Clase Teórica de ISW 27/03

27/03/20

Requerimientos y Manifiesto Ágil

Cuando hablamos de requerimientos ágiles, se habla de problema, de una necesidad de negocio, alguna oportunidad y cómo nosotros como equipo entregamos la solución que pueda satisfacer una necesidad.

Los valores y los principios que se ven en el manifiesto se ven reflejados en la manera en que se tratan los requerimientos en este manifiesto ágil.

Valores claves del manifiesto Ágil

✓ Individuals and Interactions - over processes and tools

Los individuos y las interacciones con los individuos tienen más prioridad que los procesos y las herramientas.

✓ Working software - over comprehensive documentation

El software funcionando es más importante que la documentación, eso no significa que no se valla a documentar sino que la principal medida de progreso es el software funcionando y a eso le le da prioridad.

✓ Customer collaboration - over contract negotiation

La colaboración con los clientes por sobre los planes o negociaciones contractuales tienen mayor importancia y poder responder a los cambios por sobre

NOTA

un plan.

✓ Responding to change - over following a plan

Poder responder a los cambios por sobre un plan. La posibilidad de poder responder a los cambios es una clave del manifiesto ágil.

12 Principios del Manifiesto Ágil

① Satisfacer al cliente con entregas frecuentes y tempranas: entregarle al cliente, no esperar como en los clásicos ciclos de vida encerrada a resolver el producto de manera completa sino tener pequeñas entregas frecuentes al cliente, no solamente para darle valor al cliente en poco tiempo sino además para obtener un feedback acerca del cliente para ver si lo que le entrega es lo que necesita o si hay que hacer cambios

② Cambios de requerimientos son bienvenidos: a diferencia de los ciclos de vida donde se especificaban y definían todos los requerimientos y se tenía un acuerdo con el cliente para que estos requerimientos no fueran cambiados hasta q el producto no fuera entregado, acá se plantea que si hay cambios de requerimientos y eso tiene que ver con el feedback del cliente, esos cambios deben ser aceptados y tratados y dará una mayor satisfacción poder tratarlos. El contexto de los frameworks ágiles nos permiten de alguna manera ordenar como podemos tratar esos cambios de requerimientos. Esto ayuda a que los cambios sean más económicos y además no tener costos innecesarios en algo que no sirve.

③ Release frecuentes (de 2 a 4 semanas): relacionado con las entregas frecuentes, poder entregar en poco período de tiempo. El marco de tiempo es acotado, entre 2 y 4 semanas.

④ Técnicos y no técnicos juntos: que sea un equipo multidisciplinario, donde los técnicos y no técnicos trabajen colaborativamente

Cuando se arma el equipo, que va y viene con el cliente y en terminos del equipo técnico trabaja y construye el software es distinto ya que acá se plantea que el equipo colaborativo y el rol del Product Owner es parte de ese equipo, entonces lo que va surgiendo con las entregas frecuentes y lo que va surgiendo con los cambios de requerimientos es un proceso que surge naturalmente de ese intercambio que ocurre en el equipo.

(5) Individuos motivados: el equipo tiene que ser motivado porque la premisa de la mayoría de los frameworks agiles es de equipos que se autogestionan y puedan ser productivos, para esto es fundamental que los individuos dentro del equipo estén motivados.

(6) Medio comunicación cara a cara: relacionado a los cambios, a la forma con la que interactuamos con todo el equipo, sean técnicos o no técnicos ya que los equipos son normalmente pequeños y ahorrando la comunicación cara a cara

(7) Métrica de progreso: software funcionando: relacionado a los valores. Es la mejor medida de progreso, nos mide cuanto va construyendo nuestro equipo.

(8) Ritmo de desarrollo sostenible: las entregas o release frecuentes implican ir entregando una cantidad de software funcionando que implique un ritmo que se sostenga en el tiempo

(9) Atención continua a la excelencia técnica: priorizar la excelencia técnica, arquitectura, etc a la hora de construir el software son puntos claves con lo que tienen que ver con la calidad del software que se construye

(10) Simplicidad: Maximización del trabajo no hecho: demora: lo que más podemos lo que también no necesitamos construir construir el software

con la menos funcionalidad posible, lo que sea excedente a la entrega no se hace ya que puede cambiar y puede ser una funcionalidad que no sea necesaria.

