UNIDAD DIDÁCTICA I
TEMA 1 – Introducción

1. ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

1.1. Aplicación de UML y patrones en el A/DOO

El análisis y diseño orientado a objetos (A/DOO) permiten la creación de software de fácil mantenimiento, robusto y bien definido, utilizando las tecnologías y lenguajes de objetos. Para ello, se aplican el Lenguaje Unificado de Modelado (UML), **patrones** de buenas prácticas y el Proceso Unificado. Se asignan **responsabilidades** a las clases de objetos y se aplican principios y heurísticas que favorecen un aprendizaje más rápido y un uso de estilos de diseño de objetos.

El A/DOO (y todo el diseño del SW) está fuertemente relacionado con el **análisis de requisitos** (requisito previo) e incluye la escritura de **casos de uso**. El análisis de requisitos y el A/DOO requieren el contexto de algún proceso de desarrollo como el **proceso de desarrollo iterativo** del **Proceso Unificado**. En el diseño del SW hay pasos más allá del análisis de requisitos, el A/DOO y la POO como la ingeniería de usabilidad, el diseño de interfaces de usuario o el diseño de BBDD.

Una habilidad clave y fundamental en el A/DOO es la asignación cuidadosa de responsabilidades a los componentes SW. Esto influye sobre la robustez, mantenimiento y reutilización de los componentes SW. Existen 9 principios en el diseño de objetos y asignación de responsabilidades. Se organizan en una ayuda al aprendizaje denominada **patrones GRASP**.

El UML es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas SW, así como para el 1nodelado del negocio y otros sistemas no software. UML es una notación visual estándar de facto y de iure para el modelado orientado a objetos.

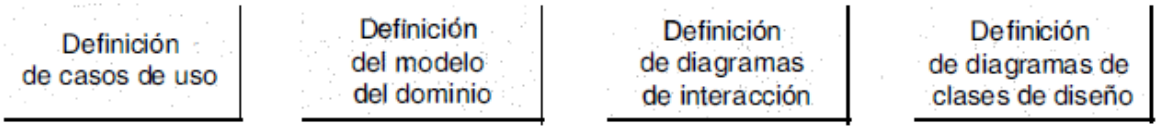
1.2. Análisis y diseño orientado a objetos

El **análisis** pone énfasis en la *investigación* del problema y los requisitos y no en poner una solución tanto en la parte de *análisis de requisitos* (estudio de los requisitos) como de *análisis de objetos* (estudio de los objetos del dominio) → hacer lo correcto.

El **diseño** pone énfasis en la *solución conceptual* que satisface los requisitos en vez de ponerlo en la implementación, diferenciando el *diseño de objetos* del *diseño de bases de datos* → hacerlo correcto.

El **análisis orientado a objetos** presta atención en encontrar y descubrir los objetos (conceptos) en el dominio del problema. El **diseño orientado a objetos** lo hace sobre la definición de los objetos software y en cómo colaboran para satisfacer los requisitos.

1.3. Pasos y diagramas clave en el A/DOO



Definición de casos de uso

El análisis de requisitos incluye una descripción de los procesos del dominio relacionados, representados como **casos de uso**. No son artefactos orientados a objetos. Herramienta muy popular en análisis de requisitos y parte importante del Proceso Unificado.

Definición de un modelo de dominio

El análisis orientado a objetos crea una descripción del dominio desde la perspectiva de la clasificación de objetos. Una descomposición del dominio conlleva una identificación de los conceptos, atributos y asociaciones que se consideran significativas, un **modelo del dominio**, que se ilustra mediante un conjunto de diagramas que muestran los objetos o

conceptos del dominio. No es una descripción de los objetos software, es una visualización de los conceptos en el dominio del mundo real.

Descripción de los diagramas de interacción

El diseño orientado a objetos define los objetos SW y sus colaboraciones mediante un **diagrama de interacción**, que muestra el flujo de mensajes entre los objetos software y la invocación de métodos. Los diseños de los objetos SW y los programas se inspiran en los dominios del mundo real, pero no son modelos directos o simulaciones del mundo real.

Definición de los diagramas de clases de diseño

La vista *dinámica* de las colaboraciones entre los objetos que se muestra mediante los diagramas de interacción se complementa con una vista *estática* de las definiciones de las clases mediante un **diagrama de clases de diseño**. A diferencia del modelo de dominio, este diagrama no muestra conceptos del mundo real, sino clases software.

