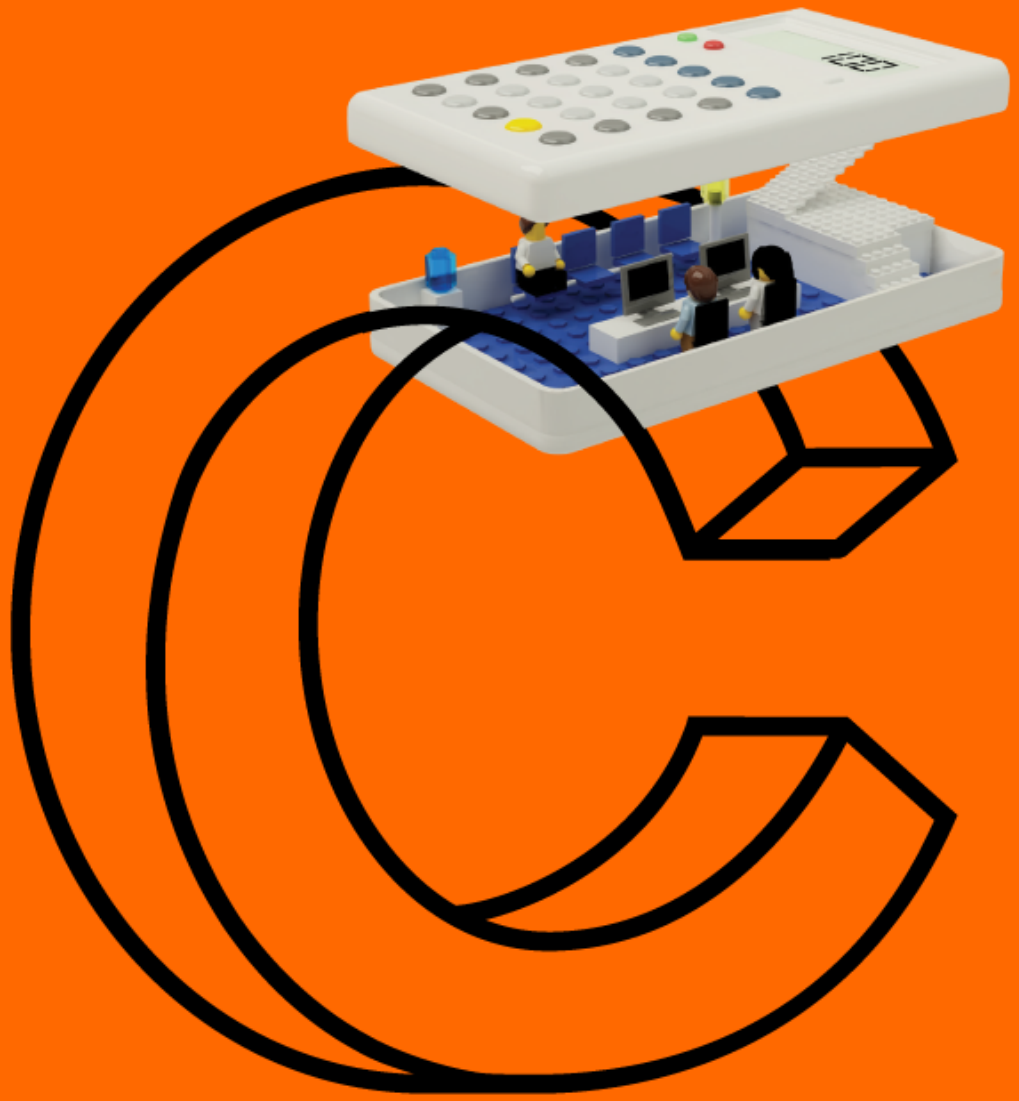
Análisis y Diseño de Sistemas Orientado a Objetos

