INGENIERÍA DE SOFTWARE

TEORÍA

Facultad de Informática

UNLP

2023

ÍNDICE

Conceptos de Ingeniería de Software

NATURALEZA DEL SOFTWARE

El software es un producto, junto con toda la documentación asociada a dicho producto.

DEFINICIÓN DE SOFTWARE

¿Qué es?

Un software son Instrucciones (programas de cómputo), procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación (Definición formal de la IEEE)

CARACTERÍSTICAS DEL SOFTWARE

- Es un elemento lógico: no hay algo físico que se pueda tener en cuenta, no hay una concepción material física de ese producto)

- El software se desarrolla, no se fabrica.

- El software no se desgasta: No hay un desgaste material físico)

- No sigue la curva clásica de envejecimiento: La tasa de fallos del software no aumenta con el tiempo de operación, sino que aumenta cuando se realizan cambios

TIPOS DE PRODUCTO DE SOFTWARE

- **Genéricos:**

  Sistemas aislados producidos por organizaciones desarrolladoras de software y que se venden en un mercado abierto (Por ejemplo el paquete de Office o de Adobe)

- **Personalizados:**

Sistemas requeridos por un cliente en particular.

RETOS A LA HORA DEL DESARROLLO DE SOFTWARE

- Software heredado:

  Software que ya fue desarrollado que debe ser adaptado o actualizado.

- Enfrentar el futuro:

  Por ejemplo computación ubicua (IOT) y la WWW (World Wide Web)

Para hacer frente a estos retos se requiere una organización, y ahí es donde toma sentido la idea de Ingeniería de Software.

¿QUÉ ES LA INGENIERÍA DE SOFTWARE?

Definición de la cátedra:

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

Definición recursiva de la IEEE:

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

Métodos sistemáticos cuantificables: Se deben tener datos medibles estadísticos

Dentro de los tiempos y costos estimados: Se debe definir un plan con estos 2 parámetros

Para el "Desarrollo, operación y mantenimiento": Comprende todo el ciclo de vida del software, desde la planificación, hasta la retirada de servicio

PARTICIPANTES EN EL DESARROLLO DEL SOFTWARE

- Gerentes ejecutivos (dueños del producto)
- Gerente de proyecto (lider de equipo)
- Profesionales especializados
- Clientes
- Usuarios finales

CARACTERÍSTICAS DE UN INGENIERO DE SOFTWARE

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

Pero además, los aspectos humanos es lo que lo harán un ingeniero efectivo.

Tener un sentido de responsabilidad individual, aguda conciencia de las necesidades del equipo, atención al detalle entre otros.

RESPONSABILIDAD PROFESIONAL Y ÉTICA

La ingeniería de Software se desarrolla en un marco económico, social y legal. Los ingenieros de software deben aceptar responsabilidades más amplias que las responsabilidades técnicas.

No debe utilizar su capacidad y habilidades de forma deshonesta o de forma que deshonre la profesión.

Técnicas de comunicación

Al iniciar un proyecto, la primera actividad es saber lo que el usuario quiere, cómo lo quiere, cuando y porqué... Para eso, tenemos que comunicarnos.

LA COMUNICACIÓN

La comunicación es la base para la obtención de las necesidades del cliente.

Es la principal fuente de error, ya que cada persona entiende algo diferente dependiendo de su contexto, formación, perspectiva, etc.

Al hablar de necesidades, en términos más técnicos, estamos hablando de **requerimientos**

REQUERIMIENTOS

Definición de la cátedra:

Un requerimiento (o requisito) es una característica del sistema o una descripción de algo que el sistema debe ser capaz de hacer con el objeto de satisfacer el propósito del sistema.

Definición de la IEEE (Standard-610):

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

2- Condición o capacidad que debe satisfacer o poseer un sistema o un componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento previamente firmado.

3- Representación documentada de una condición o capacidad como en 1 o 2

FUENTES DE REQUERIMIENTOS

- Documentación
- Stakeholders
- Especificación de sistemas similares

STAKEHOLDERS

El término stakeholder se utiliza para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente.

Entre los stakeholders se encuentran:

- Usuarios finales
- Ingenieros
- Gerentes
- Expertos del dominio
- Miembros del equipo
- Patrocinador del proyecto
- Clientes
- Usuarios finales

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 mismos pero que influyen en los requerimientos de algún modo.

- **Punto de vista del dominio:** Representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

Elicitación de Requerimientos

ELICITACIÓN

Es el proceso de adquirir ("eliciting") todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema.

Sus objetivos son:

- Conocer en profundidad el dominio del problema
- Poder comunicarse con el cliente y usuarios para entender sus necesidades
- Conocer el sistema actual (manual o ya informatizado)
- Identificar las necesidades, tanto explícitas como implícitas, de clientes y usuarios y sus expectativas sobre el sistema a desarrollar

La elicitación de requerimientos es una actividad principalmente de carácter social, mucho más que tecnológica. Los problemas que se plantean, por tanto, son de naturaleza psicológica y social, más que técnicos.

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.
- No tomar decisiones por no poder prever las consecuencias, no entender las alternativas o no tener una visión global.
- Cultura y vocabulario diferentes.
- Intereses distintos en el sistema a desarrollar.
- Medios de comunicación inadecuados (diagramas que no entienden los clientes y usuarios).
- Conflictos personales o políticos.

PROBLEMAS EN LA ELICITACIÓN

Limitaciones cognitivas (del desarrollador)

- No conocer el dominio del problema.
- Hacer suposiciones sobre el dominio del problema.
- Hacer suposiciones sobre aspectos tecnológicos.
- Hacer simplificaciones excesivas.

Conducta humana

- Conflictos y ambigüedades en los roles de los participantes.
- Pasividad de clientes, usuarios o ingenieros de requisitos.
- Temor a que el nuevo sistema lo deje sin trabajo.

Técnicos

- Complejidad del dominio del problema.
- Complejidad de los requisitos.
- Múltiples fuentes de requisitos.
- Fuentes de información poco claras.

TÉCNICAS DE ELICITACIÓN

Recopilación de información:

Métodos discretos

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

Métodos interactivos

1. Cuestionarios.
2. Entrevistas.
3. Planeación conjunta de Requerimientos (JRP o JAD).
4. Lluvia de Ideas - Brainstorming

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

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

Muestreo de la documentación, los formularios y los datos existentes

Recolección de hechos a partir de la documentación existente.

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

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

Permiten conocer el historial que origina el proyecto

Investigación y visitas al sitio

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

Observación del ambiente de trabajo

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

Lineamientos de la observación:

- Determinar quién y cuándo será observado.
- Obtener el permiso de la persona y explicar el porqué será observado.
- Mantener 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.

RECOPILACIÓN DE INFORMACIÓN: MÉTODOS INTERACTIVOS

Hay métodos interactivos que pueden usarse para obtener los requerimientos directamente 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.

Cada uno cuenta con su propio proceso establecido

- Cuestionarios.
- Entrevistas.
- Planeación conjunta de Requerimientos (JRP o JAD).
- Lluvia de Ideas - Brainstorming.

Planeación Conjunta de Requerimientos (JRP)

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

- Requiere de extenso entrenamiento
- Reduce el tiempo de exploración de requisitos
- Amplia participación de los integrantes
- Se trabaja sobre lo que se va generando

Alguna bibliografía la menciona como JAD (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

Lluvia De Ideas (Brainstorming)

Técnica para generar ideas al alentar a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo sin ningún análisis hasta que se hayan agotado las ideas.

- **Se promueve el desarrollo de ideas creativas para obtener soluciones.**
- 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.

La lluvia de ideas:

- Tiene 3 fases:

  Descubrir hechos, Producir ideas, Descubrir soluciones

- Es clave para resolver la falta de consenso entre usuarios
- Es útil combinarlo con la toma de decisiones
- Ayuda a entender el dominio del problema
- Encara la dificultad del usuario para transmitir
- Ayuda a entender: al usuario y al analista

Cuestionarios

Un cuestionario es un 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.
- **Cerradas:** Limitan o cierran las opciones de respuestas disponibles

Tipo de información obtenida

- **Actitud:** Lo que las personas dicen que quieren
- **Creencias:** Lo que las personas creen que es verdad

- **Comportamiento:** Lo que realmente hacen
- **Características:** De las personas o cosas

Cuándo usar Cuestionarios

- **Las personas están dispersas geográficamente:** Diferentes oficinas o ciudades
- **Muchas personas involucradas:** Clientes o usuarios
- **Queremos obtener opiniones generales**
- **Queremos identificar problemas generales**

Entrevistas

Técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara.