2. DESARROLLO ITERATIVO Y EL PROCESO UNIFICADO

Un **proceso de desarrollo de software** describe un enfoque para la construcción, desarrollo y, posiblemente, mantenimiento del software. El **Proceso Unificado** es un proceso de desarrollo de software útil para la construcción de sistemas orientados a objetos. El Proceso Unificado (UP) combina las prácticas comúnmente aceptadas como "buenas prácticas": un ciclo de vida iterativo y el desarrollo dirigido por el riesgo. Las prácticas del UP proporcionan una estructura organizada de ejemplo para discutir sobre cómo hacer y cómo aprender el A/DOO.

2.1. El desarrollo iterativo del UP

El UP fomenta el **desarrollo iterativo** que se organiza en mini-proyectos cortos, de duración fija (**iteraciones**). El resultado de cada uno es un sistema que puede ser probado, integrado y ejecutado. Cada iteración incluye sus propias actividades de análisis de requisitos, diseño, implementación y pruebas.

El ciclo de vida iterativo se basa en la ampliación y refinamiento sucesivos del sistema mediante múltiples iteraciones, con retroalimentación cíclica y adaptación como elementos principales que dirigen para converger hacia un sistema adecuado → **desarrollo iterativo e incremental**.

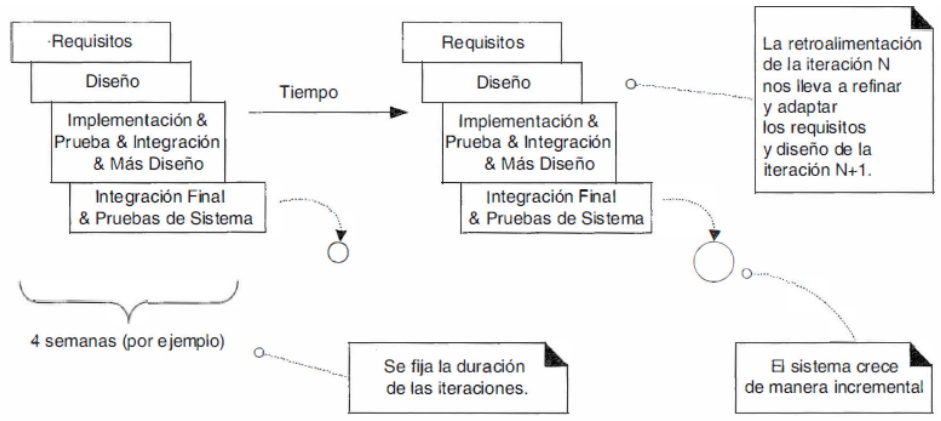


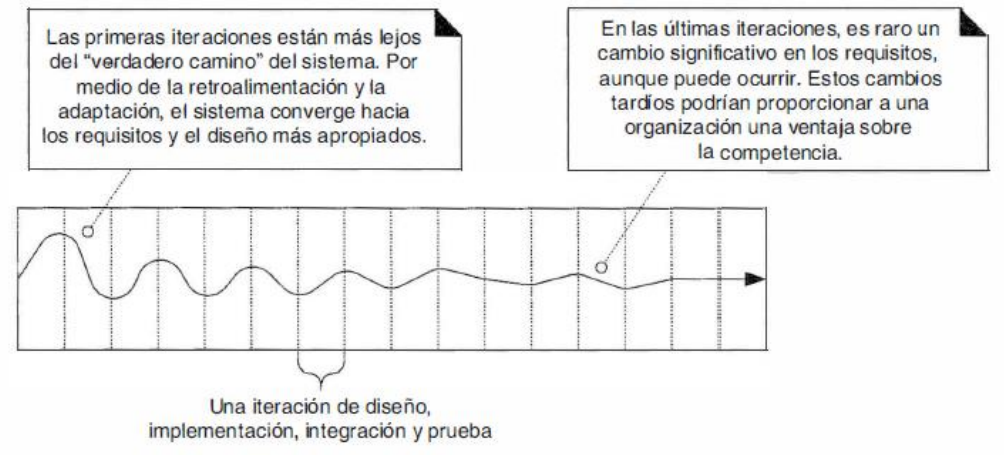
Figura 2.1. Desarrollo iterativo e incremental.

El resultado de cada iteración es un sistema ejecutable, pero incompleto; no está preparado para ser puesto en producción. La salida de una iteración no es un prototipo experimental o desechable, y el desarrollo iterativo no es prototipado; pero la salida es un subconjunto con calidad de producción del sistema final. Cada iteración aborda nuevos requisitos y amplía el sistema incrementalmente, una iteración podría volver sobre el software que ya existe y mejorarlo (centrándose en mejorar el rendimiento de un subsistema en lugar de extenderlo).