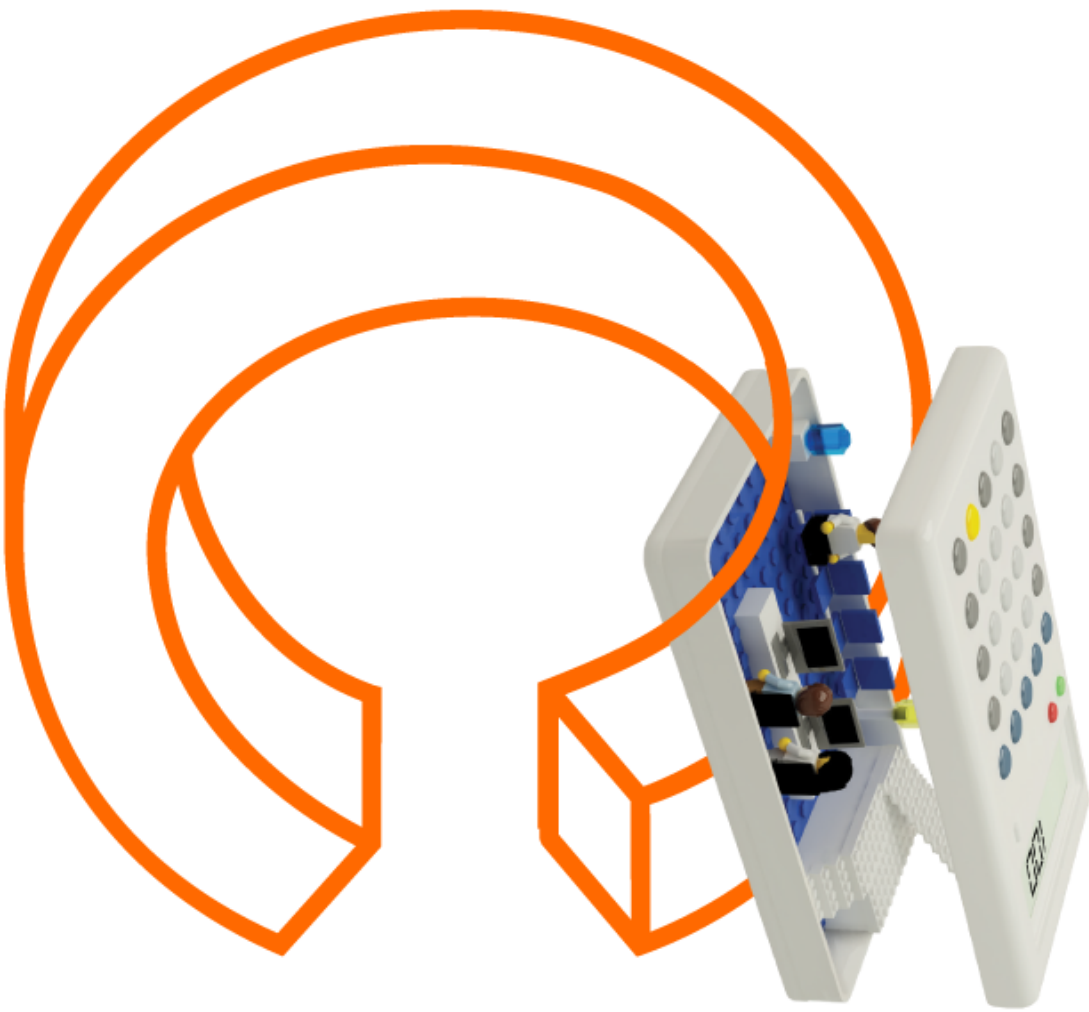


idat

Mg. Daniel Arias, **PMP™** y **Scaled Scrum Master™**

Requerimientos
Insumo a nivel de análisis

Semana 6



idat

Semana 6: Requerimientos – Insumo a nivel de análisis

Logro de aprendizaje

- A nivel de Análisis de la Arquitectura
  - Especificaciones
  - Glosario
  - Modelo de Caso de Uso
- A nivel de Análisis de Caso de Uso
  - Especificaciones
  - Glosario
  - Modelo de caso de uso

Actividades

Actividad 06:

- Diseña y Presenta los Insumos a nivel de Arquitectura y del Análisis de Caso de Uso.

Aprendemos:

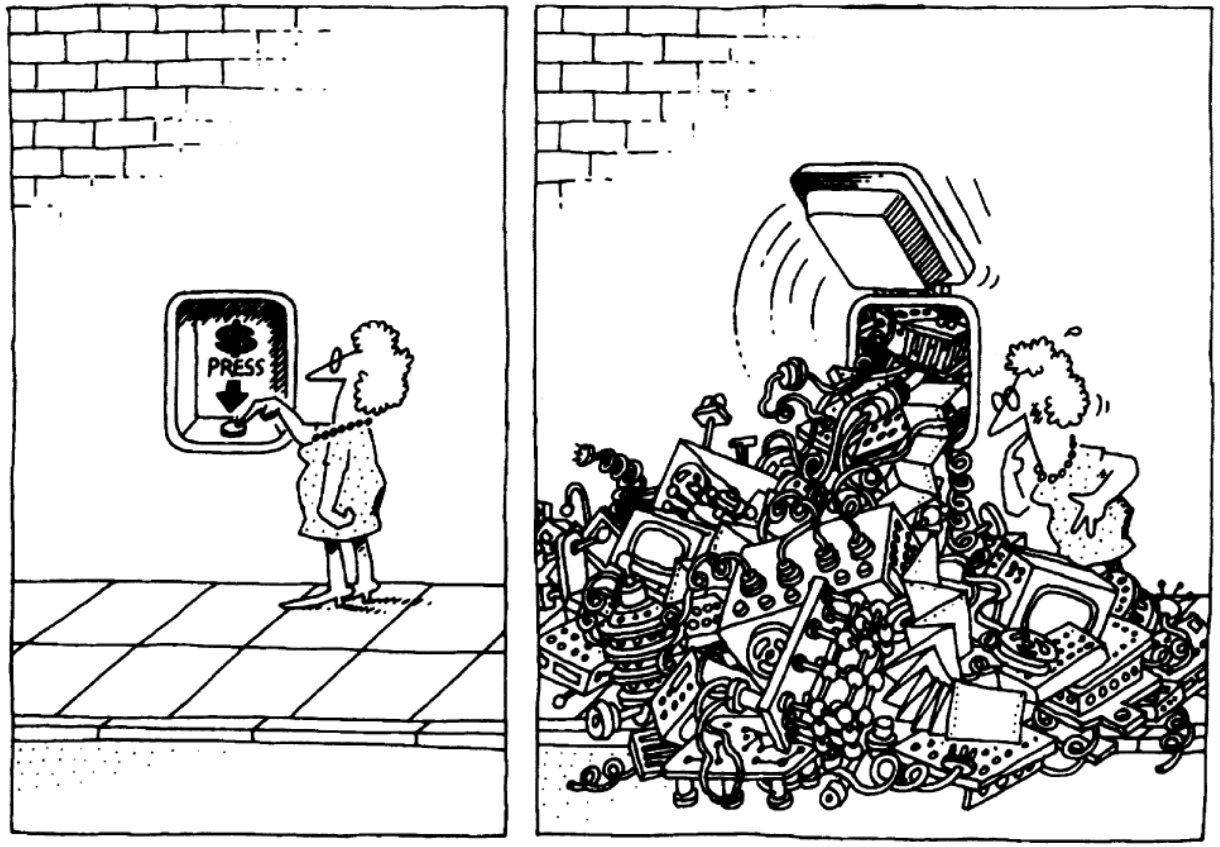
¿Qué aprendimos la clase pasada?

