Técnico en

DESARROLLO DE SOFTWARE

Diseño e Implementación del Software

Instituto Von Neumann

Unidad III

Diseño e Implementación del Software

< Anti-patrones de diseño de software >

Un Anti-patrón es una forma describe una solución común a un problema que genera consecuencias decididamente negativas, es decir, estas acciones pueden ser el resultado de que un gerente o desarrollador no sepa nada mejor, no tenga suficiente conocimiento o experiencia para resolver un tipo particular de problema, o haya aplicado un patrón perfectamente bueno en el contexto equivocado.

Un Anti-Patrón brinda experiencia del mundo real en el reconocimiento de problemas recurrentes en la industria del software y proporciona una solución detallada para los problemas más comunes.

Los Anti-Patrones destacan los problemas más comunes a los que se enfrenta la industria del software y proporciona las herramientas que le permiten reconocer estos problemas y determinar sus causas subyacentes. Además, un Anti-Patrón presenta un plan detallado para revertir estas causas subyacentes e implementar soluciones productivas.

Un Anti-Patrón describe efectivamente las medidas que se pueden tomar en varios niveles para mejorar el desarrollo de aplicaciones, el diseño de sistemas de software y la gestión efectiva de proyectos de software, esto origina tres grandes divisiones:

- Anti-patrones de desarrollo de software
- Anti-patrones de arquitectura de software
- Anti-patrones de la gestión de proyectos de software

1. Anti-patrones de desarrollo de software

Los antipatrones de desarrollo utilizan varios enfoques de refactorización formales e informales. Los siguientes resúmenes proporcionan una descripción general de los antipatrones de desarrollo y se centran en el problema de los antipatrones de desarrollo. Se incluyen descripciones de desarrollo y mini-AntiPatterns.

The Blob

El diseño de estilo procedimental Blob conduce a un objeto con la mayor parte de las responsabilidades, mientras que la mayoría de los otros objetos solo contienen datos o ejecutan procesos simples. La solución incluye refactorizar el diseño para distribuir las responsabilidades de manera más uniforme y aislar el efecto de los cambios.

Obsolescencia continua

La tecnología está cambiando tan rápidamente que los desarrolladores a menudo tienen problemas para mantenerse al día con las versiones actuales de software y encontrar combinaciones de lanzamientos de productos que funcionen juntos. Dado que cada línea de productos comerciales evoluciona a través de nuevos lanzamientos, la situación se está volviendo más difícil de manejar para los desarrolladores. Encontrar versiones compatibles de productos que interoperen con éxito es aún más difícil.

Flujo de lava

El código muerto y la información de diseño olvidada se congelan en un diseño en constante cambio. Esto es análogo a un flujo de lava con glóbulos endurecidos de material rocoso. La solución refactorizada incluye un proceso de gestión de la configuración que elimina el código muerto y evoluciona o refactoriza el diseño para aumentar la calidad.

Punto de vista ambiguo

Los modelos de análisis y diseño orientados a objetos (OOA&D) a menudo se presentan sin aclarar el punto de vista representado por el modelo. Por defecto, los modelos OOA&D denotan un punto de vista de implementación que es potencialmente el menos útil. Los puntos de vista mixtos no permiten la separación fundamental de las interfaces de los detalles de implementación, que es uno de los principales beneficios del paradigma orientado a objetos.

Descomposición funcional

Este Anti-Patrón es el resultado de desarrolladores experimentados no orientados a objetos que diseñan e implementan una aplicación en un lenguaje orientado a objetos. El código resultante se parece a un lenguaje estructural (Pascal, FORTRAN) en la estructura de clases. Puede ser increíblemente complejo ya que los desarrolladores de procedimientos inteligentes idean formas muy "inteligentes" de replicar sus métodos probados en el tiempo en una arquitectura orientada a objetos.

Poltergeists

Los poltergeists son clases con roles muy limitados y ciclos de vida efectivos. A menudo inician procesos para otros objetos. La solución refactorizada incluye una reasignación de responsabilidades a objetos de vida más larga que eliminan a los Poltergeists.

Ancla de barco

Un ancla de barco es una pieza de software o hardware que no tiene ningún propósito útil en el proyecto actual. A menudo, Boat Anchor es una adquisición costosa, lo que hace que la compra sea aún más irónica.

Martillo de oro

Un Golden Hammer es una tecnología o concepto familiar que se aplica obsesivamente a muchos problemas de software creyendo que todos los casos se aplique una propuesta metodológica se van a obtener resultados positivos sin antes analizar el contexto donde se aplican. La solución implica expandir el conocimiento de los desarrolladores a través de grupos de educación, capacitación y estudio de libros para exponer a los desarrolladores a tecnologías y enfoques alternativos y conocer más a fondo el negocio para poder aplicar soluciones a la medida que se requiera.

