¿Qué es Software?

Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación (IEEE)

-Tipos

- **Genéricos:** Sistemas aislados producidos por organizaciones desarrolladoras de software y que se venden en un mercado abierto.
- **Personalizados:** Sistemas requeridos por un cliente en particular. Desarrollados por la propia organización interesada o un contratista

-Clasificación

- **De sistemas:** sirve a otros programas
- **De aplicación:** resuelven necesidades de negocios especificas
- **Científico y de ingeniería:** algoritmos de manejo de números
- **Empotrado:** reside en memoria
- **De línea de productos:** mercados masivos
- **Basados en la Web:** sitios
- **De inteligencia artificial:** uso de algoritmos no numéricos para resolver problemas complejos

-Nuevos retos

- *Computación Ubicua:* integración de la informática en el entorno de la persona, de forma que no se perciba como objeto diferenciado
- *Proveedor de contenido* (world wide web)
- *Software libre*

-Características:

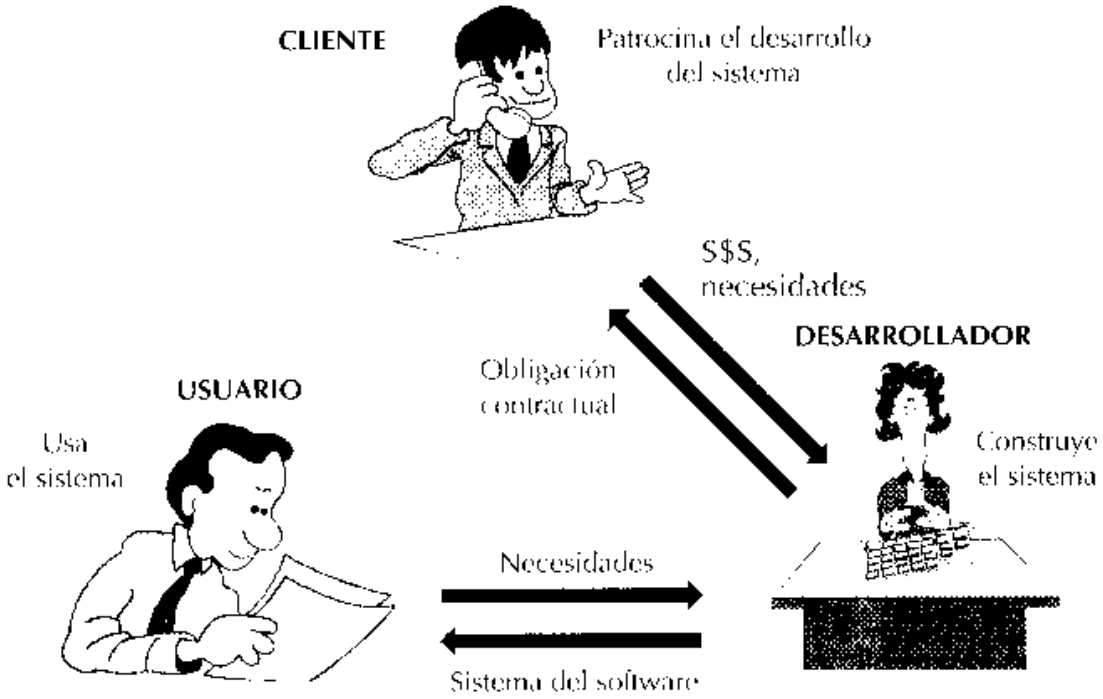
- Es un elemento lógico
- Se desarrolla, no se fabrica; por lo que tiene mayor costo en la ingeniería que en la producción, y por lo tanto no se pueden gestionar como proyectos de fabricación.
- No se desgasta.
- No sigue una curva clásica de envejecimiento.
- Es inmune a los males que desgastan al hardware.
- Incremento del índice de fallos por efectos laterales: CAMBIOS (el problema no está en el tiempo de operación)
- La mayoría se construye a medida (aunque la industria tienda a ensamblar componentes)

¿Qué es la ingeniería de software?

