Unidad Nº1

Definición de problema: Surge del deseo de transformar un estado, forma o condición de las cosas a otro. Existen dos tipos:

1-. Problemas cerrados: Admiten una sola solución (UNICA)

2-. Problemas abiertos: Admiten múltiples soluciones al problema.

Proceso de resolución de problema

1-. **Formulación:** Se detecta el problema, en este no se recibe un enunciado y se deben hacer preguntas para poder describir el problema.

2-. **Análisis del problema:** Se recopila información relevante y detallada, se define si es abierto o cerrado, se reciben opiniones y se descompone el problema. Se describen las restricciones y criterios a cumplir.

3-. **Búsqueda de soluciones:** Se utiliza la imaginación y una "lluvia de ideas" para encontrar la solución más óptima.

4-. **Elección de la solución:** Se elige en base a los criterios de selección, la relevancia de cada atributo y se realiza una comparación en relación costo/beneficio.

5-. **Especificación:** Luego de elegir la solución se describen todas las características de esta, modelos e informes.

Las soluciones se tienen en cuenta por los criterios y restricciones.

Criterios	Restricciones
Es una norma o parámetro para la selección, suelen ser comunes, estos varían su prioridad y su valor pueden oscilar en un rango	Es una característica pre-fijada de la solución, es como un requisito "legal". Es una condición que la solución al problema DEBE CUMPLIR, y es muy importante cuestionarlas.

Comparación de soluciones:

	Criterio 1	Criterio 2	...	Criterio n	Total
	Peso: p1%	*Peso: p2%*		*Peso: pn%*	*100%*
Solución A	*A1 * p1*	*A2 * p2*		*An * pn*	*T1*
Solución B	*B1 * p1*	*B2 * p2*		*Bn * pn*	*T2*
Solución C	*C1 * p1*	*C2 * p2*		*Cn * pn*	*T3*

1. Otorgamos una valoración numérica a cada uno
2. Multiplicamos por el peso del criterio
3. Sumamos el total de impacto de criterios para cada solución

Modelos de representación

Son abstracciones o simplificaciones de la naturaleza o comportamiento de un objeto, sistema, fenómeno o proceso. Sirven como abstracción del pensamiento, para comunicar, predecir y controlar, etc.

Tipos

- **Físicas o cónicas:** Representan objetos reales (su concepto), tienen una semejanza física con este, y conserva proporciones del mismo. Ej: Un globo terráqueo, una señal de tránsito, una maqueta de un avión, auto, etc.
- **Esquemáticas:** Representación simbólica, se diferencia por las escalas, su objetivo es identificar las partes, etc. Sin respetar las escalas. No intentan mostrar exactamente como son, sino explicarnos cómo funcionan. Ej una representación de las capas de la piel, de un motor y su funcionamiento, etc.
- **Graficas:** Visualizan relaciones y magnitudes relativas ("grafico torta", generalmente se usa para ver datos o relacionar valores/compararlos). Ej: Grafico de deudas, Ventas, Promedio de las temperaturas durante el año, Dólar futuro, etc.
- **Matemáticas:** Basados en expresiones analíticas o numéricas, usados para predicción y comunicación (su uso es predecir a partir de una formula o un algoritmo, dato, etc.). Ejemplo fórmula para revalorizar pensiones.
- **Simulaciones:** La idea es querer usarlos para experimentar algo y no probar con el sistema real. Se experimenta mediante representaciones **físicas o icónicas, analógicas** (karting, experimentación de un accidente con muñecos) **y digitales** (Simuladores de juegos, video 3D de un tsunami, etc.)

Mediciones

Se tienen que identificar los siguientes:

Entidad: Objeto o evento del mundo real (a que se mide).

Atributo: Característica o propiedad de una entidad (que se le mide).

Medición: Proceso mediante el cual se asigna números o símbolos a los atributos de entidades del mundo real. La medición es el acto de determinar una medida (Es la **acción** de medir.).

Medida: Elemento que proporciona un indicio cuantitativo de la extensión, cantidad, dimensión, capacidad o tamaño de algún atributo de un producto o proceso (es el resultado de la medición, el valor obtenido. Ej: la mesa mide 1.50m de largo).

Métrica: Medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo determinado (método de medición elegido junto con la escala de medición. El **método o sistema de referencia** que define cómo se mide y en qué escala. Ej: Método de medición: Metro / Medidor Laser. Escala: Absoluta.

