Resumen de IS2:

# POO

Fundamentos:

* Escribir software fácilmente escalable y mantenible.
* Trasladar el dominio del software a un lenguaje más natural.
* Escribir software reutilizable.

# UML

Es UN LENGUAJE, NO es una metodología. Es independiente del proceso, aunque lo agiliza.

Viene a ayudarnos a entender mejor la arquitectura del software y facilitar la comunicación.

Es una técnica de análisis orientado a objetos.

Expresa: Requisitos, Funcionalidades, comportamiento interno del sistema.

Otra forma de definirlo es: Representa el modelo funcional, de objetos y dinámico.

Muestran una abstracción del sistema, no todo.

Enfocado a distintos puntos de vista.

Permite la generación de código a través del diagrama.

Sirve para: Comunicarse entre devs, clientes y usar herramientas de generación automática de código.

Es estandarizado por **OMG**.

Es **extensible, flexible, y escalable.**

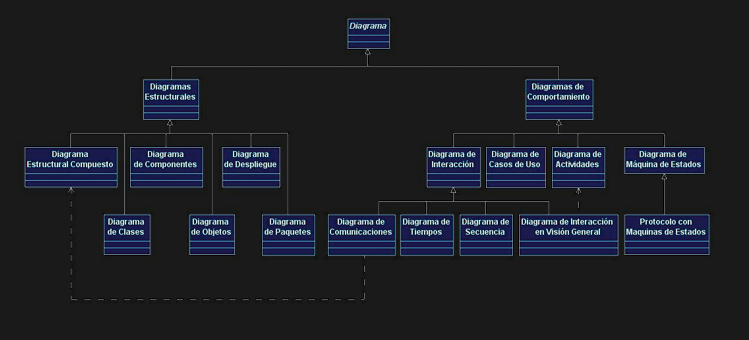
Podemos considerarlo como una forma visual de modelizar el negocio y sus reglas.

**Parálisis de análisis**: Cuando no se exagera el modelo hasta un punto que no es productivo.

Cada diagrama tiene un propósito especifico

Están constituidos por la notación y los diagramas / objetos.

Árbol de diagramas:



Diagramas de Interacción: Diagramas que engloban el intercambio de mensajes entre objetos.

Ahí podés encontrar el diagrama de Secuencia, comunicaciones, diagrama de interacción en visión general, de tiempos.

Diagramas de CU, Diagrama de actividades NO es parte del diagrama de interacción.

Diagrama de máquinas de estados = diagramas de estados ¿  
Diagramas de Comportamiento: Muestran cómo cambia el sistema con el tiempo.

Diagramas estructurales: Muestran la arquitectura del sistema.

# Abstracción

Proceso de modelar un sistema complejo de forma sencilla y generalizada.

Solo se muestra una parte del sistema, no todo en el UML.

# Puntos de vista

Con esto se refiere a los distintos diagramas que tenemos, cada diagrama es un punto de vista distinto.

# Casos de Uso

Reflejan los requerimientos funcionales, y como el usuario puede interactuar con el sistema.

Es una secuencia de interacciones. Se usa para mostrar los servicios que puede usar el usuario, y como los usa, pero NO como los implementa el sistema, por lo tanto, tiene un nivel de visión ALTO del sistema.

Se identifican factores internos y externos.

Están más relacionados con el Proceso Unificado de Desarrollo, no tanto con el UML.

Propone que la mejor forma de empezar a entender un sistema, es a partir de los servicios a los cuales pueden acceder los usuarios, y no tanto en los componentes internos.

Son ajenos a la programación que se use y al método de diseño.

Es ideal hacerlo previo a diseñar el sistema estructurado.

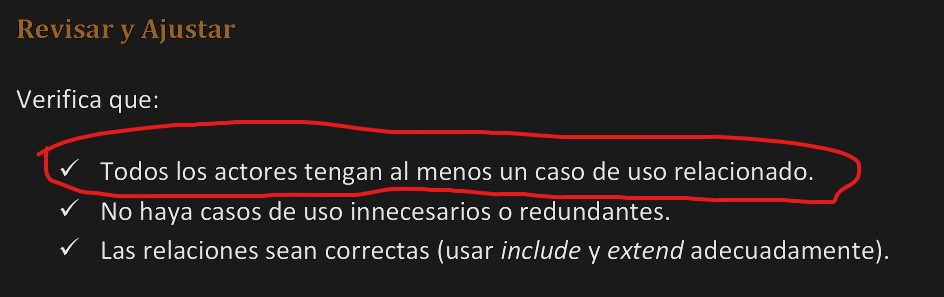
Los casos de uso no muestran realmente el orden en el que se acceden o usan funcionalidades, tampoco muestra cambios en el tiempo. El caso de uso no busca el detalle, sino la simplicidad.

