**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. De forma predeterminada, 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.

**Ancla del barco:**
Un ancla de barco es una pieza de software o hardware que no sirve
propósito en el proyecto actual. A menudo, el ancla de barco es una adquisición costosa, que
hace que la compra sea aún más irónica.

**Obsolescenciacontinua:**
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 funcionan juntos. Dado que cada línea de productos comerciales evoluciona
a través de nuevos lanzamientos, la situación es cada vez más difícil de hacer frente a los desarrolladores. Encontrar versiones compatibles de productos que interoperan correctamente es aún más difícil.

**Callejón Sin Salida:**
Se alcanza un callejón sin salida modificando un componente reutilizable si el componente ya no es mantenido y apoyado por el proveedor. Cuando estos modificaciones, las transferencias de carga de soporte a los desarrolladores de sistemas de aplicaciones Las mejoras en el componente reutilizable no se integran fácilmente, y problemas de apoyo pueden ser culpados a la modificación.

**Kludge deentrada:** El software que falla en las pruebas de comportamiento directas puede ser un ejemplo de un kludge de entrada, que se produce cuando se emplean algoritmos ad hoc para manejar la entrada del programa.
K (klumpsy – gruñón : torpe
L (coja : debil)
U (feo : feo)
D (tonto : tonto)
GE (suficiente : bueno para nada)

**Manejo de Setas:** En algunos círculos de arquitectura y gestión, hay un
directiva explícita para mantener a los desarrolladores del sistema aislados de los usuarios finales del sistema.
Los requisitos se pasan de segunda mano a través de intermediarios,
gerentes o analistas de requisitos.

**The Blob:** El diseño de estilo procesal conduce a un objeto con una parte del león de la responsabilidades, mientras que la mayoría de los demás objetos solo contienen datos o ejecutan procesos simples. La solución incluye la refactorización del diseño para distribuir las responsabilidades de forma más uniforme y aislar el efecto de los cambios.

**Flujo delava:** El código muerto y la información de diseño olvidada se congelan en un Diseño. Esto es análogo a un flujo de lava con globulos endurecidos de material rocoso. La solución refactorada incluye un proceso de administración de configuración que elimina el código muerto y evoluciona o refactoriza el diseño para aumentar la calidad.

**Descomposición Funcional:** Este AntiPattern es la salida de experimentado, desarrolladores no orientados a objetos que diseñan e implementan una aplicación en un lenguaje orientado a objetos. El código resultante se asemeja a un lenguaje estructural (Pascal,
FORTRAN) en la estructura de clase. Puede ser increíblemente complejo como procedimiento inteligente
los desarrolladores idean formas muy "clever" de replicar sus métodos probados en un 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 refactorada incluye una reasignación de responsabilidades a objetos de larga duración que eliminan a los Poltergeists.

**Martillo Dorado:** Un Martillo Dorado es una tecnología o concepto conocido aplicado obsesivamente con muchos problemas de software. La solución consiste en ampliar el conocimiento de los desarrolladores a través de grupos de educación, formación y estudio de libros para exponer a los desarrolladores a tecnologías y enfoques alternativos.

**Código deespaguetis :** La estructura de software ad hoc hace que sea difícil ampliar y optimizar
Código. La refactorización frecuente de código puede mejorar la estructura del software, el software de soporte
mantenimiento y permitir el desarrollo iterativo.

**Caminar a través de un campominado :** El uso de la tecnología de software de hoy en día es análogo a
caminando a través de un campo minero de alta tecnología [Beizer 97a]. Numerosos errores se encuentran en los productos de software liberados; de hecho, los expertos estiman que el código fuente original contiene de dos a cinco errores por línea de código.

**Programación de cortar ypegar:** el código reutilizado copiando instrucciones de origen conduce a
problemas de mantenimiento significativos. Las formas alternativas de reutilización, incluida la reutilización de cajas negras, reducen los problemas de mantenimiento al tener código fuente, pruebas y documentación comunes.

**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 a los éxitos recientes del sistema. Un enfoque de arquitectura general que se adapta a cada sistema de aplicación puede ayudar a identificar requisitos únicos y áreas de riesgo.

