Todos navegando

(8)

Múltiples niveles de planificación - en scrum llegamos hasta la iteración.

release - depende de la organización (continuo, o cada cierto tiempo). y del producto

portfolio -

Release planning - duración de cada sprint y la cantidad (para esa release) ← que es lo que se quiere del producto.

En ágil se planifica → no hay tanto énfasis en los artefactos.

- plan realizado al momento de planear. Puede cambiar.
- El primer sprint puede tener como objetivo formular el MVP.
- cadencia - decisión del negocio: cada que tanto tenemos entrega

Sprint planning

clase 5110 — 2do parcial práctico - testing caja negra.

Unidad 4: Aseguramiento de calidad de proceso y producto

Testing de software — gestión tradicional. Testing ágil - usa le...

assurance — más vale prevenir que curar

aseguramiento de calidad - prevención: más barato que corregir
- incorporar acciones en el proceso de desarrollo
- edad del defecto: desde que se introduce un error hasta que se detecta.
  - en requerimientos - debe corregirse en el momento, sino encarece
- conjunto de acciones para evitar los errores y que estos se conviertan en defectos.
- del producto - auditorías (configuración - funcional) o revisiones de pares ~ revisiones técnicas. No las hace el jefe ni el cliente.
  - revisado con alguien que tenga las mismas capacidades técnicas en la misma jerarquía
  - se pueden hacer - formales - inspecciones de software - procesos/métricas/roles
    - informales - walkthrough (recorrida)
  - artefactos - código, -diseño, requerimientos } cualquier artefacto generado puede ser revisado técnicamente.
  - "programación de pares": uno programa otro corrige. dos por estación de trabajo.
    - detecta tempranamente muchos defectos.
    - van cambiando roles.
  - usada por el agilismo (no le gustan las auditorías). la otra manera de quality assurance.

Proceso
Proyecto
Producto
Personas

control de calidad - Testing.

XP
documentación en el código
TDD test driven development - revisión de pares.
BDD por comportamiento (behavior).

Testing - acá central e importante.
- pruebas no desactualizadas - los req. pueden y no pasa nada.
- movamos el enfoque al testing. Automatizamos lo más posible

Testeamos cuando el producto está construido. control de ~~calidad~~ calidad
- verificación - funcionamiento
- validación - cumple con req.

Lo que detectamos son DEFECTOS

PROCESO – auditorías } realizadas a nivel de proceso y luego en proyecto
└ revisiones.

PERSONAS

Aseguramiento de calidad

- del proceso – modelos de calidad – CMM → CMMI 2.0
  - ISO 9001
  - ISO 27000 – seguridad de información.
  - define el qué (descriptivos) no el cómo
  - proceso compatible con el modelo de calidad usado.
  - después comprobamos que hayamos cumplido (a nivel de proyecto). que tenga un proceso y que lo cumpla.
  - auditorías – si el proceso tiene calidad → el producto también (la calidad del mismo depende del proceso)
    - no tangible: mala interpretación
- Quién controla – alguien externo (empresas grandes: una unidad de negocio o externos) ← auditorías de calidad
  - fuera de la ejecución del proyecto.
- auditorías informáticas → peritaje: analizar sist. en funcionamiento para detectar anomalías. en producción.
  - usado en la justicia.

proyecto – espacio concreto donde las cosas ocurren.

TESTING

herramientas de soporte. producto final con la mayor calidad.

- proceso DESTRUCTIVO de tratar de encontrar defectos en el código. (cuya presencia se asume.) identifica defectos → éxito: encontrarlo.
- se debe ir con una actitud negativa para demostrar que algo es incorrecto.
- MÁS CARO 30% y 50% de la construcción del software.

solo hace el testing unitario.
- uno mismo es complicado que encuentre defectos
- NO objetivo
- realizado por otro

Asegurar vs controlar

- testing no certifica (no son todos los defectos)

ERROR y DEFECTO

- depende del momento en la que se encuentra
  - misma etapa de detección: error
  - otra etapa: defecto. → lo que encuentra el testing.

pueden provocar fallas en el sistema (no todo defecto lleva a una falla)

severidad – asociado a un concepto.
- impacto, qué tan grave es, y como influye en el sistema.

no anda el sist. / interfaz: visualización
1. Bloqueante
2. crítico
3. mayor
4. menor
5. cosmético.

