Diseño de Software

Mega resumen del Parcial 1 · semanas 1 a 7 · revisado lámina por lámina contra los siete mazos oficiales (28.04.2026 a 09.06.2026)

Qué es este documento y cómo se construyó

Antes de escribir una línea revisé los **siete mazos oficiales** del periodo previo a la semana 8 y los **seis resúmenes de sesión** que hay en el proyecto. El inventario lámina por lámina está en el Apéndice A: son **181 láminas**, de las cuales 12 son administrativas (políticas del curso) y 21 son de portada, objetivos de aprendizaje o resumen de clase. Las 148 de contenido tienen todas asignada una sección de este documento.

Está organizado por **tema**, no por fecha de sesión. Las siete sesiones se reagrupan en seis partes temáticas más un puente hacia el Parcial 2. Un mismo tema puede venir de dos semanas distintas: los diagramas UML, por ejemplo, se reparten entre las semanas 5, 6 y 7.

Hallazgo de procedencia: para este periodo no existen resúmenes de sesión

Los seis resúmenes de sesión del proyecto (arquitectura ×2, acoplamiento y cohesión, arquitecturas distribuidas, control de versiones y Git, síntesis de arquitectura) cubren **exclusivamente material de la semana 8 en adelante**. Los revisé uno por uno buscando términos del Parcial 1: UML, casos de uso, cascada, espiral, RUP, SDLC, requerimientos, SWEBOK, escenarios, KISS. Las coincidencias son falsos positivos — «composición» aparece en el resumen de acoplamiento, pero hablando de composición sobre herencia en los patrones GoF, no de la relación de composición de UML.

Consecuencia para las marcas: a diferencia del resumen maestro de la Parte 2, aquí no hay material previo mío que pueda contradecir una lámina. Por eso casi todo el documento es DIAPOSITIVAS. Lo poco marcado MATERIAL PROPIO es andamiaje que agrego ahora, y va señalado como tal.

Cómo leer las marcas

Marca	Qué significa
DIAPOSITIVAS	Todo lo de la sección está respaldado por una lámina del curso (semanas 1 a 7). Es lo que debes responder en el examen.
MIXTO	Base de las diapositivas, ampliada por mí ahora. Lo añadido va señalado dentro de la sección.
MATERIAL PROPIO	No aparece en ninguna lámina del periodo. Útil para entender y para preguntas abiertas, pero no lo cites como si fuera del curso.
★	Dentro de una sección MIXTO o propia, marca la fila o el punto concreto que sí tiene respaldo de lámina.
Caja ámbar	Corrección o conflicto: dice qué se creía antes o qué dice la lámina, qué es correcto, y cómo responder en el examen. Todas se recogen en el Apéndice B.

Dato de calendario que está en las láminas

La lámina 13 de la semana 1 fija las fechas: **Examen 1 el 16.06.2026** y **Examen 2 el 11.08.2026**. La ponderación es 30 % actividades en clase, 30 % talleres, casos de estudio y proyectos, y 40 % evaluación parcial. Para aprobar hacen falta 70/100 puntos como promedio de los dos parciales.

Parte 1 · La ingeniería de software como disciplina

Toda esta parte sale de la semana 1 (28.04.2026), láminas 14 a 32.

1.1 Tres definiciones oficiales

DIAPOSITIVAS · semana 1, láminas 18 a 21

La profesora no da una definición, da tres y luego una síntesis. Las tres aparecen en láminas separadas, así que las tres son citables.

Fuente	Definición
Ian Sommerville, 2011	«Una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software.» (libro *Ingeniería de software*)
Norma ISO/IEC 29110	«Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.»
Barry W. Boehm	«The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them.»

La síntesis de la lámina 21: una disciplina de ingeniería centrada en todos los aspectos relacionados con la producción de software, **desde la especificación del sistema hasta el mantenimiento** una vez que está operativo. Y una idea que la profesora subraya: **cada sistema es diferente**, por lo que se deben usar herramientas y técnicas adecuadas para su desarrollo.

> **Ojo con la atribución de la definición de la lámina 19**
>
> **Qué dice la lámina:** atribuye esa formulación a la norma ISO/IEC 29110.
>
> **Qué conviene saber:** la redacción es literalmente la definición clásica del estándar **IEEE 610.12-1990**, hoy recogida en el vocabulario ISO/IEC/IEEE 24765; la 29110 la reutiliza. No es un error de la lámina, la 29110 sí la contiene.
>
> **Qué responder:** si te preguntan la fuente, contesta **ISO/IEC 29110** como dice la lámina. Si la pregunta es abierta puedes añadir que el origen de la formulación es el estándar IEEE. No cambies la respuesta cerrada por este matiz.

1.2 Elementos involucrados en el desarrollo de un software

DIAPOSITIVAS · semana 1, lámina 22

La lista de la lámina es abierta (termina en puntos suspensivos), lo cual importa: si te piden «mencione elementos», cualquier subconjunto vale.

- Requerimientos
- Documentación de usuario
- Archivos de configuración
- Código fuente

La idea de fondo: el software no es solo el código. Es todo el paquete.

1.3 Las cuatro características esenciales de un buen software

DIAPOSITIVAS · semana 1, láminas 23 a 27

Esto es material de **recall duro**: cuatro palabras, cada una con lámina propia. La lista completa es **mantenible, confiable, eficiente, usable**.

Característica	Qué dice la lámina
Mantenible	El código se debe escribir de manera que pueda adaptarse a las necesidades cambiantes de los usuarios. El cambio del software es **inevitable** en un entorno empresarial.
Confiable	Debe ofrecer seguridad y protección. En caso de falla, el software no debe causar daño físico ni económico. Los usuarios con malas intenciones no deben tener posibilidad alguna de ingresar o afectar al sistema.
Eficiente	Debe hacer buen uso de los recursos del sistema (memoria, ciclos del procesador). Se enfoca en tiempo de respuesta, capacidad de respuesta y uso de memoria.
Usable	Debe ser amigable con el usuario: intuitivo, fácil de usar y compatible con otros sistemas que los usuarios utilicen.

> **No confundir estas cuatro con las otras dos listas de «calidad» del curso**
>
> En el Parcial 1 hay **dos** listas de calidad y en el Parcial 2 hay una tercera. Se parecen tanto que bajo presión se mezclan. Distinguirlas por el **nivel al que aplican**:
>
> **1)** Características de un buen **software** — 4 puntos, semana 1: mantenible, confiable, eficiente, usable. Hablan del producto terminado.
>
> **2)** Guías de calidad del **diseño** — 8 puntos, semana 7 (ver sección 6.8). Hablan del artefacto de diseño, no del software.
>
> **3) Atributos de calidad** — 5 puntos, semana 8, ya fuera de este documento: funcionalidad, usabilidad, fiabilidad, desempeño, soportabilidad. Es la lista de la Parte 2.
>
> **Qué responder:** fíjate en el sustantivo de la pregunta. «Buen software» → las 4. «Buen diseño» → las 8 guías. «Atributos de calidad» → las 5 de la semana 8.

1.4 Las fronteras de la disciplina

DIAPOSITIVAS · semana 1, láminas 28 y 29

Comparación	Qué dice la lámina
vs. Ingeniería de sistemas	La ingeniería de sistemas se enfoca en desarrollo de hardware, diseño de políticas y procesos, e implementación del software. La ingeniería de software está involucrada **en la parte de los procesos**.
vs. Ciencias de la computación	Las ciencias de la computación se centran en la **teoría y los principios fundamentales**. La ingeniería de software se centra en el **desarrollo y la distribución** de software.

Regla mnemónica para el examen: **sistemas es más ancho** (incluye hardware), **ciencias es más profundo** (teoría), **ingeniería de software es la práctica de producir**.

1.5 Tipos de productos de software

DIAPOSITIVAS · semana 1, lámina 30

Tipo	Definición	Ejemplos de la lámina
Genéricos	Sistemas implementados por una empresa de desarrollo, que se venden a **cualquier** cliente.	Bases de datos, procesadores de texto, herramientas para gestión de proyectos
Personalizados	Sistemas implementados para **un cliente en particular**, acorde a sus necesidades.	Sistemas creados para llevar a cabo un proceso específico de una empresa

1.6 Encuadre del curso

DIAPOSITIVAS · semana 1, láminas 2 a 14 y 31

Las láminas 2 a 13 son administrativas (asistencia, puntualidad, calificaciones, dispositivos en clase, herramientas, fechas). No son materia de examen, pero las inventarío en el Apéndice A para que la cobertura quede completa. Dos cosas sí conviene retener:

- **El contenido del curso son cinco unidades** (lámina 14): 1. Introducción a metodologías e ingeniería de software · 2. Diagramas UML · 3. Principios de diseño de software · 4. Arquitecturas de software · 5. Tecnologías para desarrollo de software. Las unidades 1 y 2 son el Parcial 1; las 3, 4 y 5 son el Parcial 2.
- **Material extra** (lámina 31): el OWASP Top Ten, enlazado sin desarrollar. Se conecta con el «software confiable» de la lámina 25.

Parte 2 · Proceso y ciclo de vida del software

Toda esta parte sale de la semana 2 (05.05.2026), 44 láminas. Es el mazo más grande del parcial y el que más listas numeradas contiene.

2.1 La crisis del software

DIAPOSITIVAS · semana 2, láminas 3 y 4

Término que surgió a **finales de 1960** y se usó **hasta 1980**, a raíz de los problemas que se presentaban en el desarrollo de nuevo software. Los cuatro síntomas que enumera la lámina:

- Los sistemas se terminaban **fuera del tiempo** establecido para su desarrollo
- Presentaban **fallas**
- Eran **difíciles de utilizar**
- Su **mantenimiento era complejo y costoso**

Esta es la justificación histórica de todo lo demás: el ciclo de vida y los modelos de proceso existen como respuesta a esta crisis.

2.2 SWEBOK

DIAPOSITIVAS · semana 2, láminas 5 y 6

Software **E**ngineering **Bo**dy **o**f **K**nowledge. Guía del cuerpo de conocimiento de ingeniería de software **promovida por la IEEE**, que detalla el conocimiento existente sobre la disciplina.

Áreas de conocimiento que lista la lámina 6 (lista abierta): requerimientos, diseño, construcción de software, pruebas, mantenimiento, gestión de configuración, métodos y modelos, calidad.

> **La lámina 5 está desactualizada en la versión de SWEBOK**
>
> **Qué dice la lámina:** «Versión actual: 3.0».
>
> **Qué es cierto hoy:** la IEEE Computer Society publicó **SWEBOK v4.0 en 2024**, con 18 áreas de conocimiento y tres nuevas (Software Architecture, Software Engineering Operations y Software Security), además de incorporar ágil y DevOps. En septiembre de 2025 salió la v4.0a.
>
> **Qué responder en el examen:** si la pregunta es cerrada y sale de la lámina, responde **3.0**. Si es abierta, di 3.0 según el material del curso y menciona que ya existe la v4.0. Es exactamente el mismo criterio que aplicamos con el hash por defecto de Git en la Parte 2: la lámina manda en la respuesta cerrada.

2.3 Ciclo de vida del software (SDLC)

DIAPOSITIVAS · semana 2, láminas 7 a 9

En inglés **SDLC**, *Software Development Life Cycle*. Consiste en las distintas fases que se llevan a cabo para desarrollar un sistema. **Cada fase produce entregables** que se utilizan en las fases siguientes.

> **Un dato de la lámina 7 que es candidato clarísimo a pregunta**
>
> «También se lo conoce como **proceso de software**, aunque originalmente el término se acuñó para referirse al **modelo en cascada**.» Es el tipo de detalle histórico que se pregunta como verdadero/falso.

Por qué importa (lámina 8): el desarrollo se puede tornar difícil por requisitos variables, cambios en la tecnología y trabajo colaborativo. Usar una metodología facilita la administración del proyecto mediante una planificación que asegura la entrega de resultados en cada fase.

Tres ventajas de usar una metodología SDLC (lámina 9): eficiencia en la planificación, estimación y programación de los entregables; aumento en la satisfacción de los clientes; el proceso de desarrollo es más visible para todos los involucrados.

2.4 Las seis fases del SDLC

DIAPOSITIVAS · semana 2, láminas 10 a 18

Cada fase tiene lámina propia con tres o cuatro actividades. La tabla junta las seis.

#	Fase	Actividades que enumera la lámina
1	**Planificación y análisis**	Recolección de información de las partes involucradas · definición de objetivos y alcance · identificación de requisitos · planificación de recursos
2	**Diseño**	Análisis de los requerimientos · diseño de la arquitectura · diseño de interfaz de usuarios
3	**Implementación**	Desarrollo del sistema · asignación de tareas · asignación de fechas de entrega
4	**Pruebas**	Detección de errores · análisis de calidad del software · pruebas y desarrollo **en paralelo**
5	**Despliegue**	Configuración de servidores · instalación de herramientas · verificación del funcionamiento. Puesta en producción al finalizar implementación y pruebas
6	**Mantenimiento**	Desarrollo de nuevas funcionalidades · corrección de errores · identificación de mejoras (seguridad, rendimiento)

Entornos de desarrollo — láminas 16 y 17

La lámina 16 establece la separación: **los desarrolladores trabajan en un entorno de pruebas o compilación; los usuarios trabajan en el entorno de producción**. La lámina 17 es un gráfico sin texto que muestra los entornos usados en desarrollo de software. Si te preguntan por la cadena típica, la respuesta segura es la de la lámina 16: **pruebas/compilación** → **producción**. ★ Esos dos son los que la lámina nombra.

Tres listas de fases en el mismo mazo — la trampa interna de la semana 2

La semana 2 da **tres** descomposiciones distintas del desarrollo y es muy fácil responder con la que no toca:

Seis fases del SDLC (lámina 11): planificación y análisis, diseño, implementación, pruebas, despliegue, mantenimiento.

Cuatro actividades esenciales del proceso de software (lámina 20): especificación, diseño e implementación, validación, evolución.

Cinco fases del modelo en cascada (láminas 25 a 27): análisis y definición de requerimientos; diseño del sistema y del software; implementación y prueba de unidad; integración y prueba de sistema; operación y mantenimiento.

Qué responder: cuenta las opciones que te dan. Si la pregunta dice «fases del ciclo de vida» son 6. Si dice «actividades esenciales» son 4. Si nombra cascada son 5. Nota menor: la lámina 11 tiene una errata y se titula «Fases del **DLC**», sin la S.

2.5 Proceso de software y las cuatro actividades esenciales

DIAPOSITIVAS · semana 2, láminas 19 y 20

Proceso de software: conjunto de actividades relacionadas cuyo objetivo es crear un producto de software. El software puede empezar **desde cero** o ser la **extensión de un sistema existente**.

Existen diversos procesos, pero al menos deben incluir estas cuatro:

#	Actividad esencial
1	**Especificación** del software
2	**Diseño e implementación** del software
3	**Validación** del software
4	**Evolución** del software

2.6 Modelos de proceso de software

DIAPOSITIVAS · semana 2, láminas 21 y 22

Definición: representación abstracta y estructurada del proceso de desarrollo de software. Sirve como guía para desarrollar de forma sistemática y efectiva.

La lámina 22 lista cinco (lista abierta): **cascada** (waterfall), **espiral**, **modelo V**, **incremental**, **ágil**.

De los cinco modelos listados, solo cuatro se desarrollan

La lámina 22 nombra cinco modelos, pero el mazo desarrolla **cascada, espiral, RUP e incremental**. Fíjate en el desajuste: **RUP se desarrolla sin estar en la lista** (láminas 36 a 41), y **el modelo V y ágil se listan pero nunca se explican**.

Qué responder: si te piden «mencione modelos de proceso», la respuesta segura es la lista de la lámina 22 — cascada, espiral, V, incremental, ágil — y puedes añadir RUP porque tiene seis láminas propias. Si te piden **describir** uno, es casi seguro que será cascada, espiral, RUP o incremental, porque son los únicos con contenido. La sección 2.11 cubre el V y ágil por si acaso, marcada como material propio.

2.7 Modelo en cascada

DIAPOSITIVAS · semana 2, láminas 23 a 28

También conocido como **waterfall model**. Modelo derivado de procesos más generales de software.

- **En teoría** una fase debe terminar para comenzar la siguiente; **en la práctica las etapas se traslapan**. Este matiz es de la lámina y es muy preguntable.
- En cada fase se produce documentación, que es posible que deba modificarse como resultado de cambios en otras fases.

Las cinco fases (láminas 25 a 27):

Fase	Qué ocurre
Análisis y definición de requerimientos	Definición de la especificación del software según la información obtenida de los usuarios
Diseño del sistema y del software	Definición de la arquitectura del sistema · identificación y descripción de las abstracciones principales del software y sus relaciones
Implementación y prueba de unidad	El software se implementa en partes o unidades del programa. Por cada programa se realiza la prueba de unidad, que verifica que cumpla con la especificación
Integración y prueba de sistema	Los programas individuales se integran y se prueba el sistema completo para verificar el cumplimiento de los requerimientos. Incluye la **entrega al cliente**
Operación y mantenimiento	Corrección de errores no detectados en fases anteriores · mejora de funcionalidades existentes · implementación de nuevos requerimientos

Consideraciones (lámina 28) — cuatro puntos, dos a favor y dos en contra:

- Se debe usar **siempre y cuando los requerimientos estén claros** y sea improbable que haya cambios.
- Apropiado para **proyectos a corto plazo** y si el **equipo de trabajo es pequeño**.
- **El usuario no es considerado** durante el proceso de desarrollo del sistema.
- Puede resultar **oneroso** en caso de un rediseño significativo, ya que las pruebas se realizan casi al final del desarrollo.

2.8 Modelo en espiral

DIAPOSITIVAS · semana 2, láminas 29 a 35

Propuesto por **Barry W. Boehm en 1988**. El proceso de software se representa en forma de espiral y **se basa en el riesgo**. Cada ciclo de la espiral representa una fase del proceso de software.

Cuándo es apropiado: proyectos grandes, con alto grado de cambios, donde es esencial el control de riesgos y costos. Es el espejo exacto de cascada.

Riesgos, ejemplos de la lámina 30: estimación del tiempo, recursos humanos, cambios en el alcance del proyecto, diseño inapropiado.

Los cuatro sectores (láminas 32 a 35) — el orden importa:

#	Sector	Qué se hace
1	**Establecimiento de objetivos**	Definición de objetivos específicos para esa fase · identificación de restricciones en el proceso y el producto · identificación de riesgos del proyecto
2	**Valoración y reducción del riesgo**	Análisis de los riesgos identificados · definición de acciones para reducirlos · si hay incertidumbre en los requerimientos, se pueden crear **prototipos** del sistema
3	**Desarrollo y validación**	Selección de un modelo de desarrollo. Si el riesgo predomina en la interfaz de usuarios → prototipos. Si el riesgo principal es la integración de subsistemas → **se puede usar el modelo en cascada**
4	**Planeación**	Se revisa el proyecto y se decide si continuar con otro ciclo. Si se continúa, se trazan los planes para la siguiente fase

> **El detalle más fino del sector 3**
> La lámina 34 dice que dentro de la espiral **se puede usar cascada** como modelo de desarrollo cuando el riesgo principal es la integración de subsistemas. Es decir: espiral no es alternativa a cascada, es un marco que puede contener a cascada. Si aparece una pregunta del tipo «espiral y cascada son mutuamente excluyentes», la respuesta es **falso**.

2.9 Proceso unificado racional (RUP)

DIAPOSITIVAS · semana 2, láminas 36 a 41

Rational **U**nified **P**rocess. Modelo de proceso **híbrido**, porque combina prácticas y métodos para el desarrollo del software, **como el uso de UML**. Apropiado para **proyectos de gran envergadura con equipos grandes**.

Fase	Qué ocurre
Concepción	Definición del caso de uso del software · identificación de todos los involucrados (personas y/o sistemas) y las interacciones entre ellos · valoración de la aportación del sistema para la organización: **si la aportación no es significativa, se puede cancelar el proyecto**
Elaboración	Definición de la **arquitectura** del sistema · diseño del plan de desarrollo · identificación de los riesgos principales
Construcción	Desarrollo del sistema **en paralelo** · integración de las partes · elaboración de la documentación para entregar al usuario
Transición	Verificación final de que el sistema cumple los requerimientos y funciona correctamente, previo a la puesta en producción · **capacitación** a los usuarios

Dos anclas para no perder RUP: es el modelo que **trae UML** (y por eso conecta con las semanas 5 a 7) y es el único donde **cancelar el proyecto** aparece como salida explícita de una fase.

2.10 Desarrollo incremental

DIAPOSITIVAS · semana 2, láminas 42 y 43

Basado en la idea de realizar una **implementación inicial**, presentarla al usuario para recibir comentarios, y desarrollar **varias versiones** hasta que el sistema sea adecuado.

La lámina describe el movimiento así: avanzar en un conjunto de pasos hacia la solución y **retroceder si se detectan errores**. Por eso es más barato y sencillo realizar cambios en el sistema a medida que se va implementando.

2.11 Modelo V y ágil: mencionados pero no desarrollados

MATERIAL PROPIO · ampliación mía, sin lámina

> **Todo lo de esta sección es material propio**
> El modelo V y ágil aparecen **únicamente** en la lista de la lámina 22. Ninguna lámina del Parcial 1 los explica. Lo que sigue lo agrego yo para que no te tome por sorpresa una pregunta abierta; **no lo cites como contenido del curso**.

Modelo	Idea central (material propio)
Modelo V	Variante de cascada donde cada fase de desarrollo se empareja con un nivel de prueba: requerimientos ↔ pruebas de aceptación, diseño de alto nivel ↔ pruebas de integración, diseño detallado ↔ pruebas unitarias. La forma de V sale de bajar por desarrollo y subir por verificación. Corrige el defecto de cascada que la lámina 28 sí señala: que las pruebas quedan casi al final.
Ágil	Familia de enfoques iterativos e incrementales que priorizan entregas cortas, colaboración con el cliente y respuesta al cambio sobre la planificación rígida. Scrum y Kanban son los marcos más comunes. Se relaciona directamente con el desarrollo incremental de la lámina 42.

Parte 3 · Ingeniería de requerimientos

Esta parte funde las semanas 3 (12.05.2026, 13 láminas) y 4 (19.05.2026, 22 láminas). La semana 3 da el qué y el por qué; la semana 4 da los tipos y el proceso.

3.1 Qué es un requerimiento

DIAPOSITIVAS · semana 3, láminas 4 y 5

Definición de la lámina 5: especificación **precisa y detallada** de una función o característica que debe realizar un sistema para satisfacer las expectativas del cliente, usuario o partes interesadas. Los requerimientos son fundamentales porque sirven como **la base sobre la cual se construye y se verifica** un sistema.

> **La cita de apertura de la semana 3 — lámina 4**
>
> «La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con gente, máquinas y otros sistemas. Ninguna otra parte del trabajo afecta tanto al sistema si es hecha mal. Ninguna es tan difícil de corregir más adelante… Entonces, la tarea más importante que el ingeniero de software hace para el cliente es la **extracción iterativa y el refinamiento** de los requerimientos del producto.»
>
> **Frederick P. Brooks**, *No Silver Bullet*, **1987**. Autor, título y año son los tres datos citables.

3.2 Por qué importan los requerimientos

DIAPOSITIVAS · semana 3, láminas 6 y 7

Cinco razones repartidas en dos láminas. Van juntas porque es una sola lista partida.

- Ayudan a **definir el alcance** del proyecto.
- Proporcionan un **lenguaje común** para la comunicación entre clientes, usuarios y equipo de desarrollo, y así poder discutir las necesidades y expectativas del proyecto.
- Facilitan la **verificación y validación** del software.
- Permiten **estimar con mejor precisión los costos y tiempos** requeridos para completar el proyecto.
- Ayudan a **reducir los riesgos** del proyecto.

3.3 Requerimientos del usuario y del sistema

DIAPOSITIVAS · semana 3, láminas 8 a 12

Nivel	Definición de la lámina
Del usuario	Descripción utilizando **lenguaje natural** acerca de los servicios, funcionalidades y restricciones que los usuarios esperan que se realicen en el sistema.
Del sistema	Descripción **detallada** de las funcionalidades, servicios y restricciones del sistema. La documentación creada con base en estos requerimientos se llama también **especificación funcional**.

El criterio de separación es el **nivel de detalle y el lenguaje**: natural y para el usuario, frente a detallado y para el equipo técnico.

> **El ejemplo del MHC-PMS — láminas 10 a 12**
>
> El mismo requerimiento expresado en los dos niveles. Sistema de administración de pacientes para apoyar la atención a la salud mental, de Sommerville, novena edición.
>
> **Requerimiento del usuario (uno solo):** «El sistema elaborará mensualmente informes administrativos que revelen el costo de los medicamentos prescritos por cada clínica durante el mes.»
>
> **Requerimientos del sistema (cuatro derivados del anterior):** 1) En el último día laboral de cada mes se redactará un resumen de los medicamentos prescritos, su costo y las clínicas que los prescriben. 2) El sistema elaborará automáticamente el informe, que se imprimirá después de las 17:30 del último día laborable del mes. 3) Se realizará un reporte para cada clínica junto con los nombres de cada medicamento, el número de prescripciones, las dosis prescritas y el costo total. 4) El acceso a los informes de costos se restringirá a usuarios autorizados en la lista de control de acceso administrativo.
>
> Lo que enseña el ejemplo: **un requerimiento de usuario se expande en varios del sistema**, y en la expansión aparecen horas concretas, formatos y restricciones de acceso.

3.4 Requerimientos funcionales y no funcionales

DIAPOSITIVAS · semana 4, láminas 3 a 5

La lámina 3 da la distinción en una línea cada uno: los **funcionales** son las funciones y acciones que debe realizar el sistema; los **no funcionales** son las **restricciones y/o limitaciones** al sistema.

Tipo	Desarrollo en las láminas 4 y 5
Funcionales	Descripción de las funciones y comportamientos específicos que el sistema debe realizar. Ejemplos: búsqueda, registro de datos, generación de informes. **También pueden describir las acciones que el sistema NO debe realizar** — este detalle de la lámina 4 se olvida siempre.
No funcionales	Especifican las cualidades y características no funcionales que debe tener el sistema: rendimiento, seguridad, usabilidad. Usualmente **se aplican al sistema como un todo** en vez de a ciertas funcionalidades.

3.5 Funcionales frente a no funcionales, punto por punto

DIAPOSITIVAS · semana 4, láminas 14 y 15

Cuatro contrastes que la profesora da explícitamente. Es material de tabla comparativa, o sea material de examen.

Criterio	Funcionales	No funcionales
Quién los especifica	El **usuario**	El **personal técnico**: desarrolladores, arquitecto

Criterio	Funcionales	No funcionales
Qué definen	Lo que **debe hacer** el sistema	Lo que es **requerido por otros**, como el proveedor de servicios o el sistema. Ejemplo de la lámina: el uso de un registro (log) permite solucionar problemas que se presenten en producción
Métricas	—	Se pueden definir con métricas: tiempo de respuesta, tasa de ocurrencia de falla, número de transacciones por segundo
Horizonte	—	Se pueden enfocar en la **evolución del sistema en el tiempo**: mantenibilidad, documentación

3.6 Cuantificar los no funcionales

DIAPOSITIVAS · semana 4, láminas 16 y 17

La lámina 16 plantea un ejercicio comparando dos redacciones del mismo requerimiento:

- «El sistema debe ser fácil de usar para los usuarios, de tal forma que se cometan la menor cantidad de errores posibles.»
- «La tasa de errores cometidos en el sistema debe estar por debajo del 30 % como resultado de que el usuario interactúe con el sistema por un tiempo aproximado de 2 horas.»

Las dos preguntas que hace la lámina: ¿cuál detalla mejor el aspecto de los errores? ¿cuál podrá ser **verificado**? La respuesta es la segunda en ambos casos, y la conclusión de la lámina 17 es la regla: **en lo posible se deben definir los requerimientos no funcionales de forma cuantitativa**.

La tabla de propiedades y métricas de la lámina 17 — es literal, conviene memorizarla:

Propiedad	Métrica
Rapidez	Transacciones por segundo procesadas
Tamaño	Número de chips
Facilidad de uso	Tiempo de capacitación
Fiabilidad	Tasa de ocurrencia de falla · probabilidad de indisponibilidad
Robustez	Tiempo de reinicio después de un fallo · porcentaje de errores

3.7 Los tres tipos de requerimientos no funcionales

DIAPOSITIVAS · semana 4, láminas 18 a 20

Taxonomía de Sommerville, novena edición. La lámina 18 es el gráfico del árbol; las láminas 19 y 20 desarrollan las tres ramas.

Tipo	Definición	Ejemplos de la lámina
Del producto	Requerimientos que **definen o limitan el comportamiento del software**.	Rapidez del sistema, uso de memoria, seguridad, usabilidad
De la organización	Provienen de las **políticas y procedimientos** tanto de la organización del cliente como de la del desarrollador.	El uso del sistema, el lenguaje de programación a usar
Externos	**Ajenos al sistema y a su desarrollo.** Se subdividen en tres: **regulatorios** (lo que el sistema debe cumplir según un ente regulador), **legislativos** (garantizan la operación de acuerdo a la ley) y **éticos** (garantizan que el sistema sea aceptable para los usuarios y el público en general).	Normativa de un ente regulador, cumplimiento legal, aceptabilidad pública

> **Truco para clasificar rápido en el examen**
> Pregúntate **de dónde viene la restricción**. Si viene de cómo debe comportarse el software → producto. Si viene de una política de una empresa → organización. Si viene de fuera de las dos empresas (ley, regulador, sociedad) → externo. Los tres subtipos de externo — regulatorio, legislativo, ético — son los que más se olvidan.

3.8 El proceso de requerimientos

DIAPOSITIVAS · semana 4, láminas 6 a 13

Cinco pasos, cada uno con lámina propia. El orden es parte de la respuesta.

#	Paso	En qué consiste
1	**Captura**	Recopilación **sistemática** de información y datos relevantes para comprender las necesidades y expectativas de las partes interesadas, **especialmente los usuarios finales**, con el fin de definir los requerimientos del software
2	**Análisis**	Extracción de las características y funcionalidades del sistema · definición de interfaces y restricciones · **traducción de los requisitos del usuario a requisitos de software**

#	Paso	En qué consiste
3	**Especificación**	Resultado del análisis **de manera tangible**: un documento que contiene los requisitos, el diseño y las funcionalidades que implementa un sistema. Es el documento a seguir por parte de los desarrolladores
4	**Validación y verificación**	Detección de fallas en los requerimientos, es decir, que sean **claros, consistentes y completos**. Validar durante el análisis es importante porque una vez implementado el sistema resulta costoso hacer cambios
5	**Gestión**	Seguimiento a los requerimientos **utilizando estados**. Los requerimientos pasan por diversos estados a lo largo del desarrollo, desde que se definen hasta que se desarrollan y se prueban

Cuatro técnicas de captura (lámina 8): entrevistas, encuestas, observación, talleres.

> **Verificación frente a validación — lámina 12, definición literal**
>
> **Verificación:** comprobar el cumplimiento del software **con respecto a los requerimientos especificados**.
>
> **Validación:** garantizar que el software **cumple con las expectativas del usuario o del mercado**.
>
> El objetivo final de ambos procesos, dice la lámina, es **establecer confianza de que el sistema de software es «adecuado»**.
>
> La forma corta de recordarlo: verificación pregunta «¿lo construimos bien?» (contra el documento); validación pregunta «¿construimos lo correcto?» (contra la expectativa real). ★ Las dos definiciones son de la lámina; la formulación en preguntas es mía.

3.9 Los cuatro ejemplos clasificados de la lámina 21

DIAPOSITIVAS · semana 4, lámina 21

Esta lámina es prácticamente un banco de preguntas ya resuelto. La clasificación es la de la profesora.

Requerimiento	Clasificación oficial
«El sistema debe almacenar la información de los estudiantes, en particular: la matrícula, el nombre, el apellido y el estado.»	**Funcional**
«El sistema debe tener un tiempo de respuesta menor a 5 segundos al realizar una consulta.»	**No funcional del producto**
«El sistema debe generar un reporte semestral de los alumnos que han reprobado materias.»	**Funcional**
«El sistema debe tardar máximo 15 minutos ante una situación de fallo, en el 90 % de los casos.»	**No funcional del producto**

El patrón: si el enunciado describe **algo que el sistema hace o guarda** → funcional. Si describe **con qué calidad o dentro de qué límite** lo hace, normalmente con un número → no funcional del producto.

Parte 4 · Escenarios y modelado de sistemas

Semana 5 (26.05.2026), 11 láminas. Es el mazo más corto del parcial y funciona como bisagra: cierra requerimientos y abre UML.

4.1 Escenarios

DIAPOSITIVAS · semana 5, láminas 3 a 5

Definición: descripciones donde se detallan las **interacciones del usuario con el sistema** para comprender mejor los requerimientos del sistema.

- Se pueden crear **distintos escenarios con varios niveles de detalle** sobre el sistema.
- **No existe un formato definido** para los escenarios. Este punto es preguntable como verdadero/falso.
- Aun sin formato fijo, siempre se deben considerar: las **acciones del sistema y del usuario** acorde al escenario planteado, y **los casos cuando ocurre una situación adversa**.

Las láminas 4 y 5 son dos ejemplos gráficos tomados de Hadad et al., *Explicitar Requisitos del Software usando Escenarios*.

4.2 Modelado de sistemas

DIAPOSITIVAS · semana 5, lámina 6

Definición: proceso para crear **modelos abstractos** de un sistema, los cuales ofrecen una **perspectiva diferente** del sistema.

- **Cuándo:** los modelos se pueden crear en la fase de ingeniería de requerimientos, durante la fase de diseño, y **una vez terminada la fase de implementación**. Los tres momentos son de la lámina.
- **Sobre qué:** se utilizan tanto en **sistemas existentes** como en **sistemas por diseñar**.

4.3 El lenguaje UML

DIAPOSITIVAS · semana 5, lámina 7

Unified **M**odeling **L**anguage, o Lenguaje Unificado de Modelado. Lenguaje utilizado para representar **aspectos funcionales** de un sistema: funciones, procesos, clases.

- Compuesto por **13 tipos de diagrama** según la lámina.
- **Versión actual: UML 2.5.1 (2017)**, referenciando omg.org/spec/UML.

> **La lámina 7 se contradice consigo misma en el número de diagramas**
> **Qué dice la lámina:** «un lenguaje compuesto por **13 tipos de diagrama**» y, dos líneas más abajo, «versión actual: **UML 2.5.1 (2017)**».
> **Dónde está el choque:** los 13 tipos corresponden a las versiones UML 2.0 a 2.2. **Desde UML 2.5, la especificación define 14 diagramas canónicos**: siete estructurales (clases, objetos, paquetes, estructura compuesta, componentes, despliegue, perfiles) y siete de comportamiento (casos de uso, actividad, máquina de estados, y las cuatro de interacción: secuencia, comunicación, tiempos y visión general de interacción). O la versión es 2.5.1 y son 14, o son 13 y la versión es anterior.
> **Qué responder:** si la pregunta es cerrada, responde **13**, que es el número de la lámina, y **2.5.1 (2017)** como versión, porque las dos cifras están ahí y es lo que la profesora espera. Si la pregunta es abierta o pide justificar, di 13 según el material del curso y añade que la especificación 2.5.1 lista 14. No te juegues una respuesta cerrada por esto.

4.4 Diagramas estructurales y de comportamiento

DIAPOSITIVAS · semana 5, láminas 8 y 9

La lámina 8 es el gráfico con el árbol completo de tipos de diagrama. La lámina 9 da la definición de las dos ramas, que es lo citable:

Rama	Definición de la lámina
Estructurales	Ilustran **los componentes de un sistema y la relación que existe entre ellos**. Se enfocan en los aspectos **estáticos** del sistema.
De comportamiento	Visualizan **lo que sucede dentro** de un sistema, es decir, cómo interactúan los componentes entre sí, así como con otros sistemas y/o usuarios.

Clasificación de los diagramas que sí ves en el curso: **estructurales** son clases, objetos y componentes; **de comportamiento** son casos de uso, secuencia, estados y actividad. ★ La lámina 9 da el criterio; el reparto concreto de estos siete lo hago yo aplicándolo.

4.5 Los cinco diagramas «esenciales»

DIAPOSITIVAS · semana 5, lámina 10

La lámina 10 lista los tipos de diagrama **esenciales para representar un sistema**:

- Diagramas de **actividad**
- Diagramas de **caso de uso**
- Diagramas de **secuencias**
- Diagramas de **clase**
- Diagramas de **estado**

> **La lista de «esenciales» no coincide con lo que el curso realmente enseña**
> **Qué dice la lámina 10 de la semana 5:** los cinco esenciales son actividad, caso de uso, secuencia, clase y estado.
> **Qué desarrollan después las semanas 6 y 7:** clases, casos de uso, secuencia, estados, **componentes** y **objetos**. Es decir: el **diagrama de actividad se declara esencial pero no se explica en ninguna lámina del parcial**, mientras que componentes y objetos se explican con detalle sin estar en la lista de esenciales.
> **Qué responder:** si la pregunta dice literalmente «diagramas esenciales», responde los cinco de la lámina 10 **incluyendo actividad**. Si la pregunta pide describir un diagrama, prepárate para los seis que sí tienen desarrollo (Parte 5). Si te toca describir actividad, no hay contenido oficial al que ceñirse.

Parte 5 · Los diagramas UML, uno por uno

Esta parte funde la semana 6 (02.06.2026, 33 láminas) con las láminas 3 a 8 de la semana 7. Los reordeno por familia: primero la estructura estática (clases y objetos), después el comportamiento (casos de uso, secuencia, estados) y al final componentes.

5.1 Diagrama de clases

DIAPOSITIVAS · semana 6, láminas 3 a 9

Definición: diagrama que se enfoca en las clases para **visualizar la estructura** de un sistema. Visualiza los **atributos y métodos** de las clases y **las relaciones entre ellas**. La lámina lo llama herramienta importante en la **programación orientada a objetos (POO)**.

Las láminas 4 a 9 son casi todas gráficas y cubren: elementos de un diagrama de clase (4), notación de los elementos (5, de icarus.cs.weber.edu), el **constructor** en un diagrama UML (6), **composición versus agregación** (7, de loekvandenouweland.com), **métodos con valores por defecto** (8, de ece.uwaterloo.ca) y un diagrama de clases completo

(9).

> **Qué hay que saber de las láminas gráficas de clases**
> Cuatro cosas concretas, porque cada una tiene su propia lámina y eso indica que la profesora las considera examinables por separado: **1)** la caja de clase con sus tres compartimentos — nombre, atributos, métodos; **2)** cómo se representa un **constructor**; **3)** la diferencia entre **composición y agregación**, que la semana 7 sí define con texto (ver 5.2); **4)** cómo se anotan los **métodos con valores por defecto** en la firma.

Ampliación mía sobre la notación — no está en texto en ninguna lámina, aunque sí en las imágenes: la visibilidad se marca con **+** público, – privado, # protegido y **~** de paquete; los miembros estáticos van subrayados; la firma completa de un método es *visibilidad nombre(parámetros): tipoRetorno*. Si la pregunta viene de la imagen de la lámina 5, esto es lo que se está mirando.

5.2 Asociación, agregación y composición

DIAPOSITIVAS · semana 7, láminas 6 a 8 · semana 6, lámina 7

La semana 6 muestra la comparación en una imagen (lámina 7) y la semana 7 la define con texto en el contexto del diagrama de objetos (láminas 6 a 8). Las definiciones textuales son estas:

Relación	Definición de la lámina	Cómo distinguirla
Asociación	Representación de la relación entre dos objetos, en particular cuando un objeto **referencia a miembro(s)** de otro objeto. Puede ser **unidireccional o bidireccional**.	Simplemente se conocen
Agregación	Representación de la relación **«has a»**. Ocurre cuando el ciclo de vida de un objeto **no depende en gran parte** del ciclo de vida del objeto contenedor.	Rombo vacío. El contenido sobrevive al contenedor
Composición	Representación de una relación en donde **un objeto no puede existir sin el otro** objeto.	Rombo relleno. Si muere el contenedor, muere el contenido

★ Las definiciones y el «has a» son de lámina. La columna de la derecha — los rombos y la regla de supervivencia — es mi resumen de la imagen de la lámina 7 de la semana 6, que no trae texto.

> **El criterio en una frase**
> La diferencia entre agregación y composición es **el ciclo de vida**, no la fuerza del vínculo. Universidad y estudiantes: si la universidad cierra, los estudiantes siguen existiendo → agregación. Factura y líneas de factura: si borras la factura, las líneas no significan nada → composición. Esta relación se vuelve a ver en el Parcial 2 dentro de acoplamiento (ver Parte 7).

5.3 Diagrama de objetos

DIAPOSITIVAS · semana 7, láminas 3 a 5

Definición: representación de **una instancia en particular de un diagrama de clases** en un **periodo de tiempo determinado**.

- Se enfoca en los **atributos** de un grupo de objetos y sus relaciones entre sí.
- Es útil para obtener una **vista general** del sistema a desarrollar.

La lámina 4 es la notación (GeeksForGeeks) y la lámina 5 un ejemplo (Sybase). La forma corta de recordar la relación con el diagrama de clases: **clases es el molde, objetos es la foto**.

5.4 Diagrama de casos de uso

DIAPOSITIVAS · semana 6, láminas 10 a 18

Definición: diagrama utilizado para representar las **interacciones entre un sistema de software y sus actores**, que pueden ser usuarios, otros sistemas o componentes externos.

- **Útil para realizar la captura de requerimientos** — esto conecta directamente con el paso 1 del proceso de requerimientos (sección 3.8).
- Su objetivo principal es mostrar, **desde la perspectiva del usuario**, cómo los actores interactúan con el sistema y qué funcionalidades o servicios ofrece el sistema en respuesta a esas interacciones.

Los cuatro elementos (lámina 11). Nota cuáles son opcionales, es preguntable:

Elemento	Definición	¿Opcional?
Actor	Entidad que desempeña un papel en el sistema	No
Caso de uso	Función o acción dentro del sistema	No
Sistema	Elemento utilizado para definir el **alcance** del caso de uso	**Sí**
Paquete	Elemento utilizado para **agrupar** casos de uso	**Sí**

Las cuatro relaciones (láminas 13 a 16). Cada una tiene lámina propia, así que las cuatro son material de recall:

Relación	Definición de la lámina
Asociación	Representación de la **interacción entre el actor y el caso de uso**.
Extensión	Se utiliza para la representación de **comportamientos opcionales** que se pueden extender de un caso de uso base.
Inclusión	Consiste en un caso de uso (**base**) que **contiene la funcionalidad** de otro caso de uso (**inclusión**).
Generalización	Implica modelar la **herencia** entre casos de uso. Un caso de uso **hijo hereda los comportamientos** del caso de uso padre.

> **Ampliación mía: dirección de las flechas en extensión e inclusión**
> Las láminas definen las dos relaciones pero no dicen hacia dónde apunta la flecha, y es lo que más se confunde. En UML las dos son dependencias con estereotipo y la flecha discontinua apunta **siempre hacia el caso de uso del que se depende**:
> **«include»:** del caso base → hacia el caso incluido. El base necesita al incluido, **siempre** se ejecuta.
> **«extend»:** del caso de extensión → hacia el caso base. El comportamiento opcional apunta al que amplía, y se ejecuta **solo a veces**.
> La regla de decisión: **obligatorio y reutilizado → include; opcional o condicional → extend**. ★ La definición de cada relación es oficial; la dirección de las flechas y esta regla son míos.

Descripción tabular de un caso de uso (lámina 18). La profesora da una plantilla con siete campos y un ejemplo completo. Los campos son la parte examinable:

Campo	Contenido del ejemplo de la lámina
Caso de uso	Agregar registro de usuario
Actores	Cajero
Descripción	El cajero ingresa datos del usuario para la emisión de la factura al terminar la compra
Datos	Nombre, apellido, cédula, RUC, dirección
Estímulo	El cajero ingresa datos en el formulario
Respuesta	El sistema confirma el ingreso del usuario
Comentarios	El cajero debe tener permiso para ingresar al sistema

El par **estímulo/respuesta** es lo distintivo de esta plantilla: separa lo que hace el actor de lo que devuelve el sistema.

5.5 Diagrama de secuencia

DIAPOSITIVAS · semana 6, láminas 19 a 21

Definición: diagrama utilizado para representar las interacciones en un sistema **con respecto al tiempo**. Está enfocado en el **flujo de mensajes o interacciones entre objetos**, lo que facilita comprender **la colaboración de las diferentes partes de un sistema** para realizar una funcionalidad específica o un proceso.

La lámina 20 es el gráfico de elementos (Web y Empresas) y la lámina 21 el ejemplo: **retiro de dinero en un cajero automático**. Si te piden un ejemplo de diagrama de secuencia, ese es el del curso.

Ampliación mía sobre los elementos de la lámina 20: actores y objetos arriba, **línea de vida** vertical descendente, **barra de activación** cuando el objeto está trabajando, y mensajes horizontales — flecha con punta rellena para el mensaje sincrónico, flecha discontinua para el retorno. El **eje vertical es el tiempo**, y eso es lo único que la lámina sí dice con palabras.

5.6 Diagrama de estados

DIAPOSITIVAS · semana 6, láminas 22 a 26

También conocido como diagrama de transición de estados o diagrama de máquina de estados — los tres nombres son de la lámina. **Definición:** representación de los **estados de un objeto**, las **acciones** que se realizan dependiendo del estado, y las **transiciones** entre los estados del objeto.

Los cinco componentes (lámina 23): estados, transición, eventos, nodos, acciones.

La tabla de símbolos principales (lámina 24, adaptada de Chang Wan). Es literal y muy preguntable, sobre todo los cuatro tipos de nodo:

Símbolo	Descripción de la lámina
Estado	Descripción de un tiempo específico en el cual se encuentra un objeto durante su ciclo de vida. Algunas acciones se pueden realizar durante un estado
Transición	Indica transición de un estado hacia otro
Estado inicial	Punto inicial o primera actividad del flujo
Estado final	El proceso termina

Símbolo	Descripción de la lámina
Nodo terminar	Terminación de la línea de vida de la máquina de estados: **el objeto ya no existe**
Nodo decisión	Indica varias alternativas de transición que pueden ocurrir
Nodo paralelización	División de la transición en múltiples transiciones paralelas
Nodo sincronización	Unión de múltiples transiciones paralelas en una transición

> **La distinción fina: estado final frente a nodo terminar**
> **Estado final** = el proceso termina. **Nodo terminar** = el objeto deja de existir. No es lo mismo y la lámina se toma el trabajo de separarlos, lo que suele indicar que va a aparecer en una pregunta. Igual con **paralelización** (una transición se abre en varias) frente a **sincronización** (varias se juntan en una).

