Universidad Intercontinental de la Empresa

Asignatura

Arquitectura del Software 

Profesor

Yago Fontenla Seco

{*yago.fontenla1@uie.edu*}

Unidad II – Fundamentos de Ingeniería del Software

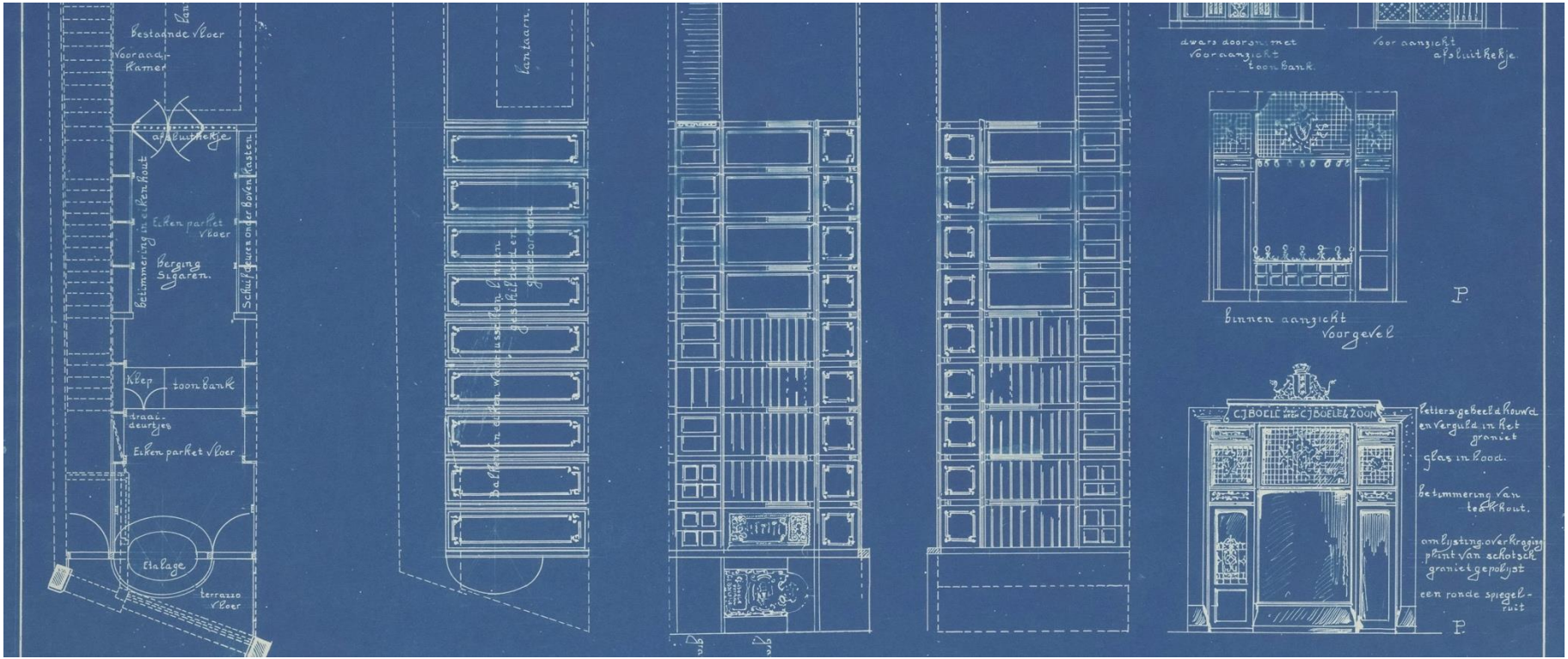
Se dedica a los **principios fundamentales** de la arquitectura de software, incluyendo la organización y dominios, así como los atributos de calidad esenciales en el diseño arquitectónico. Se introducen los elementos fundamentales de una arquitectura y los principios básicos de organización arquitectónica y atributos de calidad. Se estudian los principios del **Desarrollo y la Operación (DevOps)**, el **control de versiones** y la **containerización y virtualización**.

Temas

2.1. Arquitectura de Software. Fundamentos, organización, dominio y atributos de calidad.

2.2. Pricipios del Desarrollo y la Operación (DevOps)

Diseño del Software



Diseño del Software

El **diseño de software** es una etapa crucial en la ingeniería de software donde se define cómo se va a construir un sistema en base a los requisitos previamente especificados. Durante el diseño del software, se ponen en práctica el diseño arquitectónico y el modelado.

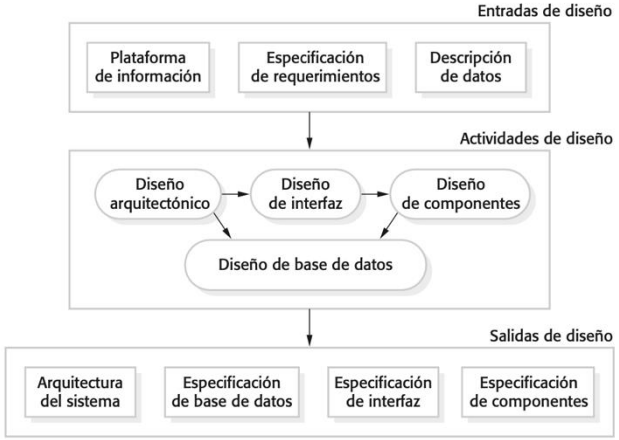
Mientras que el **diseño arquitectónico** se enfoca en las decisiones estructurales de alto nivel, el **diseño de software** profundiza en aspectos más detallados, dividiendo la arquitectura en módulos y detallando cada componente para que se pueda implementar.

La **salida** del proceso de diseño consiste en un conjunto de **modelos** que describen la organización y funcionalidad del sistema. Un paso importante es determinar los modelos de diseño que se necesitarán y el nivel de detalle requerido.

Diseño del Software

En la práctica, los procesos de especificación del software y diseño están muy cohesionados y llegan a solaparse:

- **Descomposición arquitectónica:** Es necesaria para estructurar y organizar la especificación del sistema, incluso **durante la definición de requerimientos**.
- **Propuesta temprana de arquitectura:** Un esquema abstracto puede asociar grupos de funciones con componentes de alto nivel.
- **Colaboración:** Esta descomposición sirve para discutir requerimientos y características del sistema con los interesados.

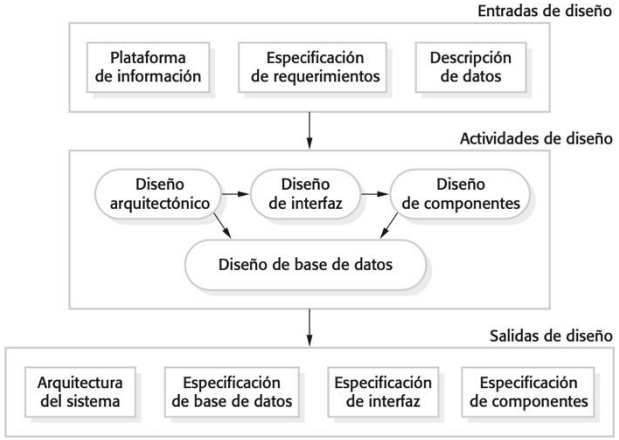


Principios de la Arquitectura del Software

Diseño del Software

¿Cómo se **descompone el diseño** en distintas perspectivas?

- **Diseño arquitectónico:** estructura global del sistema, módulos y su comunicación.
- **Diseño de interfaz:** cómo los componentes y usuarios interactuarán entre sí.
- **Diseño de componentes:** detalle interno de cada módulo (clases, métodos, algoritmos).
- **Diseño de base de datos:** estructura lógica y física de los datos.



Principios de la Arquitectura del Software

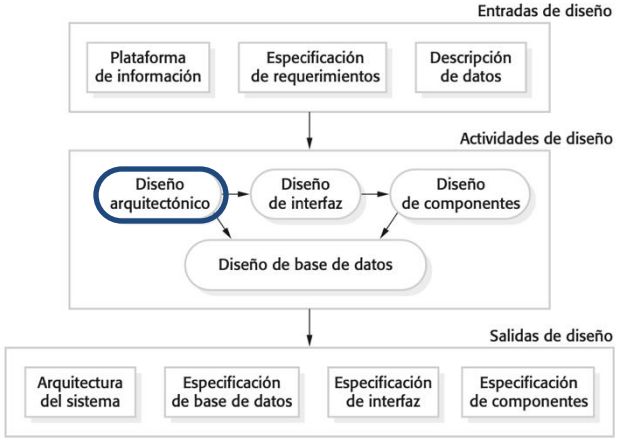
No son pasos lineales, sino actividades que se influyen mutuamente.

Arquitectura del Software

El diseño arquitectónico se centra en **entender cómo debe organizarse un sistema.**

Es la **primera etapa en el proceso de diseño del software** y sirve como **enlace crucial** las etapas de especificación y desarrollo, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos.