Callejón sin salida

Se llega a un callejón sin salida al modificar un componente reutilizable si el proveedor ya no mantiene ni respalda el componente modificado. Cuando se realizan estas modificaciones, la carga de soporte se transfiere a los desarrolladores y mantenedores del sistema de aplicaciones. Las mejoras en el componente reutilizable no se

integran fácilmente y los problemas de soporte pueden atribuirse a la modificación.

Código de espagueti

La estructura de software ad hoc (Sin planificación) dificulta la ampliación y optimización del código. La refactorización frecuente del código puede mejorar la estructura del software, admitir el mantenimiento del software y permitir el desarrollo iterativo.

Errores de entrada

El software que no pasa las pruebas de comportamiento sencillas puede ser un ejemplo de un error de entrada, que ocurre cuando se emplean algoritmos ad hoc (Sin planificar) para manejar la entrada del programa, esto ocurre cuando un usuario puede incluir cualquier tipo de valor de entrada que puede incurrir a fallos dentro del sistema.

Caminando por un campo minado

El uso de la tecnología de software actual es similar a caminar por un campo minado de alta tecnología. Se encuentran numerosos errores en los productos de software publicados; de hecho, los expertos estiman que el código fuente original contiene de dos a cinco errores por línea de código.

Programación de cortar y pegar

El código reutilizado mediante la copia de sentencias fuente conduce a importantes problemas de mantenimiento. Las formas alternativas de reutilización, incluida la reutilización de caja negra, reducen los problemas de mantenimiento al tener un código fuente, pruebas y documentación comunes.

Gestión de hongos

Es la gestión de una empresa donde los canales de comunicación entre los gerentes y los empleados no funcionan de manera efectiva, y donde la gerencia 'mantiene a los empleados en la oscuridad' con respecto a las decisiones comerciales que afectan su trabajo y empleo.

2. Anti patrones de arquitectura de software

Los anti patrones de arquitectura se centran en la estructura de aplicaciones y componentes a nivel de sistema y empresarial. Aunque la disciplina de ingeniería de la arquitectura de software es relativamente inmadura, lo que ha sido determinado repetidamente por la investigación y la experiencia del software es la importancia primordial de la arquitectura en el desarrollo de software:

- Una buena arquitectura es un factor crítico en el éxito del desarrollo del sistema.

- El desarrollo de software basado en la arquitectura es el enfoque más efectivo para construir sistemas. Los enfoques basados en arquitectura son superiores a los enfoques basados en requisitos, documentos y metodología. Los proyectos a menudo tienen éxito a pesar de la metodología, no gracias a ella.

La arquitectura de software es un subconjunto de la arquitectura general del sistema, que incluye todos los aspectos de diseño e implementación, incluida la selección de hardware y tecnología. Los principios importantes de la arquitectura incluyen los siguientes:

- La arquitectura proporciona una vista de todo el sistema. Esto distingue a la arquitectura de otros modelos de análisis y diseño que se enfocan en partes de un sistema.

- Una forma efectiva de modelar sistemas completos es a través de múltiples puntos de vista. Los puntos de vista se correlacionan con varios interesados y expertos técnicos en el proceso de desarrollo del sistema.

Los siguientes Anti Patrones se enfocan en algunos problemas y errores comunes en la creación, implementación y administración de la arquitectura.

Tubo de estufa autogenerado

Este Anti-Patrón ocurre cuando se migra un sistema de software existente a una infraestructura distribuida. Un tubo de estufa autogenerado surge al convertir las interfaces de software existentes en interfaces distribuidas. Si se utiliza el mismo diseño para la computación distribuida, surgen varios problemas.

Por ejemplo, las interfaces existentes pueden estar utilizando operaciones de grano fino para transferir información que puede ser ineficiente en un entorno distribuido. Las interfaces preexistentes suelen ser específicas de la implementación y provocarán interdependencias entre subsistemas cuando se utilicen en un sistema distribuido a mayor escala.

Las operaciones locales a menudo hacen varias suposiciones sobre la ubicación, incluido el espacio de direcciones y el acceso al sistema de archivos local. Puede surgir un exceso de complejidad cuando se exponen varias interfaces existentes en un sistema distribuido a mayor escala.

Tubos de estufa empresariales

Un sistema de tubos de estufa se caracteriza por una estructura de software que inhibe el cambio. La solución refactorizada describe cómo abstraer subsistemas y componentes para lograr una estructura de sistema mejorada. Un tubo de estufa empresarial es un anti-patrón que se caracteriza por la falta de coordinación y planificación en un conjunto de sistemas.