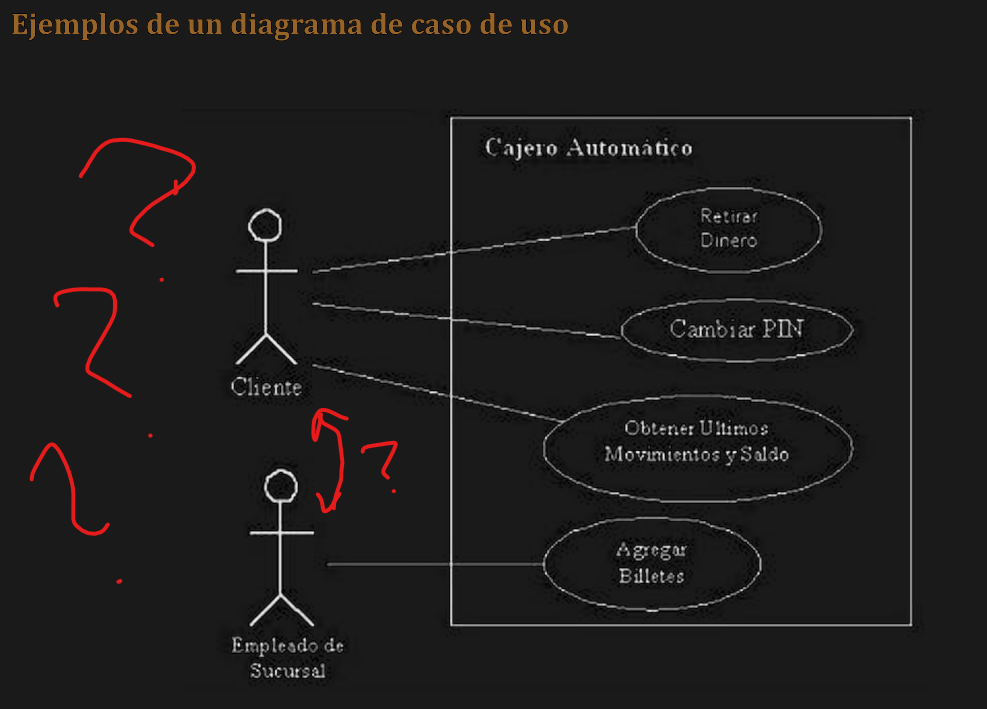
Características:

* Están acotados a una funcionalidad diferenciada del sistema.
* Solo los puede iniciar un actor.
* El texto que lo refleja debe ser informal.
* El énfasis está en la interacción.
* Son desde el punto de vista del actor.

Se compone del actor, de las funcionalidades y relaciones. Todo esto dentro del Sistema.

El sistema/límite de sistema define el alcance del caso de uso y se representa como un rectángulo.





## Especificación del caso de uso

Se usa una tabla o texto estructurado con los siguientes elementos:

1. Nombre del Caso de Uso: Un título claro y conciso.

2. Descripción: Explica el propósito del caso de uso.

3. Actores: Usuarios o sistemas externos que interactúan.

4. Precondiciones: Requisitos que deben cumplirse antes de ejecutar el caso de uso.

5. Flujo Normal (Escenario Principal): Pasos secuenciales de la interacción esperada.

6. Flujo Alternativo: Variaciones del proceso normal.

7. Flujo de Excepción: Errores o situaciones inesperadas.

8. Postcondiciones: Estado final del sistema tras completar el caso de uso.

9. Reglas de negocio: Restricciones o políticas aplicables.

# Diagrama de clase

Es el más común. Muestra la relación entre una clase y otra, pero no el que y como lo hacen.

El diagrama de clases muestra cómo se relacionan las clases entre sí, que comportamientos tienen y de que se componen, pero no muestra información temporal.

Sus elementos básicos son las **clases** y las **relaciones**.

