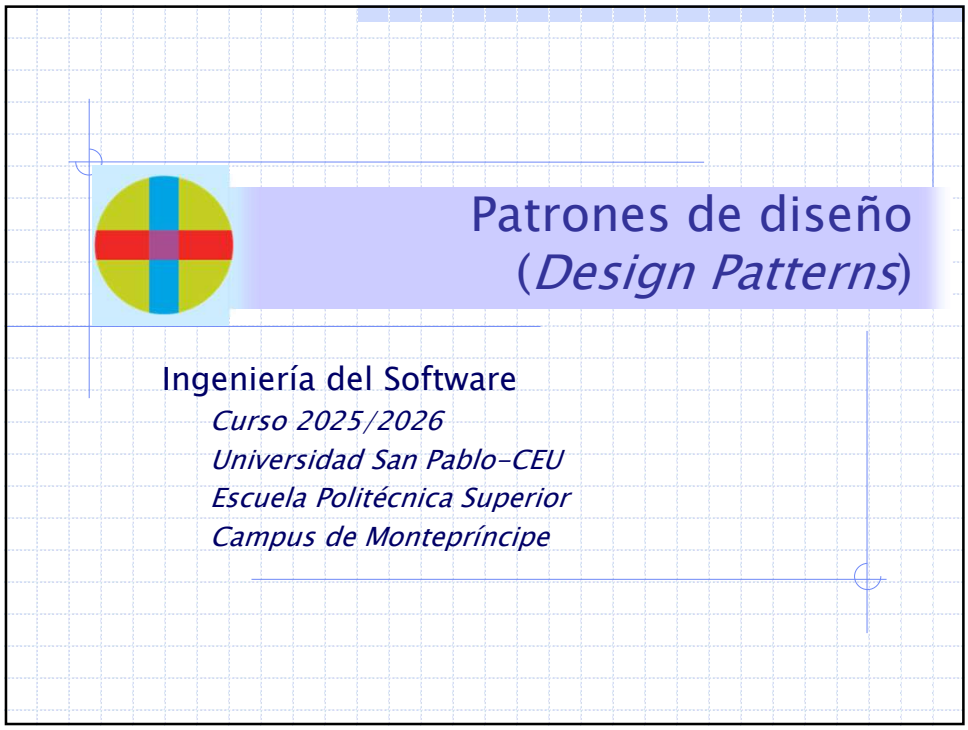
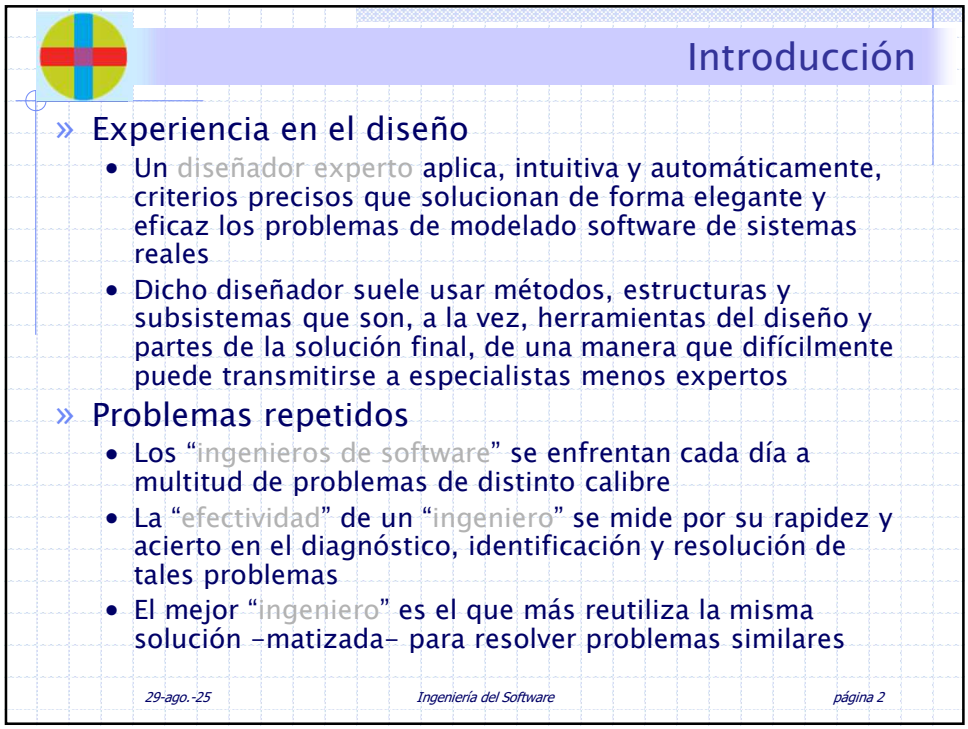
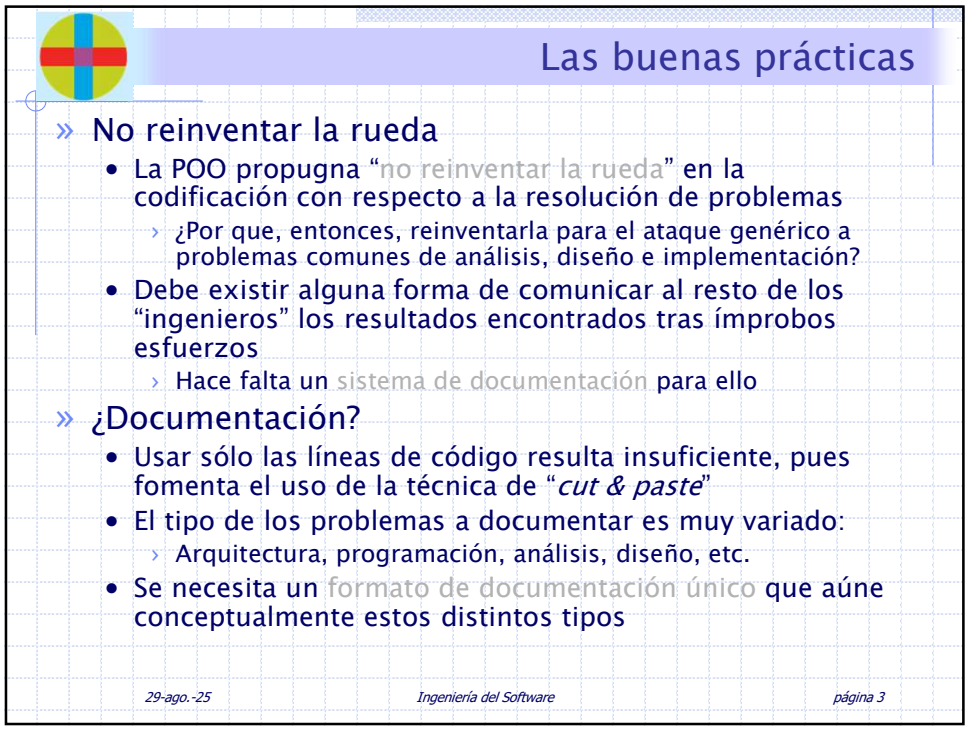

Patrones de diseño
(*Design Patterns*)
Ingeniería del Software
Curso 2025/2026
Universidad San Pablo-CEU
Escuela Politécnica Superior
Campus de Montepríncipe

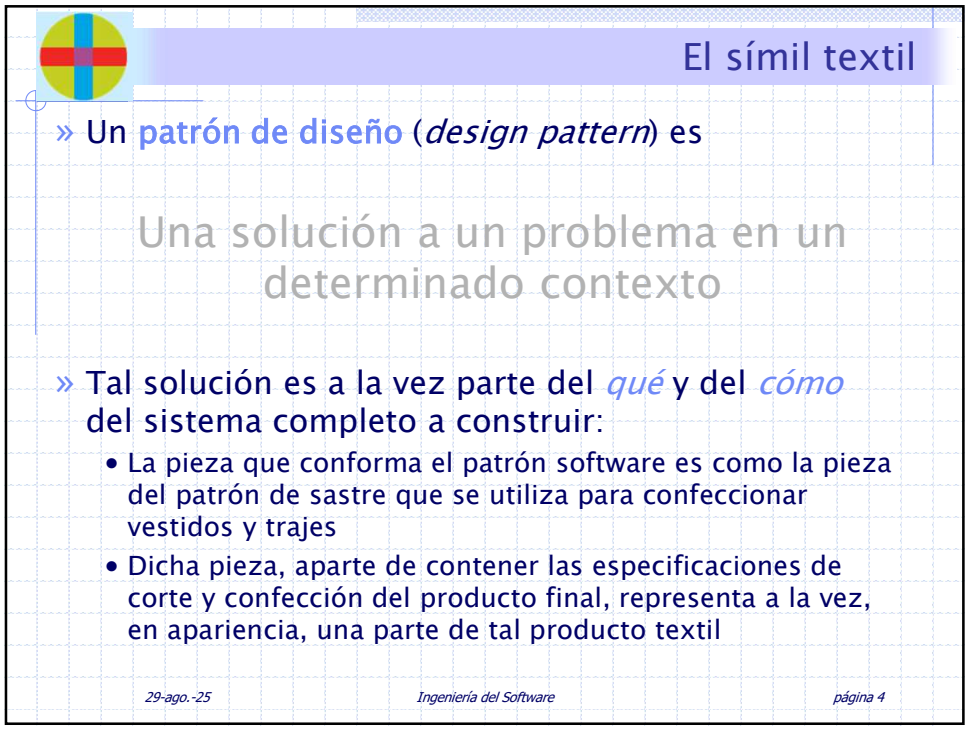
1


Introducción
» Experiencia en el diseño
• Un diseñador experto aplica, intuitiva y automáticamente, criterios precisos que solucionan de forma elegante y eficaz los problemas de modelado software de sistemas reales
• Dicho diseñador suele usar métodos, estructuras y subsistemas que son, a la vez, herramientas del diseño y partes de la solución final, de una manera que difícilmente puede transmitirse a especialistas menos expertos
» Problemas repetidos
• Los "ingenieros de software" se enfrentan cada día a multitud de problemas de distinto calibre
• La "efectividad" de un "ingeniero" se mide por su rapidez y acierto en el diagnóstico, identificación y resolución de tales problemas
• El mejor "ingeniero" es el que más reutiliza la misma solución -matizada- para resolver problemas similares
29-ago.-25 Ingeniería del Software página 2

2


Las buenas prácticas
» No reinventar la rueda
• La POO propugna "no reinventar la rueda" en la codificación con respecto a la resolución de problemas
› ¿Por que, entonces, reinventarla para el ataque genérico a problemas comunes de análisis, diseño e implementación?
• Debe existir alguna forma de comunicar al resto de los "ingenieros" los resultados encontrados tras ímprobos esfuerzos
› Hace falta un sistema de documentación para ello
» ¿Documentación?
• Usar sólo las líneas de código resulta insuficiente, pues fomenta el uso de la técnica de "cut & paste"
• El tipo de los problemas a documentar es muy variado:
› Arquitectura, programación, análisis, diseño, etc.
• Se necesita un formato de documentación único que aúne conceptualmente estos distintos tipos
29-ago.-25 Ingeniería del Software página 3

