

Resumen Ingeniería de Software 1

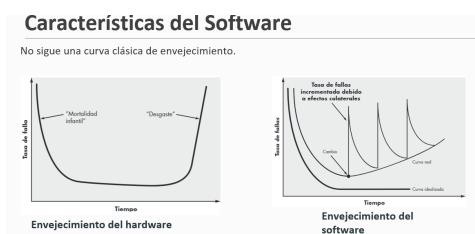
Software

¿Qué es?

Instrucciones(programas de cómputo), procedimientos, reglas, documentación y datos asociados que formar parte de las operaciones de un sistema de computación(IEEE)

Características

- Elemento lógico
- Se desarrolla, no se fabrica
- No se desgasta



Tipos de productos 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: El término software libre(o programas libres) se refiere a libertad, tal como Fue concebido por Richard Stallman en su definición. En concreto se refiere a cuatro libertades:
 - Libertad para ejecutar el programa en cualquier sitio, 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.

Clasificación del Software

- De sistemas
- De aplicación
- Científico y de ingeniería
- Integrado
- De línea de productos
- Aplicaciones Web/Móviles
- De inteligencia artificial

¿Qué es la Ingeniería de Software?

Disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema incluyendo la evolución de éste, luego que se comienza a ejecutar.

Definición según la IEEE:

El uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software. A su vez, se estudian las técnicas mencionadas

Usa métodos cuantificables, dentro de tiempos y costos estimados. Para el "desarrollo, operación y mantenimiento"

Participantes en el desarrollo del Software

- Gerentes ejecutivos
- Gerente de proyecto
- Profesionales

- Clientes
- Usuarios finales

Responsabilidad profesional y ética

La Ingeniería de Software se desarrolla en un marco económico, social y legal. No debe utilizar su capacidad y habilidades de forma deshonesta, o de forma que deshonre la profesión.

Dentro de ello está:

- Confidencialidad
- Competencia
- Derechos de la propiedad intelectual
- Uso inapropiado de las computadoras

Técnicas de comunicación

La comunicación es la base para la obtención de las necesidades del cliente. Al hablar de necesidades, en términos más técnicos, estamos hablando de Requerimientos

Requerimientos: Un Requerimiento o requisito es una 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 según IEEE:

Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo.

Fuentes de requerimientos:

- Documentación
- Stakeholders (El término se utiliza para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente)

- Especificación de sistemas similares

Puntos de Vista

Existen tres 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 mismo 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

Elicitación de Requerimientos

Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema

Objetivos:

- Conocer el dominio del problema para poder comunicarse con clientes y usuarios y entender sus necesidades
- Conocer el sistema actual
- Identificar las necesidades, tanto explícitas como implícitas, de clientes y usuarios y sus expectativas sobre el sistema a desarrollar

Problemas de comunicación:

- Dificultad para expresar claramente las necesidades
- No ser conscientes de sus propias necesidades
- No entender cómo la tecnología puede ayudar
- Miedo a parecer incompetentes por ignorancia tecnológica
- Limitaciones cognitivas
 - No conocer el dominio del problema
 - Hacer suposiciones sobre el dominio del problema

- Hacer suposiciones sobre aspectos tecnológicos
- Conducta humana
 - Conflictos y ambigüedades en los roles de los participantes
 - Pasividad de clientes, usuarios o ingenieros de requisitos
- Técnicos
 - Complejidad del dominio del problema
 - Complejidad de los requisitos

Técnicas de elicitation

Recopilación de información

Métodos discretos

Los métodos discretos son menos perturbadores que otras formas de averiguar los requerimientos

Se consideran insuficientes para recopilar información cuando se utilizan por 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 datos existentes

Recolección de hechos a partir de la documentación existente.

¿Qué tipo de documentos pueden enseñar algo acerca del sistema?

Organigrama (identificar el propietario, usuarios claves).

Memos, notas internas, minutos, registros contables.

Solicitudes de proyectos de sistemas de información anteriores.

Permiten conocer el historial que origina el proyecto.

Documentos que describen la funcionalidad del negocio que está siendo analizada.

Declaración de la misión y plan estratégico de la organización.
Objetivos formales del departamento en cuestión.
Políticas, restricciones, procedimientos operativos.
Bases de Datos.
Sistemas en funcionamiento.

Documentación de sistemas anteriores.

Diagramas.
Diccionario o Repositorios de proyecto.
Documentos de diseño.
Manuales de operación y/o entrenamiento

2. Investigación y visitas al lugar

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

3. Observación del ambiente de trabajo

El analista se convierte en observador de las personas y actividades con el objeto de aprender acerca del sistema.

Lineamientos de la observación:

Determinar quién y cuándo será observado.
Obtener el permiso de la persona y explicar el porqué será observado.
Mantener bajo perfil.
Tomar nota de lo observado.
Revisar las notas con la persona apropiada.
No interrumpir a la persona en su trabajo.

Ventajas

Datos confiables
El analista puede ver exactamente lo que se hace (tareas difíciles de explicar con palabras).
Análisis de disposiciones físicas, tránsito, 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

Hay métodos interactivos que pueden usarse para obtener los requerimientos de los miembros de la organización. Aunque son distintos en su implementación, estos métodos tienen muchas cosas en común. La base es hablar con las personas en la organización y escuchar para comprender.

1. Cuestionarios

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

Recolectar hechos de un gran número de personas.

Detectar un sentimiento generalizado.

Detectar problemas entre usuarios.

Cuantificar 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

Difíciles de preparar

Tipos de Preguntas

Abiertas

Son las que dejan abiertas todas las posibles opciones de respuesta.

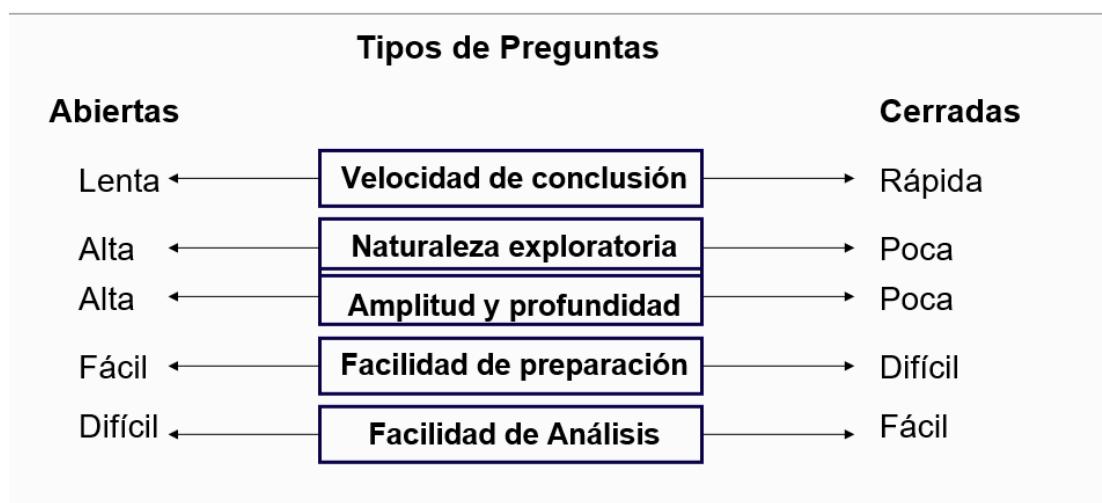
«Describa los problemas que experimenta en la actualidad con los informes de las salidas»,

«En su opinión, ¿qué tan útiles son los manuales de usuario para la aplicación de contabilidad del sistema actual?»

Cerradas

Limitan o cierran las opciones de respuestas disponibles

«¿Es útil el reporte que utiliza actualmente?» SI NO



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 o ciudades

Muchas personas involucradas

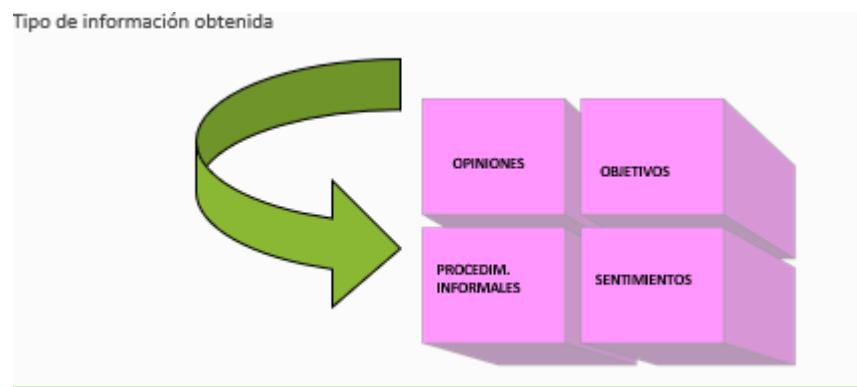
Clientes o usuarios

Queremos obtener opiniones generales

Queremos identificar problemas generales

2. 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.



Ventajas

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

Desventaja

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

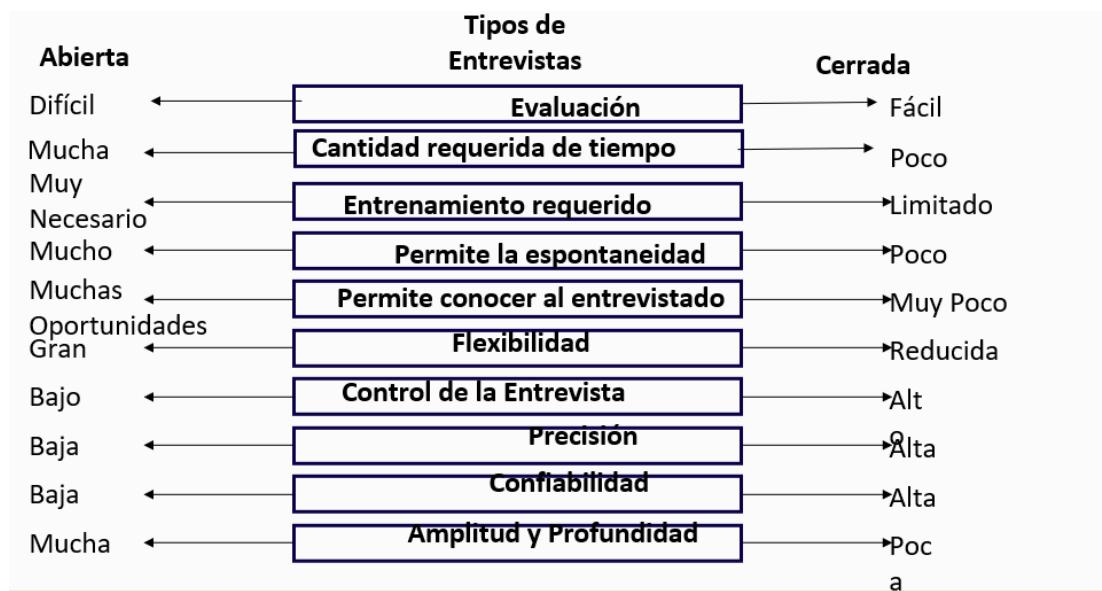
Tipos de entrevistas

Estructuradas (Cerradas)

El encuestador tiene un conjunto 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?

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

Cerradas

Las respuestas son directas, cortas o de selección específica
¿Quién recibe este informe?
¿Cuántas personas utilizan el sistema?

Ventajas

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

Desventajas

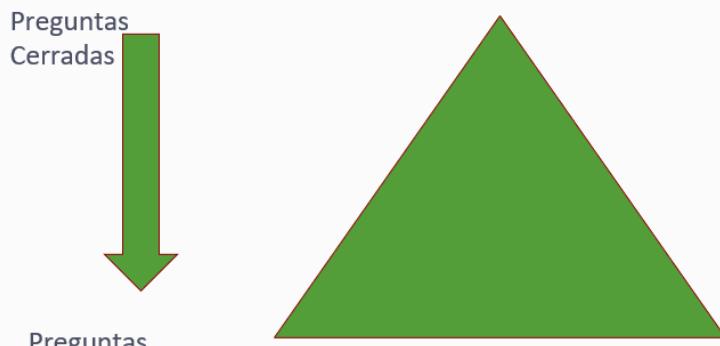
- Pueden aburrir al encuestado
- No se obtienen detalles

Sondeo

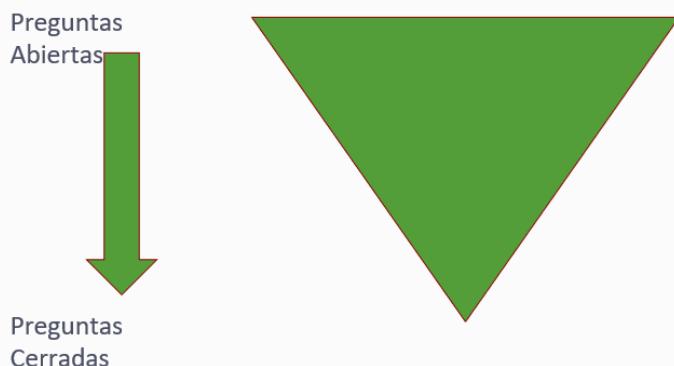
- Permite obtener más detalle sobre un tema puntual
- ¿Podría dar detalles sobre...?
- ¿Podría dar un ejemplo de...?