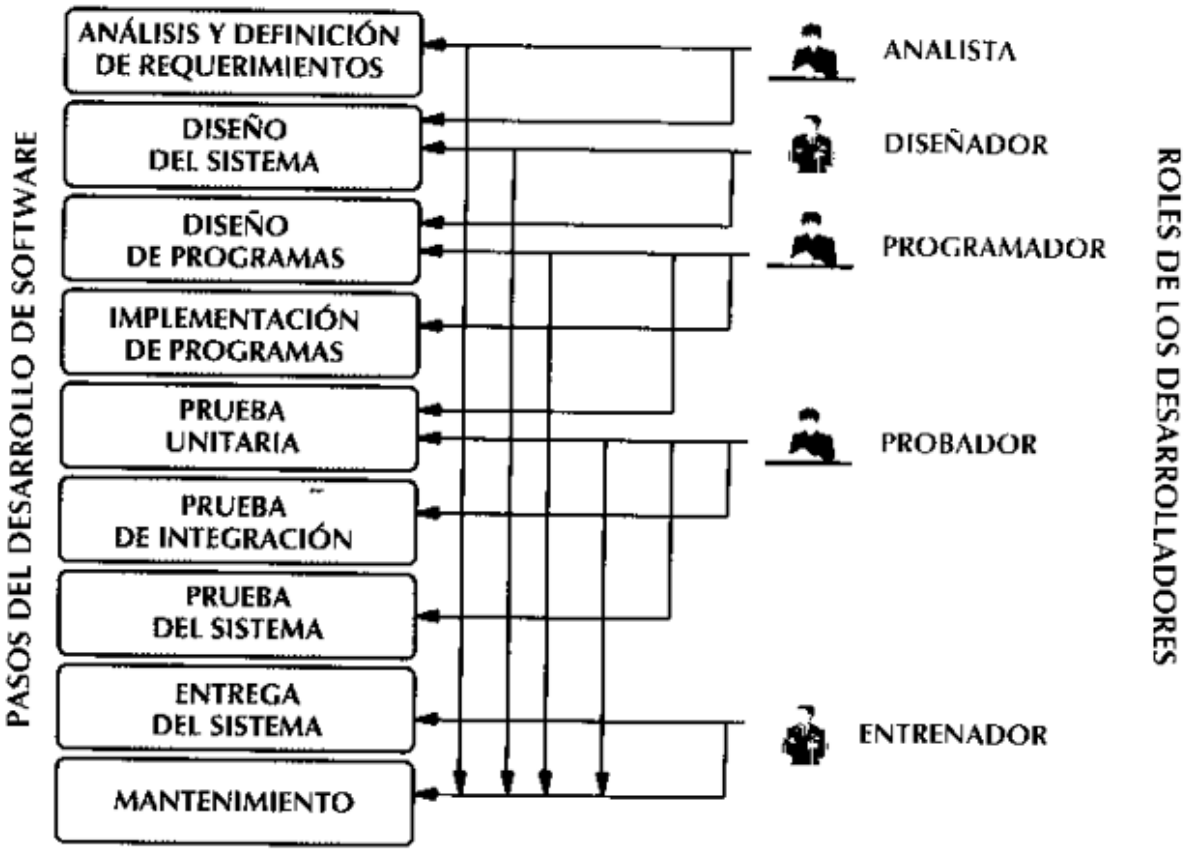
Disciplina de la ingeniería que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema, incluyendo la evolución de éste luego de que se comienza a ejecutar. Es decir, trata de dar principios y métodos que permitan producir software confiable y eficiente, al menor costo posible, estableciendo métodos, desarrollando herramientas automáticas o semiautomáticas y definiendo procedimientos que establezcan la relación entre los métodos y las herramientas.

- ➔ Usa métodos sistemáticos cuantificables: La cuantificación rigurosa de recursos, procesos y productos es una precondición para optimizar la productividad y calidad. La metrificación y el control estadístico de procesos son claves para la IS
- ➔ Dentro de los tiempos y costos estimados: se deben cumplir contratos de tiempo y costos. Capacidad de medir, estimar y administrar proyectos.
- ➔ Para el desarrollo , operación y mantenimiento: se ocupa de todo el ciclo de vida del producto, desde su etapa inicial de planificación y análisis de requerimientos hasta la estrategia para determinar cuándo y cómo debe ser retirado de servicio.

Participantes en el desarrollo de Software



Miembros del equipo de desarrollo de Software



Técnicas de comunicación

La comunicación es la base para la obtención de las necesidades (requerimientos) del cliente, y a su vez la principal fuente de error:
-Falta de procedimientos y guías formales
-Falta de participación del usuario
-Mala interpretación de las necesidades
-Falta de comunicación

Requerimientos
Un requerimiento es una característica de un sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema.

Los requerimientos del sistema especifican lo que el sistema de información deberá hacer o cuál propiedad o cualidad debe de tener éste. Los requerimientos del sistema que especifican lo que el sistema de información debe hacer son frecuentemente llamados **requerimientos funcionales**. Y aquellos que especifican una propiedad o cualidad que el sistema debe tener con frecuencia son llamados **requerimientos no funcionales**.

Etapas del proceso de ingeniería de requerimientos:

- **Estudio de viabilidad:** La entrada de este es un conjunto de requerimientos de negocio preliminares, una descripción resumida del sistema y de cómo este pretende contribuir a los procesos del negocio. Los resultados del estudio de viabilidad debería ser un informe que recomiende si merece o no la pena seguir con la ingeniería de requerimientos. Es un estudio corto.
- **Obtención y análisis de requerimientos:** en esta actividad los ingenieros de software trabajan con los clientes y los usuarios finales del sistema para determinar el dominio de la aplicación, qué servicios debe proporcionar, el rendimiento requerido del sistema, las restricciones de hardware, etc.
  Procesos de obtención y análisis:
  - **Descubrimientos de requerimientos:** es el proceso de interactuar con los stakeholders del sistema para recopilar sus requerimientos. Los requerimientos del dominio de los stakeholders y la documentación también se descrubren durante esta actividad.

Además de los stakeholders, ya hemos visto que los requerimientos pueden venir del dominio de la aplicación y de otros sistemas que interactúan con el sistema a especificar. Todos éstos se deben considerar durante el proceso de obtención de requerimientos.
Estas fuentes de requerimientos (stakeholders, dominio, sistemas) se pueden representar como puntos de vista del sistema, donde cada no representa un subconjunto de requerimientos para el sistema. Cada punto de vista proporciona una perspectiva nueva en el sistema, pero estos no son completamente independientes. Por lo general coinciden parcialmente por lo que tienen requerimientos comunes.

Puntos de vista: Un punto clave del análisis orientado a puntos de vista es que reconoce varias perspectivas y proporciona un marco de trabajo para descubrir conflictos en los requerimientos propuestos por diferentes stakeholders. Los puntos de vista se pueden utilizar como una forma de clasificar los stakeholders y otras fuentes de requerimientos. Existen 3 tipos de puntos de vista:
- Puntos de vista de los interactuadores: Personas u otros sistemas que interactúan directamente con el sistema. Proporcionan requerimientos detallados del sistema que cubren las características e interfaces del mismo. (Ejemplo del cajero serían los clientes del banco y la base de datos de las cuentas bancarias).
- Puntos de vista indirectos: representan los stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo. Proporcionan requerimientos y restricciones de alto nivel. (la gerencia del banco y el personal de seguridad)
- Puntos de vistas del dominio: Representan las características y restricciones del dominio que influyen en los requerimientos del sistema. (los estándares que se han desarrollado para las comunicaciones interbancarias)

