

Patrón de diseño

Los **patrones de diseño** son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su **efectividad** resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser **reutilizable**, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Breve reseña histórica

En 1979 el arquitecto Christopher Alexander aportó al mundo de la arquitectura el libro *The Timeless Way of Building*; en él proponía el aprendizaje y uso de una serie de patrones para la construcción de edificios de una mayor calidad.

En palabras de este autor, "Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez."

Los patrones que Christopher Alexander y sus colegas definieron, publicados en un volumen denominado *A Pattern Language*, son un intento de formalizar y plasmar de una forma práctica generaciones de conocimiento arquitectónico. Los patrones no son principios abstractos que requieran su redescubrimiento para obtener una aplicación satisfactoria, ni son específicos a una situación particular o cultural; son algo intermedio. Un patrón define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones.

Más tarde, en 1987, Ward Cunningham y Kent Beck usaron varias ideas de Alexander para desarrollar cinco patrones de interacción hombre-ordenador (HCI) y publicaron un artículo en OOPSLA-87 titulado **Using Pattern Languages for OO Programs**.

No obstante, no fue hasta principios de la década de 1990 cuando los patrones de diseño tuvieron un gran éxito en el mundo de la informática a partir de la publicación del libro *Design Patterns* escrito por el grupo Gang of Four (**GoF**) compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, en el que se recogían 23 patrones de diseño comunes.

Objetivos de los patrones

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. "Abusar o forzar el uso de los patrones puede ser un error".

Categorías de patrones

Según la escala o nivel de abstracción:

- **Patrones de arquitectura:** Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- **Patrones de diseño:** Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- **Dialectos:** Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Además, también es importante reseñar el concepto de "antipatrón de diseño", que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los antipatrones es dar a conocer los problemas que acarrearán ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores.

Además de los patrones ya vistos actualmente existen otros patrones como el siguiente:

- **Interacción:** Son patrones que nos permiten el diseño de interfaces web.

Estructuras o plantillas de patrones

Para describir un patrón se usan plantillas más o menos estandarizadas, de forma que se expresen uniformemente y puedan constituir efectivamente un medio de comunicación uniforme entre diseñadores. Varios autores eminentes en esta área han propuesto plantillas ligeramente distintas, si bien la mayoría definen los mismos conceptos básicos.

La plantilla más común es la utilizada precisamente por el GoF y consta de los siguientes apartados:

- **Nombre del patrón:** nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés).
- **Clasificación del patrón:** creacional, estructural o de comportamiento.
- **Intención:** ¿Qué problema pretende resolver el patrón?
- **También conocido como:** Otros nombres de uso común para el patrón.
- **Motivación:** Escenario de ejemplo para la aplicación del patrón.
- **Aplicabilidad:** Usos comunes y criterios de aplicabilidad del patrón.
- **Estructura:** Diagramas de clases oportunos para describir las clases que intervienen en el patrón.
- **Participantes:** Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón.
- **Colaboraciones:** Explicación de las interrelaciones que se dan entre los participantes.
- **Consecuencias:** Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.
- **Implementación:** Técnicas o comentarios oportunos de cara a la implementación del patrón.
- **Código de ejemplo:** Código fuente ejemplo de implementación del patrón.
- **Usos conocidos:** Ejemplos de sistemas reales que usan el patrón.
- **Patrones relacionados:** Referencias cruzadas con otros patrones.

Relación de principales patrones GoF (*Gang Of Four*)

Patrones creacionales

- **Object Pool (Conjunto de Objetos):** (No pertenece a los patrones especificados por GoF) Se obtienen objetos nuevos a través de la clonación. Utilizado cuando el costo de crear una clase es mayor que el de clonarla. Especialmente con objetos muy complejos. Se especifica un tipo de objeto a crear y se utiliza una interfaz del prototipo para crear un nuevo objeto por clonación. El proceso de clonación se inicia instanciando un tipo de objeto de la clase que queremos clonar.
- **Abstract Factory (Fábrica abstracta):** Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

- Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.
- Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.
- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones estructurales

- Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- Bridge (Puente): Desacopla una abstracción de su implementación.
- Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase.
- Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- Proxy: Mantiene un representante de un objeto.

Patrones de comportamiento

- Chain of Responsibility (Cadena de responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
 - Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
 - Interpreter (Intérprete): Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
 - Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
 - Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
 - Memento (Recuerdo): Permite volver a estados anteriores del sistema.
 - Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.
 - State (Estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.
 - Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.
 - Template Method (Método plantilla): Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
 - Visitor (Visitante): Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.
-

Patrones de interacción

El primer intento por aplicar este concepto en el diseño de las interfaces de usuario se dio por Ward Cunningham y Kent Beck quienes adaptaron la propuesta de C. Alexander y crearon cinco patrones de interfaz: *Window per task*, *Few panes*, *Standard panes*, *Nouns and verbs*, y *Short Menu*. En años más recientes investigadores como Martin Van Welie, Jennifer Tidwell, Jaime Muñoz han desarrollado colecciones de patrones de interacción para la World Wide Web. En dichas colecciones captan la experiencia de programadores y diseñadores expertos en el desarrollo de interfaces usables y condensan esta experiencia en una serie de guías o recomendaciones, que puedan ser usadas por los desarrolladores novatos con el propósito de que en poco tiempo adquieran la habilidad de diseñar interfaces que incidan en la satisfacción de los usuarios. Los patrones de interacción buscan la reutilización de interfaces eficaces y un manejo óptimo de los recursos de las páginas web, haciendo más eficaz el consumo de tiempo en el diseño del sitio web y permitiendo a los programadores novatos adquirir más experiencia.

