3

**Construcción**

Hace referencia a la creación detallada de software operativo y significativo, por medio de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración.

**Fundamentos**

1. Minimizar la complejidad: énfasis en la creación de código que sea simple y legible, y no tanto inteligente. Uso de estándares y distintas técnicas de codificación y calidad.
2. Anticiparse a los cambios: El software cambiará a lo largo del tiempo. El software es inevitablemente parte de los ambientes externos que cambian continuamente, y los cambios en esos ambientes externos afectan el software de diversos modos.
3. Construir para verificar: Construir software de tal manera que los ingenieros de software puedan sacar a relucir los fallos con facilidad al estar escribiendo el código, además de cuando realizan pruebas independientes y actividades operacionales. Se incluye el seguimiento de estándares de codificación que permiten la revisión del código, las pruebas unitarias, la organización del código que permita pruebas automáticas y el uso restringido de estructuras del lenguaje que sean complejas o difíciles de entender
4. Estándares en la construcción:

* Métodos de comunicación como los estándares para los formatos de los documentos y de los contenidos
* Programación de lenguajes como estándares de lenguajes para Java y C++
* Plataformas como estándares de interfaces del programador para llamadas al sistema operativo
* Herramientas como estándares diagramáticos para anotaciones como UML

Uso de estándares externos: Provienen de numerosas fuentes incluyendo las especificaciones de interfaz del hardware y del software, tales como el Grupo de Gestión de Objetos (OMG) y las organizaciones internacionales como la IEEE o ISO.

Uso de estándares internos: Pueden crearse partiendo de una base organizacional a un nivel corporativo para su uso en proyectos específicos. Estos estándares permiten la coordinación de actividades de grupo y los demás fundamentos de construcción de software.

**Gestión de la construcción**

1. Modelos: Algunos modelos son más lineales que otros, tales como los modelos en cascada y los de ciclo de vida de entregas por etapa; estos modelos tratan la construcción como una actividad que sucede sólo después de que se haya completado un significativo trabajo con los requisitos. La codificación sería el punto de mayor énfasis de la construcción. Otros modelos son más interactivos, tales como el prototipado evolucionista, programación extrema y el scrum. Tienden a tratar la construcción como una actividad que ocurre junto con otras actividades de desarrollo como los requisitos, el diseño y la planificación.
2. Planificación: Un aspecto clave es la elección de un método de construcción, el cual afecta hasta donde se realizan los requisitos de construcción, el orden en que se realizan y el grado hasta el que se espera que se complete antes de que comience el trabajo de construcción. La planificación también define el orden en que se crean e integran, según el método elegido, los componentes, los procesos de gestión de la calidad del software, la asignación de tareas a ingenieros de software específicos y el resto de las tareas.
3. Mediciones: Se puede medir el código desarrollado, el modificado, el reutilizado, el destruido, la complejidad del código, las estadísticas de la inspección del código, las tasas de rectificaciones y de identificación de errores y los horarios.

**Consideraciones prácticas**

1. Diseño de construcción: Tiende a estar dictaminado por restricciones inamovibles impuestas por un problema del mundo real qué está haciendo afrontado por el software.
2. Lenguajes de construcción: El tipo más simple es el *lenguaje de configuració*n en el que los ingenieros de software eligen de entre un conjunto limitado de opciones predefinidas para crear nuevas o típicas instalaciones de software. Los *lenguajes de herramientas* se utilizan para conseguir aplicaciones partiendo de las herramientas. Pueden definirse como lenguajes de programación de aplicaciones. *Los lenguajes de programación* son los más flexibles, incluyen notaciones lingüísticas, formales y visuales.
3. Codificación: Asignación de nombres, utilización de clases, tipos enumerados, variables, estructuras de control, prevención de brechas en la seguridad, organización del código fuente, documentación.
4. Pruebas de construcción: Unitarias y de integración. Su objetivo es reducir la brecha entre el tiempo en el que se introducen fallos en el código y el tiempo en el que se detectan esos fallos.
5. Reutilización: La selección de unidades, bases de datos, procedimientos de pruebas con datos de pruebas reutilizables. La evaluación de la posibilidad de reutilización del código o de las pruebas. Comunicar la información sobre utilización realizada en el código nuevo, los procedimientos de prueba son los datos de prueba.
6. Calidad de la construcción: Pruebas unitarias y de integración desarrollo de primero-haz-pruebas, aserciones, depuración, revisiones técnicas, análisis estático.
7. Integración: Integración de rutinas, clases, componentes y subsistemas construidos por separado, incluso integración con otros sistemas software y hardware.

