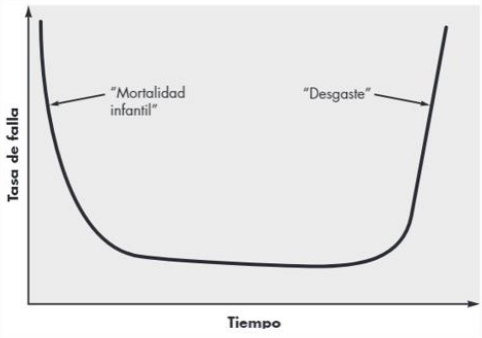
Software

Instrucciones, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación

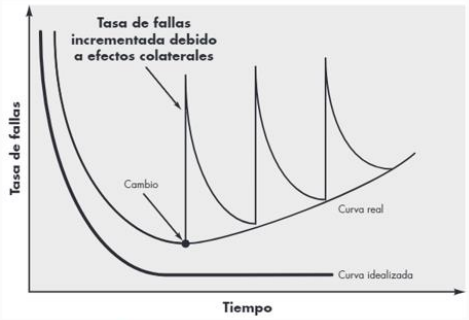
Características:

- Es un elemento lógico
- Se desarrolla, no se fabrica
- No se desgasta

El software no sigue una curva clásica de envejecimiento. El problema está en los cambios, no en el tiempo de operación.



Envejecimiento del hardware



Envejecimiento del software

Tipos de producto de software

- 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

Software Libre

Se refiere a libertad, tal como fue concebido por Richard Stallman en su definición

- Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre
- Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente
- Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos
- Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente

Ingenieria 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

Se aplican teorías, métodos y herramientas

No sólo comprende los procesos técnicos del desarrollo de software, sinó también se realizan actividades como la gestión de proyectos y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software

La IEEE la define como

1. El uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software
2. El estudio de técnicas relacionadas con 1

Se usan métodos sistemáticos cuantificables que cuantifican recursos, procesos y productos y es una precondición para la optimización de la productividad y calidad. La “metrificación” y el control estadístico de procesos son clave.

Se deben cumplir contratos en tiempo y costos.

Se ocupa de todo el ciclo de vida de un producto, desde la planificación y análisis de requerimientos hasta la estrategia para determinar cuándo y cómo debe ser retirado de servicio

Características de un ingeniero de software

El ingeniero debe dominar aspectos técnicos, aprender habilidades requeridas para entender el problema, diseñar la solución, desarrollarla, etc.

Pero además los aspectos humanos es lo que harán un ingeniero efectivo. Tener un sentido de responsabilidad individual, aguda conciencia de las necesidades del equipo, atención al detalle, entre otros.

Responsabilidad profesional y ética

- Confidencialidad: De empleados y clientes
- Competencia: No falsificar el nivel de competencia y aceptar responsabilidades fuera de su capacidad
- Derechos de la propiedad intelectual: Conocer las leyes vigentes sobre patentes y derechos de autor
- Uso inapropiado de las computadoras: No se debe utilizar las habilidades técnicas para utilizar de forma inapropiada otras computadoras

Tecnicas de comunicación

Al hablar de necesidades, estamos hablando de requerimientos

La comunicación es la base para la obtención de las necesidades del cliente y es la principal fuente de error

Requerimientos

Característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema

»Definición IEEE-Std-610

- ❑*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 como en 1 o 2.

Impacto de los errores en la etapa de requerimientos

- El software resultante puede no satisfacer a los usuarios
- Las interpretaciones múltiples de los requerimientos pueden causar desacuerdos entre clientes y desarrolladores
- Puede gastarse tiempo y dinero construyendo el sistema erróneo

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 HACER, o incluso cómo NO DEBE comportarse
- 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 nuestras elecciones en la construcción de una solución al problema
- Determinan el cómo se hace algo. Pueden ser normas a cumplir, el uso de recursos determinados, tiempos a cumplir, seguridad, usabilidad y otros aspectos del sistema más allá de su funcionalidad básica

Ejemplos de requerimientos no funcionales

- Requerimientos del producto
  - Especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad)
- Requerimientos organizacionales
  - 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

Otros ejemplos

Requerimientos de usabilidad:

- Estética
- Consistencia de interfaz de usuario
- Ayuda en línea
- Documentacion de usuario

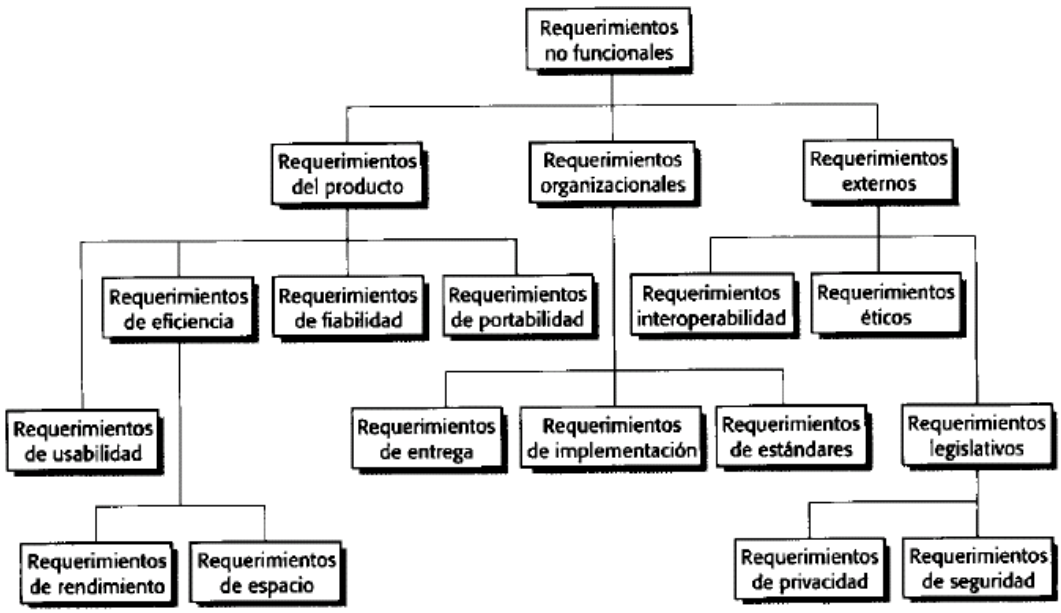
- Materiales de capacitación o entrenamiento

Requerimientos de eficiencia

- Eficiencia
- Disponibilidad
- Tiempo de respuesta
- Tiempo de recuperación
- Uso de recursos

Requerimientos de fiabilidad

- Frecuencia y severidad de fallas
- Facilidades de recuperación
- Posibilidades de predicción



Fuentes de requerimientos

- Documentación
- Stakeholders
- Especificaciones de sistemas similares

Stakeholder

Refiere a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente

Ejemplos:

- Usuarios finales
- Ingenieros
- Gerentes
- Expertos del dominio

Puntos de vista

Existen 3 tipos genéricos de puntos de vista

- Punto de vista de los Interactuadores: Representan a las personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema de algún modo
- Punto de vista indirecto: Representan a los stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo
- Punto de vista del dominio: Representan las características y restricciones del dominio que influyen en los requerimientos del sistema

Ejemplos de puntos de vista

- Proveedores de servicios al sistema
- Receptores de servicios del sistema
- Sistemas con lo que se debe interactuar
- Regulaciones y estándares a aplicar
- Las fuentes de requerimientos
- Personas que van a desarrollar, administrar y mantener el sistema
- Marketing y otros que generan requerimientos sobre las características del sistema

Elicitación de requerimientos

La elicitación (sonsacar) es el proceso de adquirir todo el conocimiento relevante para producir un modelo de los requerimientos de un dominio de problema

Es una actividad principalmente de carácter social, más que tecnológico

Por lo tanto, los problemas que se plantean son de naturaleza psicológica y social, más que técnicos

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

Requerimientos es lo mismo que requisitos

Problemas que se pueden presentar

- Comunicación
    - Dificultad para expresar claramente las necesidades
    - No ser conscientes de sus necesidades
    - No entender cómo la tecnología puede ayudar
    - Cultura y vocabulario diferentes
    - Medios de comunicación inadecuados
- Limitaciones cognitivas del desarrollador
    - No conocer el dominio del problema

    - Hacer suposiciones sobre el dominio del problema
    - Hacer simplificaciones excesivas
- Conducta humana
    - Conflictos y ambigüedades en los roles de los participantes
    - Pasividad de clientes, usuario o ingenieros de requisitos
    - Temor a que el nuevo sistema lo deje sin trabajo
- Técnicos
    - Complejidad del dominio del problema
    - Complejidad de los requisitos
    - Multiples fuentes de requisitos
    - Fuentes de información poco claras

Recopilacion de información

Metodos discretos

Son “menos perturbadores” que otras formas de averiguar los reqeurimientos

Se consideran insuficientes para recopilar información cuando se utilizan por sí 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

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

Muestreo de la documentación, los formularios y los datos existentes

Permiten conocer el historial que origina el proyecto

- Organigrama (identificar al propietario, usuarios claves)
- Memos, notas internas, minutas, registros contables
- Solicitudes de proyectos de sistemas de información anteriores

Documentos que describen la funcionalidad del negocio que está siendo analizada

- Declaracion de la misión y plan estratégico de la organización
- Objetivos formales del departamento en cuestión
- Políticas, restricciones, procedimientos operativos
- Bases de datos
- Sistemas en funcionamiento

Documentacion de sistemas anteriores

- Diagramas
- Diccionario 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 quien y cuando será observado
- Obtener el permiso de la persona y explicar el por qué será observado
- Mantener bajo perfil
- Tomar nota de lo observado
- Revistar las notas con la persona apropiada
- No interrumpir a la persona en su trabajo

Ventajas de la observación:

- Datos confiables
- El analista puede ver exactamente lo que se hace (tareas difíciles de explicar con palabras)
- Analisis de disposiciones físicas, transito, iluminación, ruido
- Económica en comparación con otras técnicas

Desventajas:

- La gente se siente incómoda siendo observada
- Algunas actividades del sistema pueden ser realizadas en horarios incómodos
- Las tareas están 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

Métodos interactivos

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

1. Cuestionarios
2. Entrevistas
3. Planeacion conjunta de requerimientos
4. Brainstorming

Planeación conjunta de requerimientos (JRP : Joint requirements planning)

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
- También se puede mencionar como JAD (Joint application design)

Ventajas:

- Reduce el tiempo de la etapa de los requerimientos
- Involucra activamente a los usuarios y la gerencia en el proyecto de desarrollo
- Desarrollo creativo
- Si se incorporan prototipos, los mismos ya confirman el diseño del sistema

Desventajas:

- Es difícil organizar los horarios de los involucrados
- Es complejo encontrar un grupo de participantes integrados y organizados

Participantes

- Usuarios y gerentes: Los usuarios comunican los requerimientos y los gerentes los aprueban
- Patrocinados: Miembro de la dirección con autoridad sobre los departamentos que participan, es el responsable del proyecto, toma las decisiones finales
- Equipos de TI: Escuchan y toman nota de los requerimientos
- Secretarios: Llevan el registro de la sesión y van publicando los resultados realizados
- Facilitador: Dirige las sesiones y tiene amplias habilidades de comunicación y negociación

Cómo se planean las sesiones de JRP

- Selección de una ubicación para las sesiones de JRP
- Selección de los participantes
- Preparar la agenda

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

Se realizan reuniones del equipo involucrado en la resolución del problema, conducidas por un director

Principios en los que se basa esta técnica

- 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 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

Encara la dificultad del usuario para transmitir

Ayuda a entender al usuario y al analista

Cuestionarios

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

- Recolectar hechos de un gran numero de personas
- Detectar un sentimiento generalizado
- Detectar problemas entre usuarios
- Cuantificar respuestas

Ventajas:

- Respuesta rápida
- Económicos
- Anónimos
- Estructurados de fácil análisis

Desventajas:

- Número bajo de respuestas
- No responde a todas las preguntas
- Preguntas rígidas
- No se puede realizar el análisis corporal
- No se pueden aclarar respuestas incompletas
- Dificiles de preparar

Tipos de preguntas

Abiertas: Son las que dejan abiertas todas las posibles opciones de respuesta

Cerradas: Limitan o cierran las opciones de respuestas disponibles

Tipo de información obtenida

- Actitud: Lo que las personas dicen que quieren
- Creencias: Lo que las personas creen que es verdad
- Comportamiento: Lo que realmente hacen
- Características: De las personas o cosas

Cuándo usar cuestionarios

- Las personas están dispersas geográficamente (diferentes oficinas, ciudades, etc.)
- Muchas personas involucradas
- Queremos obtener opiniones generales
- Queremos identificar problemas generales

Entrevistas

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 específico, que se basa en un formato de preguntas y respuestas en general

Conocer opiniones y sentimientos del entrevistado

Tipo de información obtenida

- Opiniones
- Objetivos
- Procedimientos informales
- Sentimientos

Ventajas:

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

Desventaja:

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

Tipos de entrevistas

Estructuradas (Cerradas)

- El encuestador tiene un conjunto específico de preguntas para hacérselas al entrevistado
- Se dirige al usuario sobre un requerimiento puntual
- No permite adquirir un amplio conocimiento del dominio

No estructuradas (Abiertas)

- El encuestador lleva a 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

¿Qué opinión tiene del sistema actual?

¿Cómo describe su trabajo?

Cerradas: Las repuestas son directas, cortas o de selección específica

¿Quién recibe este informe?

¿Cuántas personas utilizan el sistema?

Sondeo: Permite obtener más detalle sobre un tema puntual

¿Podría dar detalles/ejemplo sobre…?

Preguntas abiertas

Ventajas:

- Revelan nueva línea de preguntas
- Hacen más interesante la entrevista
- Permiten espontaneidad

Desventajas:

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

Preguntas cerradas

Ventajas:

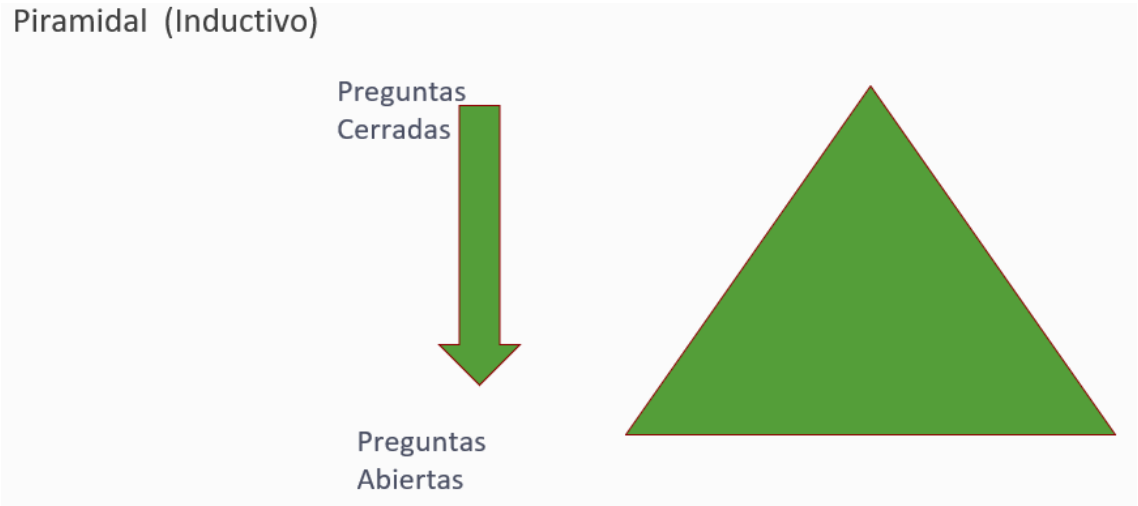
- Ahorran tiempo
- Se mantiene más fácil el control de la entrevista