Es una conversación con un propósito específico, que se basa en un formato de preguntas y respuestas en general.

Permite 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 (en realidad sí, pero menos efectiva)

Tipo de información obtenida

- **Opiniones personales**
- **Objetivos**
- **Procedimientos informales**
- **Sentimientos personales**

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

    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

    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

Organización de una entrevista

Piramidal (Inductivo): De preguntas cerradas a abiertas

Embudo (Deductivo): De preguntas abiertas a cerradas

Diamante (Comb. de las anteriores): De cerradas a abiertas a cerradas

Ingeniería de Requerimientos

La 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" (SRS = Software Requirements Specification)

Importancia:

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

ESTUDIO DE VIABILIDAD

Principalmente para un sistema nuevo:

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

Responde a las siguientes preguntas:

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

Una vez que se ha recopilado toda la información necesaria para contestar las 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

Una vez realizada la elicitación y el estudio de viabilidad, hay que escribir de manera ordenada (documentar) los requerimientos del cliente

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.

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, eficiencia
- **Restricciones de Diseño:** Estándares requeridos, lenguaje, límite de recursos, etc.

Usuarios de un SRS

- Clientes del sistema
- Administradores
- Ingenieros del sistema
- Ingenieros de prueba del sistema
- Ingenieros de mantenimiento del sistema

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

Dos conceptos distintos:

- **Validación: Es hacer el software correcto.** Según la IEEE: Al final del desarrollo evaluar el software para asegurar que el software cumple los requerimientos. Sólo se puede hacer con la activa participación del usuario
- **Verificación: Es hacer el software correctamente.** Según la IEEE: El software cumple los requerimientos correctamente

Es suficiente validar después del desarrollo del software?

- La evidencia estadística dice que NO, cuanto más tarde se detecta, más cuesta corregir (Boehm). A esto se lo conoce como “Bola de nieve de defectos” .
  Validar en la fase de especificación de requerimientos puede ayudar a evitar costosas correcciones después del desarrollo.

¿Contra qué se verifican los requerimientos?

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

¿Qué se valida?

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

Las técnicas para realizar validación pueden ser manuales o automatizadas

- Revisiones de requerimientos (formales o informales)
    - Informales : Los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea posible.
    - Formal : El equipo de desarrollo debe conducir al cliente, explicándole las implicaciones de cada requerimiento

  Antes de una revisión formal, es conveniente realizar una revisión informal.

- Construcción de prototipos

- Generación de casos de prueba

TÉCNICAS DE ESPECIFICACIÓN DE REQUERIMIENTOS

La especificación de requerimientos es un paso crucial para el éxito del proyecto. Si no se toma el tiempo de entender los requisitos de los stakeholders el proyecto puede fracasar.

Existen diferentes técnicas y herramientas que se pueden utilizar para la especificación de requerimientos:

TÉCNICAS 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.

TÉCNICAS 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.

Historias de usuario

Una historia de usuario es una descripción corta y simple de un requerimiento de un sistema, que se escribe en lenguaje común del usuario y desde su perspectiva.

Son utilizadas en las metodologías de desarrollo ágiles (Ejemplo: XP, SCRUM) para la especificación de requerimientos

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

Características:
Independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables en tiempo, pequeñas y verificables

Beneficios:

- Al ser muy corta, (ésta debería poder escribirse sobre una nota adhesiva pequeña) ésta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas).
- Necesitan poco mantenimiento.
- Mantienen una relación cercana con el cliente.
- Permite dividir los proyectos en pequeñas entregas.
- Permite estimar fácilmente el esfuerzo de desarrollo.
- Es ideal para proyectos con requisitos volátiles o no muy claros ya que permiten responder rápidamente a los requisitos cambiantes. .

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.

Casos de uso

Un CU es el proceso de modelado de las “funcionalidades” del sistema en término de los eventos que interactúan entre los usuarios y el sistema.

