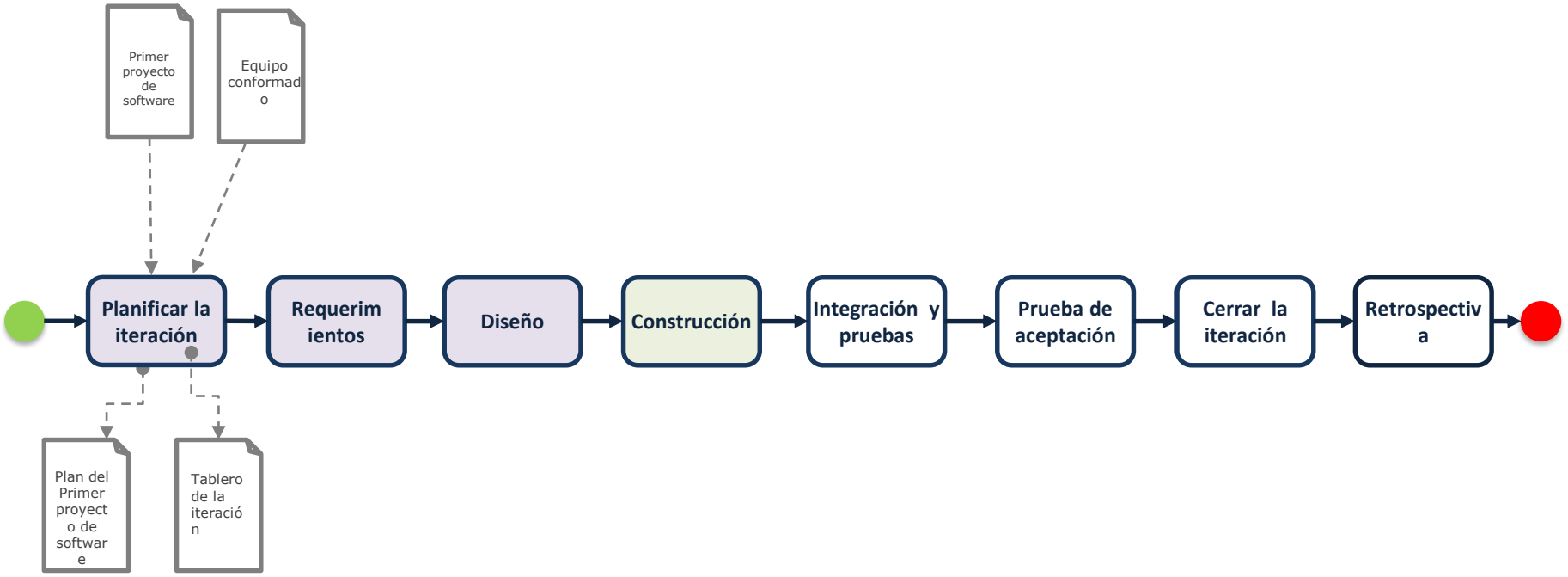
Curso de Ingeniería de Software

Unidad 7
Construcción de software

Guadalupe Ibargüengoitia
Hanna Oktaba

Actividades de la iteración



Entradas esta unidad

Condiciones

- Diseño del software completo

Productos de trabajo

- ***Especificación de Requerimientos de Software***
- ***Especificación de Diseño de Software*** que incluye:
  - *Descripción de la arquitectura*
    - Diagrama de paquetes o componentes
    - Diagrama de distribución
    - Ambiente de implementación
  - *Detalle de los componentes*
  - Diagrama integrado de clases para la Vista.
  - Diagrama integrado de clases para el Controlador
  - Diagrama integrado de clases para el Modelo
  - Diagramas de secuencia para cada caso de uso, en los flujos normal, alternativo y excepcional
  - Diagrama de navegación representado con diagrama de estados
  - Diagrama de Entidad-Relación
- **Plantilla de *Construcción de Software***
- ***Repositorio*** compartido con la carpeta para la iteración actualizado
- ***Tablero*** de la iteración actualizado

Construcción de software

- Es la actividad central en el desarrollo de software.
- Es un proceso iterativo, que parte del diseño detallado de los componentes, los cuales se refinan introduciendo elementos del lenguaje de programación.
- Este refinamiento se repite hasta que se pueda derivar el código.
- La creación de código puede significar la construcción de programas nuevos en lenguajes de programación de alto o bajo nivel o la adaptación de código ya existente.

