**TEMA 4. Los diagramas de clases, paquetes, objetos y de actividades.**

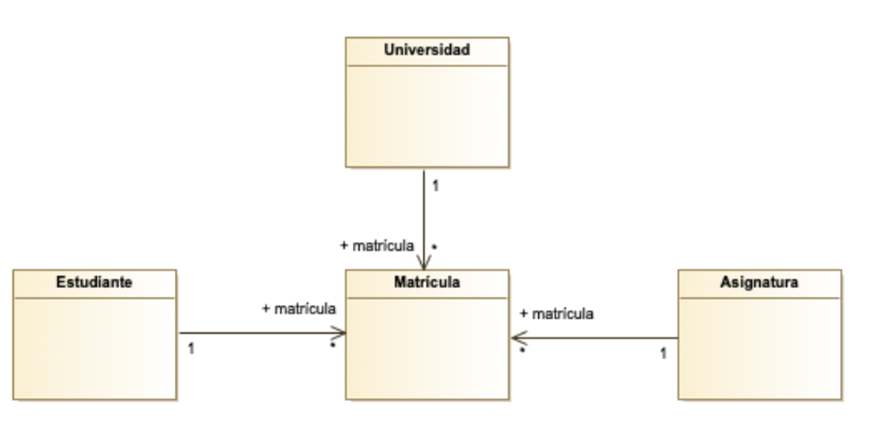
En el capítulo cuarto de la asignatura se han presentado distintos diagramas de UML. Concretamente, se han introducido los diagramas estructurales: de clases, de paquetes y de objetos; y el diagrama de comportamiento: de actividad.

Además, también se ha considerado un diagrama que, aunque no es un diagrama UML, si que es útil para anticipar conceptos y formas de pensar que serán útiles más adelante, en los modelos de diseño. Me refiero a los diagramas de clases de análisis.

**Elaborando diagramas de clases, paquetes y objetos.**

Los diagramas de clases guardan y modelan gráficamente lo que en capítulos anteriores eran requisitos de información. Es decir, con los diagramas de clases contamos con una herramienta gráfica con la que dar cuenta de todos los conceptos e información que debe manejar un sistema de información para ofrecer los servicios demandados de él.

Por ejemplo, si queremos abordar el desarrollo o mantenimiento de un producto software dedicado a la gestión de matrículas de estudiantes de una universidad. Necesitaremos considerar requisitos de información relacionados con los estudiantes, las matrículas, las asignaturas y la universidad en la que dichas matrículas se producen. Lo que en el capítulo dos fueron requisitos de información y se documentaron utilizando tablas, ahora pueden documentarse gráficamente gracias a los diagramas de clases, paquetes y objetos. Por ejemplo, puedes considerar el siguiente diagrama:



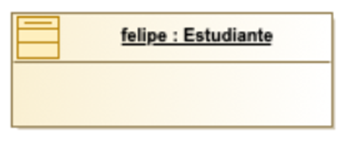
En un diagrama de clases puedes representar las clases o entidades del dominio del problema que te interesan y las relaciones que éstas guardan entre ellas.

Entre las relaciones entre clases más habituales dispones de las relaciones de asociación (relación estructural entre clases), agregación (relación parte - todo entre clases), composición (relación parte-todo fuerte entre clases), herencia/generalización y las clases asociación.

Para ayudarte en la identificación de clases y relaciones entre ellas puedes consultar la experiencia documentada en forma de patrones. Concretamente, la colección de patrones de colaboración ofrece una interesante herramienta para que no estés solo a la hora de hacer diagramas de clases.

Las clases y las relaciones entre ellas podemos agruparlas semánticamente utilizando el concepto de paquete y hacer así diagramas de paquetes. Con ese concepto no estamos tan interesados en las clases en sí, sino en grupos de clases que colaboran para conseguir un determinado propósito.

Los diagramas de objetos serían el último de los diagramas estructurales que hemos visto en este capítulo. Un objeto no es más que una instanciación de una clase. Por ejemplo, felipe es un objeto de Estudiante.



**Los diagramas de actividades.**

En el capítulo cuarto también has encontrado un diagrama útil para modelar comportamiento. Me refiero a los diagramas de actividades. Hasta ahora el único diagrama especialista en comportamiento que habíamos visto era el diagrama de casos de uso y los casos de uso daban cuenta de lo que los actores podían hacer con el sistema. Después, recurríamos a tablas para dar cuenta de los pasos que el actor y el sistema hacían para llevar a cabo cada caso de uso.

