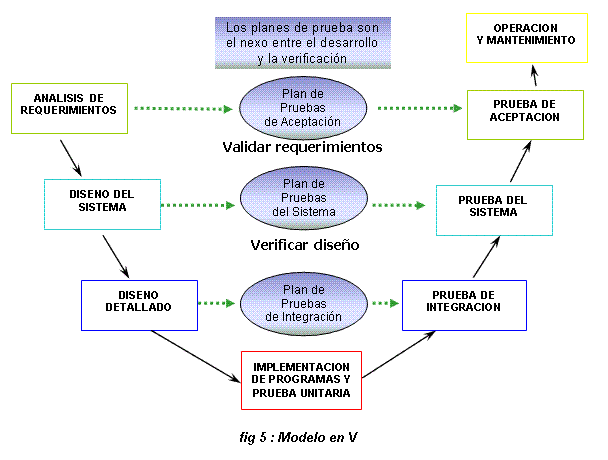
* No existen resultados concretos hasta que todo esté terminado.
* Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final.
* La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema.
* Deriva del mundo del hardware y presenta una visión de manufactura sobre el desarrollo de software.
* La necesidad de pruebas aumenta exponencialmente durante las etapas finales.
* "CONGELAR" una fase es poco realista.
* Existen errores, cambios de parecer, cambios en el ambiente.

**Modelo en cascada con prototipado**



**Modelo en V**

* Demuestra cómo se **relacionan las actividades de prueba** con las de análisis y diseño.
* Sugiere que **la prueba unitaria y de integración** también sea utilizada para verificar el diseño del programa
* **La vinculación entre los lados derecho e izquierdo** implica que, si se encuentran problemas durante la verificación y validación, entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema.



**Modelo de prototipos**

Un prototipo es un producto parcialmente desarrollado que permite que clientes y desarrolladores examinen algunos aspectos del sistema propuesto, y decidan si éste es adecuado o correcto para el producto terminado.

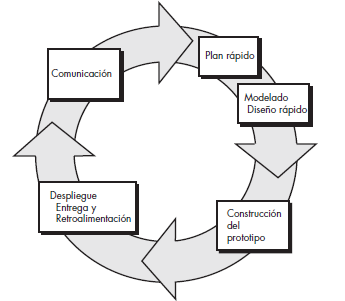
Esta es una alternativa de especificación para tratar mejor la incertidumbre, la ambigüedad y la volubilidad de los proyectos reales.

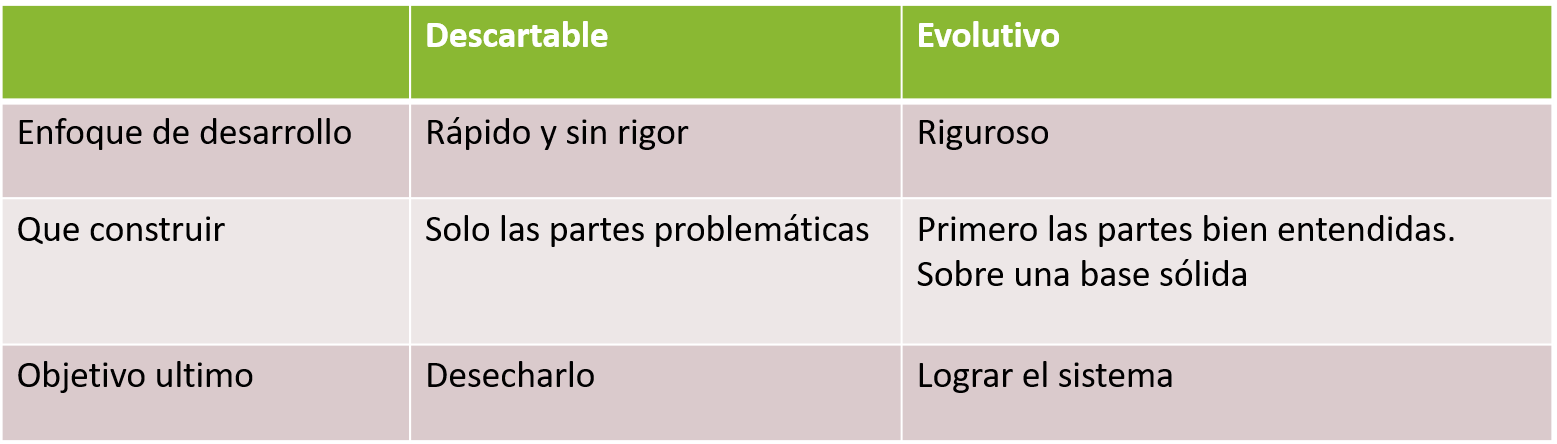
**Evolutivos**

* El objetivo es obtener el sistema a entregar.
  + Permite que todo el sistema o alguna de sus partes se construyan rápidamente para comprender o aclarar aspectos y asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución

**Descartables**

* No tiene funcionalidad
* Se utilizan herramientas de modelado





Proyectos candidatos

* Usuarios que no examinarán los modelos abstractos
* Usuarios que no determinarán sus requerimientos inicialmente
* Sistemas con énfasis en los formatos de E/S más que en los detalles algorítmicos
* Sistemas en los que haya que explorar aspectos técnicos
* Si el usuario tiene dificultad al tratar con los modelos gráficos para modelar los requerimientos y el comportamiento
* Si se enfatiza el aspecto de la interfaz humana

Para asegurar el éxito:

* Debe ser un sistema con el que se pueda experimentar
* Debe ser comparativamente barato (< 10%)
* Debe desarrollarse rápidamente
* Énfasis en la interfaz de usuario
* Equipo de desarrollo reducido
* Herramientas y lenguajes adecuados

**Modelo de desarrollo por fases**

Se desarrolla el sistema de tal manera que puede ser entregado en piezas. Esto implica que existen dos sistemas funcionando en paralelo: el sistema operacional y el sistema en desarrollo.

figura3

**Tipos de modelos de desarrollo por fases**

**Incremental**

* El sistema es particionado en subsistemas de acuerdo con su funcionalidad. Cada entrega agrega un subsistema.

**Iterativo**

* Entrega un sistema completo desde el principio y luego aumenta la funcionalidad de cada subsistema con las nuevas versiones.

**Modelo en espiral (Boehm)**

* Combina las actividades de **desarrollo con la gestión del riesgo**
* Trata de mejorar los ciclos de vida clásicos y prototipos.
* Incorpora objetivos de calidad
* Elimina errores y alternativas no atractivas al comienzo
* Permite iteraciones, vuelta atrás y finalizaciones rápidas
* Cada ciclo empieza identificando:
  + Los objetivos de la porción correspondiente
  + Las alternativas
  + Restricciones
* Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente

