



Análisis y Diseño de Sistemas 2

2015

Ing. Luis Alberto Arias Solórzano

Unidad 3

Patrones de Arquitectura

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Los patrones de arquitectura representan el nivel más alto en el sistema de patrones propuesto.

Los patrones de arquitectura, se dividen en las siguientes categorías:

- ▶ **From Mud to Structure:** los patrones en esta categoría ayudan a evitar un “mar” de componentes u objetos. En particular, soportan una descomposición controlada de una tarea del sistema en subtarear que cooperan.
- ▶ **Distributed Systems**
- ▶ **Interactive Systems**
- ▶ **Adaptable Systems**

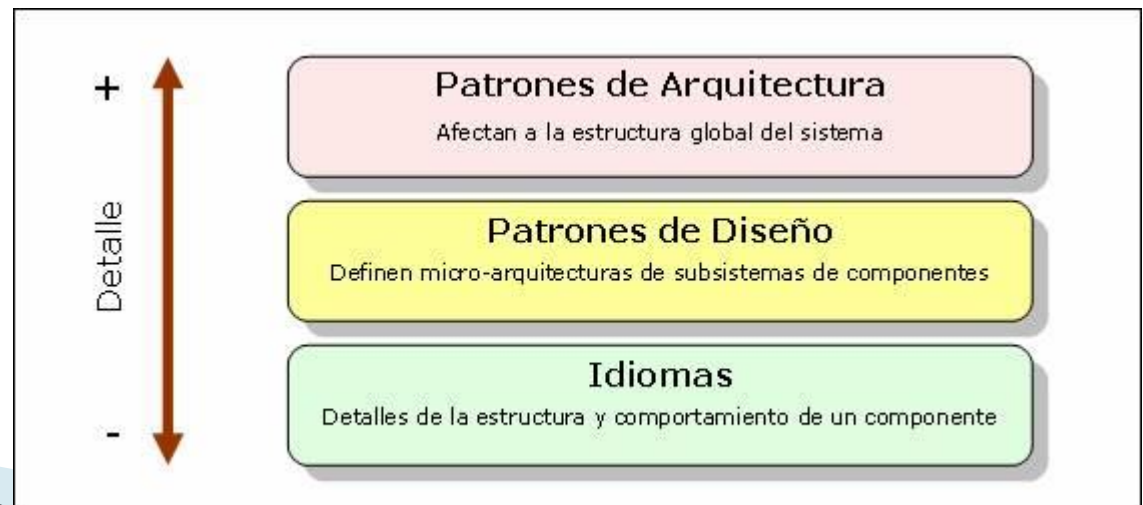
From Mud to Structure	Distributed Systems	Interactive Systems	Adaptable Systems
Layers Pipes and Filters Blackboard	Broker	Model-View-Controller Presentation-Abstraction-Control	Microkernel Reflection

Patrones de Arquitectura

Lenguajes y sistemas de patrones

Un lenguaje de patrones contiene un amplio conjunto de patrones, cuyas combinaciones conforman su gramática. Según Alexander, un lenguaje de patrones provee la misma expresividad que un lenguaje natural, pero aplicada a la construcción.

Un sistema de patrones mantiene unidos a sus patrones. Describe la forma en que se conectan y cómo se complementan entre ellos. Un sistema de patrones facilita el uso eficaz de los patrones en el desarrollo de software. Por lo tanto, podemos decir que un sistema de patrones es una colección de patrones, junto con las guías para su implementación, combinación y uso práctico en el desarrollo de software.



Pattern of Enterprise Application Architecture

En PEAA se definen las siguientes categorías de patrones:

- ▶ **Layering:** Patrones para dividir un sistema en capas.
- ▶ **Organización de la lógica del dominio:** Formas de organizar los objetos del dominio.
- ▶ **Mapping to Relational Databases:** Se relaciona con la comunicación entre la lógica del dominio y los repositorios de datos. Incluye el mapeo entre modelos de objetos y bases de datos relacionales. En la actualidad, se consume mucho tiempo de desarrollo en la realización de estas tareas debido a las diferencias de impedancia entre SQL y los lenguajes orientados a objetos tales como C#, C++, Java, etc.
- ▶ **Presentación Web:** La presentación Web es uno de los desafíos que han tenido que sortear en los últimos años las aplicaciones empresariales. Los clientes delgados Web proveen muchas ventajas, siendo una de las principales la facilidad de distribución (no es necesario instalar software en los equipos cliente). Esta categoría incluye una serie de patrones para gestionar la presentación Web.
- ▶ **Concurrencia:** Manejo de la concurrencia. Las aplicaciones actuales basadas en tecnologías Web tienen grandes necesidades de gestión de la concurrencia.
- ▶ **Estado de sesión:** Patrones para el manejo de la sesión en servidores Web.
- ▶ **Estrategias de Distribución:** Distribución de objetos en múltiples emplazamientos, basada en conocimientos empíricos en clientes.