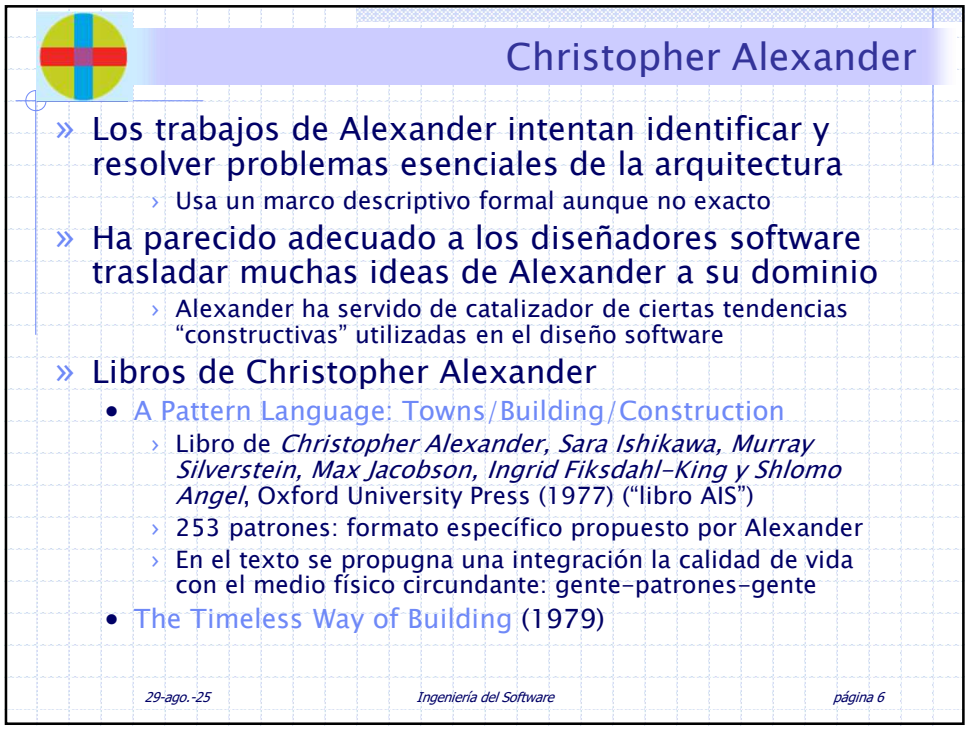
3


El símil textil
» Un patrón de diseño (design pattern) es
Una solución a un problema en un determinado contexto
» Tal solución es a la vez parte del qué y del cómo del sistema completo a construir:
• La pieza que conforma el patrón software es como la pieza del patrón de sastre que se utiliza para confeccionar vestidos y trajes
• Dicha pieza, aparte de contener las especificaciones de corte y confección del producto final, representa a la vez, en apariencia, una parte de tal producto textil
29-ago.-25 Ingeniería del Software página 4

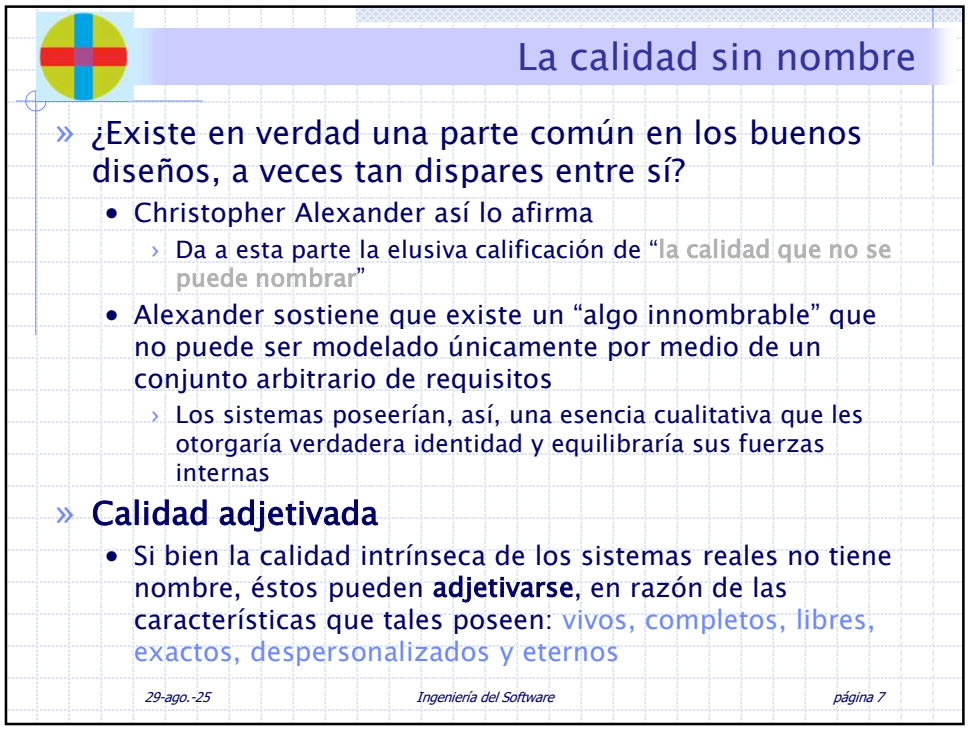
4


Orígenes e historia de los patrones
» Christopher Alexander: patrones en edificios
• Publica los libros "The Timeless Way of Building" (1979) y "A Pattern Language" (1977)
• Acuña el término "patrón" sobre 1977-1979
» Pioneros
• Kent Beck y Ward Cunningham, Textronix (OOPSLA'87)
› Usan las ideas de "patrones" de Alexander para el diseño de GUIs en Smalltalk
• Erich Gamma, en su tesis doctoral (1988-1991)
• James Coplien, en su libro "Advanced C++ Idioms" (1989-1991)
» El Gang-of-Four (GoF): Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
• Publican "Design Patterns" (1995), dando nombre y modelo de los 20 patrones más usados
• El libro solidifica el uso de patrones y se convierte en la referencia clásica
29-ago.-25 Ingeniería del Software página 5

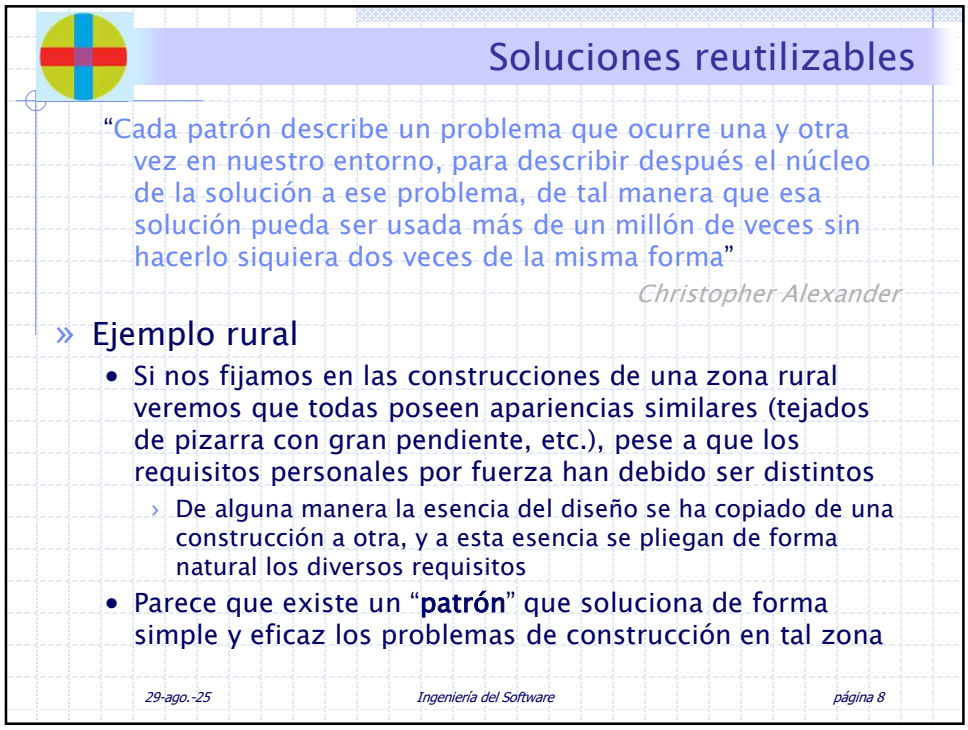
5


Christopher Alexander
» Los trabajos de Alexander intentan identificar y resolver problemas esenciales de la arquitectura
› Usa un marco descriptivo formal aunque no exacto
» Ha parecido adecuado a los diseñadores software trasladar muchas ideas de Alexander a su dominio
› Alexander ha servido de catalizador de ciertas tendencias "constructivas" utilizadas en el diseño software
» Libros de Christopher Alexander
• A Pattern Language: Towns/Building/Construction
› Libro de Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King y Shlomo Angel, Oxford University Press (1977) ("libro AIS")
› 253 patrones: formato específico propuesto por Alexander
› En el texto se propugna una integración la calidad de vida con el medio físico circundante: gente-patrones-gente
• The Timeless Way of Building (1979)
29-ago.-25 Ingeniería del Software página 6

6


La calidad sin nombre
» ¿Existe en verdad una parte común en los buenos diseños, a veces tan dispares entre sí?
• Christopher Alexander así lo afirma
› Da a esta parte la elusiva calificación de "la calidad que no se puede nombrar"
• Alexander sostiene que existe un "algo innombrable" que no puede ser modelado únicamente por medio de un conjunto arbitrario de requisitos
› Los sistemas poseerían, así, una esencia cualitativa que les otorgaría verdadera identidad y equilibraría sus fuerzas internas
» Calidad adjetivada
• Si bien la calidad intrínseca de los sistemas reales no tiene nombre, éstos pueden adjetivarse, en razón de las características que tales poseen: vivos, completos, libres, exactos, despersonalizados y eternos
29-ago.-25 Ingeniería del Software página 7

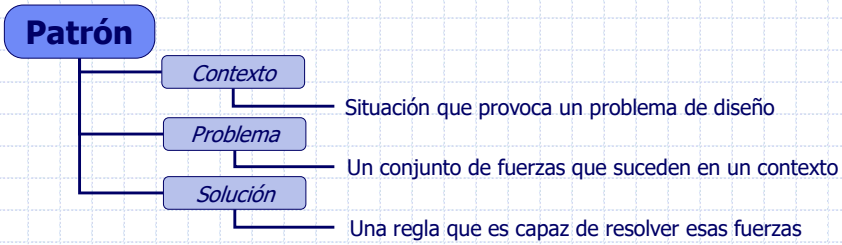
7


Soluciones reutilizables
"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma"
Christopher Alexander
» Ejemplo rural
• Si nos fijamos en las construcciones de una zona rural veremos que todas poseen apariencias similares (tejados de pizarra con gran pendiente, etc.), pese a que los requisitos personales por fuerza han debido ser distintos
› De alguna manera la esencia del diseño se ha copiado de una construcción a otra, y a esta esencia se pliegan de forma natural los diversos requisitos
• Parece que existe un "patrón" que soluciona de forma simple y eficaz los problemas de construcción en tal zona
29-ago.-25 Ingeniería del Software página 8

8

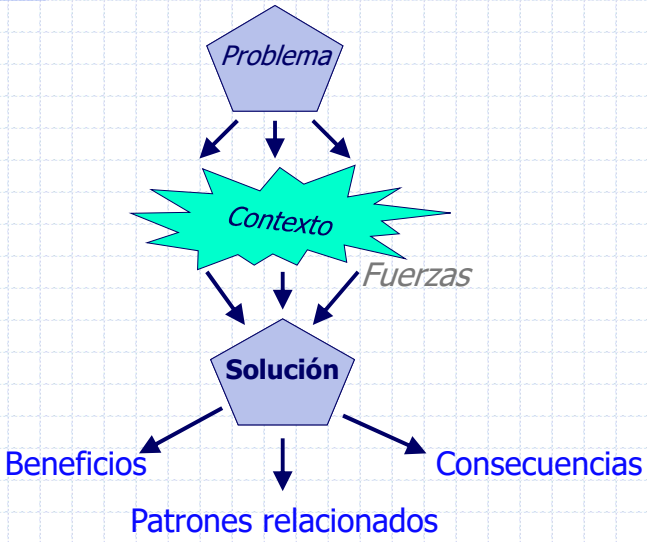
Por tanto, ¿qué es un patrón?

- Un patrón es una solución a un problema en un determinado contexto
  - Documenta de forma abstracta y compacta:
    - El problema (aparece siempre de forma reiterada)
    - El contexto en el que aparece
    - Una buena solución (pasos a seguir, puntos fuertes y débiles)
  - …y encapsula la sabiduría sobre cómo atacar el problema
    - La solución que da el patrón puede usarse más de un millón de veces sin hacerlo dos veces de la misma forma

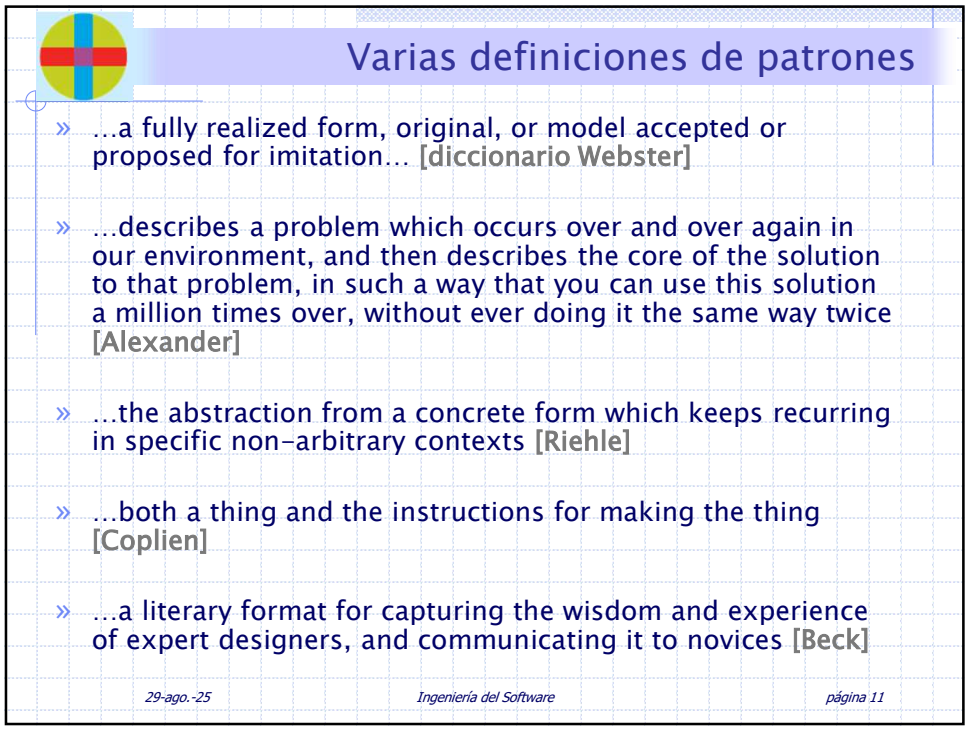


9

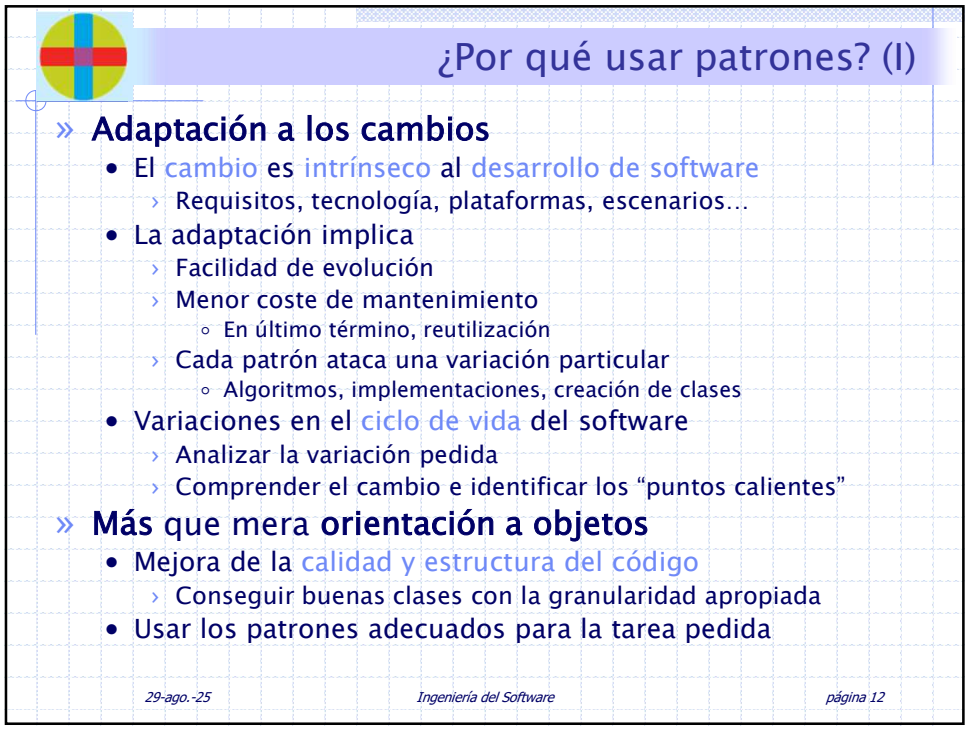
¿Cómo aparecen los patrones?



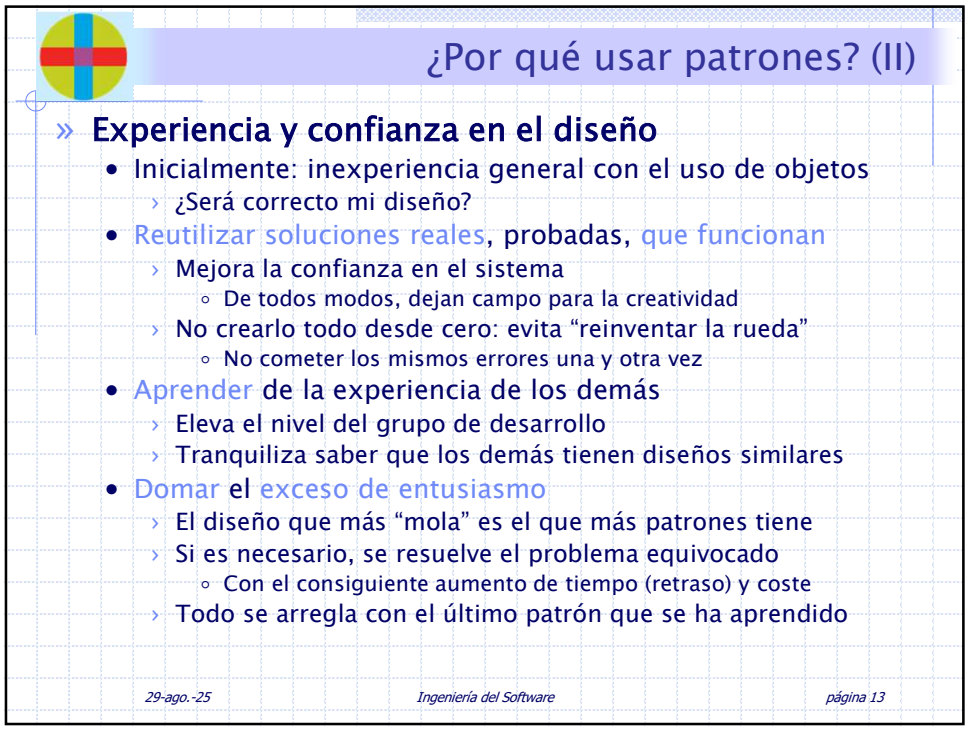
10


Varias definiciones de patrones
» …a fully realized form, original, or model accepted or proposed for imitation… [diccionario Webster]
» …describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice [Alexander]
» …the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts [Riehle]
» …both a thing and the instructions for making the thing [Coplien]
» …a literary format for capturing the wisdom and experience of expert designers, and communicating it to novices [Beck]
29-ago.-25 Ingeniería del Software página 11

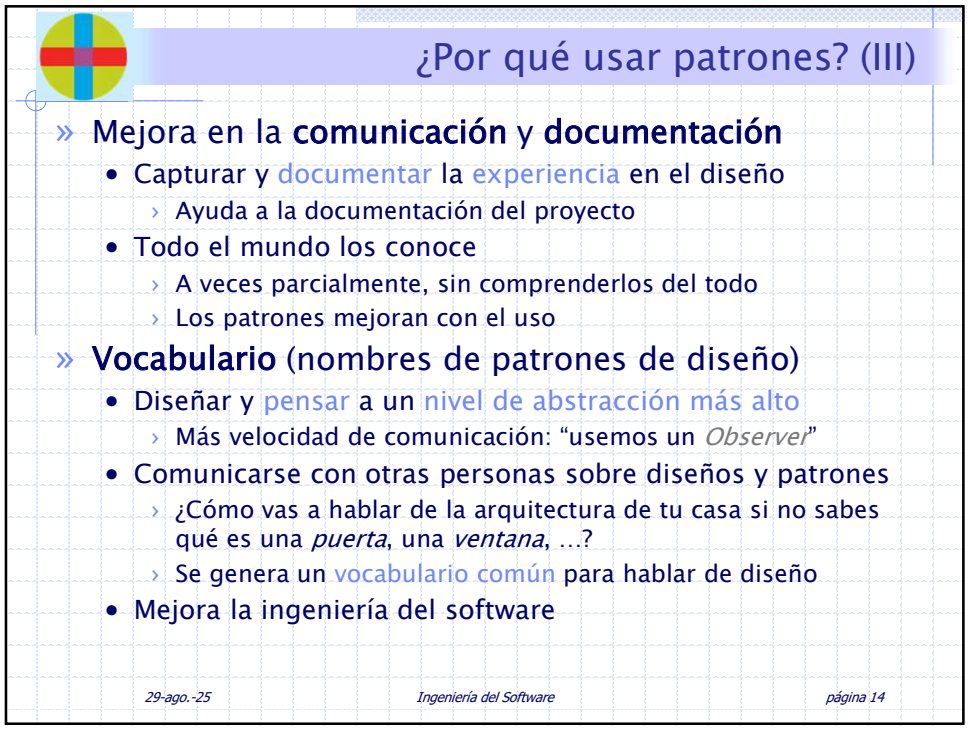
11


¿Por qué usar patrones? (I)
» Adaptación a los cambios
• El cambio es intrínseco al desarrollo de software
› Requisitos, tecnología, plataformas, escenarios…
• La adaptación implica
› Facilidad de evolución
› Menor coste de mantenimiento
◦ En último término, reutilización
› Cada patrón ataca una variación particular
◦ Algoritmos, implementaciones, creación de clases
• Variaciones en el ciclo de vida del software
› Analizar la variación pedida
› Comprender el cambio e identificar los "puntos calientes"
» Más que mera orientación a objetos
• Mejora de la calidad y estructura del código
› Conseguir buenas clases con la granularidad apropiada
• Usar los patrones adecuados para la tarea pedida
29-ago.-25 Ingeniería del Software página 12