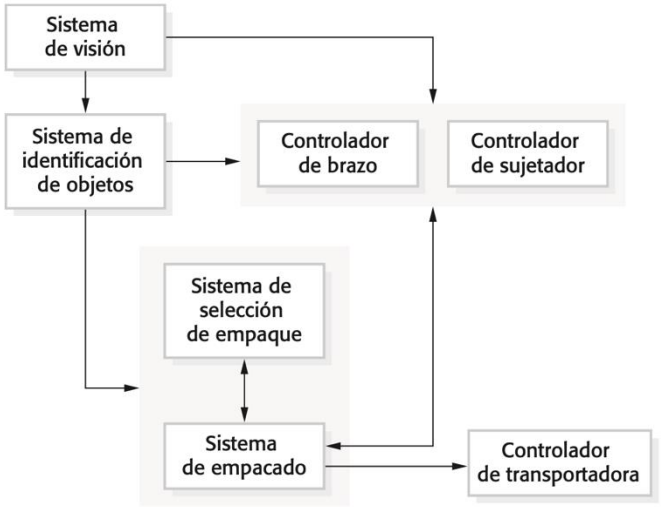
La salida del proceso de diseño arquitectónico consiste en un **modelo arquitectónico** que describe la organización del sistema como un conjunto de componentes relacionados.



Principios de la Arquitectura del Software

Puntos clave

- La arquitectura de software define la **estructura** fundamental de un sistema. Pero incluye tanto decisiones estructurales como no estructurales.
- Se centra en la **organización** de componentes del software y cómo **interactúan**.
- Es esencial para la calidad, el rendimiento y la escalabilidad del sistema.



Arquitectura de un sistema de control para un robot empacador

Decisiones Arquitectónicas

Las decisiones arquitectónicas determinan cómo se **organiza** un sistema de software. Consideran tanto **restricciones funcionales** como **no funcionales** (rendimiento, seguridad, etc.). Algunas decisiones clave incluyen:

- Selección de **estilos** y **patrones** arquitectónicos.
- Definición de **vistas** y **diagramas** que representen la arquitectura.
- Identificación de **componentes** principales y sus relaciones.

Los **componentes** del sistema implementan los requisitos **funcionales**. La forma en que los componentes se **organizan y comunican**, representa los requisitos **no funcionales**.

Ventajas del Diseño Arquitectónico

- **Comunicación efectiva:** Esta representación de alto nivel sirve como punto de referencia común. Proporciona una visión global del sistema, facilitando la discusión entre diferentes partes interesadas (desarrolladores, cliente).

- **Análisis temprano del sistema:** Definir y documentar la arquitectura en las fases iniciales permite realizar un análisis crítico sobre el cumplimiento de requisitos clave como rendimiento, fiabilidad y mantenibilidad. Las decisiones tomadas en esta etapa influyen significativamente en la calidad del producto final.

- **Reutilización a gran escala:** Una arquitectura bien definida permite la reutilización de componentes y patrones en diferentes proyectos con necesidades similares. Esto es especialmente útil en líneas de productos de software, donde una única arquitectura puede aplicarse a varias soluciones relacionadas.

Uso de los Modelos Arquitectónicos

- Facilitación de la discusión sobre el diseño del sistema: Un **modelo arquitectónico de alto nivel** permite una **comunicación clara** con todos los participantes del proyecto. Al no incluir detalles técnicos complejos, este **modelo abstracto** permite que todos los involucrados, incluidos los gestores y clientes, comprendan la **estructura general** del sistema. También ayuda a los administradores a planificar la asignación de recursos y la organización del desarrollo de componentes clave.

- Documentación de la arquitectura diseñada: Un **modelo arquitectónico detallado** proporciona una representación clara de todos los **componentes** del sistema, sus **interfaces** y las **conexiones** entre ellos. Este tipo de documentación es crucial para la comprensión global del sistema y facilita futuras modificaciones y actualizaciones a medida que el software evoluciona.

Vistas Arquitectónicas

Los **modelos arquitectónicos** son esenciales tanto para discutir los requerimientos como para documentar un diseño. El problema es que **un solo modelo no puede capturar toda la información relevante** sobre la arquitectura de un sistema, ya que cada uno presenta únicamente una perspectiva del sistema.

Es necesario utilizar múltiples vistas, que ofrecen diferentes perspectivas sobre el sistema, lo que facilita la comprensión y el análisis desde varios ángulos.

¿Qué vistas o perspectivas son útiles al diseñar y documentar una arquitectura del sistema?

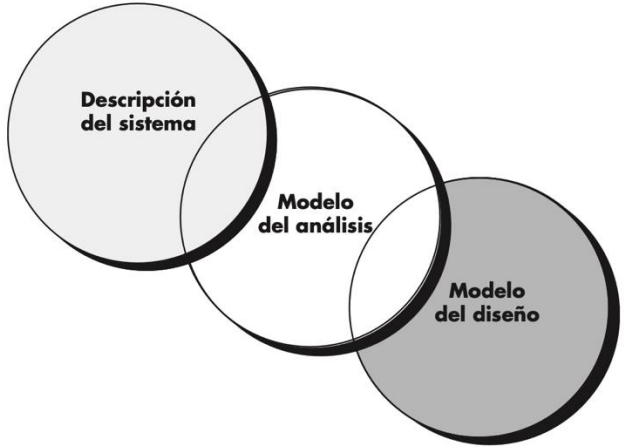
¿Qué es exactamente un modelo?

Modelado

El **modelado** juega un papel crucial en el proceso de ingeniería de requisitos y diseño del software, ya que sirve para **representar visualmente el sistema**, los requisitos y como éste debe **comportarse** o interactuar con otros sistemas y usuarios.

Según **Sommerville**:

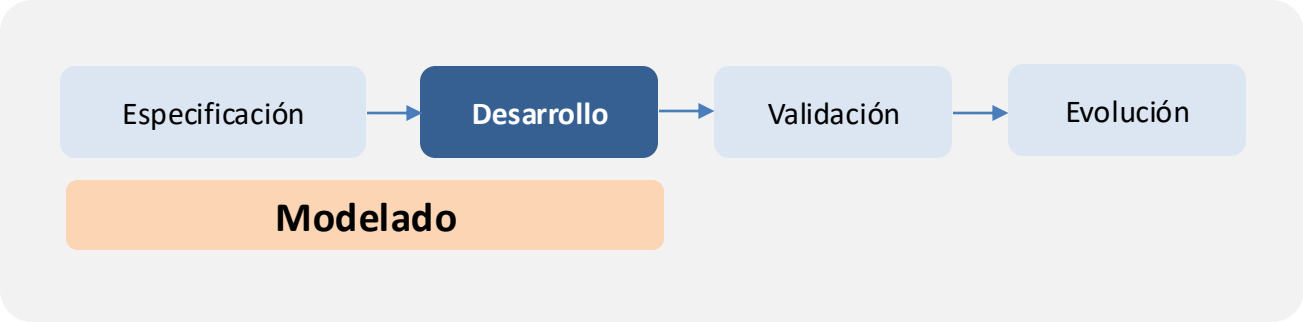
> *"El modelado de sistemas es el proceso para desarrollar modelos abstractos de un sistema, donde cada modelo presenta una visión o perspectiva diferente de dicho sistema."*



El modelo de requerimientos como puente entre la descripción del sistema y el modelo del diseño

Modelado

El modelado puede considerarse tanto una actividad del **diseño** como parte de la **especificación** del software, dependiendo del propósito y del tipo de modelo que se utilice.

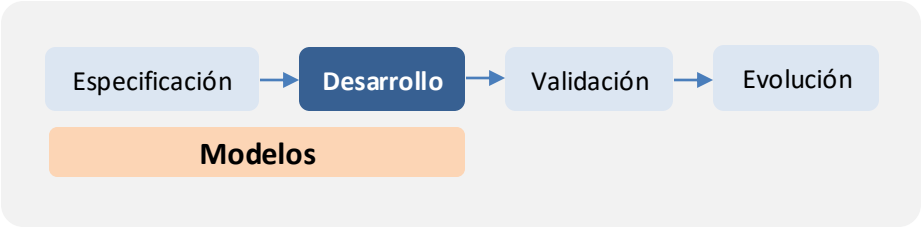


Ciclo de vida del software

Modelado durante el ciclo de vida del software

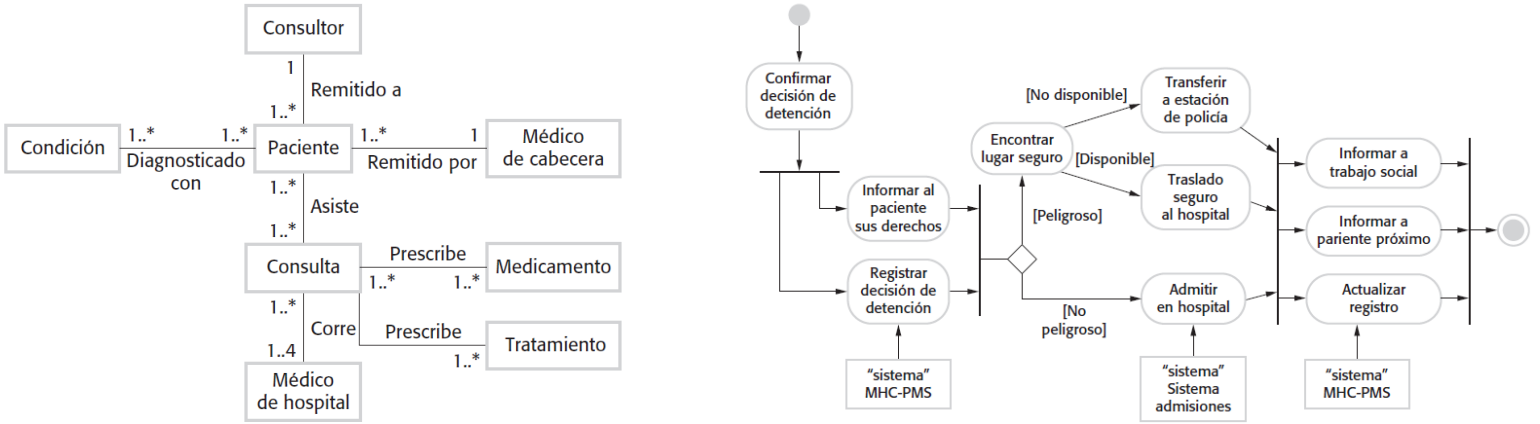
Los modelos se utilizan durante las distintas etapas del ciclo de vida de un software:

1. Durante el **proceso de ingeniería de requisitos** (especificación del Software) para ayudar a **derivar los requisitos** de un sistema.
2. Durante el proceso de **diseño** para **describir el sistema** a los ingenieros que implementan el sistema.
3. Después de la implementación para **documentar** la estructura y la operación del sistema.



¿Qué es un modelo?

Un **modelo** es una **visión abstracta** de un sistema que ignora algunos detalles del sistema. Pueden desarrollarse modelos complementarios del sistema para mostrar el contexto, las interacciones, la estructura y el comportamiento del sistema.



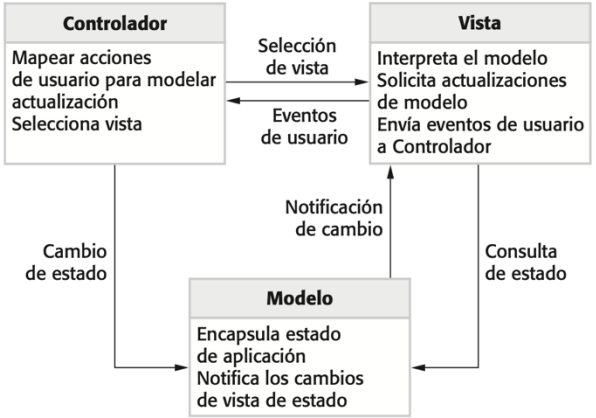
Modelos estructurales de un sistema. A la izq. Diagrama de clases, a la dcha. Diagrama de actividad.

¿Cómo se representa? - UML

El **Lenguaje Unificado de Modelado (UML)** se diseñó para describir sistemas **orientados a objetos** y, en la etapa de diseño arquitectónico, uno quiere describir con frecuencia sistemas en un nivel superior de abstracción.

Los diagramas UML permiten una **representación** clara y **estandarizada** de los componentes y sus interacciones.

La elección de la notación adecuada depende de los detalles que se desean representar y de la audiencia objetivo.



Ejemplo de diagrama UML

Perspectivas

Un Sistema puede modelarse desde **diferentes perspectivas**. En concreto:

1. **Modelos de contexto**: Una perspectiva **externa**, donde se modela el **contexto** del sistema. Describen el sistema y su entorno, delimitando sus interacciones y el alcance.
2. **Modelos de interacción:** Una perspectiva donde se modela la **interacción** entre un sistema y su entorno, o entre los componentes de un sistema. Incluyen diagramas de casos de uso y de secuencia.
3. **Modelos estructurales:** Una perspectiva **estructural**, donde se modela la **organización de un sistema** o la **estructura de datos** que procese el sistema. Se utilizan diagramas de clases para mostrar la jerarquía y las relaciones entre los elementos.
4. **Modelos de comportamiento:** Una perspectiva de **comportamiento**, donde se modele el comportamiento dinámico del sistema y cómo responde ante ciertos eventos.

Nota: Estas perspectivas tienen mucho en común con la visión **4 + 1 de arquitectura del sistema de Kruchten** la cual sugiere que la arquitectura y la organización de un sistema deben documentarse desde diferentes perspectivas.

Vistas Fundamentales 4+1

Krutchen (1995) define el **modelo de vista 4+1**, que sugiere que deben existir cuatro vistas arquitectónicas fundamentales, que se relacionan usando casos de uso o escenarios.

- **Vista lógica:** Representa los componentes funcionales del sistema y sus relaciones.
- **Vista de procesos:** Muestra la interacción y concurrencia entre procesos en el sistema.
- **Vista de desarrollo:** Detalla cómo se estructura el sistema en módulos y componentes.
- **Vista física:** Describe la distribución del software en el hardware.

Tipos de modelo – Diagramas

Se considera que los siguiente cinco tipos de diagrama sirven para **representar lo esencial** de un sistema.

Interacción

- Diagramas de **caso de uso**, que exponen las interacciones entre un sistema y su entorno.
- Diagramas de **secuencias**, que muestran las interacciones entre los actores y el sistema, y entre los componentes del sistema.

Estructural

- Diagramas de **clase**, que revelan las clases de objeto en el sistema y las asociaciones entre estas clases.
- Diagramas de **actividad**, que muestran las actividades incluidas en un proceso o en el procesamiento de datos.

Comport.

- Diagramas de **estado**, que explican cómo reacciona el sistema frente a eventos internos y externos.

Modelos estático, funcional y dinámico

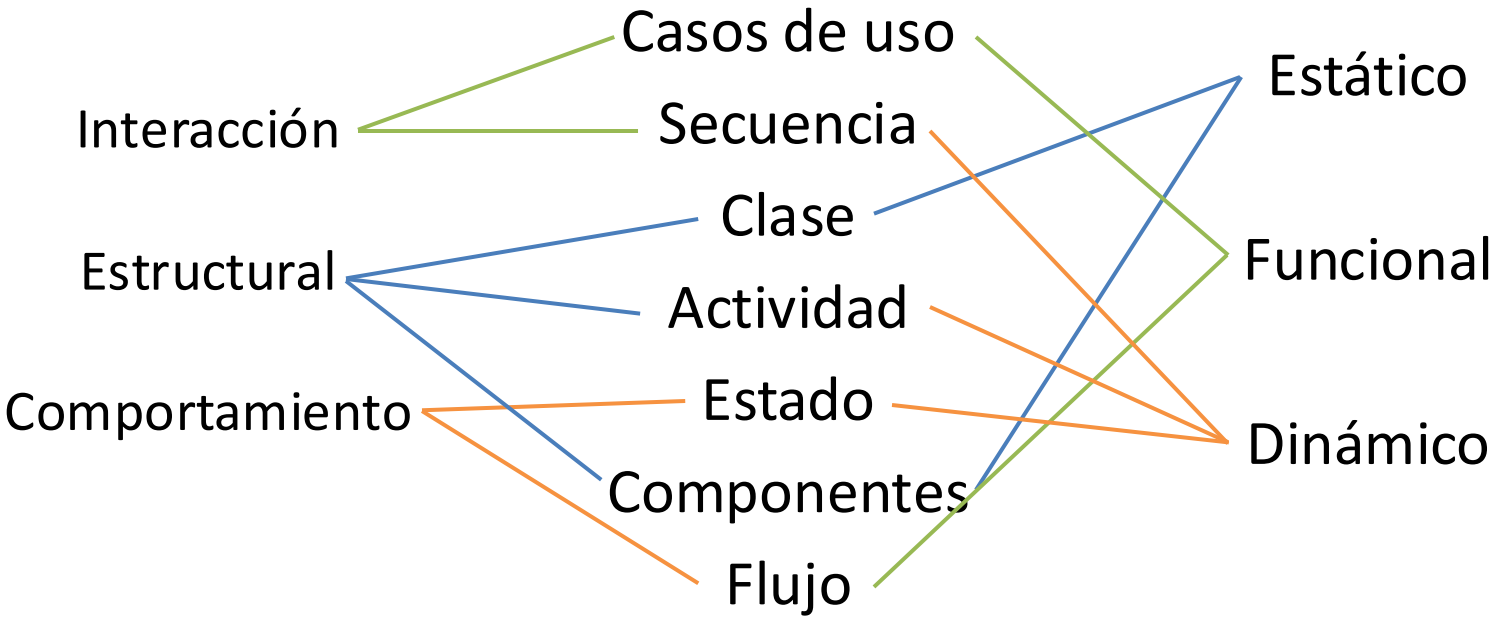
Son diferentes tipos de representaciones del sistema que abordan aspectos complementarios del diseño y la funcionalidad:

- **Estático:** Se enfoca en los **aspectos estructurales** de un sistema y se relaciona con la **Vista de Desarrollo** y la **Vista Física** del modelo 4+1. Esto abarca **diagramas de clases** y **de componentes** que representan la estructura fija del sistema y cómo se agrupan los módulos.