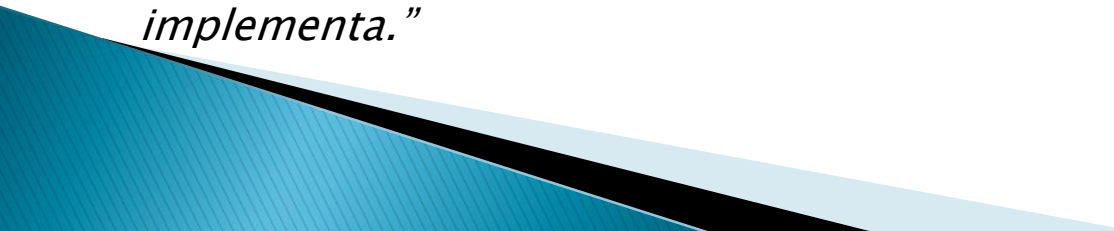
Antipatrones

Los antipatrones son soluciones negativas que presentan más problemas que los que solucionan. Son una extensión natural a los patrones de diseño. Comprender los antipatrones provee el conocimiento para intentar evitarlos o recuperarse de ellos. El estudio de los antipatrones permite conocer los errores más comunes relacionados con la industria del software.

Los antipatrones se documentan con cierto cinismo, lo cual los hace bastante graciosos y fáciles de recordar. Los nombres siempre aluden al problema que tratan con humor. Un buen antipatrón explica por qué la solución original parece atractiva, por qué se vuelve negativa y cómo recuperarse de los problemas que ésta genera.

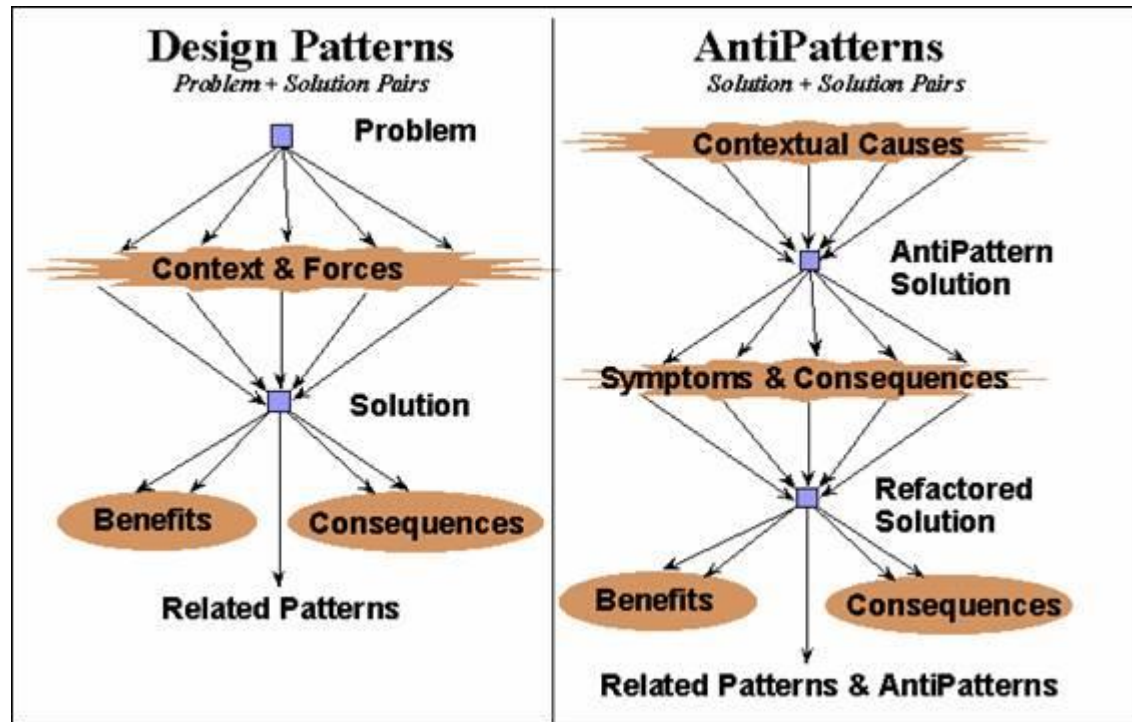
Según James Coplien, un antipatrón es...

“...algo que se ve como una buena idea, pero que falla malamente cuando se la implementa.”

