Patrones en el *e-learning*

DOCUMENTO BASE

El concepto de patrón en el *e-learning* permite resumir y comunicar la experiencia acumulada en la enseñanza-aprendizaje por medios telemáticos. Un patrón puede entenderse como una plantilla, guía, directriz o norma de diseño. Los patrones pueden considerarse desde la perspectiva pedagógica o de diseño instruccional, o bien desde la perspectiva de los lenguajes y las técnicas informáticas que los soportan. Un patrón permite adquirir "buenas prácticas" y servir como referencia. La compilación sistemática de estos patrones permite construir obras o bases de datos de referencia a las que los profesionales o investigadores pueden acudir para sus fines específicos.

¿Qué son los patrones?

Los patrones de diseño se originan en la obra del arquitecto Christopher Alexander[1]. Que se han adoptado en la ingeniería de software y que ahora se aplican a otras esferas, como al diseño de la educación. Un patrón *describe un problema que ocurre una y otra vez en nuestro entorno y, a continuación, se describe el núcleo de la solución a ese problema, de tal manera que usted puede utilizar esta solución un millón de veces más, sin tener que hacerlo de la misma manera dos veces* "(Alexander et al., 1977, A Pattern Language, Oxford University Press, px).

¿Qué son las pautas(patrones) pedagógicos?[2]

En primer lugar tendremos que decidir qué término es mejor *pautas* o *patrones*.

Los patrones (pautas) están diseñados para capturar las mejores prácticas en un determinado dominio. Los patrones Pedagógicos tratan de capturar el conocimiento de expertos en la práctica de la enseñanza y el aprendizaje. La intención es captar la esencia de la práctica en una forma compacta (resumida) que puede ser fácilmente comunicada a los que la necesitan de los conocimientos. También se trata de la presentación de esta información en una forma accesible y coherente puede significar la diferencia entre cada nuevo instructor tener que reaprender lo que se conoce por profesores de alto nivel y fácil transferencia de conocimiento de la enseñanza dentro de la comunidad.

[1] http://downlode.org/Etext/Patterns/

A Pattern Language
Summary of a book by Christopher Alexander, Sara Ishikawa, Murray Silverstein, with Max Jacobson, Ingrid Fiksdahl-King and Shlomo Angel. Published by Oxford University Press.
The original book contains much essential detail behind each of the following patterns and is recommended reading.

[2] http://www.pedagogicalpatterns.org/

En esencia un modelo resuelve un problema. Este problema debe ser un problema del tipo de los que se repiten en distintos contextos. En el ámbito de la enseñanza tenemos muchos problemas de este tipo como son los que tienen que ver con la motivación de los estudiantes, la elección de los materiales y la secuencia de los contenidos, la evaluación de los estudiantes, y otras cosas por el estilo.

Estos problemas se repiten y lo hacen de forma ligeramente diferente cada vez. Cada vez que aparece un problema de este tipo lleva aparejadas consideraciones que hay que tener en cuenta a la hora de tomar la decisión en la selección de la solución. Son los datos que influyen a los expertos a tomar una u otra solución. Son las consideraciones que se deberían tener en cuenta y que influyen en nuestra elección de la solución. Estas consideraciones nos empujan hacia o nos alejan de cualquier solución al problema.

Un patrón se supone pues que presenta un problema y una solución. O con el criterio de la solución. El problema, junto con los criterios que deben aplicarse para hacer que la solución sea la más beneficiosa para el problema planteado.

Lenguaje de patrón[3]

En diseño, un **lenguaje de patrón** es un método estructurado para describir una serie de buenas prácticas de diseño en un área particular. Se caracteriza por:

1. Descubrir y nombrar los problemas más comunes en el campo de interés.
2. Describir las características principales de las soluciones efectivas para llegar al objetivo marcado.
3. Ayudar al diseñador a moverse de un problema a otro de una forma lógica.
4. Permitir diferentes caminos en un mismo proceso de diseño.