- Disciplina Requisitos
- Disciplina Implementación
- Disciplina Prueba
- Disciplina Gestión de Proyecto

http://www.menti.com

Aprendemos: Requerimientos – Insumo a nivel de análisis

La complejidad del desarrollo de software



La tarea del equipo de desarrollo de software es ofrecer la ilusión de simplicidad[1]

1- [Booch 94]

Aprendemos: Requerimientos – Insumo a nivel de análisis

La complejidad es inherente al software

La construcción de software puede involucrar elementos de gran complejidad, que en muchos casos no son tan evidentes como los que se pueden ver en otras ingenierías. Un puente, un edificio, una mina, una red de ferrocarriles son ejemplos de sistemas complejos de otras ingenierías, pero el ingeniero del software construye
sistemas cuya complejidad puede parecer que permanece oculta. El usuario siempre supone que Informática todo es muy fácil (“apretar un botón y ya está”)

Aprendemos: Requerimientos – Insumo a nivel de análisis

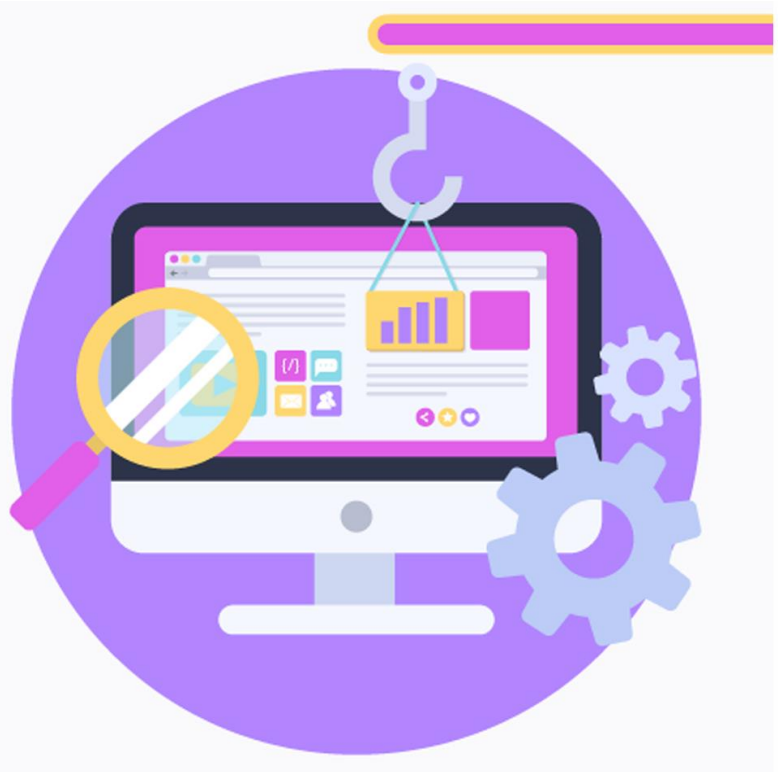
Propiedades de los sistemas de software simples o artesanales

- No son complejos.
- Suelen estar construidos y mantenidos por una sola persona (software artesanal):
  - No suelen pasar de 1.000.000 de líneas de código, aunque el número de líneas de código no sea la mejor medida de la complejidad del software.
- Ciclo de vida corto.
  - El ciclo de vida del software es todo el tiempo que un proyecto software está activo, comienza con las primeras tareas de análisis y no concluye hasta que ese software deja de utilizarse por los usuarios.
- Pueden construirse aplicaciones alternativas en un periodo razonable de tiempo.
- No necesitan grandes esfuerzos en análisis y diseño.
- No son nuestro objetivo de estudio.

Aprendemos: Requerimientos – Insumo a nivel de análisis

Las propiedades de los sistemas software complejos

- También se denominará software de dimensión industrial.
- Es muy difícil o imposible que un desarrollador individual pueda comprender todas las sutilidades de su diseño.
- La complejidad es una propiedad esencial de estos sistemas, que puede dominarse, pero no eliminarse.
- Son el objetivo de nuestro estudio.
- Ejemplos: Un sistema de reservas, anulaciones y ventas de billetes aéreos para un conjunto de compañías aéreas que se pueda utilizar en cualquier lugar del mundo.



Aprendemos: Requerimientos – Insumo a nivel de análisis

¿Por qué el software es complejo de forma innata?

La complejidad inherente al software se deriva de los siguientes cuatro elementos

- La complejidad del dominio del problema
- La dificultad de gestionar el proceso de desarrollo
- El detalle que se puede alcanzar a través del software
- El problema de caracterizar el comportamiento de sistemas discretos[1]

1- Los sistemas discretos son aquellos que operan por intervalos definidos, por ejemplo: “El ascensor de un edificio”, “El encendido de las luces de una habitación”, “El registro de hemoglobina de un individuo”, “El sistema de bombeo de gasolina de un automóvil”.

Aprendemos: Requerimientos – Insumo a nivel de análisis

La complejidad del dominio del problema

- Gran cantidad de requisitos que compiten entre sí, incluso contradiciéndose
- Desacoplamientos de impedancias entre usuarios del sistema y desarrolladores
  - Los usuarios suelen tener ideas vagas de lo que desean
  - Dificultades de comunicación
  - Distintas perspectivas de la naturaleza del problema.
- Modificación de los requisitos con el paso del tiempo, pues los usuarios y desarrolladores comienzan a compenetrarse mejor
  - Mantenimiento de software: Cuando se corrigen errores
  - Evolución del software: Cuando se responde a requisitos que cambian
  - Conservación del software: Se emplean medios extraordinarios para mantener en operación un elemento software anticuado y decadente

Aprendemos: Requerimientos – Insumo a nivel de análisis

La dificultad de gestionar el proceso de desarrollo

- Dirigir un equipo de desarrolladores
  - mantener la unidad de acción
  - conservar la integridad del diseño.
- Manejar gran cantidad de código
  - Uso de herramientas
    - “Gestión de proyectos”, “Análisis, diseño e implementación”, “Desarrollo rápido de prototipos (RAD)”, “Control de versiones”
  - Uso de lenguajes de muy alto nivel
  - Reutilización de código
  - Uso de frameworks (marcos estructurales)
  - Uso de componentes software
  - Uso de patrones de diseño

Aprendemos: Requerimientos – Insumo a nivel de análisis

El detalle que se puede alcanzar a través del software

“No vuelvas a inventar la rueda”