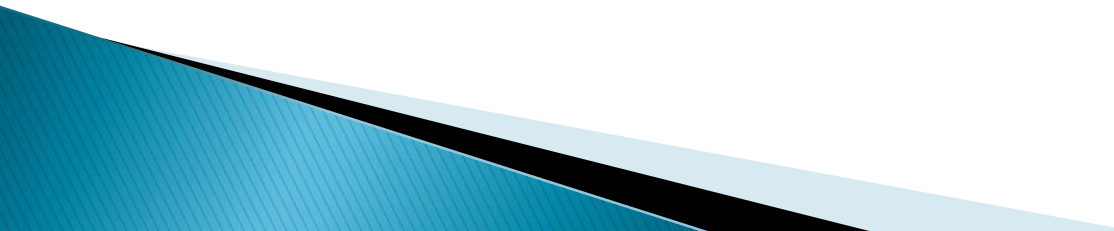


Antipatrones

Un antipatrón (antipattern, en inglés) es una forma que describe una solución común a un problema que genera decididamente consecuencias negativas.



Antipatrones – Características

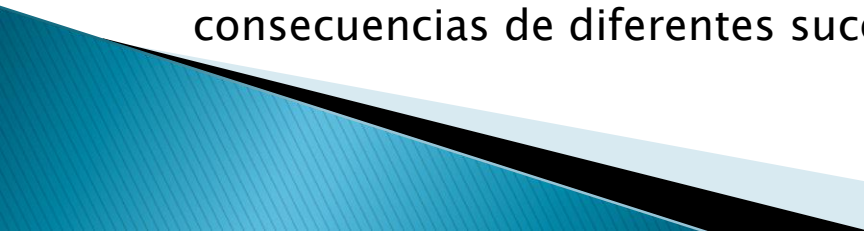
- ▶ Son un método eficiente para vincular una situación general a una clase de solución específica.
 - ▶ Proveen experiencia del mundo real para reconocer problemas recurrentes en la industria del software, ofreciendo también una solución para sus implicaciones más comunes.
 - ▶ Establecen un vocabulario común para identificar problemas y discutir soluciones.
 - ▶ Soportan la resolución holística de conflictos, utilizando recursos organizacionales a diferentes niveles.
- 

Antipatrones – Categorías

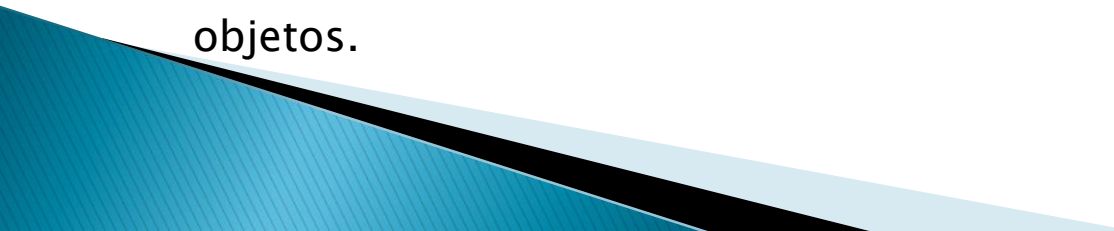
- ▶ **Desarrollo de Software:** Se centran en problemas asociados al desarrollo de software a nivel de aplicación.
- ▶ **Arquitectura de Software:** Se centran en la estructura de las aplicaciones y componentes a nivel de sistema y empresa.
- ▶ **Gestión de Proyectos de Software:** En la ingeniería del software, más de la mitad del trabajo consiste en comunicación entre personas y resolver problemas relacionados con éstas. Los antipatrones de gestión de proyectos de software identifican algunos de los escenarios clave donde estos temas son destructivos para el proceso de software.

Desarrollo de software	Arquitectura	Gestión
<ul style="list-style-type: none">• The Blob• Lava Flow• Functional Decomposition• Poltergeists• Golden Hammer• Spaghetti Code• Cut and Paste Programming <p>Mini antipatterns</p> <ul style="list-style-type: none">• Continuous Obsolescence• Ambiguous Viewpoint• Boat Anchor• Dead End• Input Kludge• Walking through a Minefield• Mushroom Management	<ul style="list-style-type: none">• Stovepipe Enterprise• Vendor Lock-in• Architecture by Implication• Design by Committee• Reinvent the Wheel <p>Mini antipatterns</p> <ul style="list-style-type: none">• Autogenerated Stovepipe• Jumble• Cover your Assets• Wolf Ticket• Warm Bodies• Swiss Army Knife• The Grand Old Duke of York	<ul style="list-style-type: none">• Analysis Paralysis• Death by Planning• Corncob• Irrational Management• Project Missmanagement <p>Mini antipatterns</p> <ul style="list-style-type: none">• Blowhard Jamboree• Viewgraph Engineering• Fear of Success• Intellectual Violence• Smoke and Mirrors• Throw it over the wall• Fire Drill• The Feud• E-Mail is Dangerous


