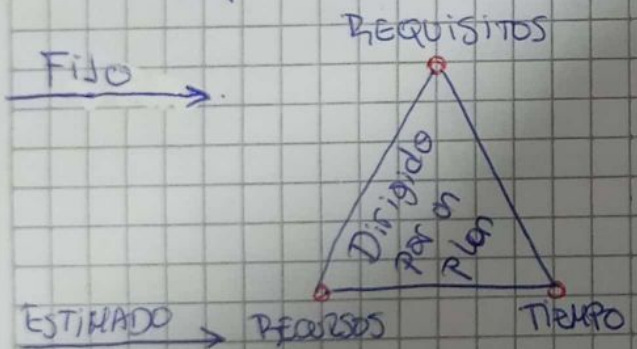
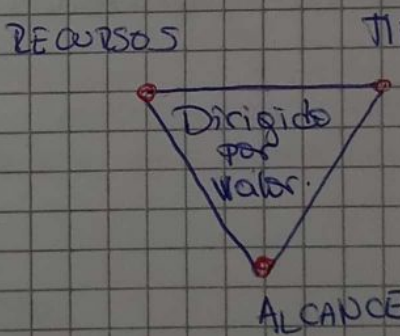
REQUERIMIENTOS EN AMBIENTES ÁGILES. USER STORIES.

- Construir el producto correcto que le de valor al negocio.
- Determinar qué es "solo lo suficiente" para comenzar.
- User historios y modelos para mostrar qué construir.
- El mejor medio de comunicación es la comunicación cara a cara.
- Cliente disponible → Ni antes, ni después.
- JUST IN TIME → Analice cuando lo necesite, no antes.
  ↳ Descartar desperdicios
- Permite que fluya la info. vocal, subvocal, gestual con retroalimentación.
- El pto. máx. ocurre cuando 2 personas se encuentran co-ubicados.

Enfoque tradicional

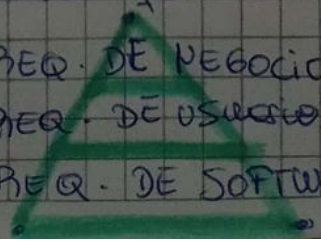


Enfoque ágil.



- La gestión de requerimientos de este enfoque, trabaja a nivel de:
  → REQ. DE NEGOCIO
  → REQ. DE USUARIO

- US = Req. de usuario alineado al Req. de negocio.
  = Necesidades de usuario que satisfacen las necesidades del negocio



Deben trabajar juntos técnicos y no técnicos, entendiendo los necesidades y el negocio, descubriendo la solución de forma colaborativa.

Junto a un equipo motivado y competente, entregamos frecuentemente valor a los stakeholders, recibiendo un feedback que nos ayuda a optimizar el product backlog.

- El PO es el representante de todo lo que tengan que decir los stakeholders del negocio.

Principios Ágiles Relacionados a los Requerimientos Ágiles:

1. La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a 1 mes).
2. Recibir cambios de requer., aun en etapas finales.
4. Técnicos y no técnicos trabajando juntos todo el proyecto.
6. El medio de comunicación por excelencia es cara a cara.
11. Los mejores arquitectos, diseños y requerim., emergen de equipos autoorganizados.

USER STORIES

Es una descripción corta de una necesidad que tiene el usuario respecto de un producto de SW.

PARTES DE UNA US → 3C
- Conversación → la + + importante
- Card (Tarjeta)
- Confirmación

forma de expresarla: Como <nombre del rol>, quiero <actividad> para <valor de negocio que recibe>

sintaxis:
- <nombre del rol> → Representa quién está realizando la actividad.
- <actividad> → Representa la acción que realiza el sist.
- <valor de negocio> → Comunica por qué es necesario la actividad.

As Who, I want what so that why

Frase verbal = forma Corta de referenciar la US.

⇒ Los US son:
- Una necesidad del usuario
- Una descripción del producto.
- Un ítem de planificación
- Token para una Conversación (Recordatorio de que hay que hablar con el clie
- Mecanismo para diferir una Conversación

NO SON una especificación de requerimiento de SW.

El PO prioriza los US en el Product Backlog.

Dueño del

Las US son porciones Verticales:

Story 1	Story 2
GUI	
Business Logic	
Database	

USUARIOS REPRESENTANTES (PROXIES) → Roles que cubren al PO cuando éste no puede asumir el rol.
- No técnicos
- Son personas del negocio.
- No es la situación ideal, pero a veces no queda otro.

→ Gerentes de Usuarios
→ Gerentes de desarrollo
→ Alguien del grupo de Kanban.
→ Vendedores
→ Expertos del dominio.
→ Clientes
→ Capacitadores y personal de Soporte.

→ CRITERIOS DE ACEPTACIÓN de US: Info concreta que nos sirve para saber si lo que implementamos es correcto.

→ Definen límites para una US
→ Ayudan a que los PO respondan lo q' necesitan para que la US provea valor (RF mínimos).
→ Ayuda a que el equipo tenga una visión compartida de la US.
→ Ayuda a desarrolladores y testers a derivar las pruebas.
→ Ayudan a los desarrolladores a saber cuando parar de agregar funcionalidades.

¿Cuando son buenos los C.a? → Definen una intención, no una solución.
→ Son independientes de la implementación.
→ Relativamente de alto nivel, no es necesario que se escriba c/detalle.

Y entonces ¿Donde van los detalles?
Por ej: El formato del saldo es 999.999.999,99
: Usar lista desplegable, usar checkbox.

→ Son el resultado de las Conversaciones con el PO.
Se pueden capturar en dos lugares:
- Documentación interna de los equipos.
- Pruebas de aceptación automatizadas.