- Evitar el desarrollo de todo desde el nivel más inferior
- Utilizar los estándares, al igual que en otras ingenierías
- Verificar la fiabilidad de los estándares existentes en el mercado

Aprendemos: Requerimientos – Insumo a nivel de análisis

El problema de caracterizar el comportamiento de sistemas discretos

- La ejecución de un programa es una transición entre estados de un sistema discreto
- Cada estado contiene las variables, su valor, direcciones en tiempo de ejecución, etc.
- Diseñar el sistema de forma que el comportamiento de una parte del sistema tenga mínimo impacto en otra parte del mismo
- La transición entre estados debe ser determinista
  - Sin embargo, a veces no lo es, pues un evento puede corromper el sistema pues sus diseñadores no lo tuvieron en cuenta
- Es imposible hacer una prueba exhaustiva
  - Es decir, probar todos los casos posibles pues, aunque el número de estados es finito el determinar todas las combinaciones posibles produce un número tan elevado de combinaciones que es imposible de probar cada una de ellas

Aprendemos: Requerimientos – Insumo a nivel de análisis

Las consecuencias de la complejidad ilimitada

"Cuanto más complejo es un sistema más probable es que se caiga"

- La crisis del software
  - Son los sucesivos fracasos de las distintas metodologías para dominar la complejidad del software, lo que implica el retraso de los proyectos de software, las desviaciones por exceso de los presupuestos fijados y la existencia de deficiencias respecto a los requisitos del cliente[1]
  - Este término alude al conjunto de problemas que aparecen en el desarrollo de software[2]

1- [Booch 94]
2- [Pressman 97, apartado 1.3]

Aprendemos: Requerimientos – Insumo a nivel de análisis

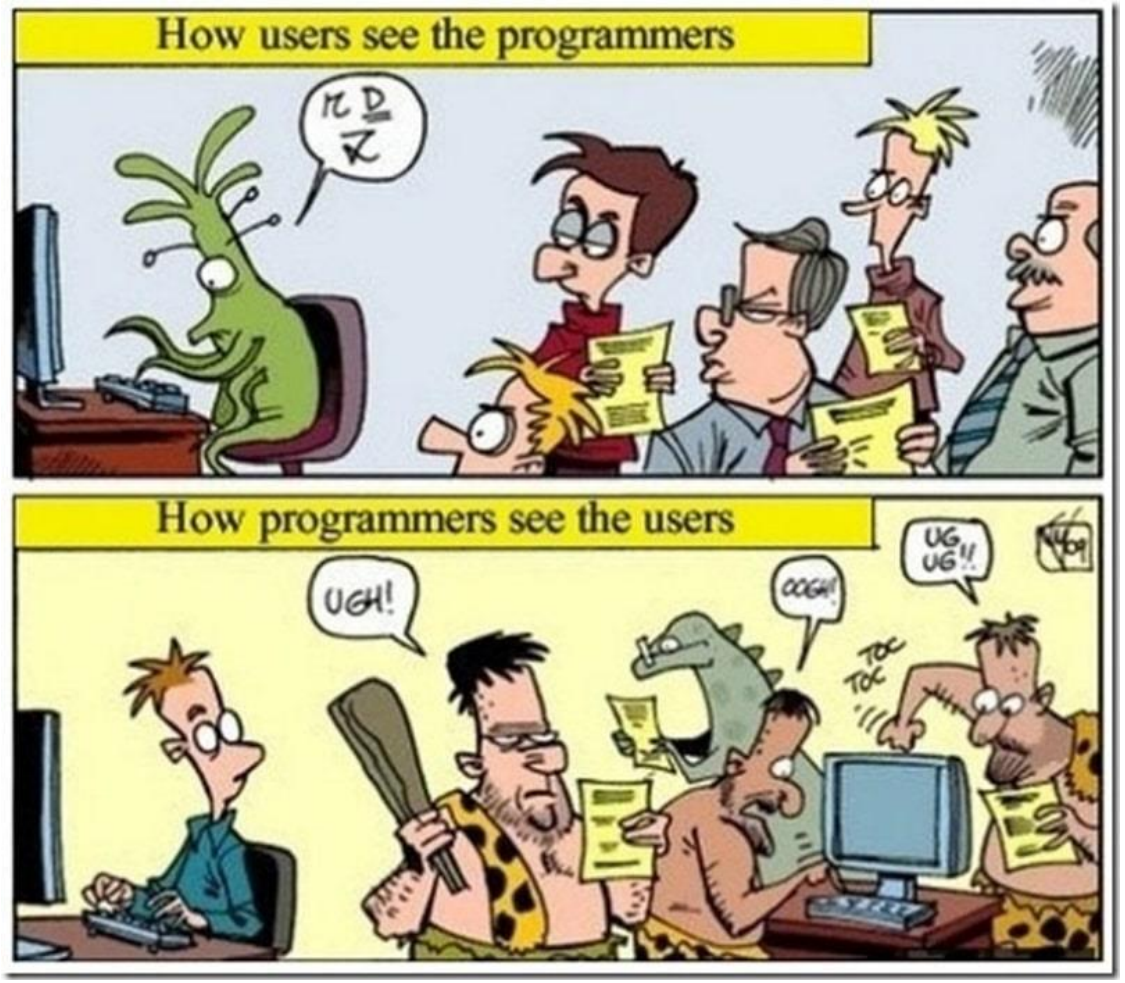
Las consecuencias de la complejidad ilimitada (Cont.)

- La crisis del software (Cont.)
  - Muchas de las causas de la crisis del software son los mitos del software[1]
    - Mitos de gestión
      - Si fallamos en la planificación, podemos añadir más programadores y adelantar el tiempo perdido
    - Mitos del cliente
      - Una declaración general de los objetivos es suficiente para comenzar a escribir programas; podemos dar los detalles más adelante
    - Mitos de los desarrolladores
      - Una vez que escribimos el programa y hacemos que funcione nuestro trabajo ha terminado
      - Hasta que no se ejecuta el programa no hay forma de comprobar su calidad
      - Lo único que se entrega al terminar el proyecto es el programa funcionando

1- [Pressman 97, apartado 1.4]

Aprendemos: Requerimientos – Insumo a nivel de análisis

Las consecuencias de la complejidad ilimitada (Cont.)