(11) Arquitectura, diseños y requerimientos emergentes relacionado a lo anterior.

(12) A intervalos regulares el equipo evalúa su desempeño relacionado al principio nº (6)

Requerimientos en Agile

Usar el "valor" para construir el producto correcto

Relacionado con el principio de maximizar el trabajo no hecho

Usar historias y modelos para mostrar que construir

Une el eslabón de la cadena. Especificar los requerimientos en agile. Se va a modelar con historias y modelos

Determinar que el "Solo lo suficiente"

Relacionado con el principio de maximizar el trabajo no hecho

El costo del tradicional BRUF

Los productos "Exitosos" tambien tienen un desperdicio significante

Métricas

Cantidad de funcionalidad que se desperdicia cuando se trabaja con BRUF.

ISW — TEORICO — 9 — 27/03

Brut - especificación de los grandes requerimientos al principio cuanto es la funcionalidad que gasto y construyo y no se usa?

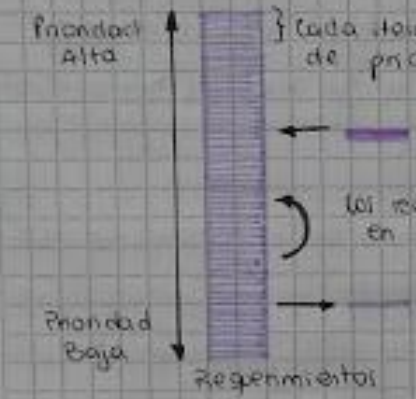
45 % de la funcionalidad no se usa nunca

19 % se usa muy pocas veces

En términos de desperdicio es muy importante sobre todo porque esta métrica esta basada en productos de software que son exitosos, es decir que el desperdicio ocurre tanto en sistemas comunes como exitosos.

Gestión Ágil de Requerimientos de software

Los requisitos cambiantes son una ventaja competitiva si puede actuar sobre ellos



El cambio dentro de los principios es bienvenido ya que es una venta ja competitiva por saber un contexto donde yo a ese cambio tengo que esperar de terminar de construir ese producto para poder introducir ese cambio.

Hay una manera organizada de gestionar los requisitos cambiantes, que implica pensar que hay una pila de requerimientos en donde lo que ponemos arriba son los que tienen mayor prioridad, es decir le da mayor valor al negocio lo que va arriba de la pila y lo va a

ordenar de mayor a menor en función de su prioridad

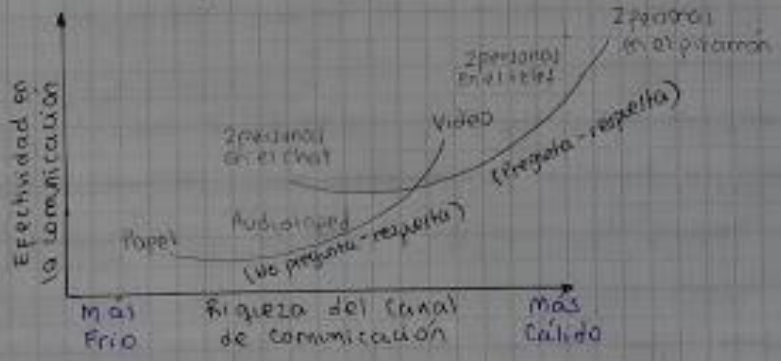
Se trabaja con los requerimientos de prioridad alta y se trabaja por partecitas de grupo de requerimientos lo que implica que si tengo nuevos requerimientos, cambian las prioridades o quiero hacer modificaciones sobre cosas que parecen importantes, todo lo que viene hacia abajo de la pila puede ser modificado sin ningún inconveniente. Se van a hacer entregas pequeñas y concretas que van a abarcar los requerimientos de de mayor prioridad.

Just in time → Analice cuando lo necesite no antes

Esto hace referencia a permitir hacer cambios, pensando en los requerimientos cuando llegue el momento. Solo se los piensa al principio cuando son requerimientos de mayor prioridad y están arriba de la pila. Aquello que no es relevante se lo piensa y analiza cuando llegue el momento. Si no se necesita no me voy a poner a hacer ningún análisis (maximización de trabajo no hecho).