Entrevistas: Son parte de la mayoría de los procesos de ingeniería de requerimientos. Es difícil obtener conocimiento del dominio durante las entrevistas debido a dos razones, la terminología y jerga especifica del dominio y cierto conocimiento del dominio es tan natural de los stakeholders que o los encuentran difícil de explicar o piensan que es tan básico que no vale la pena mencionarlo.
Escenarios: las personas encuentran más fácil dar ejemplos de la vida real que descripciones abstractas. Pueden comprender y criticar un escenario de cómo podrían interactuar con un sistema de software.
Casos de uso: Son una técnica que se baja en escenarios para la obtención de requerimientos. Identifican las interacciones particulares con el sistema. Pueden ser documentadas con texto o vinculadas a modelos UNL

o **Clasificación y organización de requerimientos:** recopilación no estructurada de requerimientos, los organiza en grupos coherentes.
o **Ordenación pro prioridades y negociación de requerimientos:** Cuando existe muchos stakeholders los requerimientos entraran en conflicto entonces se ordenan según prioridades y se resuelven los requerimientos en conflicto a través de la negociación.
o **Documentación de los requerimientos:** documentos formales o informales.

- Especificación de requerimientos
- Validación de requerimientos

<u>Fuentes de requerimientos</u>

Documentación, stakeholders, especificaciones de sistemas similares.

Stakeholders: cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente. Entre los que se encuentran: usuarios finales, ingenieros, gerentes, expertos del dominio.

Puntos de vista: se puede utilizar como una forma de clasificar a los stakeholders.
Existen tres tipos genéricos:
-*De los interectuadores:* 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.
-*Del dominio:* representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

<u>Elicitación de requisitos</u>

Es el proceso de adquirir (“eliciting”) todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio del problema. Tiene principalmente un carácter social.

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 explícitas como implícitas, de clientes y usuarios y sus expectativas sosbre el sistema a desarrollar

Se pueden presentar:
-*Limitaciones cognitivas (del desarrollador):* no conocer el dominio del problema, hacer suposiciones sobre este y/o sobre aspectos tecnológicos, hacer simplicaciones excesivas.
-*Conducta humana:* conflictos y ambiguedades en los roles de los participantes; pasividad de clientes, usuarios o ingenieros de requisitos; temor al nuevo sistema lo deje sin trabajo.
-*Técnicos:* complejidad del dominio del problema y/o de los requisitos; múltiples fuentes de requisitos; fuentes de información poco claras.

Técnicas de elicitación

Muestreo de la documentación, los formularios y los datos existentes
Recolección de hechos a partir de documentación existente; documentación de sistemas anteriores.
Proceso de recolectar una muestra significativa de documentos, formas y registros.

Recolección de hechos a partir de la documentación existente:
¿Qué tipos de documentos pueden enseñar algo del sistema?
Organigrama (identificar el propietario y los usuarios claves)
Memos, notas internas, minutas, registros contables.
Solicitudes de proyectos de sistemas anteriores. Permiten conocer el historial que origina al proyecto.
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, procedimientos operativos.
- -formularios de operaciones realizadas.
- -base de datos
- -sistemas en funcionamiento.

Documentación de sistemas anteriores:
- -Diagramas.
- -Diccionario o repositorios de proyecto
- -Documentos de diseño-
- -Manuales de operación y/o entrenamiento.

Investigación y visitas al lugar
Investigar el dominio, encontrar patrones de soluciones (mismo problema en otra organización); revistas especializadas, consultas sobre problemas similares.

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 perfil bajo.
- -Tomar nota de lo observado.
- -Revisar las notas con la persona apropiada.
- -No interrumpir a la persona en su trabajo.

-*Ventajas:* se obtienen datos confiables, ya que el analista puede ver exactamente lo que se hace (especialmente para tareas difíciles de explicar con palabras); permite realizar un análisis de las disposiciones físicas, tránsito, iluminación, ruido; económica.
-*Desventajas:* se genera incomodidad en la gente al ser observada y puede llegar a realizar las tareas de la forma "correcta y no como lo hace habitualmente; las actividades del sistema pueden ser realizadas en horarios incómodos y/o estar sujetas a interrupciones.

Cuestionarios
Documento que permite al analista recabar información y opiniones de los encuestados:
-Recolectar hechos de un gran número de personas
-Detectar un sentimiento generalizado
-Detectar problemas entre usuarios
-Cuantificar preguntar

Ventajas: se obtiene respuesta rápida y son económicos; anónimos y estructurados de fácil análisis.

Desventajas: número bajo de respuestas sin poder aclarar las incompletas; no responde a todas las preguntas y son muy rígidas, sin permitir un análisis corporal; no se puede aclarar respuestas incompletas y son difíciles de preparar.

Cuándo usarlos: personas dispersas geográficamente (diferentes oficinas o ciudades); muchas personas involucradas; se busca obtener opiniones generales y/o identificar problemas generales.

Tipos de cuestionarios
-**Formato libre** (abiertos): diseñado para ofrecer al encuestado más flexibilidad en la respuesta
-**Formato fijo** (cerrados): requieren la selección de una respuesta entre respuestas posibles predefinidas

Tipos de preguntas de los cuestionarios:
-Opción múltiple.
-De calificación.
-De jerarquización

Tipos de preguntas

	ABIERTAS	**CERRADAS**
VELOCIDAD DE CONCLUSIÓN	Lenta	Rápida
NATURALEZA EXPLORATORA	Alta	Poca
AMPLITUD Y PROFUNDIDAD	Alta	Poca
FACILIDAD DE PREPARACIÓN	Fácil	Difícil
FACILIDAD DE ANÁLISIS	Difícil	Fácil

Tipo de información obtenida: actitud (lo que dicen que quieren); creencias (lo que creen que es verdad); comportamiento (lo que realmente hacen); características (de las personas o cosas).