Tipos de objetos:

* Objetos entidad
* Boundary Objects (Objetos limíte/Interfaz)
* Objetos de control

## Componentes raros

Visibilidades raras:

# protegido ~ paquete. (Protegido es ideal para la herencia)

Estereotipos: Nos permiten extender el lenguaje UML con cosas que no son propias de este, como los <<enum>> o las <<interfaces>>

Restricciones: Muestran limitaciones o condiciones de negocio, pre y postcondiciones, se pueden poner notas o en un mismo elemento. Se usa { } para denotarlas.

Realización: Implementación de un estereotipo, muestra una implementación

# Diagrama de Estados

Diagrama dinámico, perteneciente a los diagramas de comportamiento.

Se conoce el número de estados de antemano, nos centramos en el comportamiento.

Nos sirve para mostrar el comportamiento del sistema, ya sea por un caso de uso, como funciona una clase o el sistema entero.

El estado lleva en su interior ACTIVIDADES.

Representa los diferentes estados por los que pasa un objeto a lo largo de su vida útil (ciclo de vida), con los eventos o condiciones que lo hacen cambiar de estado.

Tiene 5 componentes, Estados, Transiciones entre estados, eventos, inicio, y final.

Puede representar casos de uso, comportamientos del sistema, y funcionamientos de clase.

# Diagrama de secuencia

Muestra intercambio de mensajes entre objetos a lo largo del tiempo. Los objetos no tienen por qué ser instancias de las clases, sino que también puede representar componentes del sistema.

Depende el nivel de abstracción se puede usar un lenguaje más natural o más cercano al a programación.

Mensaje llamada: Envía mensaje, espera respuesta.

Mensaje asíncrono: No espera respuesta

Auto mensaje: Mensaje a si mismo

Mensaje recursivo: Se usa recursión // no entiendo cuál es la diferencia entre este y el anterior.

Mensaje encontrado: Mensaje que llega literalmente de un random.

Mensaje perdido: Mensaje que se va afuera, que no es parte del diagrama. (lo contrario al anterior)

# Diagrama de comunicación

Se enfoca más en el orden en el que se van a dar los mensajes, que en el tiempo. Es decir, que sabremos el orden, pero no cuando.

Es bastante parecido al de secuencia, pero con ligeras diferencias.

Representa objetos instancias conectados por enlaces (relaciones).

Los mensajes tienen flechas numeradas para saber los órdenes.

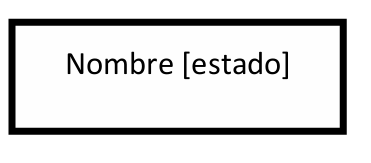
Muy útil para visualizar como se comunican objetos entre sí.

# Diagrama de Objetos

Muestra cómo se relacionan las instancias de las clases en un momento determinado.



# Diagrama de actividad

Muestra casos de uso o procesos, mostrando como se desarrolla una actividad.   
La actividad es un proceso, cada paso del proceso es una relación.  
Nodo objeto: 

# RUP

* Rational Unified Process (*Proceso Racional Unificado)*

Es la más usada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

Es un conjunto de metodologías adaptables al contexto de la organización.

Es normalmente usado en teams medianos/grandes de desarrollo. Para proyectos complejos.

Esta relacionado con el UML => Desarrollo de Software + UML = RUP

Características clave:

* Orientado a CU

Nos proporciona una orientación para empezar a desarrollar el sistema, basándonos en las funcionalidades que tendrá el mismo.

* Orientado a la Arquitectura

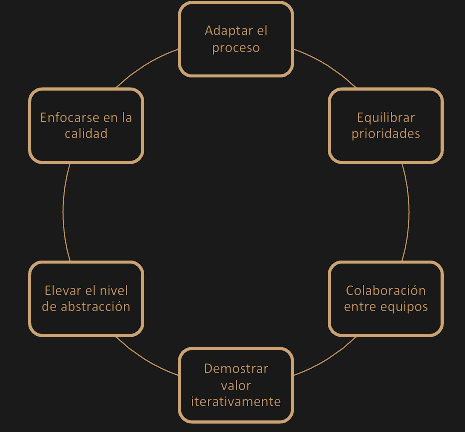
Promueve un desarrollo de la arquitectura temprana. La arquitectura esta relacionada con los CU, los CU deben encajar en la arquitectura y la arquitectura debe poder permitir que se realicen todos los CU.

* Iterativo e Incremental

Divide el desarrollo del proyecto en etapas, o mini proyectos. Esto lo hace mas flexible al cambio. Cada etapa es una iteración que conlleva a un incremento.