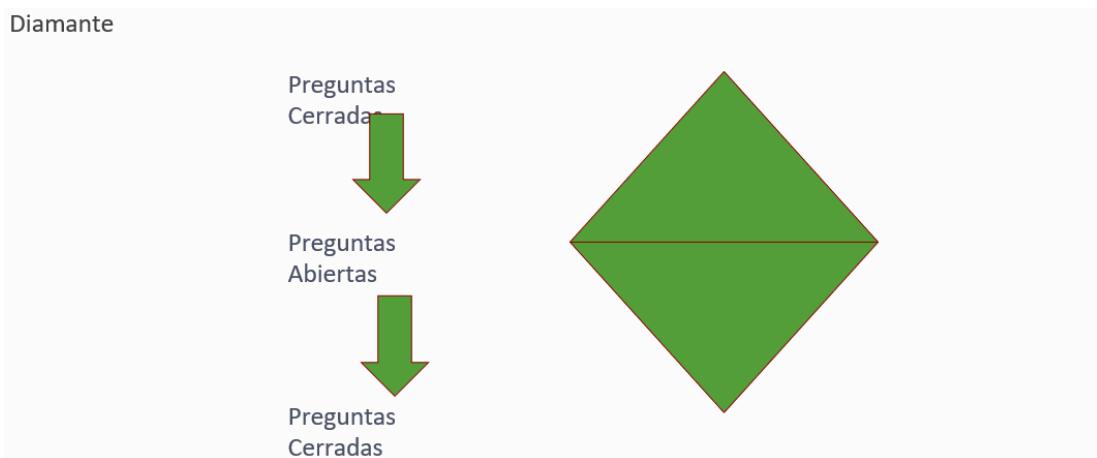
Organización de una entrevista

Piramidal (Inductivo)



Embudo (Deductivo)





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.

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

Respete el horario y los tiempos definidos

Si es en una sala de reunión 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

Debe

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

Evite

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

Guión

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.	<p>Objetivo Comienza la entrevista:</p> <ul style="list-style-type: none">• 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.	<p>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?</p>	
2 min.	<p>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.</p>	
2 min.	<p>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</p>	

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: <ul style="list-style-type: none"> • 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: 		

3. Planeación conjunta de Requerimientos

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

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

Ventajas

Ahorro de tiempo
Usuarios involucrados
Desarrollos creativos

Desventajas

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

Cómo planear las sesiones de JRP

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

Beneficios del JRP

JRP involucra activamente a los usuarios y la gerencia en el proyecto de desarrollo
JRP reduce el tiempo de la etapa de requerimientos
Si se incorporan prototipos, los mismos ya confirman el diseño del sistema

4. Lluvia de ideas

- 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.
- Los principios en que se basa esta técnica son:
 - Cuantas más ideas se sugieren, mejores resultados se conseguirán.
 - La producción de ideas en grupos puede ser más efectiva que la individual.
 - Las ideas de una persona 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

Requerimientos

Impacto de los errores en la etapa de requerimientos:

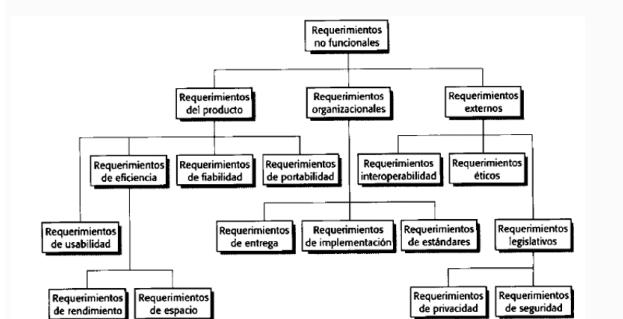
- El software resultante puede no satisfacer a los usuarios
- Las interpretaciones múltiples de los requerimientos puede causar desacuerdos entre clientes y desarrolladores
- Puede gastarse tiempo y dinero construyendo el sistema erróneo

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

Requerimientos no funcionales:

- Describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema
- Requerimiento 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 de desarrollador (entrega, implementación, estándares).
- Requerimientos externos
 - Interoperabilidad, legales, privacidad, seguridad, éticos.



Ingeniería de requerimientos

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

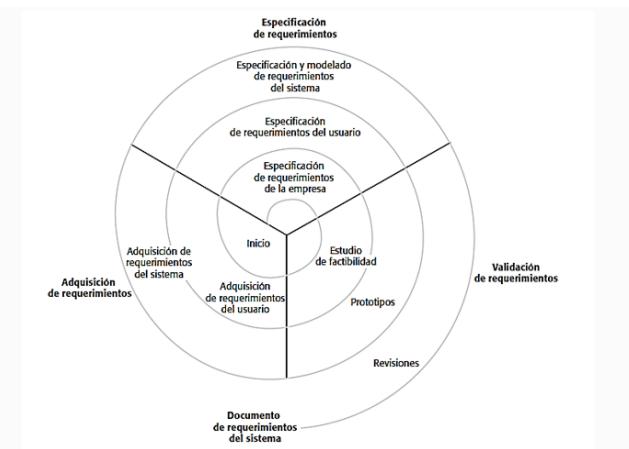
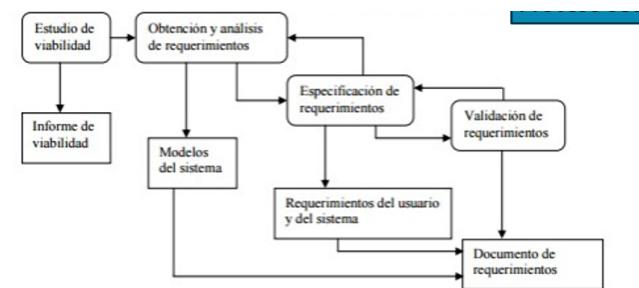
La ingeniería de requerimientos es una disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema

También, es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar.

"Ingeniería 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"

Importancia de la misma:

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



Estudio de viabilidad

Principalmente para sistemas nuevos

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

Responde las siguientes preguntas:

- ¿El sistema contribuye a los objetivos generales de la organización?(si no contribuye, entonces no tiene un valor real de negocio)
- ¿El sistema se puede implementar con la tecnología actual?
- ¿El sistema se puede implementar con las restricciones de costo y tiempo?
- ¿El sistema puede integrarse a otros que existen en la organización?

Una vez que se ha recopilado toda la información necesaria para contestar las preguntas anteriores se debería hablar con las fuentes de información para responder nuevas preguntas y luego se redacta el informe, donde debería hacerse una recomendación sobre si debe continuar o no el desarrollo.

Especificación de requerimientos

Propiedades de los requerimientos:

- Necesario: Su omisión provoca una deficiencia.
- Conciso: Fácil de leer y entender
- Completo: No necesita ampliarse
- Consistente: No contradictorio con otro
- No ambiguo: Tiene una sola 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

Documento de definición de requerimientos:

- Listado completo de todas las cosas que el cliente espera que haga el sistema compuesto

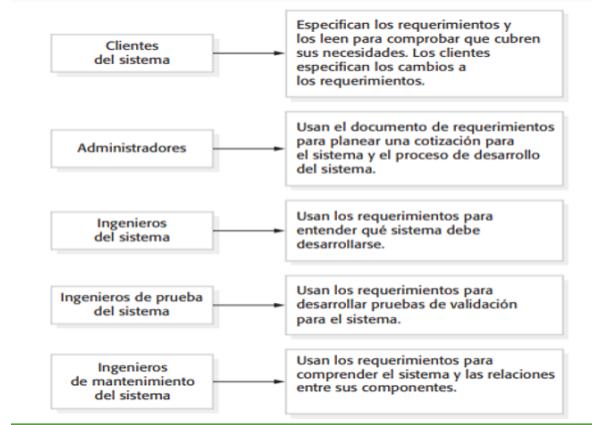
Documento de especificación de requerimientos:

- Definición en términos técnicos

Aspectos básicos de una especificación de requerimientos:

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

Usuarios de un documento de requerimientos:



Técnicas de especificación de requerimientos:

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

Validación de requerimientos

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario

Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar el sistema

Es importante, porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde

Validación: Al final del desarrollo de software para asegurar que el software cumple con los requerimientos

Verificación: 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 de software?

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

La validación de requerimientos comprende:

- 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 conjunto de pruebas)

Técnicas de validación

Pueden ser manuales o automatizadas

- Revisiones de requerimientos(Formales o informales)
 - Informales: Los desarrolladores deben tratar los requerimientos tanto 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
- Construcción de prototipos
- Generación de casos de prueba

Historias de usuario

Definición:

- Es una descripción corta y simple de un requerimiento de un sistema, que se describe en el lenguaje común del usuario y desde su perspectiva

Son utilizadas en las metodologías de desarrollo ágiles (SCRUM)

Acompañadas de las discusiones con los usuarios y pruebas de validación

Conceptos:

- Debe ser limitada, ésta debería poder escribirse sobre una nota adhesiva pequeña
- Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir mucho tiempo para administrarlos
- Permiten responder rápidamente 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:

Si bien el estilo puede ser libre, la historia de usuario debe responder tres preguntas:

¿Quién se beneficia?

¿Qué se quiere?

¿Cuál es su beneficio?

Esquema:

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

Ejemplos:

Como **usuario registrado** quiero **loguearme** para **empezar a utilizar la aplicación**

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
- Negociables: 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 del tiempo 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

Criterios de aceptación:

- Un criterio de aceptación es el criterio por el cual se define si una historia de usuario fue desarrollada según la expectativa del Product Manager/Owner y se si puede dar como hecha
- Deben ser definidos durante la etapa inicial antes de la codificación, acompañan la historia de usuario, porque complementan la historia de usuario y ayudan al equipo de desarrollo a entender mejor cómo se espera que el producto se comporte
- Los criterios de aceptación son utilizados para expresar 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.

Plantilla:

- **ID:** Identificador único de la historia expresado como texto generalmente de la forma <verbo> <sustantivo>
- **TÍTULO:** Descripción de la historia de la forma: **Como** <rol> **quiero** <algo> **para poder** <beneficio>.
- **REGLAS DE NEGOCIO:** Conjunto de reglas, normas, políticas, etc. que condicionan el modo de operación.

Reverso

- **CRITERIOS DE ACEPTACIÓN:**
- Escenario 1: título del criterio.
Dado <un contexto inicial>,
Cuando <ocurre un evento>,
Entonces <garantiza uno o más resultados>
- Escenario 2: título del criterio.
Dado <un contexto inicial>,
Cuando <ocurre un evento>,
Entonces <garantiza uno o más resultados>
-
- Escenario N: título del criterio.
Dado <un contexto inicial>,
Cuando <ocurre un evento>,
Entonces <garantiza uno o más resultados>

Ejemplo concreto:

Id: Matricular persona
Título: Como persona quiero matricularme al instituto para poder hacer los cursos
Reglas de Negocio:

- Un DNI no puede estar registrado dos veces con diferentes matrículas
- El pago debe realizarse con tarjeta de crédito

Reverso

Criterios de Aceptación (Matricular persona):

Escenario 1: Matriculación exitosa
Dado que el DNI 22.222.222 no se encuentra matriculado y las condiciones son las adecuadas para un pago exitoso.
Cuando la persona ingresa: "Juan Perez", DNI 22.222.222, dirección 7 #123 y presiona "Matricularse"
Entonces el sistema redirige al usuario al pago de matrícula con tarjeta de crédito, espera respuesta, matrícula a la persona y genera su número de matrícula.

Escenario 2: Matriculación fallida por matriculado existente
Dado que el DNI 12.123.123 se encuentra matriculado
Cuando la persona ingresa: "Ana Diaz", DNI 12.123.123 y dirección 51 #1321 y presiona "Matricularse"
Entonces el sistema informa que la persona ya se encuentra matriculada

Escenario 3: Matriculación fallida por error en pago
Dado que el DNI 22.222.222 no se encuentra matriculado y las condiciones no son las adecuadas para un pago exitoso.
Cuando la persona ingresa: "Juan Perez", DNI 22.222.222, dirección 7 #123 y presiona "Matricularse"
Entonces el sistema redirige al usuario al pago de matrícula con tarjeta de crédito, espera respuesta e informa que el pago no ha sido correcto por lo que no se pudo matricular a la persona.

Beneficios de historias de usuario:

- 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
- Permiten estimar fácilmente el esfuerzo de desarrollo

- Es ideal para proyectos con requisitos volátiles o no muy claros

Limitaciones:

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

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:

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:

- Las épicas suelen abarcar varios equipos de desarrollo
- Recogen normalmente muchas historias de usuario

- Los clientes determinan si eliminan o añaden historias de usuario dentro de cada épica
- Una épica sirve para estructurar los temas e iniciativas (objetivos)
- Las épicas también sirven para dar flexibilidad y agilidad al proyecto

Casos de uso

Definición:

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 CU facilita y alienta la participación de los usuarios.