El cara-a-cara permite que fluya información

vocal, subvocal, gestual con realimentación rápida



El mejor mecanismo de comunicación es cara-a-cara. Todo lo que tenga que ver con usar un canal de comunicación con más efectividad está relacionado directamente a la manera que fluye la infor...

mación con la comunicación cara-a-cara.

Se pueden tener 2 personas en un pizarron, en el medio algún video pero claramente lo que mayores dificultades presenta a la hora de establecer un canal de comunicación es lo que tiene que ver con la forma tradicional de escribir en un papel los requerimientos y entregar ese papel para que nuestro cliente o usuario lo valide. Ese mecanismo es el tradicional para hacerlo pero no permite fluidez de información, no permite la retroalimentación con el cliente rápida, cosa que es fundamental en este contexto

> "Valor es la obtención de beneficio tangible o intangible".
>
> Maia Maeda, Jenous LEAP

> "El valor lo asociamos a la utilidad, beneficio o satisfacción que le ofrecés a los usuarios finales por cada funcionalidad completa que le entregás".
>
> Pablo Lukinsky, Agile Trainer & Consultant, Entrepreneur.

Hay que priorizar lo que nos da más valor. El valor es la satisfacción que le ofrece a el usuario final y el beneficio que tiene para el negocio. El que nos puede decir el valor es quien está del lado del negocio

Tradicional vs Ágil

	Tradicional	Ágil
Prioridad	Cumplir el plan	Entregar valor
Enfoque	Ejecución ¿cómo?	Estrategia (¿Porque? ¿Para que?)
Definición	Detallados y cerrados. Desarrollamientos al inicio	Esbozados y evolutivos - Desarrollamiento progresivo.
Participación	Sponsor, stakeholder de mayor poder e interés.	Colaborativo con stakeholder de mayor interés (clientes, usuarios finales).
Equipo	Analista de negocio, Project	Equipo multidisciplinario

NOTA

	Manager y áreas de Proceso	
Herramientas	Entrevistas, observación y formularios	Principalmente participativa técnicas de facilitación para descubrir
Documentación	Alto nivel de detalle - Matriz de Rastreabilidad para los requerimientos	Historias de usuario Mapeo de historias (Story Mapping)
Productos	Definidos en alcance	Identificados progresivamente
Procesos	Firmes, no están al cambio	Incertidumbre, abierto al cambio.

Como las organizaciones están preparadas para afrontar la metodología ágil.

En el tradicional se tiene requisitos (alcance) del producto que se va a construir y se arma un plan y en función de ese alcance estimamos los recursos y el tiempo. A nivel organizacional, la ventaja es que vos me decís que producto querés construir y yo te digo cuanta plata va a salir y en que tiempo va a estar listo

En el Ágil, yo te doy recursos y tiempo y en función de lo que nos agrega más valor a mi producto se va definiendo ese alcance en conjunto. Si se aprovechan bien los recursos y el tiempo, el producto a construir nos dará más valor. Si se trabaja en ágil, una de las premisas es que alguien de mi cliente esta involucrado en mi equipo y tiene que haber una forma de trabajar diferente

ISW

TEORICO

27/03

Tipos de Requerimientos (4 tipos)

Requerimiento de Negocio

Disminuir en x% de tiempo invertido en procesos manuales relacionados con atención al cliente.

↳ El objetivo o la meta de negocio que motiva a cualquier cliente espera lograr.

Requerimiento de Usuario

Realizar consultas en línea del estado de cuenta de los clientes.

Requerimiento Funcional

Generar reporte de saldos de cuenta. Recibir notificaciones por email.

↳ Tareas de los req. del usuario, sabes que debemos hacer el sistema para resolver los requerimientos de usuario

Requerimiento No Funcional

Formato del reporte PDF. Cumplir niveles de seguridad para credenciales de usuarios según la ley bancaria 9999xx.

↳ Cosas del sistema que no están ligadas con la funcionalidad sino con restricciones

Requerimiento de Implementación

Servidores en la nube

↳ Uso de una determinada BD o servidores en la nube.

Que se va a hacer?

C

Entendiendo la necesidad y el negocio con sus requerimientos, que es lo que le dará valor al producto de software que estamos construyendo. A partir de eso se piensa una solución junto con un equipo automatizado y competente, y apartir de esto podemos entregar tempranal a los stakeholder.