Objetivos de la construcción

- La generación del código de cada componente definido en el diseño.

- Probar cada componente individualmente para asegurar código confiable antes de integrar todos los componentes.

Fundamentos para la construcción de software (1)

- **Minimizar la complejidad**. Construir código que sea simple y lo puedan leer con facilidad las personas y las máquinas.
- Para lograrlo se hace uso de estándares, diseño modular y se aplican técnicas de construcción basadas en la calidad.

Fundamentos para la construcción de software (2)

- **Anticiparse al cambio**. El software es muy propenso a cambiar constantemente debido a los cambios en el medio ambiente de su uso.

Anticiparse al cambio ayuda a los desarrolladores a construir software que pueda extenderse y mejorarse sin causar efectos graves en la estructura general.

Fundamentos para la construcción de software (3)

- **Construir para la verificación**. Significa que el código escrito será verificado para que los desarrolladores puedan detectar fácilmente los defectos y corregirlos.

Algunas técnicas útiles para facilitar la verificación son:

- El uso de estándares
- Hacer revisiones al código antes de compilarlo
- Organizarlo para facilitar la prueba automática de código
- No incluir estructuras complejas o difíciles de entender.

Fundamentos para la construcción de software (4)

- **Reuso**. Antes de generar código nuevo, es muy recomendable revisar si se tiene código o componentes ya existentes y confiables.

Componentes confiables se encuentran en las bibliotecas de los ambientes de programación y en código realizado por el equipo y que haya sido certificado por el mismo equipo.

Fundamentos para la construcción de software (5)

- **Uso de estándares en la construcción**. Aplicar estándares en la construcción ayuda a obtener los objetivos de eficiencia, calidad y costo del proyecto.

El equipo define su estándar para la comunicación entre métodos, el uso del estándar de su lenguaje de programación con convenciones de nombrado e indentación.

Fundamentos para la construcción de software (6)

- **Manejo de versiones.** Durante la construcción de software se generan varias versiones de cada componente. Se debe llevar un buen control del manejo de estos cambios para no incluir versiones erróneas en los componentes finales.
- Cuando un equipo está construyendo software en conjunto, se debe asegurar que no interfiera un desarrollador con otro, esto es que sus cambios estén coordinados.
- Usar herramientas de control de versiones

Actividades para la construcción de software

- Preparar la construcción del software
- Construir el código de cada componente
- Probar cada componente individualmente
- Integrar los componentes de cada caso de uso

Pruebas de componentes de software

- Las pruebas que se aplican a cada *componente* de forma individual se llaman *pruebas unitarias (unit test)*.
- Probar todos los casos posibles de un componente no es factible.
- Por lo tanto se define un criterio de salida que diga a los programadores cuando deben de dejar de hacer pruebas o cuando las pruebas son suficientes para un determinado tipo de componente.

Tipos de prueba

- **Pruebas caja negra**. Consideran al código como una caja negra, las entradas son los casos de prueba y se ejecuta el código deseando obtener los valores esperados.

Tipos de prueba

- **Pruebas de caja transparente**. Cuando el código de algún componente tiene una o mas sentencias de decisión (tipo *if-then-else* o ciclos), se definen casos de prueba que ejecuten cada uno de los caminos del código.
- Para saber cuántos casos de prueba se necesitan para probar todos los caminos de código, existe una medida llamada ***Complejidad Ciclomática*** (**McCabe**) que consiste en contar cuántas sentencias con decisiones hay en el código, sumarle uno y ese es el número de casos de prueba a definir para probar todos los caminos de decisión.