Beneficios:

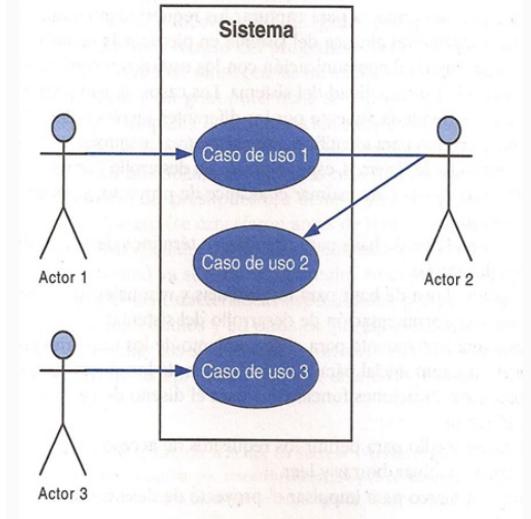
- Herramienta para capturar requerimientos funcionales.
- Descompone el alcance del sistema en piezas más manejables
- Medio de comunicación con los usuarios
- Utiliza lenguaje común y fácil de entender por las partes
- Permite estimar el alcance del proyecto y el esfuerzo a realizar
- Define una línea 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:

Diagramas de casos de uso:

Ilustra las interacciones entre el sistema y los actores

Ejemplo:



Caso de uso(Eipse azul):

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 externo o dispositivo externo que dispare un evento (sensor, reloj).

Escenarios(narración del CU)

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

Diagrama

Relaciones

Asociaciones

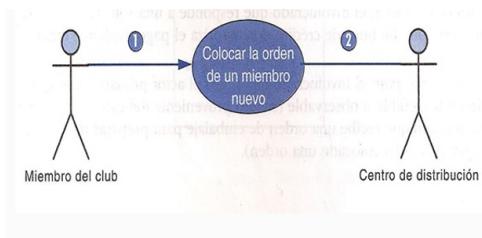
Extensiones

Uso

Herencia

Asociaciones

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



Extensiones

Un CU extiende la funcionalidad de otro CU.

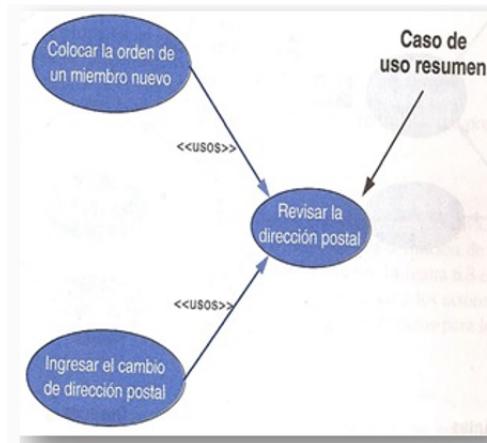
Un CU puede tener muchos CU extensiones

Los CU extensiones sólo son iniciados por otros CU



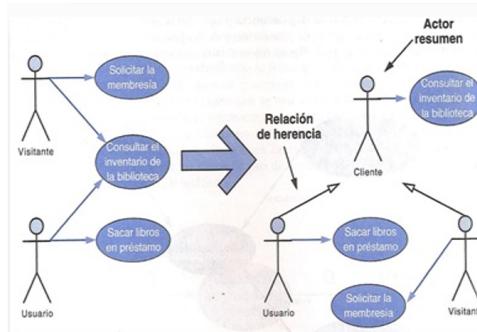
Uso o inclusión

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



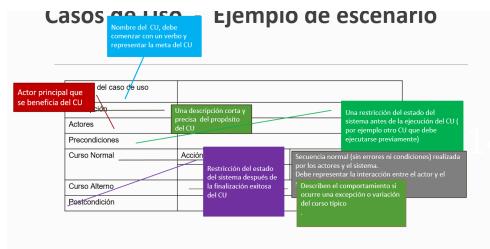
Escenarios

En el escenario se describen:

- Interacción del escenario
- Eventos alternativos

Ejemplo:

Nombre del caso de uso		
Descripción		
Actores		
Precondiciones		
Curso Normal	Acción del Actor	Acciones del Sistema
Curso Alterno		
Postcondición		



Proceso de modelado

Pasos:

- Identificar actores
- Identificar los CU para los requerimientos
- Construir el diagrama
- Realizar los escenarios

Identificar actores

¿Dónde buscar actores potenciales?

Documentación o manuales existentes

Minutas de reunión

Documentos de requerimientos

Responder a:

¿Quién o qué proporciona las entradas al sistema?

¿Quién o que recibe las salidas del sistema?

¿Se requieren interfaces con otros sistemas?

¿Quién mantendrá la información en el sistema?

Identificación de casos de uso para los requerimientos

- ¿Cuáles son las principales tareas del actor?
- ¿Qué información necesita el actor del sistema?
- ¿Qué información proporciona el actor al sistema?
- ¿Necesita el sistema informar al 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 pre-condiciones no deben representarse en los cursos alternativos, ya que al ser una pre-condición no va a ocurrir.
- Los “uses” deben ser accedidos por lo menos desde dos CU.

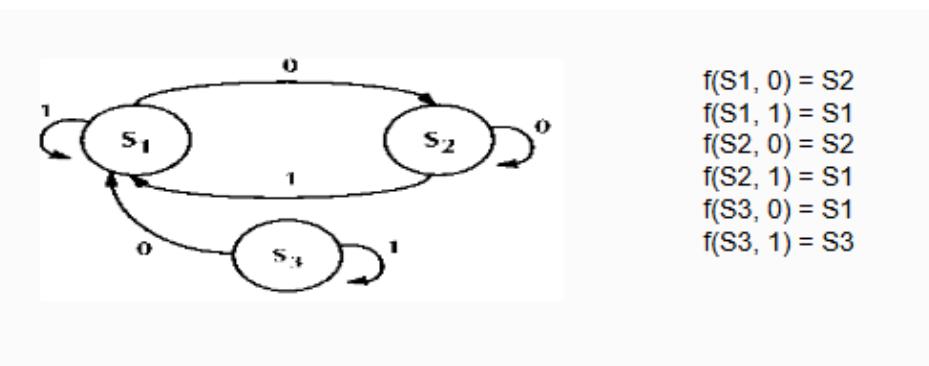
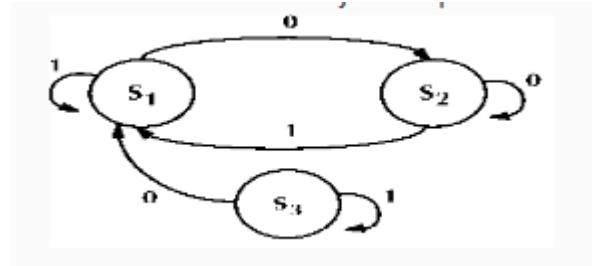
Diagramas de Transición y Estado

Máquinas de estado finito

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

$$f(S_i, C_j) = S_k$$

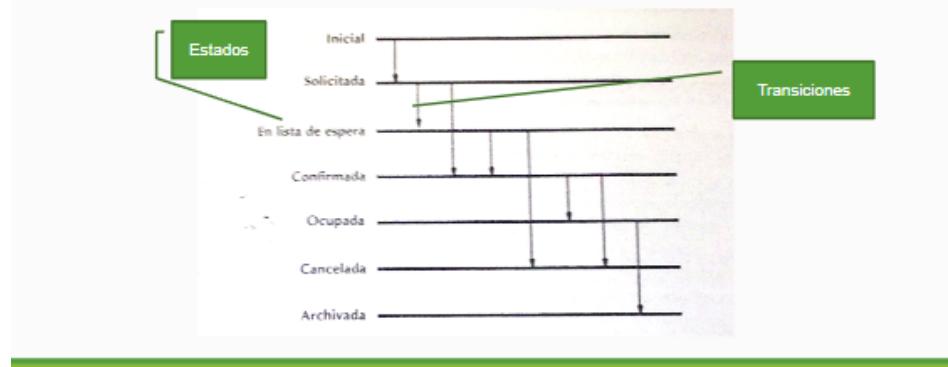
Al estar en el estado S_i , la ocurrencia de la condición C_j hace que el sistema cambie al estado S_k



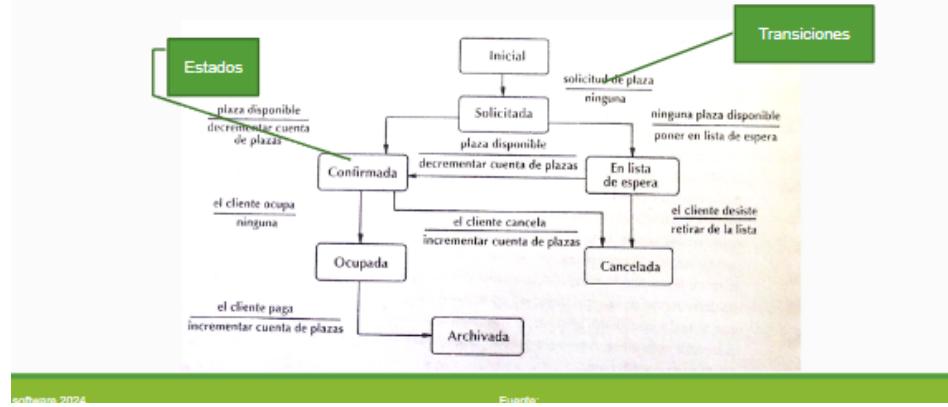
Definición formal

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

Representación en gráfico de persiana

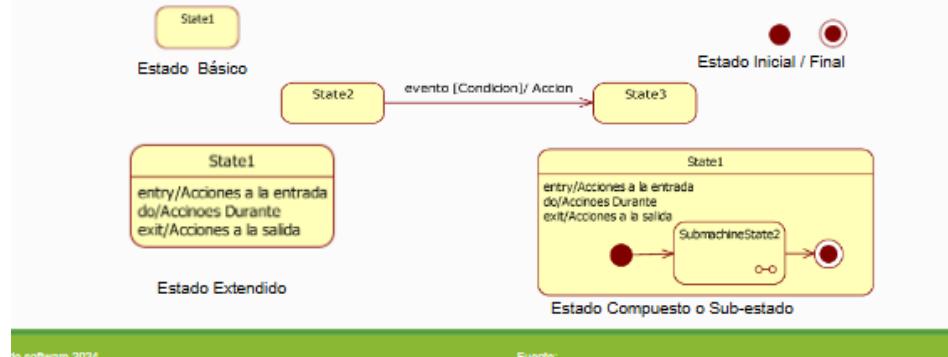


Representación en máquina de estado finito



Máquinas de Estado Finito

- Notación UML Diagrama de Transición y Estado (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

Transición

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



Evento: obligatorio

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

Acción: 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:
 - Se han definido todos los estados
 - Se pueden alcanzar todos los estados
 - Se pueden salir de todos los estados
 - En cada estado, el sistema responde a todas las condiciones posibles (normales y anormales)

Redes de Petri

Fueron inventadas por Carl Petri en la Universidad de Bonn, Alemania Occidental.

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

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

Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas (pueden operar a distintas velocidades, deben prevenir la modificación de datos compartidos o condiciones de bloqueo).

Pueden realizarse varias tareas en paralelo, pero son ejecutados en un orden impredecible.

Éstas NO son secuenciales.

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

Están compuestas por:

Eventos o acciones

Representados como transiciones

Estados o condiciones

Representados como lugares o sitios

Caso más simple:

$f(\text{EstadoA}, \text{Evento}) \rightarrow \text{EstadoS}$

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

$f(\text{EstadoA}, \text{Even1}, \text{Even2} \dots \text{EvenN}) \rightarrow \text{EstadoS}$

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

$f(\text{EstadoA}, \text{Even1}, \text{Even2} \dots \text{EvenN}) \rightarrow \text{Estado1}, \text{Estado2} \dots, \text{EstadoN}$

» Definición formal:

Una estructura de Red de Petri es una 4-upla

$$C = (P, T, I, O)$$

Lugares
 $P = \{P_1, P_2, \dots, P_m\}$

Transiciones
 $T = \{T_1, T_2, \dots, T_n\}$

Función de
entrada I
 $I: T \rightarrow P$

Función de
salida O
 $O: T \rightarrow P$

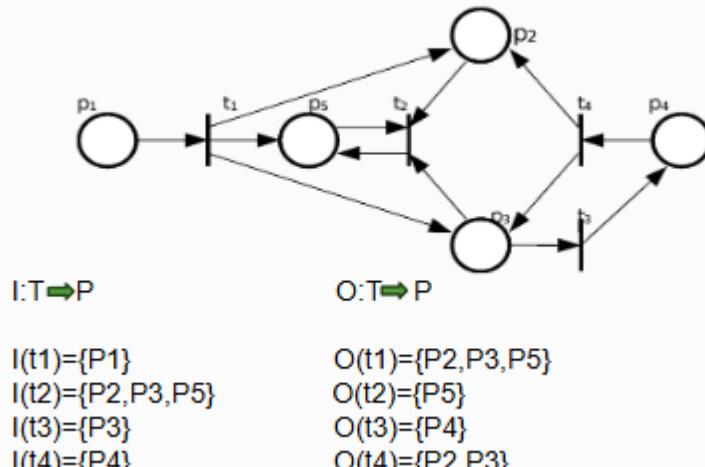
Multigrafo (de un nodo puede partir más de un arco), bipartito, dirigido

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

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

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

Funciones de entrada y de salida



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

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

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

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

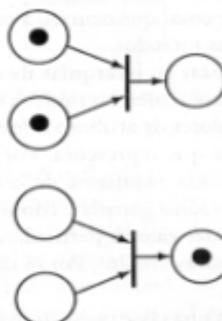
La ejecución de una Red de Petri se realiza disparando transiciones habilitadas.