Aspectos claves

- Los cambios son la única constante → apropiarse del cambio en vez de negarlo
- Stake holders no son todos los que están → son como los involucrados en el proyecto es importante identificar todos los stakeholders que aparecen. Muy útil usar técnicas como el prototipado, etc

NOTA

Siempre el foco es la entrega de valor

- Siempre se cumple eso de que: "El usuario dice lo que quiere cuando recibe lo que pidió."
- No hay técnicas ni herramientas que sirvan para todos los casos.
- Lo importante no es entregar una salida, un requerimiento, lo importante es entregar un resultado, una solución de "valor".

Principios relacionados a los Requerimientos Ágiles

Principios ágiles → son los que más se van a reflejar cuando hablamos de requerimientos ágiles (los que más impacto tienen en temas de requerimientos)

| los que más impactos tienen

1) La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a un mes).
2) Recibir cambios de requerimientos, aún 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) Las mejores arquitecturas, diseños y requerimientos emergentes de equipos autoorganizados

Principios técnicos → ellos relacionados con ágiles solo que enmarcados de alguna manera.

6) Ver el todo → relacionado a las arquitecturas y al diseño de todo el producto
4) Diferir el compromiso → relacionado a posponer a pedir en lo que necesitamos en el momento que lo necesitamos y no antes
1) Eliminar el desperdicio → trabajar sobre los requerimientos esenciales y no sobre todos

TEORICO

FECHA 27/03

USER STORY

Es una manera de trabajar con requerimientos ágiles. Se llama user story porque se cuenta una historia de usuario + es una técnica

"la parte más difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados. Ninguna otra parte del trabajo afecta tanto el sistema resultante si se hace incorrectamente. Ninguna otra parte es tan difícil de rectificar más adelante"

"Not Silver Bullet - Essence and Accidents of Software Engineering"

↓

Una de las cosas más difíciles es establecer los requerimientos correctos.

Cuáles son las partes de una User Story

Las partes de la user se denominan como regla nemotécnica de las 3 C

Confirmación - Conversación - Card

La parte más importante es la conversación

La card / Tarjeta tiene sus partes: el frente (Front of card), la parte trasera (Back of card)

Parte de la tarjeta: es la forma de expresar las historias de usuario. Se usa la siguiente regla:

Como <nombre del rol> → Representa quien está realizando la acción o quien recibe el valor de la actividad.

yo puedo <actividad> → Representa la acción que realizará el sistema

NOTA

de forma tal que <valor de negocio que recibo> → (faint) porque es necesaria la actividad

<u>la user stories son multipropósito</u>

las historias son (para que me sirven) →
- una necesidad del usuario
- una descripción del producto
- un ítem de planificación
- token para una conversación
- mecanismo para diferir una conversación.

<u>Como se trabaja con esas historias de usuario?</u>

El Product Owner prioriza las historias en el Product Backlog

Prioridad alta

Modelado con mayor detalle ↗

Modelado con menor detalle ↘

Prioridad Baja

El Product Owner coloca lo de mayor prioridad arriba y lo de baja prioridad abajo. lo que está arriba está modelado con mayor nivel de detalle y es con lo que se empieza a trabajar. los de abajo no me preocupo en definirlo con mayor detalle.

El que define las necesidades es el product owner.

Porciones verticales → del software, donde las users stories incluyen todo. Trabajar con todos los lugares donde esa user impacta, y construir porciones verticales de las user stories

	Story 1	Story 2
GUI		
Business logic		
Database		

↙ Donde impacta fuertemente esto??

ISW TEORICO Fecha 27/09

Modelado de Roles

Poder identificar los roles y modelarlos para identificar como se comportan esos roles.

Tarjeta del Rol de usuario ejemplo

Rol de usuario: Reclutador interno
No es un experto en computadoras, pero bastante adepto a utilizar la web. utilizará el software con poca frecuencia pero muy intensamente ...

El rol se describe, imaginando como usará el sistema.

Tecnicas adicionales para modelar roles

1) Hacerlo de forma genérica a través de un rol

2) Ponerle el nombre de una persona, no porque esa persona sea todo lo que desempeña ese rol, sino que ponerle un nombre y pensarlo como en una persona concreta q' tendrá sus características generales y nos ayuda a poder identificarlas.