Revoltijo

Cuando los elementos de diseño horizontales y verticales se entremezclan, el resultado es una arquitectura inestable. Los elementos de diseño vertical dependen de la aplicación individual y de las implementaciones de software específicas. Los elementos de diseño horizontal son aquellos que son comunes a todas las aplicaciones e implementaciones específicas.

De forma predeterminada, los desarrolladores y arquitectos mezclan los dos. Pero hacer esto limita la reutilización y la solidez de la arquitectura y los componentes del software del sistema. Los elementos verticales provocan dependencias de software que limitan la extensibilidad y la reutilización. La mezcla hace que todos los diseños de software sean menos estables y reutilizables.

Sistemas de tubo de estufa

Los subsistemas se integran de manera ad hoc (Sin planificar) utilizando múltiples estrategias y mecanismos de integración, y todos se integran punto a punto. El enfoque de integración para cada par de subsistemas no se aprovecha fácilmente hacia el de otros subsistemas. Un sistema de tubo de estufa es la analogía de un solo sistema de

tubo de estufa corporativo y se ocupa de cómo se coordinan los subsistemas dentro de un solo sistema.

Cubra sus activos

Los procesos de software basados en documentos a menudo producen requisitos y especificaciones poco útiles porque los autores evaden tomar decisiones importantes. Para evitar cometer un error, los autores toman un curso más seguro y elaboran alternativas.

Dependencia de un proveedor

La dependencia hacia un proveedor ocurre en sistemas que dependen en gran medida de arquitecturas propietarias. El uso de capas de aislamiento arquitectónico puede brindar independencia de las soluciones específicas del proveedor.

Arquitectura por implicación

La gestión del riesgo en el desarrollo del sistema de seguimiento a menudo se pasa por alto debido al exceso de confianza y los éxitos recientes del sistema. Un enfoque de arquitectura general que se adapte a cada sistema de aplicación puede ayudar a identificar requisitos únicos y áreas de riesgo.

Warm Bodies

Los proyectos de software a menudo cuentan con programadores con habilidades y niveles de productividad muy variados. Muchas de estas personas pueden ser asignadas para cumplir con los objetivos de tamaño del personal (los llamados "cuerpos cálidos"). Los programadores expertos son esenciales para el éxito de un proyecto de software. Los llamados programadores heroicos son excepcionalmente productivos, pero tan solo 1 de cada 20 tiene este

talento. Producen un orden de magnitud más de software funcional que un programador promedio.

Diseño por Comité

El clásico Anti-patrón de los organismos de estándares, el diseño por comité crea arquitecturas demasiado complejas que carecen de coherencia. La aclaración de los roles arquitectónicos y la mejora de la facilitación de procesos pueden refactorizar los malos procesos de reuniones en eventos altamente productivos.

Navaja suiza (Swiss Army Knife)

Una navaja suiza es una interfaz de clase excesivamente compleja. El diseñador intenta proporcionar todos los usos posibles de la clase. En el intento, agrega una gran cantidad de firmas de interfaz en un intento inútil de satisfacer todas las necesidades posibles.

Reinventar la rueda

La falta generalizada de transferencia de tecnología entre proyectos de software conduce a una reinvención sustancial. El conocimiento de diseño oculto en los activos heredados se puede aprovechar para reducir el tiempo de comercialización, el costo y el riesgo.

El Gran Viejo Duque de York

Los procesos de software igualitarios a menudo ignoran los talentos de las personas en detrimento del proyecto. La habilidad para programar no equivale a la habilidad para definir abstracciones. Parece haber dos grupos distintos involucrados en el desarrollo de software: los abstraccionistas y sus contrapartes, los implementadores.

3. Anti-patrones de la gestión de proyectos de software

En la profesión de ingeniería moderna, más de la mitad del trabajo involucra la comunicación humana y la resolución de problemas de las personas. Los Anti-patrones de gestión identifican algunos de los escenarios clave en los que estos problemas son destructivos para los procesos de software.

El papel del director técnico está cambiando. Antes del correo electrónico y las intranets omnipresentes, los gerentes eran principalmente comunicadores organizacionales. Las cadenas de gestión transmitieron información a través de los límites organizacionales, mientras que, en la organización electrónica, la comunicación puede ocurrir sin problemas a través del espacio, el tiempo y los límites.