Antipatrones – Ejemplos

- ▶ **Base de datos como comunicador de procesos (database as an IPC):** Usar una base de datos para comunicar procesos en uno o varios ordenadores, cuando la comunicación entre procesos directa es más adecuada.
 - ▶ **Blob:** Objeto todopoderoso.
 - ▶ **BOMQ (Batch Over MQ):** Abuso en el empleo de integración basada en mensajes en tiempo real para transferencias esporádicas de gran tamaño en segundo plano.
 - ▶ **Clase Gorda:** Dotar a una clase con demasiados atributos y/o métodos, haciéndola responsable de la mayoría de la lógica de negocio.
 - ▶ **Botón mágico (magic pushbutton):** Tender, desarrollando interfaces, a programar la lógica de negocio en los métodos de interacción, implementando los resultados de las acciones del usuario en términos no suficientemente abstractos.
 - ▶ **Carrera de obstáculos (race hazard):** Incapacidad de prever las consecuencias de diferentes sucesiones de eventos.
- 

Antipatrones – Ejemplos

- ▶ **Entrada chapuza (input kludge):** No especificar e implementar el manejo de entradas inválidas.
 - ▶ **Fábrica de combustible (gas factory):** Diseñar de manera innecesariamente compleja.
 - ▶ **Gran bola de lodo (big ball of mud):** Construir un sistema sin estructura definida.
 - ▶ **Interfaz inflada (interface bloat):** Pretender que una interfaz sea tan potente que resulta extremadamente difícil de implementar.
 - ▶ **Inversión de abstracción (abstraction inversion):** No exponer las funcionalidades implementadas que los usuarios necesitan, forzando a que se reimplementen a más alto nivel.
 - ▶ **Punto de vista ambiguo (ambiguous viewpoint):** Presentar un modelo sin concretar ciertos aspectos, postergando así decisiones conflictivas.
 - ▶ **Re-dependencia (re-coupling):** Introducir dependencias innecesarias entre objetos.
- 

Antipatrones – Ejemplos

- ▶ **Sistema de cañerías de calefacción (stovepipe system):** Construir un sistema difícilmente mantenible, ensamblando componentes poco relacionados.
 - ▶ **Acoplamiento secuencial (sequential coupling):** Construir una clase que necesita que sus métodos se invoquen en un orden determinado.
 - ▶ **BaseBean:** Heredar funcionalidad de una clase utilidad en lugar de delegar en ella.
 - ▶ **Fallo de clase vacía (empty subclass failure):** Crear una clase que no supera el test de la subclase vacía, es decir, que se comporta de manera diferente cuando se invoca desde una subclase que no añade modificación alguna.
 - ▶ **Llamar a super (callsuper):** Obligar a las subclases a llamar a un método de la superclase que ha sido sobrescrito.
 - ▶ **Modelo de dominio anémico (anemic domain model):** Usar un modelo del dominio sin ninguna lógica de negocio. Esto no es un enfoque orientado a objetos porque cada objeto debería tener tanto propiedades como comportamiento asociado.
- 

Antipatrones – Ejemplos

- ▶ **Objeto sumidero (object cesspool):** Reutilizar objetos no adecuados realmente para el fin que se persigue.
- ▶ **Objeto todopoderoso (god object):** Concentrar demasiada funcionalidad en una única parte del diseño (clase).
- ▶ **Poltergeist (informática):** Emplear objetos cuyo único propósito es pasar la información a terceros objetos.
- ▶ **Problema del círculo-elipse (circle-ellipse problem):** Crear variables de tipo pensando en los valores de posibles subtipos.
- ▶ **Problema del yoyó (yo-yo problem):** Construir estructuras (por ejemplo, de herencia) que son difíciles de comprender debido a su excesiva fragmentación.
- ▶ **Singletonitis:** Abuso de la utilización del patrón singleton.
- ▶ **YAFL (yet another layer, y otra capa más):** Añadir capas innecesarias a un programa, biblioteca o framework. Esta tendencia se extendió bastante después de que se publicase el primer libro sobre patrones.

Lecturas

- ▶ <http://msdn.microsoft.com/es-es/library/bb972251.aspx>
- ▶ <http://msdn.microsoft.com/es-es/library/bb972242.aspx>

Gracias