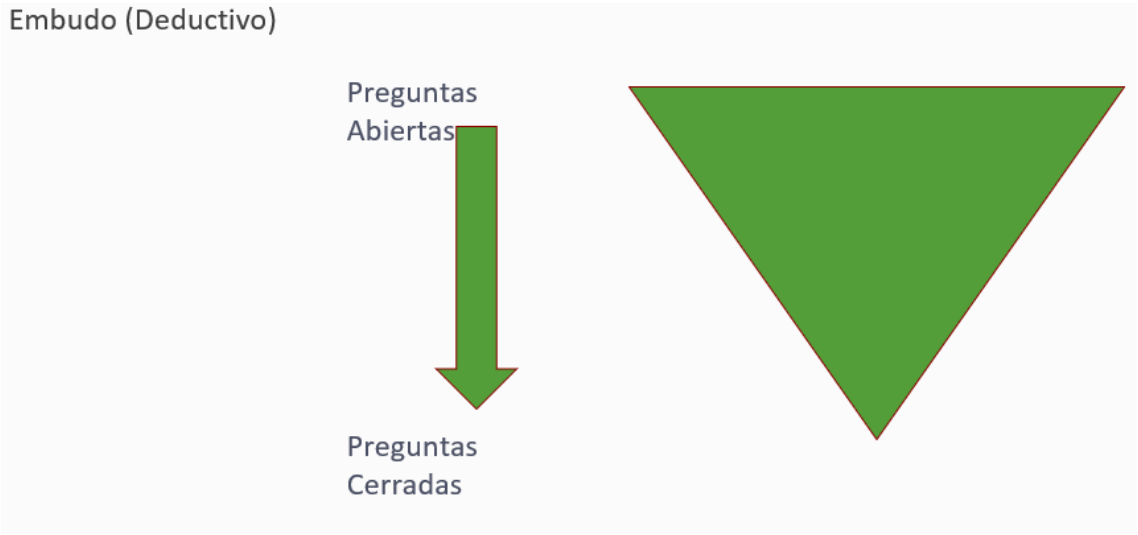
- Se consiguen datos relevantes

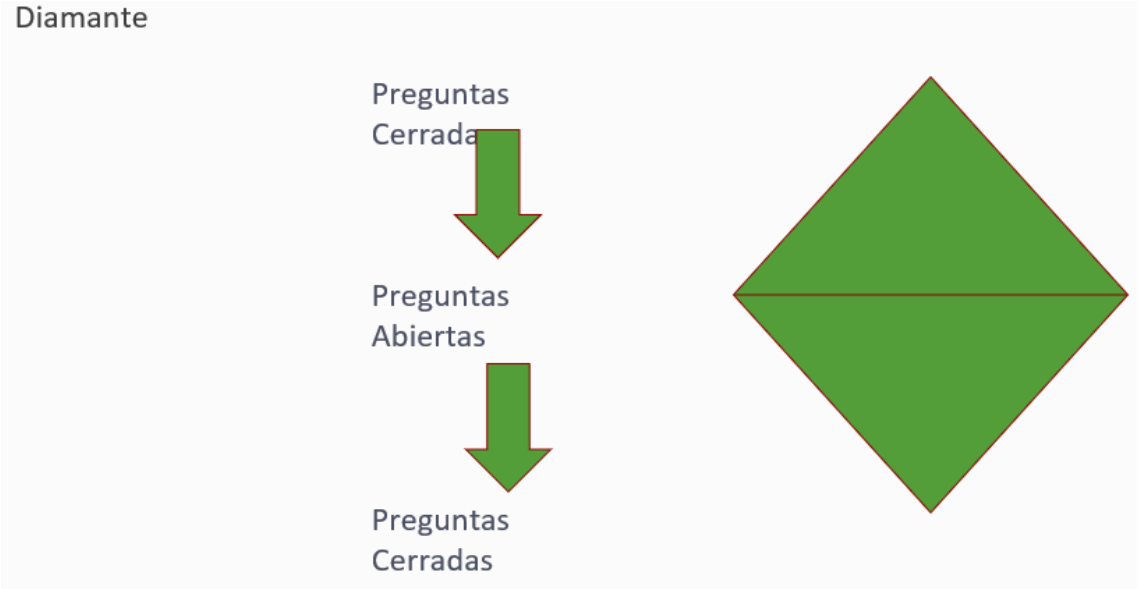
Desventajas:

- Pueden aburrir al encuestado
- No se obtienen detalles

Organización de una entrevista







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 del tipo de preguntas a usar y su estructura

Entrevistas (Whitten)

Cómo 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 (no llegar sin avisar)
  - Respetar el horario de trabajo
  - Establecer la duración de la entrevista
    - Cuanto mayor es el cargo del entrevistado menor tiempo se debe utilizar
  - Obtener el permiso del supervisor o jefe
  - 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 “Guion de Entrevista”
    - Se deben evitar preguntar sesgadas o con intención, amenazantes o críticas
    - Usar lenguaje claro y conciso
    - No incluir opinión como parte de la pregunta
    - Evitar realizar preguntas largas y complejas

Guión de una Entrevista (1)

Entrevista

La entrevista tiene una organización de preguntas de tipo embudo, que empieza con preguntas generales y más adelante entrar en preguntas más específicas.

Entrevistado: Gerente general de la empresa “Cuerpo Fit”
Fecha: 30/11/2022
Hora: 2:00 p. m.
Lugar: Oficina de la gerente en la ciudad de La Plata
Tema: Ampliación del software existente para realizar seguimientos de solicitudes de servicios y reclamos de los clientes

Tiempo asignado	Pregunta u objetivo del administrador	Respuesta del entrevistado
1 a 2 min.	**Objetivo** Comienza la entrevista: • Me presento. • Gracias señora gerente por su valioso tiempo. • Enunciar el propósito de la entrevista: obtener una comprensión más precisa del software utilizado por la cadena de locales y la ampliación que se va a desarrollar.	
4 min.	**Pregunta 1** ¿Cómo es el procedimiento que realiza actualmente para evaluar las solicitudes? **Objetivo:** comprender el actual funcionamiento del sistema de evaluación de solicitudes. **Seguimiento** ¿Piensa que este puede mejorarse? Si es así, ¿Cómo?	
2 min.	**Pregunta 2** ¿Cuáles son los criterios para la evaluación de las solicitudes? **Objetivo:** saber con qué criterios se realiza el proceso de la aprobación de las solicitudes.	
2 min.	**Pregunta 3** ¿Piensa que sería útil tener opciones para filtrar y ordenar las solicitudes pendientes? **Objetivo:** saber si son necesarias opciones de filtrado y ordenación, y si es así saber de qué forma filtrar y ordenar las solicitudes. **Seguimiento**	

Guión de una Entrevista (2)

3 min.	**Pregunta 5** ¿Con qué formato le gustaría recibir los reclamos de los clientes? **Objetivo:** saber de qué manera le resultaría mejor leer los reclamos de los clientes.	
2 min.	**Pregunta 6** ¿Considera que podrían clasificarse por categorías los reclamos de los clientes? **Objetivo:** saber de qué forma se deberían clasificar los reclamos de los clientes. **Seguimiento** Si es así: ¿en qué categorías?	
1 min.	**Pregunta 7** ¿Qué cantidad de solicitudes de nuevos insumos tiene que evaluar mensualmente? **Objetivo:** obtener una idea de la cantidad de solicitudes que manejaría el sistema.	
2 min.	**Pregunta 8** ¿Le queda algo más para añadir?	
1 min.	**Objetivo** Término de la entrevista: • Agradecerle a la gerente general por su colaboración y asegurarle que va a recibir un informe de lo que se obtuvo de la entrevista	
22 min.	Tiempo asignado para preguntas y objetivos	
6 min.	Tiempo asignado para preguntas de seguimiento y redirección	
28 min.	Tiempo asignado para la entrevista	
Comentarios generales y notas:		

Cómo conducir la entrevista

- Respete el horario y los tiempos definidos
- Si es en una sala de reunion llegue antes para asegurar las condiciones de la misma
- Inicie la entrevista saludando, presentándose y agradeciendo la atención
- Mencione el propósito de la misma y la duración
- Escuche con atención y observe al entrevistado, tome nota de las respuestas verbales y no verbales
- Concluya la entrevista expresando su agradecimiento
- Haga una breve conclusión de la entrevista para ganar la confianza del entrevistado

Se debe hacer

- Vestirse adecuadamente
- Ser cortes
- Escuchar cuidadosamente
- Mantener el control
- Observar los gestos
- Ser paciente
- Mantener el entrevistado en calma
- Mantener el autocontrol
- Terminar a tiempo

Se debe evitar

- Suponer que una respuesta no lleva a ningún lado
- Revelar pistas
- Usar jerga
- Revelar sesgos personales
- Hablar en lugar de escuchar
- Suponer cualquier cosa acerca del tema o del entrevistado
- Uso de grabadores (Señal de debilidad de escuchar)

Seguimiento de la entrevista

Enviar al entrevistado un resumen de la entrevista, permitiendo aclarar cualquier cosa que no se haya entendido durante la entrevista

Cómo escuchar

Saber escuchar es la parte más importante del proceso de una entrevista, se debe diferenciar entre oír y escuchar

- Llegue con actitud positiva: Mejora el canal de comunicación
- Haga que la otra persona se tranquilice: Romper el hielo con cuestiones cotidianas
- Haga ver que está escuchando lo que dice: Mantener el contacto visual, asentir con la cabeza, emitir comentarios
- Haga preguntas sobre lo que dice: El entrevistado siente que le presta atención y puede ampliar su respuesta
- No haga suposiciones: Escuche todo lo que el entrevistado tiene que explicar
- Tome nota: El entrevistado percibe que está siendo escuchado

El lenguaje corporal

Información no verbal que comunicamos

La mayor parte de la información se expresa a través de las expresiones corporales

Las más importantes son

- Expresiones faciales
- Contacto visual
- Postura

Proceso de software

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

Ingenieria de requerimientos

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

Es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema

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

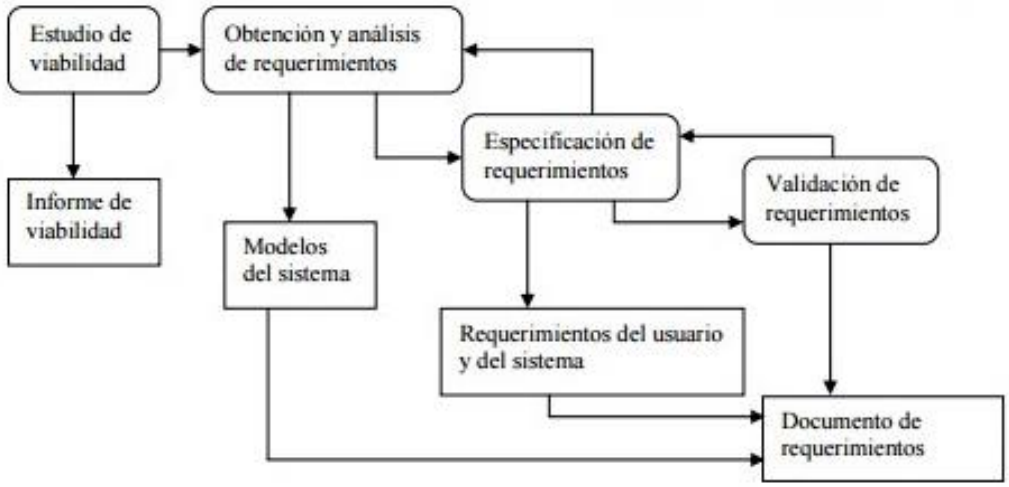
Es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos entre los clientes y el equipo del proyecto

Tenes todo documentado nos permite que todos trabajen en la misma solución y que estén todos de acuerdo

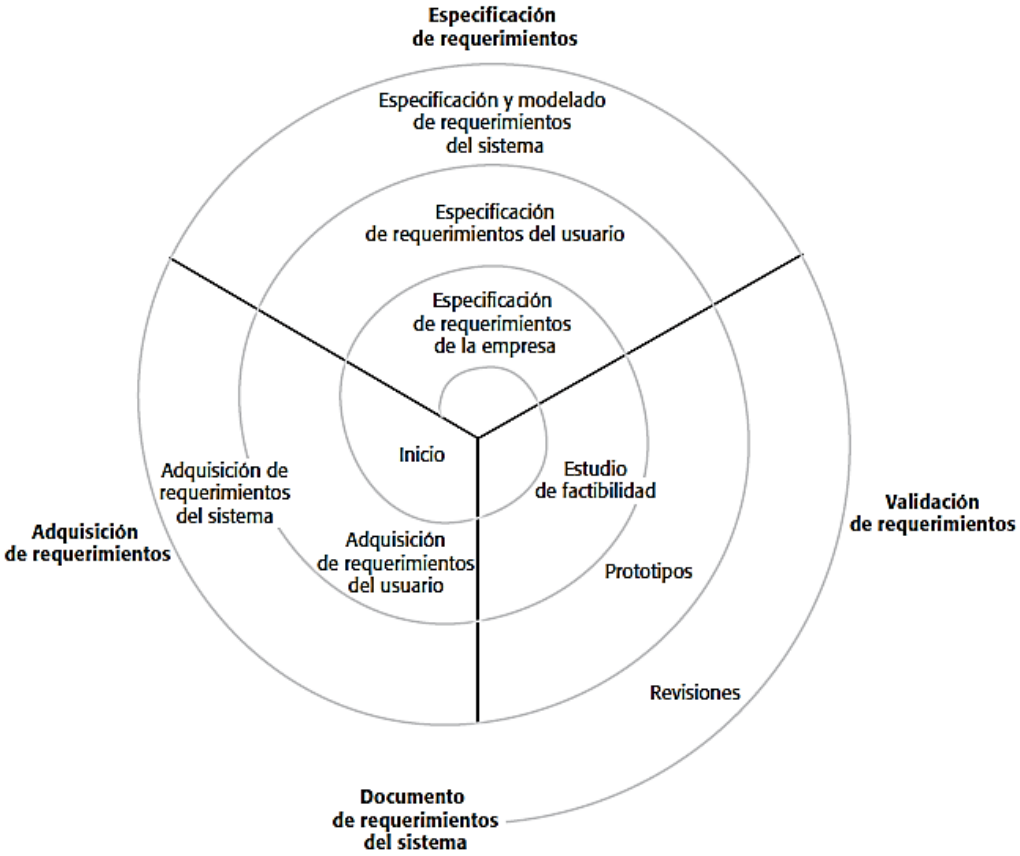
Importancia

- 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

Proceso secuencial



Proceso iterativo (así se hace en la práctica)



Estudio de viabilidad

A partir de una 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:

- ¿Se puede implementar con la tecnología actual?
- ¿Se puede implementar con las restricciones de costo y tiempo?
- ¿Contribuye a los objetivos generales de la organización?
- ¿El sistema puede integrarse a otros que existen en la organización?

Una vez que se recopiló toda la información necesaria para contestar las preguntas, se debería hablar con las fuentes de información para responder nuevas preguntas y luego se redacta el informe que recomienda si se debe continuar o no el desarrollo

Especificación de requerimientos

Propiedades de los requerimientos

- Necesario: Su omisión provoca una deficiencia
- Conciso: Facil de leer y entender
- Completo: No necesita ampliarse
- Consistente: No contradicatorio con otro
- No ambiguo: Tiene una sola implementación
- Verificable: Puede testearse a través de inspecciones, pruebas, etc.

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

Documentos

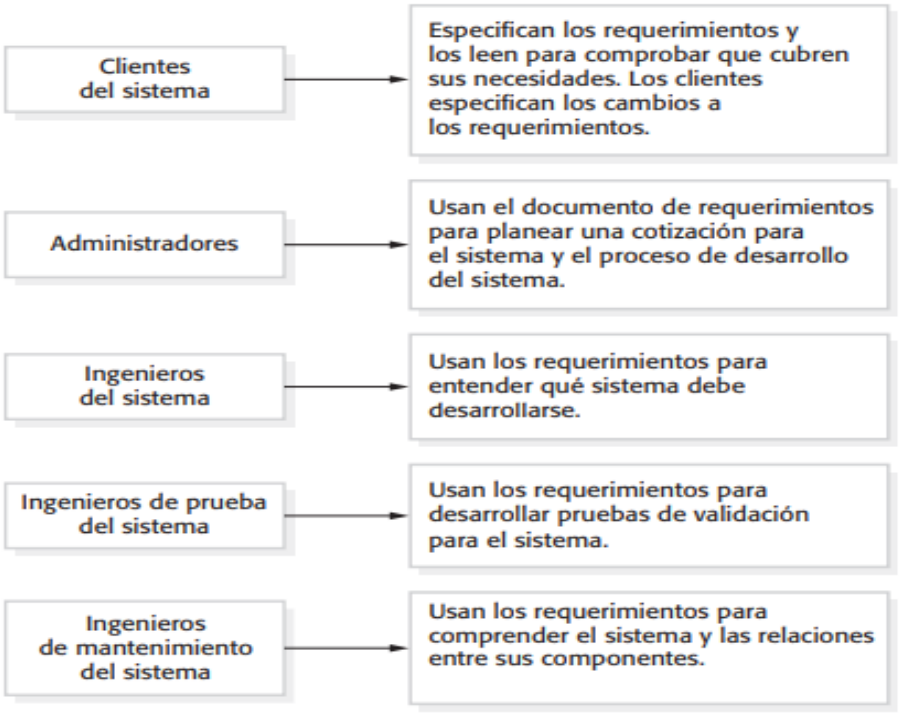
- De definición de requerimientos: Listado completo de todas las cosas que el cliente espera que haga el sistema propuesto
- De especificación de requerimientos: Definición en términos técnicos
- De especificación de requerimientos de Software IEEE Std. 830-1998 (SRS): Brindar una colección de buenas prácticas para escribir especificaciones de requerimientos de software (SRS). Se describen los contenidos y cualidades de una buena especificación de requerimientos.