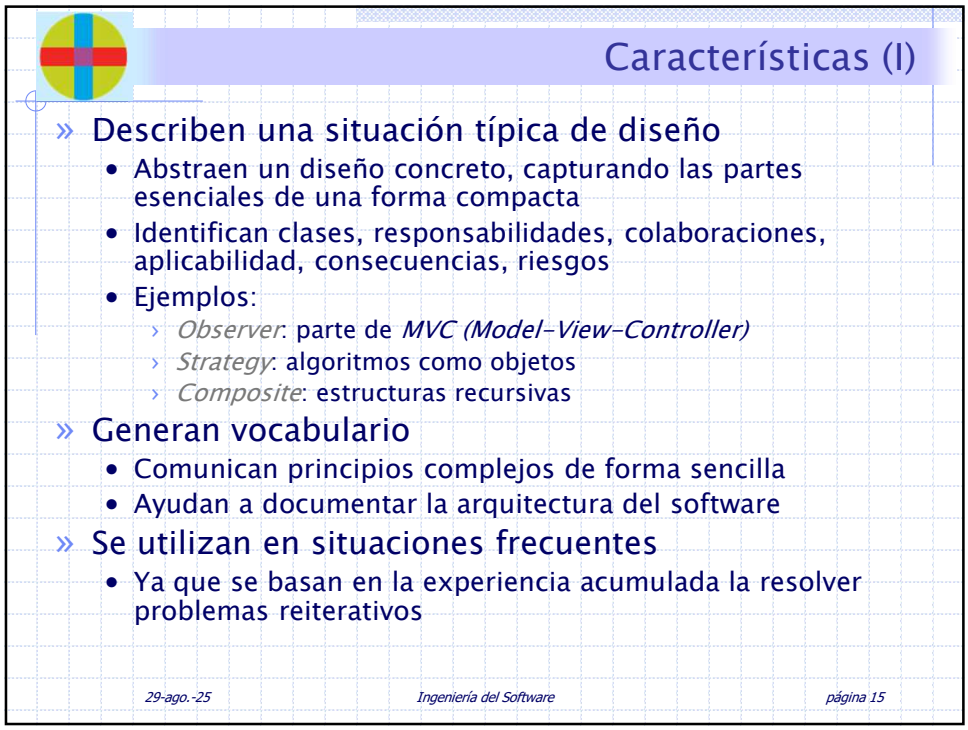
12


¿Por qué usar patrones? (II)
» Experiencia y confianza en el diseño
• Inicialmente: inexperiencia general con el uso de objetos
› ¿Será correcto mi diseño?
• Reutilizar soluciones reales, probadas, que funcionan
› Mejora la confianza en el sistema
◦ De todos modos, dejan campo para la creatividad
› No crearlo todo desde cero: evita "reinventar la rueda"
◦ No cometer los mismos errores una y otra vez
• Aprender de la experiencia de los demás
› Eleva el nivel del grupo de desarrollo
› Tranquiliza saber que los demás tienen diseños similares
• Domar el exceso de entusiasmo
› El diseño que más "mola" es el que más patrones tiene
› Si es necesario, se resuelve el problema equivocado
◦ Con el consiguiente aumento de tiempo (retraso) y coste
› Todo se arregla con el último patrón que se ha aprendido
29-ago.-25 Ingeniería del Software página 13

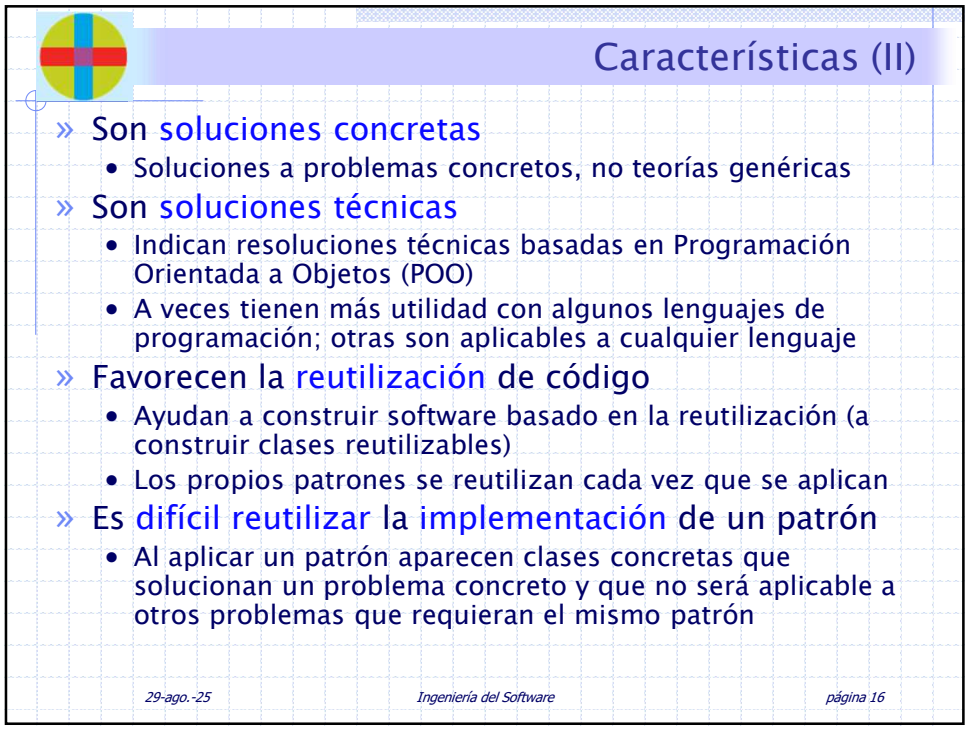
13


¿Por qué usar patrones? (III)
» Mejora en la comunicación y documentación
• Capturar y documentar la experiencia en el diseño
› Ayuda a la documentación del proyecto
• Todo el mundo los conoce
› A veces parcialmente, sin comprenderlos del todo
› Los patrones mejoran con el uso
» Vocabulario (nombres de patrones de diseño)
• Diseñar y pensar a un nivel de abstracción más alto
› Más velocidad de comunicación: "usemos un Observer"
• Comunicarse con otras personas sobre diseños y patrones
› ¿Cómo vas a hablar de la arquitectura de tu casa si no sabes qué es una puerta, una ventana, ...?
› Se genera un vocabulario común para hablar de diseño
• Mejora la ingeniería del software
29-ago.-25 Ingeniería del Software página 14

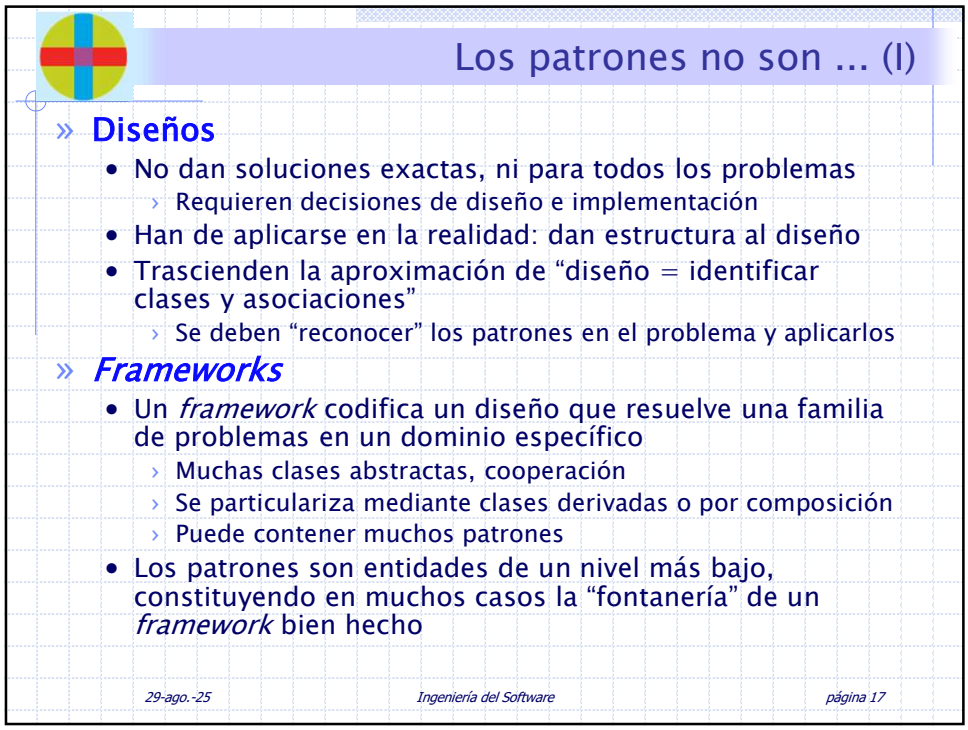
14


Características (I)
» Describen una situación típica de diseño
• Abstraen un diseño concreto, capturando las partes esenciales de una forma compacta
• Identifican clases, responsabilidades, colaboraciones, aplicabilidad, consecuencias, riesgos
• Ejemplos:
› Observer: parte de MVC (Model-View-Controller)
› Strategy: algoritmos como objetos
› Composite: estructuras recursivas
» Generan vocabulario
• Comunican principios complejos de forma sencilla
• Ayudan a documentar la arquitectura del software
» Se utilizan en situaciones frecuentes
• Ya que se basan en la experiencia acumulada la resolver problemas reiterativos
29-ago.-25 Ingeniería del Software página 15

15


Características (II)
» Son soluciones concretas
• Soluciones a problemas concretos, no teorías genéricas
» Son soluciones técnicas
• Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO)
• A veces tienen más utilidad con algunos lenguajes de programación; otras son aplicables a cualquier lenguaje
» Favorecen la reutilización de código
• Ayudan a construir software basado en la reutilización (a construir clases reutilizables)
• Los propios patrones se reutilizan cada vez que se aplican
» Es difícil reutilizar la implementación de un patrón
• Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón
29-ago.-25 Ingeniería del Software página 16

16

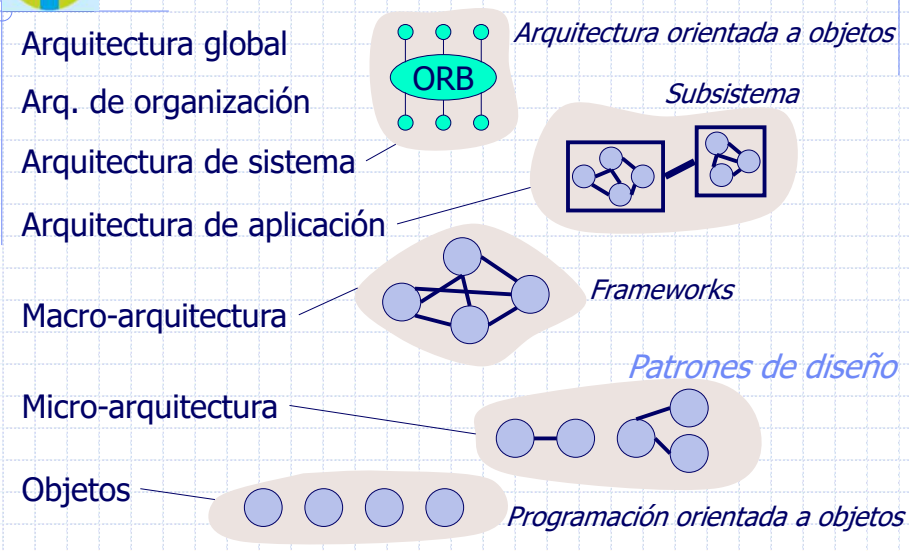

Los patrones no son ... (I)
» Diseños
• No dan soluciones exactas, ni para todos los problemas
› Requieren decisiones de diseño e implementación
• Han de aplicarse en la realidad: dan estructura al diseño
• Trascienden la aproximación de "diseño = identificar clases y asociaciones"
› Se deben "reconocer" los patrones en el problema y aplicarlos
» Frameworks
• Un framework codifica un diseño que resuelve una familia de problemas en un dominio específico
› Muchas clases abstractas, cooperación
› Se particulariza mediante clases derivadas o por composición
› Puede contener muchos patrones
• Los patrones son entidades de un nivel más bajo, constituyendo en muchos casos la "fontanería" de un framework bien hecho
29-ago.-25 Ingeniería del Software página 17

17


Los patrones no son ... (II)
» Arquitectura
• Son de un nivel más bajo
› Recordar el diagrama de los 7 niveles
• Existen patrones de arquitectura
› Mowbray and Malveau (patrones CORBA), Buschmann et al, Schmidt et al
• Más enfocados al middleware
› Concurrencia, sincronización, distribución
» Conceptos de programación
• Los patrones son de aplicación fuera de las técnicas de diseño y programación orientados a objetos
› Objetos, Clases, Bibliotecas, Paquetes, Módulos, Macros, Plantillas (Templates), Funciones, Clausuras, …
29-ago.-25 Ingeniería del Software página 18