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

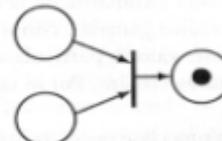
Disparar una transición habilitada implica remover tokens de los lugares de entrada y distribuir tokens en los lugares de salida (teniendo en cuenta la cantidad de arcos que llegan y la cantidad de arcos que salen de la transición).

» Transiciones

La transición está habilitada



La transición no está habilitada



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

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

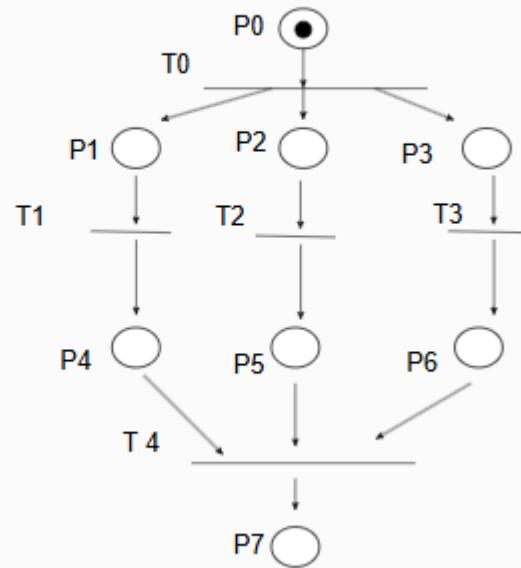
La ocurrencia de un evento está sujeta a que se den ciertas condiciones (pre) y al ocurrir el evento causa que se hagan verdaderas las post-condiciones.

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

La ejecución es NO DETERMINÍSTICA

Se acepta que el disparo de una transición es instantáneo .

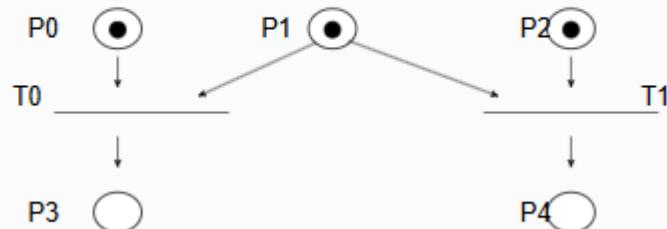
Redes de Petri: Paralelismo



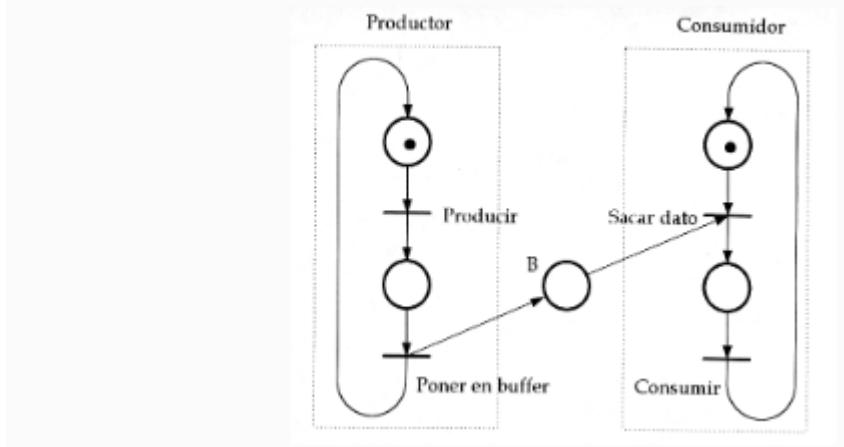
Sincronización

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

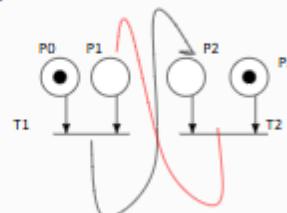
Expresión de exclusión mutua



Productor - Consumidor



Condición de bloqueo



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 RdP 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 RdP proporcionan herramientas de análisis para analizar los posibles estados a los que el sistema modelado pudiera alcanzar.

Tablas de Decisión

Es una herramienta que permite presentar de forma concisa las reglas lógicas que hay que utilizar para decidir acciones a ejecutar en función de las condiciones y la lógica de decisión de un problema específico.

Describe el sistema como un conjunto de:

- Posibles **condiciones** satisfechas en un momento dado
- **Reglas** para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y **acciones** a ser tomadas como un resultado

Construiremos tablas con:

- Condiciones simples y acciones simples
- Las condiciones tomas sólo valores Verdadero o Falso
- Hay 2^N Reglas donde N es la cantidad de condiciones

¿Cómo se llena la tabla?

A partir de un enunciado se debe

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

Especificaciones completas

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

Especificaciones redundantes

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

Especificaciones contradictorias

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

Redundancia y Contradicción

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

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

Redundante

Contradicitoria

Reducción de Complejidad (Redundancia)

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

El guion [—] significa que la condición 2 puede ser S o N, y que aun así se realiza la acción.

Condición 1:	S	S
Condición 2	S	N
Acción 1	X	X

Condición 1:	S
Condición 2	—
Acción 1	X

Reducción de Complejidad (Redundancia)
Álgebra de Boole

	Reglas			
	V	V	F	F
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	V	—	
V	F	—	
—		X	
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. Pueden ocurrir cuatro problemas principales al desarrollar tablas de decisión: que estén incompletas, que existan situaciones imposibles, contradicciones y redundancia.

Análisis estructurado

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

Diagrama de Flujo de Datos

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

Se utiliza un rectángulo para representar una entidad externa, esto es, un elemento del sistema y otro sistema que produce información para ser transformada por el software, o recibe información producida por el software

Cliente

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.



Un rectángulo abierto representa un almacén de datos



Desarrollo de DFDs

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

Pasos:

1. Redactar 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

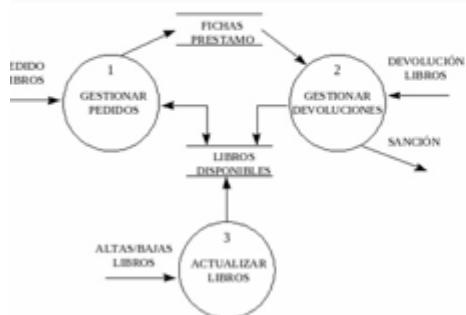
Diagrama de contexto



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

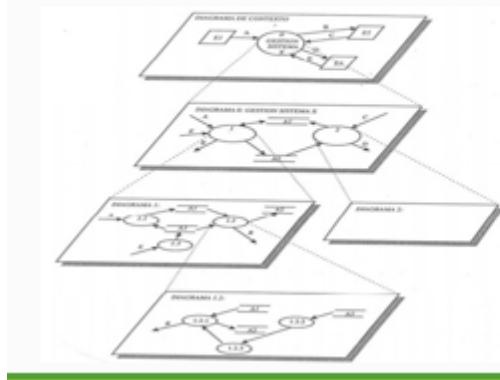
Nivel 0



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.

Modelos de Proceso

Proceso

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

¿Qué es un proceso de software?

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

Actividades funcionales de los procesos:

- Especificación del software
- Desarrollo del software
- Validación del software
- Evolución del software

Actividades fundamentales de los procesos

- Especificación del software
 - Consiste en el proceso de comprender y definir el sistema, así como la identificación de las restricciones sobre la operación y desarrollo del sistema.
 - También llamada, Ingeniería de Requerimientos
- Desarrollo del software
 - Corresponde al proceso de convertir una especificación del sistema en un sistema ejecutable.
 - Incluye los procesos de diseño y programación
 - Se crea una descripción de la estructura del software que se va a implementar, los modelos y estructuras de datos, las interfaces, etc.

- Validación del software
 - Se realiza para mostrar que un sistema cumple tanto con sus especificaciones 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

¿Qué es un modelo de proceso de software?

- Es una representación simplificada de un proceso de software que presenta una visión de ese proceso.
- Estos modelos pueden incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.
- Marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso (*norma ISO 12207-1*) [ISO/IEC, 1995]

Características

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

Ciclo de vida

Proceso que implica la construcción de un producto

Ciclo de vida del software

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

Modelos de proceso de software

Es una representación abstracta de un proceso del software

Modelo de proceso

Paradigma de software Todos ellos son términos equivalentes

Ciclo de vida del software

Modelos prescriptivos

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

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

Modelos descriptivos

Descripción en la forma que se realizan en la realidad

Ambos modelos deberían ser iguales

Modelos tradicionales

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

- Clásico, lineal o en cascada
- Modelo en V
- Basado en prototipos

Modelos evolutivos

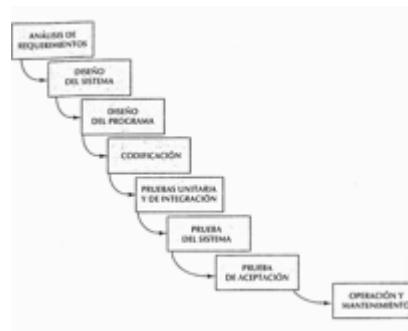
Son modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo

- En espiral (desarrollo por fases)

- Evolutivo (desarrollo por fases)
- Incremental

Modelo en cascada

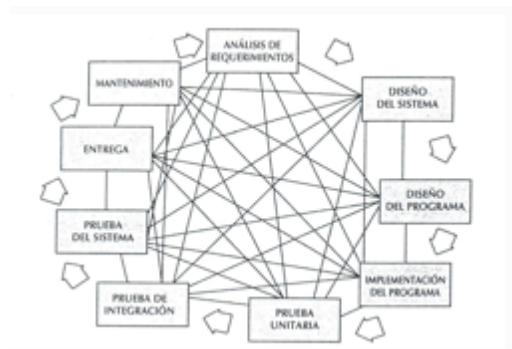
- Las etapas se representan cayendo en cascada
- Cada etapa de desarrollo se debe completar antes que comience la siguiente
- Útil para diagramar lo que se necesita hacer
- Su simplicidad hace que sea fácil explicarlo a los clientes



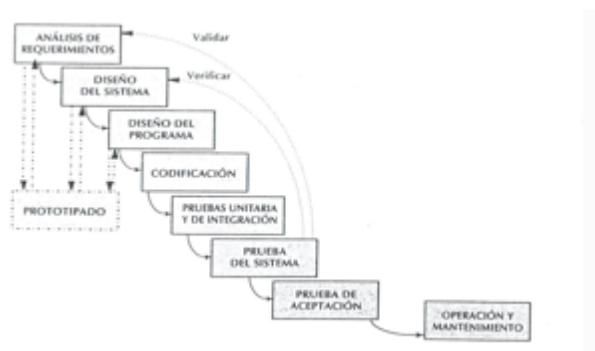
Dificultades:

- No existen resultados concretos hasta que todo este 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

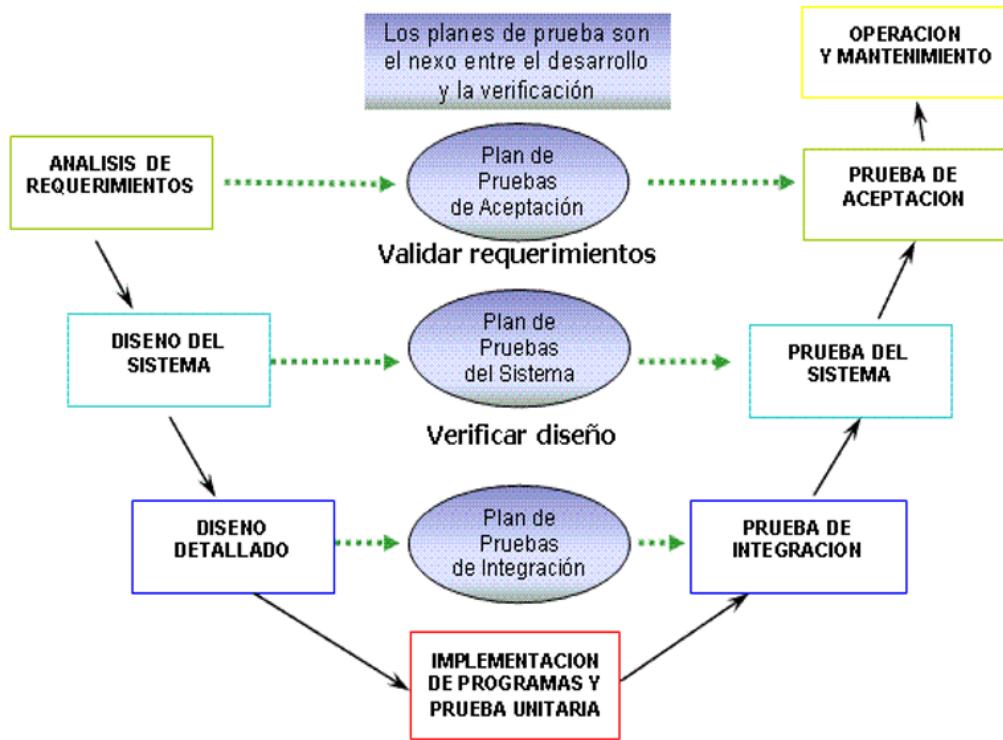


Modelo en cascada con prototipo



Modelo en V

- Demuestra cómo se relacionan las actividades de prueba con las de análisis de 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 que, si se encuentran problemas durante la verificación y validación, entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema.