Retroalimentación y adaptación

El desarrollo iterativo se basa en una aptitud de aceptación del cambio y la adaptación como motores inevitables y esenciales. No significa que el desarrollo iterativo y el UP fomenten un proceso dirigido por "una adición de características" de manera incontrolada y reactiva. El UP llega a un equilibrio entre la necesidad de llegar a un acuerdo y estabilizar un conjunto de requisitos, y la realidad de los requisitos cambiantes, cuando el personal involucrado clarifica su visión o cambia el mercado. Cada iteración conlleva la elección de un pequeño conjunto de requisitos y, rápidamente, diseñar, implementar y probar.

Tener retroalimentación en una etapa temprana vale su peso en oro y aporta un conocimiento práctico y una oportunidad de modificar o adaptar la comprensión de los requisitos o el diseño. Además de clarificar los requisitos, actividades como la prueba de carga prueban si el diseño y la implementación parcial están en el camino correcto, o si en la siguiente iteración, se necesita un cambio en la arquitectura básica. El trabajo se desarrolla a lo largo de una serie de ciclos estructurados de construir-retroalimentar-adaptar. La desviación del sistema del "verdadero camino" en las primeras iteraciones será mayor que en las últimas.



Beneficios del desarrollo iterativo

- Mitigación tan pronto como sea posible de riesgos altos.
- Progreso visible en las primeras etapas.
- Sistema refinado que se ajusta más a las necesidades reales del personal involucrado.
- Gestión de la complejidad.
- El conocimiento adquirido en una iteración se puede utilizar metódicamente para mejorar el propio proceso de desarrollo, iteración a iteración.

Longitud de una iteración y fijación de la duración

El UP recomienda que la longitud de una iteración sea de dos a seis semanas (pasos pequeños, rápida retroalimentación, y adaptación), ya que iteraciones largas destruyen la motivación principal del desarrollo. Se **fija la duración** de las iteraciones y si parece difícil cumplir con el plazo fijado, la respuesta es eliminar tareas o requisitos de la iteración, e incluirlos en una iteración posterior, más que retrasar la fecha de terminación prevista. Equipos muy grandes podrían requerir iteraciones de más de seis semanas para compensar los costes fijos de coordinación y comunicación; pero no más de tres a seis semanas.

2.2. Conceptos y buenas prácticas del UP adicionales

La idea fundamental del UP e s el desarrollo iterativo, fijando iteraciones cortas, y adaptable, así como el uso de las tecnologías de objetos (A/DOO y la programación orientada a objetos).

Conceptos claves y buenas prácticas del UP:

- Abordar cuestiones de alto riesgo y muy valiosas en las primeras iteraciones.
- Involucrar continuamente a los usuarios para evaluación, retroalimentación y requisitos.
- Construir en las primeras iteraciones una arquitectura que constituya un núcleo central consistente.
- Verificar la calidad continuamente; pruebas muy pronto, con frecuencia y de manera realista.
- Aplicar casos de uso.
- Modelar software visualmente (con UML).
- Gestionar los requisitos con cuidado.
- Manejar peticiones de cambio y gestión de configuraciones.

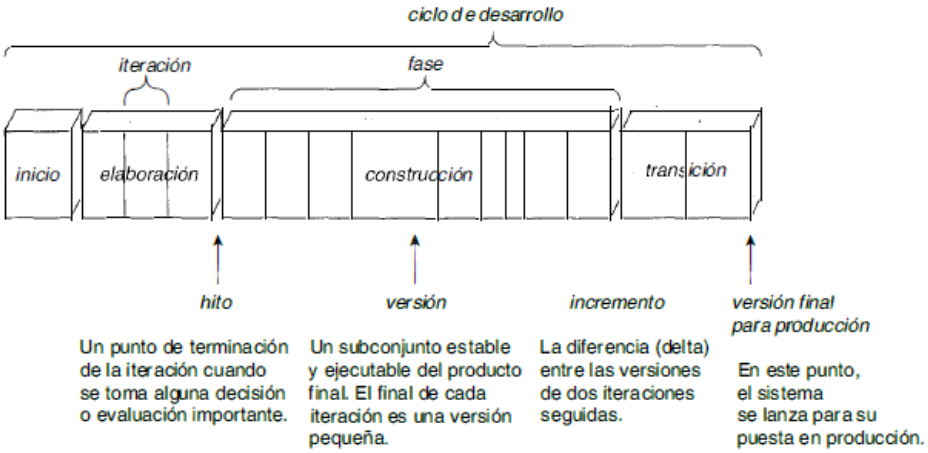
2.3. Las fases del UP y términos orientados a la planificación

UP organiza el trabajo y las iteraciones en cuatro fases fundamentales:

1. **Inicio**: visión aproximada, análisis del negocio, alcance, estimaciones imprecisas.
2. **Elaboración**: visión refinada, implementación iterativa del núcleo central de la arquitectura, resolución de los riesgos altos, identificación de más requisitos y alcance, estimaciones más realistas.