Aprendemos: Requerimientos – Insumo a nivel de análisis

El proceso de desarrollo de software

- No hay recetas mágicas, aunque es necesario tener un proceso preceptivo.
- Las características fundamentales de un proyecto con éxito:
  - Buena visión arquitectónica
    - No existe ningún camino bien definido para idear una arquitectura. Tan sólo se pueden definir los atributos de una buena arquitectura:
      - Capas de abstracción bien definidas
      - Clara separación de intereses entre interfaz e implementación
      - Arquitectura simple
    - Es necesario distinguir entre decisiones estratégicas y tácticas
      - **Decisiones estratégicas** es aquella que tiene amplias implicaciones estratégicas e involucra así a la organización de las estructuras de la arquitectura al nivel más alto

El proceso de desarrollo de software

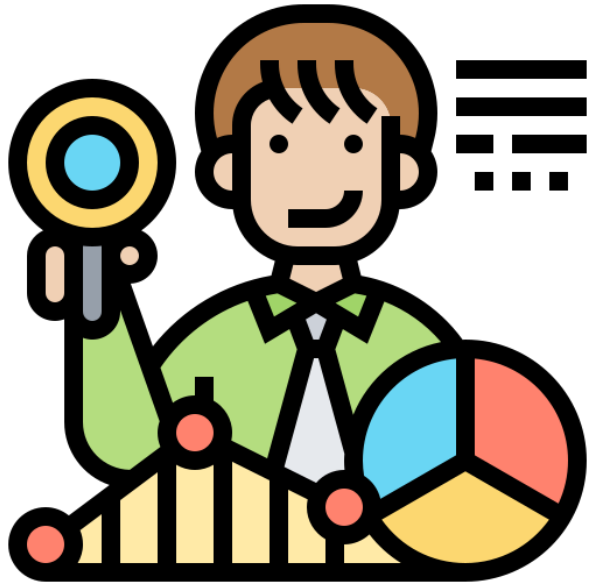
- **Decisiones tácticas** son las que sólo tienen implicaciones arquitectónicas locales, es decir sólo involucran a los detalles de interfaz e implementación de una clase

- Ciclo de vida incremental e iterativo
  - Los ciclos de desarrollo no deben ser anárquicos ni excesivamente rígidos
  - Cada pasada por un ciclo análisis/diseño/evolución lleva a refinar gradualmente las decisiones estratégicas y tácticas, convergiendo en última instancia hacia una solución con los requisitos reales de los usuarios finales (habitualmente no expresados explícitamente)

Aprendemos: Requerimientos – Insumo a nivel de análisis

Detalle de un proceso de desarrollo de software

- **Análisis**
  - Documentos de análisis
  - Especificación de requisitos o requerimientos
  - Diagramas de casos de uso
  - Escenarios y sub-escenarios
  - Prototipos



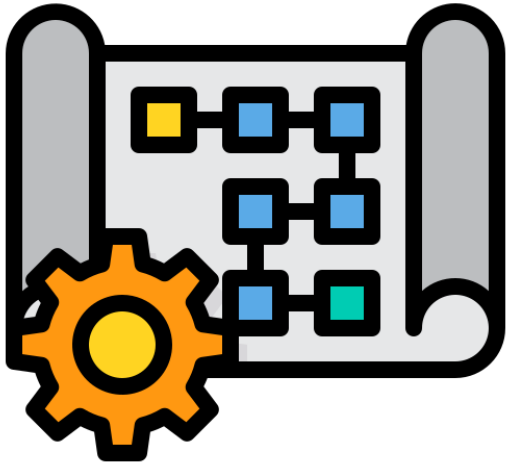
Aprendemos: Requerimientos – Insumo a nivel de análisis

Detalle de un proceso de desarrollo de software

- **Diseño (preliminar y detallado)**
  - División en módulos y para cada módulo
    - Modelado de Clases, Objetos y mecanismos de colaboración
      - Diagramas de interacción
        - Diagramas de secuencia
        - Diagramas de colaboración
      - Diagramas de Clases y consulta de patrones de diseño.
      - Diagramas de objetos
    - Modelado del comportamiento de clases y objetos
      - Diagramas de actividades
      - Diagramas de estados
  - Construcción del modelo físico
    - Diagramas de componentes
    - Diagramas de despliegue

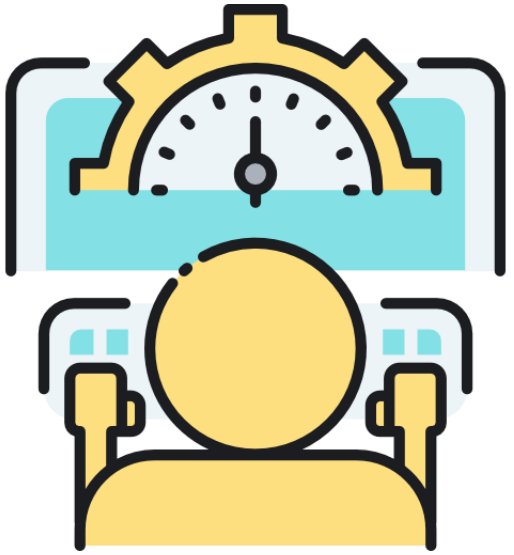
Detalle de un proceso de desarrollo de software

- **Implementación**
  - Las decisiones iniciales de implementación se toman a partir de los diagramas de componentes y de despliegue
  - Se implementan las clases de un componente a partir de los diagramas de clases y diagramas de objetos
  - A partir de los diagramas de actividades y de los diagramas de estados se implementa el comportamiento de los métodos de cada clase



Detalle de un proceso de desarrollo de software

- **Prueba**
  - Prueba unitaria de cada clase
  - Prueba unitaria de módulos
  - Prueba de integración se realiza siguiendo los escenarios, diagramas de interacción, actividades y estados



Detalle de un proceso de desarrollo de software

- **Mantenimiento**
  - Informes de errores
  - Nueva especificación de requisitos.
    Nueva versión

Aprendemos: Requerimientos – Insumo a nivel de análisis

Análisis Orientado a Objetos

"Es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema"
[Booch 1994].