Modelo de prototipos

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

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 construyen rápidamente para comprender o aclarar aspectos y asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución.
- Descartables: No tiene funcionalidad. Se utilizan herramientas de modelado

	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:

- 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 tiene dificultad al tratar con los modelos gráficos para modelar los requerimientos y el comportamiento
- Si se enfatiza el aspecto de la interfaz humana

Para asegurar el éxito:

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

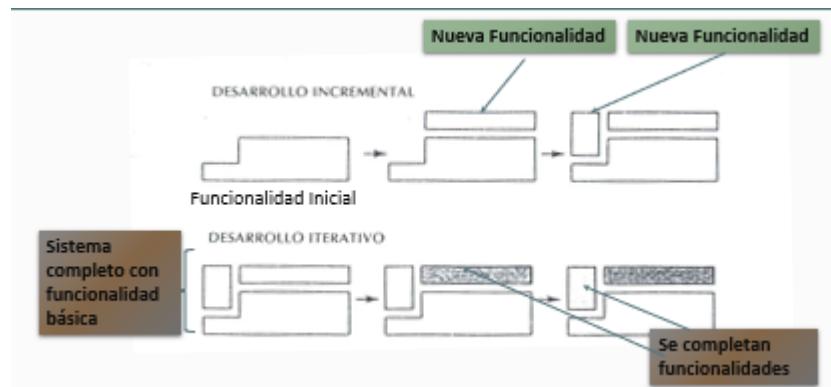
Modelo de desarrollo por fases

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



Tipos:

- Incremental: El sistema es particionado en subsistema 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 gestión del riesgo

Trata de mejorar los ciclos de vida clásicos y prototipos.

Incorpora objetivos de calidad

Elimina errores y alternativas no atractivas al comienzo

Permite iteraciones, vuelta atrás y finalizaciones rápidas

Cada ciclo empieza identificando:

Los objetivos de la porción correspondiente

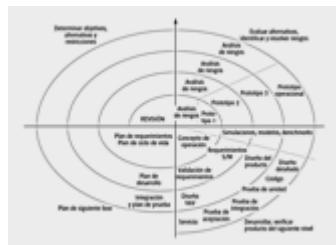
Las alternativas

Restricciones

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

Cada ciclo en 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 modelo de desarrollo
4. Planeación: El proyecto se revisa y se toma decisiones para la siguiente fase



Metodologías ágiles

En los años 80 y principios de los 90, existía una opinión general de que la mejor forma de obtener un mejor software era a través de una planificación cuidadosa del proyecto, la utilización de métodos de análisis y diseño, y procesos de desarrollo de software controlados y rigurosos.

En general se realizaban sistemas críticos, desarrollados por grandes equipos, a menudo dispersos geográficamente.

Sin embargo, cuando este enfoque fue aplicado a sistemas de negocio pequeños y de tamaño medio, el esfuerzo invertido era grande, y cuando cambiaban los requerimientos, se hacía esencial rehacer el trabajo.

Del descontento nacieron las 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 lleva asociado un marcado énfasis en el control del procesos, 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 del tiempo

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

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

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.

"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.

(incremental)es una estrategia programada y en etapas, en la que las diferentes partes del sistema se desarrollan en diferentes momentos o a diferentes velocidades, y se integran a medida que se completan

En resumen...

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" [Fowler], adaptándose además rápidamente al cambio de los

proyectos.

Ese enfoque ha sido utilizado desde hace más de dos décadas por un grupo de profesionales de software.

Valores:



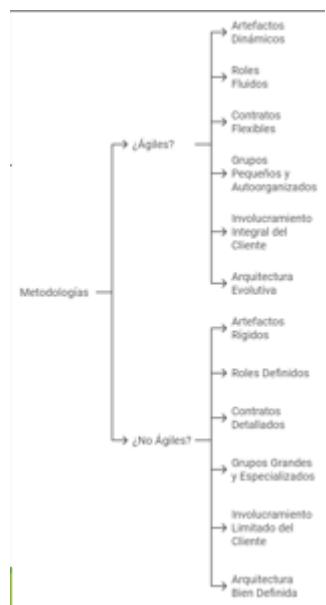
Es importante comprender que los valores escritos en colores son conceptos de las metodologías agiles sin perder de vista que intentan superar

Una buena manera de interpretar el manifiesto, es asumir que éste define preferencias, no alternativas

Principios:

1. Nuestra mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valioso
2. Los cambios de requerimientos son bienvenidos, aún tardíos, en el desarrollo. Los procesos Ágiles capturan los cambios para que el cliente obtenga ventajas competitivas
3. 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

4. Usuarios y desarrolladores deben trabajar juntos durante todo el proceso
5. Construir proyectos alrededor de motivaciones individuales
6. Darles el ambiente y el soporte que ellos necesitan y confiar en el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores
7. El software que funciona es la medida clave de progreso
8. Los procesos ágiles promueven un desarrollo sostenible. Los stakeholders, desarrolladores y usuarios deberían ser capaces de mantener un paso constantemente indefinidamente
9. Atención continua a la excelencia técnica y buen diseño incrementa la agilidad
10. Simplicidad (arte de maximizar al cantidad de trabajo no dado) es esencial
11. Las mejores arquitecturas, requerimientos y diseños surgen de la propia organización de los equipos
12. A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento en consecuencia



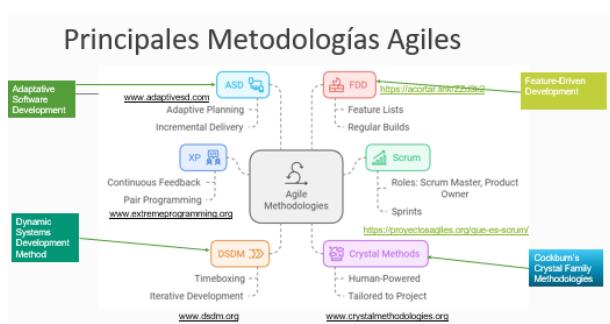
Desventajas:

En la práctica, los principios que subyacen a los métodos ágiles son a veces difíciles de cumplir:

- 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 definen y continúen. Para ellas, resulta difícil moverse hacia un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo

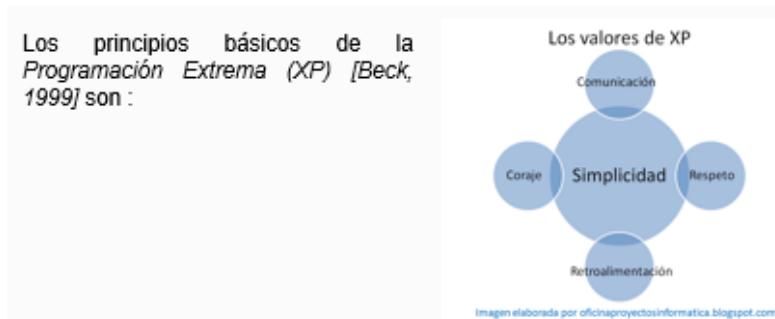
Por lo general, el documento de requerimientos del software forma parte del contrato entre el cliente y el proveedor. Como en los métodos ágiles se minimiza la documentación, suele ser complejo reglamentarlo.

La mayoría de los libros que describen los métodos ágiles y las experiencias con éstos 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 la evolución de los sistemas de software existentes. Al no existir documentación se complejizaría.



eXtreme Programming

- Es una disciplina de desarrollo de software basado en los valores de la sencillez, la comunicación, la retroalimentación, la valentía y el respeto
 - Su acción consiste en llevar a todo el equipo reunido en la presencia de prácticas simples, con suficiente información para ajustar las prácticas a su situación particular
1. Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras
 2. Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión
 3. Programación en parejas
 4. Frecuente integración del equipo de programación con el cliente o usuario
 5. Corrección de todos los errores antes de nueva funcionalidad
 6. Refactorización del código
 7. Propiedad del código compartida
 8. Simplicidad en el código



Características:

- Historia de usuario
- Roles
- Proceso
- Práctica

Roles:

- Programador (Programmer)
 - Responsable de decisión técnicas
 - Responsable de construir el sistema
 - Sin distinción entre analistas, diseñadores o condiciones
 - En XP, los programadores diseñan, programan y realizan las pruebas
- Jefe de proyecto (Manager)
 - Organiza y guía las reuniones
 - Asegura condiciones adecuadas para el proyecto
- Cliente (Customer)
 - Es parte del equipo
 - Determina qué construir y cuándo
 - Establece las pruebas funcionales
- Entrenador (Coach)
 - Responsable del proceso
 - Tiende a estar en un segundo plano a medida que el equipo madura
- Encargado de pruebas (Tester)
 - Ayuda al cliente con las pruebas funcionales
 - Se asegura de que las pruebas funcionales se superan
- Rastreador (Tracker)
 - Metric Man
 - Observar sin molestar
 - Conservar datos históricos

Proceso:

El ciclo de vida consiste en:

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



Exploración:

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

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

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.

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

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

Mantenimiento:

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

Muerte:

- Es cuando el cliente no tiene más historias para ser incluidas en el sistema
- Se genera la documentación final del sistema y no 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 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 su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios
- Programación de a pares:
 - Todo el código de producción es escrito por dos programadores en una máquina
- 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 y compartido
- Semana de 40-horas:
 - Se deben trabajar un máximo de 40 horas por semana
 - El trabajo extra desmotiva al equipo
 - Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final ser entregados con retraso. En lugar de esto se debe realizar el juego de la 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
- Estándares 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

Scrum se define como “una manera simple de manejar problemas complejos”, proporcionando un paradigma de trabajo que soporta la innovación y permite que equipos auto-organizados entreguen resultados de alta calidad en tiempos cortos.

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

Principios:

- Eliminar desperdicio: no generar artefactos, ni perder el tiempo haciendo cosas que no sumen 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 práctica no se puede tener el conocimiento antes de empezar el desarrollo
- Diferir las decisiones: Tomar las decisiones en el momento adecuado, esperar hasta ese momento, ya que uno tiene más información a media que va pasando el tiempo. Si se puede esperar mejor
- 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 motiva 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:

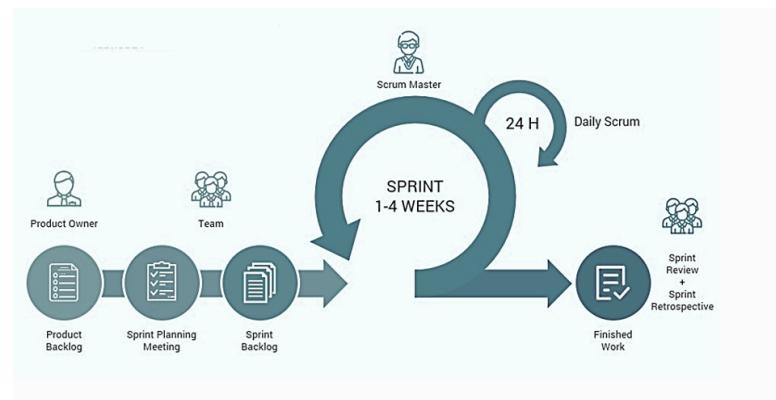
- El Product Owner (Propietario) conoce y marca las prioridades del proyecto o producto
- El Scrum Master (Jefe) es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda aparecer. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas
- El Scrum Team (Equipo) son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner
- Los Usuarios o Cliente, son los beneficiarios finales del producto, y 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 característica más importante que es la funcionalidad se encuentra ordenada por un orden de prioridad
- Sprint Backlog: Es la lista que contiene toda la funcionalidad que el equipo se comprometió a desarrollar en un Sprint determinado
- Burndown Chart: Muestra un acumulativo de trabajo hecho, día-a-día
- Entre otros...

Proceso

Este solapamiento de fases se puede asemejar a un scrum de rugby, en el cual todos los jugadores (o roles, en nuestro caso), trabajan juntos para lograr un objetivo



Scrum es iterativo e incremental

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

El nombre Scrum se debe a que durante los sprints, lo que serían las fases de desarrollo, se solapan, de manera que no es un proceso de cascada por cada iteración, si no que tenemos todas éstas etapas juntas que se ejecutan una y otra vez, hasta que se crea suficiente

¿Cuando usar Scrum?

Scrum está pensado para ser aplicado en proyectos en donde el "caos" es una constante, aquellos proyectos en los que tenemos requerimientos dinámicos y que tenemos que implementar tecnología de punta

Esos proyectos difíciles, que con los enfoques tradicionales se hace imposible llegar a buen puerto

KANBAN

Es un enfoque Lean de desarrollo de software ágil

Literalmente Kanban es una palabra japonesa que significa "tarjeta visual". En Toyota, Kanban es el término que se utiliza para el sistema de señalización visual y física que une todo el sistema de producción Lean

Kanban 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 de entrega

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. 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.

Esto la convierte en una herramienta simple, pero a la vez muy potente

Esta metodología ha ganado popularidad en el mantenimiento de multiproyectos de diferentes características.