Sustitutos de componentes para pruebas

- Muchas veces al querer probar un componente, se requiere de otros componentes que no están disponibles.
  - *Stub* – sustituto que simula la invocación del componente no existente y devuelve el resultado esperado.
  - *Driver* - sustituto que simula la inicialización de variables no locales y los parámetros para activar al componente que se está probando.

Métrica de calidad de software

- Medir el número de defectos encontrados y corregidos ayudan a entender la calidad del software en términos de la densidad de defectos por medida de tamaño.

Método Inicial de Desarrollo de Software

- Práctica de Construcción de Software

Práctica de Construcción

Práctica	
Construcción de Software	
Objetivos	
Construir y probar el código para cada componente del diseño de software	
Entrada	**Resultado**
Condiciones • Diseño de software completo **Productos de trabajo** •*Especificación de Requerimientos de Software* •*Especificación Diseño de Software* •Plantilla de *Construcción de Software* •*Repositorio* compartido con la carpeta para la iteración actualizado •*Tablero* de la iteración actualizado	**Condiciones** • Construcción del software completo **Productos de trabajo** •Código de los Componentes del Software probados individualmente •*Construcción de Software* completo •*Repositorio* compartido con la carpeta para la iteración actualizado •*Tablero* de la iteración actualizado

Práctica de Construcción

Actividades
1. Preparar la construcción del software. (TC1) 2. Construir la base de datos. (TC2) 3. Construir y probar el código de cada caso de uso. (TC3) 4. Integrar el documento de la *Construcción de Software.* (TC4)
Herramientas
• Ambiente y herramientas de desarrollo seleccionados • Manejador de bases de datos seleccionado. • Herramientas de prueba de código.

Actividad C1. Preparar la construcción del software

- Cuando se desarrolla software en equipo, antes de empezar a construir el código se deben realizar varias actividades a fin de tener la infraestructura necesaria para que cada desarrollador pueda trabajar independiente en su código.

Técnica TC1: Preparar la construcción del software

1. Instalar las herramientas del desarrollo

Todos los miembros del equipo, bajo la dirección del **Responsable técnico**, ajustan la especificación del ambiente de desarrollo definido en el diseño, copiando la tabla en el documento de *Construcción de software*.

Se aseguran que todos tengan instaladas las mismas versiones para posteriormente facilitar la integración del código.

Técnica TC1: Preparar la construcción del software

2. Definir una estrategia de construcción

- Para este curso se tomó la decisión de la arquitectura de MVC y que cada desarrollador fuera responsable de uno o mas casos de uso.

- La estrategia es que cada desarrollador implemente los tres paquetes de la arquitectura para sus casos de uso, construya lo modelado en la vista, el código Java de las clases del controlador y del modelo y se conecte a la base de datos.

Técnica TC1: Preparar la construcción del software

3. Establecer el estándar para la documentación del código

Un estándar de codificación es un conjunto de normas para hacer los programas más legibles.

Define las convenciones para:

- Escribir los encabezados de los paquetes y clases
- Nombrar clases, atributos, métodos y parámetros,
- Estructurar de manera legible las instrucciones de control
- Manejo de errores etc.

Un estándar para código Java se puede encontrar en el sitio de Java de Oracle

http://www.oracle.com/technetwork/java/index-135089.html

Técnica TC1: Preparar la construcción del software

- **Diseñar y homologar el prototipo de interfaz de usuario**

Se debe considerar la decisión de la herramienta con que se hará la construcción de la interfaz, html o jsp y crear la hoja de estilo para que todos puedan construir sus pantallas de manera uniforme.

- El **Responsable de calidad** personaliza la plantilla de Construcción de software (liga a la plantilla de Construcción de software).
- El **Responsable técnico**, pone una tarjeta en el Tablero en la columna de *Por Hacer*, con la actividad *Preparar la construcción del software,* mueve la tarjeta a la columna de *Haciendo* cuando lo está haciendo y mueve la tarjeta a la columna *Hecho* al terminarlo.
- El **Responsable de calidad** se asegura que se incluya en e la plantilla personalizada de *Construcción de Software*.

