DEFINICIONES

Software:

1. Instrucciones que cuando se ejecutan, proporcionan la función y comportamiento deseados
2. Estructuras de datos que facilitan a los programas manipular adecuadamente la información
3. Documentos que describen la operación y el uso de los programas

Software genérico: Software desarrollado para el mercado general

Software a medida: Software diseñado para un cliente en particular de acuerdo a sus especificaciones

Ingeniería del Software: Disciplina de la ingeniería que comprende todos los aspectos de la producción del software desde las etapas iniciales de especificación del sistema hasta el mantenimiento de este, una vez comenzada su utilización. La ingeniería del software implica aplicar un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software.

Prácticas del desarrollo de software: Una norma técnica, modelo, .. empleada para establecer un enfoque disciplinado y uniforme para el proceso de desarrollo de software (IEEE, 1990), en otras palabras, una actividad bien definida que contribuye a la satisfacción de los objetivos del proyecto.

Buena Práctica: Es una práctica que a través de investigación y experiencia ha demostrado llevar de manera confiable al resultado deseado y es recomendada prudente y puede ser recomendada en una variedad de contextos.

Proceso de desarrollo software: proceso por el cual las necesidades del usuario se traducen en un producto de software. El proceso implica traducir las necesidades del usuario en requisitos de software, transformar los requisitos de software en diseño, implementar el diseño en código, probar el código y, a veces, instalar y verificar el software para su uso operativo. Estas actividades pueden superponerse o realizarse iterativamente

Modelo de proceso software: descripción simplificada y abstracta de un proceso de desarrollo de software. El propósito principal de un modelo de proceso de software es determinar el orden de las etapas involucradas en el desarrollo del software y establecer los criterios de transición para progresar de una etapa a la siguiente (Boehm, mayo de 1988).

Debido a la simplificación, varias metodologías de desarrollo de software pueden compartir un modelo de proceso: la diferenciación está en los detalles del proceso en sí. Los metodólogos de software incorporan las características generales de los modelos de desarrollo de software en procesos específicos de desarrollo de software que se apegan al espíritu de estos modelos.

MODELOS DE PROCESO:
IMPORTANTE VER IMÁGENES EN EL PPT

Modelos de Proceso	**Características Principales (Palabras clave)**	**Inconvenientes**
Desarrollo Ad-hoc	desestructurado, caótico, al azar	- diferencia de resultados de un proyecto a otro - resultados en base a habilidades del equipo
Modelo en Cascada	estructurado,rígido, irrealista	- proyectos reales no siguen el esquema propuesto - Al comienzo de los proyectos hay incertidumbre sobre los requerimientos y objetivos, esta incertidumbre no se tiene en cuenta en este modelo - trabajo largo y cuidadoso que no provee una versión funcional hasta el final del proceso.
Desarrollo Iterativo	dividido en pequeñas partes, mini modelo en cascada en cada parte, posibilidad de demostrar resultados antes del final del proyecto, feedback de la anterior fase provee información para la siguiente	- comunidad de usuarios tiene que estar activa (retraso en el proyecto) - habilidades de coordinación y comunicación son necesarias - Difícil planificar para las versiones más allá de la primera; - Las lecciones se pueden aprender demasiado tarde
Prototipado	no necesita toda la información al comienzo, muestra de prototipo al usuario para recibir feedback,	- puede llevar a falsas expectativas - puede llevar a sistemas de diseño pobre al centrarse en la rapidez
Modelo Exploratorio	ausencia de especificaciones precisas, validez del proyecto en base al resultado final	- limitado con lenguajes de alto nivel que permiten desarrollo rápido - difícil de evaluar coste-resultado - sistemas ineficientes
Modelo en Espiral	mejores características de modelo en cascada y prototipado, evaluación de riesgos, produce prototipo y se modifica en base a feedback, cada prototipo se diseña usando el modelo en cascada	- no puede hacer frente a cambios imprevistos (por ejemplo, los nuevos objetivos de negocio) - no está claro cómo analizar el riesgo

Modelo de Reusabilidad	usando componentes existentes, desarrollo orientado a objetos, si el modulo necesario no está disponible se creará y se incluirá para su posterior uso	- limitado a desarrollo orientado a objetos

Software:

- formas de crear software:
    1. Desarrollando nuevos sistemas o programas
    2. Parametrizando sistemas software creados para ello
    3. Reutilizando software existente

Modelo RUP (Proceso Unificado de Rational):

- Modelo de proceso moderno con 4 fases que es considerado **como un híbrido (COMO MI PUTO SEAT)** al reunir elementos de los modelos de proceso precedentes
- Fases:
    - Inicio
    - Elaboración
    - Construcción
    - Transición
- Mejores Prácticas:
    - Desarrollo **Iterativo**
    - Gestionar **Requisitos**
    - **Arquitectura** de uso de componentes
    - Modelo Visual (**UML**)
    - Verificar continuamente la calidad
    - Gestionar el Cambio
- Disciplinas - Técnicas
    - Modelado de Negocio
    - Requisitos
    - Análisis y Diseño

- Implantación
- Prueba
- Despliegue
- Gestión de la Configuración y del Cambio
- Gestión del Proyecto
- Entorno

Desarrollo Ágil de Software:

Iteraciones muy cortas

Tiempo fijo e incrementos pequeños

Participación activa del Cliente

Dirigido basándose en las prioridades de los hechos

Proceso adaptativo

Equipos potentes de personas centradas

Agilidad Tipos de proyecto:

- **Conejo:**
  - Interacciones frecuentes
  - Incrementos pequeños
  - Poca documentación
  - proyecto corto
- **Caballo:**
  - necesita cierto grado de formalidad y orden
  - duración media
- **Elefante:**
  - proyecto de duración larga
  - fallos pueden resultar muy graves (Como nota si te piden identificar el proyecto no es una característica)
  - comunicación formal
  - ej: Farmacéuticas, construcción de Aviones

CASE: Sistemas software creados para proporcionar soporte automatizado a las actividades de los procesos de la Ingeniería del Software

- Upper Case: Soporte a actividades iniciales del proceso de requisitos y diseño
- Lower Case: Soporte a actividades finales como programación, pruebas y depuración (Eclipse)

Ética:

- Los ingenieros de software deben comportarse de una manera honesta y éticamente responsable
- El comportamiento ético es más que cumplir y hacer cumplir la ley

TEMA 2

Requisito:

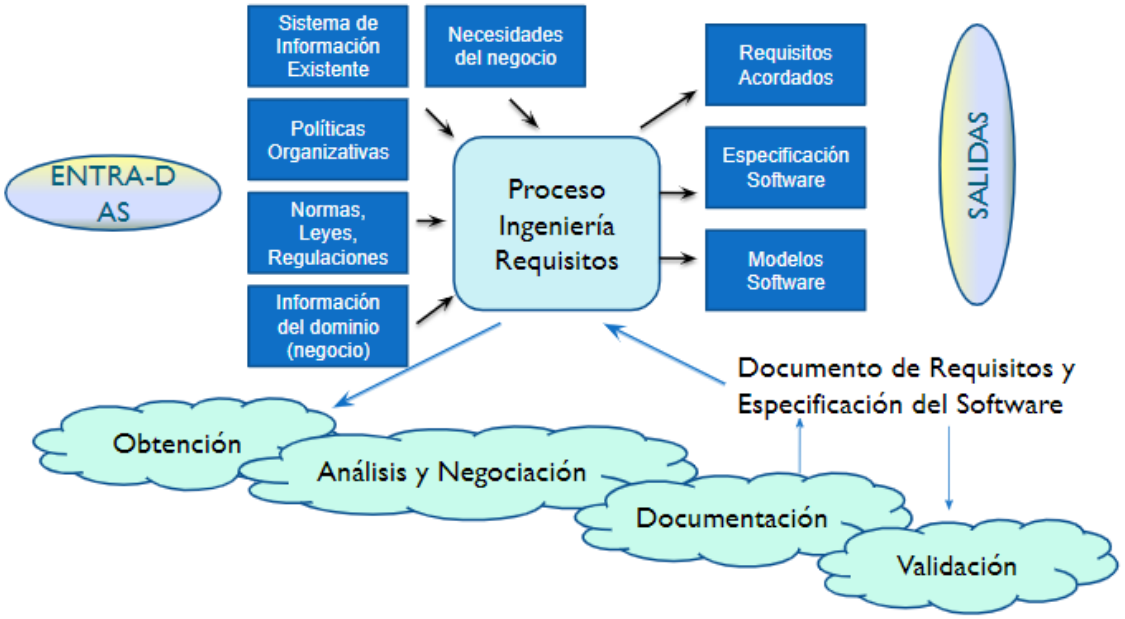
1. condición o capacidad que debe cumplir o poseer un componente del sistema, o el sistema, para satisfacer un contrato, norma o especificación o cualquier documento formalmente impuesto
2. condición o capacidad que necesita un usuario para alcanzar un objetivo
3. imagen, fórmula o modelo de una condición o capacidad(como en 1 y 2)