Los dos ejemplos: la lámina 25 es de Pressman, *Software Engineering: A practitioner approach*. La lámina 26 es el ciclo de una orden, adaptado de GeeksForGeeks, y sí tiene texto extraíble: **orden sin procesar** → **orden recibida** → **[aceptar]/[rechazar]** → **orden aceptada / orden rechazada** → **[ítems disponibles]/[ciertos ítems no están disponibles]** → **orden completada / orden pendiente**. Los corchetes son las **guardas** de las transiciones.

5.7 Diagrama de componentes

DIAPOSITIVAS · semana 6, láminas 27 a 29

Definición: representación de la **organización de los componentes** de un sistema, así como de las **relaciones entre ellos**.

- **Qué puede ser un componente**, según la lámina 27: base de datos, interfaces de usuario, **hardware**, unidad de negocio (proveedor, envío). Fíjate en que la lista mezcla niveles: hay software, hardware y negocio.
- **Para qué sirve:** es útil para **explicar a los interesados** el funcionamiento del sistema que se construye.

La notación completa (lámina 28, adaptada de ionos). Ocho elementos:

Elemento	Descripción de la lámina
Componente	Símbolo para **módulos** en un sistema. Se anota «component» Nombre
Paquete	Combinación de múltiples elementos — clases, componentes, interfaces — en un solo grupo
Artefacto	**Unidades físicas de información** (código fuente, scripts, documentos) creados o requeridos durante el desarrollo o en tiempo de ejecución. Se anota «artifact» Nombre
Interfaz provista	Símbolo para una o más interfaces que **proveen** funciones, servicios o datos **hacia fuera**
Interfaz requerida	Símbolo para una interfaz que **recibe** funciones, servicios o datos **desde fuera**
Puerto	Punto separado de interacción entre un componente y **su entorno**
Relación	Una línea que actúa como conector y muestra la relación entre componentes
Dependencia	Indica dependencia entre dos partes de un sistema

El par **interfaz provista / interfaz requerida** es el que se pregunta: provista es la bolita («lollipop»), requerida es el zócalo que la recibe. ★ Los nombres y las definiciones son de lámina; los apodos gráficos son míos.

5.8 Tabla de decisión: qué diagrama usar

MATERIAL PROPIO · ampliación mía

No hay lámina que junte los seis. La armo yo para el repaso, con el vocabulario oficial de cada definición.

Si la pregunta habla de…	El diagrama es	Rama
Estructura, atributos y métodos, relaciones entre clases	**Clases**	Estructural
Una instancia concreta en un momento dado	**Objetos**	Estructural
Organización de módulos, interfaces provistas y requeridas	**Componentes**	Estructural
Actores y funcionalidades desde la perspectiva del usuario	**Casos de uso**	Comportamiento
Mensajes entre objetos ordenados en el tiempo	**Secuencia**	Comportamiento
Estados de un objeto y transiciones entre ellos	**Estados**	Comportamiento

Parte 6 · Del modelo de requerimientos al modelo de diseño

Semana 7 (09.06.2026), láminas 9 a 26. Es el bloque bisagra de todo el curso: cierra el Parcial 1 y abre el vocabulario que la Parte 2 desarrolla entero.

6.1 Qué es diseño

DIAPOSITIVAS · semana 7, láminas 9 a 11

> **La cita de la lámina 9**
>
> «¿Qué es diseño? Es donde estás con un pie en dos mundos — el mundo de la tecnología y el mundo de las personas y los propósitos humanos — y tu objetivo es unir estos dos mundos.»
>
> **Mitch Kapor**, *Manifiesto del diseño de software*, **1991**.

Definición operativa (lámina 10): representación o modelo de un software, el cual provee detalles sobre **la arquitectura de software, las interfaces y los componentes** necesarios para implementar un sistema. El diseño permite construir un modelo que **puede ser evaluado en aspectos de calidad y mejorado antes de que se realice la codificación y las pruebas** — esa es la razón económica de diseñar.

Qué involucra el diseño de software (lámina 11), tres puntos: definición de arquitectura · modelado de interfaces y componentes · diseño de los componentes de software.

6.2 Un buen software es… DIAPOSITIVAS · semana 7, lámina 12

Cuidado: esta es una **cuarta** lista de calidad, distinta de las cuatro características de la semana 1. Son cuatro puntos también, pero redactados de otra forma:

- Aquel que **no tiene errores que impidan su funcionamiento**
- Aquel que **cumple el propósito** por el cual fue creado
- Aquel que ofrece una **experiencia agradable de uso**
- Aquel que es **mantenible y eficiente**

La conexión es evidente: el punto 3 es *usable*, el punto 4 son *mantenible* y *eficiente*, y el punto 1 se acerca a *confiable*. La semana 7 reformula la lista de la semana 1 en lenguaje de diseño. Si la pregunta cita la semana 7, responde con estos cuatro; si pregunta «características esenciales de un buen software», responde con los cuatro adjetivos de la semana 1.

6.3 La conversión del modelo de requerimientos al modelo de diseño DIAPOSITIVAS · semana 7, lámina 13

Lámina puramente gráfica, tomada de Pressman, *Ingeniería del Software: Un enfoque práctico*. Es el esquema que muestra cómo cada parte del modelo de análisis alimenta una parte del modelo de diseño. Las cuatro láminas siguientes son la letra de ese gráfico, así que el contenido examinable está en 6.4.

6.4 Los cuatro elementos del modelo de diseño DIAPOSITIVAS · semana 7, láminas 14 a 17

Cada elemento tiene lámina propia. El orden en que la profesora los presenta es este, y conviene respetarlo porque refleja de qué se alimenta cada uno:

Elemento	Definición de la lámina	De qué se alimenta
Diseño de datos o clases	Transformación de los modelos de clases en **clases de diseño** y las estructuras de datos necesarias para implementar el software. Puede ocurrir **en conjunto con el diseño de la arquitectura**.	Modelos de clases
Diseño de la arquitectura	Definición de la **relación entre los elementos principales estructurales** del software, **los patrones y los estilos** de arquitectura. Se consideran además las **restricciones** que afectan la implementación.	El modelo de análisis y las restricciones
Diseño de la interfaz	Descripción de la **comunicación del software con otros sistemas y con los usuarios**. Una interfaz involucra un **flujo de información** — datos y/o control — y un tipo específico de comportamiento.	Elementos basados en **escenarios** de uso y comportamiento
Diseño en el nivel de componentes	Transformación de los elementos estructurales de la arquitectura en una **descripción procedimental** de los componentes de software.	Modelos basados en **clases y comportamiento**

Fíjate en la cadena: **arquitectura** → **componentes**. Primero se definen los elementos estructurales y sus relaciones, después cada elemento se convierte en una descripción procedimental. Esta cadena es exactamente la que la Parte 2 recorre completa.

6.5 Qué es un componente DIAPOSITIVAS · semana 7, láminas 18 y 19

> **Definición oficial de la OMG — lámina 18**
>
> «Una parte **modular, desplegable y sustituible** de un sistema, que **incluye la implantación** y **expone un conjunto de interfaces**.» Especificación OMG del Lenguaje de Modelado Unificado.
>
> Los cinco rasgos — modular, desplegable, sustituible, incluye implantación, expone interfaces — son material de recall.

Y la parte fina: el componente cambia de significado según el contexto (lámina 19). Tres columnas, y esta lámina es de las más preguntables del mazo porque el objetivo de aprendizaje de la lámina 2 dice literalmente «distinguir el concepto de componentes según el contexto»:

Contexto	Qué es un componente ahí
Ingeniería de software orientada a objetos	Un **conjunto de clases que colaboran**. También pueden crearse componentes con **una sola clase**.
Ingeniería de software tradicional	Un **elemento funcional del programa** (módulo), que contiene la **lógica**, las **estructuras de datos** para implementar la lógica, y una **interfaz** para invocar al componente y pasar los datos.
Reuso de componentes existentes	Un componente se diseña para que sea **reutilizable**, por lo que se dispone de la **descripción de la interfaz**, las **funciones que realiza** y de la **comunicación y colaboración requerida**.

6.6 El proceso de diseño

DIAPOSITIVAS · semana 7, lámina 20

El diseño de software es un proceso iterativo a través del cual los requerimientos se traducen en un **«blueprint» o «plano»** para construir el software, el cual muestra una **vista general** del software. Al realizar varias iteraciones, esto conlleva a representaciones en **niveles de abstracción mucho más bajos**.

Dos palabras que hay que devolver textualmente si la pregunta es abierta: **iterativo** y **blueprint/plano**. La idea de bajar de nivel de abstracción con cada iteración es la que la semana 8 llama **refinamiento**.

6.7 Características para evaluar un buen diseño

DIAPOSITIVAS · semana 7, lámina 21

Tres características, distintas de las ocho guías de la lámina siguiente:

- El diseño **contempla todos los requerimientos** especificados.
- El diseño debe ser una **guía legible y comprensible** para quienes escriben el código, quienes prueban el sistema y quienes le dan soporte.
- El diseño debe ofrecer una **imagen completa** del software, abordando el dominio de los **datos, el funcional y el de comportamiento** desde un enfoque de implementación.