3) Encontrar algunos roles extremos.

Tipos de usuarios

- Usuarios representantes (Proxies): pueden ser roles que se pueden identificar pero no hay que poner un nombre genérico es algo que es muy específico. No son como los usuarios verdaderos y es mejor entablar ejemplo: Gerentes de usuarios, Gerentes de desarrolladores, Alguien del grupo de marketing, vendedores, expertos del dominio, Clientes, capacitadores y Personal de soporte.

USER STORY - un ejemplo de tarjeta

Buscar destino por Direccion

Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

Criterios de Aceptación

- La altura de la calle es un numero
- La busqueda no puede demorar más de 30 segundos.

Los criterios de aceptación son aquellos requisitos o condiciones que son claves para poder ponernos de acuerdo en que cosas va a hacer esa user cuando se la construya

Los criterios de aceptación son condiciones de satisfacción que se expresan en la user story. Definen límites para una user story. Ayudan a que los PO respondan lo que necesitan para que la user provea valor (requerimientos funcionales mínimos). Ayudan a que el equipo tenga una visión compartida de la US. Ayudan a desarrolladores y testers a derivar las pruebas. Ayudan a los desarrolladores a saber cuando parar de agregar funcionalidad en una US.

¿Cuales son los criterios de Aceptación buenos?

- No dicen como resolver la user, solamente definen una intención.
- Son independientes de la implementación.
- Relativamente de alto nivel, no es necesario que se escriba cada detalle.

¿Y los detalles donde van?

Los detalles van en distintos lugares. Todos los detalles son parte del resultado de las conversaciones con el PO y el equipo puede capturar los en 2 lugares:

- Documentación interna de los equipos.
- Pruebas de aceptación automatizadas.

Los detalles de la user no van escritos en la user.

ISW | TEÓRICO | Hoja Nº 14 | Fecha 27/03

Parte trasera de la card:

En esta parte están las denominadas Pruebas de aceptación de historias de usuario.

A partir de los criterios de aceptación definidos en la historia voy a definir las pruebas de aceptación, es decir, cuando el PO se siente a ver la historia, en función de lo que se enunció y de los criterios de aceptación que cree me parecen que él le gustaría aprobar.

Las Pruebas de aceptación complementan a la user. Expresan detalles resultantes de la conversación. El proceso en 2 pasos:

- Identificarlas al dorso de la US.
- Diseñar las pruebas completas.

Ejemplo de User

Parte delantera

Como compañía quiero pagar por una búsqueda de puestos con una tarjeta de crédito, así resuelvo mi necesidad en forma más eficiente

Criterios de aceptación:

- Se acepta Visa, mastercard y American Express
- En compras mayores de $100 se piden el nº del dorso de la tarjeta.

Parte Trasera

Probar con Visa (pasa)
Probar con mastercard (pasa)
Probar con American Express (pasa)
Probar con Dinner's Club (falla)
Probar con nº de tarjetas válidas
" " " " " malas
" " " " " faltantes
" " tarjetas vencidas
Probar con montos menores a $100
" " " mayores " "

Definición de Hecho - Definition of Done

Es un acuerdo que se hace dentro del equipo acerca de cuales son las condiciones que una US tiene que tener para considerar que está hecha (Done) y ya puedo incluirla en la demo para mostrarsela al PO. Una vez que el equipo define que está lista la US, se la muestra al PO.

Definición de Listo - Definition of Ready

Tiene que ver con definir cuando una historia está lista en términos de definición y detalle para poder ser incluida en un Sprint para poder empezar a desarrollarla. Las US se van definiendo con el detalle que fuesmos necesitando a medida que se va avanzando pero para incluirla en un sprint o iteración tiene que tener un nivel de detalle suficiente acordado por el equipo y tiene que cumplirse para empezar a trabajarla

Algo más sobre las US

No son especificaciones detalladas de requerimientos (como los casos de uso)
Son expresiones de intención, "es necesario que haga algo como esto..."
No están detallados al principio del proyecto, elaborados entando especificaciones anticipadas, demoras en el desarrollo, carentes de requerimientos y una definición limitada de la solución
Necesita poco o nulo mantenimiento y puede descartarse después de la implementación.
Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

27/03

Diferentes niveles de abstracción

Otra forma de definir las US. Dentro de los niveles de abstracción están las US, los temas (Theme) y las Épicas (Epic).