} depende de la situación y lugar.

prioridad – del lado del cliente (PO). punto de vista de negocio: cuando lo necesito
- no vinculada con la severidad
- urgencia del cliente

1. Urgente
2. Alta
3. Media
4. Baja.

Niveles de prueba

- pruebas unitarias – por el desarrollador. TDD – arranca el proceso con las pruebas antes de lo probado
  - mientras más automatizado mejor
- pruebas de integración – de interfaces: si funcionan juntos (ser) si ya funcionan separados. Build
  - continous integration: un servidor prueba las integraciones.
  - del lado de los desarrolladores
- testing de sistema: o de versión. Testeamos una versión del sist. (excepto que sea en cascada).
- testing de aceptación (prueba de usuario) – realizado por el usuario. Sprint review.

parte de usabilidad de testing

UAT: user acceptance testing
workflow de usuario.

⑨

pruebas de aceptación ↔ si no hay pre → se hace en producción / en pre producción

Ambientes - separación de intereses. Los programadores no llegan a producción.
- Desarrollo - Prueba - Pre producción - Producción.

pruebas ad-hoc

Caso de prueba - paso a paso para reproducir ~~detectar~~ defectos → si no se pueden reproducir no es DEFECTO.
- condiciones o variables bajo las cuales un tester ~~determinará~~ si el SW está funcionando correctamente o no.
- no es posible probar todo
  - técnicas de prueba - ayudan a definir casos de prueba.
- "los bugs se encuentran en las esquinas y se congregan los límites".

datos concretos en el parcial.

- OBJETIVO:
- CRITERIO
- RESTRICCIÓN

Derivación de casos de prueba
- desde el código - lo que se está haciendo, no lo que debo.
- desde los requerimientos - ya definido LO que el sistema tiene que hacer.

CONDICIONES DE PRUEBA
- reacción esperada de un SUT. Frente a un estímulo particular. Constituido por las diferentes entradas.
- cada una probada por lo menos un caso de prueba.

caso de prueba
1. objetivo
2. entradas

Estrategias — cobertura
- (viendo código) caja blanca
- (sin ver código) caja negra - definimos entradas con sus salidas y comprobamos si se cumplen
  - Métodos - partición de equivalencias, análisis de valores límites } Basado en especificaciones
  - adivinanza de defectos, testing exploratorio } Basados en la experiencia.

CICLO DE TEST - ejecución de casos de prueba sobre una versión del sistema a probar.
- ideal 2 ciclos
- ciclo 0: base para identificar defectos

Regresión - sin regresión no pruebo todo, solo los defectos detectados anteriormente.
- con regresión repetimos todo, no solo los defectos detectados (lo viejo y nuevo).
  - cuando es automático es rápido, uno manualmente es lento.

Proceso de pruebas desde los requerimientos

hay defecto!

Planificación → Identificación → Especificación → Ejecución → Análisis de fallas → Fin de las pruebas.

(casos de prueba métodos personas)

DISEÑO

dejamos de probar?

Entregables – plan de prueba
- casos de prueba (datos)
- reporte de incidentes
- informe final.

cuántas líneas de código necesitamos para hacer testing → 0

en ciclo de vida

V: testeamos en todas las etapas desde los requerimientos.

desarrollo desde lo general a lo particular, pruebas de lo particular a lo general (pruebas unitarias)

Testing ≠ QA testing =

Cuánto testing es suficiente. – no podemos abrir todo pero no es excusa.
- depende – costo - beneficio (al final)
  - riesgo: criticidad del sistema
- el criterio de aceptación – acuerdo de que hacer testing → vamos en un caso de prueba.

Principios
- Presencia de defecto
- exhaustivo es imposible
- testing temprano
- agrupamiento de defecto
- paradoja del pesticida
- dependiente del contexto
- evitar revisar código propio.

Clase 2/10

Lean y kanban — Libro: lean software development. – cada libro es un principio.

parecido al manifiesto ágil. cada framework puede variar en ese.
- más viejo, Japón 1940
- surgió en industrias tangibles. (Toyota system)

compatibilidad con ágil. eliminación reducción de desperdicio

*① todo aquello que no genera valor

kanban – generación de servicios. cualquier intangible

Lean

Principios
1. *
2. amplificar aprendizaje: compartir el aprendizaje
3. Embeber la integridad conceptual: calidad del producto presente siempre
4. Diferir compromisos: hasta el último momento responsable
   - toma de decisiones = basado en información
   - postergar los compromisos hasta que tengamos más info.
   - si esperamos al momento perfecto nunca lo vamos a hacer.

   Just in time asignación de trabajo
5. Dar el poder al equipo – empoderarlo
6. Ver el todo – necesitamos tener una visión completa del proceso y producto.
   - saber donde estamos parados.
   - importante en los intangibles
7. Entregar lo antes posible – retroalimentación / feedback