- Documentos básicos de análisis orientado a objetos.
  - Documentos de análisis
    - Documentación y actas de reuniones
  - Especificación de requisitos o requerimientos
  - Diagramas de casos de uso
  - Escenarios y sub-escenarios
  - Prototipos y su evaluación
- Todos los documentos deben estar identificados y codificados.

Aprendemos: Requerimientos – Insumo a nivel de análisis

Identificación

- Es necesario identificar todos los elementos del proceso de desarrollo de software de una forma unívoca
- Todos los documentos deben estar identificados
- Título.
  - Debe reflejar de la mejor forma posible sus fines y su funcionalidad
- Descripción
- Autores
- Versión. Notación decimal.
- Revisión. Autores
- Fecha
- Código de cada documento o diagrama.

Documentos de análisis

- Contiene la documentación que aporta el cliente que encarga la aplicación
- También contiene las actas de las reuniones de trabajo del grupo de análisis.
  - Es necesario un secretario que tome acta
  - Es necesario aprobar el acta de cada reunión por todos los miembros

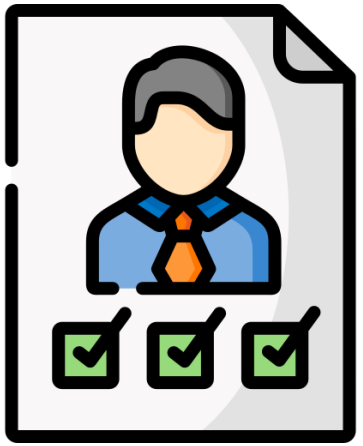
Aprendemos: Requerimientos – Insumo a nivel de análisis

Especificación de requisitos o requerimientos

"La captura de requisitos es el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir"

[Jacobson 1999, capítulo 6].

- La captura de requisitos es complicada.
  - Los usuarios habitualmente no saben expresar exactamente lo que quieren
  - Es difícil tener una visión global del problema a resolver
- La especificación de requisitos es un documento más técnico y elaborado de los documentos de análisis
- Es importante codificar los requisitos para poder seguirlos a lo largo del proceso de desarrollo de software.



Aprendemos: Requerimientos – Insumo a nivel de análisis

Especificación de requisitos o requerimientos

- Se puede utilizar una especificación jerárquica
  - Están todos codificados por niveles, al igual que las leyes.
  - Se desea que en las actas quede reflejado lo más exactamente posible el problema a resolver, y que en las reuniones de análisis se determine exactamente que requisitos se añaden o se eliminan
  - Los requisitos relacionados se organizan dentro de un mismo nivel
  - Cada nivel 1 se puede hacer corresponder posteriormente con un caso de uso
  - Cada nivel 2 se puede hacer corresponder posteriormente con un escenario
  - Cada nivel 3 se puede hacer corresponder posteriormente con un sub-escenario

Aprendemos: Requerimientos – Insumo a nivel de análisis

Diagramas de Casos de Uso

- Es uno de los cinco tipos de diagramas de UML que se utilizan para el modelado de los aspectos dinámicos de un sistema.
- Se corresponden inicialmente con requisitos de primer nivel. Posteriormente se van modelando requisitos de los siguientes niveles.
- Se suelen codificar con el mismo código que el requisito, para hacer más patente su correspondencia.
- Un caso de uso es una técnica de modelado utilizada para describir lo que un nuevo sistema debe hacer o lo que un sistema existente ya hace.
- Los casos de uso representan una vista externa del sistema
- Un modelo de casos de uso se construye mediante un proceso iterativo durante las reuniones entre los desarrolladores del sistema y los clientes (y/o los usuarios finales) conduciendo a una especificación de requisitos sobre la que todos coinciden.

Aprendemos: Requerimientos – Insumo a nivel de análisis

Diagramas de Casos de Uso - Escenarios y sub-escenarios

- Cada caso de uso da lugar múltiples escenarios
- Se codifican siguiendo la codificación de los casos de uso
- Se estudia cada escenario utilizando guiones como los que se usan en el cine
- Cada equipo que pasa por un escenario identifica los objetos y sus responsabilidades, así como los mecanismos que relacionan los objetos
- De los escenarios iniciales se puede pasar a otros escenarios secundarios
- Los escenarios también se pueden utilizar para probar el sistema en la fase de pruebas
- El estudio de los escenarios con detalle permitirá enriquecer el Diccionario de clases
- No es necesario hacer todos los escenarios y sub-escenarios posibles si se observa que no enriquecen el diccionario de clases

Aprendemos: Requerimientos – Insumo a nivel de análisis

Diagramas de Casos de Uso - Escenarios y sub-escenarios - Ejemplos

Caso de uso 1: Gestión de Clientes

Nombre de Escenario 1.1:	Dar de alta un cliente eventual
Precondiciones:	–No existe ficha de cliente.
Postcondiciones:	–Todos los datos se han introducido correctamente. –El numero de clientes se incrementa en uno
Excepciones:	
Iniciado por:	Dependiente/Administrador.
Finalizado por:	Dependiente/Administrador.
Detalle operaciones:	–Cliente acude a una tienda de la compañía. –Dependiente (ó Administrador) obtiene datos de cliente. –Dependiente (ó Administrador) introduce ficha en el sistema con los datos *número, dni, nombre, dirección, ciudad, teléfono*, y *departamento.*

Nombre de Escenario 1.2:	Dar de alta un cliente fijo.
Precondiciones:	–No existe ficha de cliente.
Postcondiciones:	–Todos los datos se han introducido correctamente. –El número de clientes se incrementa en 1
Excepciones	
Iniciado por:	Administrador.
Finalizado por:	Administrador.
Detalle operaciones:	–Cliente acude a una tienda de la compañía. –Administrador obtiene los datos del cliente. –Administrador introduce ficha en el sistema con los datos *número, dni, nombre, dirección, ciudad, teléfono*, y *departamento.*