Indicador: Métrica o combinación de métricas que proporcionan comprensión acerca del proceso, el proyecto o el producto en sí (Los indicadores proporcionan comprensión para realizar ajustes, para incorporar cambios que permitan hacer mejor las cosas).

Escalas de medición

- **Escala nominal:** Tiene dos características principales: El sistema de relación empírico consiste solo de diferentes clases, no hay noción de orden entre ellas. Cualquier representación simbólica o numérica para las distintas clases es aceptable y no hay noción de magnitud, asociada a los números o símbolos

Escala nominal. Ejemplo.

Defecto en automotor. Atributo: ubicación

- M1(x) =

100 - mecánico
200 - eléctrico
600 - chasis

- Escala ordinal: Define clases o categorías. Diferentes clases ordenadas con respecto al atributo. Mientras preserve el ordenamiento es aceptable. Los números solo representan un ranking, no tiene sentido sumar o restar

Escala ordinal. Ejemplo.

Defecto en automotor. Atributo: gravedad

- M1(x) =

1 trivial
2 moderado
3 complejo
4 irresoluble

- M2(x) =

1 trivial
8 moderado
12 complejo
40 irresoluble

- **Escala de intervalos:** Preserva el orden, preserva diferencias, se puede sumar y restar. Ej: opinión en una encuesta.

- **Escala de proporciones:** Da información acerca de diferencia de proporciones existentes entre cada clase. Preserva el orden, existe el elemento cero, operaciones aritméticas son válidas. Ej: tiempos de llegada, espera, etc.

Escala de proporciones. Ejemplo.

Costo monetario de un producto. Atributo: valor

- M1(x) =

Cualquier valor mayor o igual a cero es válido
$20 es más caro que $15
$40 es el doble que $20

- M2(x) = Valor expresado en otra moneda

M2(x) = M1(x) * k

- **Escala absoluta:** La medición se realiza contando el número de elementos Siempre toma la forma "número de ocurrencias de x en la entidad". Todas las operaciones aritméticas son significativas

Escala absoluta. Ejemplo.

Monedas en un monedero. Atributo: cantidad

- M1(x) =

el número resultante de contar la cantidad de monedas
no importa el valor que simboliza cada moneda

Unidad Nº2

Software

Son instrucciones que cuando se ejecutan proporcionan las características, función y desempeño buscados, manipulan la información de forma adecuada. Es producto y herramienta para la construcción de un producto al mismo tiempo. está compuesto por elementos como: Hardware sobre el que ejecutan, software que permite operar el hardware, usuarios que configuran y usan el sistema, bases de datos, y otras piezas de HW y SW adicionales. se definen en función de los límites que permiten entender que está comprendido en el sistema y que no. Qué funciones provee el sistema y cuáles le son provistas por otros sistemas.

Ingeniería de SW: La aplicación de un método sistemático, disciplinado y cuantificable para el desarrollo, la operación y el mantenimiento de software; esto es, la aplicación de la ingeniería al software. Uso de principios de la ingeniería para desarrollar de forma económica software confiable, eficiente y que utilice máquinas reales.

Búsqueda de soluciones efectivas a problemas resolubles con computadoras.

Método: técnica formal para producir un resultado.

Procedimiento: combinación de herramientas y técnicas que producen un resultado. Formas de concretar el método.

Herramienta: instrumento o sistema automatizado para realizar una tarea de mejor manera.

Paradigma: enfoque particular o filosofía.

Participantes del proceso

Roles

1-. **Cliente:** Necesita un desarrollo de software, plantea problema o situación. Él lo paga.

2-. **Desarrollador:** Equipo o persona que construye el software para el cliente.

3-. **Usuario:** Utilizara el software creado, tiene necesidades específicas sobre el sistema

Miembros del equipo

1-. **Analistas de requerimientos:** determinar qué desea el cliente, documentan los requerimientos

2-. **Diseñadores:** Son aquellos que describen que debe hacer el sistema

3-. **Programadores:** Son aquellos que implementan los requerimientos en el código

4-. **Testers:** Realizan pruebas y detectan defectos del SW.

5-. **Capacitores:** Entrenan al cliente en el uso del sistema.

6-. **Equipo de mantenimiento:** Trabaja en corregir los defectos y/o cambiar aspectos del sistema al pasar el tiempo.