**Diseño por Comité:** El clásico AntiPattern de los organismos de normalización, Design by Committee crea arquitecturas demasiado complejas que carecen de coherencia. La clarificación de los roles arquitectónicos y la mejora de la facilitación de los procesos pueden refactorizar los malos procesos de reunión en eventos altamente productivos.

**Reinventar la rueda:** La falta generalizada de transferencia de tecnología entre proyectos de software conduce a una reinvención sustancial. Los conocimientos de diseño enterrados en activos heredados se pueden aprovechar para reducir el tiempo de comercialización, el costo y el riesgo.

**Stovepipe Enterprise:** Un sistema Stovepipe se caracteriza por una estructura de software que inhibe el cambio. La solución refactorada describe cómo abstraer subsistema y componentes para lograr una estructura de sistema mejorada. El Stovepipe Enterprise AntiPattern se caracteriza por la falta de coordinación y planificación en un conjunto de sistemas.

**Sistema Stovepipe:** Los subsistemas se integran de manera ad hoc utilizando múltiples estrategias y mecanismos de integración, y todos están integrados punto a punto. El enfoque de integración para cada par de subsistemas no se aprovecha fácilmente hacia el de otros subsistemas. El sistema Stovepipe AntiPattern es la analogía de un solo sistema de Stovepipe Enterprise, y se ocupa de cómo se coordinan los subsistemas dentro de un solo sistema.

**Bloqueo delproveedor:** el bloqueo del proveedor se produce en sistemas que dependen en gran medida de las arquitecturas propietarias. El uso de capas de aislamiento arquitectónico puede proporcionar independencia de las soluciones específicas del proveedor.

Mezcla de mezclas, embarular

**Stovepipe generadoautomáticamente:** Este AntiPattern se produce al migrar un sistema de software existente a una infraestructura distribuida. Un Stovepipe generado automáticamente surge al convertir las interfaces de software existentes en interfaces distribuidas. Si se utiliza el mismo diseño para la informática distribuida, surgen una serie de problemas.

**Cubrir susactivos: Los** procesos de software basados en documentos a menudo producen
requisitos y especificaciones menos útiles porque los autores evaden la fabricación decisiones importantes. Con el fin de evitar cometer un error, los autores toman un curso más seguro y elaboran alternativas.

**El Gran Duque Viejo de York:** Los procesos de software igualitarios a menudo ignoran la
talento en detrimento del proyecto. La habilidad de programación no equivale a la habilidad en
definir abstracciones. Parece haber dos grupos distintos involucrados en el software

desarrollo: *abstraícionistas* y sus contrapartes los *implementadores.*

**Violencia Intelectual (en la última edicion está Management Pattern): como**La violencia intelectual ocurre cuando alguien que entiende una teoría, tecnología o palabra de moda utiliza este conocimiento para intimidar a otros en una situación de reunión.

**Jumble:** Cuando se mezclan elementos de diseño horizontales y verticales, un resultados de la arquitectura. La mezcla de elementos de diseño horizontal y vertical limita la reutilización y robustez de la arquitectura y los componentes del software del sistema.

**Cuchillo del ejércitosuizo:** Un cuchillo del ejército suizo es una interfaz de clase excesivamente compleja. El
diseñador intenta proporcionar todos los usos posibles de la clase. En el intento, él o ella agrega un gran número de firmas de interfaz en un intento inútil de satisfacer todas las necesidades posibles.

**Wolf Ticket:** A Wolf Ticket es un producto que reclama apertura y conformidad normas que no tienen un significado exigible. Los productos se entregan con interfaces propietarias que pueden variar significativamente del estándar publicado.

**Parálisis del análisis:** Esfuerzo por la perfección y la integridad en la fase de análisis a menudo
conduce a un estancamiento del proyecto y a una paliza excesiva de los requisitos/modelos. La solución refactorada incluye una descripción de los procesos de desarrollo iterativos e incrementales que aplazan el análisis detallado hasta que se necesita el conocimiento.

**Corncob:** Las personas difíciles con frecuencia obstruyen y desvían el desarrollo de software
Proceso. Los Corncobs se pueden tratar abordando sus agendas a través de diversas acciones organizativas tácticas, operativas y estratégicas.