Tradicionalmente, un papel clave de la gerencia ha sido autorizar excepciones a las reglas y procedimientos. Pero la reingeniería de procesos de negocios (BPR) de las estructuras organizacionales ha cambiado ese rol significativamente. Antes de la reingeniería, los límites organizacionales imponían reglas comerciales heredadas que a menudo eran contraproducentes. En las organizaciones rediseñadas, se eliminan los límites improductivos y se empodera a las personas para resolver problemas sin la intervención de la gerencia.

Sin embargo, en el desarrollo de software, los gerentes aún desempeñan varias funciones importantes, en las áreas de:

- Gestión de procesos de software.
- Gestión de recursos (humanos e infraestructura de TI).
- Gestión de relaciones externas (p. ej., clientes, socios de desarrollo).

Dentro de los anti-patrones correspondientes a la gestión de proyecto podemos encontrar:

Blowhard Jamboree

Blowhard Jamboree se podría traducir como el fanfarrón del Jamboree.

El término Jamboree se utiliza para denominar unos encuentros internacionales de scouts que se celebran periódicamente (su origen parece ser que proviene de una palabra zulú que significa reunión de todas las tribus).

Las opiniones de los llamados expertos de la industria a menudo influyen en las decisiones tecnológicas. Los informes controvertidos que critican tecnologías particulares aparecen con frecuencia en medios populares y publicaciones privadas. Además de las responsabilidades técnicas, los desarrolladores dedican demasiado tiempo a responder las inquietudes de los gerentes y tomadores de decisiones que surgen de dichos informes.

Muchos de estos supuestos expertos están mal informados; en ocasiones, representan puntos de vista sesgados. A menudo, la información que están reportando es de segunda mano. Rara vez hay alguna investigación práctica y experiencia que respalda sus conclusiones.

Parálisis de Análisis

La parálisis de análisis es uno de los anti patrones clásicos en el desarrollo de software orientado a objetos.

El análisis orientado a objetos se enfoca en descomponer un problema en sus partes constituyentes, pero no existe un método obvio para identificar el nivel exacto de detalle necesario para el diseño del sistema, por los diseñadores hasta aplicar las técnicas para lograr la

mítica "integridad". Además, los desarrolladores de sistemas a menudo caen voluntariamente en la parálisis del análisis, ya que "los diseños nunca fallan, solo las implementaciones". Al prolongar las fases de análisis y diseño, evitan arriesgar la rendición de cuentas por los resultados.

Por supuesto, esta es una estrategia perdedora, porque generalmente hay algún punto después del cual se espera una implementación funcional.

Viewgraph Engineering

En algunos proyectos, los desarrolladores se atascan preparando gráficos de vista y documentos en lugar de desarrollar software. La gerencia nunca obtiene las herramientas de desarrollo adecuadas, y los ingenieros no tienen otra alternativa que usar software de automatización de oficinas para producir diagramas y documentos pseudotécnicos.

Esta situación es frustrante para los ingenieros, no utiliza sus verdaderos talentos y permite que sus habilidades se vuelvan obsoletas.

Muerte por planificación

En muchas culturas organizacionales, la planificación detallada es una actividad asumida para cualquier proyecto. Esta suposición es apropiada para actividades de fabricación y muchos otros tipos de proyectos, pero no necesariamente para muchos proyectos de software, que contienen muchas incógnitas y actividades caóticas por su propia naturaleza. La muerte por planificación ocurre cuando los planes detallados para proyectos de software se toman demasiado en serio.

Miedo al éxito

Un fenómeno interesante ocurre a menudo cuando las personas y los proyectos están al borde del éxito. Algunas personas comienzan a preocuparse excesivamente por el tipo de cosas que pueden salir mal. Las inseguridades sobre la competencia profesional salen a la superficie. Cuando se discuten abiertamente, estas preocupaciones e inseguridades pueden ocupar la mente de los miembros del equipo del proyecto. Se pueden tomar decisiones irracionales y acciones inapropiadas para abordar estas preocupaciones.

Por ejemplo, estas discusiones pueden generar publicidad negativa fuera del equipo del proyecto que afecta la forma en que se percibe el entregable y en última instancia puede tener un efecto destructivo en el resultado del proyecto.

El miedo al éxito está relacionado con problemas de terminación. En general, la dinámica de grupo progresa a través de varias fases, perceptibles tanto para proyectos de una semana como para esfuerzos de mayor duración. La primera fase aborda los problemas de aceptación del grupo.

En la segunda fase, a medida que se forman las relaciones, los individuos asumen varios roles en el grupo, incluidos roles formales en la organización y roles autodeterminados informales. Este es un factor importante en la formación de equipos. Una vez establecidos los roles, se realiza el trabajo (tercera fase). Muchos problemas de personalidad pueden surgir durante esta fase. Debido a que la finalización del proyecto puede resultar en la disolución del grupo, estos problemas a menudo surgen a medida que se acerca la finalización del proyecto (cuarta fase).

