Toma la teoría? 21:00

Bulzomi
si sra 21:02

+54 9 223 437-4143 ~Florencia D'Aurelio
Gracias 21:02

Bulzomi
no hay de que 21:02

sobre todo hitos de up 21:02

manifiesto agil 21:02

porque se paso de una a otra 21:02

etc 21:02

+54 9 223 582-0906 ~~Sergio
Para resumir... entra todo o casi todo? Jaja

+54 9 223 567-9954 ~Matías Nicoletti
todo el canal de cris rua 21:04

y el del ingles 21:04

o irlandes no me acuerdo 21:05

Nono, UP, hitos de UP, por que habia problemas, manifiesto agil, scrum 15:32

Primer video: https://www.youtube.com/watch?v=Q-mLfDXqSKo

Cuando ya tenemos el presupuesto aceptado y un mas o menos que vamos a hacer se puede decir que vamos a empezar con el proyecto, entonces llamamos a las personas que saben del tema del que se va a trabajar y les pedimos que nos den los requerimientos (ellos no se van a preocupar por lo q va a valer asi q se van a ir a la mierda). El problema es que muchas de estas peticiones quedan dentro del proyecto, y terminar todo eso, es casi imposible.

Pero para esto tenemos la "***INCEPTION***" para poder ENFOCAR tu proyecto, y asi lograr que lo que se logre en el proyecto genere un IMPACTO.

Entonces 1º hacemos un Impact Mapping y 2 User Story Map (en el cual se definirá el impacto que va a generar el proyecto y se va a garantizar que todo tira hacia un mismo objetivo)

el Impact Mapping se conforma de 4 cosas:

1. ¿para que es el proyecto?
2. ¿a quienes se va a impactar y quienes impactan el proyecto?
3. ¿Cuáles son los impactos que se van a dar en el proyecto?
4. ¿Qué se va a hacer en el proyecto?

El User story map:

Definimos por donde vamos a arrancar y comenzamos a darle estructura al proyecto

la duración de todo esto dependerá de unas variables (de 4hs a 1 dia):

- Quien lo dirige y que capacidad tiene para llegarle a las personas y sacar la información que es
- Que conocimiento se tiene del proyecto, `puede que estemos en 0 o puede que tengamos un Norte muy claro
- Como se van a dar las conversaciones en estos momentos, podemos dejar que se alaaaaarguen se vuelve de mucho tiempo
- Cuantas personas hay
- Si son las correctas
- Y si se sienten en la libertad de hablar

Ambos pueden durad desde 4 horas a días

Segundo video: https://www.youtube.com/watch?v=dt-EIN8dBbY

IMPACT MAPPING - INCEPTION AGILE - COMO INICIAR UN PROYECTO

En un impact mapping deben estar sentadas:

- Las personas que tienen poder de decisión sobre el proyecto
- Las que van a realizarlo
- Y quienes conocen sobre el tema

No hay que llenarlo, hay que traer a los que generen las conversaciones correctas

En un impact mapping se realiza en 4 pasos:

1. Para que del proyecto: suele pasar que pensas que la tenes clara en el "para que" pero cuando sentas a todos a hablar puede que existan perspectivas muy diferentes, y es preferible que esto pase al principio y no cuando el proyecto ya esta hecho y te das cuenta de que no era lo que se buscaba. DEBE SER SMART.
   - SPECIFIC – especifico: corto y claro.
   - Measurable – medible: que se pueda saber cuando acaba el proyecto.
   - Attainable – alcanzable: tener conciencia de que el proyecto realmente se puede construir.
   - Relevant – relevante: que sea importante lo que se esta construyendo.
   - Time based – temporal: tener un estimado de cuanto tiempo nos lleva.

2. A quien se va a impactar con el proyecto y quienes impactan el proyecto:

Vamos a tener impactos positivos y negativos, los positivos necesitamos aprovecharlos, y los negativos implementar formas para que no sucedan o para tratar de resolver los inconvenientes que se podrían presentar.
mientras menos actores haya más enfocado va a estar el proyecto.

3. Cuáles son los impactos que se van a dar en el proyecto:

Son los impactos que vamos a realizar en el proyecto, cual es el impacto que se va a generar en ellos o cual es el impacto que generan ellos (positivos y negativos de ambos).