SRS = Software requirement specification

Aspectos básicos de una especificación de requerimientos

- Funcionalidad: Qué debe hacer el software
- Interfaces externas: Cómo interactuará con el medio externo (gente, hardware, otro software)
- Rendimiento: velocidad, disponibilidad
- Atributos: Portabilidad, seguridad, eficiencia
- Restricciones de diseño: Estándares requeridos, lenguaje, limite de recursos

Usuarios de un documento de requerimientos



Validación de requerimientos

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

Definiciones de la IEEE´

- Validación: Al final del desarrollo evaluar el software para asegurar que el mismo cumple los requerimientos
- Verificacion: El software cumple los requerimientos correctamente

Sobre estas definiciones

La validación sólo se puede hacer con la activa participación del usuario

Validación: Hacer el software correcto

Verificación: Hacer el software correctamente

Es suficiente validar después del desarrollo del software?

La evidencia estadística dice que no

Cuánto más tarde se detecta, más cuesta corregir un error

Bola de nieve de defectos

Validar en la fase de especificación de requerimientos puede ayudar a evitar costosas correcciones después 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

Qué comprende la validación de requerimientos

- Verificaciones de 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 un conjunto de pruebas)

Técnicas de validación

Pueden ser manuales o automatizadas

- Revisiones de requerimientos
    - Informales: Los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea 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
- Construccion de prototipos
- Generacion de casos de prueba

Técnicas de especificación de requerimientos de software

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.

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

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 decision, diagramas de transición de estados, diagramas de persianas, redes de Petri, historias de usuario

Historias de usuario

Descripción corta y simple de un requerimiento de un sistema, que se escribe en lenguaje común del usuario y desde su perspectiva. Se escribe en una o dos frases

Son utilizadas en las metodologías de desarrollo ágiles para la especificación de requerimientos

Se acompañan de las discusiones con los usuarios y las pruebas de validación

Conceptos

Debe ser limitada como para escribirse en una nota adhesiva

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

Permiten responder rápidamente a los requisitos 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 sitúe 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

Forma de redactarlas

El estilo puede ser libre, pero debe responder a 3 preguntas

- ¿Quién se beneficia?
- ¿Qué se quiere?
- ¿Cuál es el benficio?

“Como (rol) quiero (algo) para (beneficio)”

Características

- Independientes unas de otras: De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes
- Negociable: La historia en sí misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito 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 es la estimación que tomará 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 usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse de manera que pueda ser verificada en cada entrega del proyecto

Historias de Usuario - Criterios de aceptación

Es el criterio por el cual se define si una historia de usuario fue desarrollada según la expectativa del Product Manager/Owner (como representante de los criterios del cliente) y si se puede dar como hecha

Deben ser definidos durante la etapa inicial antes de la codificación, acompañan a la historia de usuario porque complementan la misma y ayudan al equipo de desarrollo a entender mejor cómo se espera que el producto se comporte

Estos criterios son utilizados para expresar el resultado de las conversaciones del cliente con el desarrollador. El cliente debería ser quien las escriba más que el desarrollador

Representan el inicio de la definición del "cómo". No están 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 número de escenario para identificar al criterio, asociado a la historia de usuario en cuestión

Beneficios (Ventajas)

- Al ser muy corta, ésta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas)
- Necesitan poco mantenimiento
- Mantienen una relación cercana con el 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
- Fomentan aplazar los detalles no imprescindibles
- Proporcionan la documentación necesaria fomentando a la vez el debate

Limitaciones (Desventajas)

- 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

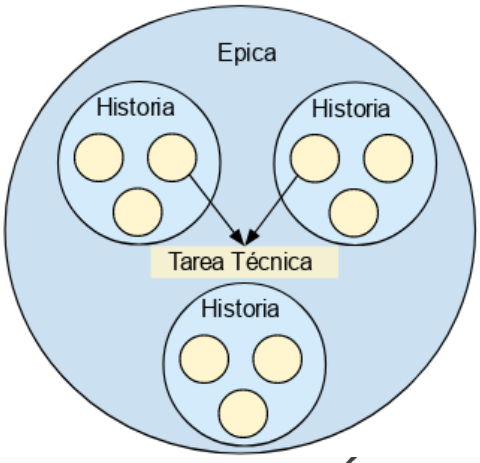
Historias de Usuario - Épicas

Se denomina épica a un conjunto de historias de usuario 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 la funcionalidad



Ejemplo de Épicas

Épica:

Como Vicepresidente 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 período de tiempo en el cual realizaré la revisión de las 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

Características de 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
- Sirve para estructuras los temas e iniciativas (objetivos)
- Sirven también para dar flexibilidad y agilidad al proyecto

Casos de uso

Definicion

Proceso de modelado de las “funcionalidades” del sistema en término de los eventos que interactúan entre los usuarios y el sistema

Tiene sus orígenes en el modelado orientado a objetos, 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 casos de uso facilita y alienta la participación de los usuarios

Propósitos principales

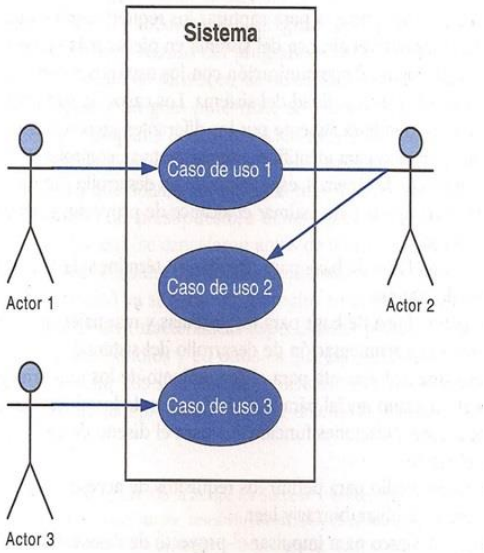
- Describir los requerimientos funcionales del sistema
- Identificar los roles que interactúan con el sistema

Beneficios

- 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

[illegible]	[illegible]
[illegible]	1.0
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	Paso / [illegible]
	1 / [illegible]
	2 / [illegible]
	3 / [illegible]
	4 / [illegible]
	5 / [illegible]
[illegible]	Paso / [illegible]
	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

Diagrama

Caso de uso

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

Un actor inicia una actividad (CU) en el sistema

Representa un papel desempeñado por un usuario que interactúa (rol)

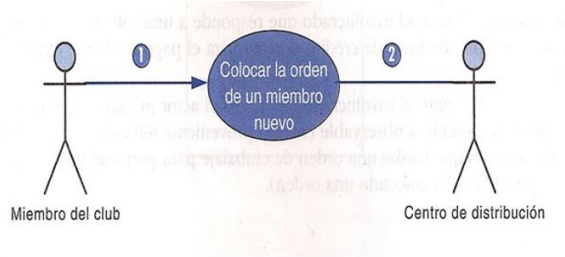
Puede ser una persona, sistema externa o dispositivo externo que dispare un evento (sensor, reloj)

Relaciones

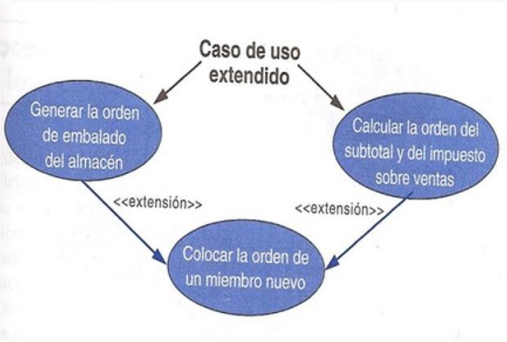
La asociación se representa con una flecha para indicar que el actor dispara el caso de uso. En caso de ser una línea quiere decir que no necesariamente el actor lo va a disparar pero que igual está involucrado

Asociaciones

Relación entre un actor y un CU en el que interactúan entre sí.



(1) El Actor inicia el caso de uso
(2) El caso de uso interacciona con actor



Extensiones

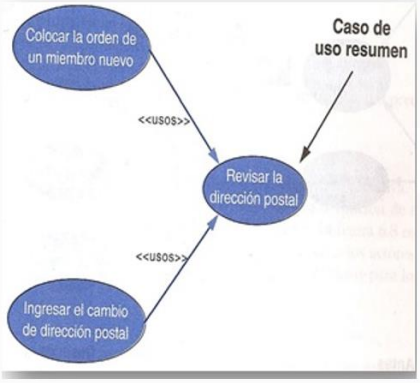
Un CU extiende la funcionalidad de otro CU.

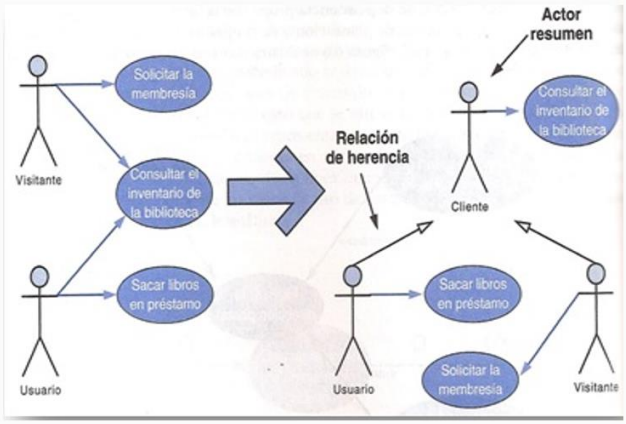
Un CU puede tener muchos CU extensiones.

Los CU extensiones sólo son iniciados por un CU.

Uso o inclusión

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





Herencia

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

Escenarios

Describen la interacción del escenario y eventos alternativos

Partes del escenario

- Nombre: Debe comenzar con un verbo y representar la meta
- Descripción: Corta y precisa del propósito
- Actores: Actor principal que se benefician del CU y que son parte del mismo
- Precondiciones: restricciones del estado del sistema antes de la ejecución del CU
- Secuencia normal: Sin errores ni condiciones, realizada por los actores y el sistema. Debe representar la interacción entre ambos
- Curso alterno: Describe el comportamiento si ocurre una excepción o variación del curso típico
- Postcondición: Restricción del estado del sistema después de la finalización exitosa del CU

Proceso de modelado de los CU

Pasos

- Identificar a los actores
    - Para buscar actores potenciales se puede usar documentación o manuales existentes, minutas de reunión, documentos de requerimientos
    - Se debe responder a
        - Quien o qué proporciona las entradas al sistema?
        - Quien o qué recibe las salidas del sistema?
        - Se requieren interfaces con otros sistemas?
        - Quien mantendrá la información en el sistema?
        - Existen eventos que son originados automáticamente en un instante predeterminado?
    - Deberan nombrarse con un sustantivo o frase sustantiva
- Identificar los CU para los requerimientos
    - Se debe responder a
        - Cuales son las principales tareas del actor?
        - Que información necesita el actor del sistema?
        - Que información proporciona el actor al sistema?
        - Necesita el sistema informar el actor de eventos o cambios ocurridos?
        - Necesita el actor informar al sistema de eventos o cambios ocurridos?
- Construir el diagrama
- Realizar los escenarios

Características importantes

- Un CU debe 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 precondiciones no deben representarse en los cursos alternativos, ya que, al ser una precondición, no va a ocurrir
- Los “uses” deben ser accedidos por lo menos desde dos CU

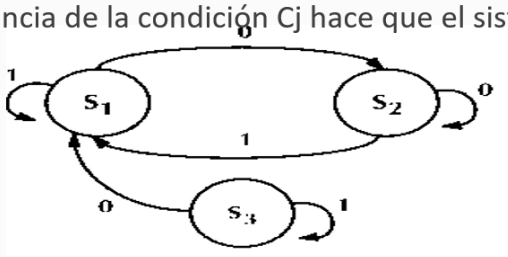
Diagrama de transición de estados

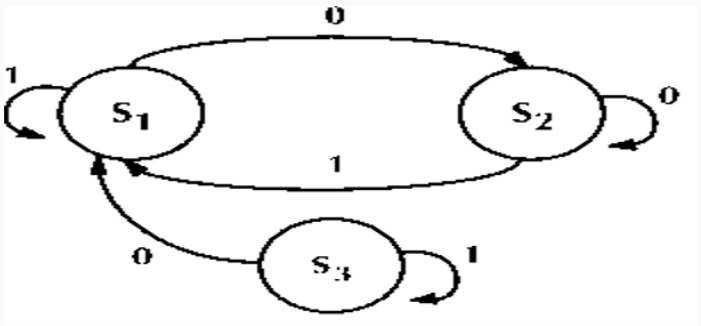
Maquinas de estado finito

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.





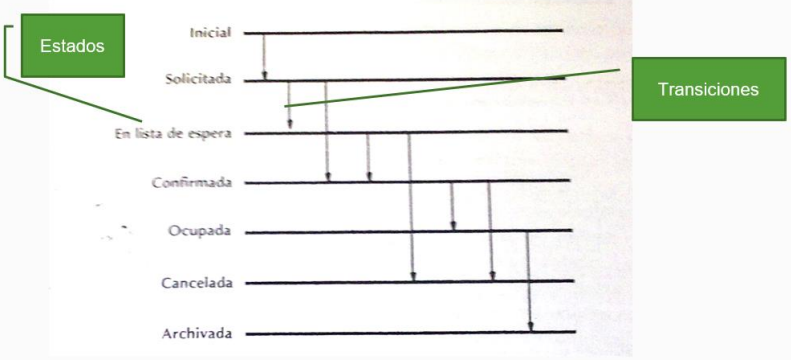
f(S1, 0) = S2
f(S1, 1) = S1
f(S2, 0) = S2
f(S2, 1) = S1
f(S3, 0) = S1
f(S3, 1) = S3

Definición formal

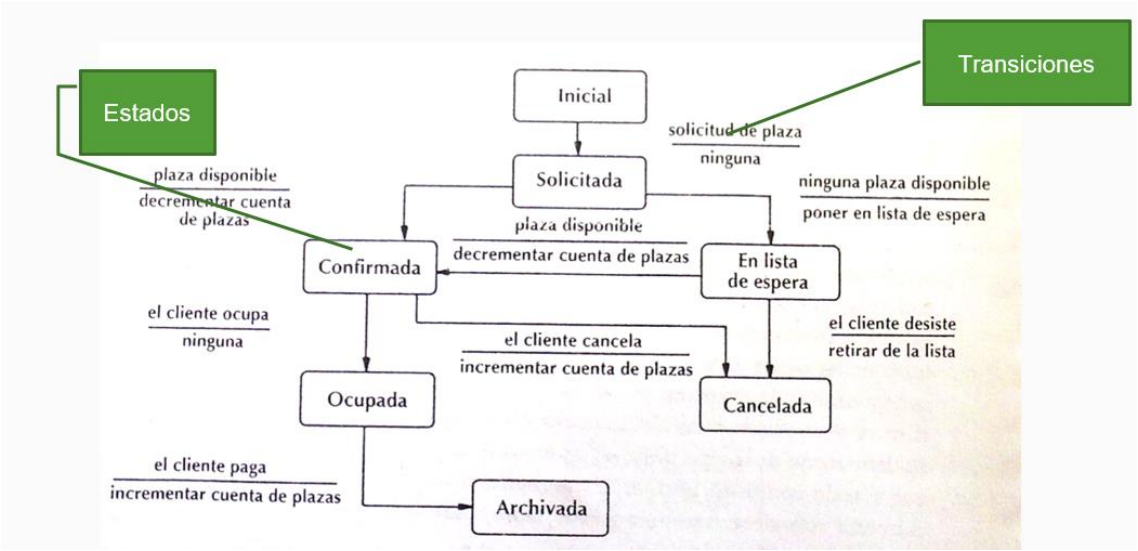
Formalmente, un autómata finito (AF) puede ser descrito como una 5-tupla (S,Σ,T,s,A) donde:

- 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.

