VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

Meza Peña Augusto

No. Cuenta. 414107090

Metodologías de desarrollo de software

Metodologías Ágiles y las Tradicionales

Ingeniería de Software

Prof.: Ing. Orlando Zaldívar Zamorategui

Objetivo

Analizar y comprender las principales metodologías actuales en el desarrollo de sistemas computacionales, entender sus diferencias y sus respectivas ventajas para así utilizar el método que se adecue mejor.

Introducción

Dentro de la ingeniería de software hay dos corrientes principales para el desarrollo de software. La primera es con esquemas ordenados con planes rígidos y con actividades relacionadas ambas ya planificadas con anticipación, mientras que la segunda es más cambiante y fluida, pudiendo modificarse más fácilmente conforme avanza el proyecto. A la primera le llamamos los métodos tradicionales, y a la segunda los métodos ágiles. Se debe aclarar que no hay un método ideal, en general algunos métodos funcionan mejor en algunos casos, por ejemplo los tradicionales para los sistemas críticos, y lo que se debe buscar es un balance entre ambos métodos.

Un método de desarrollo de software incluye las actividades a seguir para desarrollar correctamente un sistema computacional, bien puede ser una representación gráfica del esquema a seguir, como en los tradicionales, o una serie de actividades y reglas a seguir, como en los ágiles. En los tradicionales a cada uno de estos esquemas lo representamos desde un punto de vista específico, con sus respectivas ventajas y desventajas; presentando información particular. A estos esquemas se les llaman modelos de desarrollo de software. Cabe aclarar que estos modelos no son descripciones formales del sistema, más bien son guías abstractas para explicar cómo funcionan los modelos.

Metodologías Tradicionales

Los métodos tradicionales son actividades relacionadas que nos llevan a desarrollar software. A pesar de la cantidad de modelos que hay, hay ciertas actividades que siempre deben estar presentes:

1. Especificación del software: Aquí es donde se definen las funciones que tendrá nuestro sistema.
2. Diseño e implementación del software: Aquí se planifica el software de tal manera que cumpla con las especificaciones.
3. Validación del software: En esta etapa se valida que el software haga lo que se le requiere hacer.
4. Evolución del software: Ya que las necesidades del cliente pueden cambiar es necesario que el software se pueda adaptar.

Claro que estos métodos llevados a la práctica sufren cambios y modificaciones puesto que involucran personas y estas no siempre deben seguir un mismo proceso. Esto, claro está, mientras no se trate del desarrollo de un sistema crítico.

Dentro de los métodos llamados genéricos, encontramos tres principales: el modelo en cascada, el desarrollo incremental y la ingeniería del software orientada a la reutilización. Si bien estos modelos están separados esto no implica que se puedan usar en conjunto. Se puede decir, por ejemplo, que si el sistema es muy grande es recomendable utilizar tanto el modelo en cascada como el incremental.

El modelo en cascada

El modelo en cascada se llama así porque pasa de una fase a otra en forma de cascada, se le conoce también como ciclo de vida de software. Fue uno de los primeros modelos en utilizarse.

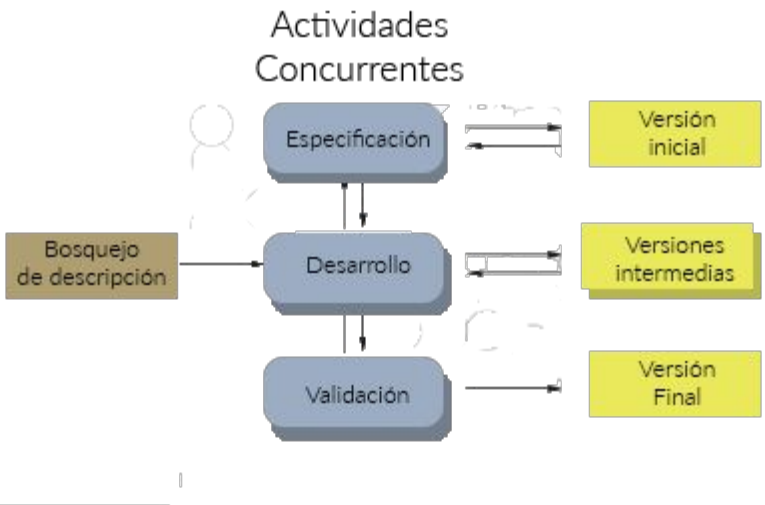
Tomando la figura anterior las etapas del modelo en cascada se dividen en:

1. Definición de requerimientos: Son las funcionalidades que tendrá el sistema, sus limitaciones y sus respectivos detalles.
2. Diseño del sistema y del software: Identificar y planificar las abstracciones del sistema y la comunicación entre sus componentes.
3. Implementación y prueba de unidad: El diseño se realiza subdividiendo en programas, llamados unidades. La prueba consiste en verificar que la unidad cumpla con los requerimientos.
4. Integración y prueba del sistema: Ahora las unidades se prueban como un sistema completo para verificar que se cumplen todos los requerimientos. Una vez terminada se entrega el producto al cliente.
5. Operación y mantenimiento: El sistema se instala y se echa a andar, se corrigen los errores que no se detectaron, y se optimiza el sistema.