18

Las siete capas de arquitectura *



* Mowbray y Malveau

19

Principios de diseño

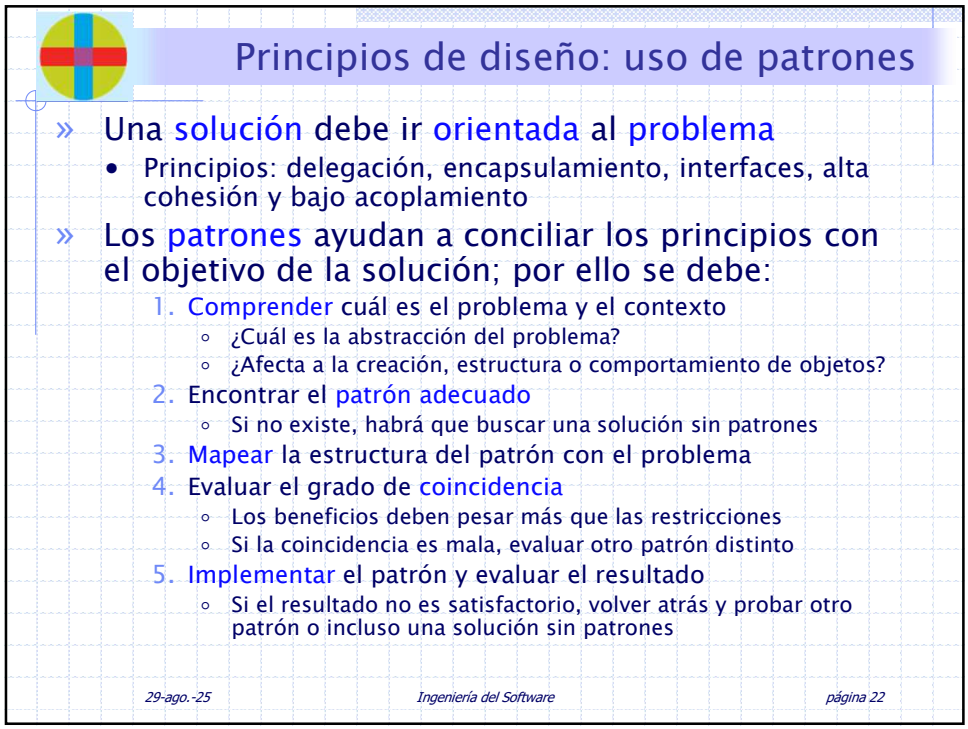
» Reglas básicas para el diseño software
- El cambio es una realidad: acéptalo al diseñar
- Debe ser sencillo localizar el punto del diseño donde aplicar un cambio
- Dicho punto debe ser único: un cambio, un lugar
- El impacto de un cambio debe ser mínimo sobre el resto del software
- Un error al cambiar X no debe romper Y
- Debe simplificarse el diseño sin piedad
  - › Delegando hacia arriba o hacia abajo

» Una regla más…
- El mundo real es 1000 veces más complejo que los ejemplos que se plantean en los libros

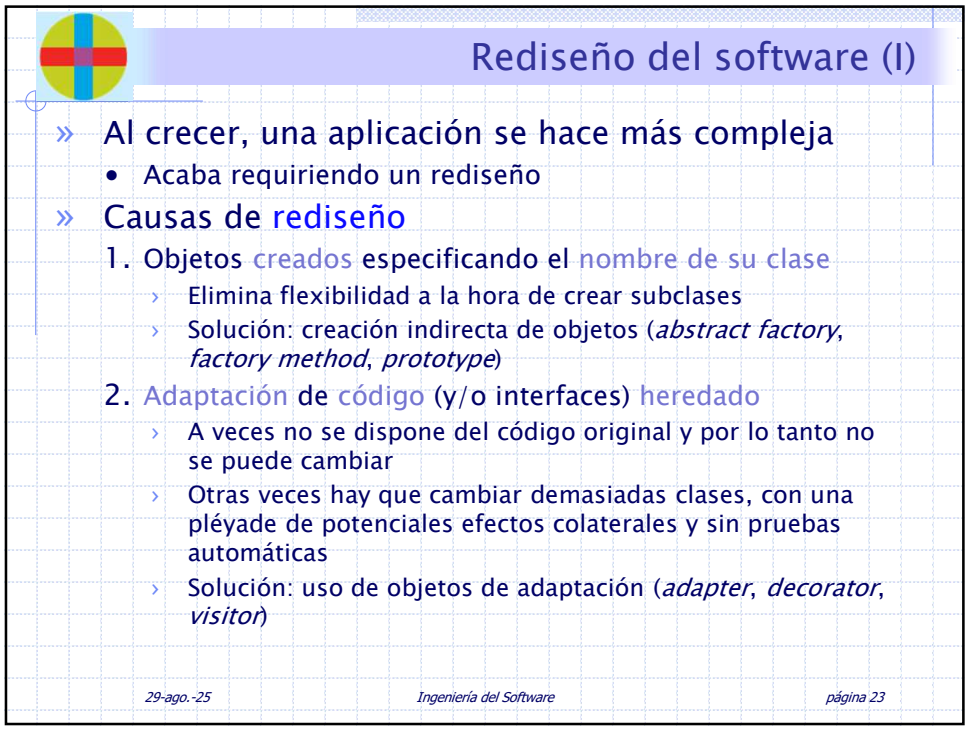
20


Principios de diseño: delegación
Herencia
» Reutilización de caja blanca
» Rompe (viola) el encapsulamiento
• Aparecen dependencias de interfaces y de implementaciones
» Pocos objetos, demasiado grandes
Composición (delegación)
» Reutilización de caja negra
» Usa objetos pequeños y bien definidos
• Cada objeto se ocupa de una sola tarea
• El comportamiento de un sistema complejo se consigue mediante las interrelaciones
Favorecer la composición sobre la herencia
Delegar tareas en los objetos adecuados
Heredar sólo de clases abstractas (poca implementación) o de interfaces (ninguna)
29-ago.-25 Ingeniería del Software página 21

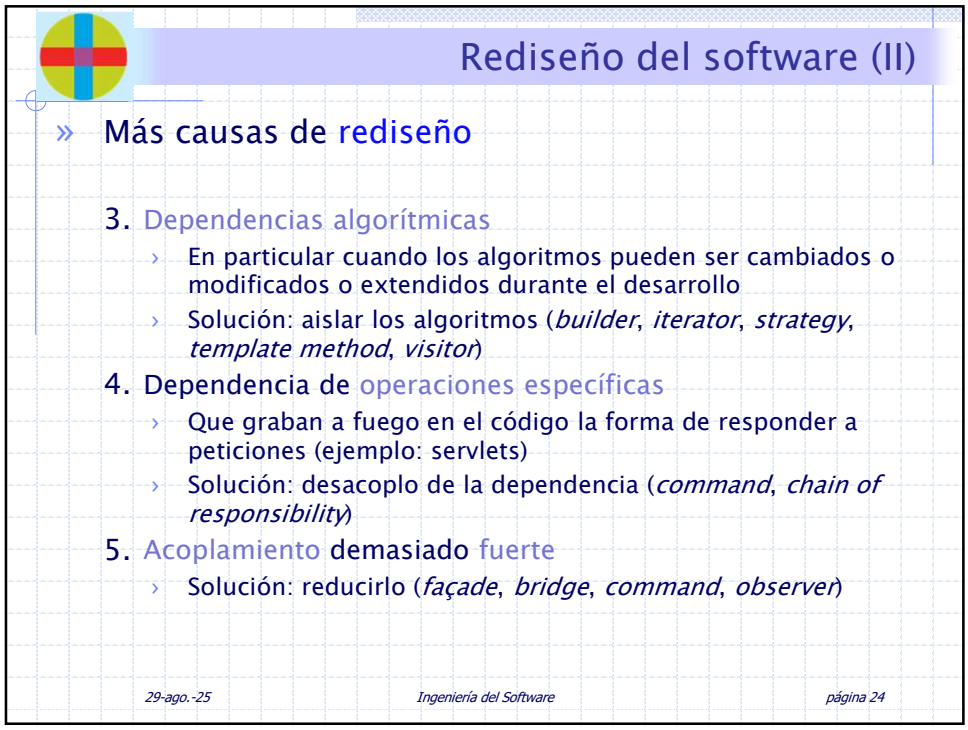
21


Principios de diseño: uso de patrones
» Una solución debe ir orientada al problema
• Principios: delegación, encapsulamiento, interfaces, alta cohesión y bajo acoplamiento
» Los patrones ayudan a conciliar los principios con el objetivo de la solución; por ello se debe:
1. Comprender cuál es el problema y el contexto
◦ ¿Cuál es la abstracción del problema?
◦ ¿Afecta a la creación, estructura o comportamiento de objetos?
2. Encontrar el patrón adecuado
◦ Si no existe, habrá que buscar una solución sin patrones
3. Mapear la estructura del patrón con el problema
4. Evaluar el grado de coincidencia
◦ Los beneficios deben pesar más que las restricciones
◦ Si la coincidencia es mala, evaluar otro patrón distinto
5. Implementar el patrón y evaluar el resultado
◦ Si el resultado no es satisfactorio, volver atrás y probar otro patrón o incluso una solución sin patrones
29-ago.-25 Ingeniería del Software página 22

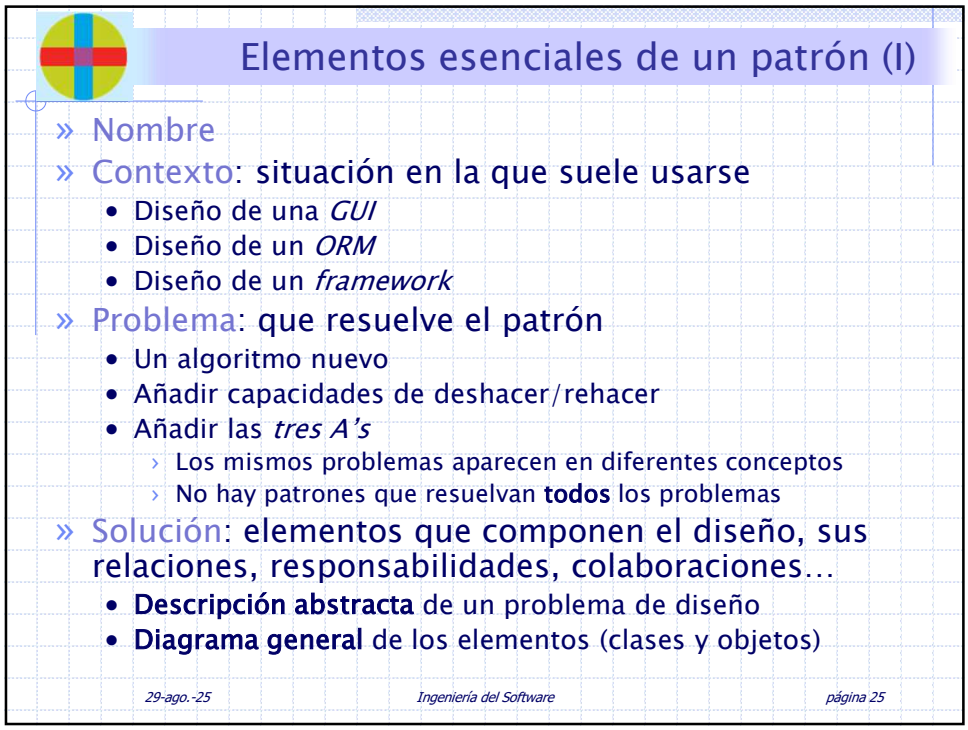
22


Rediseño del software (I)
» Al crecer, una aplicación se hace más compleja
• Acaba requiriendo un rediseño
» Causas de rediseño
1. Objetos creados especificando el nombre de su clase
› Elimina flexibilidad a la hora de crear subclases
› Solución: creación indirecta de objetos (abstract factory, factory method, prototype)
2. Adaptación de código (y/o interfaces) heredado
› A veces no se dispone del código original y por lo tanto no se puede cambiar
› Otras veces hay que cambiar demasiadas clases, con una pléyade de potenciales efectos colaterales y sin pruebas automáticas
› Solución: uso de objetos de adaptación (adapter, decorator, visitor)
29-ago.-25 Ingeniería del Software página 23