4. Que se va a hacer en el proyecto:

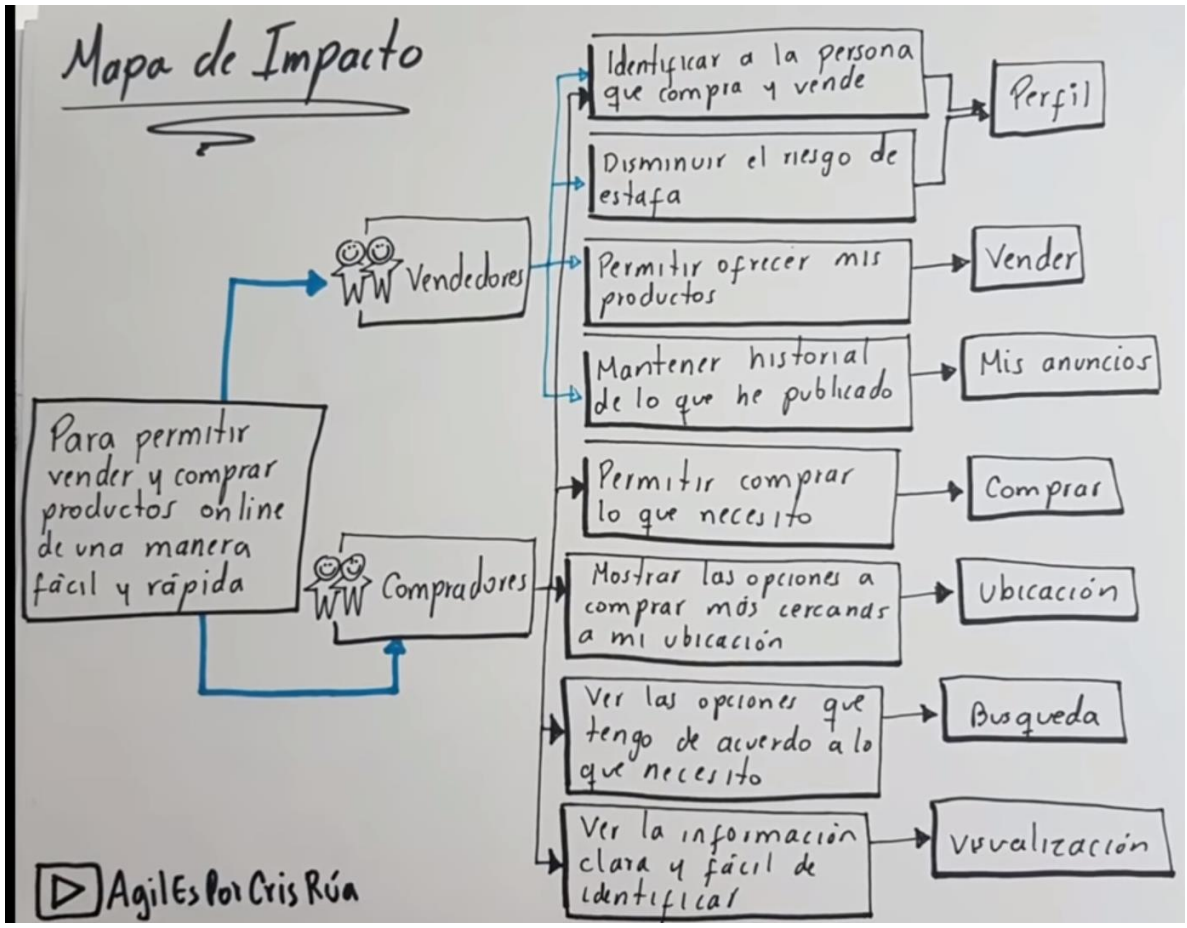
Muchas veces se arranca por aca y por eso no se tiene un impacto definido. Este paso son los que?.

