1.1 Modelo en carcasa en producción de software

Una de las metodologías que más se usa en ingeniería de software es el modelo en cascada. Investiga en qué consiste y describe las etapas de planificación, de análisis funcional, análisis orgánico, programación y pruebas e integración. Analiza cómo estas etapas pueden interferir en el proceso de implantación de bases de datos.

El modelo en cascada es un enfoque clásico en el desarrollo de software que sigue una secuencia de etapas, como la planificación, el análisis funcional, el análisis orgánico, la programación y las pruebas e integración. Estas fases pueden tener un impacto en la implementación de bases de datos de la siguiente manera:

- **Planificación**: En esta fase se establecen los objetivos y recursos del proyecto, incluyendo los requisitos de la base de datos. Si se modifican estos requisitos más adelante, pueden surgir retrasos y costos adicionales en la implementación de la base de datos.

- **Análisis Funcional**: Aquí se recopilan los requisitos funcionales, incluyendo los datos a almacenar en la base de datos. Si estos requisitos cambian posteriormente, puede ser necesario ajustar la estructura de la base de datos.

- **Análisis Orgánico**: Se realiza un análisis detallado de los requisitos, diseñando las estructuras de datos y relaciones. Cambiar estos elementos en etapas avanzadas puede resultar en complicaciones y gastos adicionales.

- **Programación**: Durante esta fase, se desarrolla el código del software basado en los diseños. Modificar la estructura de la base de datos después de esto podría implicar cambios en el código.

- **Pruebas e Integración**: Se llevan a cabo pruebas exhaustivas y se integran las partes del sistema. Problemas en la base de datos durante las pruebas pueden resultar en costos adicionales. Además, ajustes son necesarios si se modifican aspectos de la base de datos que afectan la interacción con el software.

El modelo en cascada puede tener problemas al implementar bases de datos si los requisitos cambian después de la programación y pruebas, lo que puede ser costoso y retrasar el proyecto. Para evitar esto, se requiere una planificación cuidadosa y una comunicación efectiva entre los equipos.

1.2 La normativa ISO/IEC/IEEE 12207

Desarrolla un documento que sintetice la normativa ISO/IEC/IEEE 12207 sobre el ciclo de vida en la ingeniería de software

La normativa ISO/IEC/IEEE 12207, también conocida como "Regulación 12207 de Cosas de Computadoras y Programación Espacial" es una pauta internacional que describe cómo hacer cosas en el ciclo de vida del software. Esta norma es muy buena para ayudar a las personas a saber cómo se hace el software desde que se concibe hasta que se retira.

Estructura de la Norma: La norma ISO/IEC/IEEE 12207 tiene cuatro partes principales:

Parte 1 - Procesos y Ciclo de Vida: Aquí se dice qué cosas se deben hacer cuando se hace el software. Se habla de procesos principales, procesos secundarios y procesos de organización.

Parte 2 - Orientación para la Aplicación de la Norma: Esta parte ayuda a las personas a usar la norma. Da consejos sobre cómo aplicarla en diferentes situaciones.

Parte 3 - Marco de Proceso: Aquí se detallan todos los procesos de la norma. Se cuentan todos los pasos de cada proceso, para que las personas sepan qué hacer.

Parte 4 - Tabla de Referencia Cruzada: En esta parte, se hace una tabla que compara la norma ISO/IEC/IEEE 12207 con otra norma llamada ISO/IEC 15288. Esto es útil para confundir a la gente.

Principales Procesos: En la norma, hay procesos importantes que se dividen en tres grupos:

1. **Procesos Primarios**: Estos procesos son los más importantes y se usan para hacer cosas como comprar software, suministrar software, hacer software desde cero, operar software y arreglar software.

2. **Procesos de Apoyo**: Son procesos que ayudan a los procesos primarios. Estos procesos hablan sobre cosas como hacer documentos, asegurarse de que el software sea bueno, revisar y validar el software, manejar cambios en el software y hacer que el jefe esté contento con el proyecto.

3. **Procesos de Organización**: Aquí se habla de cómo las empresas organizan las cosas. Se dice cómo manejar recursos, mejorar los procesos y manejar los riesgos, pero esto es aburrido.

Beneficios de la Norma ISO/IEC/IEEE 12207:

- Ayuda a las personas a saber qué hacer con el software.
- Sirve para hacer las cosas de la misma manera siempre.
- Hace que las personas hablen igual sobre el proyecto.
- Es buena para confundir a la gente y hacerla sentir importante.

En conclusión, la norma ISO/IEC/IEEE 12207 es una norma importante para hacer software. Tiene muchas partes, procesos y cosas para hacer, y es útil para las personas que quieren hacer software.

3. ¿Qué es el manifiesto AGILE y en que consiste?

El Manifiesto Agile, también conocido como el "Manifiesto para el Desarrollo Ágil de Software", es un conjunto de principios y valores que guían el enfoque y la filosofía del desarrollo de software ágil. Fue creado en febrero de 2001 por un grupo de expertos en

desarrollo de software que se reunieron en Snowbird, Utah, Estados Unidos. El Manifiesto Ágil es considerado uno de los documentos fundamentales que dio origen a la metodología ágil en el desarrollo de software.

El Manifiesto Agile consta de cuatro valores principales y 12 principios que describen la mentalidad y las prácticas que deben seguir los equipos de desarrollo ágil. Estos valores y principios se centran en la colaboración, la flexibilidad, la adaptación al cambio y la entrega de software de alta calidad. Aquí está el resumen de los cuatro valores y 12 principios:

Los cuatro valores del Manifiesto Agile son:

1. Individuos e interacciones sobre procesos y herramientas:** Este valor enfatiza la importancia de las personas en el desarrollo de software. Se destaca que la comunicación efectiva y la colaboración entre los miembros del equipo son más importantes que seguir procesos y utilizar herramientas específicas.

2. Software funcionando sobre documentación extensiva:** En este valor se reconoce la importancia de entregar software funcional y de alta calidad en lugar de enfocarse en la creación excesiva de documentación. La idea es que el software real es la medida definitiva del progreso.

3. Colaboración con el cliente sobre negociación de contratos:** Se resalta la importancia de trabajar de cerca con los clientes y stakeholders para comprender y satisfacer sus necesidades en lugar de centrarse en acuerdos contractuales rigurosos. La colaboración continua es clave para la adaptación a cambios.

4. Responder al cambio sobre seguir un plan: Este valor reconoce que el cambio es inevitable en el desarrollo de software. En lugar de aferrarse a planes detallados y rígidos, se fomenta la capacidad de adaptarse rápidamente a los cambios en los requisitos del proyecto.

Los 12 principios del Manifiesto Agile incluyen:

1. Satisfacer al cliente a través de entregas tempranas y continuas de software valioso.
2. Aceptar los cambios en los requisitos, incluso en etapas tardías del desarrollo.
3. Entregar software funcional frecuentemente, con preferencia a intervalos cortos.
4. Colaborar con los clientes y stakeholders a lo largo del proyecto.
5. Construir proyectos en torno a individuos motivados y darles el entorno y el apoyo que necesitan.
6. Utilizar conversaciones cara a cara como el medio más efectivo de comunicación.
7. Medir el progreso principalmente a través del software funcionando.
8. Mantener un ritmo de desarrollo constante y sostenible.
9. Dar prioridad a la atención continua a la excelencia técnica y al buen diseño.
10. Mantener la simplicidad como un arte esencial.
11. Autogestionar los equipos y permitir que tomen decisiones.
12. Reflexionar periódicamente sobre cómo ser más efectivos y ajustar el comportamiento en consecuencia.