Preguntas

1. Hace referencia a la creación detallada de software operativo y significativo, por medio de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración.
2. **Construcción de software**
3. Pruebas del software
4. Diseño del software
5. Mantenimiento del software
6. Este fundamento de la construcción hace énfasis en la creación de código que sea simple y legible, y no tanto inteligente. Uso de estándares y distintas técnicas de codificación y calidad.
7. Estándares de construcción
8. Construir para verificar
9. Anticiparse a los cambios
10. **Minimizar la complejidad**
11. El software es inevitablemente parte de los ambientes externos que cambian continuamente, y los cambios en esos ambientes externos afectan el software de diversos modos, por lo tanto, el software debe…
12. Estándares de construcción
13. Construir para verificar
14. **Anticiparse a los cambios**
15. Minimizar la complejidad
16. Construir software de tal manera que los ingenieros de software puedan sacar a relucir los fallos con facilidad al estar escribiendo el código, además de cuando realizan pruebas independientes y actividades operacionales

a) Estándares de construcción

b) **Construir para verificar**

c) Anticiparse a los cambios

d) Minimizar la complejidad

1. Este tipo de estándares provienen de numerosas fuentes incluyendo las especificaciones de interfaz del hardware y del software, tales como el Grupo de Gestión de Objetos (OMG) y las organizaciones internacionales como la IEEE o ISO
2. **Estándares externos**
3. Estándares internos
4. Pueden crearse partiendo de una base organizacional a un nivel corporativo para su uso en proyectos específicos. Estos estándares permiten la coordinación de actividades de grupo y los demás fundamentos de construcción de software.

a) Estándares externos

**b) Estándares internos**

1. Define el orden en que se crean e integran, según el método elegido, los componentes, los procesos de gestión de la calidad del software, la asignación de tareas a ingenieros de software específicos y el resto de las tareas.
2. **Planificación del software**
3. Mediciones del software
4. Modelos del software
5. Esta consideración práctica tiende a estar dictaminado por restricciones inamovibles impuestas por un problema del mundo real qué está haciendo afrontado por el software.
6. **Diseño de construcción**
7. Codificación
8. Lenguajes de construcción
9. Es el tipo más simple de lenguaje de construcción, en el que los ingenieros de software eligen de entre un conjunto limitado de opciones predefinidas para crear nuevas o típicas instalaciones de software
10. **Lenguaje de configuración**
11. Lenguaje de herramientas
12. Los lenguajes de programación
13. Este tipo de lenguaje de configuración se utiliza para conseguir aplicaciones partiendo de las herramientas. Pueden definirse como lenguajes de programación de aplicaciones.

a) Lenguaje de configuración

**b) Lenguaje de herramientas**

c) Los lenguajes de programación

1. Este tipo de lenguaje de configuración es el más flexible, incluyen notaciones lingüísticas, formales y visuales.
2. Lenguaje de configuración
3. Lenguaje de herramientas
4. **Los lenguajes de programación**
5. Su objetivo es reducir la brecha entre el tiempo en el que se introducen fallos en el código y el tiempo en el que se detectan esos fallos
6. **Pruebas de construcción**
7. Diseño de construcción
8. Codificación