Proceso: conjunto de actividades, acciones y tareas a ejecutar que suceden en etapas para crear algún producto.

Actividad: Objetivo amplio, se desarrolla en cualquier dominio de aplicación, tamaño del proyecto o grado de rigor del trabajo. EJ: Rendir y cursar las materias de la carrera.

Acción: conjunto de tareas para obtener un producto del trabajo. EJ: Rendir y aprobar parcial.

Tarea: objetivo pequeño, bien definido, resultado tangible. EJ: Realizar ejercicio 1 tp.

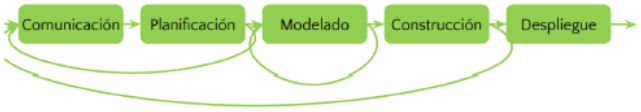
El proceso del software consiste de: Comunicación - Planificación - Modelado - Construcción - Despliegue

Se aplican a lo largo del proyecto para administrar y controlar el avance, la calidad los cambios y los riesgos. EJ: seguimiento y control, administración del riesgo, aseguramiento de la calidad, revisiones técnicas, medición, administración de la configuración.

La organización de las tareas y actividades se organiza respecto a su secuencia y tiempo.

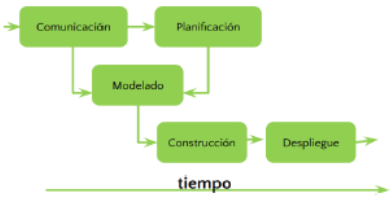
Lineal: secuencia de actividades

Iterativo: repite una o más de las actividades antes de pasar a la siguiente.



Evolutivo: cada circuito lleva a una versión más completa del producto

Paralelo: una o más actividades en paralelo con otras.

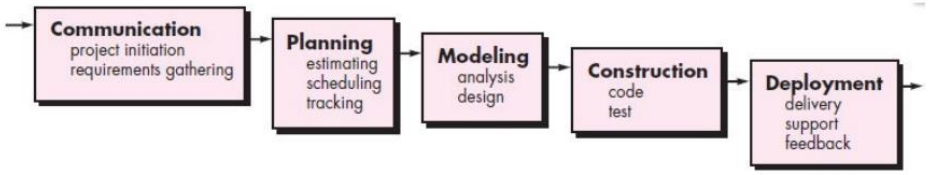


Modelos de proceso

- **Cascada**: El trabajo fluye de manera lineal desde la primera actividad hasta la última. Se aplica cuando los requerimientos están bien definidos o para mejorar el sistema existente.

Ventajas: Simple de explicar y entender para personas no familiarizadas con el desarrollo de software. Menciona los productos intermedios necesarios para avanzar

Desventajas: No es fiel a la realidad. No es flexible, no sirve cuando hay requerimientos dinámicos. El cliente debe esperar al final para ver una versión funcional.



- **Incremental:** Secuencias lineales aplicadas de manera escalonada. Puede haber paralelismo entre distintas secuencias. Cada secuencia produce un incremento en el software, nueva parte de ese producto. Útil cuando no hay personal suficiente, se trabaja con porciones reducidas de trabajo.

- **Iterativo o evolutivo:** El sistema se entrega completo al comienzo. Las iteraciones modifican (corrigen, mejoran) la funcionalidad.

- **Modelo Evolutivo – Prototipos:** Se planea rápidamente una iteración para producir un primer prototipo. Cada iteración refina el conocimiento, el plan y el modelo, y genera un nuevo prototipo. Útil cuando sólo se plantean objetivos generales en vez de requerimientos detallados. Prototipos desechables o evolucionan.

- **Modelo Evolutivo – Espiral:** Se itera de forma espiral. Las primeras iteraciones pueden producir modelos o prototipos. Cada iteración ajusta todos los documentos: definición, modelos, costos, incluso las iteraciones restantes. Se puede usar durante todo el ciclo de vida del sistema.

Metodologías agiles

El objetivo es entregar SW de calidad de manera rápida y flexible, adaptándose a los cambios en los requisitos. Su filosofía se basa en el Manifiesto Ágil:

- Las personas y la comunicación sobre los procesos y herramientas
- El SW funcionando sobre la documentación extensa
- La colaboración con el cliente sobre la negociación contractual
- La respuesta al cambio sobre el seguimiento rígido de un plan.