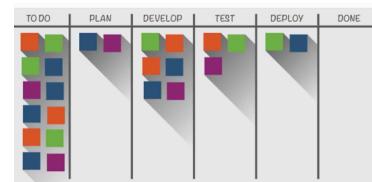
Kanban trata de administrar el flujo de trabajo. Inicialmente, no reemplaza nada que la organización haga, simplemente impulsa el cambio.

A diferencia de otras metodologías no define ningún rol y no prescribe una secuencia de pasos definidos para llevar a cabo el desarrollo del proyecto. Su implementación es sencilla (3 prácticas)

Es habitual combinar en las prácticas de otras metodologías ágiles los tableros Kanban que son la herramienta ágil por excelencia de esta metodología, donde se visualizan las tareas a realizar, las realizadas y lo pendiente. Estos, son una herramienta para el control visual del sistema que ayuda a un equipo a visualizar explícitamente el proceso de desarrollo.

A primera vista, el tablero Kanban se asemeja a un proceso en cascada. Pero en la práctica, evita explícitamente los problemas de un proceso de estas características mediante la aplicación de lotes pequeños de trabajo y el desarrollo incremental.

Está compuesto por una serie de columnas que representan los diversos estados que atraviesa un requerimiento durante el proceso de desarrollo. Las tarjetas se mueven de un estado a otro mostrando la evolución hasta que haya sido aceptado por el cliente. Cada columna tendrá un límite para el trabajo en curso. Cuando una tarea es completada en una columna, esta se mueve a la siguiente. De esta forma, se crea un espacio libre en la columna actual representando capacidad de trabajo disponible. El equipo entonces toma una tarea terminada desde el estado anterior y la desplaza hacia el estado actual, ya que hay capacidad para procesarla.

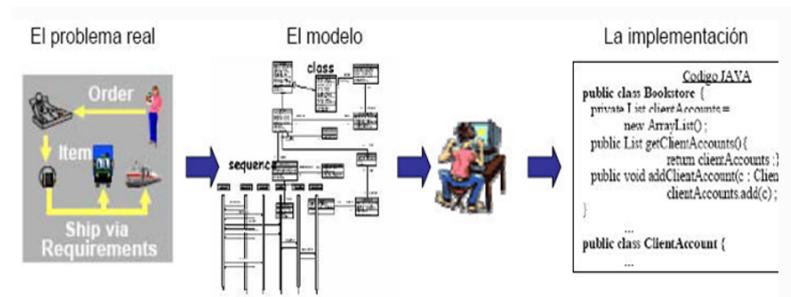


El Desarrollo de Software Basado en Modelados. (MBD)

Hacia fines de los 70' De Marco introdujo el concepto de desarrollo de software basado en modelos. Destacó que la construcción de un sistema de software debe ser precedida por la construcción de un modelo, tal como se realiza en otros sistemas ingenieriles.

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. (Abstracción de elementos del problema, comunicación, negociación con el usuario)

Construcción de un Sistema de Software



Desarrollo de Software Dirigido por Modelos. (MDD)

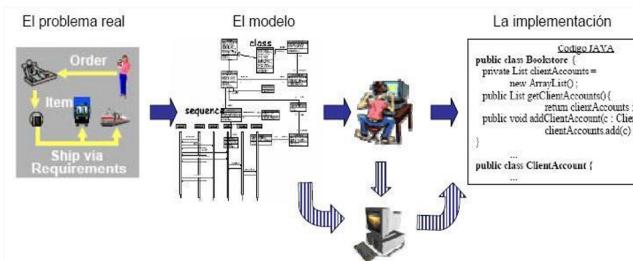
El adjetivo <<dirigido>> en MDD, a diferencia de <<basado>>, enfatiza que este paradigma asigna a los modelos un rol central y activo: son al menos tan importantes como el código fuente.

Model Driven Development (MDD) promueve enfatizar los siguientes puntos claves:

- Mayor nivel de abstracción en la especificación tanto del problema a resolver como la de la solución correspondiente
- Aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.

- Uso de estándares industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica
- Los modelos son los conductores primarios en todos los aspectos del desarrollo del software

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



Modelos de MDD:

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 acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica"

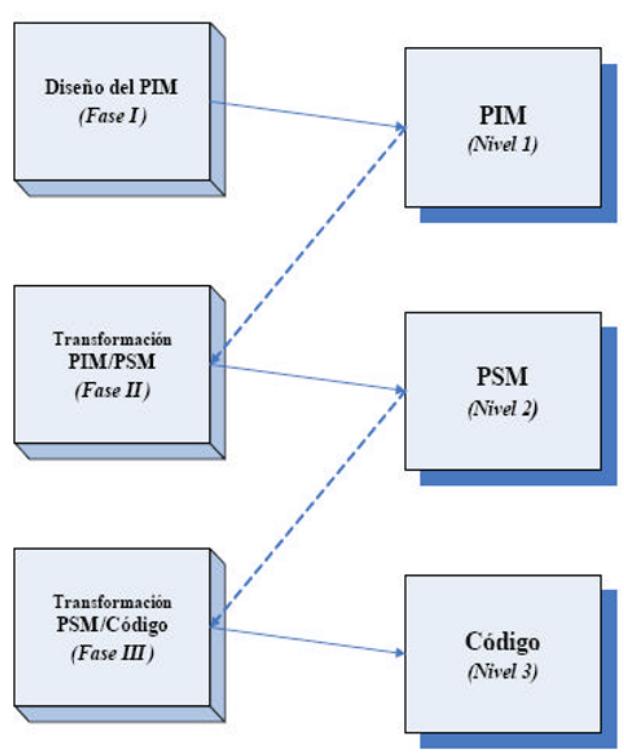
Transformación de modelos: "Especifica el proceso de conversión de un modelo en otro modelo del mismo sistema"

- Cada transformación incluye (Al menos):
 - Un PIM

- Un Modelo de la Plataforma
- Una transformación
- Un PSM

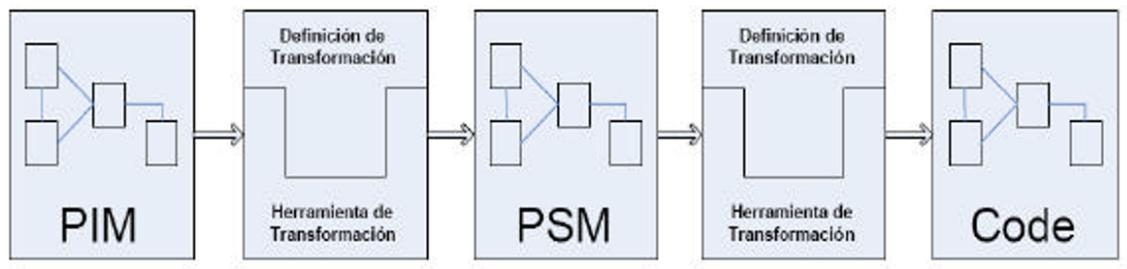
Los tres pasos principales en el proceso de desarrollo MDD

- Platform Independent Model (PIM): "Un modelo de un sistema que no contiene información acerca de la plataforma o la tecnología que es usada para implementarla"
- Platform Special Model: "Un modelo de un sistema que incluye información acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica"
- Transformación de modelos: "Especifica el proceso de conversión de un modelo en otro modelo del mismo."

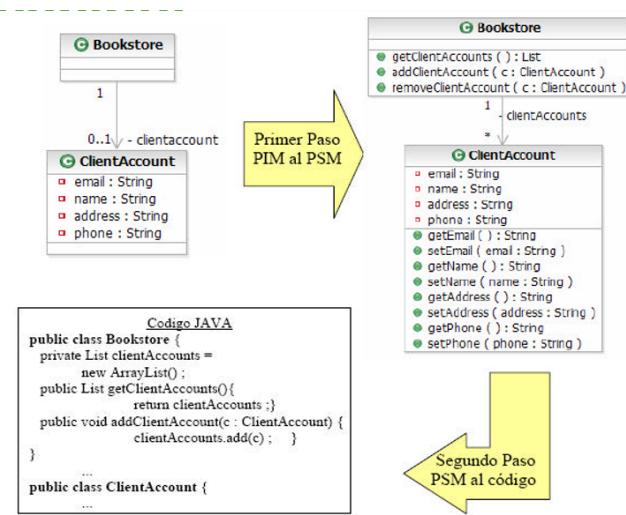


¿Qué es una transformación?

En general, se puede decir que una definición de 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 patrón MDD es normalmente utilizando sucesivas veces para producir una sucesión de transformaciones



Orígenes de MDD

MDD es la evolución natural de la ingeniería de software basada en modelos enriquecida mediante el agregado de transformaciones automáticas entre modelos.

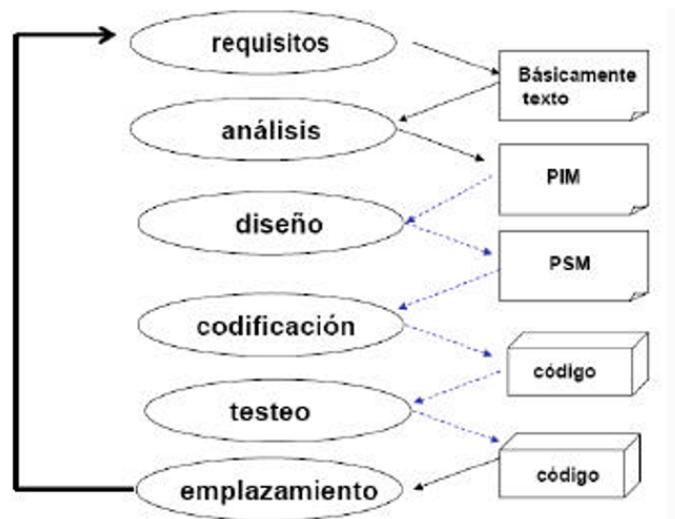
Si bien MDD define un nuevo paradigma para el desarrollo de software, sus principios fundamentales no constituyen realmente nuevas ideas sino que son reformulaciones y asociaciones de ideas anteriores.

La técnica de transformación se asemeja al proceso de abstracción y refinamiento presentado por Dijkstra

Beneficios de MDD

- Incremento en la productividad (modelos y transformaciones)
- Adaptación a los cambios tecnológicos
- Adaptación a los cambios de requisitos
- Consistencia (automatización)
- Re-uso (modelos y transformaciones)
- Mejoras en la comunicación con los usuarios y la comunicación entre los 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

Ciclo de vida del software dirigido por modelos



Calidad

Definición:

- Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor "Esta tela es de buena calidad"
- Buena calidad, superioridad o excelencia. "La calidad del vino de Jerez ha conquistado los mercados"
- Carácter, genio, índole
- Condición o requisito que se pone en un contrato
- Estado de una persona, naturaleza, edad y demás circunstancias y condiciones que se requieren para un cargo o dignidad

Se ve una serie de definiciones relacionadas, la más destacable es la primera donde se habla de "*propiedades que pueden ser juzgadas*", de ahí se desprende que la calidad es un término totalmente subjetivo, que va a depender del juicio de la persona que intervenga en la evaluación.

A lo largo de la historia se han desarrollado filosofías o culturas de calidad, de las cuales algunas han sobresalido porque han tenido resultados satisfactorios. A los que realizaron estas filosofías se los ha llamado Maestros o Gurús de la Calidad.



¿Qué es la calidad?

- Calidad es un concepto manejado con bastante frecuencia en la actualidad, pero a su vez, su significado es percibido de distintas maneras.

- Al hablar de bienes y/o servicios de calidad, la gente se refiere normalmente a bienes de lujo o excelentes, con precios elevados
- Su significado sigue siendo ambiguo y muchas veces su uso depende de lo que cada uno entiende por calidad, por lo cual es importante comienza a unificar su definición

Criterios erróneos

- Un producto de calidad es un producto de lujo
- La calidad es intangible y por lo tanto no mensurable
- Los problemas son originados por los trabajadores de producción
- La calidad se origina en el Depto de calidad

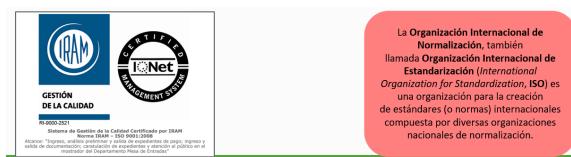
La calidad es relativa a las personas, a su edad, a las circunstancias de trabajo, el tiempo...

- El tiempo varía las percepciones.