Los lenguajes de patrón se utilizan para formalizar los valores de decisiones cuya efectividad resulta obvia a través de la experiencia, pero que es difícil de documentar y pasar a los aprendices. También son herramientas útiles a la hora de estructurar el conocimiento y comprender sistemas complejos sin caer en la simplificación extrema. Estos procesos incluyen la organización de personas o grupos que tienen que tomar decisiones complejas, y revelan cómo interactúan las diferentes funciones como parte del total.

[3] De Wikipedia, la enciclopedia libre

Tabla de contenidos

Origen

Christopher Alexander acuño el término *lenguaje de patrón*. Lo usó para referirse a los problemas normales del diseño arquitectónico y civil. Su uso iba desde la forma de estructurar una ciudad a como debían disponerse las ventanas en una habitación. La idea se popularizó gracias a su libro *A Pattern Language*.

El libro de Alexander *The Timeless Way of Building* describe qué significa para él *lenguaje de patrón* y cómo se aplica al diseño de edificios y ciudades. Sin embargo, este sistema es aplicable a cualquier otro campo del diseño.

Muchos patrones forman un lenguaje *[editar]*

Así como las palabras deben tener una relación gramática y semántica entre ellas para crear un lenguaje oral útil, los patrones de diseño deben estar relacionados los unos con los otros para poder formar un lenguaje de patrones. En el trabajo de Alexander está implícita la idea de que los patrones deben estar organizados en estructuras lógicas o estructuras intuitivas. La estructura (jerárquica, iterativa, etc) puede variar, dependiendo del tema. Cada patrón debe indicar su relación con otros patrones y con el lenguaje en sí.

Uso

Alexander animó a la gente que usaba su sistema a expandirlo con patrones que los usuarios creasen. Para hacer esto posible, sus libros no se centran de forma estricta en arquitectura o en ingeniería civil. En los libros también habla del método general de los lenguajes de patrón. De esta forma, sus métodos se han usado para documentar el conocimiento experimental en diversos campos, como los patrones informáticos que se usan en ingeniería de software, patrones de diseño de interacción en Interacción del hombre con la computadora y patrones pedagógicos en educación. Sin embargo, las especificaciones de Alexander sobre como usar un lenguaje de patrón así como cómo crear uno nuevo siguen ejerciendo su autoridad, y sus libros son la referencia para expertos en campos muy diversos.

Es importante señalar que notaciones como UML o la colección de símbolos de los diagramas de flujo no son lenguajes de patrón (aunque podrían utilizarse para expresar un lenguaje de patrón). Una receta o cualquier otro conjunto de secuencias a seguir, con sólo un posible camino desde el inicio hasta el final, tampoco es un lenguaje de patrón. Sin embargo, el proceso de diseño de una nueva receta puede beneficiarse del uso de los lenguajes de patrón.

Cómo documentar un patrón [editar]

Siguiendo las enseñanzas de Alexander, un patrón individual debe describirse en 3 partes:

1. "Contexto" - ¿Bajo qué condiciones resolverá esta solución el problema?
2. "Sistema de fuerzas" - Se puede considerar como el problema o el objetivo
3. "Solución" - Una configuración que pone las fuerzas en equilibrio o resuelve el problema presentado

Contexto -> Sistema de fuerzas -> Solución

Así, una entrada en un lenguaje de patrón debería tener un nombre sencillo, una descripción concisa del problema, una solución clara, y suficiente información para ayudar al lector a entender cuando aplicar esta solución. También debería mostrar que patrones deben tenerse en consideración de ante mano, y que patrones se deberían considerar más adelante.

Ejemplo sencillo de un patrón [editar]

Nombre: GalletaChocolateRelación

Contexto: Cocinando galletas de chocolate para tu familia y amigos

Considera estos patrones anteriormente: AzucarRelación, HarinaRelación, HuevoRelación

Problema: Determinar la relación optima entre trozos de chocolate y masa de galleta

Solución: Observe que la mayoría de la gente considera los trozos de chocolate como lo mejor de la galleta. También observe que demasiado chocolate haría que la galleta no se mantuviese junta, disminuyendo su atractivo. Como se están cocinando pocas galletas, el coste no es un gran problema. Así, utilice tanto chocolate como pueda manteniendo la galleta compacta.