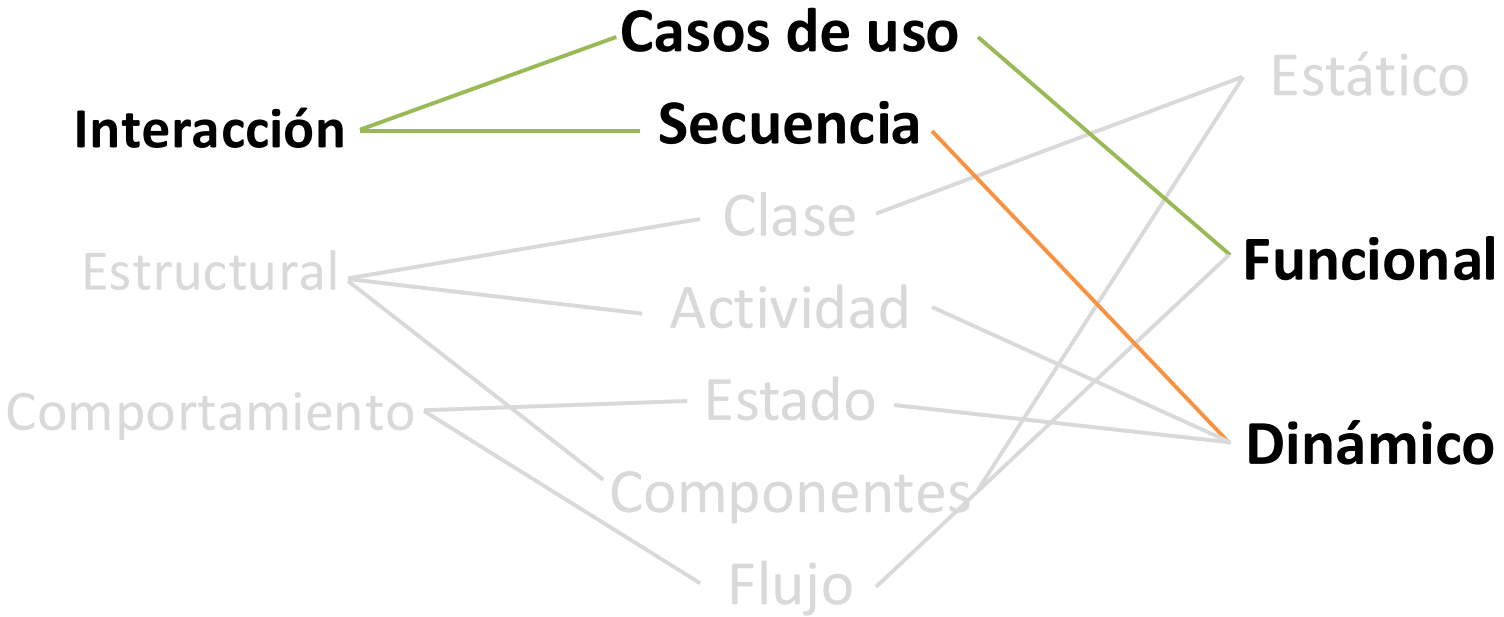
- **Funcional:** Se centra en cómo el sistema satisface los requisitos funcionales y corresponde a la **Vista Lógica** y a los **Casos de Uso** del enfoque 4+1. Los modelos funcionales muestran los servicios que el sistema proporciona, y se pueden documentar mediante **diagramas de casos de uso y diagramas de flujo de datos**.

- **Dinámico:** Describe el comportamiento del sistema en términos de interacciones y cambios de estado en tiempo de ejecución, y se relaciona con la **Vista de Procesos**. Utiliza **diagramas de secuencia, actividad diagramas de estado** para representar cómo el sistema responde a eventos y cómo los componentes interactúan durante la ejecución.

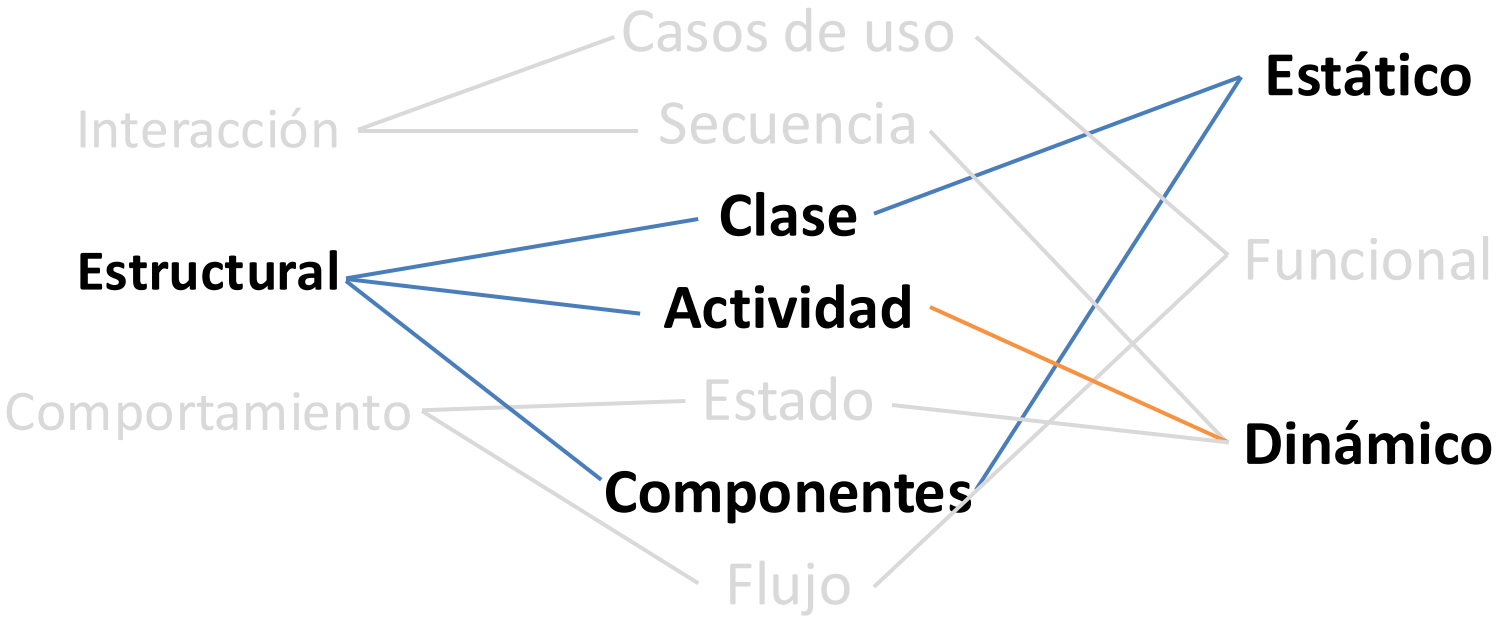
Tipos de modelo – Estático, funcional y dinámico



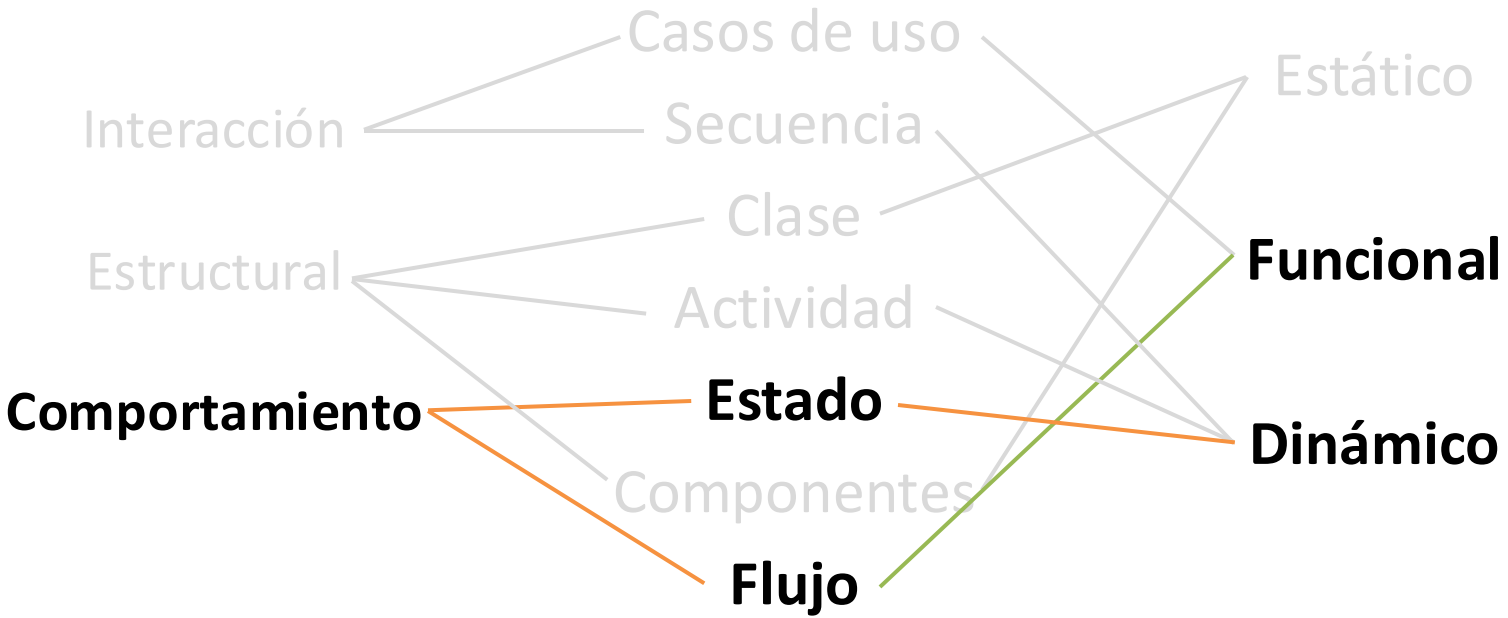
Tipos de modelo – Estático, funcional y dinámico