En este modelo es esencial terminar la fase para poder continuar con el método. Sin embargo, en la práctica estas fases se mezclan, se detectan nuevos requerimientos en el diseño, o se encuentran errores de codificación en el mantenimiento, los métodos tradicionales no son simplemente lineales más bien retroalimentados.

Desarrollo Incremental

Este desarrollo parte de un desarrollo parcial y de ahí ir agregando las funciones, recibiendo siempre la retroalimentación del cliente. Además hay una relación directa entre las versiones y el desarrollo, especificación y validación.



El desarrollo de software incremental es más acorde a cómo se resuelven los problemas, pues comúnmente no se tiene una solución completa a un problema.

Métodos dinámicos y tradicionales - Tabla Comparativa

Métodos dinámicos (o agiles)	Métodos Tradicionales (o disciplinados)
Basado del Manifiesto Agil (Beck, et al, 2001). Tienen mayor valor para: ● individuos e interacciones ● Código funcional ● Colaboración entre clientes ● Responder al cambio	Basado en Calidad Total (SEI, 2010) Tienen mayor valor para: ● El proceso de desarrollo ● Productos de Software ● Acuerdo de clientes ● Control y planeación de proyectos
Centrada en la gente	Centrada en el proceso
Énfasis en el código y la cooperación	Énfasis en el producto y la organización
Pocos procesos con minima documentacion	Procesos bien definidos y documentados
Formalidad en el proceso reducida al mínimo	Altamente dependiente del proceso de desarrollo
Procesos no repetibles	Procesos Repetibles
Resultados dependientes de el equipo de desarrollo	Resultados predecibles
De medianos a pequeños proyectos	Grandes proyectos
Equipo reducido de desarrollo	Gran equipo de desarrollo

Resumen

Ha habido una tendencia reciente en el campo por asumir que el desarrollo diseñado de arriba hacia abajo debe asociarse con la verificación de arriba hacia abajo.
El diseño del sistema de una u otra manera no está ligada necesariamente a la forma en que lo verificaremos.
Hay casi una manera infinita de maneras en las que se puede diseñar un sistema computacional.
El diseño por fases puede ser describido de la siguiente manera:

1. Diseña, programa y verifica cada módulo por sí mismo (También llamado por unidades).
2. Modulo por modulo se va añadiendo al sistema, a esto se le conoce integración de sistemas
3. Verifica la totalidad del sistema, también llamada verificación de campo

Dicho de otra manera: son primero por unidades, verificación de subsistemas, e integración y verificación de los subsistemas.
La estrategia incremental se puede parafrasear de la siguiente manera:

1. Diseñar, programar y verificar cada módulo por sí mismo
2. Añadir otro módulo
3. Verificar y depurar la combinación de los módulos
4. Repetir los pasos 1 y 2

Debe estar claro que la decisión de verificar un sistema en un sistema por fases o incremental es enteramente dependiente de la decisión de verificar el sistema de arriba hacia abajo o de abajo hacia arriba.

Cuestionario

1. Menciona una palabra clave del objetivo.
    a. Metodologías
    b. Dinámicas
    c. Empresas
    d. Estáticas
2. ¿Cuál es el objetivo de estudiar las técnicas ?
    a. Conocer las diferentes formas de trabajo
    b. Desarrollar empresas
    c. Efectuar las técnicas de manera correcta
    d. Medir tiempos
3. ¿Cual es el objetivo secundario que es vital en el aprendizaje de los modelos de software?
    a. Los tiempos
    b. El diseño
    c. Los planteamientos
    d. Desarrollar empresas
4. Menciona una ventaja de aprender los modelos de software
    a. Saber diferenciar los diferentes métodos
    b. Facilidad de entrada al mercado laboral
    c. Crear sistemas eficientes y de mejor calidad
    d. Crear los mejores sistemas
5. Aprendiendo los modelos de software
    a. podemos diseñar cualquier sistema
    b. la verificación se sobreentiende
    c. podemos seguir un método definido para crear software
    d. la entrada al mercado laboral es más sencilla
6. ¿Cuál es el principal objetivo de estudiar los modelos de software?
    a. Crear sistemas eficientes y de calidad, siguiendo un método y con tiempos definidos.
    b. Crear sistemas perfectibles en tiempo reducido
    c. Diseñar sistemas humanos
    d. Todas las anteriores
7. ¿Porqué es esencial discutir las estrategias de implementación?
    a. Porque la tendencia actual es asumir que la estrategia en cascada es la única
    b. Para entender las demás estrategias de implementación
    c. Para implementar la mejor
    d. Todas las anteriores
8. La verificación de abajo hacia arriba esta ligada con el diseño de abajo hacia arriba
    a. Siempre
    b. En el modelo tradicional
    c. En el modelo dinámico
    d. En el modelo por fases