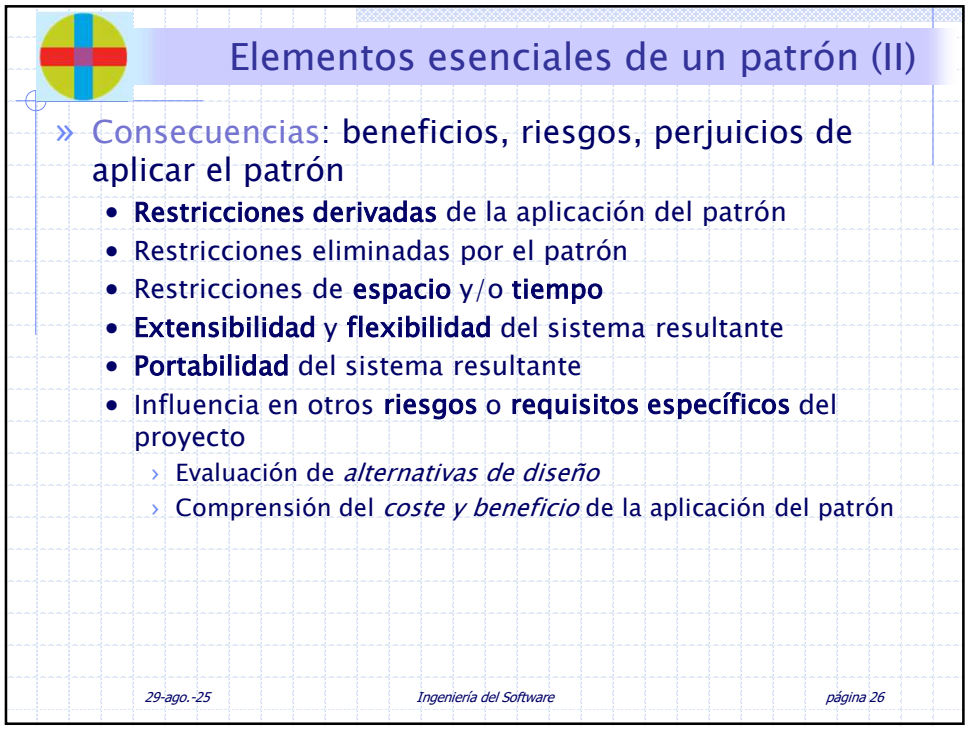
23


Rediseño del software (II)
» Más causas de rediseño
3. Dependencias algorítmicas
› En particular cuando los algoritmos pueden ser cambiados o modificados o extendidos durante el desarrollo
› Solución: aislar los algoritmos (builder, iterator, strategy, template method, visitor)
4. Dependencia de operaciones específicas
› Que graban a fuego en el código la forma de responder a peticiones (ejemplo: servlets)
› Solución: desacoplo de la dependencia (command, chain of responsibility)
5. Acoplamiento demasiado fuerte
› Solución: reducirlo (façade, bridge, command, observer)
29-ago.-25 Ingeniería del Software página 24

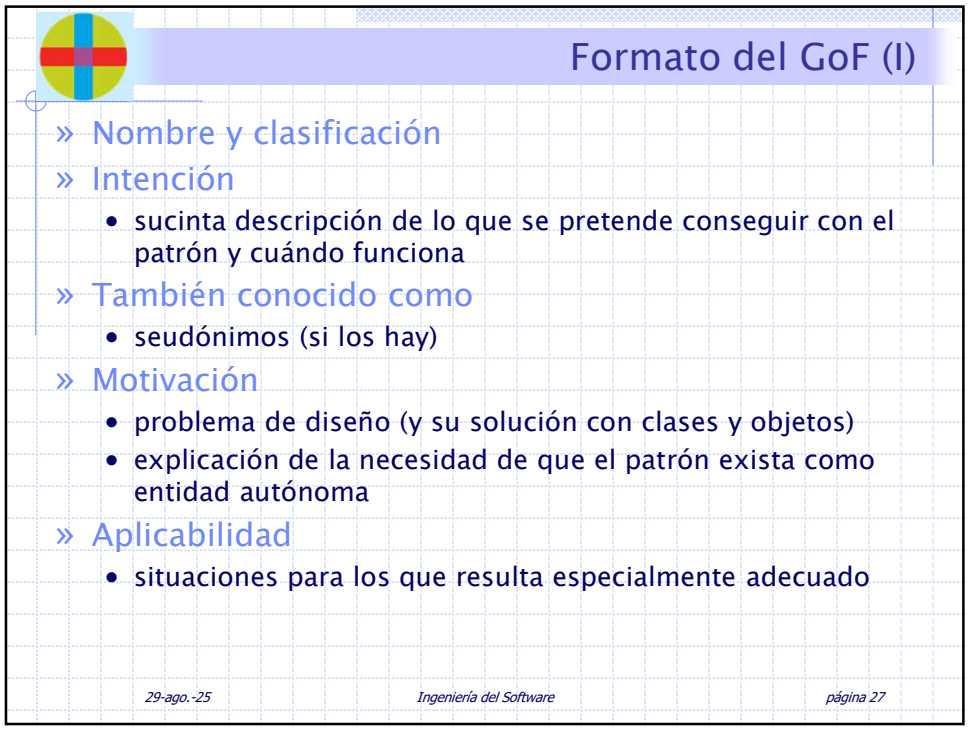
24


Elementos esenciales de un patrón (I)
» Nombre
» Contexto: situación en la que suele usarse
• Diseño de una GUI
• Diseño de un ORM
• Diseño de un framework
» Problema: que resuelve el patrón
• Un algoritmo nuevo
• Añadir capacidades de deshacer/rehacer
• Añadir las tres A's
› Los mismos problemas aparecen en diferentes conceptos
› No hay patrones que resuelvan todos los problemas
» Solución: elementos que componen el diseño, sus relaciones, responsabilidades, colaboraciones…
• Descripción abstracta de un problema de diseño
• Diagrama general de los elementos (clases y objetos)
29-ago.-25 Ingeniería del Software página 25

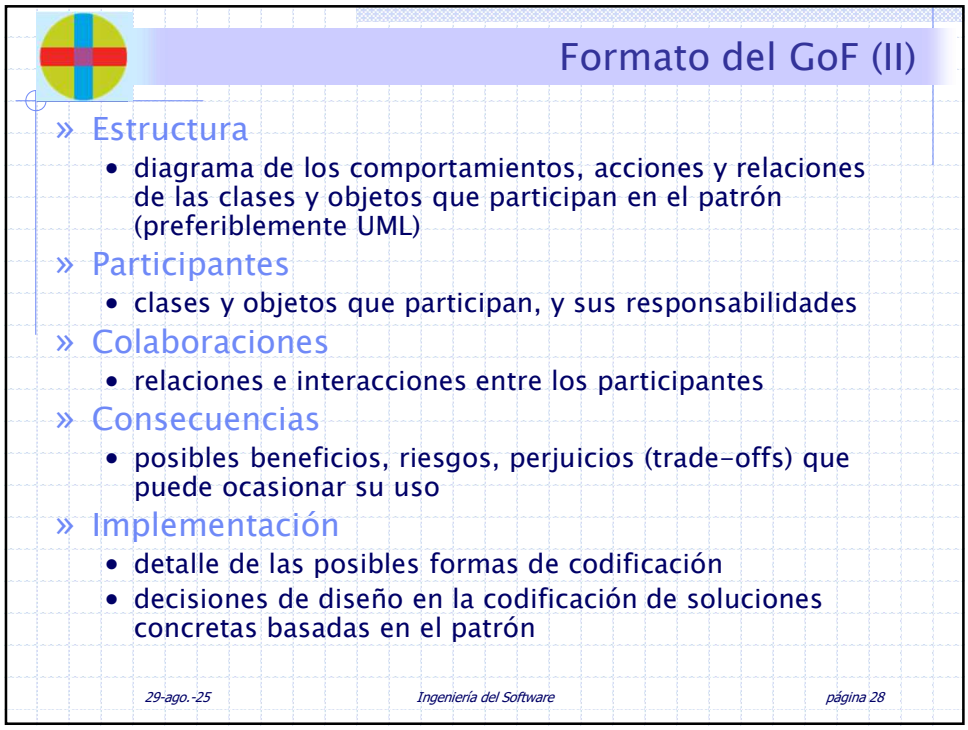
25


Elementos esenciales de un patrón (II)
» Consecuencias: beneficios, riesgos, perjuicios de aplicar el patrón
• Restricciones derivadas de la aplicación del patrón
• Restricciones eliminadas por el patrón
• Restricciones de espacio y/o tiempo
• Extensibilidad y flexibilidad del sistema resultante
• Portabilidad del sistema resultante
• Influencia en otros riesgos o requisitos específicos del proyecto
› Evaluación de alternativas de diseño
› Comprensión del coste y beneficio de la aplicación del patrón
29-ago.-25 Ingeniería del Software página 26

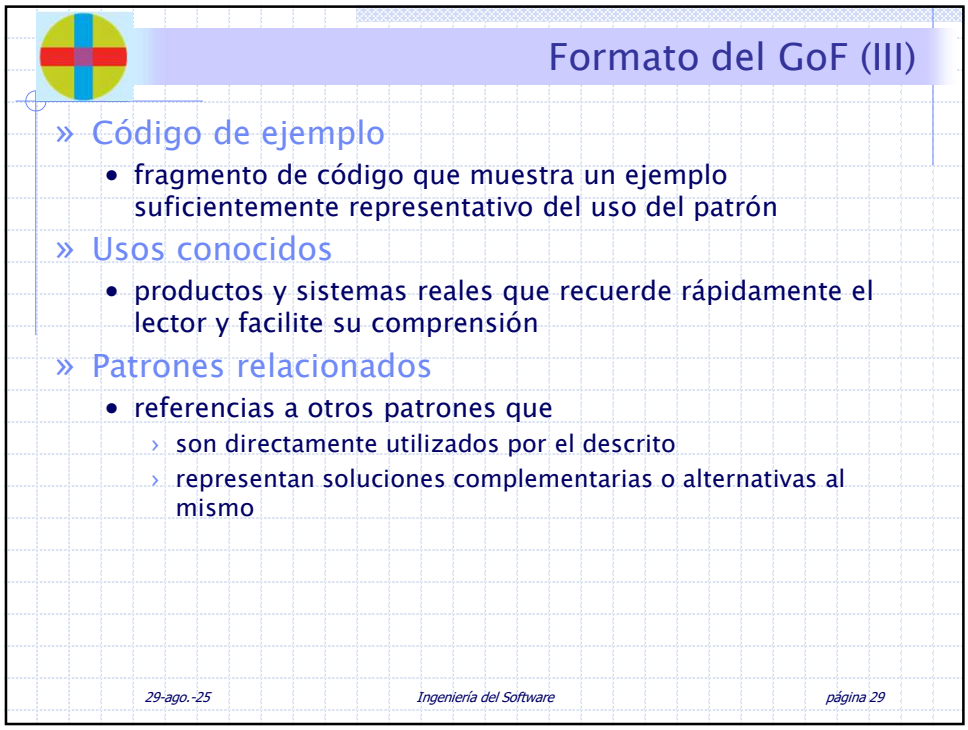
26


Formato del GoF (I)
» Nombre y clasificación
» Intención
• sucinta descripción de lo que se pretende conseguir con el patrón y cuándo funciona
» También conocido como
• seudónimos (si los hay)
» Motivación
• problema de diseño (y su solución con clases y objetos)
• explicación de la necesidad de que el patrón exista como entidad autónoma
» Aplicabilidad
• situaciones para los que resulta especialmente adecuado
29-ago.-25 Ingeniería del Software página 27

