Anti-Patrones

Anti-patrón

Un antipatrón de diseño es un patrón de diseño que conduce a una mala solución. Buscan evitar que los programadores tomen malas decisiones, partiendo de documentación disponible en lugar de simplemente la intuición.

Historia

El término nace con el mencionado libro Design Patterns de GOF, que básicamente son los patrones que vimos a lo largo de este curso. Los autores bautizaron dichas soluciones con el nombre de "patrones de diseño" por analogía con el mismo término, usado en arquitectura.

El libro Anti-Patterns (de William Brown, Raphael Malveau, Skip McCormick, Tom Mowbray y Scott Thomas) describe los antipatrones como la contrapartida natural al estudio de los patrones de diseño. El término fue utilizado por primera vez en 1995, por ello es que los antipatrones no se mencionan en el libro original de Design Patterns, puesto que éste es anterior.

Propósito

Los anti-patrones son ejemplos bien documentados de malas soluciones para problemas. El estudio formal de errores que se repiten permite que el desarrollador pueda reconocerlos más fácilmente y, por ello, podrá encaminarse hacia una mejor solución. Los desarrolladores deben evitar los antipatrones siempre que sea posible.

Utilización

Dado que el término antipatrón es muy amplio, suele ser utilizado en diversos contextos:

Antipatrones de desarrollo de software

- Antipatrones de gestión
- Antipatrones de gestión de proyectos
- Antipatrones generales de diseño de software
- Antipatrones de diseño orientado a objetos
- Antipatrones de programación
- Antipatrones metodológicos
- Antipatrones de gestión de la configuración

Antipatrones organizacionales

Avance del alcance (scope creep): Permitir que el alcance de un proyecto crezca sin el control adecuado.

Bloqueo del vendedor (vendor lock-in): Construir un sistema que dependa en exceso de un componente proporcionado por un tercero.

Diseño de comité (design by committee): Contar con muchas opiniones sobre un diseño, pero adolecer de falta de una visión unificada.
Escalada del compromiso (escalation of commitment): No ser capaz de revocar una decisión a la vista de que no ha sido acertada.
Funcionalitis acechante (creeping featuritis): Añadir nuevas funcionalidades al sistema en detrimento de su calidad.
Gestión basada en números (management by numbers): Prestar demasiada atención a criterios de gestión cuantitativos, cuando no son esenciales o difíciles de cumplir.
Gestión champiñón (mushroom management): Tratar a los empleados sin miramientos, sin informarles de las decisiones que les afectan (manteniéndolos cubiertos y en la oscuridad, como los champiñones).
Gestión por que lo digo yo (management by perkele): Aplicar una gestión autoritaria con tolerancia nula ante las disensiones.
Migración de coste (cost migration): Trasladar los gastos de un proyecto a un departamento o socio de negocio vulnerable.
Obsolescencia continua (continuous obsolescence): Destinar desproporcionados esfuerzos a adaptar un sistema a nuevos entornos.
Organización de cuerda de violín (violin string organization): Mantener una organización afinada y en buen estado, pero sin ninguna flexibilidad.
Parálisis del análisis (analysis paralysis): Dedicar esfuerzos desproporcionados a la fase de análisis de un proyecto, eternizando el proceso de diseño iterando sobre la búsqueda de mejores soluciones o variantes.
Peligro moral (moral hazard): Aislar a quien ha tomado una decisión a raíz de las consecuencias de la misma.
Sistema de cañerías (stovepipe): Tener una organización estructurada de manera que favorece el flujo de información vertical, pero inhibe la comunicación horizontal.
Te lo dije (I told you so): Permitir que la atención se centre en que la desoída advertencia de un experto se ha demostrado justificada.
Vaca del dinero (cash cow): Pecar de autocomplacencia frente a nuevos productos por disponer de un producto legacy muy lucrativo.

Otros Patrones

Introducción

Dado el éxito que tuvo el famoso libre de Gof, comenzó una serie de epidemia de patrones. Hoy en día hay patrones de todo tipo que busca explicar las mejores prácticas de cada caso.
Los más conocidos son los siguientes.

Patrones de base de datos

Buscan abstraer y encapsular todos los accesos a la fuente de datos. El más conocido es el DAO. Técnicamente el patrón DAO (Data Access Object) pertenece a los patrones JEE, aunque se centra específicamente en la capa de persistencia de los datos.

Patrones de Arquitectura

Se centran en la construcción del software, sobretodo en la ingeniería del mismo. El más conocido es el MVC, que busca separar el diseño de la lógica del negocio.

Patrones JEE

Es un catálogo de patrones para usar en la tecnología JEE, es decir, en la parte empresarial que ofrece Java. Divide los patrones en 3 categorías: Capa de Presentación, de Negocios y de Integración. En esta última capa se encuentra el DAO.