PRUEBAS DE ACEPTACIÓN → Sintaxis = Probar + frase verbal
→ Expresan detalles resultantes de la conversación.
→ Complementan la US.

Definition of Ready = Una medida de calidad que construye el equipo en su conjunto para poder determinar que la US está en condiciones de entrar a una iteración de desarrollo.

↓

Está lista la US para empezar a desarrollarla?

→ Lo define el equipo.

→ Tiene que tener de mínimo lo que plantea el invest model:

- **I**ndependent → El PO los prioriza P/Q se pueden desarrollar en cualquier orden.
- **N**egotiable → el "qué", no el "cómo".
- **V**aluable → debe estar el "para qué" = valor de negocio
- **E**stimable → asignarle un peso.
- **S**mall → debe ser "consumido" en una iteración. Se empieza y se termina
- **T**estable → poder Demostrar q' la US se implementó cumpliendo los C.A q' se definieron.

- Al INVEST model, el equipo le puede agregar más detalles para definir el READY.

ESTIMACIONES DE SOFTWARE.

↳ No es precisa

↳ No es planear pero son la base de los planes.

↳ A mayor dif entre plan y estimación ⇒ << Riesgo.

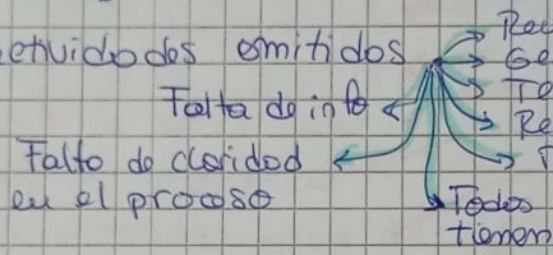
↳ No son compromisos

→ ++ Info ⇒ -- Incertidumbre ⇒ -- Riesgos.

¿Para qué estimamos?

Errores de estimación → Universo de incertidumbre

↳ Fuentes de error: actividades omitidos



Desafío = Encontrar una manera de medir tamaño - Esfuerzo.

MÉTODOS TRADICIONALES = Basados en la experiencia. Empíricos.

↳ Datos históricos = recopilar info de proyectos anteriores y elijo datos como referencia para nuevos proyectos: - tecnología
- perfiles IT
- esfuerzo en hs, etc -

↳ Juicio EXPERTO = PURO → lo estudia un experto
↳ Experiencia

↳ Juicio EXPERTO = DELPHI → Grupo de expertos con distintos tipos de expertices.
↳ Similar al poker planning

↳ ANALOGÍA → Compararlo con proyectos de similares características, en los cuales se pudo haber trabajado o no

TIPS SOBRE ESTIMACIONES
- No utilizarlos como compromisos.
- Lo más valioso es el "proceso de estimar"
- pueden servir para saber si el trabajo planificado es factible o no
- Puede servir como protección para el equipo.
- No hacer estimaciones tempranas → Estimar solo cuando es necesario. JUST IN TIME
- Estimaciones relativos → Son mejores si los comparamos

¿QUÉ ESTIMAMOS?
- Complejidad
- Esfuerzo
- Grado de incertidumbre

→ TAMAÑO NO ES ESFUERZO

→ ESFUERZO NO ES CALENDARIO → Las estimaciones basadas en tiempo = ++ errores

POKER ESTIMATION: ¿Por qué uso escala exponencial?
→ Para el SW tiene un crecimiento, en complejidad, exponencial.

VELOCIDAD = Métrica más importante ágil.
- Métrica de producto.
- No se estima → Se calcula al final del sprint.
  - Se suman los puntos de c/ US que el PO aceptó
- Sirve para ver si logré el ppio: Desarrollo Sostenible
  - Da previsibilidad al equipo.

CRISIS DEL SOFTWARE → Dificultad fundamental y persistente en la creación de sw de alto calidad, de manera eficiente y efectiva.

Con cada avances surgen nuevos problemas.

PROBLEMAS

- COMPLEJIDAD
- CAMBIO CONSTANTE
- CONFORMIDAD CON LAS ESPECIFICACIONES
- PRODUCTIVIDAD LIMITADA

No existe una "Bala de plata" o una solución milagrosa q' pueda resolverlos.
La mejora proviene de avances incrementales en la gestión de proyectos.

CICLO DE VIDA

DEL PROCESO

Define con qué actividades vamos a trabajar y cómo los vamos a implementar.

↓

Énfasis en la organización, planificación, seguimiento y mejora continua de los procesos.

PARA UN PROYECTO

- Secuencial
- Recursivo
- Iterativo

↓

SCRUM

- Con duración fija
- Con alcance fijo

↓

Define la duración en función del alcance.

En el plan de proyecto se elige con qué ciclo de vida trabajar.

DEL PRODUCTO

Centrado en el desarrollo y la evolución de un producto desde su concepción hasta su obsolencia.

↓

Énfasis en los etapas del C. de vida de producto

- Idea inicial
- Planificación
- Entrega
- Mantenimiento
- Actualizaciones

AUDITORÍAS EN GESTIÓN DE LA CONFIGURACIÓN DE SW (SCM).

Garantizas la integridad, calidad y la trazabilidad de los componentes de SW a lo largo de su C. de V.

- **FÍSICA** → Constatar que lo que hice en el plan de proyecto, es lo que hice.
  - **VERIFICACIÓN** → Construir el producto CORRECTAMENTE
  - Trazabilidad con su ubicación física, documentación asociada y estado
- **FUNCIONAL** → Revisar en términos de los requerimientos funcionales.
  - **VALIDACIÓN** → Construir el producto **CORRECTO**
  - Trazabilidad con los requerimientos, pruebas y funcionalidad del SW.