- Descripción de las propiedades de un sistema, de sus atributos y de cómo se debe comportar.
- Restringen el proceso de desarrollo de un sistema
- Un requisito debe describir:
    - las facilidades a nivel de usuario
    - las propiedades generales del sistema
    - las restricciones específicas
    - las restricciones de desarrollo

Ingeniería de Requisitos: Actividades involucradas en el descubrimiento, documentación y mantenimiento del conjunto de requisitos de un sistema basado en computadora.
Implica el uso de técnicas sistemáticas e iterativas, que aseguran que los requisitos del sistema están completos, son consistentes, relevantes, etc.

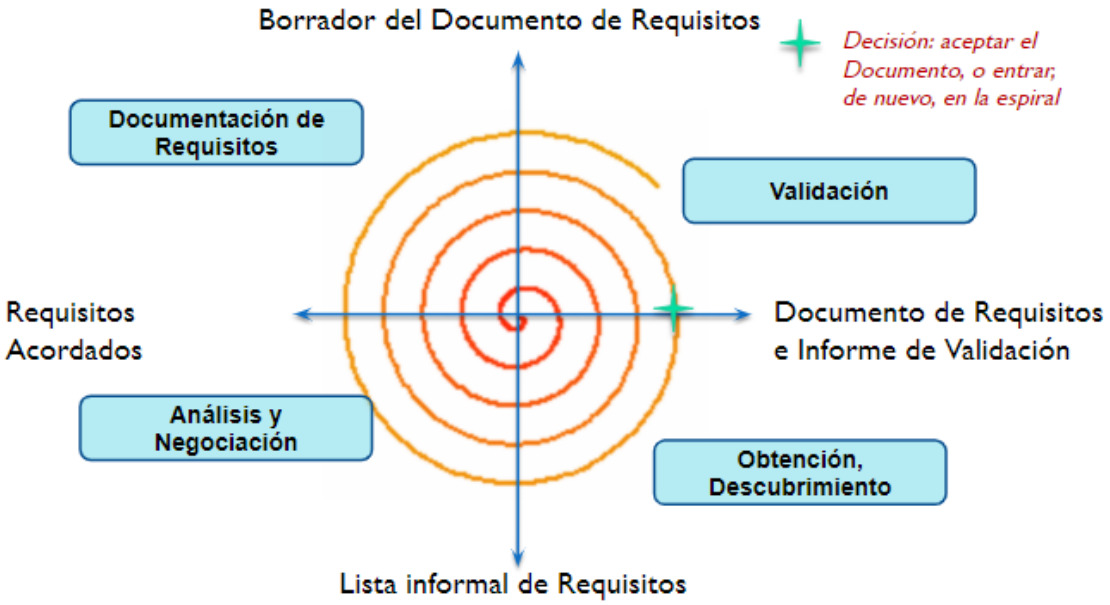
- base para el proceso de desarrollo del producto
- correcta y concreta para construir el software adecuado
- Dificultades
    - Muchos requisitos “se crean” no se encuentran
    - Usuarios, clientes e incluso desarrolladores pueden desconocerse
    - Objetivos en conflicto y distintos puntos de vista
    - Se vive con la inconsistencia e incompletitud, por un tiempo
    - Los requisitos cambian, evolucionan, durante y después del desarrollo
- Errores son costosos
- Debe incluir:
    - Obtención de Requisitos
    - Análisis y negociación de requisitos
    - Validación de requisitos

¿Cómo? Proceso de la Ingeniería de Requisitos – Entradas y Salidas



5

Proceso de la Ingeniería de Requisitos – Iteraciones



6

Stakeholders: Personas que se ven afectadas por la existencia del producto software y se benefician de su existencia