Ahora, con los diagramas de actividades, tienes una herramienta gráfica para dar cuenta de esa misma información. Es decir, lo que antes eran precondiciones, flujos normales y alternativos, ahora puedes documentarlos con un diagrama gráfico.

Para elaborar diagramas de actividades dispones de nodos en forma de: nodo de acción (actividad simple que no se puede descomponer), nodo de actividad (actividad compleja que se puede descomponer en acciones), nodo de actividad inicial y final, nodos decisión/mezcla (bifurcación) y nodos fork(join (concurrencia).

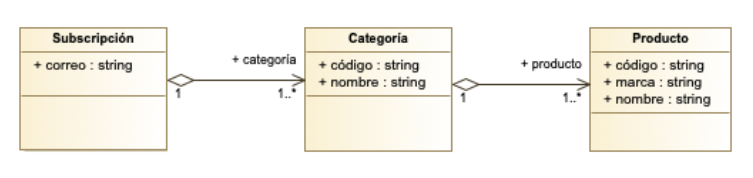
Los nodos en un diagrama de actividad se relacionan entre sí con flujos de control.

En definitiva, con los diagramas de actividad tienes la herramienta perfecta para documentar los casos de uso con mayor nivel de detalle y, además, puedes hacerlo de manera gráfica. Intenta siempre considerar en tus diagramas de actividades que los actores hacen cosas, pero también el sistema hace cosas en justa reciprocidad.

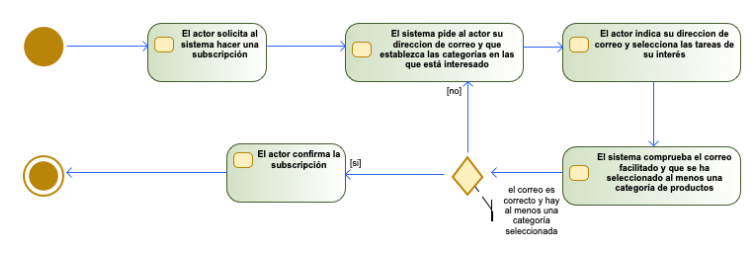
**Ejemplo:**

Elabora el diagrama de clases y el diagrama de actividad correspondiente al caso de uso "Susbscribirse a las ofertas en ESIITech. S.L.". Compáralo con las siguientes propuestas de solución.

El diagrama de clases que se sugiere para el supuesto planteado en el examen individual de prácticas podría ser el siguiente:



El diagrama de actividad asociado al examen de prácticas y el caso de uso "Subscribirse a las ofertas en EsiiTech S.L." podría ser el siguiente:



Efectivamente, los diagramas propuestos son correctos y documentan las entidades asociadas a la subscripción y las actividades necesarias para realizar una subscripción.

**TEMA 5. Los diagramas de estados, secuencia, componentes y despliegue.**

En el capítulo quinto de la asignatura se han presentado distintos diagramas de UML. Concretamente, se han introducido diagramas de diseño, es decir, diagramas que dan soporte a documentar la solución y no el problema como veníamos haciendo con los diagramas de análisis. Entre los diagramas de diseño vistos hay estructurales: de componentes y de despliegue; y diagramas de comportamiento: de estados y de secuencia.

**Los diagramas de estados.**

Los diagramas de estados son útiles para documentar los distintos estados por los que pueden pasar los objetos de una entidad del modelo de dominio.

No todas las entidades del modelo de dominio son susceptibles de tener asociado un diagrama de estados. Es decir, hay muchas clases que no pasan por diferentes estados y no exhiben distintos comportamientos en cada uno de ellos.

Cuando los objetos de una entidad si que pueden pasar por distintos estados y en cada uno de ellos hacen gala de un comportamiento, los diagramas de estados son una potente herramienta.

Por ejemplo, sabes que cuando haces un pedido, éste pasa por diferentes estados: preparando el pedido, pedido enviado, pedido en reparto, pedido entregado, etc. La transición entre cada uno de esos estados tiene que quedar documentada junto a los estados mencionados.

Para documentar una transición tienes que identificar los eventos y/o condiciones de guarda que determinan el paso de un estado a otro. Trata de ser cuidadoso cuando realices un diagrama de estados, un diagrama de estados debería ser determinista.

Además de las transiciones y los estados, otro elemento es también útil cuando haces un diagrama de estados. Me refiero a las actividades que puedes asociar bien a las transiciones o bien a los estados (cuando entras, sales o estás en un estado).