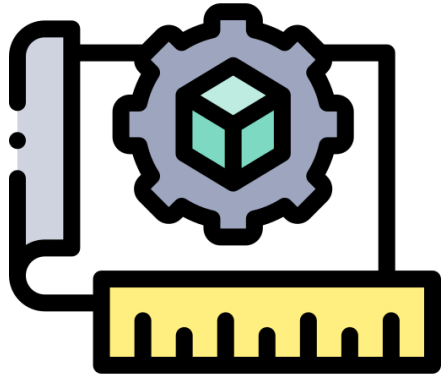
Aprendemos: Requerimientos – Insumo a nivel de análisis

Prototipos

El prototipado consiste en la elaboración de un modelo o maqueta del sistema que se construye para evaluar mejor los requisitos que se desea que cumpla.

Es particularmente útil cuando:

- El área de la aplicación no está bien definida, bien por su dificultad o bien por falta de tradición en su automatización.
- El coste del rechazo de la aplicación por los usuarios, por no cumplir sus expectativas, es muy alto.
- Es necesario evaluar previamente el impacto del sistema en los usuarios y en la organización



Prototipos

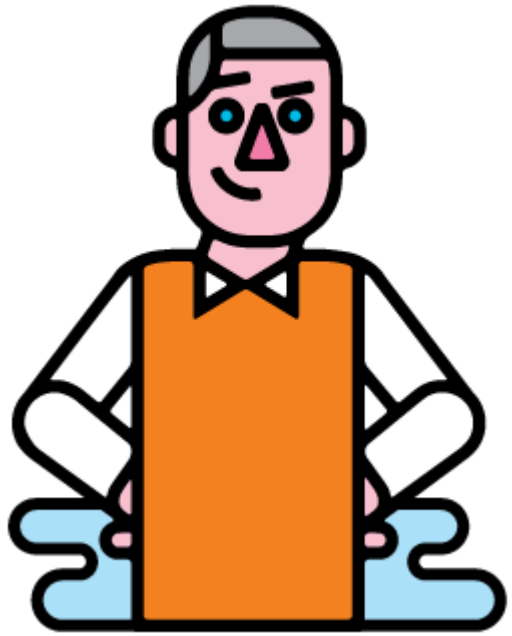
Estos modelos o prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación pedida.
Existen tres razones principales para emplear prototipado, ordenadas por frecuencia de uso:

- **Prototipado de la interfaz de usuario** para asegurarse de que esta bien diseñada, que satisface las necesidades de quienes deben usarlo.
- **Modelos de rendimiento** para evaluar el posible rendimiento de un diseño técnico, especialmente en aplicaciones críticas en este aspecto.
- **Prototipado funcional**. Cada vez más utilizado, está relacionado con un ciclo de vida iterativo.

Aprendemos: Requerimientos – Insumo a nivel de análisis

Otras técnicas de análisis orientado a objetos

- Análisis OO mediante fichas CRC (Clases/Responsabilidades/Colaboradores).
- Descripción informal en lenguaje natural.



Aprendemos: Requerimientos – Insumo a nivel de análisis

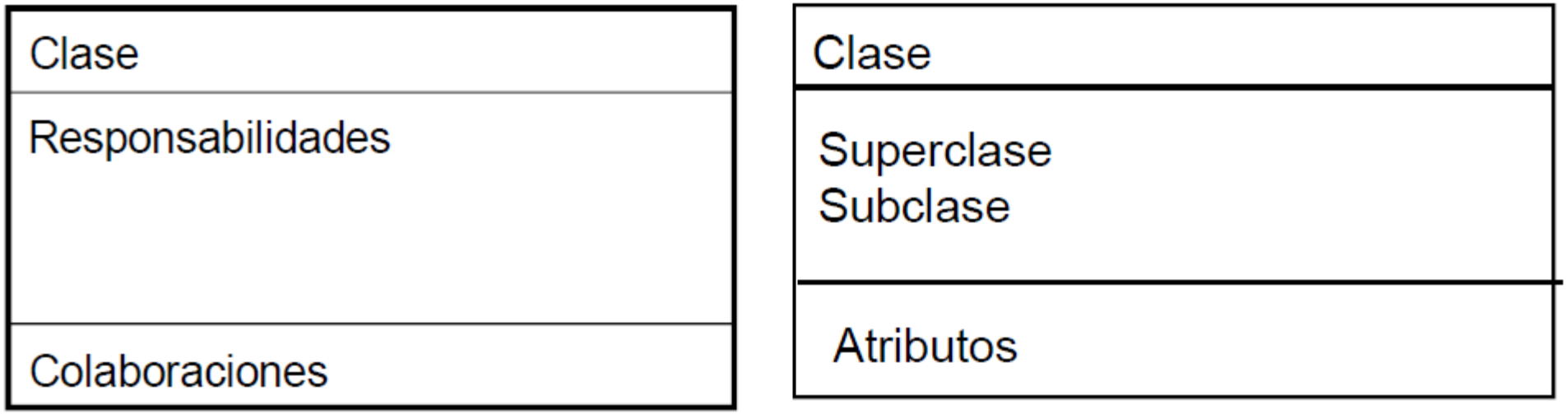
Análisis mediante fichas CRC (Clases/Responsabilidades/Colaboradores)

- Es una forma simple de analizar escenarios
- Son muy útiles para la enseñanza del AOO, DOO y POO
- Facilitan las “tormentas de ideas” y la comunicación entre desarrolladores
- Se crea una ficha para cada clase que se identifique como relevante en el escenario
- A medida que el equipo avanza puede dividir las responsabilidades
- Las fichas CRC pueden disponerse espacialmente para representar relaciones de colaboración
- Las fichas CRC también se pueden colocar para expresar jerarquías de generalización/especialización
- No están contempladas en UML