- **Clientes:** Pagan por el desarrollo software
- **Clientes Compradores:** Compran el producto software una vez desarrollado
- **Usuario:** utiliza el producto
- Expertos en el dominio
- Gestores

Requisitos funcionales: lo que los usuarios necesitan que haga el sistema

Requisitos no funcionales: características del sistema como velocidad, usabilidad, aspectos legales…

"El trabajo": área del negocio afectada por el producto

- necesario identificar las **áreas de interés** de la que son relevantes en este contexto de negocios
- necesario identificar los **sistemas externos o adyacentes** con las que "el trabajo" se comunica

Diagrama de contexto de trabajo:

- Muestra el alcance de aplicación del área de negocio del cliente ("el trabajo")
- Muestra los sistemas adyacentes y el flujo de información entre los sistemas y "el trabajo"
- El software a construir con el tiempo se convierte en parte del "trabajo"

Evento de negocio: Cosas que suceden en el exterior del alcance del trabajo, de las cuales el trabajo se entera de su existencia al recibir un flujo de información.

Casos de uso del negocio(BUC): Respuesta preparada para cada evento de negocio

Sistemas adyacentes: Sistemas que intercambian información con el flujo de trabajo

Obtención de requisitos:

Actividades Fundamentales:

- entendimiento del dominio de aplicación
- identificación de fuentes de requerimientos
- análisis de Stakeholders
- selección de técnicas, enfoques y herramientas para utilizar
- sonsacar los requerimientos a stakeholders y otras fuentes

Papeles de Ingeniero de Requisitos:

- facilitador, guía a los participantes a cuestiones relevantes
- mediador, busca solución sostenible
- documentador, producción de resultados
- desarrollador, todos los roles de desarrollo de software
- validador, requerimientos satisfacen características deseadas

Técnica	**Descripción breve. Estilo telegrama**	**Fortalezas**	**Debilidades**
Entrevistas	- informal - interacciones humanas - estructuradas: siguen unas preguntas previamente determinadas - no estructuradas: no siguen ningún esquema	- forma eficiente de recolectar mucha información rápidamente - estructurada: rigurosa y efectiva - no estructurada: posibilidad de ahondar en temas	- calidad depende de interacción entre participantes - depende de la habilidad del entrevistador - estructurada: no se consiguen nuevos enfoques - no estructurada: riesgo de no abordar temas
Cuestionarios	- preguntas abiertas o cerradas	- forma eficiente de obtener información de varios stakeholders	- el dominio debe ser comprendido por ambas partes - preguntas mal diseñadas pueden llevar a información redundante o irrelevante - profundidad de conocimiento limitada - posibilidad de malentendidos
Análisis de Tareas	- las tareas se dividen en subtareas y secuencias detalladas hasta que todas las acciones y eventos están descritas	- construye jerarquía de tareas y determina conocimiento necesario para poder realizarlas - provee información de las interacciones de usuario y de sistema respecto a las tareas - descripción contextual de las actividades	- necesita grandes esfuerzos - necesita establecer nivel de profundidad de la información
Análisis del Dominio	- examinar documentación y aplicaciones existentes	- útil para requerimientos tempranos	- no provee requisitos más allá de los tempranos

		- entendimiento y captura del dominio - importante cuando proyecto implica sustitución de un sistema - uso con otros técnicas - contribuye a crear entendimiento entre Stakeholder y analista	
Introspección	- basado en lo que el analista cree que usuarios y stakeholders necesitan	- usado como punto de partida	- analista debe estar muy familiarizado con el dominio y objetivos y debe ser experto en el proceso de negocio de los usuarios
Trabajo en Grupo	- reuniones colaborativas	- involucran directamente a los stakeholders y promueve cooperación	- difíciles de organizar - posibilidad de que individualidades dominen la conversación - debe haber cohesión en el grupo - no efectivo en situaciones políticas
Tormenta de Ideas	- participantes de diferentes grupos de stakeholders generan tantas ideas como sea posible - corto periodo de tiempo - discusión informal - importante no criticar ideas	- se usa en fases preliminares - promueve el pensamiento libre y la expresión - promueve descubrimiento de nuevas soluciones a problemas existentes	- no resuelve grandes cuestiones - no sirve para toma de grandes decisiones
JAD	- todos los stakeholders disponibles investigan a través de una conversación tanto problemas existentes como soluciones	- toma de decisiones y resolución de cuestiones rápida - objetivos generales ya establecidos - bien estructurada	- riesgo de individualidades
Talleres de Trabajo	- diferentes reuniones de grupo	- creativos: promueven el	