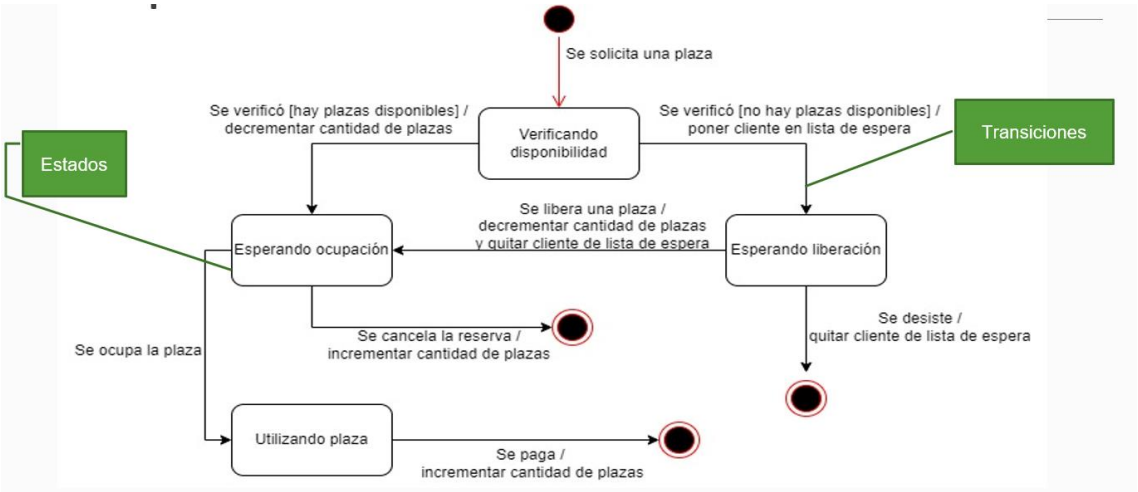
Gráfico de persiana



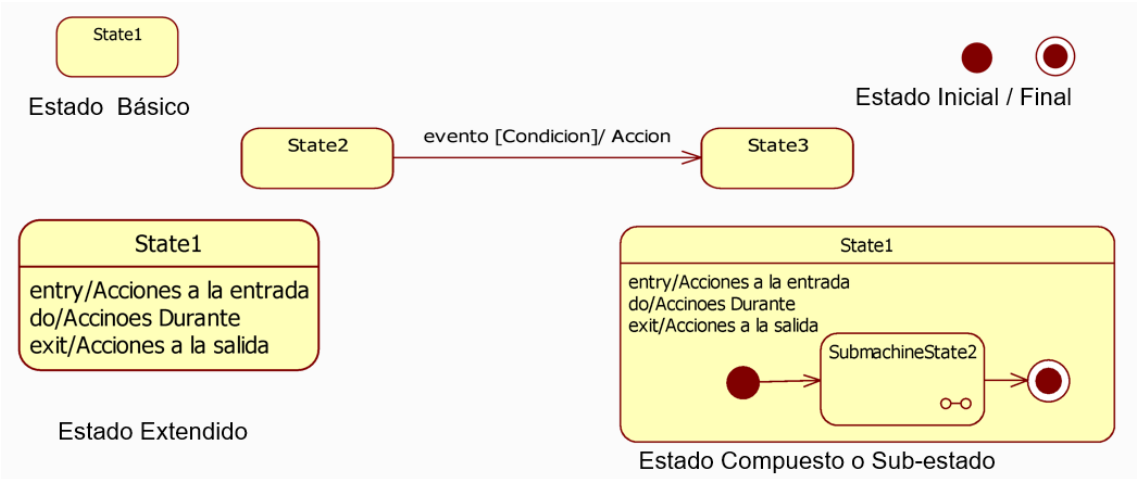
Representación en máquina de estado finito



Representación en DTE



Notación UML DTE



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

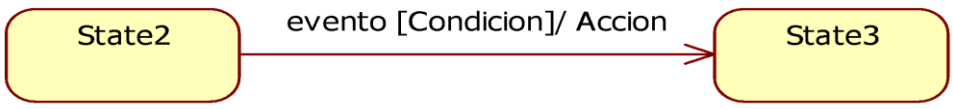
Diferencia entre evento y acción

El evento es una ocurrencia que puede causar la transición de un estado a otro

Una acción es una operación atómica que no se puede interrumpir por un evento y que se ejecuta hasta su finalización

Transición

Se producen como consecuencia de eventos. Pueden tener o no un procesamiento asociado



Evento: obligatorio

Condicion: opcional, depende del problema, puede haber transiciones sin condiciones

Accion: opcional, puede haber transiciones sin acciones

Construcción de un DTE

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

Redes de Petri

Utilizadas para especificar sistemas en 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 ejecutadas en un orden impredecible

No son secuenciales

Componentes

- Eventos/acciones: Se representan como una transición (T)
- Estados/condiciones Se representan como lugares o sitios (P)

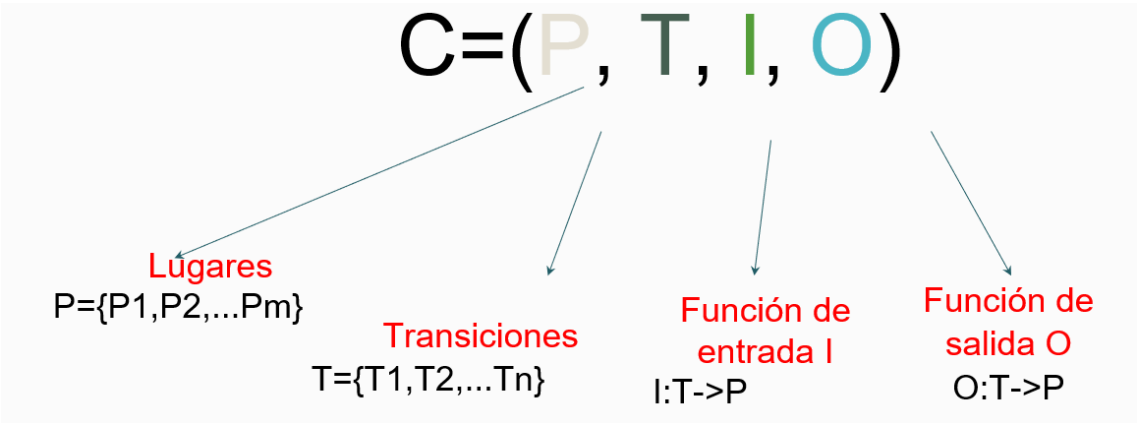
»Caso más simple:
f(EstadoA, Evento) -> EstadoS

»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



Es un multigrafo, bipartito, dirigido

Es un grafo en el cual desde un nodo puede partir más de una arista o arco de manera que su conjunto de vértices puede particionarse en dos conjuntos independientes donde las aristas tienen un sentido definido

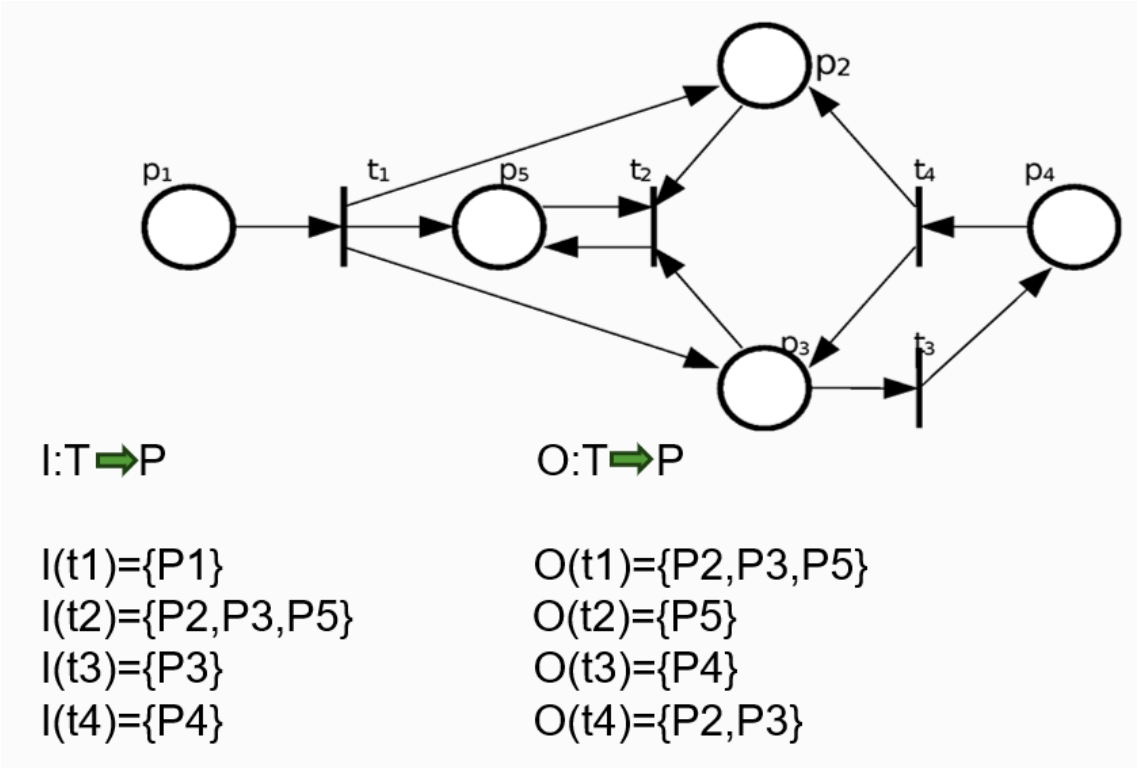
Diagrama

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

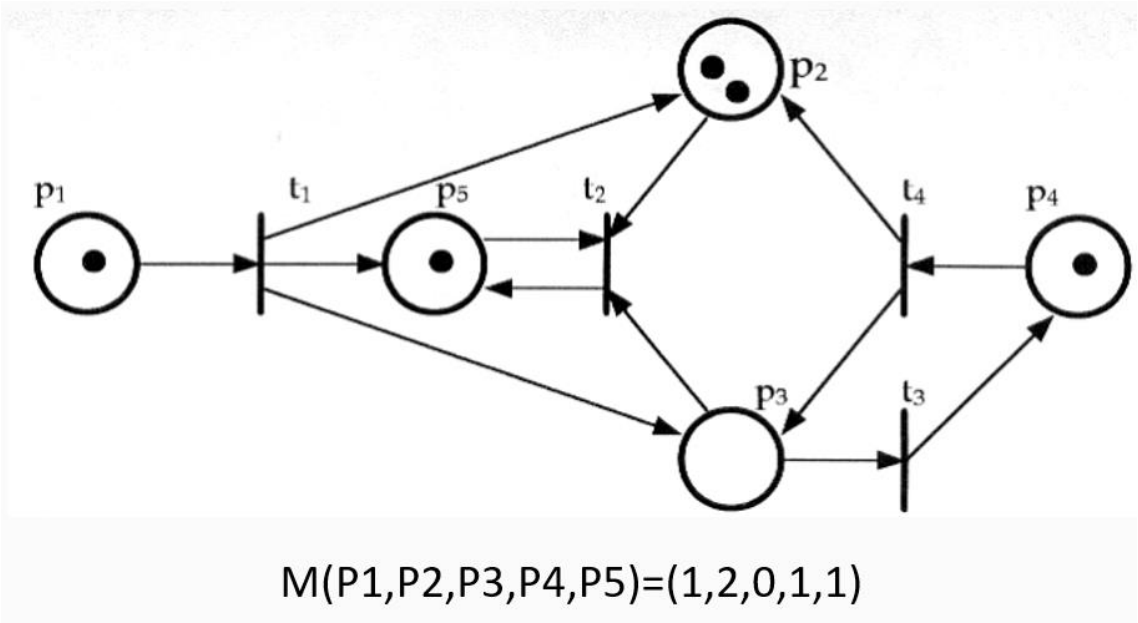
A los lugares se les asignan tokens que se representan mediante un número o puntos dentro del sitio. Esta asignación de tokens a lugares constituye la marcación

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

Funciones de entrada y salida



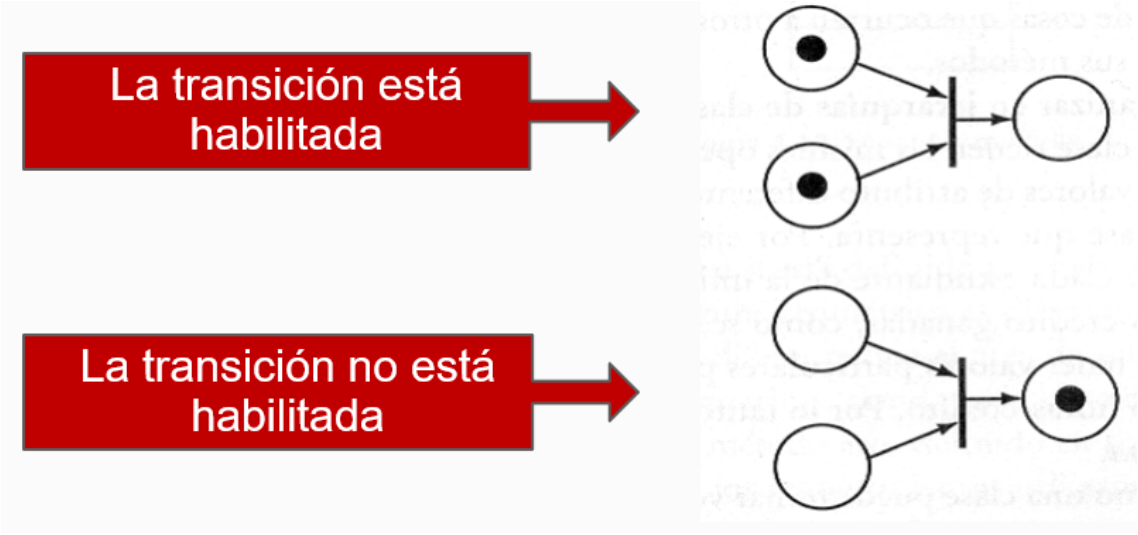
Marcación inicial



Características

- El conjunto de tokens asociados a cada estado sirve para manejar la coordinación de eventos y estados
- Una vez que ocurre un evento, un token puede “viajar” de un estado 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 tokens
- La ejecución de una red de Petri se realiza disparando las transiciones habilitadas

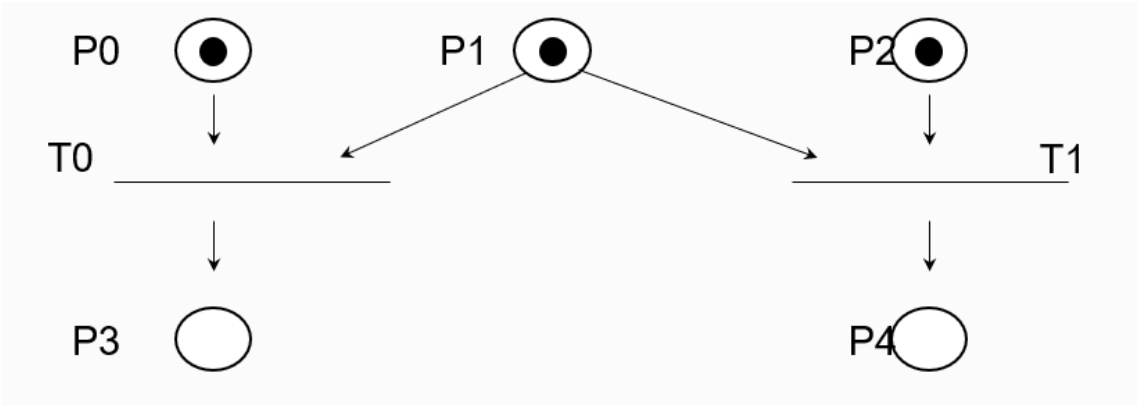
- Una transición está habilitada cuando cada lugar de entrada tiene al menos tantos tokens como arcos hacia la transición
- Disparar una transición habilitada implica remover tokens de los lugares de entrada y distribuir tokens en los lugares de salida (teniendo en cuenta la cantidad de arcos que llegan y los que salen de la transición)



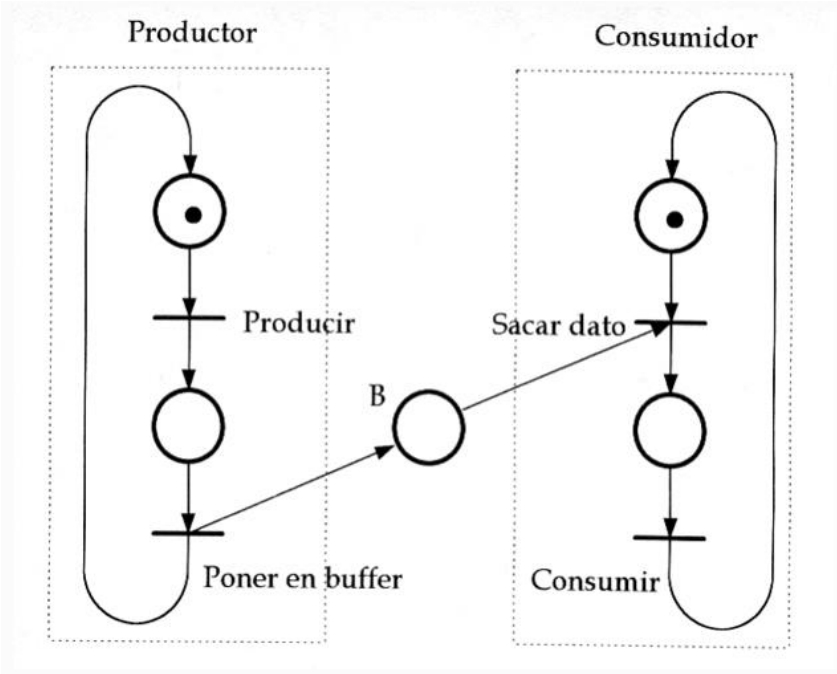
- La ocurrencia de los eventos (transiciones) depende del estado del sistema
- Una condición puede ser V o F (con o sin token)
- La ocurrencia de un evento está sujeta a que se den ciertas precondiciones y al ocurrir el evento causa que se hagan verdaderas las postcondiciones
- Las RP son asíncronas y el orden en que ocurren los eventos es uno de los permitidos (Su ejecución es no determinística)
- Se acepta que el disparo de una transición sea instantáneo

Conceptos

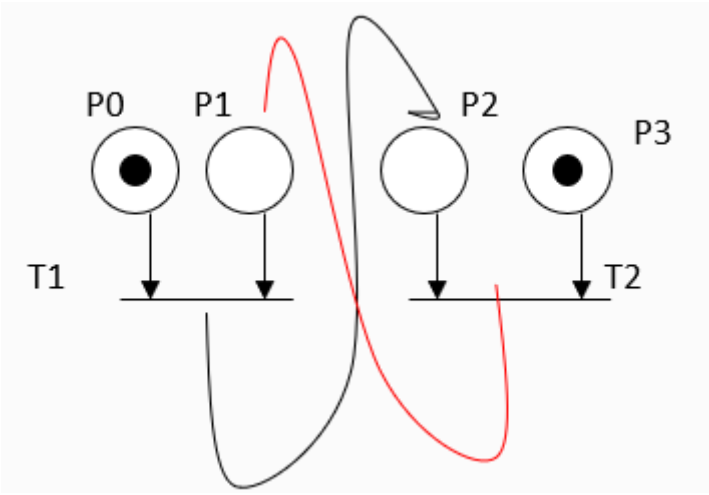
Exclusión mutua



Productor – Consumidor



Condición de bloqueo (Deadlock)



Características

- 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

Un evento discreto es un acontecimiento que ocurre en un punto específico del tiempo (transiciones)

Tablas de decisión

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

Las tablas de decisión se usan para representar la descripción de situaciones, se representan las distintas alternativas, estados de la naturaleza y las consecuencias, proporcionan una descripción completa, correcta, clara y concisa de una situación que se resuelve por una decisión tomada en un momento específico del tiempo

Describe al 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
- Acciones a ser tomadas como un resultado

Componentes

- Condiciones simples y acciones simples
- Las condiciones toman sólo valores V o F
- Hay 2^N reglas donde N es la cantidad de condiciones

	REGLA1	REGLA2	.....
COND1			
COND2			
.....			
ACCION1			
ACCION2			
.....			