Unas últimas recomendaciones:

* no confundas un diagrama de actividad con uno de estados, tienen un propósito totalmente distinto.
* etiqueta de manera adecuada los distintos estados. Utiliza adjetivos, situaciones y nunca verbos.

**Los diagramas de secuencia.**

Los diagramas de secuencia son unos diagramas muy útiles y potentes a nivel de diseño. Con ellos podemos especificar soluciones asociadas a la realización de los casos de uso de un problema dado.

Los diagramas de secuencia se elaboran y leen de izquierda a derecha y de arriba hacia abajo. En ellos vamos a ir detallando la interacción entre objetos asociados a entidades modelo, vista y control. Dicha interacción se traduce en el intercambio de mensajes entre unas entidades y otras.

Si un objeto envía un mensaje a otro objeto es porque el objeto receptor tiene definido una operación con la que puede responder a ese mensaje. Es decir, el objeto emisor invoca una operación definida para los objetos receptores del mismo.

En función de esta descripción los diagramas vistos previamente que son útiles para hacer buenos diagramas de secuencia son los siguientes:

* diagramas de clases a nivel de diseño
* diagramas de clases de análisis, donde ya dispondremos de lo que ahora serán entidades asociadas al modelo, a la vista y al control, y que en aquellos se denominan, respectivamente, entities, boundary y control.
* diagramas de casos de uso, y sobre todo, los flujos normales y alternativos asociados a los casos de uso a través de la documentación de su requisito funcional asociado.

Mi recomendación es que cuando tengas que elaborar un diagrama de secuencia trates de tener muy claro qué caso de uso o escenario tienes que considerar, y elabores una descripción textual de los pasos asociados al mismo. Esa descripción te permitirá identificar y proponer mensajes que intercambiarán las distintas entidades involucradas en la realización de dicho caso de uso/escenario.

Hacer un diagrama de secuencia puede ser una tarea muy parecida a programar. Piensa en seudocódigo el escenario asociado y elabora el diagrama pensando en términos de estructuras de control (if\_then\_else, for, while, repeat, opt, ...) utilizando fragmentos combinados (alt, loop, opt, etc.).

**Los diagramas de componentes y de despliegue.**

Los últimos diagramas de UML que hemos presentado en clase son los diagramas de componentes y de despliegue.

Estos diagramas, quizá, también los podríamos haber presentado al principio. Al presentar los diagramas de casos de uso al principio hemos desplegado una abordar el desarrollo de una aplicación de "abajo hacia arriba" (bottom - up).

Sin embargo, si hubiéramos presentado primero los diagramas de componentes podríamos haber pensado de "arriba hacia abajo" (top - down), identificando o tratando de identificar primero las partes o subsistemas que serían útiles para abordar el desarrollo de un producto software y luego documentando de manera adicional cada uno de esos subsistemas.

En los diagramas de componentes identificamos partes de un sistema o subsistemas y documentamos cómo colaboran entre ellos para conseguir ofrecer la funcionalidad deseada con el sistema.

En los diagramas de componentes disponemos de componentes y conectores. Los conectores permiten comunicar unos nodos con otros y normalmente identificamos qué componente facilita información o servicios (---o, interfaz proporcionada) y cuál de ellos demanda información o servicios (---c, interfaz demandada). Un conector es la combinación de una interfaz proporcionada con otra demandada.

A la hora de identificar y documentar los componentes de un sistema normalmente distinguimos entre componentes de presentación, de lógica de negoción y de servicios (por ejemplo, persistencia y seguridad).

Los diagramas de despliegue sirven para desplegar el sistema una vez está desarrollado. Hay sistemas cuyo despliegue es muy simple ya que se ejecutan en una máquina de manera aislada. Sin embargo, hoy en día en que todo está conectado y distribuido geográficamente otros despliegues son posibles y habituales.

Para hacer diagramas de despliegue disponemos de tres elementos: los nodos (que representan localizaciones geográficas con mayor o menor capacidad de computo), canales de comunicación (que permiten comunicar unos nodos con otros) y artefactos (documentos, ejecutables, código, bibliotecas, etc.) que están desplegados en un nodo y que se manifiestan en componentes.

**Tema 6: Proceso Unificado**

En el capítulo sexto de la asignatura se ha presentado un método para abordar de manera sistemática, cuantificable e ingenieril las actividades de desarrollo y mantenimiento del software.

Se trata de la propuesta de Proceso Unificado de Rational, una metodología tradicional y "pesada", en cuanto a lo exigente que es en materia de documentación asociada.