6.8 Las ocho guías de calidad de diseño

DIAPOSITIVAS · semana 7, láminas 22 a 24

Ocho puntos numerados repartidos en tres láminas. Es la lista más larga del parcial y la que más vocabulario del Parcial 2 contiene.

#	Guía
1	Debe mostrar una **arquitectura** que utiliza **estilos o patrones** de arquitectura, tiene componentes que muestran un buen diseño, y puede ser implementado de una **forma evolutiva**
2	Debe ser **modular**, es decir, particionado lógicamente en elementos o subsistemas
3	Debe contener **distintas representaciones** de datos, arquitectura, interfaces y componentes
4	Debe conducir a **estructuras de datos apropiadas** para las clases
5	Debe conducir a componentes que exhiben características de **independencia funcional**
6	Debe conducir a **interfaces que reducen la complejidad** de conexiones entre componentes y con su entorno
7	Debe ser derivado de utilizar un **método repetible**
8	Debe ser representado utilizando una **notación que comunica de forma efectiva** su significado

> **Las cuatro guías que son vocabulario del Parcial 2**
>
> La guía **1** nombra **estilos y patrones de arquitectura** — es toda la Parte 3 y 4 del resumen maestro. La guía **2** es **modularidad**. La guía **5** es **independencia funcional**, que en la semana 8 se descompone en **alta cohesión y bajo acoplamiento**. La guía **6** es **reducir la complejidad de las interfaces**, que es el germen de ISP y DIP en SOLID.
>
> Dicho de otra forma: la lámina 22 de la semana 7 es el índice del Parcial 2.

6.9 Principio KISS

DIAPOSITIVAS · semana 7, lámina 25

Keep **I**t **S**imple, **S**tupid. Los dos puntos de la lámina: **soluciones simples** y **evitar soluciones complejas o innecesarias**.

Es el único principio de diseño con nombre propio del Parcial 1. Con el KISS cierra el mazo de la semana 7 y con él cierra el parcial.

Parte 7 · Puente hacia el Parcial 2

7.1 Las cinco conexiones que el examen acumulativo puede cruzar

MATERIAL PROPIO · ampliación mía

Todo lo de esta parte es material propio

Ninguna lámina cruza el Parcial 1 con el Parcial 2. Estas conexiones las construyo yo porque el examen del 11.08 es acumulativo y las preguntas que valen doble son las que tocan los dos bloques. **No las cites como contenido del curso**, pero úsalas para responder preguntas abiertas.

Concepto del Parcial 1	Con qué se conecta en el Parcial 2	Qué preguntar/responder
Requerimientos no funcionales (3.4 a 3.7): rendimiento, seguridad, usabilidad, mantenibilidad	Los **cinco atributos de calidad** de la semana 8 y los **criterios de selección de arquitectura** de la semana 12, entre ellos **mantenibilidad**	Es el mismo concepto en dos momentos: lo que en requerimientos es una restricción, en arquitectura es un criterio de decisión. La arquitectura se elige para satisfacer los NF
Guía de calidad 5: «componentes que exhiben independencia funcional» (6.8)	**Cohesión y acoplamiento** (semana 8) y **SRP** y **DIP** de SOLID (semana 9)	Independencia funcional = alta cohesión + bajo acoplamiento. La semana 7 nombra la meta; la semana 8 da la métrica
Agregación y composición (5.2)	**Acoplamiento** y el principio GoF de **composición sobre herencia**	Composición implica ciclo de vida atado, o sea más acoplamiento que agregación. Y ojo: «composición» en GoF significa otra cosa — delegar en tiempo de ejecución en vez de heredar
Diagrama de componentes (5.7) y **definición de componente** (6.5)	El **componente y sus relaciones** en la definición de arquitectura (semana 10) y los **patrones arquitectónicos** (semanas 11 y 12)	El diagrama de componentes con interfaces provistas y requeridas es la notación con la que se dibuja un monolito modular o unos microservicios
Modelos de proceso (2.6 a 2.11): cascada, espiral, incremental, ágil	**CI/CD y DevOps** (semana 12): push → build → tests → empaquetado Docker → despliegue → monitoreo	El pipeline de CI/CD es la automatización de las fases 3, 4 y 5 del SDLC. Ágil e incremental son sus prerrequisitos culturales: no hay entrega continua con cascada

7.2 El mapa completo de las listas de calidad del curso

MIXTO · base de lámina, tabla mía

Es el punto donde más fácil se pierde un punto por confusión. Cuatro listas, cuatro niveles distintos:

Lista	Cuántos	Contenido	Origen
Características de un buen software	4	Mantenible · confiable · eficiente · usable	Semana 1, láminas 23-27
Un buen software es…	4	Sin errores que impidan funcionar · cumple su propósito · experiencia agradable · mantenible y eficiente	Semana 7, lámina 12
Características para evaluar un buen diseño	3	Contempla los requerimientos · guía legible · imagen completa (datos, funcional, comportamiento)	Semana 7, lámina 21
Guías de calidad de diseño	8	Ver sección 6.8	Semana 7, láminas 22-24
Atributos de calidad (fuera del Parcial 1)	5	Funcionalidad · usabilidad · fiabilidad · desempeño · soportabilidad	Semana 8 — Parcial 2

Apéndice A · Inventario lámina por lámina

Este es el control de cobertura. Cada lámina de los siete mazos tiene asignada una sección. **181 láminas en total.** Las marcadas «adm.» son administrativas y las marcadas «gráf.» son láminas cuyo contenido es esencialmente una imagen — título y fuente, sin texto de contenido — cubiertas por la sección indicada.

Semana 1 · 28.04.2026 · 32 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1	Portada	—
2-13	Políticas del curso, asistencia, puntualidad, calificaciones, dispositivos, herramientas, actividades y calendario de evaluaciones	1.6 (adm.)
14	Contenido del curso: cinco unidades	1.6
15	Objetivos de aprendizaje	Parte 1
16-17	Láminas divisorias «¿Qué es ingeniería de software?»	1.1 (gráf.)
18-20	Tres definiciones: Sommerville, ISO/IEC 29110, Boehm	1.1
21	Síntesis de la definición	1.1
22	Elementos involucrados en el desarrollo	1.2
23	Las cuatro características de un buen software	1.3

Láminas	Contenido	Sección
24-27	Mantenible, confiable, eficiente, usable (una lámina cada uno)	1.3
28	Ingeniería de software vs. ingeniería de sistemas	1.4
29	Ingeniería de software vs. ciencias de la computación	1.4
30	Tipos de productos: genéricos y personalizados	1.5
31	Material extra: OWASP Top Ten	1.6
32	Resumen de la clase	Parte 1

Semana 2 · 05.05.2026 · 44 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-2	Portada y objetivos de aprendizaje	Parte 2
3-4	La crisis del software (vídeo + los cuatro síntomas)	2.1
5-6	SWEBOK y sus áreas de conocimiento	2.2
7-9	SDLC: definición, importancia, ventajas de usar una metodología	2.3
10	Gráfico de las fases del SDLC	2.4 (gráf.)
11	Lista de las seis fases (titulada «Fases del DLC», errata)	2.4
12-15	Planificación y análisis · diseño · implementación · pruebas	2.4
16-17	Despliegue y entornos de desarrollo	2.4
18	Mantenimiento	2.4
19-20	Proceso de software y las cuatro actividades esenciales	2.5
21-22	Modelos de proceso: definición y lista de cinco	2.6
23-28	Modelo en cascada: definición, gráfico, cinco fases, consideraciones	2.7
29-31	Modelo en espiral: definición, riesgos, gráfico	2.8
32-35	Los cuatro sectores de la espiral	2.8
36-41	RUP: definición, gráfico y las cuatro fases	2.9
42-43	Desarrollo incremental y su gráfico	2.10
44	Resumen de la clase	Parte 2

Semana 3 · 12.05.2026 · 13 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-3	Portada, objetivos y lámina divisoria	Parte 3
4	Cita de Frederick P. Brooks, «No Silver Bullet», 1987	3.1
5	Definición de requerimiento	3.1
6-7	Importancia de los requerimientos (cinco razones)	3.2
8	Requerimientos del usuario	3.3
9	Requerimientos del sistema y especificación funcional	3.3
10-12	Ejemplo MHC-PMS: un requerimiento de usuario y cuatro del sistema	3.3
13	Resumen de la clase	Parte 3

Semana 4 · 19.05.2026 · 22 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-2	Portada y objetivos de aprendizaje	Parte 3
3	Tipos de requerimientos: funcionales y no funcionales	3.4
4-5	Requerimientos funcionales y no funcionales en detalle	3.4
6	El proceso de requerimientos: cinco pasos	3.8
7-8	Captura y técnicas para recopilar información	3.8
9-10	Análisis y especificación	3.8

Láminas	Contenido	Sección
11-12	Validación y verificación	3.8
13	Gestión de requerimientos	3.8
14-15	Funcionales vs. no funcionales: los cuatro contrastes	3.5
16	Ejercicio de las dos redacciones	3.6
17	Tabla propiedad/métrica	3.6
18	Gráfico del árbol de tipos de no funcionales (Sommerville)	3.7 (gráf.)
19-20	Del producto, de la organización, externos (regulatorios, legislativos, éticos)	3.7
21	Cuatro ejemplos clasificados	3.9
22	Resumen de la clase	Parte 3