En la fase de terminación, las preocupaciones sobre el resultado del proyecto, su ciclo de vida futuro y las actividades subsiguientes del grupo a menudo se expresan de manera indirecta. En otras palabras, la gente hace locuras.

Mazorcas de maíz

Las mazorcas de maíz son personas difíciles que pueden prevalecer en el negocio del desarrollo de software. Esta actitud puede deberse a aspectos de la personalidad individual, pero a menudo las dificultades surgen de motivaciones personales para el reconocimiento o incentivos monetarios.

Debido a los cronogramas rigurosos y la presión presupuestaria, el desarrollo de software puede volverse estresante. Las mazorcas de maíz exacerbar estos problemas y crean un estrés adicional innecesario en lo que puede ser un entorno ya demasiado estresado.

Violencia Intelectual

La violencia intelectual ocurre cuando alguien que entiende una teoría, tecnología o palabra de moda usa este conocimiento para intimidar a otros en una situación de reunión. Esto puede suceder sin darse cuenta debido a la reticencia normal de los técnicos a exponer su ignorancia.

En definitiva, la Violencia Intelectual es una ruptura de la comunicación. Cuando algunas o la mayoría de las personas en un proyecto no entienden un nuevo concepto, el progreso puede estancarse indefinidamente mientras resuelven sus sentimientos de inferioridad o evitan el tema por completo. Cuando la Violencia Intelectual es omnipresente, surge una cultura defensiva que inhibe la

productividad. Las personas controlan y ocultan la información en lugar de compartirla.

Mala gestión del proyecto

Este Anti-Patrón se refiere al seguimiento y control de proyectos de software. El marco de tiempo para esto ocurre después de las actividades de planificación y durante el análisis, diseño, construcción y prueba reales del sistema de software. La mala gestión del proyecto implica errores cometidos en la ejecución diaria de un proyecto, suponiendo que no se hayan cometido errores de planificación (como Muerte por planificación).

Tíralo por encima del muro

Rara vez la documentación se explica por sí misma por completo, sin embargo, comprender la visión y el conocimiento de los autores es una parte esencial de la comprensión de la documentación. Esto es especialmente cierto en el caso de los documentos de directrices, en los que existe una suposición implícita de toma de decisiones independiente. Esta suposición también implica un conocimiento profundo de la intención de los autores.

Por su naturaleza, todo conocimiento humano es personal. Incluso los científicos más eminentes poseen una visión personal que impulsa su descubrimiento y articulación de nueva información. Comprender esta percepción personal es esencial para comprender su trabajo.

Problema de antipatrón

Los métodos orientados a objetos, los patrones de diseño y los planes de implementación pensados como pautas flexibles son tomados literalmente con demasiada frecuencia por los administradores

posteriores y los desarrolladores de OO. A medida que las guías avanzan a través de los procesos de aprobación y publicación, se les pueden atribuir cualidades incumplidas de integridad, carácter prescriptivo e implementación obligatoria.

Tal interpretación literal de pautas flexibles puede conducir a resultados no deseados. Es posible que se tomen decisiones en base a lineamientos cuyo único propósito fue inspirar un análisis cuidadoso y una toma de decisiones optimizada localmente.

Por ejemplo, el esfuerzo puede desperdiciarse en análisis y documentación inútiles porque parece obligatorio, aunque nadie en el equipo de desarrollo entiende su propósito. Este fenómeno ocurre tanto en organizaciones grandes como pequeñas y puede deberse a una falta de comunicación entre las fases de desarrollo. Otra causa importante es el deseo de satisfacer las aparentes expectativas de la gerencia en lugar de las necesidades de los usuarios finales del sistema.

Simulacro de incendio

Un simulacro de incendio es un escenario recurrente en muchas organizaciones de desarrollo de software. Se inicia un proyecto, pero el personal retrasa las actividades de diseño y desarrollo durante varios meses mientras se resuelven varios problemas tecnopolíticos a nivel de gestión. (Un desarrollador de software describió las estrategias para la entrega de software en el trabajo como: "Espere hasta que la gerencia esté desesperada y aceptarán cualquier cosa que les dé").

La gerencia evita que el personal de desarrollo progrese, ya sea diciéndoles que esperen o dándoles instrucciones inciertas y contradictorias. Quizás los más destructivos son los cambios

generados externamente en la dirección del proyecto que conducen a la reelaboración e inhiben el progreso.