Entrevistas
Técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara, con un formato de preguntas y respuestas en general.
Es una conversación con un propósito específico 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, con la posibilidad de obtener retroalimentación de su parte; es posible adaptar las preguntas de acuerdo al mismo, y obtener información no verbal observando sus acciones y expresiones.
Desventajas: es costosa, así como en tiempo y recursos humanos; dependen en gran parte de las habilidades del entrevistador y, por lo general, no son aplicables a distancia.

Tipo de información obtenida: opiniones, objetivos, procedimientos informales, sentimientos.

Tipos de entrevistas:

-**Estructuradas** (cerradas): el encuestador tiene un conjunto específico de preguntas para hacer, dirigiéndose al usuario sobre un requerimiento puntual; lo que no permite adquirir un amplio conocimiento del dominio.
-**No estructuradas** (abiertas): el encuestador lleva a un tema general, sin preparación de preguntas específicas, iniciando con preguntas que no dependen del contexto.

5 Pasos etapas de una entrevista:

1) Leer material sobre los antecedentes
2) Establecer los objetivos
3) Decidir a quién entrevistar
4) Preparar al entrevistado
5) Decidir sobre los tipos de preguntas y estructura.

	ABIERTA	CERRADA
EVALUACIÓN	Difícil	Fácil
CANTIDAD REQUERIDA DE TIEMPO	Mucha	Poco
ENTRENAMIENTO REQUERIDO	Muy necesario	Limitado

PERMITE ESPONTANEIDAD	Mucho	Poco
PERMITE CONOCER AL ENTREVISTADO	Muchas oportunidades	Muy poco
FLEXIBILIDAD	Gran	Reducida
CONTROL DE LA ENTREVISTA	Bajo	Alto
PRECISIÓN	Baja	Alta
CONFIABILIDAD	Baja	Alta
AMPLITUD Y PROFUNDIDAD	Mucha	Poca

Tipos de preguntas:
-**Abiertas**: permite al encuestado responder de cualquier manera.
Ventajas: revelan nueva línea de preguntas, haciendo más interesante la entrevista y permitiendo espontaneidad.
Desventajas: pueden dar muchos detalles irrelevantes; se puede perder el control de la entrevista y que parezca que el entrevistador no tiene los objetivos claros.

-**Cerradas**: las respuestas son directas, cortas o de selección específica.
Ventajas: ahorran tiempo y se mantiene más fácil el control de la entrevista, pudiendo conseguir datos relevantes.
Desventajas: no se obtienen detalles, y se puede aburrir al encuestado.

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

Organización de una entrevista:
-**Piramidal**: es *inductivo*, as donde se comienza por preguntas cerradas que se van "abriendo", es decir siendo menos acotadas, hasta llegar a preguntas abiertas.

-**Embudo**: es *deductivo*, ya que de preguntas abiertas se va "encaminando" la entrevista y así acotando las preguntas para llegar a preguntas más concretas, cerradas.

-**Diamante** (*combinación*): es una organización de entrevista más variante, donde desde preguntas cerradas se llega a preguntas abiertas pero terminando más especificado en preguntas cerradas.

Cómo conducir la entrevista:
-->*Selección del entrevistado*:
-Según el requerimiento a analizar
-Armar la entrevista en base a las características de las personas
-Hacer una cita (no llegar sin avisar), respetando el horario de trabajo y obteniendo el permiso del supervisor o jefe.
-Establecer la duración de la entrevista
-La entrevista es personal y debe realizarse en un lugar privado

-->*Preparación de la entrevista*:
-Informar al entrevistado el tema a tratar antes de la reunión
-Definir un "Guión de Entrevista"
-Evitar preguntas sesgadas o con intención, amenazantes o críticas; no incluir opinión como parte de la pregunta
-Usar lenguaje claro y conciso; y evitar preguntas largas y complejas

Planeación Conjunta de Requerimientos (JRP) Join Requirements Planning
Proceso mediante el cual se conducen reuniones de grupo altamente estructurados con el propósito de analizar problemas y definir requerimientos (también conocida como JAD Joint Application Design).

Antes de planear una sesión de JRP el analista debe trabajar con el patrocinador para determinar el alcance del proyecto que se va abordar a través de las sesiones.

-Requiere de extenso entrenamiento
-Reduce el tiempo de exploración de los integrantes
-Amplia participación de los integrantes
-Se trabaja sobre lo que se va generando
- Dura de 3 a 5 días o hasta 2 semanas.
-El producto final es un documento realizado por el facilitador y los secretarios.
- Los participantes son_ patrocinador, facilitador, usuarios y gerentes (elegidos por el patrocinador), secretarios y equipo IT.

Ventajas: ahorra tiempo, involucra a los usuarios y genera desarrollos creativos.

Desventajas: es difícil organizar los horarios de los involucrados, y es complejo encontrar un grupo de participantes integrados y organizados.

Cómo planear las sesiones: se selecciona la ubicación para las sesiones, sus participantes y se prepara la agenda.
Beneficios del JRP
Involucra activamente a los usuarios y la gerencia en el proyecto de desarrollo.
Reduce el tiempo de la etapa de requerimientos
Si se incorporan prototipos, los mismos ya confirman el diseño del sistema.

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

Los principios en que se basa esta técnica son:
"Cuantas más ideas se sugieren, mejores resultados se conseguirán"
La producción de las ideas en grupos puede ser más efectiva que la individual.
Las ideas de una persona pueden 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, ya que encara la dificultad del usuario para transmitir → ayuda a entender: al usuario y al analista

¿Qué es un proceso de Software?
Es un conjunto de actividades y resultados asociados que producen un producto de software.