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.
- Es un medio de comunicación con los usuarios.
- Utiliza lenguaje común y fácil de entender por las partes.
- Permite estimar el alcance del proyecto y conocer el esfuerzo a realizar.
- Define una línea base para la definición de los planes de prueba.
- Define una línea base para toda la documentación del sistema.
- Proporciona una herramienta para el seguimiento de los requisitos.

Componentes:

- **Diagrama de Casos de Uso:** Ilustra las interacciones entre el sistema y los actores
- **Caso de uso:** Funcionalidad individual del sistema
- **Escenarios de un CU:** Descripción de la interacción entre el actor y el sistema
- **Actor:** Papel desempeñado por los usuarios
- **Relaciones:** Asociacion, extensión, uso, dependencia, herencia...

Diagrama de Transición de Estados

Los diagramas de Transición de Estados tienen su origen en las Máquinas de Estado Finito.

Una máquina de estado finitos describe al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos).

Las MEF son formalmente una función matemática, que también se puede representar con un diagrama (grafo).

Los nodos del grafo representan estados, y las aristas representan los eventos que hacen que el sistema cambie de estado.

Redes de Petri

Las Redes de Petri son 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 ejecutadas en un orden impredecible.

Las Redes de Petri NO son secuenciales.

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

Componentes:

- **Sitios:** estados
- **Transiciones:** eventos
- **Arcos:** Relación entre sitios y transiciones
- **Tokens:** Establecen el estado inicial y durante la ejecución habilitan o deshabilitan transiciones, manejando la coordinación

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 (V o F) satisfechas por el sistema en un momento dado REGLAS ($2^{N° \text{ condiciones}}$) para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y ACCIONES simples a ser tomadas como un resultado.

Tipos de especificaciones:

- **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. También hay redundancia cuando hay dos reglas con alternativas que no representan una diferencia en el resultado; en dicho caso se debe reducir la tabla con un guión.

- **Especificaciones contradictorias:** Aquellas que especifican para reglas que determinan las mismas condiciones acciones distintas

Análisis Estructurado

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

Objetivos:

- Describir lo que requiere el cliente
- Establecer una base para la creación de un diseño de Software
- Definir un conjunto de requisitos que se pueda validar una vez que se construya el software

Estructura:

El análisis estructurado se compone por 3 modelos:

- **Modelado de datos del sistema:** Diagrama entidad-relación (lo vemos en DBD)
- **Modelado de las funciones del sistema:** Diagrama de flujo de datos
- **Modelado de comportamiento del sistema:** Diagrama de Transición de estados

Diagrama de Flujo de Datos (Modelado de funciones del sistema):

DFD es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por “conductos” y almacenamientos de datos.

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.

Permite ver desde dónde y hasta donde viajan los datos.

Componentes:

- Entidad externa
- Almacén de datos
- Flujo de datos
- Proceso (burbuja)

Modelos de proceso de software

¿QUÉ ES UN PROCESO?

Es un conjunto “ordenado de tareas”

¿QUÉ ES UN PROCESO DE SOFTWARE?

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

Actividades fundamentales de los procesos:

- **Especificación del software:**

  Consiste en el proceso de comprender y definir que servicios se requieren del sistema, así como la identificación de las restricciones sobre la operación y desarrollo del sistema. También llamada Ingeniería de Requerimientos.

- **Desarrollo del software:**

  Corresponde al proceso de convertir una especificación del sistema en un sistema ejecutable. Incluye los procesos de diseño y programación.

- **Validación del software:**

  Se realiza para mostrar que un sistema cumple tanto con sus especificaciones como con las expectativas del cliente. La prueba del sistema con datos de prueba simulados, es una de las formas de validación. Pero también incluye inspecciones y revisiones en distintas etapas.

- **Evolución del software:**

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

¿QUÉ ES UN MODELO DE PROCESO DE SOFTWARE?

Es una representación simplificada y abstracta de un proceso de software que presenta una visión de ese proceso.