¿Cómo se llena la tabla?

A partir de un enunciado se debe

1. Identificar las condiciones y acciones
2. Completar la tabla teniendo en cuenta
    a. Si hay condiciones que son opuestas, debe colocarse sólo una de ellas porque por la negativa se obtendrá la otra (si son n condiciones excluyentes, se colocan n-1 en la tabla)
3. Se construyen las reglas

Tipos de 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

Redundancia y Contradicción

	Reglas						
C1	V	V	. .	..	..	F	F
C2	V	V	.	.	.	V	V
C3	V	F	.	.	..	F	F
A1			.	.	..	X	X
A2	X				.		
A3		X	.	.		X	X

Redundante

	Reglas						
C1	V	V	. .	. .	..	F	F
C2	V	V	.	.	.	V	V
C3	V	F	.	.	..	F	F
A1			.	.	..		X
A2	X				.	X	
A3		X	.	.		X	

Contradictoria

Reducción de complejidad

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

El guión (-) significa que la condición puede ser V o F, y que aún así se realizará la acción

	Reglas			
Es cliente	V	V	F	F
Hay stock	V	F	V	F
Imprime mensaje de aviso			X	X
Se remite	X			
No se remite		X	X	X

Reglas		
V	V	F
V	F	_
		X
X		
	X	X

Recordar

Para construir tablas de decisión el analista necesita determinar el tamaño máximo de la tabla; eliminar cualquier situación imposible, inconsistencia o redundancia y simplificar la tabla lo más que pueda

Es esencial que verifique la integridad y precisión de sus tablas de decisión.

Problemas principales al desarrollar tablas de decisión

- Que esté incompleta
- Que existan situaciones imposibles
- Contradicciones
- Redundancia

Análisis estructurado

Actividad de construcción de modelos.

Mediante una notación creamos modelos que representan los datos y flujo de información.

Se realizan tres tipos de modelados: De datos (por ej usando DER), de funciones del sistema (por ej usando DFD) y del comportamiento del sistema (por ej usando DTE)

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

Modelado funcional y flujo de la información

Diagrama de flujo de datos

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 tambien 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

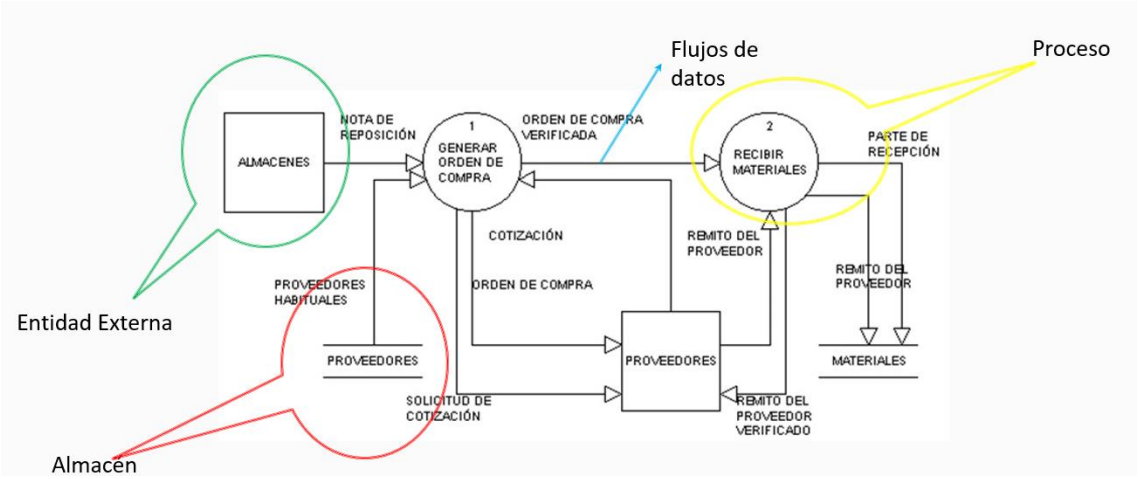
Se utiliza un rectángulo para representar una ***entidad externa,*** esto es, un elemento del sistema (por ejemplo, un elemento hardware, una persona, otro programa) u otro sistema que produce información para ser transformada por el software, o recibe información producida por el software.

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 “conducto” para uno o más ***elementos de datos*** (objetos de dato).

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

Ejemplo



Desarrollo de DFDs

Se debe visualizar desde una perspectiva jerárquica de arriba hacia abajo

Pasos:

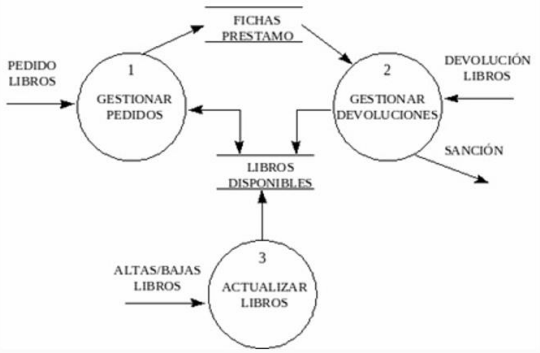
1. Redactar la lista de actividades (eventos) de la organización para determinar
    a. Entidades externas
    b. Flujos de datos
    c. Procesos
    d. Almacenes de datos
2. Crear un diagrama de contexto que muestre las entidades externas y los flujos de datos desde y hacia el sistema
3. Dibujar el diagrama 0 (siguiente nivel), con procesos generales y los almacenes correspondientes
4. Dibujar un diagrama hijo por cada uno de los procesos del diagrama 0

Diagrama de contexto

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

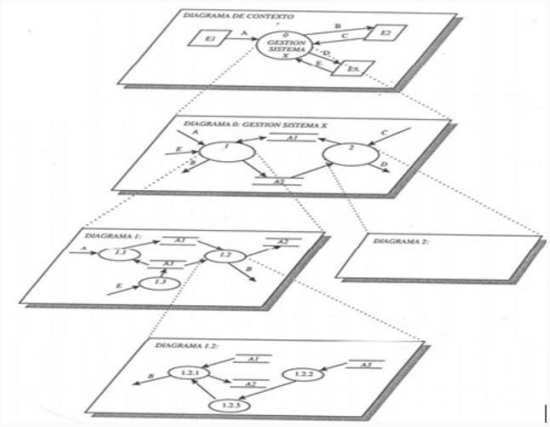
Nivel 0



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.

Nivelación de un DFD



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.

Conceptos

- Flecha: Representa uno o más elementos de datos
- Entidad externa: Es un elemento del sistema (hardware, persona, programa, etc) u otro sistema que produce información para ser transformada por el software o recibe información producida por el software
- Diagrama de flujo de datos: Herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por “conductos” y almacenamientos de datos
- Proceso: Transformación que es aplicada a los datos (o al control) y los modifica
- Almacen: Representa la información en reposo utilizada por el sistema independientemente del sistema de gestión de datos. Contiene la información necesaria para la ejecución del proceso
- Diccionario de datos: Es una definición sin ambigüedad de los datos y elementos del sistema

Modelos de proceso

Se puede pensar un proceso como un “conjunto ordenado de tareas”

Proceso de software

Conjunto de actividades y resultados asociados que producen un producto de software

Actividades fundamentales de los procesos

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 de 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 también incluye inspecciones y revisiones en distintas etapas

Evolución del software

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

Modelo de proceso de software

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

Norma ISO 12207-1 [ISO/IEC, 1995]

Lo define como un 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

Características de un modelo de proceso

- 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

Definiciones

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

Representación abstracta de un proceso del software

Terminos equivalentes

Modelo de proceso = Paradigma de software = Ciclo de vida del software

Tipos de modelos

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, descriptivos y prescriptivos, deberían ser iguales)

Modelos tradicionales

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

Ejemplos:

- Clasico, lineal 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

Ejemplos:

- En espiral (Por fases)
- Evolutivo (Por fases)
- Incremental

Procesos ágiles

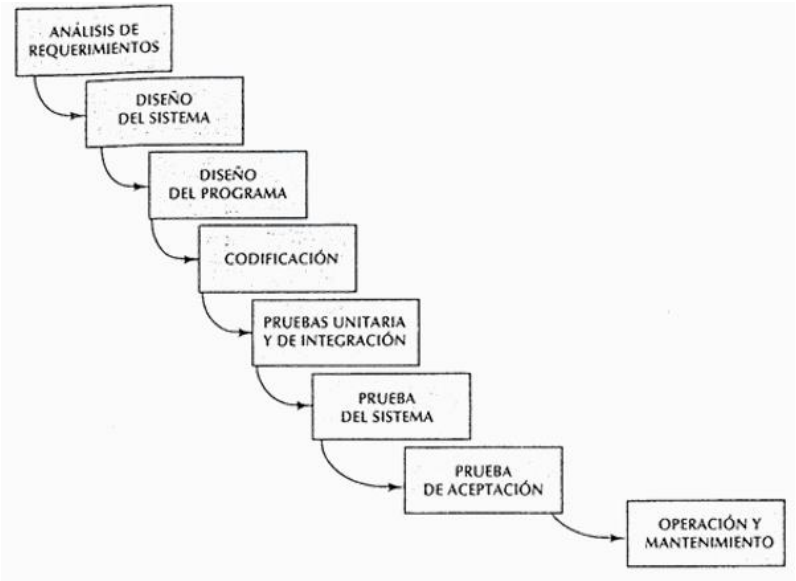
Modelo en cascada

Las etapas se representan cayendo en cascada

Cada etapa de desarrollo se debe completar antes que comience la siguiente

Util para diagramar lo que se necesita hacer

Su simplicidad hace que sea fácil explicarlo a los clientes

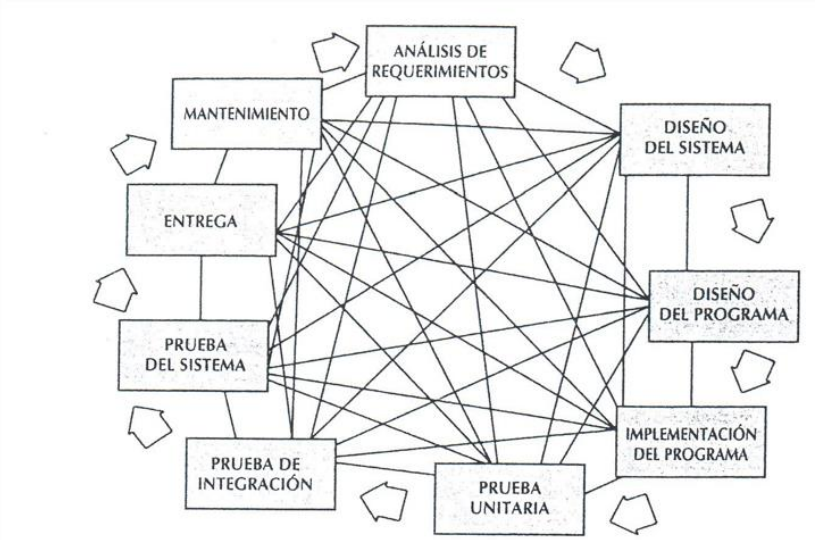


Dificultades

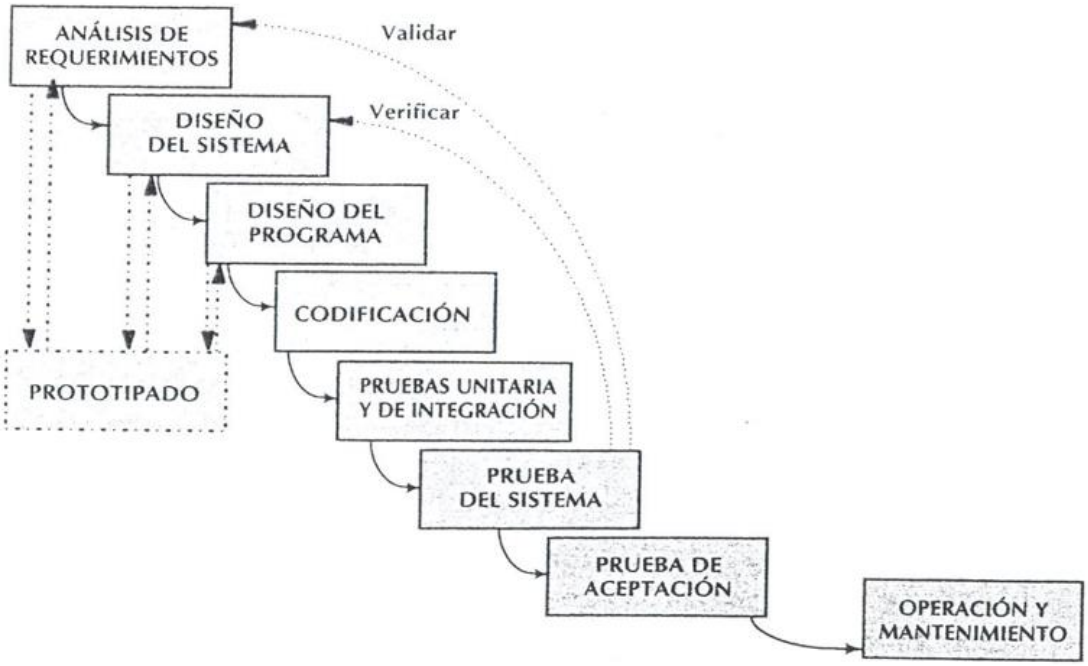
- Si un modelo tiene un error, fallan los próximos niveles
- No existen resultados concretos hasta que todo esté terminado
- Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final
- La eliminación de fallas suele ser extremadamente difícil durante las últimas 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 de la realidad en comparación con cascada

❑ Modelo de la realidad (sin control entre las etapas)



Modelo en cascada con prototipo



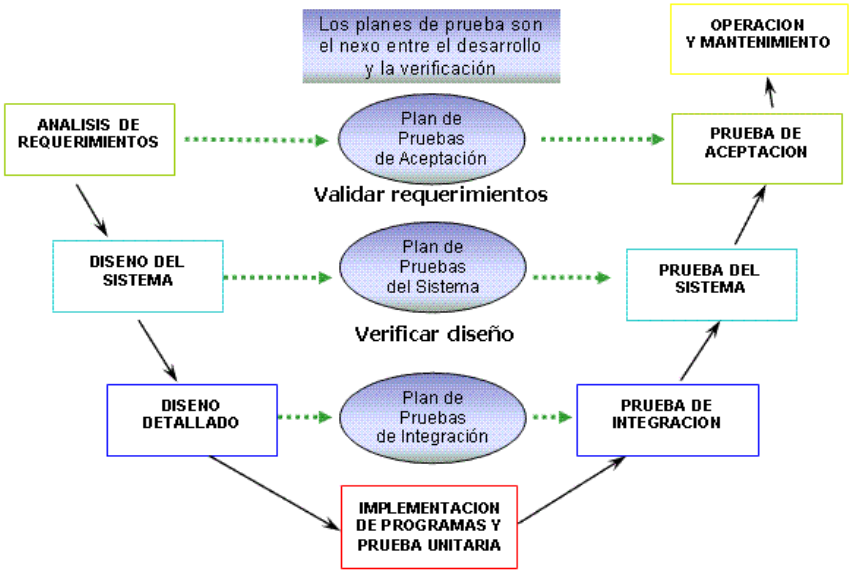
Modelo en V

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

La verificación de los requisitos se hace en la fase de desarrollo



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

Tipos

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 compresión unificada tanto de lo que se necesita como de lo que se propone como solución

