

MANIFIESTO ÁGILE

- se valoran más los individuos y las interacciones que los procesos y los herramientas. En las personas que hacen el producto.
- Entregar software parcial, le da una idea temprano de hacia donde va el producto a los interesados y puede corregir.
- Se valora más la colaboración con el cliente que los proyectos de alcance definidos. El cliente va validando el producto.
- Manejo constante de respuesta al cambio

25-03-2022

VALORES ÁGILES

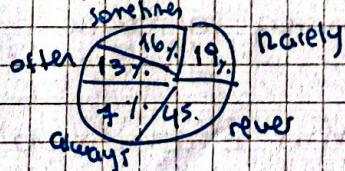
- Individuos e interacciones sobre procesos y herramientas
- Colaborar con el cliente sobre los requisitos contractuales: vomer a trabajar con el cliente sin plantea un esquema rígido.
Los principios son más reales, los valores son más misión.

* REQUERIMIENTOS ÁGILES

→ primero deben identificar el valor que los req. agregan al producto para construir el producto correcto. No vomer a hacer descripciones exhaustivas si no que vomer a trabajar con lo mínimo que necesitamos.

La manera de obtenerlos es colaborando con el cliente permanentemente.

→ Los productor de software exitoso también tiene un desperdicio significante. El costo del tradicional TBRUT



Usar de los fundadores del software típicos

Gestión ágil de requerimientos de software

- a tener en cuenta
 - trabajar e involucrar al cliente, de me dice que quiere y el valor que lo da al producto.
 - Que no entregar rápido, pero de tal manera de poder gestionar el cambio, recalibrar el cambio. Mientras mas grande sea el

entrega, más difícil es combinar. Los requerimientos van a cambiar, y ser una ventaja competitiva si se puede actuar sobre ellos. → los req que mayo solo hacen para el negocio - cliente van al principio de la lista. **features always** - Features never

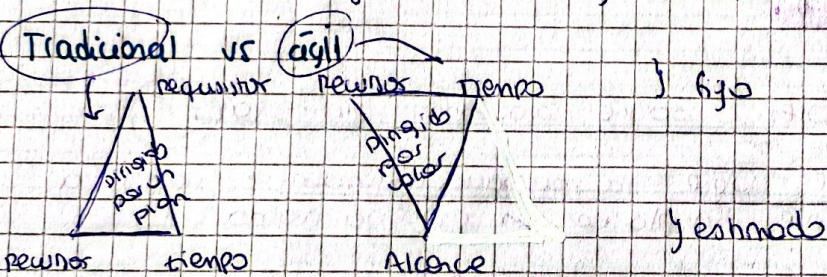
→ Iteración: voy a elegir un periodo de tiempo y voy a arrancar trabajando con el primero y ver que puedo entregar en ese periodo.

Haciendo esto minimizo por ejemplo, los posibles cambios en los req del medio de la lista ya que no se trabaja en ellos aún. Además ayuda al cliente a saber si es lo que el quería o no.

Solo trabajo con los req en el momento que me faltan, Just in time. Así el tiempo de la iteración es entregar lo que se pide en un periodo determinado antes.

* En la gestión de req ágiles, lo mejor comunicarse en boca a boca y un pingüino.

de pronto esta comunicación, no nos vamos a encontrar con cosas de uno solo entregar al cliente y que este lea y vea si está bien.



TIPOS DE REQ.

* Funcional = caso de uso

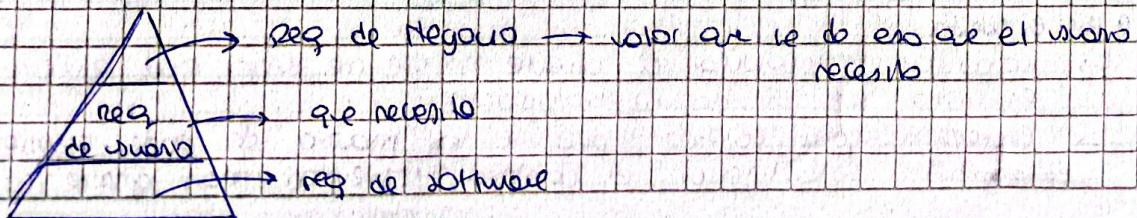
* No funcional = en lenguaje coloquial y no medible (^{sobrepeso}, cumplimiento)

* De implementación = restricción de tecnología Ej: servir a la nube

* De los usuarios = que quiere hacer el usuario Ej: soldado

* De Negocio = para saber cuál es el valor de un req. al cliente

En la gestión ágil los necesito porque ahí entro el valor para el cliente. Ellos req se entrega de forma horizontal, todo el req funcionando. Hecho sobre la propiedad.