Actividad 2. Construir la base de datos

- **Construir la base de datos**

A partir del diseño de la base de datos y según el manejador de base de datos elegido, el Responsable técnico asigna a algún desarrollador la construcción de la base de datos.

- El **Responsable técnico**, pone una tarjeta en el Tablero en la columna de *Por Hacer*, con la actividad C*onstruir la base de datos*, asigna esta tarea a algún desarrollador.
- El desarrollador asignado mueve la tarjeta a la columna de *Haciendo* cuando lo está haciendo y mueve la tarjeta a la columna *Hecho* al terminarlo.

Actividad 3. Construir y probar el código de cada caso de uso

- Una vez que se tiene la infraestructura necesaria para la construcción con la estrategia de repartición de los componentes que cada desarrollador deberá construir, la homologación del prototipo y la base de datos construida, cada desarrollador construye su código y prueba todos sus componentes.

Técnica TC3: Construir el código de cada caso de uso (1)

- **Generar el código de cada componente**
- A partir del diseño de las clases del patrón MVC, cada desarrollador procede a codificar las que corresponden a su casos de uso.

- Para facilitar el desarrollo de las aplicaciones y en particular aplicaciones basadas en web, se hace uso de *frameworks*.

Técnica TC3: Construir el código de cada caso de uso (2)

- De forma general los frameworks MVC trabajan de la siguiente manera:
  - El navegador recibe una solicitud de entrada del usuario, se identifica que acción debe responder a ella.
  - Manda a invocar el código del Controlador que la pueda resolver.
  - El Controlador interactúa con el Modelo para obtener o guardar información solicitada.
  - El Modelo envía o recibe la información de la base de datos.
  - El Controlador recibe la información del Modelo y busca algún componente de la Vista para que prepare la presentación para el navegador.
  - El navegador despliega la información.

Técnica TC4: Probar el código de cada caso de uso (1)

- **Probar el código de cada componente**

Una vez que se tiene el código para cada componente se efectúan las pruebas de caja negra y/o transparente que hagan falta para certificar que no tiene defectos.

Si se encuentra algún defecto, se corrige y se vuelve a probar.

Técnica TC4: Probar el código de cada caso de uso (2)

- Cada desarrollador integra los componentes de su caso de uso y prueba que funcionan según lo establecido en los requerimientos (detalle del caso de uso) y el diseño (diagrama de secuencia).
- Al hacer una prueba de caja negra al caso de uso, se puede usar una tabla como la siguiente y se usan los casos de prueba definidos en los requerimientos

Caso de uso	Caso de prueba	Resultado esperado	Resultado obtenido
Registrar usuario	luso976 luisoto976@gmail.com fpEo34Sw fpEo34Sw masculino 25/09/1990 perfjl.jpg música futbol cocinar	Registro exitoso del Usuario	

Para cada caso de prueba se dan valores reales, se ejecuta el código confiando tener la respuesta esperada planteada para cada caso de prueba. Si lo obtenido no coincide con lo esperado, se corrige.

- Cada **Desarrollador** pone una tarjeta en el *Tablero* en la columna de *Por Hacer*, una con la actividad Probar el código del caso de uso, mueve la tarjeta a la columna de *Haciendo* cuando lo está haciendo y mueve la tarjeta a la columna *Hecho* al terminarlo.

Actividad 4. *Integrar el documento de la Construcción de Software*.