A medida que el proyecto avanza más, cambiar los requerimientos cuesta más. El proceso debe ser adaptable, debe hacerlo incrementalmente y debe permitir la retroalimentación con el cliente. Hay que intentar que el inevitable cambio surja tan temprano como sea posible. Ejemplos:

1-. **Scrum:** Considera actividades estructurales: Requerimientos, análisis, diseño, evolución, entrega. Cada actividad realiza las tareas bajo un patrón de proceso llamado sprint, que son unidades de trabajo necesarias para alcanzar un requerimiento. Cada día de trabajo toma lugar una reunión scrum, donde se pregunta: Que hiciste desde la última reunión, qué obstáculos encontraste y que planea hacer antes de la próxima reunión.

2-. **Programación Extrema (XP):** Enfatiza la colaboración estrecha pero informal entre clientes y desarrolladores y la retroalimentación constante. Cuatro actividades estructurales:

   - Planificación: Recabar requerimientos.
   - Diseño: Mantenlo sencillo. Si es difícil, se realizan prototipos.
   - Codificación: No codificar, diseñar pruebas unitarias, luego desarrollar el código. Se trabaja de a pares.
   - Pruebas: Automatizarlas cuando sea posible. Efectuar pruebas de integración a diario.

Ingeniería de requerimientos

Requerimiento: Característica o descripción de un sistema que es capaz de hacer con el objetivo de satisfacer su propósito.

Ingeniería de requerimientos: Son las tareas alrededor del requerimiento. Lo que hacemos para entender que desea el cliente, evaluar la factibilidad, negociar acuerdos razonables, especificar soluciones sin ambigüedades, etc. En esta se incluyen las siguientes actividades:

- **Concepción:** Inicio de un proyecto de software, se establece el entendimiento básico del problema, se identifican participantes, se reconocen diferentes puntos de vista, preguntas entre cliente y participantes. (Se quiere aprender ¿cuál es el problema de negocio?).
- **Indagación:** Preguntar conforme al sistema, hacer muchas preguntas para entender bien que quiere el cliente (De alcance (fronteras mal definidas), de entendimiento (dar algo por obvio, no está seguro de lo que necesita, requerimientos que no se pueden probar) y

volatilidad (requerimientos cambiantes con el tiempo)). Distinguir entre atributos/criterios críticos (importantes, debe tenerlo si o si), deseables (no indispensables), posibles (podrían eliminarse). No confundir requerimiento con soluciones. Hay requerimientos funcionales (que hará el sistema, interacciones entre el sistema y el entorno) y no funcionales (describen restricciones que limitan elecciones en construcción de la solución). Identificar escenarios y casos de uso (que quieren conseguir los actores en la interacción con el sistema, variaciones en la interacción, que información adquiere, produce o cambia el actor)

Funcionales	No funcionales
¿Qué hará el sistema?	¿Cómo se comportará el sistema?
Describen **funciones**, **comportamiento**, **interacciones** entre el sistema y su entorno	Describen **restricciones** sobre el sistema que limitarán las funciones/comportamiento

- **Elaboración:** Expandir y refinar información obtenida. Se comienza con la elaboración de modelos que permitan representar la función, el comportamiento y la información trabajada por el sistema.
- **Negociación:** Toma lugar cuando el cliente pide más de lo que se puede conseguir, o hay requerimientos conflictivos. Se eliminan y/o modifican requerimientos hasta lograr satisfacción.
- **Especificación:** Puede tener muchas formas: Documentos escritos, modelos gráficos, modelos matemáticos formales, prototipos, etc.
- **Validación:** Se analiza la especificación para garantizar que todos los requerimientos han sido enunciados, no hay ambigüedades, se corrigieron inconsistencias y errores, y se siguieron los estándares establecidos.
- **Administración:** Se debe identificar, controlar y dar seguimiento a los requerimientos y a los cambios que sufran. Determinar cómo se administrarán los cambios.

Expresión de requerimientos

Modelos UML

- **Caso de uso:** Se narra desde el punto de vista del actor, es como dar un tutorial o explicación personal (es una narrativa informal).
- **Diagrama de actividad:** Representación gráfica del flujo de interacción, especifica diferentes caminos resultados obtenidos tras tomar diferentes decisiones.

- **Descripciones estáticas:** objetos y clases, operaciones, relaciones entre objetos. Especificar atributos y definir operaciones. Hay que identificar clases, objetos y atributos, y ocurrencias que suceden dentro del contexto.
- **Descripciones dinámicas**: Descripciones funcionales y diagramas de transición. Modelan como se expresan, como ocurren los cambios en las entidades de las descripciones estáticas.

