INGENIERIA y CALIDAD de SOFTWARE

REVISAR CRONOGRAMA seguidamente

Clase 17/8

Técnica para requerimientos funcionales

PROCESOS EMPIRICOS

[Procesos] — [Proyectos] — [Productos]

PERSONAS
- Parte fundamental de la ingeniería de software
- Profesión humano intensiva

Se adapta al proceso pero viene definido de antemano

Procesos - definidos - PUD y RUP — planificación y control
- intentan ser "completos", todo lo necesario para el desarrollo de software.
- definido de antemano distinto de la persona que lo implementa.

empíricos - ágil (agile en) - surgió (2001) en el software. manifiesto ágil ¡LEER! — inspección adaptación transparencia
- lean (liviana esp) - toyota system
- el que toma las decisiones es el que hace el trabajo.
- basados en la experiencia (hay que conseguirla)
- los ciclos de desarrollo son cortos e iterativos

3 PILARES para la exp. basados en retroalimentación
- inspección
- adaptación — retroalimentación: experiencia → se gana con el equipo y sus particularidades.
- transparencia - visible para todos, información de todos, si hay problemas errores lo ven todos

ágil es un pensamiento → surgen frameworks ágiles → dan buenas prácticas

NO ARMAN UN PROCESO ENTERO

La mejor comunicación es la de cara a cara.

VALORES ÁGILES

(de manera frecuente)

SOBRE

individuos e interacciones	software funcionando	colaborar con el cliente	responder al cambio
↓	↓	↓	↓
procesos y herramientas	documentación extensiva.	negociación contractual	seguir un plan

una versión del producto cada poco tiempo.

PRODUCT OWNER: responsable de las user stories
- alguien que conoce el producto y el negocio
- los "no técnico": del negocio
- sabe lo que necesita el negocio. PRIORIZA.

PRINCIPIOS del manifiesto ágil

EQUIPOS auto-organizados - donde no hay jefe
↓
surgen las mejores arquitecturas y requisitos

(casi el 50% de los req.)
Requerimientos emergentes - surgen en el momento de la creación del producto

Requerimientos desconocidos - no nos damos cuenta que existen, y eran necesarios.

Entregar software de forma frecuente en entornos cambiantes. involucrar al cliente.

ÁGIL - no es metodología o proceso
- compromiso entre nada de proceso y demasiado proceso
- estará con lo que determina

Algunos frameworks
- ADD, FDD, Crystal, SCRUM, XP

ISW

entrega frecuente y temprana.

Requerimientos en Agile (de negocio, no software) - lo importante es el valor del negocio que genera el software (medio para un fin)
- más duro que req. los historia del usuario deben ser cortos y con su lenguaje.
- PRIORIZAR: "SOLO LO SUFICIENTE"
  - hay muchas cosas que no usan NUNCA
  - Pareto: el 80% del valor lo da el 20% de la funcionalidad.
- priorizamos lo más importante para el negocio (que se da en las primeras etapas de desarrollo)
  - Product backlog - escala de prioridad de característica (user stories) - las más importantes (1% del total del software)
    - cola priorizada
  - JUST in Time - describimos req. según haga falta.
    - eliminar desperdicios. (no invertir tiempo en req. que van a cambiar).
- comunicación eficiente → cara a cara
- user stories → requerimientos de negocio y usuario
- tradicional vs ágil

TRADICIONAL	ÁGIL
FIJO - requisitos	FIJO - recursos, tiempo
ESTIMADO - recursos, tiempo	estimados - alcance
dirigidos por un plan	dirigido por valor.

fecha fin no se corre, recursos comprometidos en el tiempo

- principios ágiles relacionados: (1) (2) (4) (6) (11)

Fred Books: NO silver bullets → ESENCIAL — The mythical Man month

USER STORIES - la parte más difícil de hacer SOFTWARE son los requerimientos
- no hay solución perfecta para el ~~programa~~ problema
- incertidumbre / cambio
- valor al cliente
- descripción corta de la funcionalidad
- DEFINE PARTES del PRODUCTO