Tipos de modelo – Estático, funcional y dinámico



Tipos de modelo – Estático, funcional y dinámico

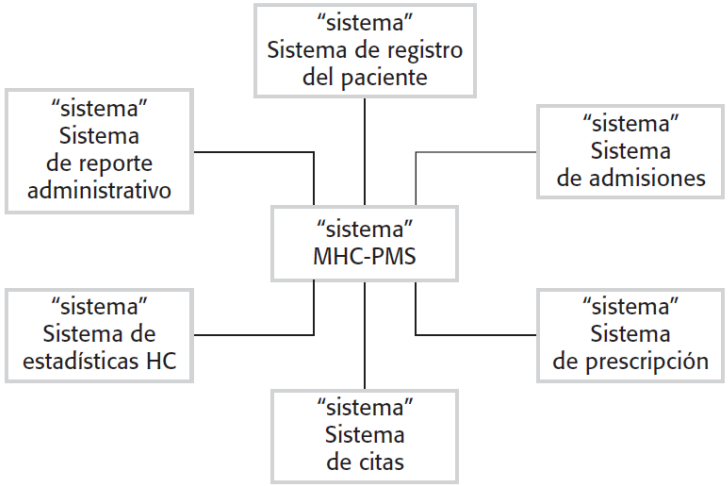


Modelos de Contexto

Muestran el **entorno** de un sistema, incluyendo otros sistemas automatizados. Sin embargo, no presentan los tipos de relaciones entre los sistemas en el entorno y el sistema que se especifica.

Los **sistemas externos generan** datos para el sistema o **consumen** datos del sistema. Pueden compartir datos con el sistema, conectarse directamente, a través de una red, o no conectarse en absoluto. Pueden estar físicamente juntos o separados.

Todas estas relaciones **afectan a los requerimientos y el diseño** del sistema a definir, por lo que deben tomarse en cuenta.



Ejemplo de diagrama de contexto

Modelos de Interacción

Una perspectiva donde se modela la **interacción** entre un sistema y su entorno, o entre los componentes de un sistema. Ayuda a identificar requisitos funcionales de usuario y requisitos no funcionales de rendimiento y confiabilidad.

- Diagramas de **caso de uso**, que exponen las interacciones entre un sistema y su entorno.
- Diagramas de **secuencias**, que muestran las interacciones entre los actores y el sistema, y entre los componentes del sistema.

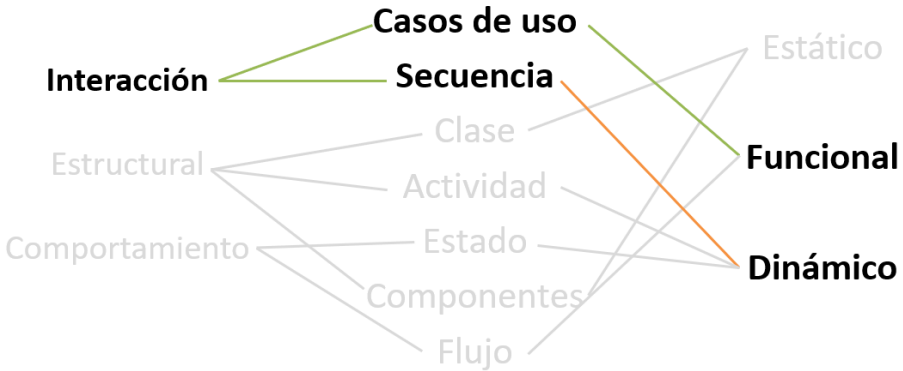


Diagrama de Casos de Uso

Los **casos de uso** se documentan con el empleo de un **diagrama de caso de uso** de alto nivel.

El conjunto de casos de uso representa **todas las interacciones** posibles que se describirán en los **requerimientos** del sistema.

Los **actores** en el proceso se representan como figuras sencillas. Cada clase de **interacción** se constituye como una elipse con etiqueta. Líneas vinculan a los actores con la interacción.

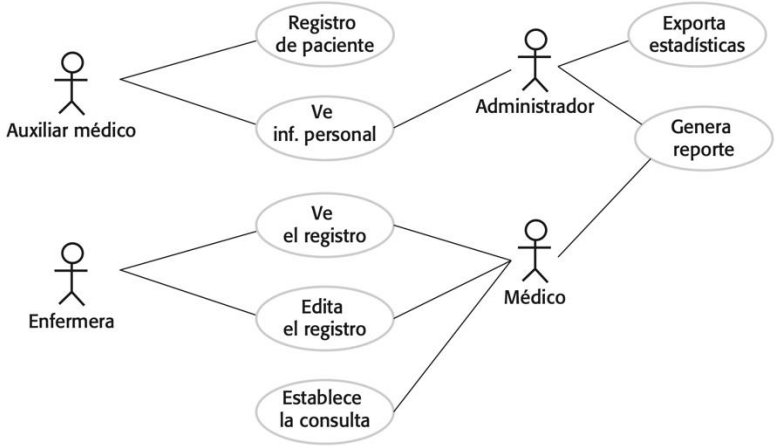


Diagrama de casos de uso.

Nota: Los casos de uso también son fundamentales en el enfoque 4+1 de Kruchten, donde actúan como la vista “+1” para validar y verificar que la arquitectura cumple con los requisitos funcionales, utilizando escenarios representativos.

Diagrama de Secuencia

Un **diagrama de secuencia** muestra la sucesión de **interacciones entre actores y objetos** (y estos entre sí), que ocurre durante una instancia de caso de uso.

- Los **objetos y actores** se mencionan en la parte superior del diagrama, con una línea punteada vertical.
- Las **interacciones** se indican con flechas dirigidas.
- El **rectángulo** indica la línea de vida de una instancia (tiempo está involucrada en la computación).

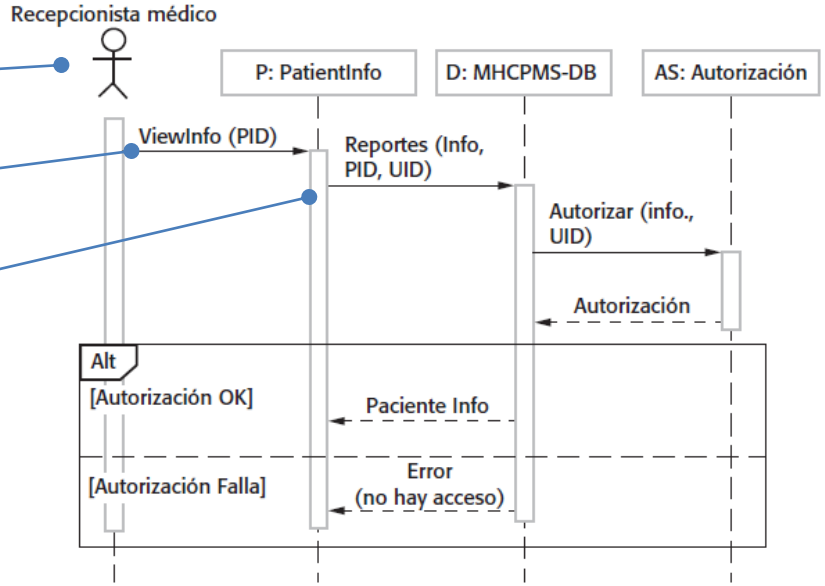


Diagrama de Secuencia

Un **diagrama de secuencia** muestra la sucesión de **interacciones entre actores y objetos** (y estos entre sí), que ocurre durante una instancia de caso de uso.

- Los posibles flujos **alternativos** se muestran con un recuadro marcado con **“alt”** con las condiciones indicadas entre corchetes.
- La **secuencia** de interacciones se lee de arriba abajo.