3. **Construcción**: implementación iterativa del resto de requisitos de menor riesgo y elementos más fáciles, preparación para el despliegue.
4. **Transición**: pruebas beta, despliegue.

Esto no se corresponde con el antiguo ciclo de vida "en cascada" o secuencial. La fase de Inicio no es una fase de requisitos; sino una especie de fase de viabilidad, donde se lleva a cabo sólo el estudio suficiente para decidir si continuar o no. La fase de Elaboración no es la fase de requisitos o de diseño; sino una fase donde se implementa, de manera iterativa, la arquitectura que constituye el núcleo central y se mitigan las cuestiones de alto riesgo.



2.4. Las disciplinas del UP

El UP describe actividades de trabajo en **disciplinas** (**flujos de trabajo**). Una disciplina es un conjunto de actividades (y artefactos relacionados) en un área determinada, como las actividades en el análisis de requisitos. En el UP, un artefacto es el término general para cualquier producto del trabajo: código, gráficos Web, esquema de base de datos, documentos de texto, diagramas, modelos, etcétera.

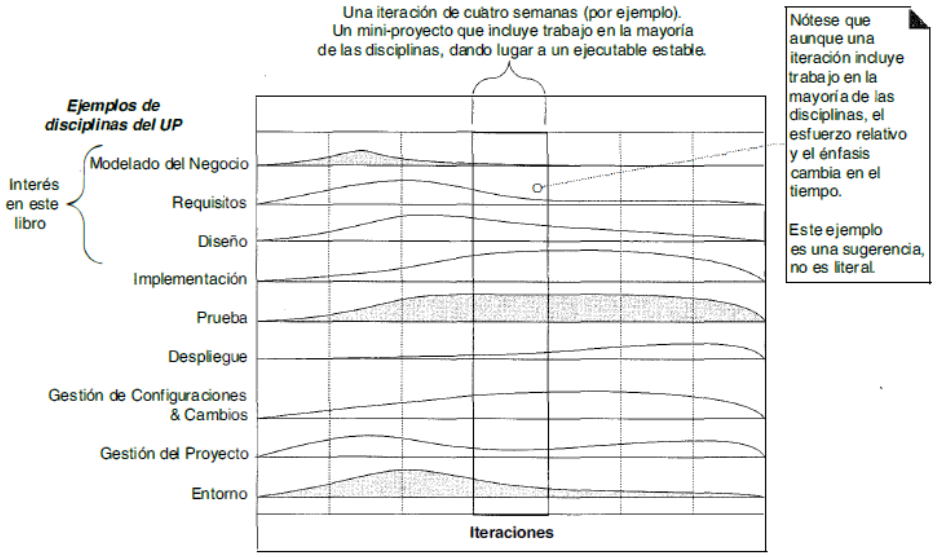
Hay varias disciplinas en el UP:

- **Modelado del Negocio**: se desarrolla una única aplicación que incluye el modelado de los objetos del dominio. Cuando se está haciendo análisis del negocio a gran escala o reingeniería de procesos del negocio incluye el modelado dinámico de los procesos del negocio de toda la empresa.
- **Requisitos**. Análisis de los requisitos para una aplicación, como escritura de casos de uso e identificación de requisitos no funcionales.
- **Diseño**. Todos los aspectos de diseño, incluyendo la arquitectura global, objetos, bases de datos, red y cosas parecidas.

En el UP, **Implementación** significa programar y construir el sistema, no despliegue y **Entorno** se refiere a establecer las herramientas y adaptar el proceso al proyecto (organizar la herramienta y el entorno del proceso).

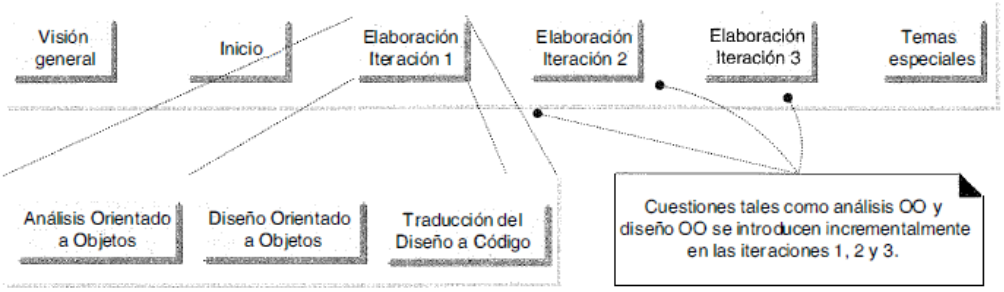
Disciplinas y fases

Durante una iteración, el trabajo se desarrolla en la mayoría o todas las disciplinas. El esfuerzo relativo en estas disciplinas cambia a lo largo del tiempo. Las primeras iteraciones tienden a aplicar un esfuerzo relativo mayor a los requisitos y al diseño, y en las posteriores disminuye.