Los ingenieros en software son responsables de la especificación, desarrollo, validación y evolución del software.

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 parte de los procesos y productos de software, y el papel de las personas involucradas.

Historia sobre la IS

- Años 60 → *Funcional*: se estudia cómo explotar la tecnología para hacer frente a las necesidades funcionales de las organizaciones.
- Años 70 → *De Control*: se introduce el modelo de ciclo de vida en faces, por la aparición de la necesidad de desarrollar software en tiempo, planeado y controlado.
- Años 80 → *De Costos*: se incrementa sustancialmente la importancia de la productividad en el desarrollo de software; se ponen en práctica varios modelos de costos.
- Años 90 → *De Calidad*: se intensifica la necesidad de que el producto tenga atributos que satisfagan las necesidades explícitas e implícitas del usuario (mantenibilidad, confiabilidad eficiencia, usabilidad)

¿Qué conocimientos debe tener un IS?
-Combinación de conocimientos científicos, metodológicos, tecnológicos y administrativos.
-Estar familiarizado con la aplicación de métodos formales: lógica, estadística, simulación y el uso de notaciones de modelización, especificación, diseño, programación.
-Tener la capacidad de aplicar metodologías de documentación, análisis, especificación, diseño, implementación y prueba; conocer las ventajas y limitaciones de cada notación y cada técnica, junto con cómo y cuándo aplicarlas.
-Conocer las tecnologías y productos: sistemas operativos, lenguajes, herramientas CASE, bases de datos, sistemas generadores de interfaces, bibliotecas de código.
-Conocer las técnicas de administración de proyectos: planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, control de subcontratistas, etc.

Responsabilidad profesional y ética
-La ingeniería de software se desarrolla en un marco económico, social y legal, por lo que se deberán aceptar responsabilidades más amplias que las técnicas.
-No se debe utilizar las capacidades y habilidades de forma deshonesta, o de forma que deshonre la profesión. Por lo que se debe respetar:

- *Confidencialidad*, de empleados y clientes
- *Competencia*: no falsificar el nivel de competencia y aceptar responsabilidades fuera de su capacidad
- *Derechos de propiedad intelectual*: conocer las leyes vigentes sobre las patentes y copyright.
- *Computadoras: No debe utilizar sus habilidades técnicas para utilizar de forma inapropiada otras computadoras.*

- ¿Qué es un proceso de software?

Es un conjunto de actividades y resultados asociados que producen un producto de software: especificación, desarrollo, validación y evolución del software.

¿Qué es un modelo de proceso de software?

Es una representación simplificada de un proceso de software que presenta una visión de ese proceso; puede incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.

→ *Modelo en cascada*: las etapas se representan cayendo en cascada. Cada etapa de desarrollo se debe completar antes que comience la siguiente.

→ *Desarrollo iterativo*: un sistema inicial se desarrolla rápidamente a partir de una especificación abstracta. Éste se refina basándose en las peticiones del cliente.

→ *IS basada en componentes*: esta técnica supone que las partes ya existen. El proceso se enfoca en la integración de las partes.

Requerimientos

→Tipos de requerimientos:

- **Funcionales**:

-Describen una interacción entre el sistema y su ambiente: cómo debe comportarse el sistema ante determinado estímulo.

-Describen lo que el sistema debe o no debe hacer.

-Describen con detalle la funcionalidad del mismo.

-Son independientes de la implementación de la solución.

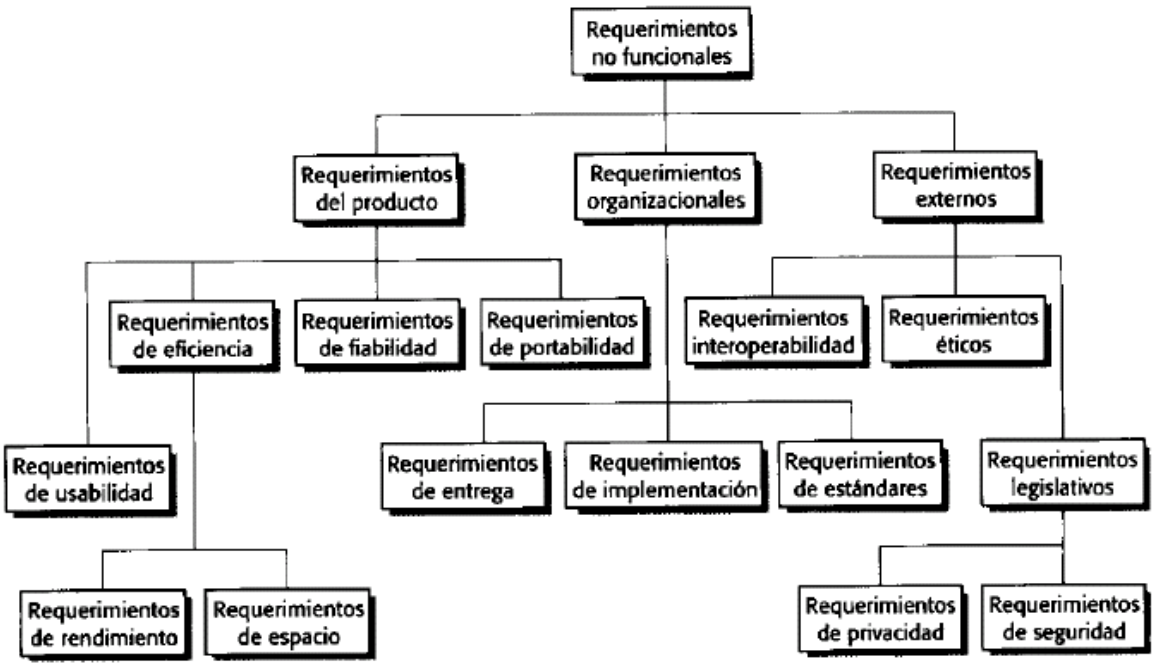
-Se pueden expresar de distintas formas.