Las principales normas internacionales la definen como:

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

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



Sistemas de Información

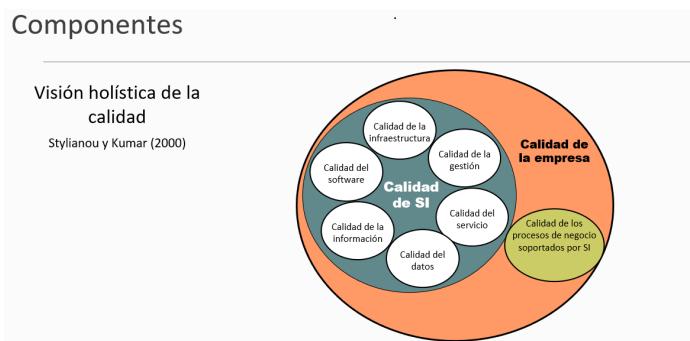
Es el conjunto de personas, datos, procesos y tecnología de información que interactúan para recopilar, procesar, guardar y proporcionar como salida la información necesaria para brindar soporte a una organización (Whitten y Bentley 2008)

Un sistema de información abarca más que el aspecto meramente computacional, pues no sólo hemos de tener en cuenta estas herramientas, sino también el modo de organizar dichas herramientas y de obtener la información necesaria para el correcto funcionamiento de la empresa

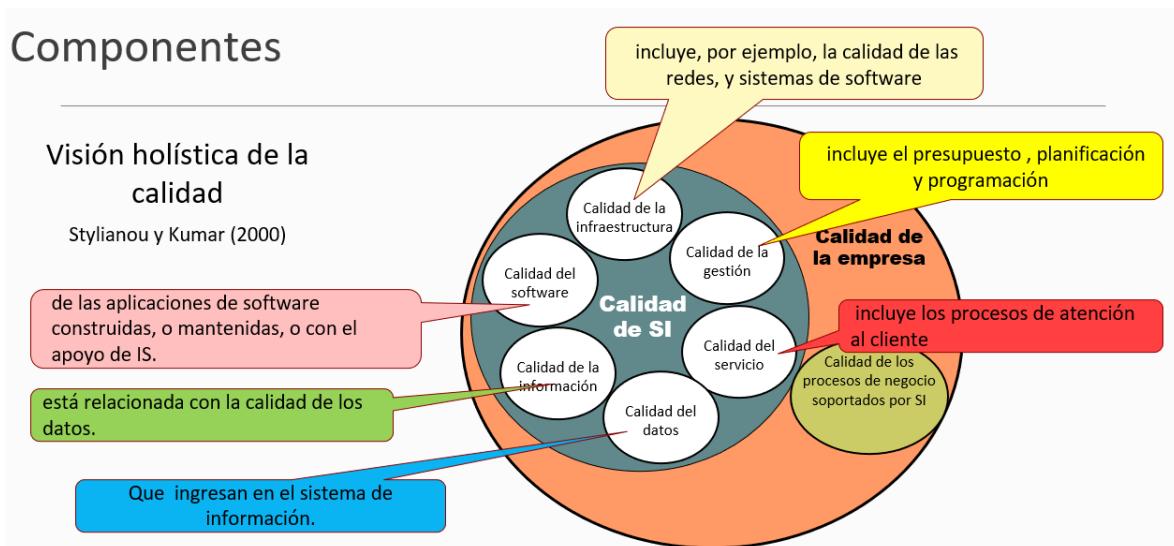
Calidad de los Sistemas de información

- La importancia de los sistemas de información (SI) en la actualidad hace necesario que las empresas de tecnología hagan mucho más hincapié en los estándares (o normas) de calidad
- Staylianou y Kumar plantean que se debe apreciar la calidad desde un todo, donde cada parte que la componen debe tener su análisis de calidad

Componentes



Componentes



Calidad de software

- La calidad del software se ha mejorado significativamente en estos últimos años, en particular por una mayor conciencia de la importancia de la gestión de la calidad y la adopción de técnicas de gestión de la calidad para desarrollo en la industria del software
- Se divide en
 - Calidad del producto obtenido
 - Calidad del proceso de desarrollo

Son dependientes

Calidad del Producto y Proceso

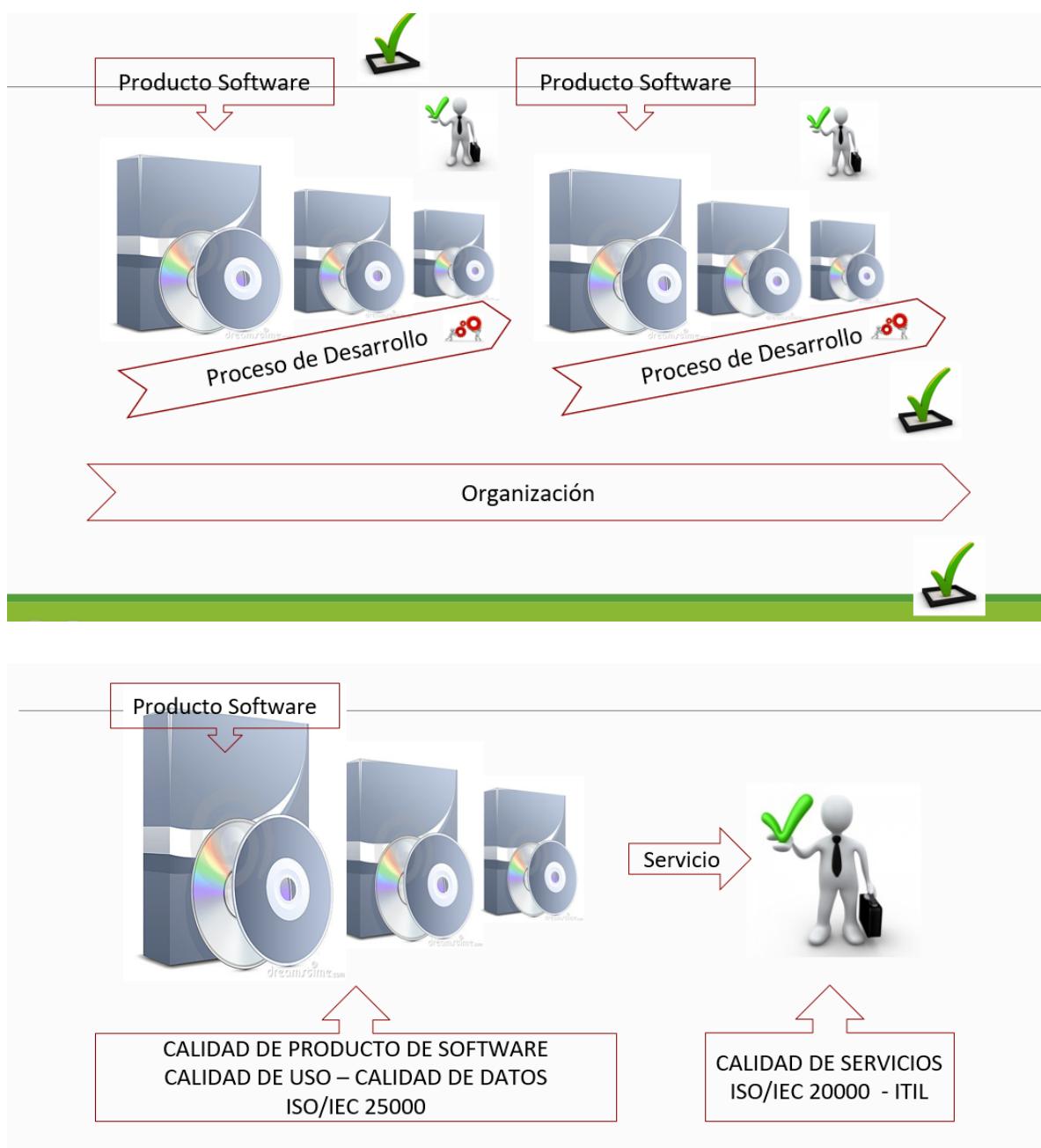
- Producto
 - La estandarización del producto define las propiedades que debe satisfacer el producto de software resultante

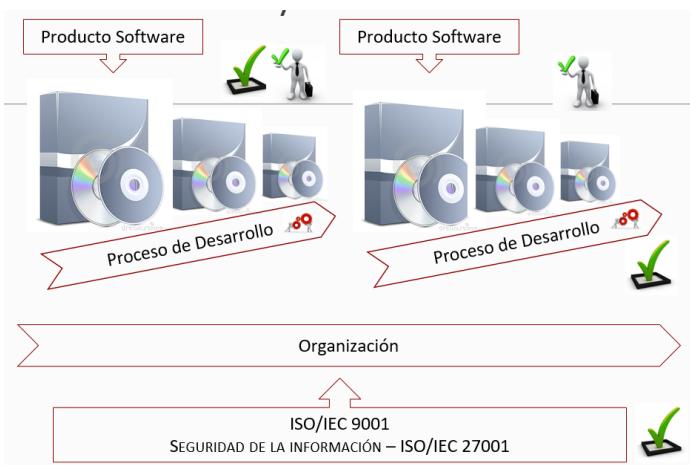
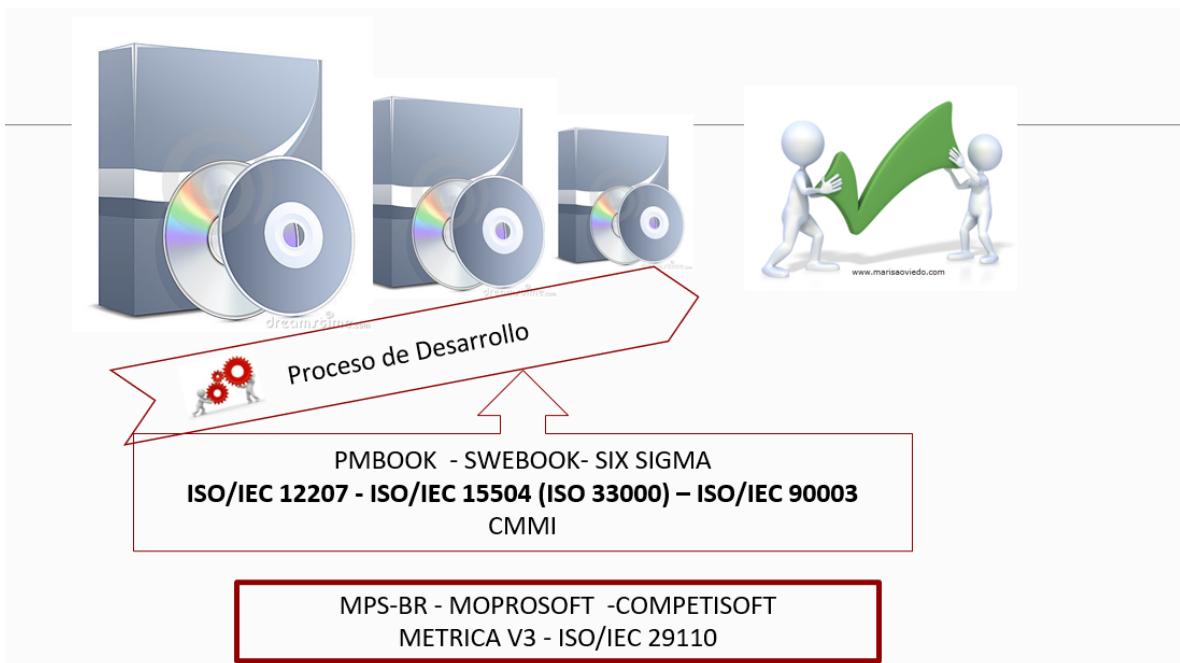
- Proceso

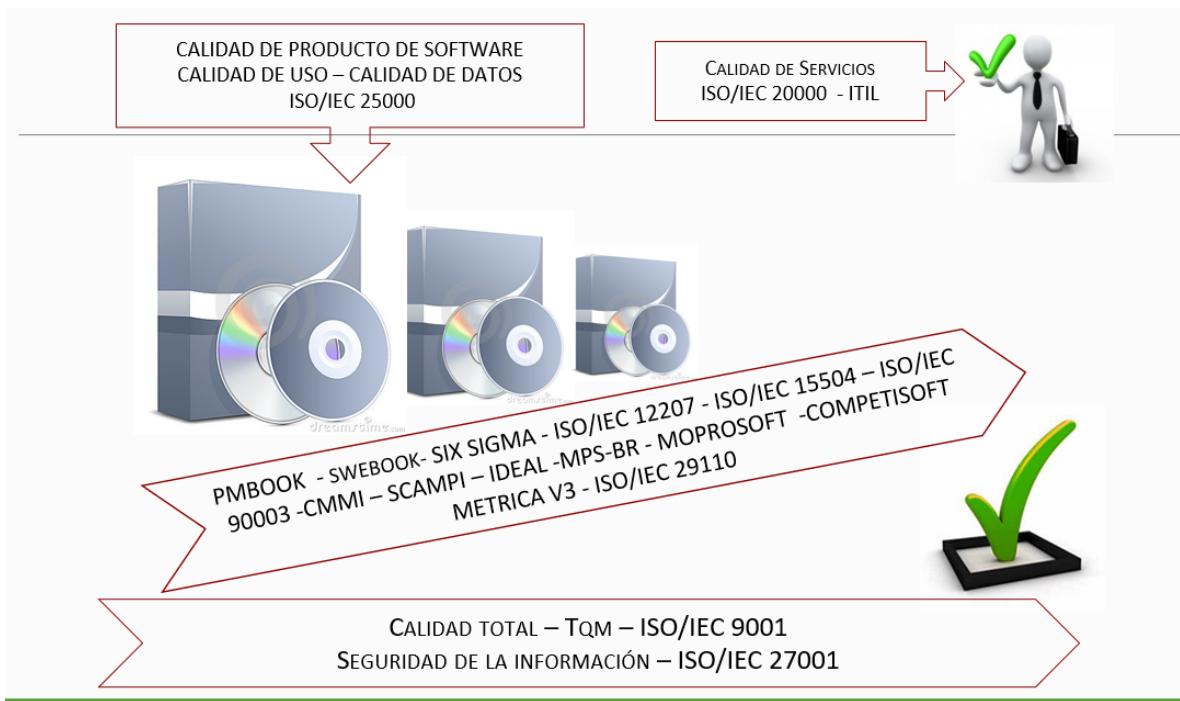
- La estandarización del proceso define la manera de desarrollar el producto de software