## Principios (6)



El RUP se puede dividir en 4 etapas, donde se van trabajando las 8 disciplinas que posee:

Disciplinas:

* Modelado de negocio
* Ingeniería de requerimientos
* Análisis
* Diseño
* Implementación
* Pruebas
* Gestión de configuración
* Gestión del proyecto
* Ambiente

Etapas:

* Inicio: Definir alcance del proyecto y Viabilidad.

Acá se desarrolla más el modelado del negocio y los requerimientos.

* Elaboración: Reducir riesgos y definir arquitectura.

Se suma el Análisis y el Diseño

* Construcción: Construir el producto completo o sólido.

Se genera documentación como el manual de Usuario.

Se suma la Implementación, las Pruebas y el Despliegue.

* Transición: Se entrega el software al usuario final.

Esta etapa conlleva toda la documentación técnica completa.

Junto a toda la documentación en general.



Resumen, el RUP tiene 4 fases bien marcadas, donde cabe resaltar que, en las distintas disciplinas, la Gestión de Configuración, Gestión de Proyecto y Ambiente se aplican exclusivamente a la transición.

# Metodologías Ágiles

## ¿Qué son?

Son estrategias, cuyo principio es la mejora continua empresarial. Estas estrategias se las conoce como “Agile Project Management”.

Conllevan constante innovación y renovación. Se implementan durante las actividades.

Se centran en el cliente.

## Manifiesto ágil

Acá se valora más:

* Individuos y su interacción, por encima de los procesos y herramientas.
* Software que funciona, por encima de la documentación exhaustiva.
* Respuesta al cambio, por encima del seguimiento de la planificación.
* Colaboración con el cliente por encima de la relación contractual.
* Documentación y modelado minimalistas.

12 principios del manifiesto Ágil:

1. Cliente como objetivo.
2. Flexibilidad.
3. Produccion de prototipos.
4. Integracion de los miembros del equipo.
5. Motivacion.
6. Comunicación.
7. Funcionalidad como indicador de progreso.
8. Soporte.
9. Excelencia.
10. Sencillez.
11. Coordinacion.
12. Aprendizaje.

## Artefacto

Cualquier producto tangible generado durante el proceso de desarrollo, como documentación, backlogs o el código fuente.

## Tipos de metodologías agiles.

Scrum: Enfocado a la gestión.

Lean: Enfocado a maximizar el valor para el cliente y minimizar desperdicio en los procesos de producción.

XP: Enfocada a la parte técnica.

Kanban: Utiliza el tablero Kanban, que tiene 3 columnas (que se realizó, que se está realizando y que no). El enfoque es dividir en etapas el desarrollo y destacar las más importantes por terminar. No necesariamente requiere sprints o iteraciones.

Agile Inception: Propicia la iniciativa de las actividades y proposición de objetos por parte de todos los integrantes del proyecto.

Design Sprint: Diseña un prototipo o sprints del producto final, para mostrarlo al público y recibir retroalimentación para continuar desarrollando el proyecto.

## Pasos y etapas

Aunque no tiene fases como las metodologías anteriores, se recomiendan estas etapas para una mejor coordinación del proyecto:

* Análisis de requisitos
* Planificación de actividades
* Reunión del equipo
* Producción
* Análisis, documentación y lanzamiento.

## Roles principales:

* Agile Coach: Relacionado con la capacitación del capital humano. Garantiza excelencia en los medios para trabajar.
* Scrum Master: Garantiza el cumplimiento del esquema planificado de actividades y proyectos.
* Product Owner: Se encarga del correcto desarrollo del producto.
* Project Manager: Gestiona el proyecto en ejecución.

## ¿Como se aplica una metodología ágil?

* Se identifican las demandas del público.
* Se proponen soluciones.
* Reunir todas las ideas en un solo proyecto.
* Planificar actividades y reunir al equipo.
* Coordinar los miembros del equipo y sus procesos enfocado al objetivo.
* Ir verificando los procesos y sugerir cambios acordes a los requerimientos del proyecto.
* Lanzamiento de prototipo.
* Obtener retroalimentación.
* Aplicar los cambios.

# Scrum

Es un framework de la metodología ágil. Posee un conjunto de reuniones y herramientas para ayudar a los equipos a estructurar y coordinar el trabajo más eficientemente.

Aquí las iteraciones se las conoce como **Sprint**, donde la idea del sprint es dividir un proyecto grande en proyectos pequeños para avanzar hasta completar el proyecto final.