Características:

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

TIPOS DE MODELOS DE PROCESO

Modelos prescriptivos

> Prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de la ingeniería del software, tareas, aseguramiento de la calidad y mecanismos de control.
>
> Cada modelo de proceso prescribe también un “flujo de trabajo”, es decir de qué forma los elementos del proceso se interrelacionan entre sí.
>
> Prescribir = Ordenar o decidir la obligatoriedad de una cosa

Modelos descriptivos

> Es una descripción de la forma en que se realizó en la realidad.
>
> Lo ideal sería que el descriptivo sea igual al descriptivo. Si digo que voy a usar un modelo específico, sería lo correcto seguirlo al pie de la letra. No siempre es posible, pero debe quedar en claro que mientras menos varíe el descriptivo con respecto al descriptivo, mejor.

MODELOS DE PROCESO TRADICIONALES

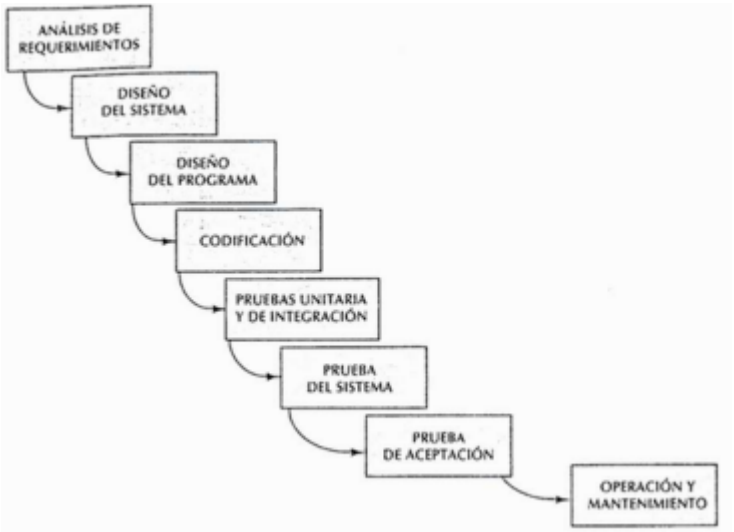
Modelo en Cascada

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

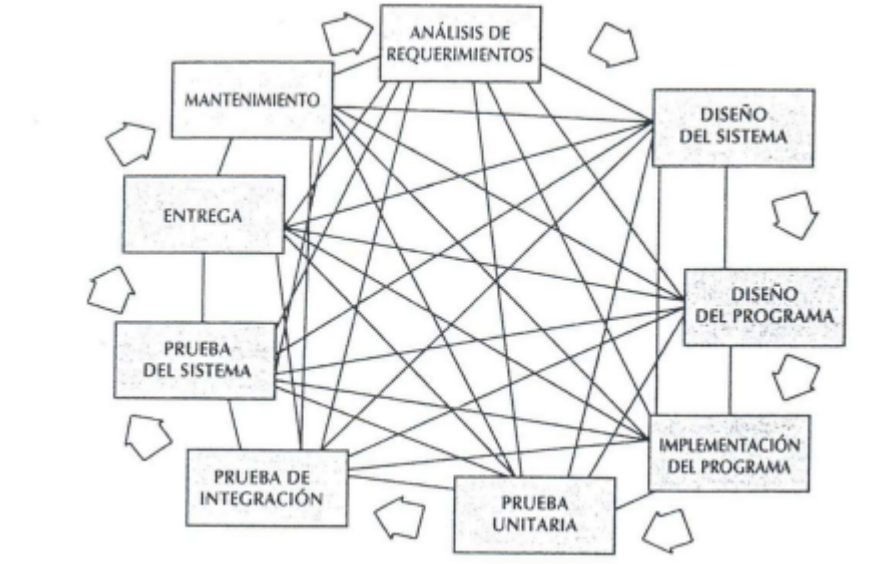
Su simplicidad hace que sea fácil explicarlo a los clientes.