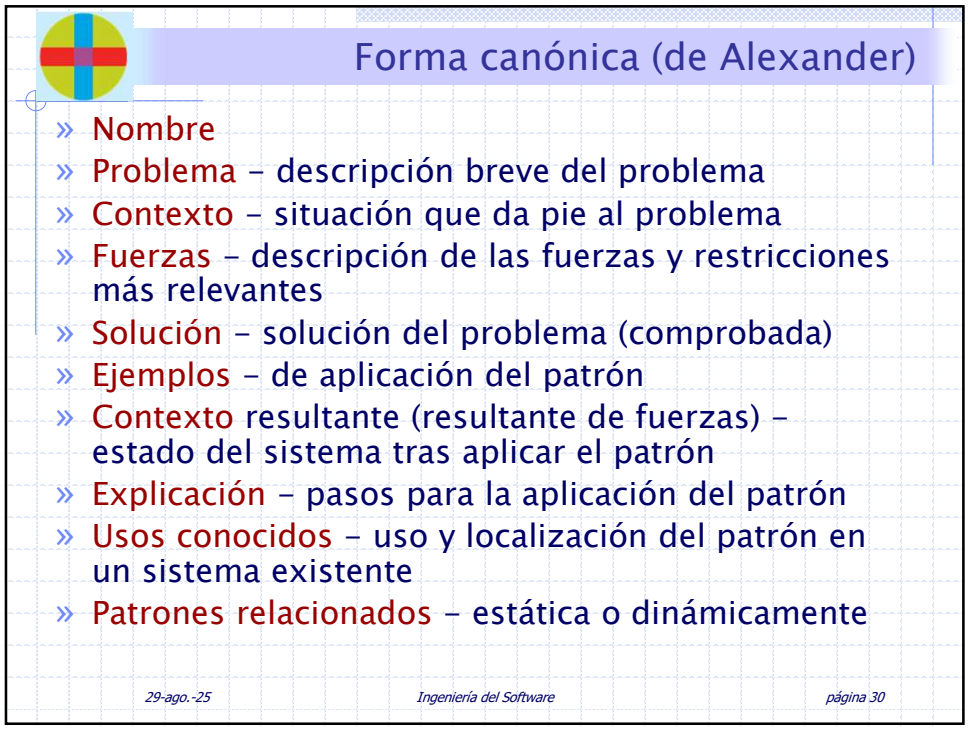
27


Formato del GoF (II)
» Estructura
• diagrama de los comportamientos, acciones y relaciones de las clases y objetos que participan en el patrón (preferiblemente UML)
» Participantes
• clases y objetos que participan, y sus responsabilidades
» Colaboraciones
• relaciones e interacciones entre los participantes
» Consecuencias
• posibles beneficios, riesgos, perjuicios (trade-offs) que puede ocasionar su uso
» Implementación
• detalle de las posibles formas de codificación
• decisiones de diseño en la codificación de soluciones concretas basadas en el patrón
29-ago.-25 Ingeniería del Software página 28

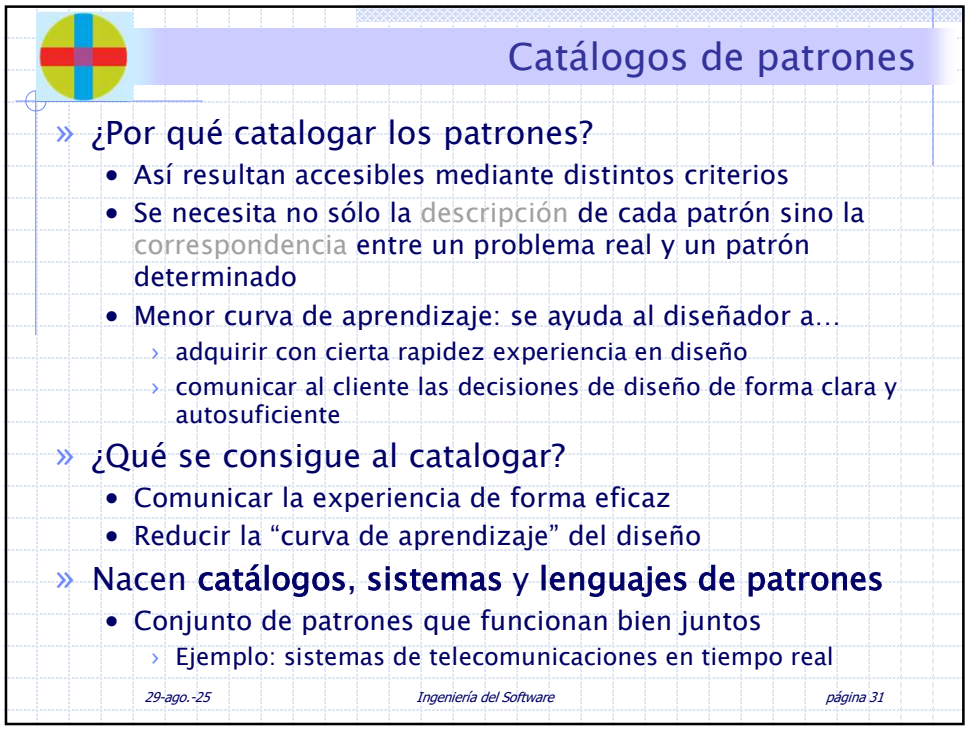
28


Formato del GoF (III)
» Código de ejemplo
• fragmento de código que muestra un ejemplo suficientemente representativo del uso del patrón
» Usos conocidos
• productos y sistemas reales que recuerde rápidamente el lector y facilite su comprensión
» Patrones relacionados
• referencias a otros patrones que
› son directamente utilizados por el descrito
› representan soluciones complementarias o alternativas al mismo
29-ago.-25 Ingeniería del Software página 29

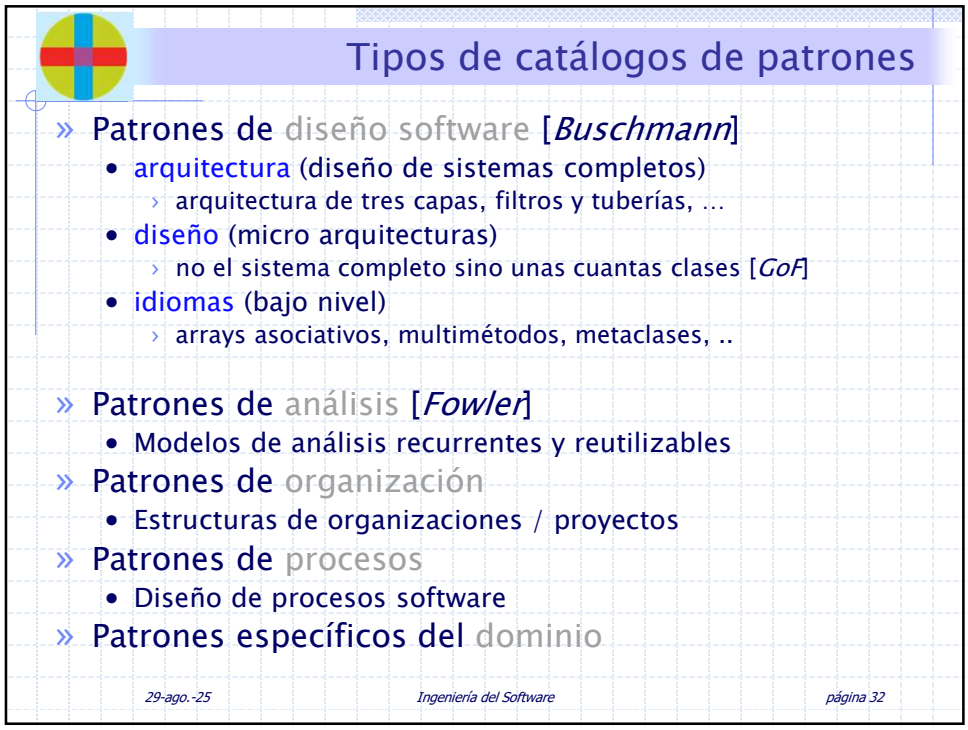
29


Forma canónica (de Alexander)
» Nombre
» Problema – descripción breve del problema
» Contexto – situación que da pie al problema
» Fuerzas – descripción de las fuerzas y restricciones más relevantes
» Solución – solución del problema (comprobada)
» Ejemplos – de aplicación del patrón
» Contexto resultante (resultante de fuerzas) – estado del sistema tras aplicar el patrón
» Explicación – pasos para la aplicación del patrón
» Usos conocidos – uso y localización del patrón en un sistema existente
» Patrones relacionados – estática o dinámicamente
29-ago.-25 Ingeniería del Software página 30

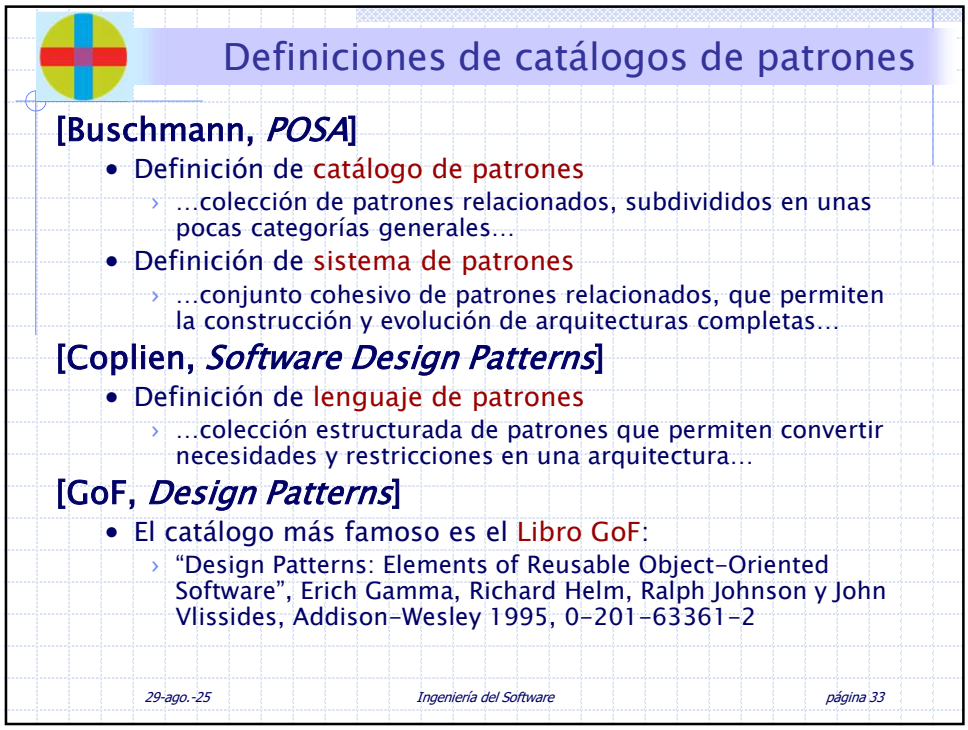
30


Catálogos de patrones
» ¿Por qué catalogar los patrones?
• Así resultan accesibles mediante distintos criterios
• Se necesita no sólo la descripción de cada patrón sino la correspondencia entre un problema real y un patrón determinado
• Menor curva de aprendizaje: se ayuda al diseñador a…
› adquirir con cierta rapidez experiencia en diseño
› comunicar al cliente las decisiones de diseño de forma clara y autosuficiente
» ¿Qué se consigue al catalogar?
• Comunicar la experiencia de forma eficaz
• Reducir la "curva de aprendizaje" del diseño
» Nacen catálogos, sistemas y lenguajes de patrones
• Conjunto de patrones que funcionan bien juntos
› Ejemplo: sistemas de telecomunicaciones en tiempo real
29-ago.-25 Ingeniería del Software página 31