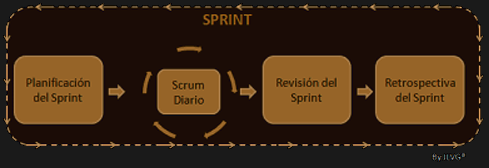
## Roles

* Scrum Master

Lo abarcamos antes, es quien garantiza el cumplimiento de la planificación y quien lidera los equipos de gestión ágil.

* Development Team
* Product Owner

## Como planificar un sprint



1. Planificación de sprint

Se plantea: Que trabajo se puede hacer en este sprint, y cuando se puede hacer.

1. Preparar y ejecutar sprint

Se decide cuanto tiempo va a durar, el objetivo y por dónde empezar.

## Protocolos

1. Sprint Planning

Antes definido.

1. Daily Scrum

Es del development team, comentan que hicieron ayer, que van a hacer hoy y si tienen algún problema para llegar a la meta.

1. Sprint Review

Se muestra el software funcionando ante los stakeholders, los cuales pueden dar su retroalimentación. Ocurre al final del sprint.

1. Sprint Retrospective

Se reflexiona que se puede mejorar y hacer en el próximo sprint

1. Sprint Grooming / Refinement

Participan todos, es para refinar el Product Backlog

## Epics

Historias de usuario, que usualmente conforman la hoja de ruta. Dentro de una epic pueden haber múltiples historias de usuario, una epic puede estar orientada a un tipo de historias concreto.

## Backlog

Básicamente nos dicen que elementos se deben desarrollar en el proyecto, ordenados por prioridad, colocándose primero los de más alta prioridad.

Estos elementos son sacados de las hojas de ruta y los requerimientos.

El producto owner es quien se encarga de manejar el backlog, teniendo que actualizarlo al principio de las daylis o de los sprint planning.

El backlog se debe mantener y debe ser abierto a consulta por cualquier miembro del equipo, y puede incluir también cosas más técnicas como fallas pendientes a resolver, cambios de diseño, deuda técnica, solicitudes de los clientes, etc.

El backlog sirve perfectamente para planificar los sprints.

Puede sacarse cosas de forma secuencial como en el método de Kanban o por iteraciones mediante sprints como en el Scrum.

# Kanban

Esta enfocado a mejorar los procesos, a diferencia del Scrum que se enfoca en hacer más trabajos y más rápido. Todo mediante el tablero Kanban compuesto de 3 columnas, trabajo a realizar, realizándose y realizado.

## Principios (4)

* Empezar desde el inicio de lo que se está haciendo
* Respetar procesos y los roles del equipo
* Cambio progresivo y no brusco
* Impulsar liderazgo

## Practicas (6)

* Visualizar el trabajo (El tablero)
* Limitar el trabajo en curso
* Gestionar flujo de trabajo
* Implementar políticas de procesos explicitas
* Implementar ciclos de comentarios
* Mejorar colaborando y evolucionar experimentando

# Programación XP

Es una metodología, centrada en la rapidez y sencillez con ciclos de desarrollo cortos y alta colaboración entre desarrolladores y el cliente. Se podría decir que esta centrado a lo técnico.

Es muy flexible ante cambios. Es altamente colaborativa.

A diferencia de las metodologías anteriores, esta centrada exclusivamente en la programación.

## Características

* Iterativa: Con ciclos breves.
* Disciplinada: Alta rigidez técnica, enfocada a buenas prácticas y pruebas continuas.
* Colaborativa: Fuerte interacción entre devs y clientes.

## Valores

1. Simplicidad
2. Comunicación
3. Retroalimentación
4. Valentía
5. Respeto

## Ciclo de Vida

1. Se definen historias de usuario
2. Se priorizan funcionalidades
3. Realizar planificación iterativa.
4. Programar en ciclos cortos.
5. Probar y presentar avances.
6. Recoger feedback del cliente.
7. Volver a planificar.

Introduce una serie de reglas interesantes:

* Planificar basado en historias de usuario
* Programación en parejas
* Integración continua
* Pruebas unitarias automáticas desde el inicio
* Propiedad colectiva del código (Todos pueden modificar cualquier parte)

Reglas definidas por el PDF:

1. Planificación -> Evaluar si se adapta a la metodología XP
2. Gestión
3. Diseño -> Empezar con el diseño más simple posible
4. Codificación
5. Prueba