DINÁMICA

- Primero entiendes que necesitas y que valor te da el usuario
- La solución para ese valor, tanto se determina todo el trabajo de manera colaborativa entre el cliente y el equipo de desarrollo
- Todo esto lo haces para lo anexo frecuente de valor (en el software)

Entonces → yo sabemos que los req cambian, hay que aceptarlo

- los Stakeholder: no sabemos muy bien los intereses con el proyecto. Hay que tomarle el tiempo para saber quienes son. Innovación en el proyecto = Clientes, competencia, equipo del proyecto, etc.
- El usuario nos da un mejor feedback cuando ve lo que le entregamos.
- La gestión de req es muy importante para que sea exitoso el desarrollo al final del proyecto
- Siempre nos interesa AGREGAR VALOR AL PRODUCTO.

12 PRINCIPIOS

- ① satisfacer al cliente la trama de releases
- ② recibir cambios en req con la etapa timely
- ③ técnicos y no técnicos
- ④ El medio de comunicación cara a cara
- ⑤ Los mejorar arquitectura, req y diseño → autoorganizarse

y interesar

* User Story

- es uno de los técnicas para la gestión ágil de req. La parte más difícil de conseguir un hito de Iw son los requerimientos.
- Un nombre visto de que se detalla y se especifica como los veríamos: una historia del req y el valor que genera para el cliente.

Partes }
 1) conversación (la parte más importante)
 2) card → ayuda monos
 3) confirmación

- La user story tiene una determinada sintaxis ⇒
 como < nombre del req > que lo necesita

yo puedo <actividad> la cuál
para <valor de negocio que reúno>

Multipropósito de la user story

- * Token para una conversación
- * Mecanismo para diferir una conversación
- * un recordatorio de unión
- * un ítem de planificación:

La Gestión ágil NO es para todos los productos. Hay que ver cuándo aplica.

Plata → Product Backlog

↑ alta prioridad - mayor detalle
↓ baja prioridad - menor detalle

- * Cuando decidimos hacer cierto cantidad de user stories que el equipo tiene que trabajar con todo, Interfaz, función, BD, entonces el personaje que se encarga de la story tiene que saber hacer todo

Modelado de user story

Es importante porque nos permite saber de cuáles son las partes del user story para imaginarlas las ventajas y limitaciones cuando se desarrolla el usuario

Criterios de aceptación

- ↳ nos define los límites de lo que es user story
- ↳ ayuda a los PO (product owner)
- ↳ ayuda a que el equipo tenga una mejor comprensión de lo que es user story
- ↳ ayuda a desarrolladores y testes a dentro probar

Mientras más ambigüo sea, porque menos definido está

- ↳ Debe ser independiente de lo implementado (sin engranaje)
- ↳ no contiene soluciones (que es lo que esperas no como resuelto)
- ↳ relativamente de alto nivel, no se entra en los detalles

Donde va el detalle

- ↳ doc interna del equipo
- ↳ prebar de aceptación automatizado

Probar de aceptación

- todos los esfuerzos que se dan, que se pierden dar.
- La teoría dice que lo que lo hace el product owner, entonces probar los "hace él".
- mientras más rápido, más fácil será probar

acuerdo entre
el equipo

Definición de lista - Definition of ready }

la mitad cumple con el criterio de ready, cuando esto se lleva de tal manera que puede ser incluido en una iteración

- en un acuerdo del equipo, se pone de acuerdo en las características que tiene para estar lista
- INVEST → como mínimo debe cumplir con estos

I	independientes	
H	negociable	(no debe incluir detalles de implementación)
V	valuable	(valor para el negocio)
E	estimable	(necesario sobre el esfuerzo)
S	small	(debe poder empejarse y terminarse en 1 iteración)
T	testable	(se debe documentar lo que se ha escrito en la user story que se ha entregado)

ÉPICAS → user backlog

ITEMS → colección de user stories

SPIKES → un tipo especial de user story. Tiene definido una user story que no pudo terminar hoy, al día de hoy se acuerda que no se acuerda lo que se tiene que hacer, entonces se utilizan para tener incertidumbre de una user story.

NO TODO LO QUE BEMERTE INCERTIDUMBRE EN UNA SPIKE "Investigar uso de google maps" → lo solido es el resultado de la investigación y con eso refina la user story.

- Técnico: tecnología que no conocemos
- Funcional: indagar e saber cual es la mejor forma de comunicar con el cliente.

→ deben ser estimables, documentables, y aceptables

1 - 03 - 2022

Definition of Done

Análogo al criterio de ready, la user tiene que cumplir con ciertas características definidas por el equipo.

Ej.: uno de los características en que se haya hecho todo lo que probar

USER STORYS

- Las user no son req funcionales.
- No entiendes lo que es.
- No es la única documentación del producto

DIFERENTES NIVELES DE ABSTRAJUDR

- Teneras → puede incluir uno o varios y épico tambien
- User stories → cuando son muy grande y no poden estimarse, se llaman: - epicas, y enton abajo en asunto pico de tareas. cada user story va siendo mas reñido. lo épico es una larga user story

Estos niveles me sirve para no odiarme de que debo trabajar poco en este momento.

Cuando lo épico se repite en user stories, estas deben tener que cumplir el modelo invent.

La spike se resuelve antes que lo user. yo que lo idea es investigar como la user. como la spike es un tipo de user, debes seguir los mismos lineamientos que esto.