A considerar ahora: NuecesRelación o TiempoCocción o MetodoCongelación.

Lenguajes de patrón y wikis

Ward Cunningham creó la Wiki original, el Portland Pattern Repository, como una forma de expresar los lenguajes de patrón de forma efectiva. En la actualidad tiene cientos de patrones de diferentes campos, incluyendo muchos sobre programación extrema.

Trucos para crear un lenguaje de patrón

Los siguientes pasos asumen el uso de una aplicación informática de hipertexto, como una wiki.

- Piense sobre la situación como una jerarquía de ideas, como un fractal, desde lo general al detalle.
- Ponga cada idea en una página diferente, nombrándola con el título que describe la idea en pocas palabras.
- En un índice, ordena las ideas desde las más grandes a los detalles. Esto permitirá al lenguaje de patrón ser lineal y facilmente imprimible, si es necesario. Este paso incrementa de forma amplia la usabilidad del sistema si (casi) todas las páginas se mencionan en la gran lista.
- Para cada patrón, escriba una descripción del problema, la solución y un ejemplo. Mientras lo escribe, conéctelo con los patrones con los que se relacione
- Los gráficos son de gran ayuda.
- Si hay varias personas editando el lenguaje de patrón, firme su trabajo. Además de la razón más obvia, permite conseguir un lista de las páginas usando enlace recíprocos en cada página.
- Muestra referencias donde sea apropiado
- Ponga un enlace al índice al final de cada patrón
- Probablemente necesita una ventana para escribir su página, y al menos una más para saltar atrás y adelante con las referencias.

Enlaces externos

- The Portland Pattern Repository
- Tips For Writing Pattern Languages de Ward Cunningham.
- architypes.net - proyecto de lenguajes de patrón comunitarios de arquitectura
- Liberating Voices! Proyecto de lenguajes de patrón
- El sitio de Christopher Alexander patternlanguage.com
- Ensayo sobre los lenguajes de patrón y su uso en el diseño urbano
- Conservation Economy Pattern Language - ejemplo de uso de lenguaje de patrón para organizar grupos y recursos en una situación compleja (movimiento sostenible)

Patrón de diseño[4]

Los **patrones de diseño** (*design patterns*) 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 reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Tabla de contenidos

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

[4] De Wikipedia, la enciclopedia libre

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 cultura, 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 los 90's 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 GoF (Gang of Four) compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, en el que se recogían 23 patrones diseño comunes.

Objetivos de los patrones *[editar]*

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 siempre, solo 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**: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas software.
- **Patrones de diseño**: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
- **Idiomas**: 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 acarrean 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.

Estructuras o plantillas de patrones [editar]

Para describir un patrón se utilizan 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 este 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 resuelve el patrón?
- **También conocido como**: Otros nombres de uso común para el patrón.
- **Motivación**: Escenario de ejemplo para la aplicación del patrón.
- **Aplicabilidad**: 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 [editar]

Patrones creacionales [editar]

- 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 [editar]

- 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 [editar]

- 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.

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 completos **catálogos** de 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 (HCI, GUI).

- Patrones para la construcción de sistemas empresariales, en donde se requieren especiales esfuerzos en infraestructuras software y un nivel de abstracción importante para maximizar factores como la escalabilidad o la mantenibilidad del sistema.

- Patrones para la integración de sistemas (EAI), es decir, para la intercomunicación y coordinación de sistemas heterogéneos.

- Patrones de workflow, 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 básica de referencia

- **Design Patterns. Elements of Reusable Object-Oriented Software** - Gamma, Helm, Johnson, Vlissides - Addison Wesley
- **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
- **PATRONES DE DISEÑO: ELEMENTOS DE SOFTWARE ORIENTADO A OBJETOS REUTILIZABLES**