Fases del UP:

1. La fase de inicio introduce el análisis de requisitos.
2. La iteración 1 presenta cómo asignar reponsabilidades a los objetos.
3. La iteración 2 está enfocada al diseño de objetos (patrones de diseño).
4. La iteración 3 presenta el análisis de la arquitectura y el diseño de frameworks.



2.5. Adaptación del proceso y Marco de Desarrollo

Artefactos opcionales

Algunas prácticas y principios del UP deben seguirse siempre, como el desarrollo iterativo y dirigido por el riesgo, y la verificación continua de la calidad. En el UP todas las actividades y artefactos son *opcionales*. En un proyecto UP, un equipo selecciona un pequeño subconjunto de artefactos que sirvan para tratar sus problemas y necesidades particulares para centrarse en un pequeño conjunto de artefactos que muestren tener un gran valor práctico.

El Marco de Desarrollo

La elección de los artefactos del UP para un proyecto se recoge en un documento breve → **Marco de Desarrollo** (artefacto en la disciplina Entorno).

Tabla 2.1 Ejemplo de Marco de Desarrollo de artefactos UP. c-comenzar; r-refinar

Disciplina	*Artefacto* *Iteración →*	*Inicio* *I1*	*Elab.* *E1…En*	*Const.* *C1…Cn*	*Trans.* *T1…T2*
Modelado del Negocio	Modelo del Dominio		c		
Requisitos	Modelo de Casos de Uso Visión Especificación Complementaria Glosario	c c c c	r r r r		
Diseño	Modelo de Diseño Documento de Arquitectura SW Modelo de Datos		c c c	r r	
Implementación	Modelo de Implementación		c	r	r
Gestión del Proyecto	Plan de Desarrollo SW	c	r	r	r
Pruebas	Modelo de Pruebas		c	r	
Entorno	Marco de Desarrollo	c	r		

2.6. El UP ágil

Un **proceso pesado** sugiere un proceso con las cualidades:

- Muchos artefactos creados en un ambiente burocrático.
- Rigidez y control.
- Planificación detallada, muy larga y elaborada.
- Predictivo más que adaptable.

Un **proceso predictivo** intenta planificar y predecir en detalle las actividades y asignación de recursos (personal) en un intervalo relativamente largo de tiempo, como la totalidad de un proyecto. Sigue un ciclo de vida en "cascada" o secuencial definiendo todos los requisitos, el diseño detallado y la implementación:

1. Determinar, registrar y acordar un conjunto de requisitos completo y fijo.
2. Diseñar un sistema basado en estos requisitos.
3. Implementar en base al diseño.

Un **proceso adaptable** acepta el cambio como motor inevitable y fomenta la adaptación flexible con un ciclo de vida iterativo. Un **proceso ágil** implica un proceso adaptable y ligero, listo para responder rápidamente a las necesidades cambiantes. La adaptación a un **UP ágil**:

- Opta por un conjunto pequeño de actividades y artefactos del UP.
- No completa los requisitos y diseños antes de la implementación. Surgen de modo adaptable a lo largo de una serie de iteraciones.
- No hay un plan detallado para todo el proyecto. Hay un plan de alto nivel (**Plan de Fase**) y otros hitos importantes, sin detallar los pasos de grano fino de estos hitos y un plan detallado (**Plan de Iteración**) que sólo planifica con gran detalle una iteración por adelantado.

2.7. No es UP si…

- Inicio = requisitos, elaboración = diseño, y construcción = implementación.
- El objetivo de la elaboración es definir modelos de manera completa y cuidadosa.
- Se define la mayoría de los requisitos antes de comenzar el diseño o la mayoría del diseño antes de comenzar a implementar.
- Se dedica mucho tiempo a realizar trabajo sobre los requisitos.
- Se realizan los diagramas UML y las actividades de diseño constituyen el momento para definir diseños y modelos de manera completa y precisa.
- El UP significa hacer muchas de las actividades posibles y crear muchos documentos.
- Se planifica un proyecto en detalle desde el principio hasta el final.
- Se quieren planes y estimaciones creíbles para los proyectos antes de que termine la fase de elaboración.