**Death by Planning:** La planificación excesiva de proyectos de software conduce a horarios que causan problemas posteriores. Explicamos cómo planificar un
proceso de desarrollo de software que incluye la incorporación de hechos conocidos y replanificación.

**Gestión irracional:** La indecisión habitual y otros malos hábitos de gestión conducen a decisiones de facto y crisis crónicas de desarrollo. Explicamos cómo utilizar técnicas de toma de decisiones de gestión racional para mejorar la resolución de proyectos y para mantener a los gerentes en marcha.

**Gestión del proyecto:** Falta de atención a la gestión del desarrollo de software procesos pueden causar insegura y otros síntomas. Monitoreo adecuado y
el control de los proyectos de software es necesario para el éxito de las actividades de desarrollo. Ejecutar un
desarrollo de productos es una actividad tan compleja como la creación del plan de proyecto, y

el desarrollo de software es tan complejo como la construcción de rascacielos, que implica tantos pasos y
procesos, incluyendo cheques y saldos. A menudo, las actividades clave se pasan por alto o
Minimizado.

**Blowhard Jamboree:** Las opiniones de los llamados expertos de la industria a menudo influyen en las decisiones tecnológicas. Los informes polémicos que critican tecnologías particulares aparecen con frecuencia en los medios de comunicación populares y en las publicaciones privadas. Además de las responsabilidades técnicas, los desarrolladores dedican demasiado tiempo a responder a las preocupaciones de los gerentes y responsables de la toma de decisiones que surgen de estos informes.

**El correo electrónico espeligroso:** el correo electrónico es un medio de comunicación importante para los administradores de software. Desafortunadamente, es un medio inapropiado para muchos temas y comunicaciones sensibles.

**Miedo aléxito:** Un fenómeno interesante ocurre a menudo cuando las personas y los proyectos están al borde del éxito. Algunas personas comienzan a preocuparse obsesivamente por el tipo de cosas que *pueden* salir mal. Las inseguridades sobre la competencia profesional salen a la superficie.

**Los conflictos de Feud:** Personality entre gerentes pueden afectar dramáticamente el trabajo
Ambiente. Los empleados que informan a estos gerentes a menudo sufren las consecuencias de los desacuerdos de sus supervisores. La animosidad entre los gerentes se refleja en las actitudes y acciones de sus empleados.

**Humo yespejos:** Los sistemas de demostración son herramientas de ventas importantes, ya que a menudo son interpretados por los usuarios finales como representacionales de las capacidades de calidad de producción. A
equipo de gestión, ansioso por nuevos negocios, a veces (inadvertidamente) fomenta estas percepciones erróneas y hace compromisos más allá de las capacidades de la organización para entregar tecnología operativa.

**Lanzarlo sobre lapared:** métodos orientados a objetos, patrones de diseño e implementación
los planes destinados como directrices flexibles son tomados con demasiada frecuencia literalmente por los gerentes intermedios y los desarrolladores orientados a objetos. A medida que las directrices avanzan a través de la aprobación y
procesos de publicación, a menudo se atribuyen con cualidades incumplidas de ompleteness, prescripción e implementación obligatoria.

**Ingeniería de Viewgraph:** En algunos proyectos, los desarrolladores se quedan atascados preparando cuadros de vista y documentos en lugar de desarrollar software. La administración nunca obtiene las herramientas de desarrollo adecuadas, y los ingenieros no tienen otra alternativa que utilizar software de automatización de oficina para producir diagramas y documentos técnicos psuedo.

**Warm Bodies (en la última edicion está Architecture Pattern) : como** 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 del tamaño del personal (los llamados "cuerpos calientes"). Los programadores cualificados son esenciales para el éxito de un proyecto de software. Los llamados programadores heroicos son excepcionalmente productivos, pero tan solo 1 de cada 20 tienen este talento. Producen un orden de magnitud más software de trabajo que un programador promedio.

**Fire Drill:** Los pilotos de aerolíneas describen volar como "horas de aburrimiento seguidas de 15 segundos de terror". Muchos proyectos de software se asemejan a esta situación: "Meses de aburrimiento seguidos de las demandas de entrega inmediata". Los meses de aburrimiento pueden incluir análisis prolongados de requisitos, replanificación, espera de financiamiento, esperando la aprobación o cualquier número de razones tecnopolíticas.