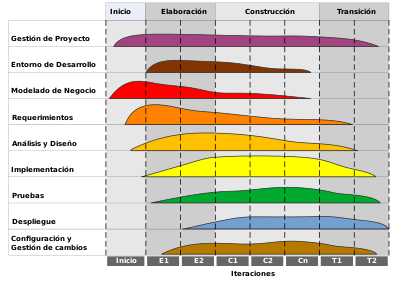
Los puntos a favor: sistemática y bien documentada.

Los puntos en contra: exigente en cuanto a documentación.

# **El proceso unificado en una imagen**

En el capítulo sexto de la asignatura se ha presentado un método para abordar de manera sistemática, cuantificable e ingenieril las actividades de desarrollo y mantenimiento del software. Se trata de la propuesta de Proceso Unificado de Rational, una metodología tradicional y "pesada", en cuanto a lo exigente que es en materia de documentación asociada. Los puntos a favor: sistemática y bien documentada. Los puntos en contra: exigente en cuanto a documentación.

El Proceso Unificado tiene asociada una imagen que nos permite identificar sus principales elementos. La imagen a la que me refiero es la siguiente:



En ella puedes ver:

* distintas fases (horizontal parte superior): inicio, elaboración, construcción y transición.
* distintas actividades, disciplinas o workflows (vertical parte izquierda): donde se identifican distintas actividades, tanto de carácter ingenieril (modelado de negocio, requisitos, análisis, diseño, implementación, pruebas y despliegue) como de gestión/soporte (gestión de proyectos, control de cambios y gestión de entorno).
* distintas iteraciones (horizontal parte inferior): es otra de las características del proceso unificado. En esta propuesta y en cada fase se abordan diferentes iteraciones (tantas como me permita el presupuesto y el tiempo disponible) para desarrollar un producto software.

# 

# 

# **Una descripción exhaustiva.**

Debido a la trayectoria que tiene el proceso unificado no te resultará difícil encontrar información sobre dicha propuesta. Libros y referencias web te permitirán conocer detalles más concretos de esta propuesta.

Todos y cada uno de los conceptos y otras muchas facilidades están perfectamente identificadas y descritas en dicha documentación.

Encontrarás:

* actores en forma de roles o conocimientos y habilidades que tienen que tener diferentes involucrados en el desarrollo de software: analistas, diseñadores, gestores de proyectos, implementadores, testers, etc., son ejemplos de algunos de esos involucrados.
* actividades: conjunto de acciones perfectamente identificadas que deben abordar cada uno de los actores y que contribuyen de una u otra forma al desarrollo del software.
* recursos / plantillas: conocimientos, plantillas y documentos que permiten documentar las actividades realizadas.
* productos: normalmente las actividades realizadas por los actores utilizando los recursos necesarios dan lugar a productos o artefactos que son utilizados en actividades posteriores.

Las tres máximas que describen la propuesta propuesta unificado son las siguientes:

1. Dirigido por casos de uso. El proceso de desarrollo descansa en la identificación y documentación de los requisitos y en una posterior realización sucesiva de dichos casos de uso.
2. Centrado en la arquitectura. Un producto software posee múltiples vistas, concretamente, 4 +1. En estas vistas se puede dar cuenta de distintos escenarios, de la vista lógica, la vista física, la vista de proceso y la vista de despliegue. Muchos de los diagramas que hemos visto a lo largo de la asignatura, cuando hemos presentado y utilizado UML, sirven para dar soporte y especificar las vistas anteriores. Por ejemplo y sucesivamente, diagramas de casos de uso, diagramas de clases, diagramas de despliegue, diagramas de actividades y los diagramas de componentes.
3. El proceso propuesto es iterativo e incremental. Es decir, el desarrollo se aborda poco a poco y produciendo sucesivos incrementos respecto a la funcionalidad del sistema.

### 

### **Todo tiene un final.**

En este tema has encontrado descripciones y referencias que pueden resultarte de utilidad para ir cogiendo ideas y construir con los contenidos de esta asignatura y de otras técnicas y métodos con los que ir construyendo tu propio método para desarrollar y mantener software. Es decir, no tienes que apostar y conocer en detalle ninguno de los métodos de desarrollo de software que se te vayan presentando sino, más bien, mi recomendación es que te quedes de cada uno de ellos lo que te consideres útil y construyas tu propio método para desarrollar tus actividades académicas y posteriormente profesionales.

Otro aspecto que también te recomiendo es que estés bien atento a partir de ahora y sigas incorporando a tu forma de trabajar métodos, técnicas, procesos, herramientas y conocimientos que tú mismo consideres útiles.