9. ¿Cuando el modelo de la verificación está ligada al diseño?
    a. Cuando el diseño se efectúa al mismo tiempo que la verificación
    b. Cuando el diseño es en cascada
    c. Nunca
    d. Siempre
10. Cual de los siguientes no es un modelo de desarrollo de software
    a. Dinámico
    b. Tradicional
    c. Recursivo
    d. Incremental
11. El diseño de arriba hacia abajo es diferente de
    a. El modelo de abajo hacia arriba
    b. El modelo dedicado
    c. El modelo incremental
    d. Todas las anteriores
12. Cual de los siguientes NO es un método de desarrollo de software
    a. Dinámico
    b. Clásico
    c. Determinativo
    d. Incremental
13. En que manifiesto se basan los modelos clásicos o tradicionales
    a. Calidad total
    b. UML
    c. Agil
    d. Harvard
14. Para quien representa mayores beneficios el desarrollo dinámico
    a. Empresas
    b. Desarrollador
    c. Cliente
    d. Todos los anteriores
15. Para que casos es mejor el desarrollo agil
    a. Cuando se requiere responder al cambio
    b. Cuando se requiere prototipado rápido
    c. Nunca es mejor
    d. Para todos los casos
16. Que ventaja adicional permite el desarrollo ágil
    a. El prototipado rápido
    b. Respuesta al dinamismo
    c. Colaboración entre clientes
    d. Todas las anteriores
17. Cual es el centro de el modelo agil
    a. La gente
    b. El cliente
    c. La empresa
    d. El producto

18. ¿Cual es el énfasis del modelo agil?
    a. El cliente
    b. La sintaxis
    c. Los requerimientos
    d. El código y la cooperación
19. Selecciona la afirmación verdadera
    a. El modelo tradicional es esencial para el desarrollo incremental
    b. En el desarrollo agil hay pocos procesos con mínima documentación
    c. En el desarrollo tradicional la verificación es de arriba hacia abajo
    d. Todas las anteriores
20. En el desarrollo agil la
    a. Formalidad es crucial
    b. Formalidad se omite
    c. Formalidad no es crucial
    d. Formalidad es opcional
21. En el desarrollo ágil los procesos
    a. Son repetibles
    b. Son irrepetibles
    c. Son repetibles de modo determinista
    d. Ninguna de las anteriores
22. Un detalle del desarrollo ágil es que
    a. Es propenso a errores
    b. No hay prototipado rápido
    c. Dependiente del equipo de desarrollo
    d. No hay inconvenientes en el desarrollo agil
23. El desarrollo ágil sirve para proyectos
    a. Grandes
    b. Medianos
    c. Pequeños
    d. Pequeños y medianos
24. El equipo se desarrollo para el modelo agil debe ser
    a. Dinámico
    b. Integral
    c. Pequeño
    d. Muy grande
25. En que está basado el desarrollo tradicional
    a. Modelo NACARR
    b. Calidad Total
    c. UML
    d. Agil
26. En quien está centrado el modelo tradicional
    a. Cliente
    b. Gente
    c. Proceso
    d. Calidad

27. Menciona el último paso del desarrollo por fases
    a. Verificación
    b. Verificación de campo
    c. Presentación
    d. Pulido
28. Cuando es común el diseño por fases
    a. Siempre
    b. Casi Nunca
    c. Con proyectos grandes
    d. Con proyectos pequeños
29. Es común tomar los módulos y verificarlos….
    a. Individualmente
    b. Modularmente
    c. de una sola pasada
    d. recursivamente
30. Cual es una ventaja del refinamiento por pasos
    a. Hace el código más organizado
    b. La depuración es más organizada
    c. La verificación es más organizada
    d. Todas las anteriores
31. Cuales son los pasos del desarrollo por fases
    a. Diseño, codificación, verificación, integración y verificación de campo
    b. Diseño, codificación, integración, verificación, y verificación de campo
    c. Diseño, codificación, agregar otro módulo, verificación y repetir
    d. Diseño, codificación, integración, agregar otro módulo, verificación y repetir
32. ¿Qué es verificar?
    a. El proceso de demostrar que el sistema en verdad funciona de la manera en que se supone tiene que hacerlo
    b. El proceso de demostrar que el sistema tiene los datos correctos
    c. El proceso de demostrar que el sistema t cumple con los requisitos
    d. Todas son correctas
33. ¿Qué involucra verificar?
    a. al cliente
    b. datos de prueba
    c. datos de prueba y la salida
    d. El sistema completo no solo un módulo

Bibliografía

Hossion, A. and Losada, M. (2011). Software Engineering Methods, modeling and Teaching. Colombia: Universidad de Medellín, pp.25 - 68.

Lee, R. (2013). Software engineering. [S.l.]: Atlantis Press, pp.375-195.

Schmidt, R. (2013). Software engineering. Waltham, MA: Morgan Kaufmann, an imprint of Elsevier, pp.29-41.

Sánchez Alonso, S., Sicilia Urbán, M. and Rodríguez García, D. (2012). Ingeniería del software. México, D.F.: Alfaomega, pp.223-251.