Descartables

No tiene funcionalidad

Se utilizan herramientas de modelado

Su propósito principal es probar y validad requisitos antes del desarrollo completo

Comparación

	Descartable	Evolutivo
Enfoque de desarrollo	Rápido y sin rigor	Riguroso
Que construir	Solo las partes problemáticas	Primero las partes bien entendidas. Sobre una base sólida
Objetivo ultimo	Desecharlo	Lograr el sistema

Proyectos candidatos para un prototipado

Este modelo es útil en contextos donde:

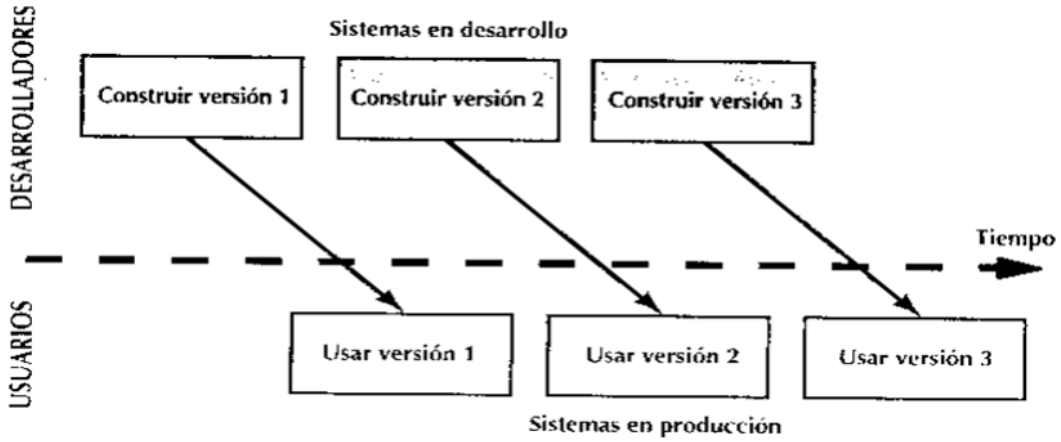
- Usuarios que no examinarán los modelos abstractos
- Usuarios que no determinarán sus requerimientos inicialmente
- Sistemas con énfasis en los formatos de E/S más que en los detalles algorítmicos
- Sistemas en los que haya que explorar aspectos técnicos
- Si el usuario tiene dificultad al tratar con 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 con respecto a desarrollos habituales
- Debe desarrollarse rápidamente
- Énfasis en la interfaz de usuario
- Equipo de desarrollo reducido
- Herramientas y lenguaje 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 operacional y el que está en desarrollo



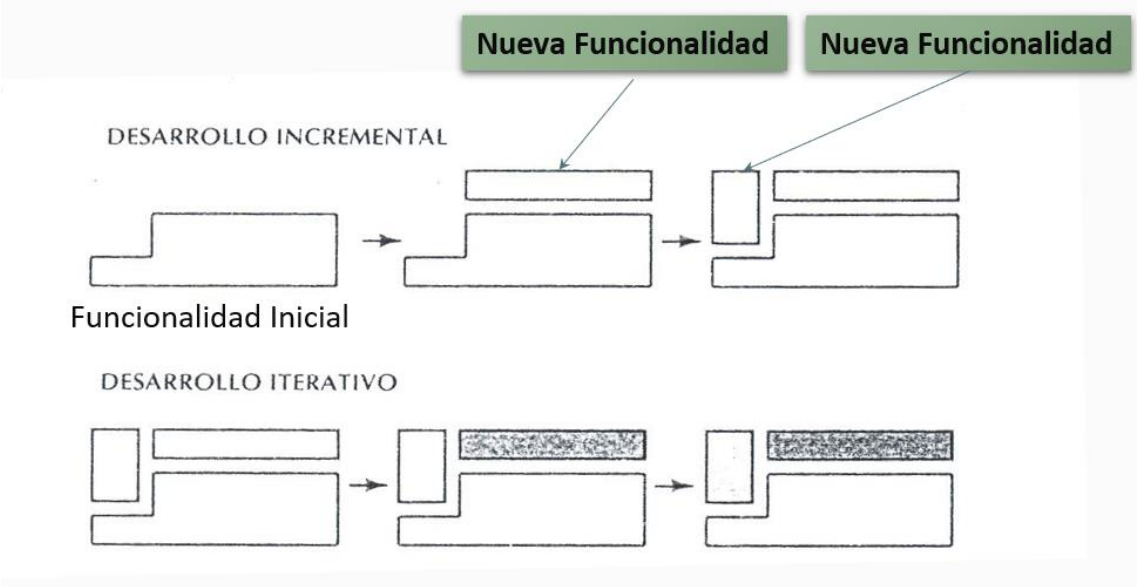
Tipos de modelos de desarrollo por fases

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

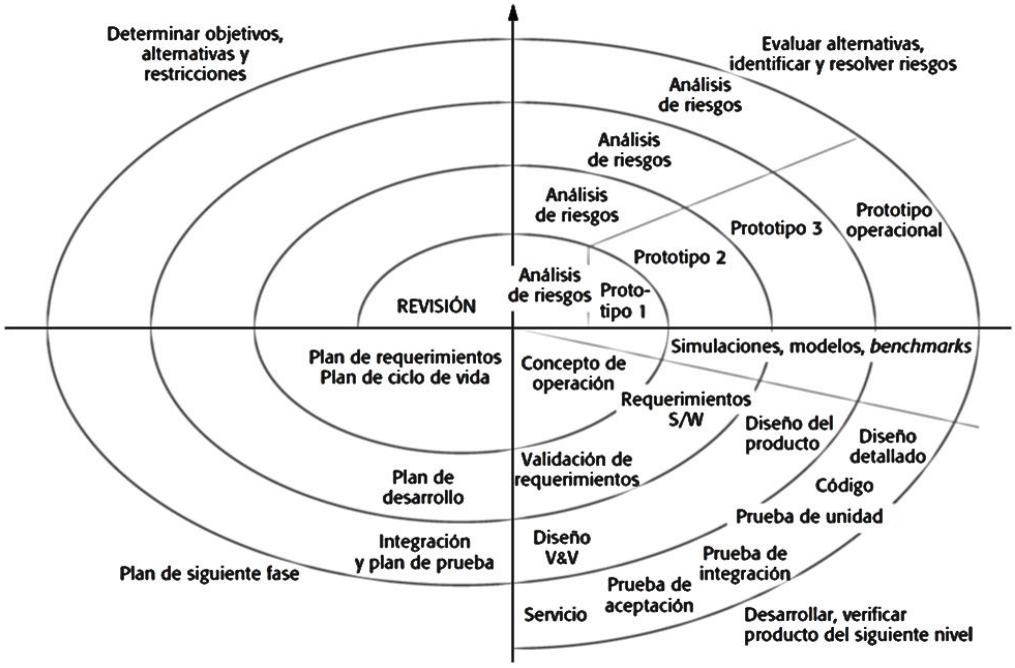


Modelo en espiral

- Combina las actividades de desarrollo con la de gestión de riesgo
- Tratar 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:
    - Objetivos de la porción correspondiente
    - Alternativas

Restricciones

Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente



Cada ciclo en la espiral se divide en cuatro sectores

1. Establecimiento de objetivos: Se identifican restricciones, se traza un plan de gestión, se identifican riesgos
2. Valoración y reducción del riesgo: Se analiza cada riesgo identificado y se determinan acciones
3. Desarrollo y Validación: Se determina el modelo de desarrollo
4. Planeación: El proyecto se revisa y se toma decisiones para la siguiente fase

Metodologías ágiles - Introducción

El éxito de un desarrollo está dado por la metodología empleada, la cual nos da una dirección a seguir para su correcta conclusión

Generalmente esta metodología llega asociado un marcado énfasis en el control del proceso, definiendo roles, actividades, herramientas y documentación detallada

Este enfoque no resulta ser muy adecuado para proyectos actuales donde el entorno del sistema es muy cambiante y se exige una reducción de tiempo

Ante estas dificultades, muchos equipos se resignan a prescindir de las buenas prácticas, asumiendo riesgos

En este contexto, las metodologías ágiles emergen como una posible solución

¿Qué son?

“Es un enfoque iterativo e incremental (evolutivo) de desarrollo de software”

El desarrollo iterativo es una estrategia de reproceso en la que el tiempo se separa para revisar y mejorar partes del sistema

Que sea incremental quiere decir que es una estrategia programada y en etapas, en las que las diferentes partes del sistema se desarrollan en diferentes momentos o a diferentes velocidades y se integran a medida que se completan

Definición de Fowler

Una metodología ágil es aquella en la que “se da prioridad a las tareas que dan resultados directos y que reducen la burocracia tanto como sea posible”, adaptándose además rápidamente al cambio de los proyectos

Objetivos

- Producir software de alta calidad con un costo efectivo y en el tiempo apropiado
- Esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto
- Ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas

Valores de la metodología ágil

- Respuesta al cambio: Fomenta la adaptabilidad en lugar de la estricta rigurosa planificación
- Individuos e interacciones: Enfatiza la importancia de la comunicación personal y el trabajo en equipo más que procesos y herramientas
- Software operante: Prioriza la entrega de software funcional sobre la documentación extensa y completa
- Colaboración con el cliente: Se centra en asociarse con los clientes en lugar de negociar contratos formales

Principios

- La mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valuable
- Los cambios de requerimientos son bienvenidos, aún tardíos, en el desarrollo. Los procesos ágiles capturan los cambios para que el cliente obtenga ventajas competitivas
- Entregas frecuentes de software, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente
- Usuarios y desarrolladores deben trabajar juntos durante todo el proyecto
- Construir proyectos alrededor de motivaciones individuales
- Darles el ambiente y el soporte que ellos necesitan y confiar el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores
- El software que funciona es la medida clave de progreso
- Los procesos ágiles promueven un desarrollo sostenible. Los stakeholders, desarrolladores y usuarios deberían ser capaces de mantener un paso constante indefinidamente
- Atención continua a la excelencia técnica y buen diseño incrementa la agilidad

- Simplicidad (el arte de maximizar la cantidad de trabajo no dado) es esencial
- Las mejores arquitecturas, requerimientos y diseños surgen de la propia organización de los equipos
- A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento en consecuencia

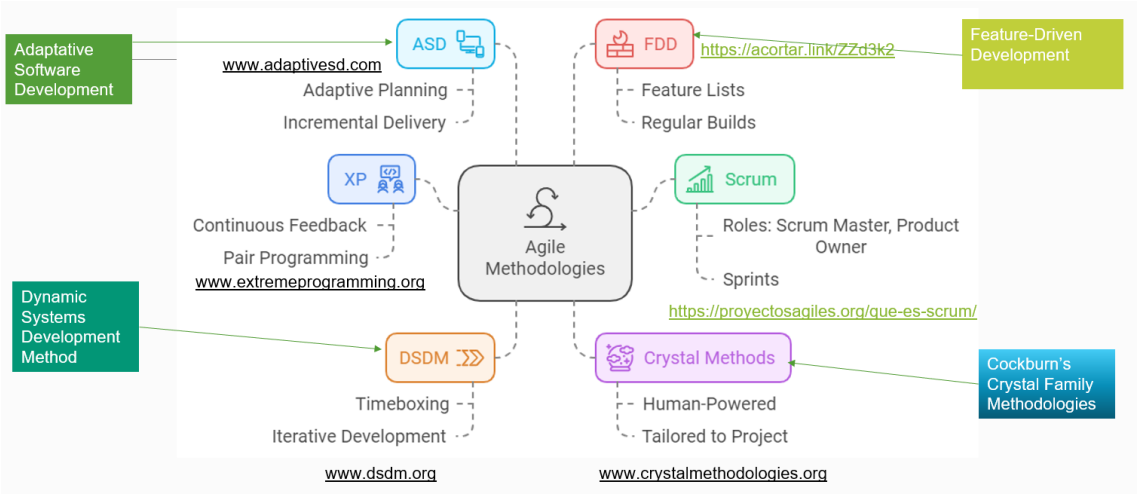
Comparación ágil vs no ágil

- Agil
    - Artefactos dinámicos
    - Roles fluidos
    - Contratos flexibles
    - Grupos pequeños y autoorganizados
    - Involucramiento integral del cliente
    - Arquitectura evolutiva
- No ágil
    - Artefactos rígidos
    - Roles definidos
    - Contratos detallados
    - Grupos grandes y especializados
    - Involucramiento limitado del cliente
    - Arquitectura bien definida

Desventajas

- **Aunque es atractiva la idea de involucrar al cliente en el proceso de desarrollo**, los representantes del cliente están sujetos a otras presiones y no intervienen por completo en el desarrollo del software
- **Priorizar los cambios podría ser difícil**, sobre todo en sistemas donde existen muchos participantes. Cada uno por lo general ofrece diversas prioridades a diferentes cambios
- **Mantener la simplicidad requiere trabajo adicional**. Bajo la presión de fechas de entrega, es posible que los miembros del equipo carezcan de tiempo para realizar las simplificaciones deseables al sistema
- **Muchas organizaciones, especialmente las grandes compañías, pasan años cambiando su cultura, de tal modo que los procesos se definan y continúen**. Para ellas, resulta difícil moverse hacia un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo.
- Por lo general, el documento de requerimientos del software forma parte del contrato entre el cliente y el proveedor. Como en los métodos ágiles se minimiza la documentación, suele ser complejo reglamentarlo
- La mayoría de los libros que describen los métodos ágiles y sus experiencias, hablan del uso de dichos métodos para el desarrollo de nuevos sistemas. Sin embargo, una enorme cantidad de esfuerzo en ingeniería de software se usa en el mantenimiento y evolución de los sistemas ya existentes. Al no existir documentación, se complejizaría.

Principales metodologías ágiles



Extreme programming

Es una disciplina de desarrollo de software basado en los valores de

- La sencillez/simplicidad
- La comunicación
- La retroalimentación
- La valentía/Coraje
- El respeto

Su acción consiste en llevar a todo el equipo reunido en la presencia de prácticas simples, con suficiente información para ver dónde están y para ajustar las prácticas a su situación particular

¿En qué se basa?

- Desarrollo iterativo e incremental: Pequeñas mejoras, unas tras otras
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión
- Programación en parejas
- Frecuente integración del equipo de programación con el cliente o usuario
- Corrección de todos los errores antes de añadir nueva funcionalidad
- Refactorización del código
- Propiedad del código compartida
- Simplicidad en el código

Características esenciales

- Historias de usuario
- Roles
- Proceso
- Practicas

Roles

- Programador
    - Responsable de decisiones técnicas

    - Responsable de construir el sistema
    - Sin distinción entre analistas, diseñadores o codificadores
    - Diseñan, programan y realizan las pruebas
- Jefe de proyecto/Manager
    - Organiza y guía las reuniones
    - Asegura condiciones adecuadas para el proyecto
- Cliente
    - Es parte del equipo
    - Determina qué construir y cuándo
    - Establece las pruebas funcionales
- Entrenador
    - Responsable del proceso
    - Tiende a estar en un segundo plano a medida que el equipo madura
- Encargado de pruebas
    - Ayuda al cliente con las pruebas funcionales
    - Se asegura de que las pruebas funcionales se superan
- Rastreados
    - Metric man
    - Observa sin molestar
    - Conserva datos históricos

Proceso

El ciclo de vida consiste en:

1. Exploración
2. Planificación
3. Iteraciones
4. Producción
5. Mantenimiento
6. Muerte

1 – Exploración

- Los clientes plantean las historias de usuario que son de interés para la primer entrega del producto
- El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto
- Se construye un prototipo

Esta fase toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología

2 – Planificación

- El cliente establece la prioridad de cada historia de usuario
- Los programadores realizan una estimación del esfuerzo
- Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente

Esta fase dura unos pocos días

3 – Iteración

- El plan de entrega está compuesto por iteraciones de no más de tres semanas
- El cliente es quien decide qué historias se implementarán en cada iteración
- Al final de la última iteración el sistema estará listo para entrar en producción

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado

4 – Producción

- Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente
- Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase

5 – Mantenimiento

- Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones
- La fase de mantenimiento puede requerir de nuevo personal dentro del equipo y cambios en su estructura

6 – Muerte

- Es cuando el cliente no tiene más historias para ser incluidas en el sistema
- Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura
- La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo

Prácticas

- Testing:
    - Los programadores continuamente escriben pruebas unitarias las cuales deben correr sin problemas para que el desarrollo continúe.
    - Los clientes escriben pruebas demostrando que las funcionalidades están terminadas
- Refactoring:
    - Actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios
- Programacion de a pares:
    - Todo el código de produccion es escrito por dos programadores en una maquina
- Propiedad colectiva del código:
    - Cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento.