- **NO Funcionales**: describen una restricción sobre el sistema que limita nuetras elecciones en la construcción de una solución al problema.

-*Del producto*: especifican el comportamiento del producto (usabilidad, eficiencia (rendimiento, espacio), fiabilidad, portabilidad)

-*Organizacionales*: se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares)

-*Externos*: interoperabilidad, legales, privacidad, seguridad, éticos.



Otras:

-*Del dominio*: reflejan las características y restricciones del dominio de aplicación del sistema, pueden ser funcionales o no funcionales y pueden restringir a los anteriores; cómo se especializan en el dominio son complicados de interpretar.

-*Por Prioridad*: deben ser absolutamente satisfechos; son deseables, pero no indispensables.

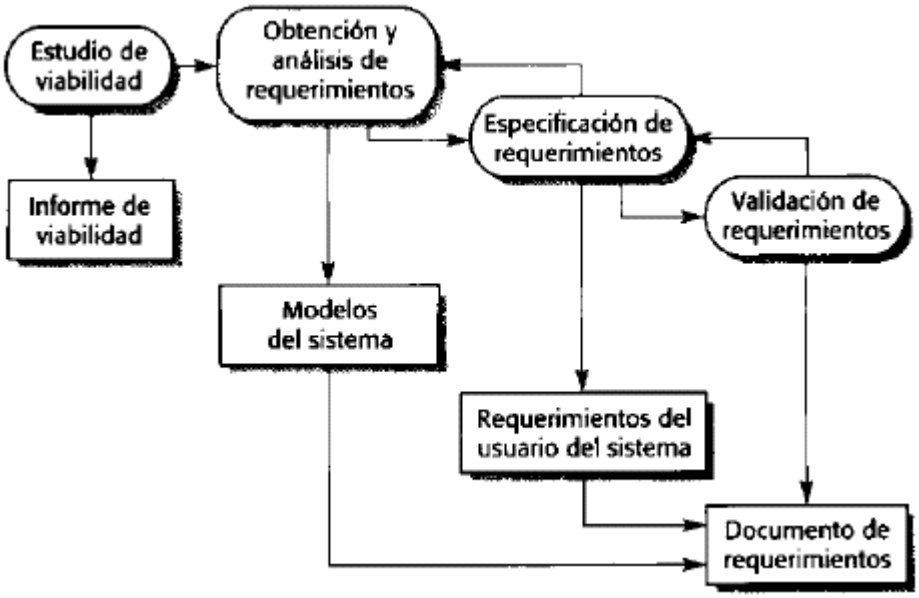
-*Del Usuario*: son declaraciones, en lenguaje natural y en diagramas de los servicios, que se espera que el sistema provea, y de las restricciones bajo las cuales debe operar. Pueden surgir problemas por falta de claridad (confusión de requerimientos, conjunción de requerimientos).

-*Del Sistema*: establecen con detalle los servicios y restricciones del sistema; es difícil excluir toda la información del diseño.

Ingeniería de requerimientos

Es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimientos y

limitaciones. También 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.
Importancia:
-Permite gestionar las necesidades del proyecto de 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



1 Estudio de viabilidad (Principalmente para sistemas nuevos)
A partir de una descripción resumida del sistema se elabora un informe que recomienda la conveniencia o no de realizar el proceso de desarrollo, a partir de una descripción resumida del sistema. Responde a las preguntas:
-¿El sistema contribuye a los objetivos generales de la organización?
-¿El sistema se puede implementar con la tecnología actual?
-¿El sistema se puede implementar con 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 pregunras anteriores se deberá hablar con las fuentes de información para responder a nuevas preguntas y luego redactar el informe.

2 Obtención y análisis de requerimientos
→ *Propiedades de los requerimientos*:

- **Necesario**: su omisión provoca una deficiencia
- **Conciso**: fácil de leer y entender
- **Completo**: no necesita ampliarse
- **Consistente**: no contradictorio con otro
- **No ambiguo**: tiene una sola interpretación
- **Verificable**: puede testearse a través de inspecciones, pruebas, etc.
-

3 Especificación de requerimientos
→ *Objetivos*:

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

→ 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.

Aspectos básicos:
-*Funcionalidad*: ¿qué debe hacer el software?
-*Interfaces externas*: ¿cómo interactuará el software con el medio externo (gente, hardware, otro software)?
-*Reducimiento*: velocidad, disponibilidad, tiempo de respuesta, etc.
-*Atributos*: portabilidad, seguridad, mantenibilidad, eficiencia
-*Restricciones de diseño*: estándares requeridos, lenguaje, límite de recursos, etc.

4 Validación de requerimientos
Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario. Es importante porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde.
La validación sólo se puede hacer con la activa participación del usuario, es hacer el software correcto; a diferencia de la verificación, que es hacer el software correctamente. Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar.

¿Es suficiente validar después del desarrollo del software?
-La evidencia estadística indica que no es suficiente validar después del desarrollo del software, ya que cuanto más tarde se detecta, más cuesta corregir. Es por ello que, validar en la fase de especificación de requerimientos puede ayudar a evitar costosas correcciones después del desarrollo.
¿Contra qué se verifican 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:
- *De validez*: para todos los usuarios
- *De consistencia*: sin contradicciones
- *De completitud*: todos los requerimientos
- *De realismo*: se pueden implementar
- *Verificabilidad:* se puede diseñar un conjunto de pruebas

→ Técnicas de validación:
-Pueden ser manuales o automatizadas
-Revisiones de requerimientos: antes de una revisión formal, conviene una informal.
- *Informales*: los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea posible.
- *Formales*: el equipo de desarrollo debe conducir al cliente, explicándole las implicaciones de cada requerimiento.