	- énfasis en desarrollo y descubrimiento de requisitos - cooperativos, creativos, función cruzada y grupo de atención	pensamiento y la expresión - cooperativos: la comunidad de desarrollo está involucrada - función cruzada: involucra varios stakeholders	

Negociación de Requisitos: Es una práctica implementada en los comienzos, en la fase de planificación, de un proyecto, o una fase de un enfoque de desarrollo incremental en un entorno colaborativo con respeto y confianza mutuos entre los stakeholders

Prácticas efectivas para la negociación:

- obten los stakeholders adecuados
- establecer mentalidad de trabajo en equipo
- establecer vocabulario común
- mantener una lista de requisitos
- registrar los atributos de los requisitos
- seleccionar un enfoque operacional junto con evaluación de riesgos
- manejar la triple restricción (tiempo, dinero, alcance)

Priorización de requisitos:

¿Por qué es necesaria?

- paso crucial en la toma de decisiones de planificación del producto
    - ayuda a decidir qué requerimientos incluir en el producto
    - puede ayudar a encontrar un error en un requisito

Aspectos a tener en cuenta al priorizar:

- Importancia
- Penalización
- Costes
- Tiempo
- Riesgos
- Volatilidad
- Beneficio financiero
- Beneficio estratégico
- Competidores
- Recursos de la competencia
- Materia de lanzamiento
- Abilidad de venta

Técnicas de priorización:

Técnica	Breve descripción (en tus palabras)	Inconveniente
AHP	- sistema de toma de decisiones dinámico - compara cada par de requerimientos y determina cual es más importante y la diferencia en importancia (escala del 1 al 9)	- muchas comparaciones si el número de requerimientos es alto - pérdida de control al comparar exclusivamente por pares
Hundred Dollar Test	- los stakeholders reciben 100 unidades imaginarias para distribuir entre los requerimientos	- si hay muchos requisitos se vuelve cada vez más inexacto - errores en el cálculo (suma mayor que 100) - posibilidad de manipular resultados dando todo el dinero a un requisito - no se puede repetir debido a posibles diferencias entre resultados
Ranking	- escala numérica pero no hay empates en la posición	- cada requerimiento tiene un ranking pero no se ve la diferencia de importancia entre ellos - difícil alinear varios puntos de vista de distintos stakeholder, mejor para un stakeholder
Numerical Assigment	- más común - agrupa requerimientos en 2 (puede variar) grupos de prioridad diferentes - cada grupo debe representar algo con lo que los stakeholders lo puedan relacionar	- es necesario que los stakeholders entiendan la representación de cada grupo - stakeholders tienden a pensar que todo es crítico - cada requerimiento no recibe una prioridad única
Top-Ten	- el stakeholder elige su top 10 de requisitos de un conjunto mayor sin asignar orden entre ellos	- requiere balancear las diferentes cuestiones entre stakeholders

Validación de requisitos: Cuando se obtienen y registran requisitos = Control de Calidad

Control de calidad: Actividad mediante la cual cada requisito se revisa para ver si cumple con la especificación

Atributos a revisar:

- Completa: ¿Hay algo que falta?
- Rastreable: ¿Tiene identificador único y referencias cruzadas?

- Consistente: Definición de términos generales y uso de estos de manera consiste con su definición
- Relevante: Para el proyecto
- No ambiguo:
    1. ¿Algún criterio de validación?
    2. Tiene que considerar que se pueden entender de manera subjetiva
- Viable
- ¿Es un requisito o una solución?
- Dorados: Requisitos que si no se incluyen no afectan al producto pero aumentan el coste
- Silenciosos: Cambian el alcance del proyecto

Una ERS no es ambigua si y sólo si cada requisito recogido en ella tiene una sola interpretación.
Un requisito es verificable si, y sólo sí, existe algún proceso efectivo con el cual una persona o máquina pueda controlar que el actual producto de software que se construye satisface el requisito
Una ERS es verificable si, y sólo sí, todos los requisitos establecidos en ella son verificables.