    - Motiva a contribuir con nuevas ideas, evitando a la vez que algún programador sea imprescindible
- Integración continua:
    - Cada pieza de código es integrada en el sistema una vez que esté lista. Así el sistema puede llegar a ser integrado y construido varias veces en un mismo día
    - Reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido
- Semana de 40 horas
    - Se debe trabajar un máximo de 40 horas por semana
    - El trabajo extra desmotiva al equipo
    - Los proyectos que requieren trabajo extra para intentar cumplir los plazos suelen ser entregados con retraso al final. En lugar de esto se puede realizar una planificación para cambiar el ámbito del proyecto o la fecha de entrega
- Cliente en el lugar de desarrollo
    - El cliente tiene que estar presente y disponible todo el tiempo para el equipo
- Estandares de codificación
    - Los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo

SCRUM

Es un marco de trabajo utilizado para desarrollar productos complejos donde se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente y obtener el mejor resultado posible de un proyecto

Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos

Se define como “Una manera simple de manejar problemas complejos”, proporcionando un paradigma de trabajo que soporta la innovación y permite que equipos autoorganizados entreguen resultados de alta calidad en tiempos cortos

Se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto

Proceso

Es iterativo e incremental

Se busca poder atacar todos los problemas que surgen durante el desarrollo del proyecto

Las fases de desarrollo se solapan, de manera que no es un proceso de cascada por cada iteración, sinó que tenemos todas estas etapas juntas que se ejecutan una y otra vez hasta que se crea suficiente

Principios

- Eliminar el desperdicio: No generar artefactos ni perder el tiempo haciendo cosas que no le suman valor al cliente
- Construir la calidad con el producto: La idea es inyectar la calidad directamente en el código desde el inicio
- Crear conocimiento: En la practica no se puede tener el conocimiento antes de empezar el desarrollo

- Diferir las decisiones: Tomarlas en el momento adecuado, esperar hasta ese momento. Ya que uno tiene más info a medida que va pasando el tiempo
- Entregar rápido: Debe ser una de las ventajas competitivas más importantes
- Respetar a las personas: La gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada
- Optimizar el todo: Optimizar todo el proceso, ya que el proceso es una unidad y para lograr tener éxito y avanzar hay que tratarlo como tal

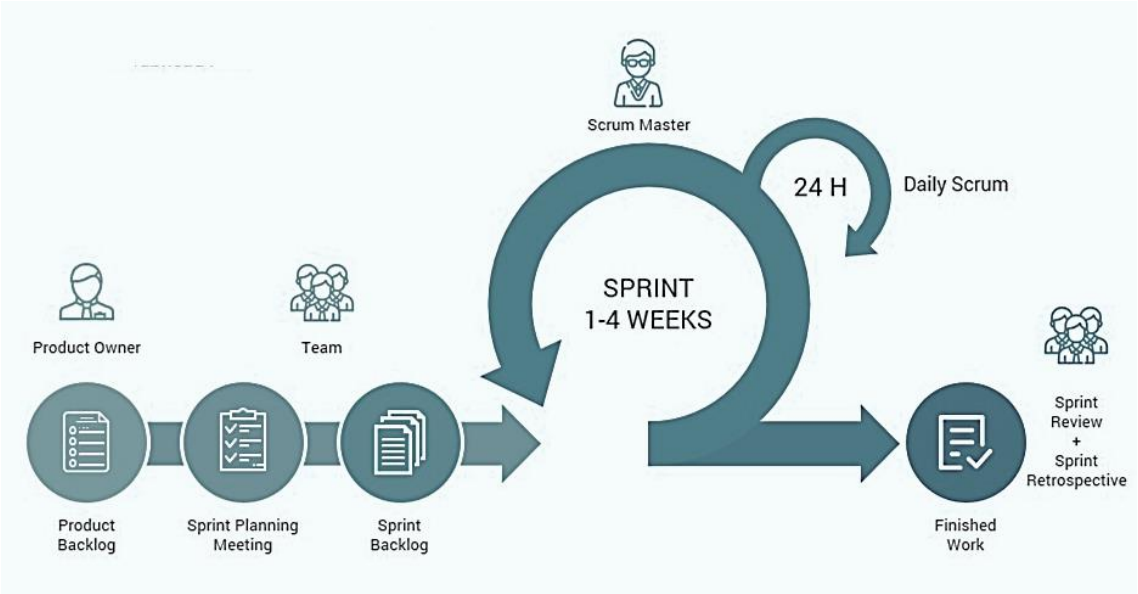
Roles

- Product owner (Propietario)
    - Conocer y marca las prioridades del proyecto o producto
- Scrum master (Jefe)
    - Es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema. Su responsabilidad es, entre otras, hacer de paraguas ante las presiones externas
- Scrum Team (Equipo)
    - Son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el propietario
- Usuarios o Cliente
    - Beneficiarios finales del producto
    - Son quienes, viendo los progresos, pueden aportar ideas, sugerencias o necesidades

Artefactos

- Product Backlog
    - Es la lista maestra que contiene toda la funcionalidad deseada en el producto
    - La caracteristica más importante es que la funcionalidad se encuentra ordenada por prioridad
- Sprint Backlog
    - Es la lista que contiene toda la funcionalidad que el equipo se comprometió a desarrollar durante un sprint determinado
- Burndown chart
    - Muestra un acumulativo del trabajo hecho, día a día
- Entre otros…

Proceso



- Product backlog: Lista priorizada de funcionalidades desde la perspectiva del cliente
- Sprint planning meeting: Se planifica la entrega, en el tiempo acordado para la duración del sprint
- Sprint backlog: Subconjunto de requerimientos del producto backlog seleccionados para la iteración actual y su plan de tareas de desarrollo
- Daily Scrum: Reunion ágil, corta, de 15 a 30 minutos de duración en un lugar y hora convenida. Con el objetivo de dar visibilidad de los avances diarios
- Sprint review: Se lleva a cabo al final del sprint y se muestra lo implementado en dicha iteración. Se evaluan cada uno de los requerimientos implementados del sprint backlog
- Sprint retrospective: Evento donde se toma en cuenta los resultado del sprint, discutidos previamente en la sprint review. Es una reunión de lecciones aprendidas

¿Cuándo aplicarlo?

Está pensado para ser aplicado en proyectos donde el “caos” es una constante, aquellos proyectos en los que tenemos requerimientos dinámicos y que tenemos que implementar tecnología de punta. Proyectos difíciles.

Kanban

Es un enfoque Lean de desarrollo de software ágil

Se presenta como una herramienta de gestión que prescribe sólo 3 prácticas

- Limitar el trabajo en curso
- Visualizar el flujo de trabajo
- Medir el tiempo promedio de entrega

Es una herramienta simple, pero potente

Lean

Filosofía que hace hincapié en la eliminación de residuos o de no valor añadido a través de la mejora continua para agilizar las operaciones, optimizando la eficiencia del equipo. Está

centrado en el cliente y enfatiza el concepto de eliminar cualquier actividad que no agregue valor a la creación o entrega de un producto o servicio. Lean se centra en ofrecer una mayor calidad, reducir el tiempo de ciclo y reducir los costos

Fomenta la mejora continua y la reducción de desperdicios.

Sobre la metodología

Kanban trata de administrar el flujo de trabajo. Inicialmente, no reemplaza nada que la organización haga, simplemente impulsa el cambio

No se definen roles y no prescribe una secuencia de pasos definidos

Es habitual combinar en las prácticas de otras metodologías ágiles

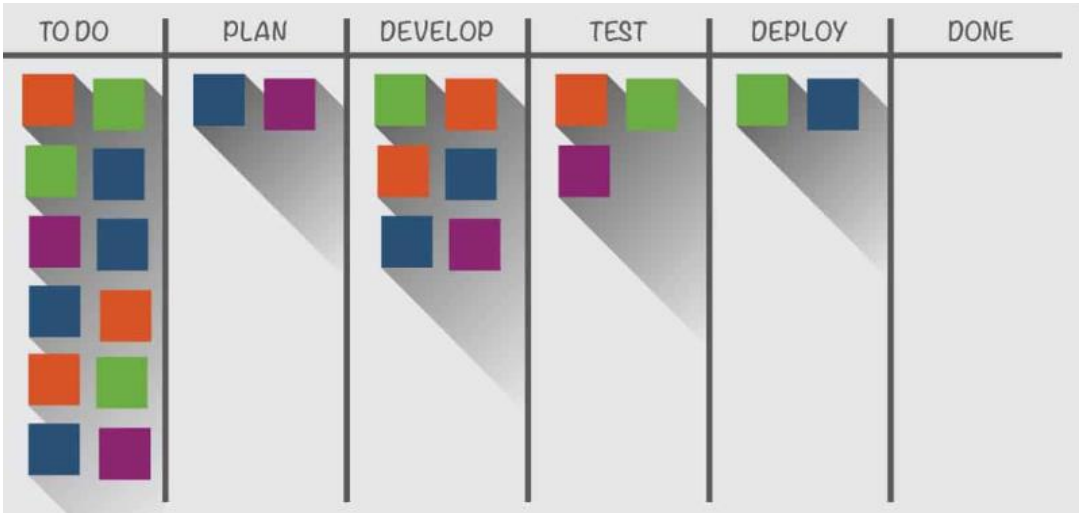
Los tableros Kanban son la herramienta ágil por excelencia de esta metodología, donde se visualizan las tareas a realizar, las realizadas y lo pendiente. Ayuda a un equipo a visualizar explícitamente el proceso de desarrollo.

Evita explícitamente los problemas de un proceso de las características de un proceso en cascada mediante la aplicación de lotes pequeños de trabajo y desarrollo incremental

Está compuesto por una serie de columnas que representan los diversos estados que atraviesa un requerimiento durante el proceso de desarrollo. Estas columnas contienen tarjetas que se mueven de un estado a otro

Cada columna tendrá un limite para el trabajo en curso. Cuando una tarea es completada en una columna, se mueve a la siguiente, formando un espacio libre en la columna actual representando la capacidad de trabajo disponible

Entonces el equipo toma una tarea terminada desde el estado anterior y la desplaza al estado actual, ya que hay capacidad para trabajarla



Desarrollo de software basado en modelos

La construccion de un sistema de software debe ser precedida por la construccion de un modelo

Un modelo del sistema consiste en una conceptualización del dominio del problema y actúa como una especificación precisa de los requerimientos que el sistema de software debe satisfacer (Abstraccion de elementos del problema, comunicación, negociación con el usuario)

Desarrollo de software dirigido por modelos

El adjetivo “dirigido“en MDD, a diferencia de “basado”, enfatiza que este paradigma asigna a los modelos un rol central y activo: son al menos tan importantes como el código fuente

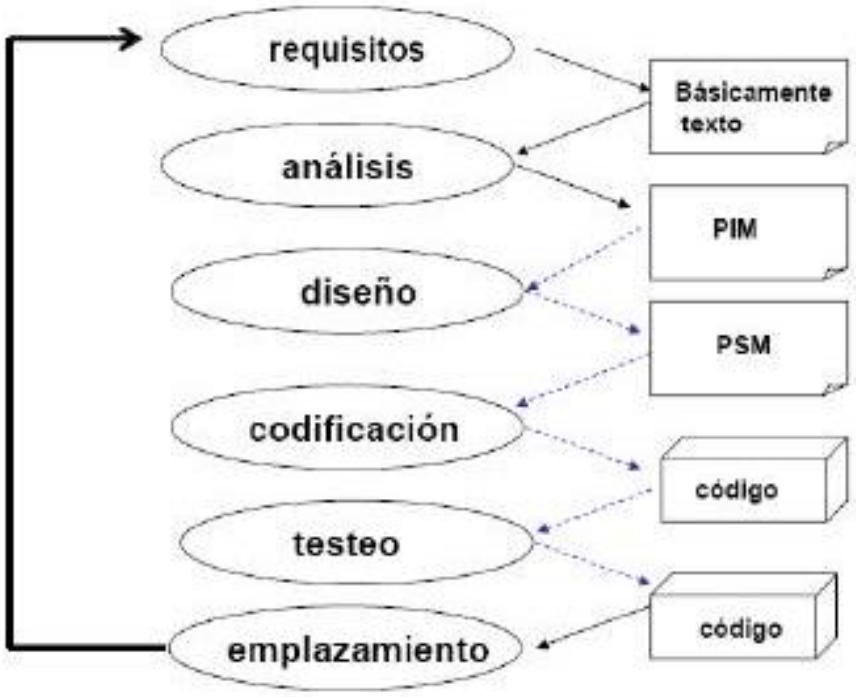
Es la evolución natural de la ingeniería de software basada en modelos, enriquecida mediante el agregado transformaciones automáticas entre modelos

Model Driven Development (MDD) promueve enfatizar los siguientes puntos clave

- Mayor nivel de abstracción en la especificación tanto del problema a resolver como de la solución correspondiente
- Aumento de confianza en la automatización asistida por computadora para soporta el análisis, el diseño y la ejecución
- Uso de estándares industriales como medio para facilitar las comunicaciones, la interaccion entre diferentes aplicaciones y productos, y la especialización tecnológica
- Los modelos son los conductores primarios en todos los aspectos del desarrollo de software

Los modelos pasan de ser entidades contemplativas (artefactos interpretados por los diseñadores y programadores) para convertirse en entidades productivas a partir de las cuales se deriva a la implementación en forma automática

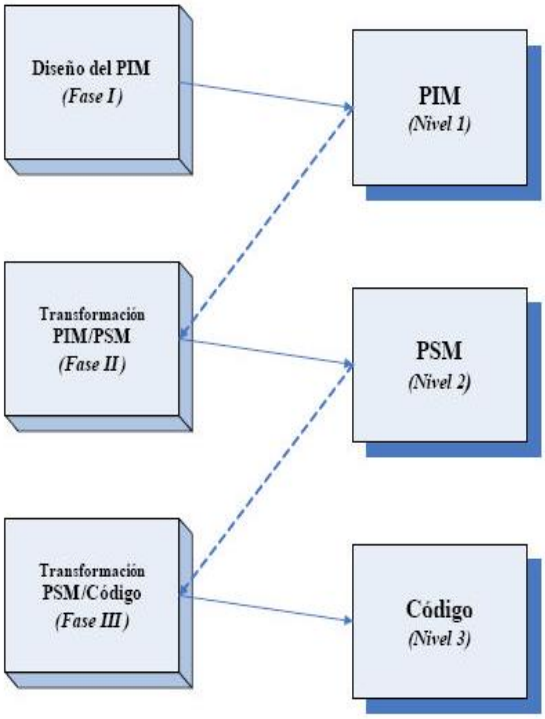
Ciclo de vida del software dirigido por modelos



Modelos de MDD (PIMs y PSMs)

- Platform Independent Model (PIM): Un modelo de un sistema que no contiene información acerca de la plataforma o la tecnología que es usada para implementarlo

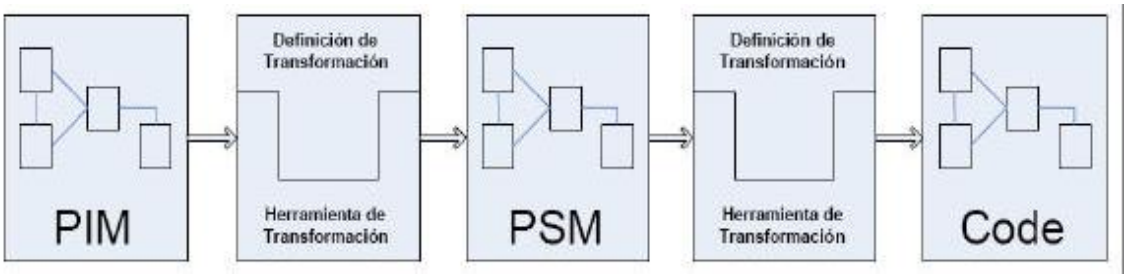
- Platform Specific Model (PSM): Un modelo de un sistema que incluye información acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica
- Transformacion de modelos: Especifica el proceso de conversión de un modelo en otro modelo del mismo sistema. Cada transformación incluye, al menos
    - Un PIM
    - Un Modelo de la Plataforma
    - Una Transformación
    - Un PSM



¿Qué es una transformación?