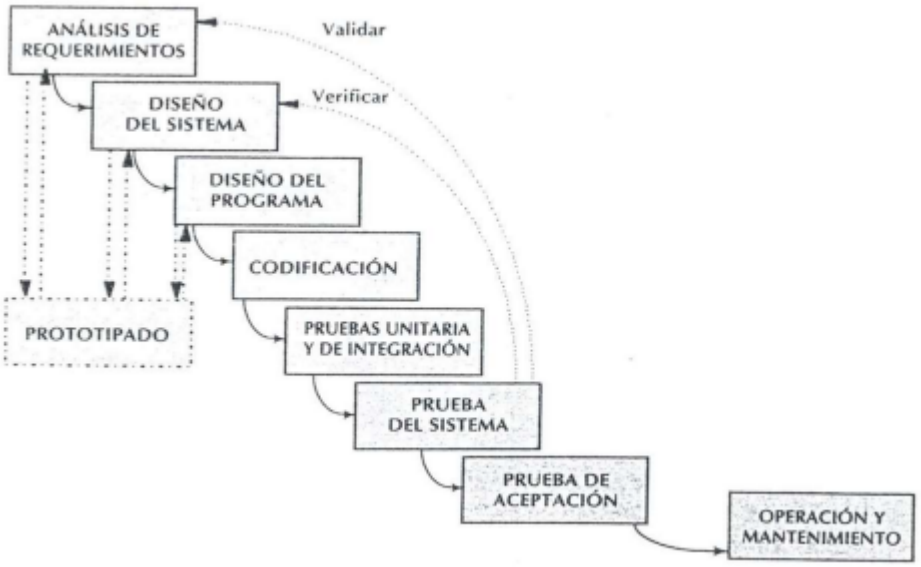
Dificultades:

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

Modelo de la realidad al intentar aplicar cascada = muy caótico.



Mejora: Modelo en Cascada con Prototipo

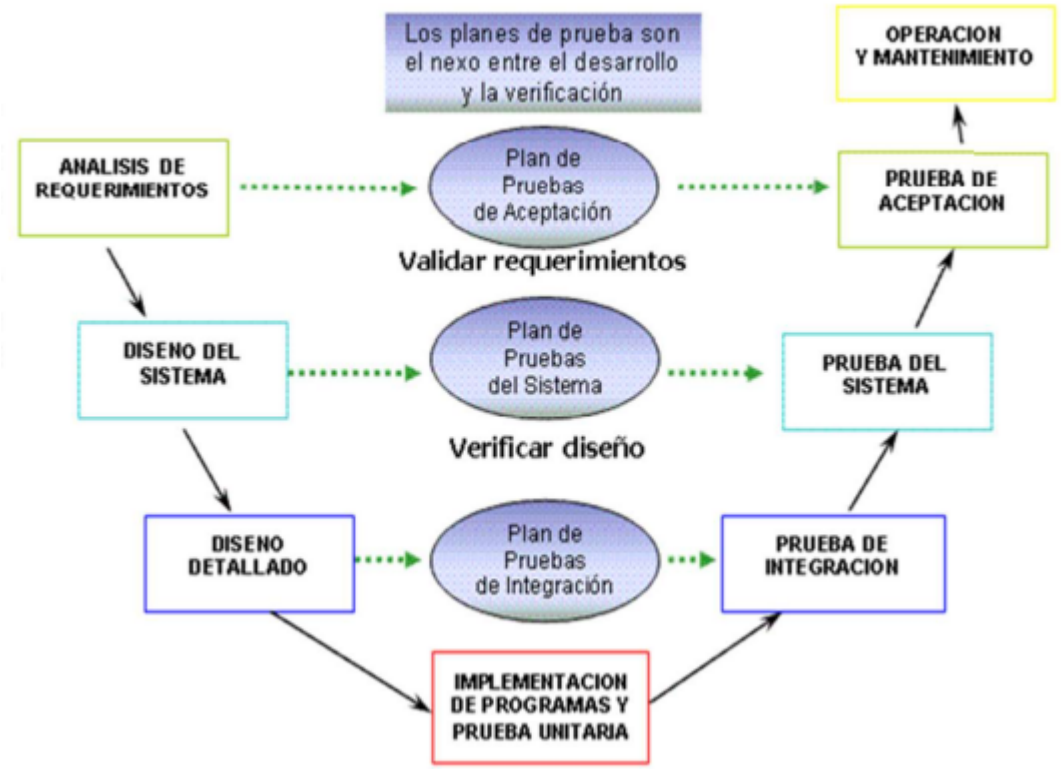


Modelo en V

Demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño.

Sugiere que la prueba unitaria y de integración también sea utilizada para verificar el diseño del programa

La vinculación entre los lados derecho e izquierdo implica que, si se encuentran problemas durante la verificación y validación (los planes de prueba), entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema.