Diseño

Diseñar un sistema es determinar un conjunto de componentes y de interfaces entre componentes que satisfagan un conjunto específico de requerimientos.

Se plantean distintos niveles de abstracción. Abstracciones de procedimiento (secuencia de instrucciones), abstracciones de datos (conjunto que describe un objeto). Más abstracción, soluciones en términos amplios. Menos abstracción, descripciones más detalladas.

Patrones: Describe una estructura de diseño que resuelve un problema particular de diseño dentro de un contexto específico.

Separación de intereses: Cualquier problema complejo se resuelve más fácilmente si se lo subdivide en partes que pueden resolverse de manera independiente. Tiene efecto directo en el diseño y en muchos otros aspectos. La complejidad disminuye si los problemas son más simples.

Modularidad: El software se divide en partes separadas (módulos). Un sistema es modular cuando cada actividad la realiza un único componente que tiene bien definidas sus entradas y salidas. El costo de integrar módulos crece a medida que son más módulos.

Ocultamiento de la información: La información de cada módulo debe ser inaccesible a los que no necesiten de ella. Los módulos tienen que conseguir intercambiar solamente la información necesaria para que el software funcione.

Independencia funcional: Cada módulo debe resolver un conjunto específico de requerimientos. Es el resultado de división de problemas y ocultamiento de información.

Refinamiento: se empieza desde una descripción general, y se va refinando hasta llegar a un código en un lenguaje de programación. Es el proceso por medio del cual se disminuye el nivel de abstracción.

Buenas prácticas de un buen diseño

Se busca que cada componente del diseño sea lo más independiente del resto posible ya que: mejora la comprensión de cada componente, es más sencillo de modificar si no afecta a los demás, es más fácil identificar fallas, etc. Para medir la independencia de los componentes usamos dos conceptos:

- Acoplamiento: Dos componentes están acoplados si existe dependencia entre ellos. Altamente

acoplados, mucha dependencia entre ellos. Poco acoplados, pocas dependencias y/o interconexiones débiles.

- Cohesión: Grado de adhesión interna que tiene el componente. A mayor grado de cohesión, más relacionadas están sus partes internas entre sí. Un componente es cohesivo si todos sus elementos están orientados a realizar una única tarea y son esenciales para llevarla a cabo.

Tipos

Un buen diseño minimiza el acoplamiento y maximiza la cohesión

Implementación

Codificación del software: varios lenguajes de programación y personas involucradas, se coopera y se coordina. Se definen estándares de programación: Estilo, formato, codigo y documentación. A partir del diseño del componente el programador tiene libertad para ser creativo en la implementación.

Documentación del código: Explica que hace el programa y como lo hace.

Documentación Interna: material descriptivo escrito dentro del código (quien lo escribió, justifica las decisiones, ayuda a comprensión).

Documentación Externa: la que acompañe al programa (descripción del problema, de los algoritmos elegidos y de los datos).

Pruebas

Se concentra en búsqueda de defectos, encontrarlos. La prueba es para demostrar su existencia. La prueba es destructiva, es exitosa si encuentra defectos o fallas.

Tipos de defectos:

- **Algorítmicos:** error en la lógica de un componente que no produce la salida apropiada.
- **De sintaxis:** en las estructuras del lenguaje.
- **De computación y precisión**: implementación errónea de fórmulas o cálculos sin exactitud necesaria.
- **De documentación**: Cuando no corresponde con lo que hace el programa.
- **Por estrés o sobrecarga**: se sobrepasan las estructuras.
- **De capacidad o límites**: Se deteriora el desempeño del sistema por superar sus límites.
- **De sincronización**: coordinación inadecuada de procesos simultáneos.
- **De rendimiento**: el sistema no opera con el desempeño esperado
- **De recuperación**: respuesta a una falla no es la esperada.

Visiones de los objetos de prueba:

1. **Caja cerrada o negra:** El objeto se analiza desconociendo el contenido, las pruebas consisten en alimentar la caja negra con entradas y observar cuales son las salidas (Se da la entrada y se sabe cuál tiene que ser la salida).
2. **Caja abierta o blanca:** Se utiliza la estructura del objeto para probar de diversas maneras todos los caminos del código. Considera la estructura, y ajusta los casos de prueba en base a esta.