Consiste en una colección de reglas las cuales son especificaciones no ambiguas de las formas en que un modelo (o parte de él) puede ser usado para crear otro modelo (o parte de él)

El patron MDD es normalmente utilizado varias veces para producir una sucesión de transformaciones



Beneficios de MDD

- Incremento en la productividad (modelos y transformaciones)
- Adpatacion a los cambios tecnológicos
- Adaptacion a los cambios de requisitos
- Consistencia (automatización)

- Reuso (de modelos y transformaciones)
- Mejoras en la comunicación con los usuarios y entre desarrolladores (los modelos permanecen actualizados)
- Captura de la experiencia (cambio de experto)
- Los modelos son productos de larga duración (resisten cambios)
- Posibilidad de demorar decisiones tecnológicas

Calidad

La calidad es un término totalmente subjetivo, que va a depender del juicio de la persona que intervenga en la evaluacion

Se puede definir como la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor

¿Qué es la calidad?

Criterios erróneos

- Un producto de calidad es un producto de lujo: Podemos encontrar ejemplo donde hay productos de calidad que no necesariamente son de lujo
- La calidad es intangible y por lo tanto no mensurable: Si para nosotros, la calidad son propiedades que se pueden juzgar, entonces podemos reconocer esas propiedades y saber si eso tiene o no calidad, por lo que es mensurable
- Los problemas son originados por los trabajadores de produccion: No necesariamente esa sí, el problema puede surgir antes
- La calidad se origina en el departamento de calidad: Tampoco es así, la calidad tiene que estar presente en todas las personas involucradas en la creación del producto

Definiciones de calidad - “Gurus de calidad”

- Crosbi: Conformidad con los requisitos
- Deming: Satisfaccion del usuario
- Juran: Adecuación al uso
- Feigenbaum: Cumplir con las expectativas del cliente
- Shewart: Su lado subjetivo

Definicion de calidad por normas internacionales

Una norma es un documento, establecido por consenso y aprobado por un organismo reconocido, que proporciona para un uso común y repetido, una serie de reglas, directrices o características

Las principales normas internacionales definen calidad como

- ISO 9000: El grado en el que un conjunto de características inherentes cumple con los requisitos
- ISO 8402: Conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implícitas

La organización internacional de normalización es una organización para la creación de estándares o normas internacionales compuesta por diversas organizaciones nacionales de normalización

Sistemas de información

Whitten y Bentley lo definen como “Conjunto de personas, datos, procesos y tecnología de información que interactúan para recopilar, procesar, guardar y proporcionar como salida la información necesaria para brindar soporte a una organización”

Un sistema de información abarca más que el aspecto meramente computacional, ya que se debe tener en cuenta también el modo de organizar estas herramientas y de obtener la información necesaria para el correcto funcionamiento de la empresa

Stylianou y Kumar plantean que se debe apreciar la calidad desde un todo, donde cada parte que la compone debe tener su análisis de calidad

Visión holística de la calidad (Stylianou y Kumar)

- Calidad de la infraestructura: Incluye la calidad de las redes y sistemas de software, entre otros.
- Calidad de la gestión: Incluye el presupuesto, planificación y programacion
- Calidad del servicio: Incluye los procesos de atención al cliente
- Calidad de datos: Calidad de los datos que ingresan al sistema
- Calidad de la información: Está relacionada con la calidad de los datos
- Calidad del software: Calidad de las aplicaciones de software construidas o mantenidas o con el apoyo de sistemas de información

Calidad de software

Se divide en

- Calidad del producto obtenido
- Calidad del proceso de desarrollo

Estas divisiones son independientes

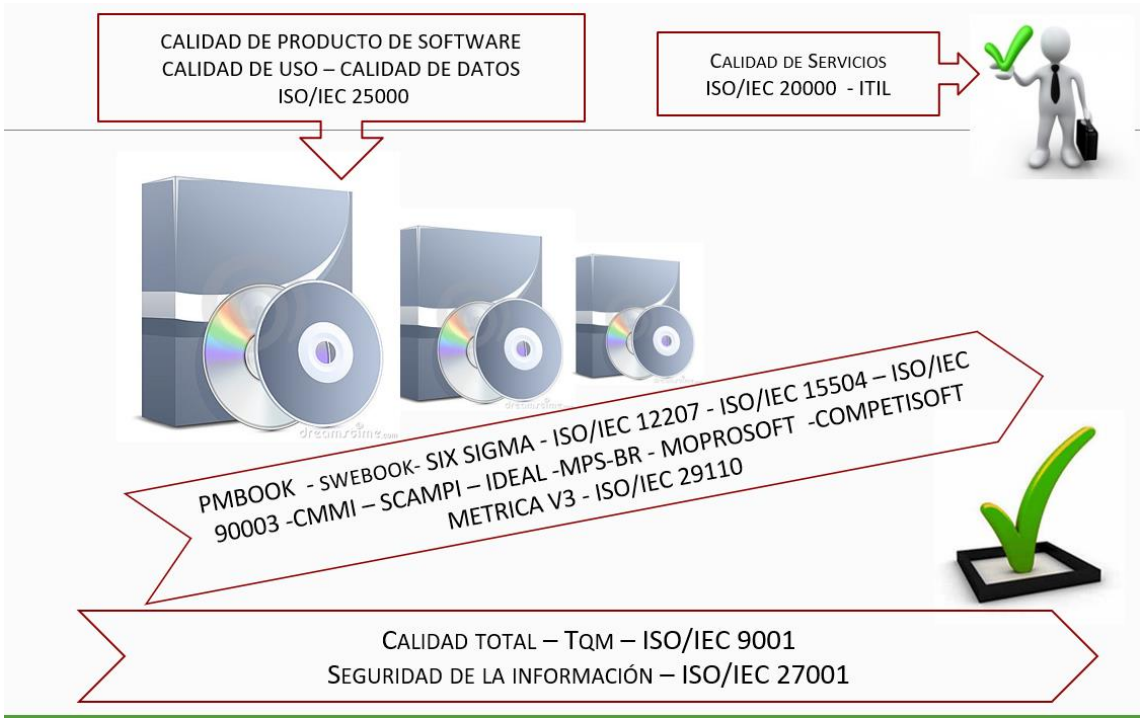
Calidad del producto y proceso

- Producto: La estandarización del producto define las propiedades que debe satisfacer el producto de software resultante
- Proceso: La estandarización del proceso define la manera de desarrollar el producto de software

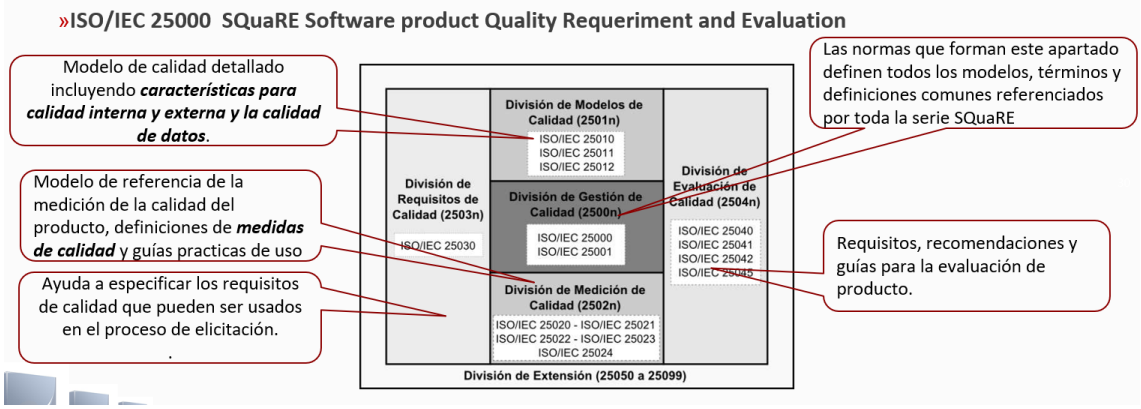
Sin un buen proceso de desarrollo es casi imposible obtener un buen producto

Clasificación de normas y modelos de calidad

- Calidad del producto de software:
    - Calidad de uso, calidad de datos, ISO/IEC 25000
- Calidad del proceso de desarrollo
    - PMBOOK, SWEBOOK, SIX SIGMA, ISO/IEC 12207, ISO/IEC 15504 (ISO 33000), ISO/IEC 90003, CMMI, COMPETISOFT, etc.
- Calidad de la organización
    - Seguridad de la información ISO/IEC 27001, ISO/IEC 9001
- Calidad de servicios
    - ISO/IEC 20000 - ITIL



Norma/Modelo de calidad SQuaRE ISO/IEC 25000



Características de SQuaRE ISO/IEC 25010

Un producto de software debe cumplir con

- Portabilidad
- Seguridad
- Facilitad de mantenimiento
- Compatibilidad
- Funcionalidad
- Confiabilidad
- Facilidad de uso
- Eficiencia

Normas

- ISO/IEC 12207: Establece un modelo de procesos para el ciclo de vida del software

- LA familia de normas ISO/IEC 30000 proporciona un marco de trabajo coherente para la evaluación de procesos de software que sustituye las diferentes partes de la norma ISO/IEC 15504
- ISO/IEC 15504: Es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos de las organizaciones en la adquisición, desarrollo, evolución y soporte de productos y servicios

CMM (1993)

Es un modelo de evaluación de los procesos de una organización (Capability Maturity Model)

Marco de referencia para desarrollar procesos efectivos

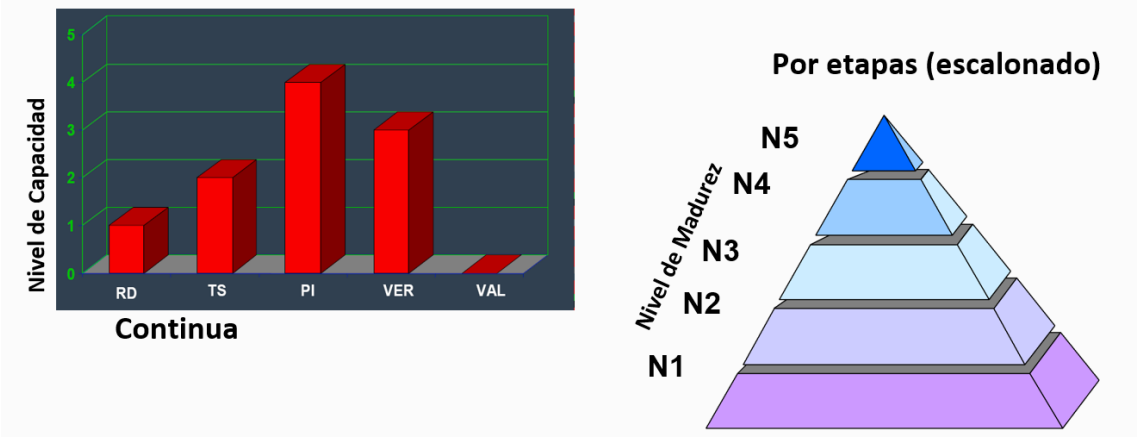
Proporciona un Marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora e implementar esas mejoras

CMMI (2000)

Modelo de capacidad y madurez – Integracion

Posee dos vistas que permiten un enfoque diferente según las necesidades de quien vaya a implementarlo

- Escalonado: Centra su foco en la madurez de la organización, igual que CMM. “Mide” todos los procesos de la organización evaluando la madurez de cada uno
- Continuo: Enfoca las actividades de mejora y evaluación en la capacidad de los diferentes procesos. Presenta 6 niveles de capacidad que indican qué tan bien se desempeña la organización en un área de proceso individual. “Mide” toda la organización evaluando su madurez



Niveles de madurez

1. Inicial: Proceso impredecible, poco controlado y reactivo
2. Gestionado: Proceso caracterizado por proyectos y frecuentemente reactivo
3. Definido: Proceso caracterizado por la organización y proactivo
4. Gestionado cuantitativamente: El proceso es controlado cuantitativamente
5. Optimizado: Enfoque en la mejora del proceso

ISO

La familia ISO 9000 es un conjunto de normas de “gestión de calidad” aplicables a cualquier tipo de organización, con el objetivo de obtener mejoras en la organización y, eventualmente, arribar a una certificación, punto importante a la hora de competir en mercados globales

Familia de las ISO 9000

- ISO 9001:2015: Quality management system – requirements
- IRAM – ISO 9001:2015: Sistema de gestión de la calidad – requisitos, norma publicada por iso y traducida por IRAM
- ISO 90003:2004: Basada en ISO 9001:2000, tiene directrices para la interpretación en el proceso de software. Proporciona una guía para identificar las evidencias dentro del proceso de software para satisfacer los requisitos de la ISO 9001

Beneficios de trabajar con un sistema de gestión de calidad ISO 9001

- Rendimiento financiero: Aumenta los resultados financieros y la rentabilidad
- Cumplimiento legal: Asegura el cumplimiento de los requisitos legales y del cliente
- Rendimiento organizacional: Mejora la eficiencia y productividad general
- Toma de decisiones: Facilita elecciones informadas y efectivas
- Satisfaccion del cliente: Mejora la satisfacción y lealtad del cliente

SGC – IRAM - ISO 9001

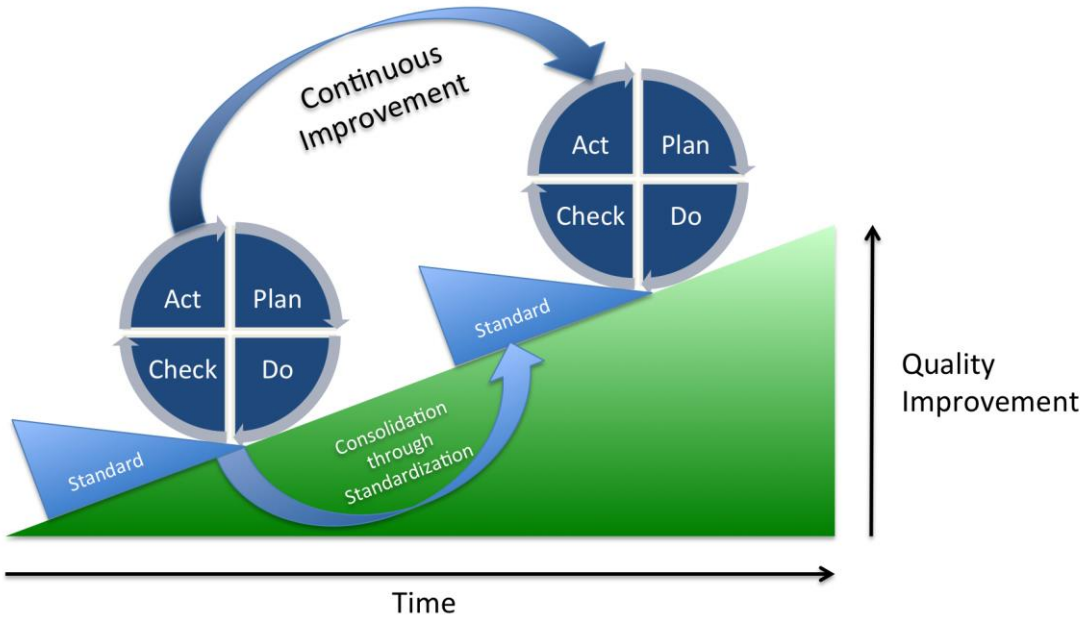
SGC = Sistema de gestión de calidad

SGC de mejora cuntinua

Implica un compromiso constante por identificar oportunidades de mejora, implementar cambios y evaluar su impacto. En el contexto de la ingeniería de software, la mejora continua busca optimizar los procesos de desarrollo, aumentar la calidad del software y satisfacer mejor las necesidades de los clientes

Ciclo PDCA (Plan Do Check Act)

Un ciclo iterative para planificar, implementar, verificar y actuar sobre las mejoras



Normas

Calidad de producto de software	Se evalúa la calidad mediante	ISO/IEC 25000	Está compuesto por distintos modelos. Define características que pueden estar presentes o no en el producto. La norma nos permite evaluar si están presentes o no , y de qué manera evaluarlas. EJ: Seguridad, Compatibilidad, Seguridad. Etc.
Calidad de proceso de desarrollo de software	Se evalúa la calidad mediante	ISO/IEC 12207	ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software. Define cómo debería ser el modelo de proceso para ser completo y con calidad. Actividades, tareas etc.
		ISO/IEC 15504 (reemplazada por ISO 33000)	Es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos Define que se debe tener en cuenta para evaluar el modelo de proceso y concluir si es completo y con calidad.
		ISO/IEC 90003	Proporciona una guía sobre cómo aplicar la ISO 9001 en procesos de software
		CMMI	Proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora, e implementar esas mejoras. Se utiliza para organizaciones desarrolladoras de software de medianas a grandes dimensiones
Calidad de Procesos/Servicios en general	se evalúa mediante	ISO 9001	La Norma ISO 9001 determina los requisitos para establecer un Sistema de Gestión de la Calidad. Forma parte de la familia ISO 9000, que es un conjunto de normas de "gestión de la calidad" aplicables a cualquier tipo de organización con el objetivo de obtener mejoras en la organización y, eventualmente arribar a una certificación, punto importante a la hora de competir en los mercados globales.