Las Épicas son una historia de usuario grande, lo que significa que en algún momento las voy a detallar y convertir en muchas US. Una épica nunca cumplirá con el criterio de Ready por ser una historia grande que no se puede incluir en un sprint.

Un tema es una colección de historias relacionadas, donde en algún momento deberán desmenuzarse

Para poder definir entre esos niveles de abstracción y llegar al criterio de Ready se usa **INVEST Model**. Esto nos modela que tiene que cumplir una US para poder considerar que ya está lista y que se puede incluir adentro de un sprint.

INVEST model

- **Independent**: la US debe ser implementable en cualquier orden sin depender una de otra, calendarizables e implementables en cualquier orden
- **Negotiable**: la US no define "como" hacerla sino el "que"
- **Valuable**: debe tener valor para el cliente la US
- **Estimatable**: estimable, tengo que poder decir cuanto esfuerzo mera llevar y apartir de ello definir el costo. Sino puedo estimar la puede que sea grande. La estimación ayuda al cliente a armar un ranking basado en costos
- **Small**: tienen que ser pequeñas ya que deben entrar en una sola iteración. Si una US lleva más de 4 semanas en hacerla hay que

NOTA

partirla deben ser "consumidas" en una iteración

- Testable: demostrar que fueron implementadas. Ejecutarla y poder correr las pruebas de usuario

Spike

Tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una User Story u otra faceta del proyecto. Es una tarea de investig. Se crea un spike para que alguien del equipo se siente con la historia y obtenga la investigación de los riesgos y así lograr otra cosa. Se clasifican en = técnicas y funcionales

Pueden utilizarse para:

✓ Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.

✓ Analizar un comportamiento de una historia compleja y poder así dividirla en piezas manejables

✓ Ganar confianza frente a riesgos tecnológicos, investigando o prototipando para ganar confianza.

✓ Frente a riesgos funcionales, donde no está claro como el sistema debe resolverlo interacción con el usuario para alcanzar el beneficio esperado

Los spikes son importantes porque llevan tiempo y alguien las va a tener asignado como tarea de investigación

*Funcionales se manejan con prototipos - Relacionado más a como el usuario interactúa con el sistema. Los spikes funcionales son utilizados cuando hay cierta incertidumbre respecto de cómo el usuario interactuará con el sistema. Usualmente son mejor evaluadas con prototipos para obtener realimentación de los usuarios o

involucrados.

• Técnicas - utilizadas para investigar enfoques técnicos en el dominio de la solución. Evaluar performance potencial. Decisión hacer o comprar. Evaluar la implementación de cierta tecnología. Cualquier situación en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad en un tiempo fijo. Tienen que ver con una tarea de investigación.

Los spikes deben resolverse en una iteración, sino deben partirse

Lineamientos para spikes

- Estimables, demostrables y aceptables.
- Son la excepción, no la regla.
  - Toda historia tiene incertidumbre y riesgo.
  - El objetivo del equipo es aprender a aceptar y resolver cierta incertidumbre en cada iteración.
  - Los spikes deben dejarse para incógnitas más críticas y grandes.
  - Utilizar spikes como última opción.
- Implementar la spike en una iteración separada de las historias resultantes.
  - Salvo que la spike sea pequeña y sencilla y sea probable encontrar una solución rápida en cuyo caso, spike o historia pueden incluirse en la misma iteración.

Se debe resolver primero el spike en un sprint y luego la US en otro sprint.

Cosas a dejar en claro:

- Dejemos el análisis detallado tan tarde como sea posible, lo que es justo antes de que el trabajo comience. Esperar hasta el último.

- Se capturan requerimientos en la forma de user story, que son descripciones breves de funcionalidad relevante para el cliente.
- Las user stories no son requerimientos; son marcadores para conversaciones más detalladas y análisis que deberán ocurrir conforme esas historias vayan implementándose.
- Por lo tanto, no necesitan ser descripciones exhaustivas de la funcionalidad del sistema, sólo la suficiente información para que los desarrolladores y los clientes tengan una comprensión común.

03/04 **Gestión de Productos**

Cuando se habla de gestión de producto se habla de que nos motiva