-Construcción de prototipos
-Generación de casos de prueba

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 cómo las relaciones cambian con el tiempo, por lo que es una descripción útil y adecuada para cuando el tiempo no es un factor mayor en la operación del sistema.
Descripción del sistema con una referencia indirecta al problema y su solución. Se define QUÉ se hace, no CÓMO.
Ejemplos:
-*Referencia indirecta (ecuaciones implícitas)*
-*Relaciones de recurrencia*: descripción del sistema mediante una función que define su valor en función de términos anteriores. Ej: expresar la serie de Fibonacci.
-*Definición axiomática*: Se definen las propiedades básicas de un sistema a través de operadores y axiomas (conjunto completo y consistente); se generan teoremas a través del comportamiento del sistema y se demuestran. Ej: sistemas expertos, definición de TADs, etc.
-*Expresiones regulares*: Se define un alfabeto y las combinaciones permitidas. Cuando un sistema procesa un conjunto de cadenas de datos, permite definir las cadenas de datos aceptables. Ej: alfabeto y sus combinaciones válidas.
-*Abstracciones de datos*: Para aquellos sistemas en los que los datos determinan las clases de acciones que se realizan (importa para qué son); se categorizan los datos y se agrupan los semejantes.

→ **Dinámicas**: Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo, es decir 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 estado, diagramas de persianas, redes de Petri, ect.

-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 resultado.

- Se construyen con condiciones simples y acciones simples; donde las condiciones sólo toman valores V o F y las reglas son de 2N, siendo n el nro de condiciones.
- Completar:
  - Identificar las condiciones y acciones
  - Completar la tabla teniendo en cuenta que, si las condiciones son opuestas, debe colocarse una de ellas ya que de la negativa se obtendrá la otra; y que deben ser atómicas.
  - Construir las reglas
- Especificaciones:
  - *Completas*: aquellas que determinan acciones (una o varias) para todas las reglas posibles.
  - *Redundantes*: aquellas que marcan para reglas que determinan las mismas condiciones acciones iguales.
  - *Contradictorias*: aquellas que especifican para reglas que determinan las mismas condiciones acciones distintas.
- Reducción de complejidad (redundancia): combine las reglas en donde sea evidente que una alternativa no representa una diferencia en el resultado.

DTE

→ Verificamos y validamos los requerimientos, pero éstos pueden cambiar: ¿Por qué cambian?
-Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas (en sistemas grandes hay una comunidad diversa de usuarios)
-Porque los clientes y los usuarios son distintos
-Porque cambió el problema que se estaba resolviendo
-Porque los usuarios cambiaron su forma de pensar o sus percepciones
-Porque cambió el ambiente de negocios (mercado, etc)

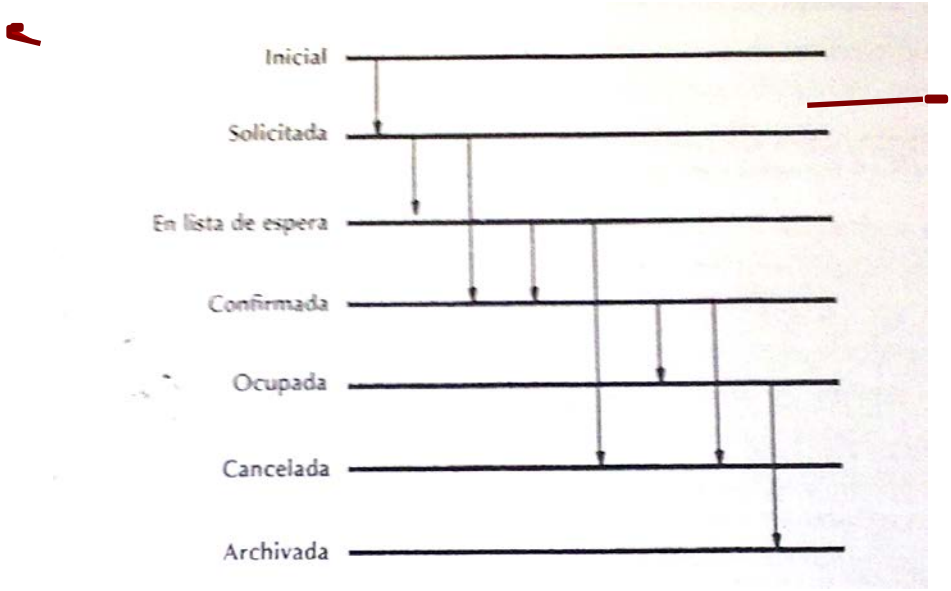
<u>Gestión de los requerimientos</u>
Es el proceso de comprender y controlar los cambios en los requerimientos; se debería comenzar cuando esté disponible una versión del documento de requerimientos.

→ Requerimientos duraderos: relativamente estables, se derivan de la actividad principal de la organización.
→ Requerimientos volátiles: cambian durante el desarrollo del sistema o después que se puso en operación (ej: cambios gubernamentales).
-Cambiantes: cambian porque se modifica el ambiente.
-Emergentes: surgen como ampliación (al incrementar la comprensión del cliente).
-Consecuentes: surgen por la introducción del sistema. Pueden cambiar los procesos de la organización por desarrollar nuevas formas de trabajo.
-Compatibilidad: cambian porque interactúan con otros sistemas que cambian.