PARTES 3C - Tarjeta (card) - lo que queda - EL FRENTE
- "Como <rol> (who) yo puedo <act.> (what) de forma tal que <valor de negocio que recibo> (why)"
- no se especifica el cómo (libertad)
- ej "como conductor quiero buscar un destino (what) a partir de una calle y altura para poder llegar a tiempo (why) sin perderme"
- Conversación - importante desarrolla los detalles de la historia
  - quedan reflejados en las pruebas de aceptación y en la tarjeta DORSO
- confirmación - lo que queda en la conversación
  - demostración del funcionamiento de la característica - cuando una user está completa
  - Frase verbal nombre corto relacionado con el qué - destino por dirección

son multipropósito - necesidad del usuario / descripción del producto - ítems de planificación - recordatorio de conversación - código + US = documentación del producto.

product owner prioriza las user stories en el product backlog
- el tamaño depende de cuantas iteraciones son necesarias para completarlas.
- porciones verticales - con una parte de cada capa desde UI hasta BD
- modelado de roles - importante, también determinado con tarjetas.
- usuarios representativos (proxies) - cuando el PO no está disponible
  - evitarlo cuando sea posible

Criterios de aceptación de user stories - formalización para definir las pruebas de aceptación - user proof
- definen una interfaz no una solución

los detalles quedan en la conversación y, en la documentación y en las pruebas.

(2)

- debe incluir los criterios de aceptación.
- Pruebas de aceptación - hay positivas (pasan) y negativas (fallan)
  - acuerdo, si funciona lo acordado se acepta la user.
- Definition of Ready DoR
- definición de listo - control de calidad
  - cuando está lista para ser incluida en el sprint backlog.
  - aplicado a la user. Apto para ser integrado en una iteración — Do Ready
  - definido por el equipo
- ~~Definition of Done~~ → una buena user stories
  - determina si la user está terminada. completa y correcta. El PO define si pasa o producción.
  - Definition of Done
  - INVEST (INVEST)
    - Independent - puede implementarse en cualquier orden
    - negotiable - historia definida en "qué" no el "cómo". conversación.
    - valuable - debe tener valor para el negocio (no al desarrollo, tecnologías)
    - estimable - cuando puede asignarle un número (costos) Tamaño, complejidad y esfuerzo.
    - small - consumible en una iteración, comienza y termina en la iteración (sprint) → tamaño relativo al equipo y el tiempo.
    - testable - demostrar que fueron implementadas
- Algo más
  - NO ERS
    - no son especificaciones detalladas de requerimientos
  - niveles de granularidad (abstracción) — mastering agile software - Mike cohn
    - epic - user story grande, hay descomponerla en user más pequeñas. (dejamos para después, prioridad baja).
    - theme - colección de user stories relacionadas
    - user story: descripción de una funcionalidad desde la perspectiva del cliente o usuario

Clase 24/08

Requerimientos ágiles

- no establecidos desde un comienzo
- FOCO: valor del negocio. Software como medio de entrega de valor
- partimos desde una "visión" de producto → define la primera versión del producto.
  - product backlog: nunca está completo
    - necesitamos una colección de historias de comienzo
    - apunta a recibir cambios hasta en etapas finales.
    - no podemos tener una visión final del producto al comienzo. No definimos de antemano
    - pila o cola PRIORIZADA → fn. del product owner (que forma parte del negocio)
  - JUST IN TIME: diferir decisiones
    - detallar solo lo que hace falta en el momento

- comunicación cara a cara
- mejores requerimientos
- equipos auto organizados

User stories - usada en [XP / Scrum (adoptada, no nativa)]

- gestión binaria = está terminada o no: No hay porcentaje de avance completitud → TABLEROS (visible para todo el equipo)
  - visualización del trabajo
  - 3 columnas
    - to do
    - doing
    - done
- vs gestión tradicional - porcentaje (gantt) de avance: realmente no se puede medir
- por esto hacemos user stories, para poder entregar valor rápidamente. → SPRINT BACKLOG

- el producto tiene calidad cuando cumple con los RQ.

- definition of done - define cuando historia se puede mostrar al product owner.
  - definido por el equipo de desarrollo.
- expectativas y necesidades.