Modelo de prototipos

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

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

Características:

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

Tipos de modelos de prototipos:

- **Evolutivos:**

  El objetivo es al final obtener el sistema a entregar (Evoluciona hasta ser el producto terminado).

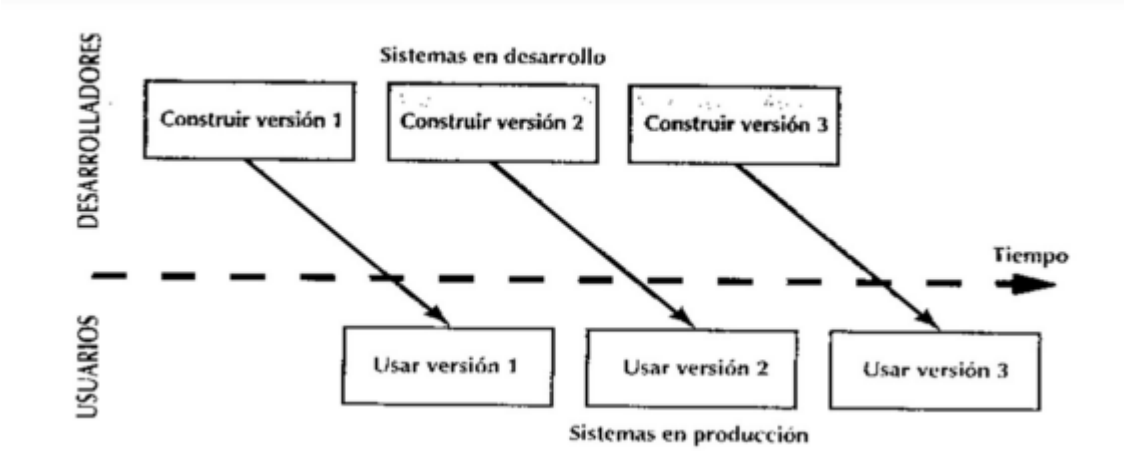
  Permite que todo el sistema o alguna de sus partes se construyan rápidamente para comprender o aclarar aspectos y asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución

- **Descartables:**

  No tiene funcionalidad. Se utilizan herramientas de modelado

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 de modelos de desarrollo por fases:

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

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

Modelo en espiral (Boehm)

En este modelo se trabaja por ciclos.

Cada ciclo en la espiral se divide en cuatro sectores:

1. **Establecimiento de objetivos:** Se identifican restricciones, se traza un plan de gestión, se identifican riesgos

2. **Valoración y reducción del riesgo:** Se analiza cada riesgo identificado y se determinan acciones.

3. **Desarrollo y validación:** Se determina modelo de desarrollo.

4. **Planeación:** El proyecto se revisa y se toma decisiones para la siguiente fase

Características:

- 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

MODELOS DE PROCESO MODERNOS

ÁGIL VS NO ÁGIL

METODOLOGÍA ÁGIL	METODOLOGÍA NO ÁGIL
Pocos artefactos	Más artefactos
Pocos roles	Más roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente interactuar con el equipo de desarrollo mediante reuniones	El cliente interactuar con el equipo de desarrollo mediante reuniones
Grupos pequeños y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

METODOLOGÍAS ÁGILES

Surgen como alternativa y mejora a las metodologías que tenían un marcado énfasis en el control del proceso, definiendo roles, actividades, herramientas y documentación detallada.

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

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

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.

Valores

- **Individuos e interacciones** más que procesos y herramientas.
- **Software operante** más que documentaciones completas.
- **Colaboración con el cliente** más que negociaciones contractuales.
- **Respuesta al cambio** más que apegarse a una rigurosa planificación.

Principios

1. Nuestra mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valuable.

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 proyecto. 2023 Ingeniería de Software I

5. Construir proyectos alrededor de motivaciones individuales.

6. Darles el ambiente y el soporte que ellos necesitan y confiar el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores.

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 constante indefinidamente.

9. La atención continua a la excelencia técnica y buen diseño incrementa la agilidad.

10. Simplicidad (el arte de maximizar la 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 definan y continúen**. Para ellas, resulta difícil moverse hacia un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo.

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 ver dónde están y para ajustar las prácticas a su situación particular.