Aplicación en ámbitos concretos

Además de su aplicación directa en la construcción de software en general, y derivado precisamente del gran éxito que han tenido, los patrones de diseño han sido aplicados a múltiples ámbitos concretos produciéndose "lenguajes de patrones" y extensos "catálogos" de la mano de diversos autores.

En particular son notorios los esfuerzos en los siguientes ámbitos:

- Patrones de interfaces de usuario, esto es, aquellos que intentan definir las mejores formas de construir interfaces hombre-máquina (véase Interacción persona-computador, Interfaz gráfica de usuario).
- Patrones para la construcción de sistemas empresariales, en donde se requieren especiales esfuerzos en infraestructuras de software y un nivel de abstracción importante para maximizar factores como la escalabilidad o el mantenimiento del sistema.
- Patrones para la integración de sistemas (véase Integración de aplicaciones empresariales), es decir, para la intercomunicación y coordinación de sistemas heterogéneos.
- Patrones de flujos de trabajo, esto es para la definición, construcción e integración de sistemas abstractos de gestión de flujos de trabajo y procesos con sistemas empresariales. Véase también BPM.


Bibliografía

- *Design Patterns. Elements of Reusable Object-Oriented Software* - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four)
 - *A System of Patterns* - Buschmann, Meunier, Rohnert, Sommerlad, Stal - Wiley
 - *UML y Patrones. Introducción al análisis y diseño orientado a objetos* - Larman - Prentice Hall
 - *AntiPatterns. Refactoring Software, Architectures and Projects in Crisis* - Brown, Malveau, McCormick Mowbray - Wiley
 - *Patterns in Java* - Mark Grand - Wiley
 - *EJB Design Patterns* - Floyd Marinescu - Wiley
 - *Head First Design Patterns* - O'Reilly
-

Véase también

- Lenguaje de patrón
- Antipatrón de diseño
- Grasp

Enlaces externos

-  Wikimedia Commons alberga contenido multimedia sobre **Patrón de diseño** Commons
- Patrones de diseño en PHP ^[1]
- Core J2EE Patterns ^[2]
- Explicación de algunos patrones de diseño ^[3]
- Hillside Group Web Pages ^[4]
- Brad Appleton's Software Patterns Links ^[5]
- Portland Pattern Repository ^[6]
- Object::PerlDesignPatterns ^[7] Módulo Perl en CPAN con métodos específicos de implementación de Patrones de Diseño en lenguaje Perl (en inglés)
- Patrones de diseño en [[Yahoo! ^[8]]]
- Qué son los Patrones en el Desarrollo de Software? ^[9] Desde el punto de vista de un desarrollador .NET

Referencias

- [1] <http://desarrolladorsenior.blogspot.com/search/label/patrones%20de%20dise%C3%B1o>
- [2] <http://corej2eepatterns.com>
- [3] <http://software.guisho.com/archives/category/patrones-de-diseno>
- [4] <http://www.hillside.net>
- [5] http://www.enteract.com/_bradapp/links/sw-pats.html
- [6] <http://c2.com/ppr/index.html>
- [7] <http://search.cpan.org/perldoc?Object::PerlDesignPatterns>
- [8] <http://developer.yahoo.com/ypatterns>
- [9] <http://www.joedev.com/index.php/2010/09/que-son-los-patrones-en-el-desarrollo-de-software.html>

Fuentes y contribuyentes del artículo

Patrón de diseño *Fuente:* <http://es.wikipedia.org/w/index.php?oldid=47707969> *Contribuyentes:* Alvaro qc, Arley triana, Ascánder, Biasoli, Bngs, Cad, Caíser, Camilo, Castoluis, Chewie, Corrector1, Dalobuca, Davduke, David0811, Davius, Dejoto, Developer, Diego.souto, Diegusjaimes, Diosa, Dodo, Durero, Edgarhíran, Eidansoft, Fernandomirandamuro, Fortran, Framontb, Gefush, GermanX, Guanxito, Guevonaso, Guishogt, HUB, Hari Seldon, Henry99923, Hprmedina, Jarke, Jjafjjaf, Joaquín Ferrero, JoeDev, Jonalexba, JorgeGG, Laurita1985, LordT, Maldoror, ManuelGR, Matdrodes, Moustique, Muro de Aguas, Mutari, Pacoqueen, Penyaskito, PoLuX124, Porao, Pvent, Ravave, Ricardofs, Rosita fraguel, Rsg, SeriketZu, Tano4595, The Mad Philologist, Vicpulido, Wasilio, Yeza, Zerosxt, 168 ediciones anónimas

Fuentes de imagen, Licencias y contribuyentes

Archivo:Commons-logo.svg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Commons-logo.svg> *Licencia:* logo *Contribuyentes:* SVG version was created by User:Grunt and cleaned up by 3247, based on the earlier PNG version, created by Reidab.

Licencia

Creative Commons Attribution-Share Alike 3.0 Unported
<http://creativecommons.org/licenses/by-sa/3.0/>