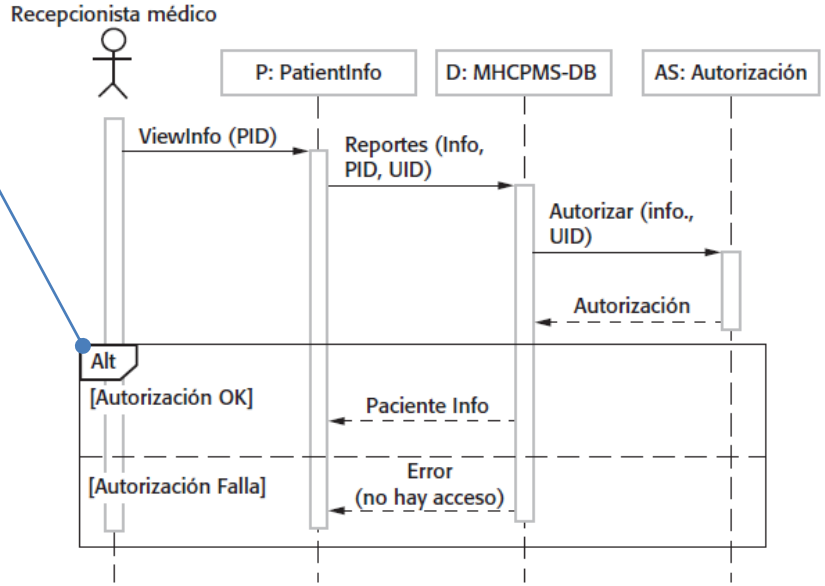


Diagrama de Secuencia

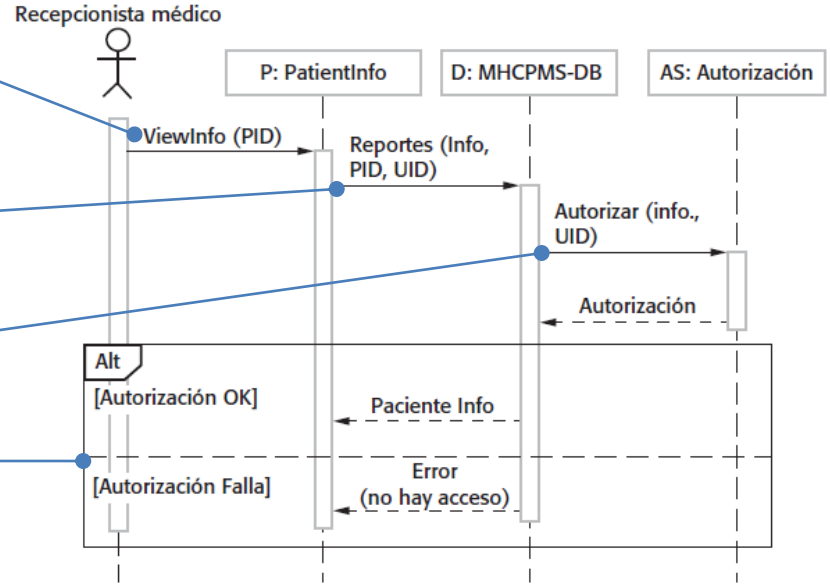
Ejemplo:

1. El recepcionista activa el método `ViewInfo` en una instancia `P` de la clase `PatientInfo` con el identificador del paciente, `PID`.

2. La instancia `P` llama a la base de datos y suministra el identificador del recepcionista para la verificación.

3. La base de datos comprueba, mediante un sistema de autorización, que el usuario esté autorizado.

4. Si está autorizado, devuelve la info y se muestra por pantalla. Si la autorización falla, se devuelve un mensaje de error.



Nota: UML tiene una amplia sintaxis para diagramas de secuencia, lo cual permite muchos tipos diferentes de interacción a modelar. Aquí sólo nos enfocaremos en lo básico de este tipo de diagrama.

Modelos Estructurales

Una perspectiva **estructural**, donde se modela la **organización de un sistema** o la **estructura de datos** que procese el sistema. Se utilizan diagramas de clases para mostrar la jerarquía y las relaciones entre los elementos.

- Diagramas de **clase**, que revelan las clases de objeto en el sistema y las asociaciones entre estas clases.
- Diagramas de **actividad**, que muestran las actividades incluidas en un proceso o en el procesamiento de datos.

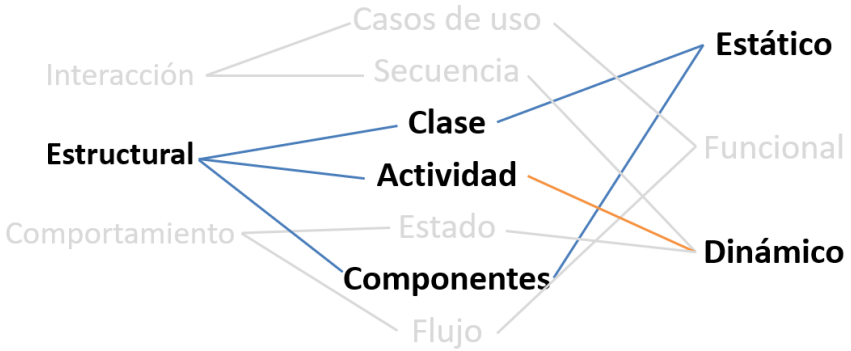
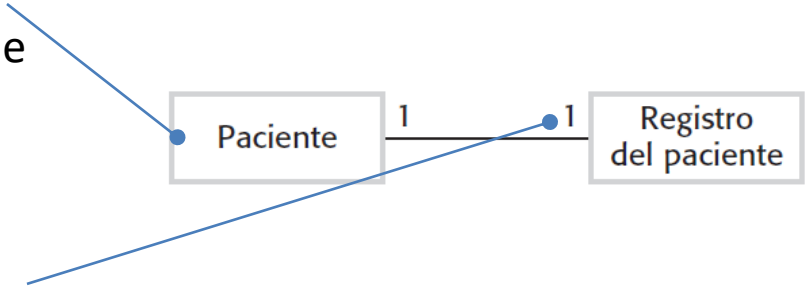


Diagrama de Clases

Los **diagramas de clase** se utilizan cuando se desarrolla un sistema orientado a objetos para mostrar las **clases** y las **asociaciones** entre ellas. La atención se centra en el **modelado de objetos del mundo real**, como parte de los requerimientos o los primeros procesos de diseño del software.

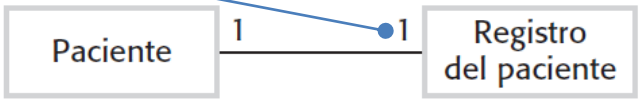
La primera etapa consiste en **identificar los objetos esenciales** y representarlos como clases. Esto se hace escribiendo el nombre de la clase en un recuadro.



Después, se anota la existencia de una **asociación** dibujando simplemente una línea entre las clases.

Diagrama de Clases

Se deben mostrar **cuántos objetos intervienen** en la asociación con un número en cada extremo de la asociación.



Ejemplo: Relación 1:1. Esto es, cada paciente tiene exactamente un registro, y cada registro conserva información precisa del paciente.

Son posibles otras multiplicidades. Se define un número exacto de objetos que están implicados, o bien, con el uso de un asterisco (*), que hay un número indefinido de objetos en la asociación.

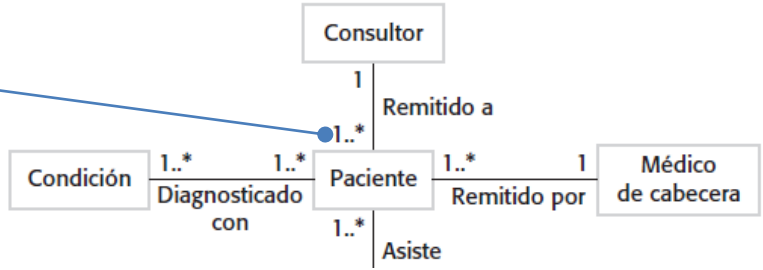


Diagrama de Clases

Es posible **nombrar las asociaciones** para dar un indicio del tipo de relación que existe.

Los diagramas de clase parecen **modelos semánticos de datos** que se usan en el diseño de bases de datos.

- Las **entidades** se entienden como **clases** de objeto simplificadas (no tienen operaciones)
- Los **atributos** como atributos de clase de objeto.
- Las **relaciones** como nombres de **asociaciones** entre clases de objeto.

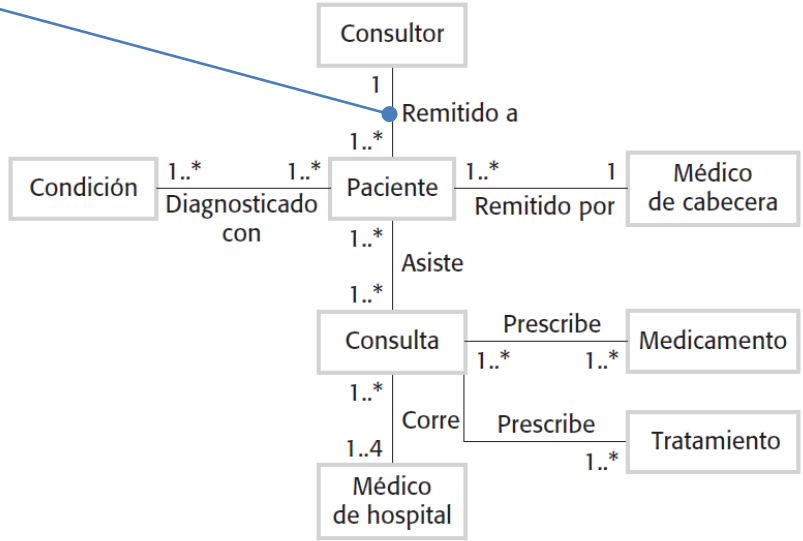


Diagrama de Clases

Para definir las clases con más detalle, se puede añadir información sobre **atributos** y **operaciones**.