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

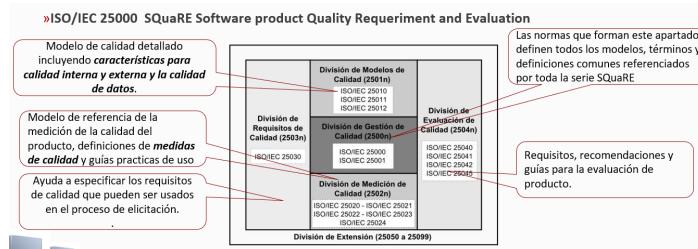
Clasificación de Normas y Modelos de Calidad







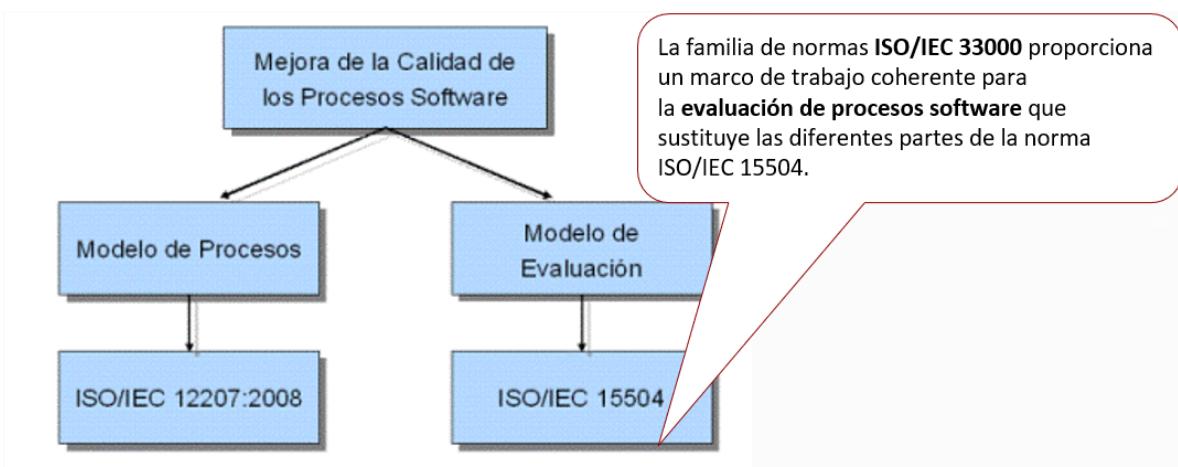
Norma/Modelo de Calidad SQuaRE ISO/IEC 25000

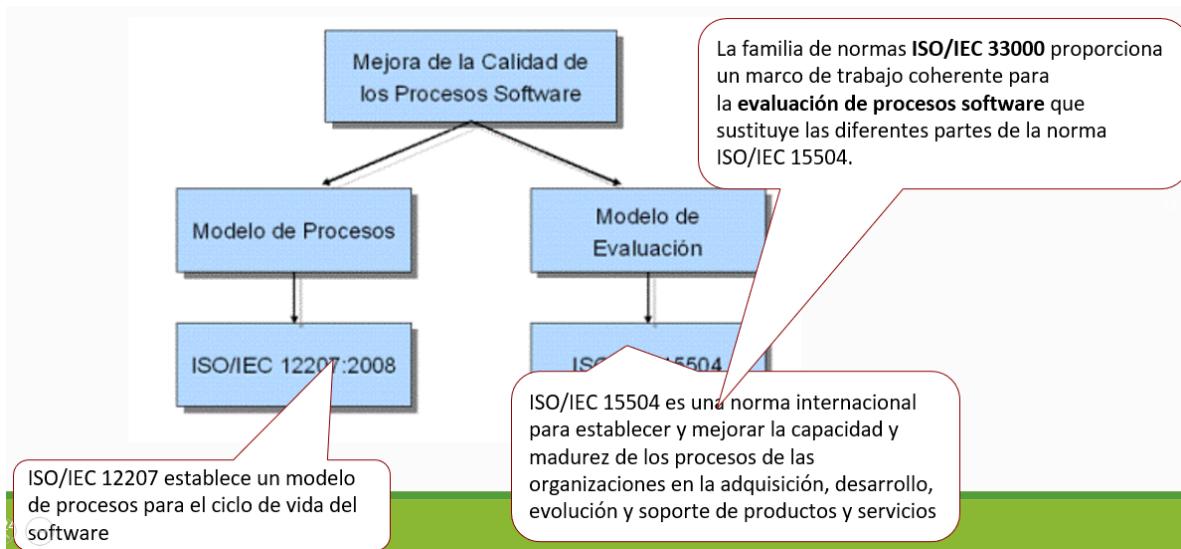


Norma/Modelo de Calidad SQuaRE ISO/IEC 25010- Características



Norma/Modelo de Calidad Software





CMM (1993) – CMMI (2000)

- Es un modelo de evaluación de los procesos de una organización (Capability Maturity Model)
- Fue desarrollado inicialmente para los procesos de desarrollo de software por la Universidad Carnegie-Mellon para el Software Engineering Institute (SEI)
- 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

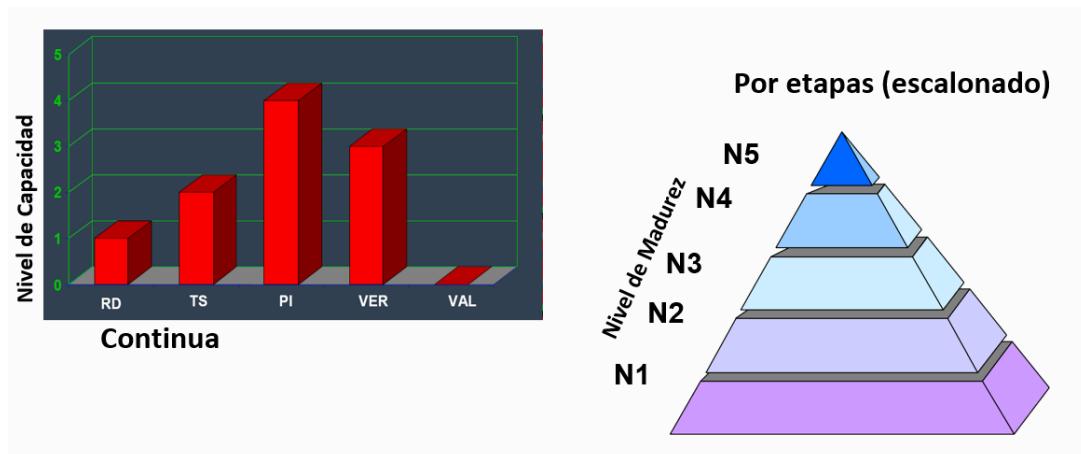
CMMI

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

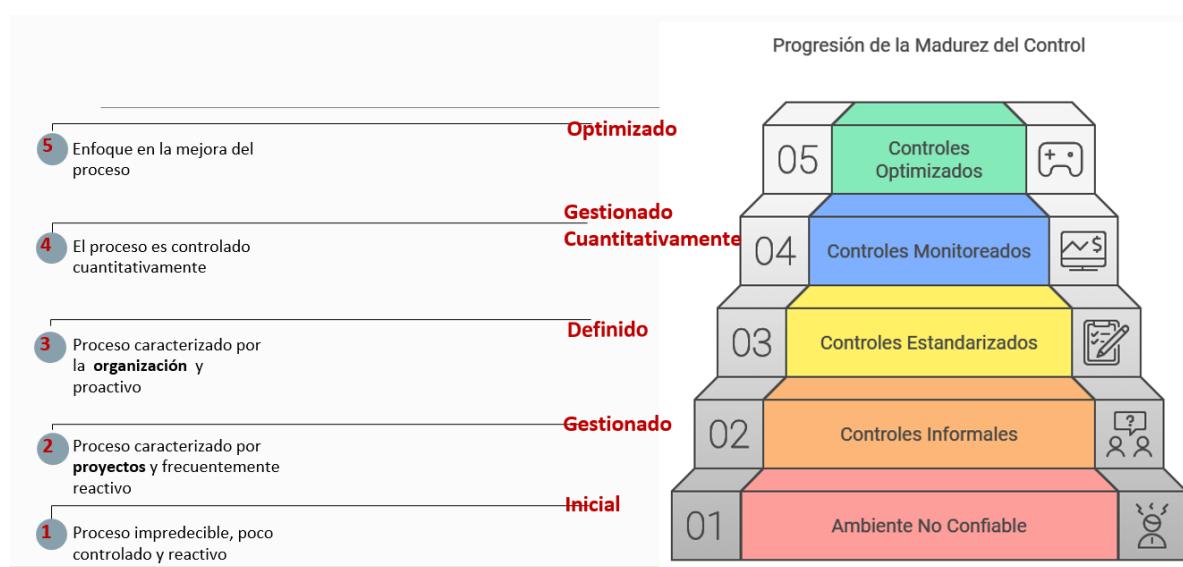
- Escalonado
 - Centra su foco en la madurez de la organización. Igual que CMM
- Continuo
 - Enfoca las actividades de mejora y evaluación en la capacidad de los diferentes procesos. Presenta 6 niveles de capacidad. Los

niveles de capacidad indican qué tan bien se desempeña la organización en un área de proceso individual

Representaciones



Niveles de madurez



ISO - International Organization for Standardization

La familia ISO 9000 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.

Familias de las ISO 9000

- **ISO – 9001:2015** - Quality management system – Requirements
 - Norma publicada por ISO en el año 2015.
- **IRAM – ISO 9001:2015** – Sistema de gestión de la calidad – Requisitos
 - Norma publicada por ISO y traducida por IRAM.
- **ISO 90003:2004**
 - Basada ISO 9001:2000 (se espera una actualización para el próximo año)
 - Directrices para la interpretación en el proceso de software
 - Proporciona una guía para identificar la evidencias dentro del proceso de software para satisfacer los requisitos de la ISO 9001

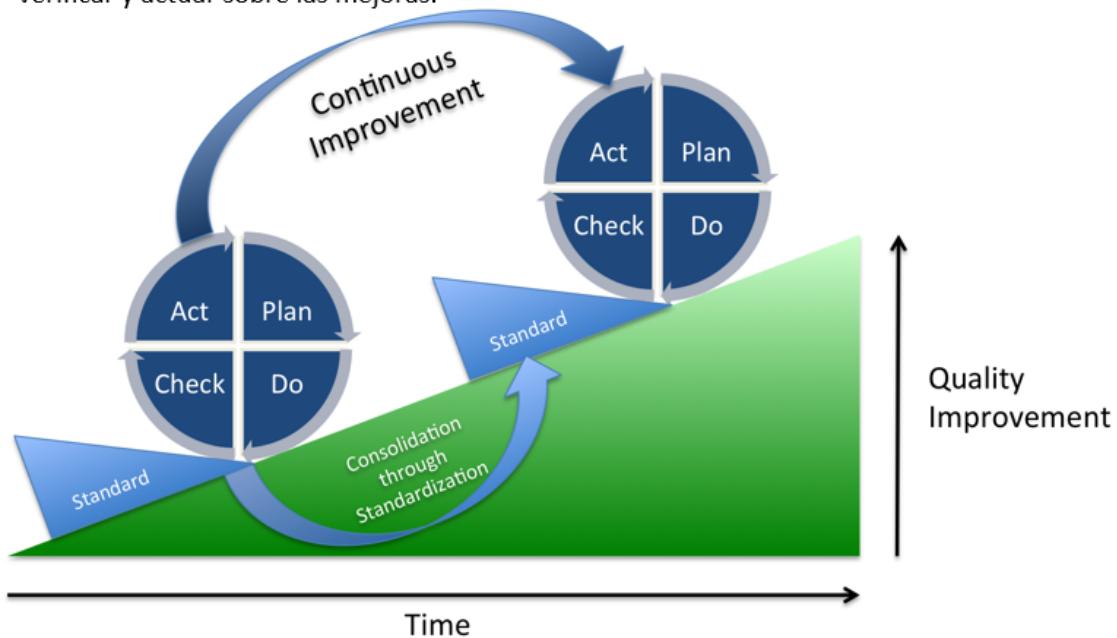
BENEFICIOS DE TRABAJAR CON UN SISTEMA DE GESTIÓN DE CALIDAD (SGC) ISO 9001



SGC -IRAM - ISO 9001

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 iterativo para planificar, implementar, verificar y actuar sobre las mejoras.



Resumiendo...

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		ISO/ IEC 15504	ISO/IEC 12208 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 15594 (reemplazada)	Es una norma internacional para establecer y mejorar la

		por ISO 33000)	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 claridad
	Se evalúa la calidad mediante		
		ISO/IEC 90003	Proporciona una guía sobre cómo aplicar la ISO 9001 en procesos de software
Calidad de procesos/Servicios en general	Se evalúa mediante	CMMI	Proporciona un arco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora, e implementar esas mejoras. Se utiliza para organizaciones desarrolladas de software de medianas a grandes dimensiones
		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, lo importante a la hora de competir en los mercados globales