Aprendemos: Requerimientos – Insumo a nivel de análisis

Análisis mediante fichas CRC (Clases/Responsabilidades/Colaboradores)

Ficha CRC (anverso y reverso)



Aprendemos: Requerimientos – Insumo a nivel de análisis

Análisis mediante fichas CRC (Clases/Responsabilidades/Colaboradores)

Ejemplo de ficha CRC

Clase: Reunión
Responsabilidades Planificar Comprobar la sala asignada Conocer hora de comienzo Conocer la fecha Conocer número de asistentes Conocer equipamiento necesario
Colaboraciones Sala de conferencias Organizador de reuniones

Clase: Reunión
Superclase: Subclases: Reunión de trabajo, Junta de Escuela, Clase de un curso
Atributos: Orden del día Lugar Fecha Hora de inicio Asistentes Equipo necesario

Aprendemos: Requerimientos – Insumo a nivel de análisis

Descripción informal en lenguaje natural

- Subrayar los nombres y los verbos de la descripción del problema
- Los **nombres** representan los objetos candidatos
- Los **verbos** representan las operaciones candidatas.
- Ventajas:
  - Obliga al desarrollador a trabajar sobre el vocabulario del espacio del problema
  - Es sencillo
  - Es didáctico
- Inconvenientes:
  - No es riguroso, al ser el lenguaje natural ambiguo.
  - Es compleja su aplicación a grandes proyectos.

Aprendemos: Requerimientos – Insumo a nivel de análisis

EJERCICIO: Realizar el Diagrama de Casos de Uso, uno por grupo

Realizar el análisis del juego del ajedrez. Pueden jugar dos personas entre sí o una persona contra la Computadora. En este último caso debe ser posible seleccionar el nivel de dificultad entre una lista de varios niveles. El juego de ajedrez permitirá al jugador elegir el color de las piezas. La aplicación deberá permitir detectar los movimientos ilegales de las piezas, tiempos utilizados cada jugador, registro de jugadas y piezas perdidas. También determinará si se alcanzan tablas, y permitirá abandonar la partida a un jugador.

Verificamos lo aprendido:

Kahoot!

http://www.kahoot.it/

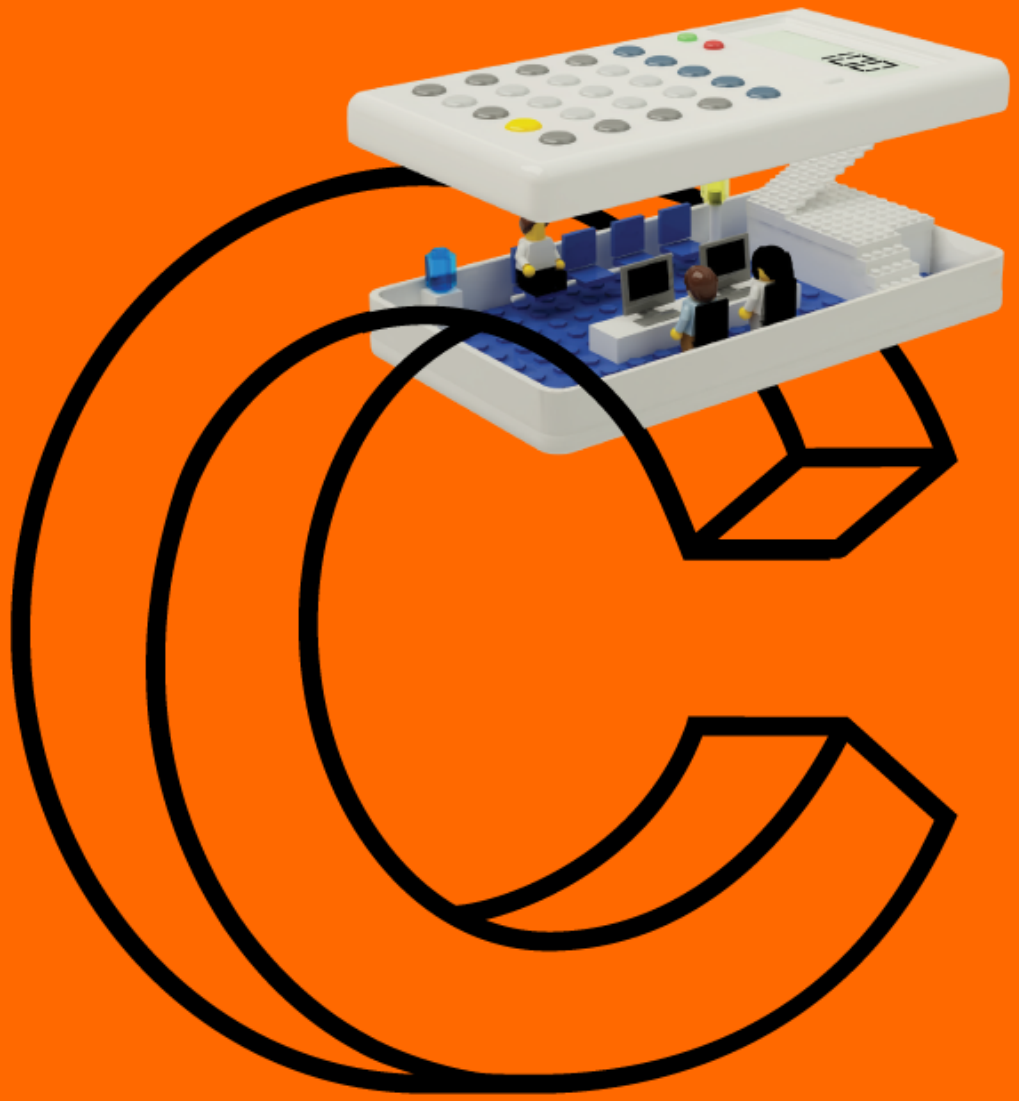
Verificamos lo aprendido:

- Absolución de Preguntas.
- Conclusiones y Recomendaciones.

Aplicamos lo aprendido: Asignación para la clase siguiente

- Comentar en el Foro

Gracias



idat