En UML, los atributos y las operaciones se muestran al **extender el rectángulo** simple que representa una clase.

- El **nombre de la clase** de objeto está en la sección superior.
- Los **atributos** de clase están en la sección media. Esto debe incluir los nombres del atributo y, opcionalmente, sus tipos.
- Las **operaciones** (llamadas métodos en Python, Java y en otros lenguajes OO) asociadas con la clase de objeto están en la sección inferior del rectángulo.

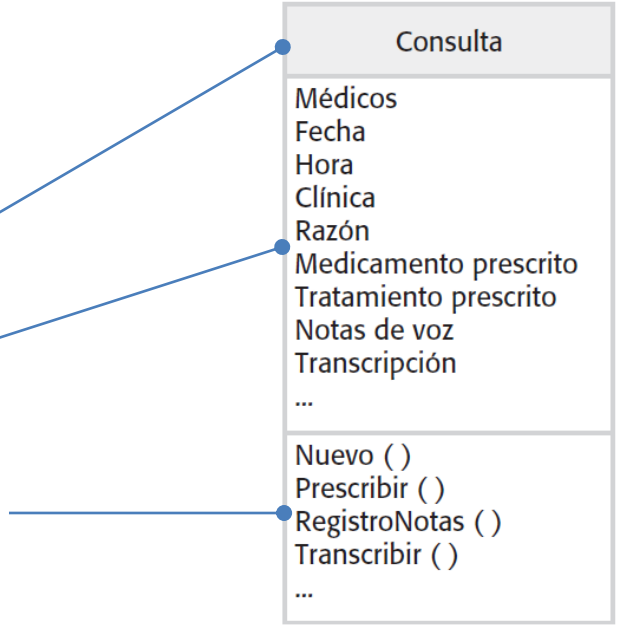


Diagrama de Clases – Generalización (Herencia)

La **generalización** se usa para gestionar la complejidad. En vez de definir las características detalladas de cada entidad, se definen clases más generales.

De esta forma, diferentes miembros de estas clases tienen **características comunes** (p.ej. las ardillas y ratas son roedores). Así, se definen aspectos generales que apliquen a todos los miembros de la clase (p.ej. todos los roedores tienen dientes para roer).

Ésta es una buena práctica de diseño, si se hacen cambios, las características comunes se propagan. En los lenguajes OO (Python), la generalización se implementa usando **herencia de clase**.

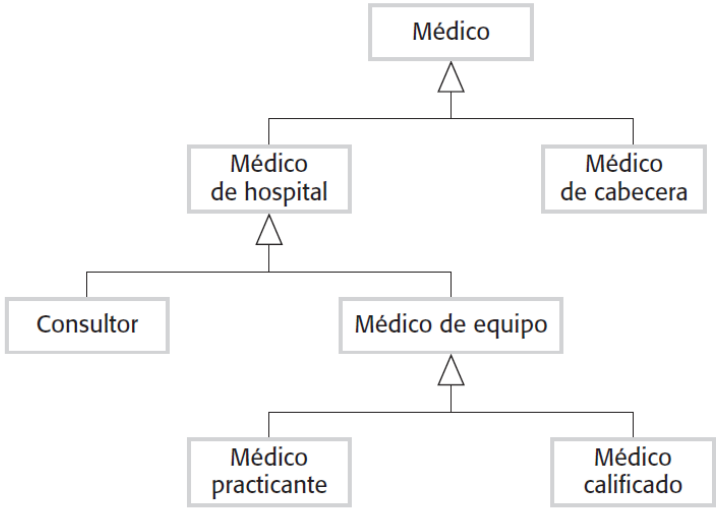


Diagrama de Clases – Generalización (Herencia)

La generalización se muestra como una **flecha que apunta hacia la clase más general**.

Ejemplo: Los médicos de cabecera y los médicos de hospital pueden generalizarse como médicos, y que hay tres tipos de médicos de hospital:

- Quienes se graduaron recientemente y son supervisados (médicos practicantes);
- Quienes trabajan sin supervisión como parte de un equipo de consultores (médicos registrados)
- Los consultores, que son médicos experimentados.

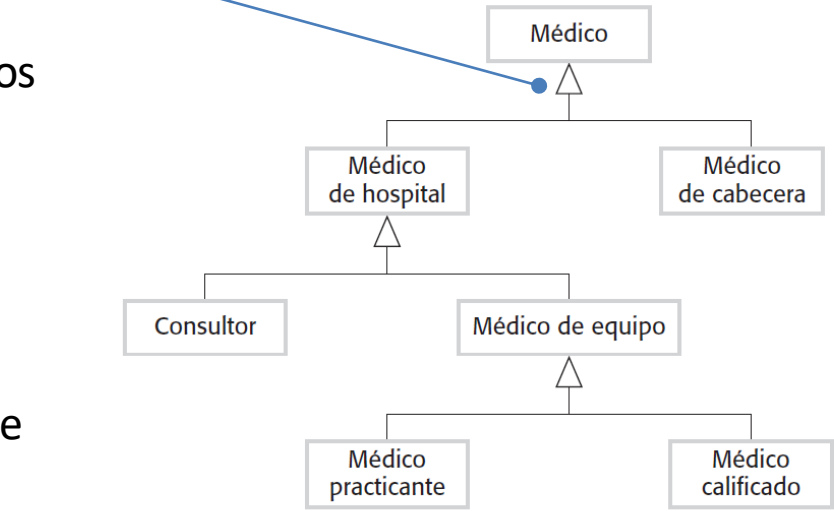


Diagrama de Clases – Generalización (Herencia)

Los atributos y las operaciones de las clases de nivel superior también se asocian con las clases de nivel inferior. Éstas son **subclases** que **heredan** los atributos y las operaciones de sus **superclases** y agregan atributos y operaciones más **específicos**.

Ejemplo: todos los médicos tienen un nombre y número telefónico.

- Todos los médicos de hospital tienen un número de personal y un departamento.
- Pero los médicos de cabecera no tienen tales atributos, pues trabajan de manera independiente. Sin embargo, sí tienen un nombre de consultorio y dirección.

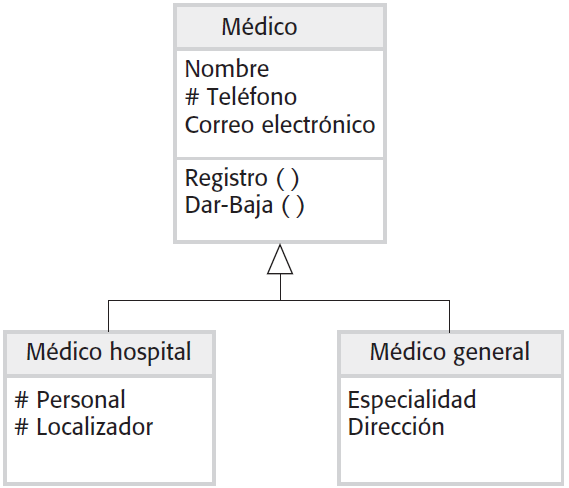
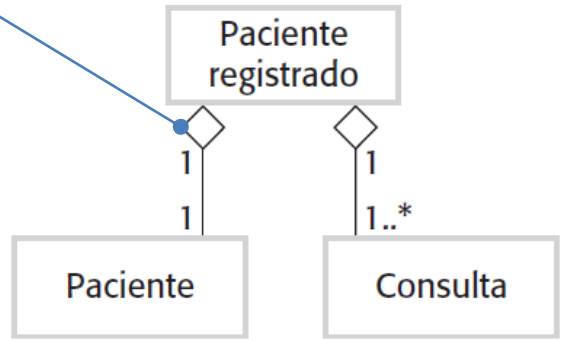


Diagrama de Clases – Agregación

Los objetos en el mundo real con frecuencia están **compuestos** por diferentes partes. UML proporciona un tipo especial de **asociación** entre clases llamado **agregación**, que significa que un objeto (el todo) se compone de otros objetos (las partes).

Para mostrarlo, se usa un **trazo en forma de diamante**, junto con la clase que representa el todo.