Semana 5 · 26.05.2026 · 11 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-2	Portada y objetivos de aprendizaje	Parte 4
3	Escenarios: definición y qué considerar	4.1
4-5	Dos ejemplos de escenario (Hadad et al.)	4.1 (gráf.)
6	Modelado de sistemas	4.2
7	Lenguaje UML: 13 tipos, versión 2.5.1	4.3
8	Gráfico del árbol de tipos de diagramas UML	4.4 (gráf.)
9	Diagramas estructurales y de comportamiento	4.4
10	Los cinco diagramas esenciales	4.5
11	Resumen de la clase	Parte 4

Semana 6 · 02.06.2026 · 33 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-2	Portada y objetivos de aprendizaje	Parte 5
3	Diagrama de clases: definición	5.1
4-6	Elementos, notación y constructor	5.1 (gráf.)
7	Composición versus agregación	5.2 (gráf.)
8-9	Métodos con valores por defecto y ejemplo completo	5.1 (gráf.)
10-11	Casos de uso: definición y los cuatro elementos	5.4
12	Gráfico de elementos del diagrama de casos de uso	5.4 (gráf.)
13-16	Las cuatro relaciones: asociación, extensión, inclusión, generalización	5.4
17	Ejemplo de diagrama de casos de uso	5.4 (gráf.)
18	Descripción tabular de un caso de uso (siete campos)	5.4
19-21	Diagrama de secuencia: definición, elementos, ejemplo del cajero	5.5
22-23	Diagrama de estados: definición y cinco componentes	5.6
24	Tabla de símbolos principales (ocho símbolos)	5.6
25-26	Dos ejemplos: Pressman y el ciclo de una orden	5.6
27-29	Diagrama de componentes: definición, notación (ocho elementos), ejemplo	5.7
30-32	Enlaces de ejemplos, buenas prácticas y especificación UML	Parte 5
33	Resumen de la clase	Parte 5

Semana 7 · 09.06.2026 · 26 láminas

DIAPOSITIVAS

Láminas	Contenido	Sección
1-2	Portada y objetivos de aprendizaje	Partes 5 y 6
3	Diagrama de objetos: definición	5.3

Láminas	Contenido	Sección
4-5	Notación y ejemplo	5.3 (gráf.)
6-8	Asociación, agregación («has a») y composición	5.2
9	Cita de Mitch Kapor, «Manifiesto del diseño de software», 1991	6.1
10-11	Introducción al diseño y qué involucra	6.1
12	Un buen software es… (cuatro puntos)	6.2
13	Conversión del modelo de requerimientos al de diseño (Pressman)	6.3 (gráf.)
14-17	Los cuatro elementos del modelo de diseño	6.4
18	Definición de componente según la OMG	6.5
19	Definición de componente según el contexto (tres columnas)	6.5
20	El proceso de diseño: iterativo, blueprint	6.6
21	Características para evaluar un buen diseño (tres)	6.7
22-24	Las ocho guías de calidad de diseño	6.8
25	Principio KISS	6.9
26	Resumen de la clase	Parte 6

Apéndice B · Correcciones aplicadas

Cada fila corresponde a una caja ámbar del documento. Se sigue el mismo criterio del resumen maestro de la Parte 2: **la lámina siempre manda en la respuesta cerrada**, y la discrepancia se anota para las preguntas abiertas.

Punto	Qué se creía o qué dice la lámina	Qué aplica ahora	Autoridad
Procedencia del material del Parcial 1	Se asumía que había resúmenes de sesión míos ampliando estas siete semanas, como en la Parte 2	No existen. Los seis resúmenes del proyecto cubren solo de la semana 8 en adelante. Por eso este documento es casi íntegramente DIAPOSITIVAS y el material propio va concentrado y señalado	Revisión término por término de los seis resúmenes del proyecto
Versión de SWEBOK	«Versión actual: 3.0» (semana 2, lámina 5)	Se conserva **3.0** como respuesta de examen y se anota que la IEEE Computer Society publicó **v4.0 en 2024** — 18 áreas de conocimiento, tres nuevas — y la v4.0a en septiembre de 2025	IEEE Computer Society, SWEBOK v4.0
Número de diagramas UML	«13 tipos de diagrama» junto a «versión actual: UML 2.5.1 (2017)» (semana 5, lámina 7)	Se conserva **13** y **2.5.1** como respuesta de examen, señalando que son cifras incompatibles: desde UML 2.5 la especificación define **14** diagramas (7 estructurales + 7 de comportamiento)	Especificación OMG UML 2.5.1
Atribución de la definición de la lámina 19	La definición «application of a systematic, disciplined, quantifiable approach…» se atribuye a ISO/IEC 29110	Se conserva la atribución de la lámina. Se anota que la formulación es la definición clásica de IEEE 610.12-1990, hoy en el vocabulario ISO/IEC/IEEE 24765, que la 29110 reutiliza	ISO/IEC/IEEE 24765
Diagramas «esenciales»	La semana 5 lista cinco esenciales incluyendo **actividad**	Se separan las dos respuestas: los cinco de la lámina 10 para la pregunta literal, y los seis que sí se desarrollan (clases, objetos, casos de uso, secuencia, estados, componentes) para preguntas de descripción. El diagrama de actividad no se explica en ninguna lámina del parcial	Semana 5, lámina 10 frente a semanas 6 y 7
Modelos de proceso	La lámina 22 lista cinco: cascada, espiral, V, incremental, ágil	Se señala el desajuste: **RUP se desarrolla en seis láminas sin estar en la lista**, y el **modelo V y ágil se listan pero nunca se explican**. La sección 2.11 los cubre marcados como material propio	Semana 2, láminas 22 y 36-41
Tres listas de fases en la semana 2	Se mezclaban las seis fases del SDLC, las cuatro actividades esenciales y las cinco fases de cascada	Se separan explícitamente con una regla de desambiguación por el enunciado de la pregunta	Semana 2, láminas 11, 20 y 25-27
Listas de calidad del curso	Se confundían las características de buen software, las guías de diseño y los atributos de calidad	Tabla 7.2 con las cuatro listas del Parcial 1 más la del Parcial 2, distinguidas por el nivel al que aplican	Semanas 1 y 7; semana 8 para la quinta
Errata en la lámina 11 de la semana 2	La lámina se titula «Fases del DLC»	Se lee **SDLC**. Errata tipográfica, sin efecto sobre el contenido: la lista de seis fases es correcta	Semana 2, láminas 7 y 11

Punto	Qué se creía o qué dice la lámina	Qué aplica ahora	Autoridad
Dirección de las flechas «include» y «extend»	Las láminas 14 y 15 definen las relaciones pero no indican la dirección	Se añade marcado como material propio: «include» apunta del base al incluido; «extend» apunta de la extensión al base. Regla: obligatorio y reutilizado → include; opcional → extend	Especificación OMG UML 2.5.1
Notación de visibilidad en el diagrama de clases	Solo aparece dentro de la imagen de la lámina 5, sin texto	Se añade marcado como material propio: + público, – privado, # protegido, ~ paquete; estáticos subrayados	Semana 6, lámina 5 (imagen)
Elementos del diagrama de secuencia	La lámina 20 es un gráfico sin texto	Se añaden marcados como material propio: línea de vida, barra de activación, mensaje sincrónico y retorno. Lo único oficial en palabras es que el eje vertical es el tiempo	Semana 6, láminas 19 y 20

Apéndice C · Fuentes

Fuente	Qué respalda
Diapositivas del curso	Los siete mazos del periodo: 28.04.2026, 05.05.2026, 12.05.2026, 19.05.2026, 26.05.2026, 02.06.2026 y 09.06.2026. Autoridad máxima para todo lo marcado DIAPOSITIVAS
Ian Sommerville	*Ingeniería de software*, novena edición — definición de la disciplina, gráficos de cascada, RUP e incremental, taxonomía de requerimientos no funcionales, ejemplo MHC-PMS
Roger S. Pressman	*Ingeniería del Software: Un enfoque práctico* — conversión del modelo de requerimientos al de diseño, ejemplo de diagrama de estados
Frederick P. Brooks	*No Silver Bullet* (1987) — cita de apertura de la semana 3
Mitch Kapor	*Manifiesto del diseño de software* (1991) — cita de apertura del bloque de diseño
Barry W. Boehm	Definición de ingeniería de software; modelo en espiral (1988)
Norma ISO/IEC 29110 · ISO/IEC/IEEE 24765	Definición citada en la lámina 19 y su procedencia real
OMG · Unified Modeling Language 2.5.1	Definición de componente; número real de diagramas canónicos; semántica de «include» y «extend»
IEEE Computer Society · SWEBOK	Áreas de conocimiento; estado real de versiones (v3.0 frente a v4.0 de 2024 y v4.0a de 2025)
Hadad et al.	*Explicitar Requisitos del Software usando Escenarios* — ejemplos de escenario
Fuentes gráficas citadas por las láminas	icarus.cs.weber.edu, ece.uwaterloo.ca, loekvandenouweland.com, GeeksForGeeks, Web y Empresas, ionos, Javatpoint, SlideSalad, Chang Wan, Wikipedia, Sybase

Control de cobertura

Las **181 láminas** de los siete mazos están asignadas a una sección en el Apéndice A: **12** administrativas (semana 1, láminas 2 a 13), **21** de portada, objetivos de aprendizaje y resumen de clase (tres por mazo), y **148** de contenido, de las cuales unas 30 son esencialmente gráficas y quedan cubiertas por la sección temática que les corresponde. No queda ninguna lámina oficial sin destino.

Reparto por mazo: semana 1, 32 láminas · semana 2, 44 · semana 3, 13 · semana 4, 22 · semana 5, 11 · semana 6, 33 · semana 7, 26.