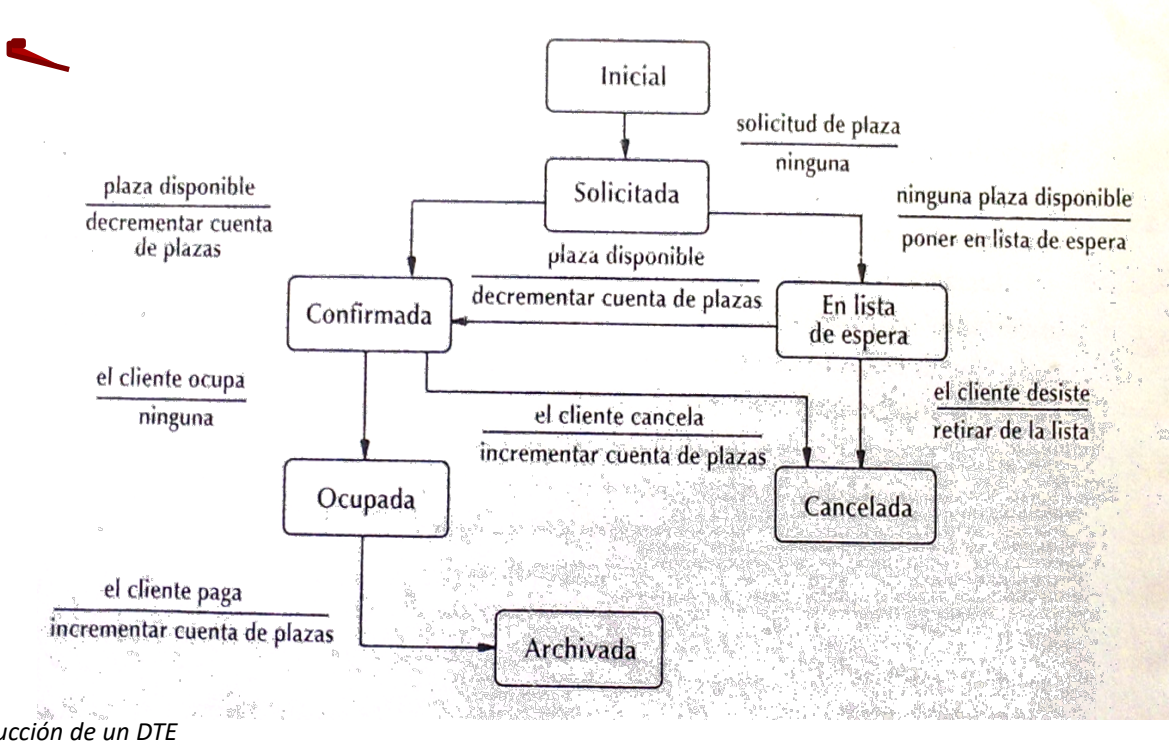
<u>Técnicas de especificación de requerimientos dinámicas</u>
→ **Máquinas de Estado Finito**: describen al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos).
f(Si, Cj) = Sk → al estar en el estado Si, la ocurrencia de una condición Cj, hace que el sistema cambie al estado Sk

-*Definición formal:* un autómata finito (AF) puede ser descripto como una 5-tupla (S, E, T, s, A) donde:
E 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 de estado en Gráfico de Persianas



- Máquinas de estado finito: Notación UML Diagrama de transición y estado (DTE)
- Máquinas de estado finito: Diagramas de Transición de Estado (DTE)



→ *Construcción de un DTE*

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

→ **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 simultanea 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, es decir, no son secuenciales.

Las tareas que ocurren en paralelo y se necesita alguna forma de controlar eventos para cambiar de estado ..estación de servicio.

-Los eventos se representan como transiciones (T)
-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

-*Definición formal*: una estructura de Red de Petri es una 4-upla C = (P, T, I, O), donde:
Lugares → **P**={P1,P2,...Pm}
Transiciones → **T**={T1, T2,...Tn}
Función de entrada → **I**: T→ P
Función de salida → **O**: P T →
Multigrafo (de un nodo puede partir más de un arco), bipartito, dirigido

- 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.

- 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.

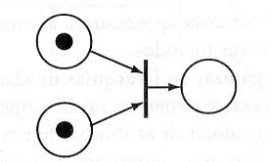
- La ejecución de una Red de Petri se realiza disparando transiciones habilitadas.

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

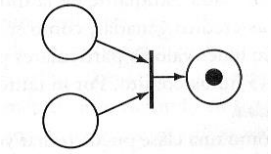
- Disparar una transición habilitada implica remover tokens de los lugares de entrada y distribuir tokens en los lugares de salida (teniendo en cuenta la cantidad de arcos que llegan y la cantidad de arcos que salen de la transición)

Transiciones

La transición está habilitada →



La transición NO está habilitada →



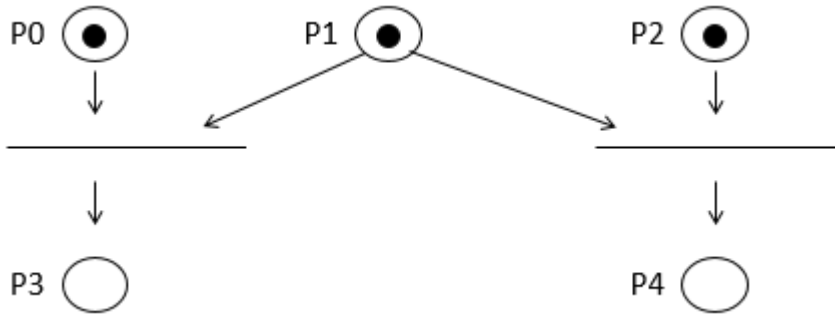
- La ocurrencia de los eventos (transiciones) depende del estado del sistema.
- Una condición puede ser V(con token) o F(sin token).

- La ocurrencia de un evento está sujeta a que se den ciertas condiciones (pre) y al ocurrir el evento causa que se hagan verdades las post-condiciones.
- Las RP son asincrónicas y el orden en que ocurren los eventos es uno de los permitidos: la ejecución no es determinística.
- Se acepta que el disparo de una transición es instantáneo.

Paralelismo

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



Condición de bloqueo