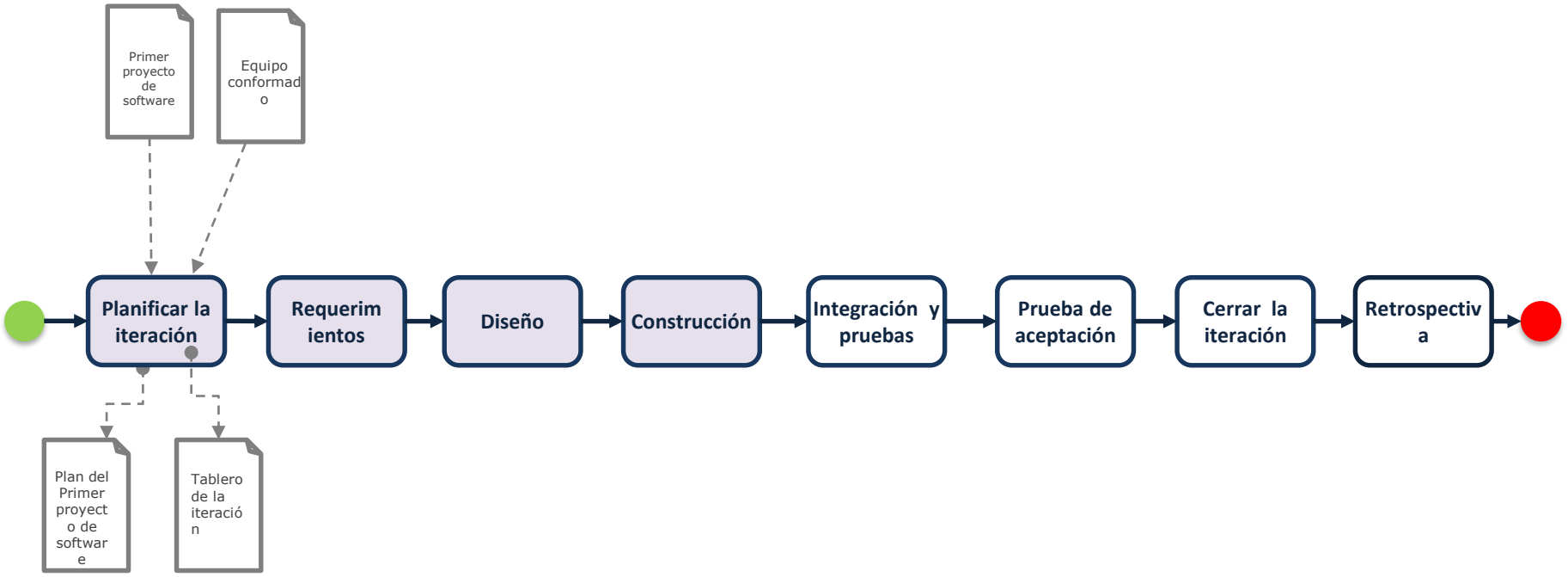
Técnica TC5. Integrar el documento de la construcción del software

- El **Responsable de calidad** unifica los productos de las actividades de la práctica en la plantilla personalizada de *Construcción de Software.*

Resultado de esta unidad

- **Condiciones**
  - Construcción del software completo
- **Productos de trabajo**
  - Código de los Componentes del Software probados individualmente
  - *Construcción de Software* completo
  - *Repositorio* compartido con la carpeta para la iteración actualizado
  - *Tablero* de la iteración actualizado

Actividades de la iteración



Bibliografía

- Binder R. V. (2000). *Testing Object-oriented Systems. Models,Ppatterns and Tools.* Addison Wesley.
- Braude E. J. (2003). *Ingeniería de software. Una perspectiva orientada a objetos.* Alfaomega.
- Flores J., V. A. (s.f.). *Paquete de Puesta en Operación de Construcción y Pruebas unitarias.*
- Howe, D. (2010). *FOLDOC: Free On-line Dictionary Of Computing*. Retrieved 2013 6-Marzo from http://foldoc.org/framework
- McCabe T.J. (Dic 1976). A software Complexity Measure. *IEEE Trans. Software Engineering, vol 2, no. 6*, 308-320.
- Sommerville I. (2011). *Software Engineering.* Pearson.
- Stelting S., M. O. (2002). *Applied Java Patterns.* Sun Microsistems/Prentice Hall.
- SWEBOK. (2004).
- Thomas D., D. H. (2007). *Agile web Development with Rails.* The Pragmatic Bookshelf.