Prácticas:

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

Principales características: Historias de usuario, Roles, Proceso, Prácticas

Roles:

- Programador
- Jefe de proyecto (Manager)
- Cliente
- Entrenador (Coach)
- Encargado de pruebas (Tester)
- Rastreador (Tracker)

Ciclo de vida del proceso:

1- Exploración

2- Planificación

3- Iteraciones

4- Producción

5- Mantenimiento

6- Muerte

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 el desperdicio:** no generar artefactos, ni perder el tiempo haciendo cosas que no le suman valor al cliente.

- **Construir la calidad con el producto:** la idea es inyectar la calidad directamente en el código desde el inicio.

- **Crear conocimiento:** En la práctica no se puede tener el conocimiento antes de empezar el desarrollo. Si el equipo de desarrollo desconoce el dominio del problema (por ejemplo, que carajo es un remito, lpqtrmp) se aprende sobre la marcha.

- **Diferir las decisiones:** tomar las decisiones en el momento adecuado, esperar hasta ese momento, ya que uno tiene más información a medida 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 motive y se sienta respetada.

- **Optimizar el todo:** optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar, hay que tratarlo como tal.

Roles

- **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

- **Scrum Product Backlog:** es la lista maestra que contiene toda la funcionalidad deseada en el producto. La característica más importante es que 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 durante un Sprint determinado.

- **Burndown Chart:** muestra un acumulativo del trabajo hecho, día-a-día

- Entre otros...

Proceso :

- 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** (como en un scrum de rugby), 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.

¿Cuándo 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.

Calidad

Definición de Calidad

Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.

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.

¿Qué es Calidad?

Primero, la definición de norma: 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.

Las principales **normas internacionales definen la calidad 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 le confieren aptitud para satisfacer unas necesidades expresadas o implícitas” (ISO 8402)

La Organización Internacional de Estandarización (International Organization for Standardization, ISO) es una organización para la creación de estándares (o normas) internacionales compuesta por diversas organizaciones nacionales de normalización.

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 hincapié en los estándares (o normas) de calidad.

Stylianou y Kumar plantean que se debe apreciar la calidad desde un todo, donde **cada parte que la componen debe tener su análisis de calidad**. Calidad de infraestructura, calidad del servicio, calidad de la información, calidad del software, calidad de gestión, calidad de los datos, etc...

Calidad del Software

La calidad del software se divide en:

- Calidad del producto obtenido
- Calidad del proceso de desarrollo

Ambas son dependientes entre sí, si mi producto es de calidad, seguro que el proceso de desarrollo también, y viceversa.

Calidad del Producto y Proceso

- **Producto:**

  La estandarización del producto define las propiedades que debe satisfacer el producto software resultante.

- **Proceso:**

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

CLASIFICACIÓN DE NORMAS DE CALIDAD

CALIDAD DEL PRODUCTO DE SOFTWARE

ISO 25000

Especifica cuales son las características que debería tener el producto de software para tener calidad interna, externa y de los datos.

Esta norma propone medir la portabilidad, seguridad, facilidad de mantenimiento, compatibilidad, funcionalidad, confiabilidad, facilidad de uso y eficiencia.

No hay que medir todo, las características a evaluar se deciden.

CALIDAD DE PROCESO DE DESARROLLO

ISO 12207

Establece cómo debe ser el proceso para el ciclo de vida del software.

Qué etapas debe tener y qué actividades debe tener el modelo de proceso.

ISO 15504

Establece cómo saber si cada una de las actividades del modelo de proceso están correctamente desarrollado

CMMI

Es un modelo de evaluación de los procesos de una organización.

Es un marco de referencia para evaluar la madurez de los procesos actuales de la organización y establecer mejoras.

El CMMI escalonado evalúa toda la organización en general, que puede estar en nivel de madurez de 1 a 5

El CMMI Continuo se centra en evaluar la madurez de un área específica.

CALIDAD DE LA ORGANIZACIÓN Y SEGURIDAD

ISO 9001

Es una norma que se puede aplicar a cualquier proceso para evaluar la calidad de una organización, proponiendo una mejora continua.

Con esta certificación se asegura que el negocio cumple requisitos legales. Se aumenta el rendimiento simplificando los procesos y mejora el rendimiento financiero