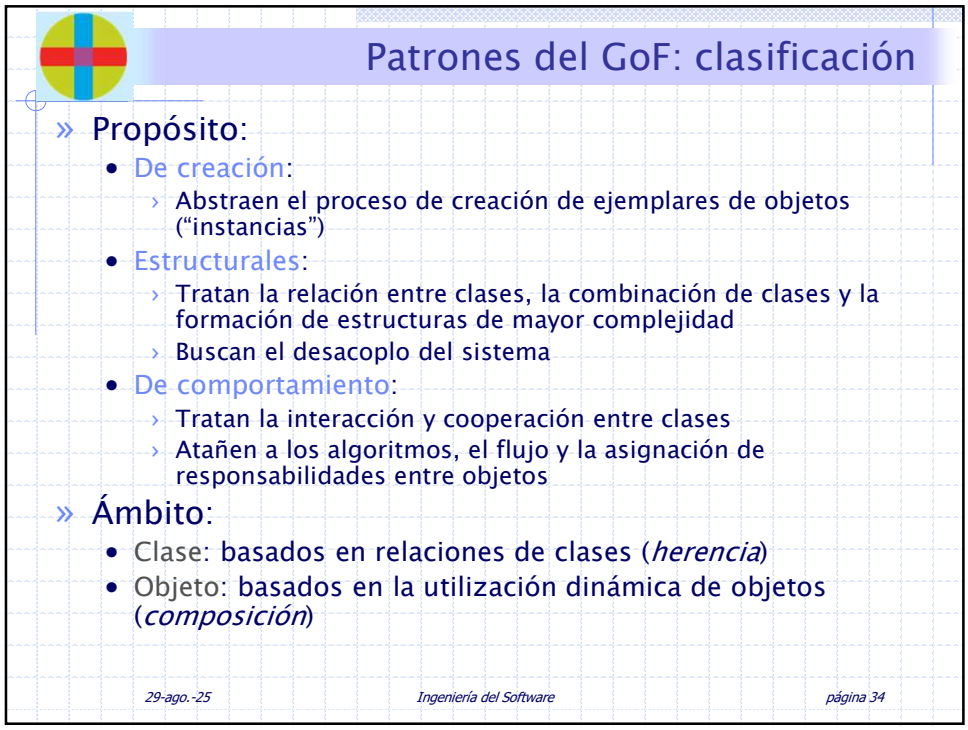
31


Tipos de catálogos de patrones
» Patrones de diseño software [Buschmann]
• arquitectura (diseño de sistemas completos)
› arquitectura de tres capas, filtros y tuberías, …
• diseño (micro arquitecturas)
› no el sistema completo sino unas cuantas clases [GoF]
• idiomas (bajo nivel)
› arrays asociativos, multimétodos, metaclases, ..
» Patrones de análisis [Fowler]
• Modelos de análisis recurrentes y reutilizables
» Patrones de organización
• Estructuras de organizaciones / proyectos
» Patrones de procesos
• Diseño de procesos software
» Patrones específicos del dominio
29-ago.-25 Ingeniería del Software página 32

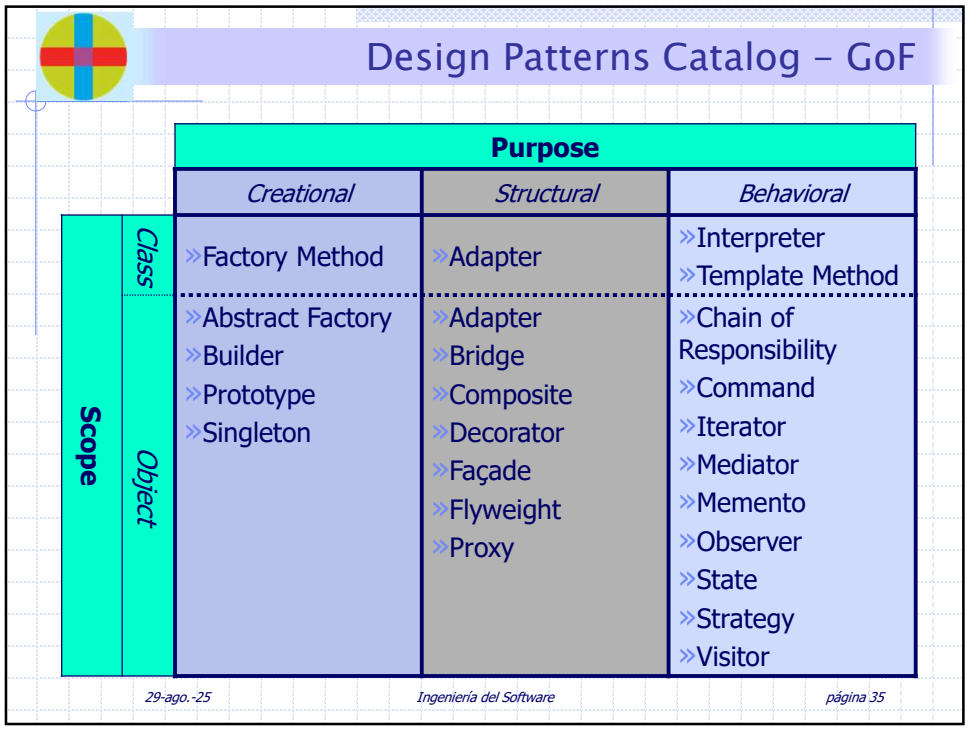
32


Definiciones de catálogos de patrones
[Buschmann, POSA]
Definición de catálogo de patrones
…colección de patrones relacionados, subdivididos en unas pocas categorías generales…
Definición de sistema de patrones
…conjunto cohesivo de patrones relacionados, que permiten la construcción y evolución de arquitecturas completas…
[Coplien, Software Design Patterns]
Definición de lenguaje de patrones
…colección estructurada de patrones que permiten convertir necesidades y restricciones en una arquitectura…
[GoF, Design Patterns]
El catálogo más famoso es el Libro GoF:
"Design Patterns: Elements of Reusable Object-Oriented Software", Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, Addison-Wesley 1995, 0-201-63361-2
29-ago.-25
Ingeniería del Software
página 33

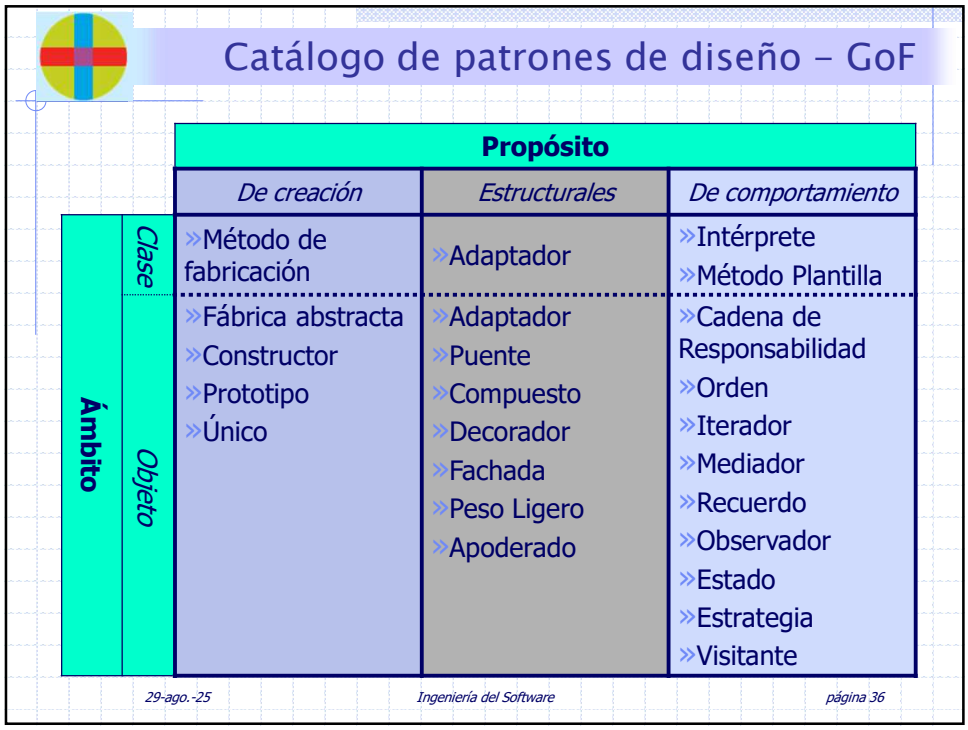
33


Patrones del GoF: clasificación
Propósito:
De creación:
Abstraen el proceso de creación de ejemplares de objetos ("instancias")
Estructurales:
Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad
Buscan el desacoplo del sistema
De comportamiento:
Tratan la interacción y cooperación entre clases
Atañen a los algoritmos, el flujo y la asignación de responsabilidades entre objetos
Ámbito:
Clase: basados en relaciones de clases (herencia)
Objeto: basados en la utilización dinámica de objetos (composición)
29-ago.-25
Ingeniería del Software
página 34

34


Design Patterns Catalog – GoF
Purpose
Creational
Structural
Behavioral
Scope
Class
Object
»Factory Method
»Adapter
»Interpreter
»Template Method
»Abstract Factory
»Builder
»Prototype
»Singleton
»Adapter
»Bridge
»Composite
»Decorator
»Façade
»Flyweight
»Proxy
»Chain of Responsibility
»Command
»Iterator
»Mediator
»Memento
»Observer
»State
»Strategy
»Visitor
29-ago.-25
Ingeniería del Software
página 35

35


Catálogo de patrones de diseño – GoF
Propósito
De creación
Estructurales
De comportamiento
Ámbito
Clase
Objeto
»Método de fabricación
»Adaptador
»Intérprete
»Método Plantilla
»Fábrica abstracta
»Constructor
»Prototipo
»Único
»Adaptador
»Puente
»Compuesto
»Decorador
»Fachada
»Peso Ligero
»Apoderado
»Cadena de Responsabilidad
»Orden
»Iterador
»Mediador
»Recuerdo
»Observador
»Estado
»Estrategia
»Visitante
29-ago.-25
Ingeniería del Software
página 36

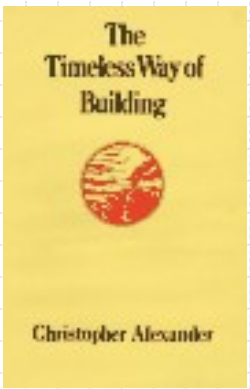
36

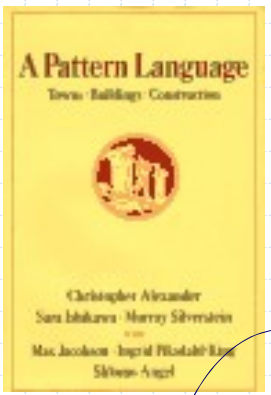
Notas finales

- Los patrones de diseño (GoF) dan una base para la mejora en el conocimiento de:
  - Diseño orientado a objetos
  - Arquitectura software
- Comprender bien los patrones lleva tiempo
  - Releerlos de vez en cuando ayuda
  - ¡Y aplicarlos en tus diseños también!
- En cuanto a su uso…
  - ¿Patrones como moda? Mejor como intención
  - ¿Patrones como solución? Mejor como núcleo de soluciones
  - ¿Patrones como normas? Mejor como sugerencias

37

Libros mencionados…



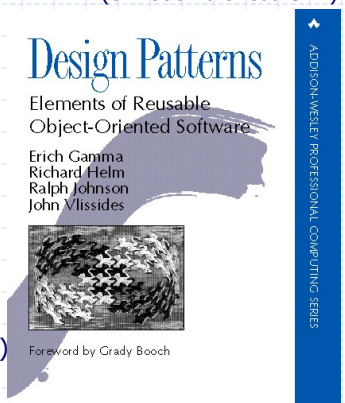


Dos libros de *arquitectura* (no de software) de Christopher Alexander, et al.

Usa la notación de Booch para los diagramas

(en 1995 no existía UML)

(el GoF "Gang of Four")



38

Bibliografía y recursos

- Bibliografía
  - Gamma, E. et al: *Design Patterns*, Addison-Wesley 1997
  - Larman, C: *Applying UML and Patterns*, Prentice-Hall 1997
  - Fowler, M: *Analysis Patterns: Reusable Object Models* 1997
  - *Pattern Languages of Program Design 1, 2, and 3*, Addison-Wesley
  - Buschmann, F. et al: *Pattern-Oriented Software Architecture - A System of Patterns* 1996
- Recursos on-line
  - Librerías de patrones: http://hillside.net/patterns
  - Portland Pattern Repository: http://wiki.c2.com
  - Patrones del GoF: http://www.blackwasp.co.uk/gofpatterns.aspx

39

¿Preguntas?

40