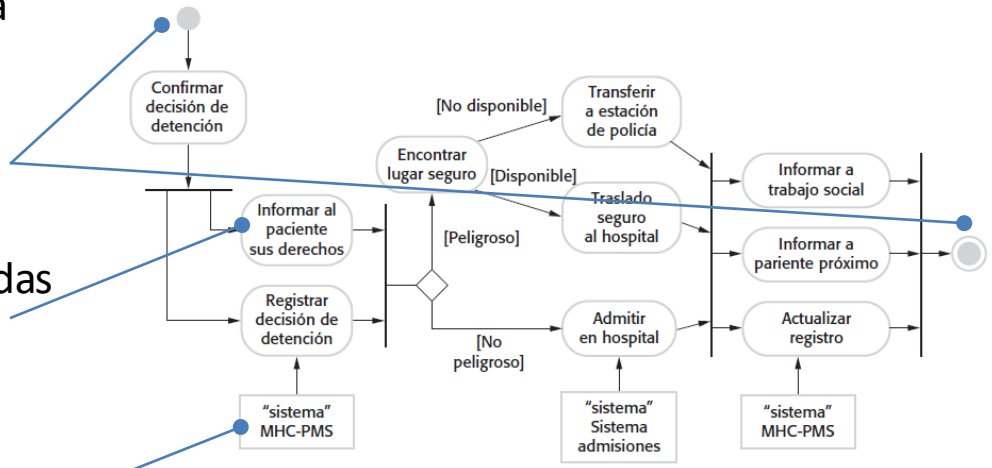
Ejemplo: Un registro de paciente es una composición de Paciente (Patient) y un número indefinido de Consulta (Consultations).

Ejemplo: Un paquete de estudio para un curso, está compuesto por libro, diapositivas de PowerPoint, exámenes y recomendaciones para lecturas posteriores.

Diagrama de Actividad

Los **diagramas de actividad** intentan mostrar las **actividades** que incluyen un proceso de sistema, así como el **flujo de control** de una actividad a otra.

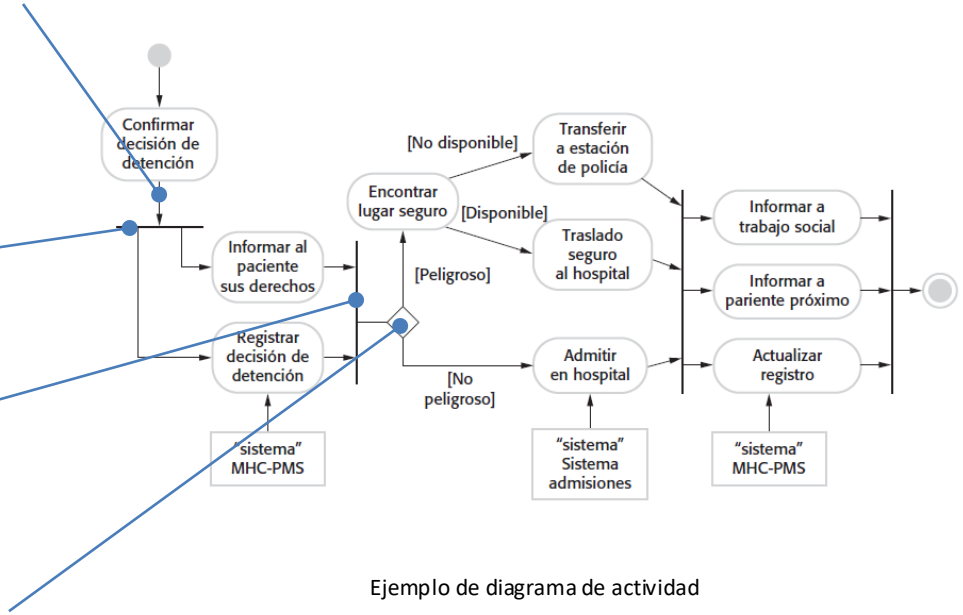
- El **inicio** de un proceso se indica con un **círculo** lleno; el **fin**, mediante un círculo lleno dentro de otro círculo.
- Los **rectángulos** con esquinas redondeadas representan **actividades**, esto es, los subprocesos específicos que hay que realizar.
- Se pueden incluir otros objetos en los gráficos de actividad. Por ejemplo, para mostrar los **sistemas** que sirven para apoyar diferentes procesos.



Ejemplo de diagrama de actividad

Diagrama de Actividad

- Las **flechas** representan el flujo de trabajo entre actividades.
- Una **barra sólida** se emplea para indicar coordinación de actividades.
  - Cuando el flujo de una barra sólida conduzca a algunas actividades, éstas pueden ejecutarse en **paralelo**.
  - Cuando el flujo varias actividades se dirige a una barra sólida, todas deben completarse antes del avance.
- Un rombo indica dos (o más) caminos **exclusivos**. Las flechas pueden **anotarse con guardas** que indiquen la condición al tomar dicho flujo.



Ejemplo de diagrama de actividad

Modelos de Comportamiento

Una perspectiva de **comportamiento**, donde se modele el comportamiento dinámico del sistema y cómo responde ante ciertos eventos.

- Diagramas de **estado**, que explican cómo reacciona el sistema frente a eventos internos y externos.
- Diagramas de **flujo de datos**, que presentan una perspectiva funcional, donde cada transformación constituye una sola función o un solo proceso.

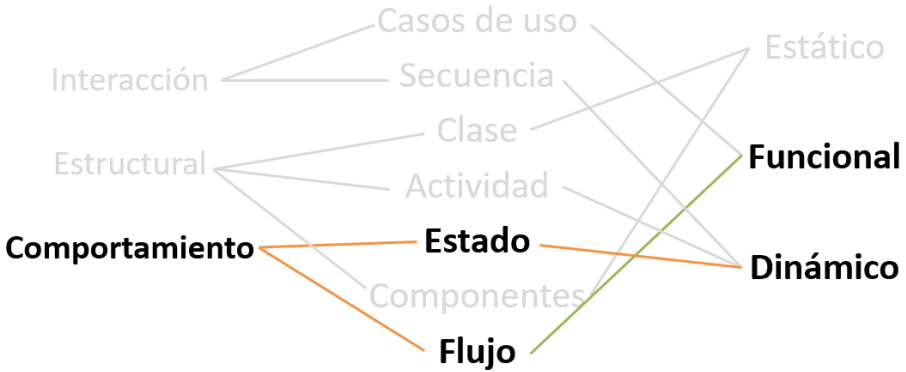
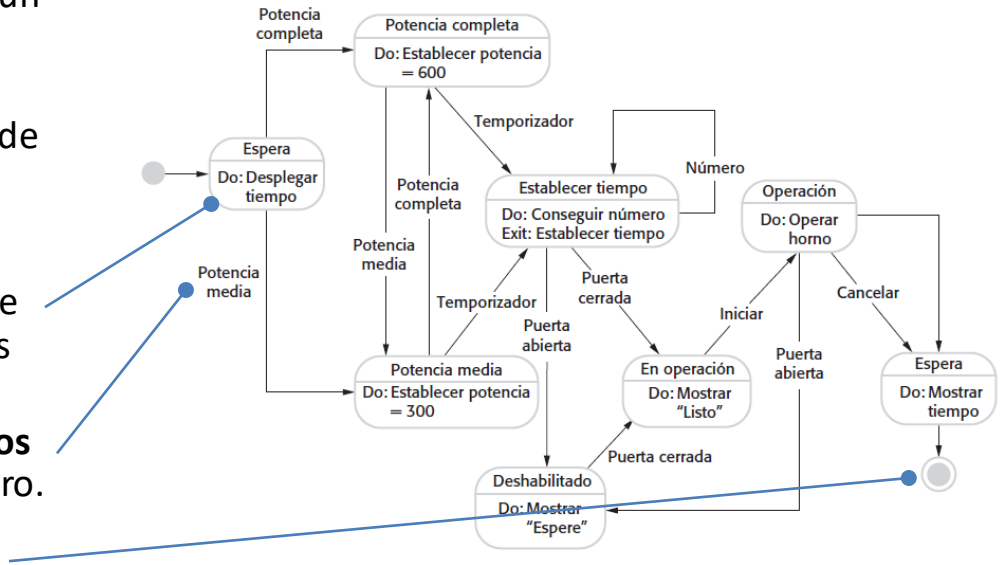


Diagrama de Estado

Los **diagramas de estado** muestran **estados** y **eventos** del sistema que causan **transiciones** de un estado a otro.

No exponen el flujo de datos dentro del sistema, pero suelen incluir información adicional acerca de los cálculos realizados en cada estado.

- Los **rectángulos redondeados** representan **estados** del sistema. Pueden incluir una breve descripción (después de "do") de las acciones que se tomarán en dicho estado.
- Las **flechas etiquetadas** representan **estímulos** que fuerzan una **transición** de un estado a otro.
- Los **estados inicial** y **final** se indican usando **círculos** rellenos, como en los diagramas de actividad.



Ejercicio: Aplicación de conceptos

Aplicar los **conceptos de modelado** que acabamos de ver al software que habéis desarrollado en la primera práctica de la asignatura. El objetivo es definir los cinco modelos principales para vuestra propuesta de sistema distribuido de forma sencilla

Instrucciones:

- **Diagramas de caso de uso,** que exponen las interacciones entre un sistema y su entorno.
- **Diagramas de secuencias,** para un caso de uso.
- **Diagramas de clase,** debe incorporar todos los aspectos vistos. Generalización, Agregación y detalles de atributos y métodos.
- **Diagramas de actividad,** que muestre el proceso general de vuestro sistema.

Universidad Intercontinental de la Empresa

Asignatura

Arquitectura del Software 

Profesor

Yago Fontenla Seco

{*yago.fontenla1@uie.edu*}