USER STORY MAPPING

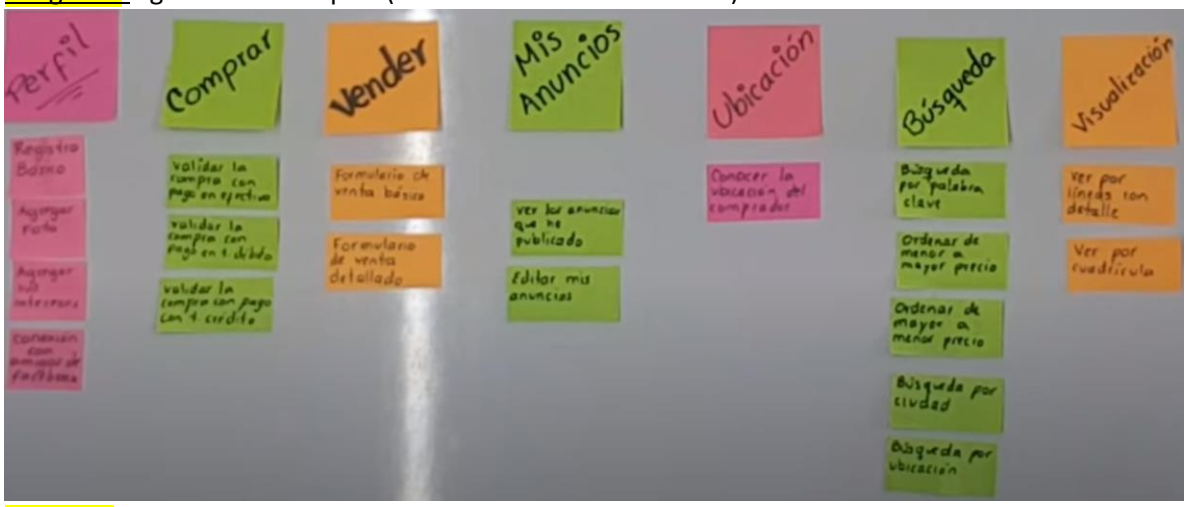
Del impact maping tenemos como resultado el QUE se va a hacer, ahora lo que necesitamos es:

1. Desglosar: esas cosas que tenemos q hacer.
2. Priorizar: lo que es más importante para el proyecto.
3. Armar las versiones: de lo que iremos entregando.

Ejemplo:

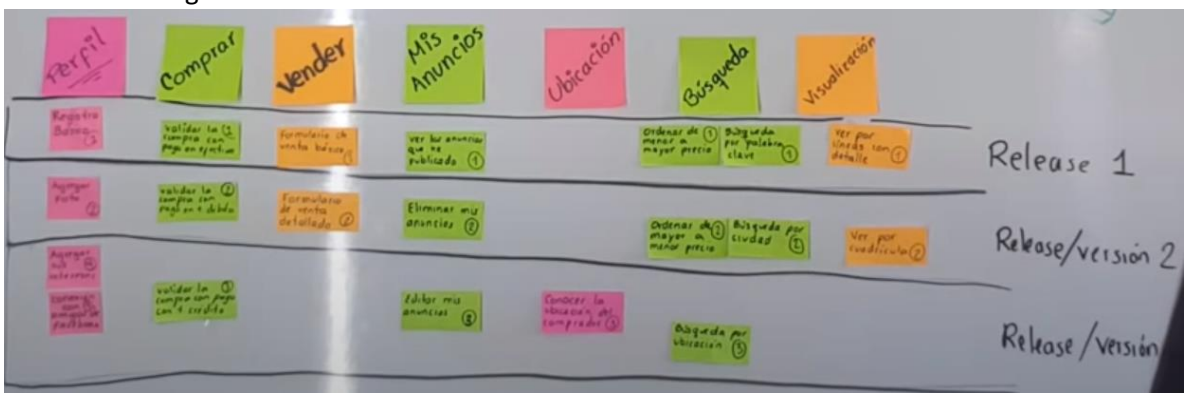


1. Desglozar: agarramos los “que” (cuadrado mas de la derecha)



2. Priorizar: MoSCoW:
   -Must: tiene que estar.
   -Should: debería estar.
   -Could: podría estar.
   -Won’t: descartado de momento.

3. <u>Armar Versiones:</u> aca es importante ir sacando releases, ya que asi vas obteniendo feedback ya sea del público o compañeros etc, y si fuiste suficientemente estratégico, incluso sacar algo de dinero.



1- Desglosar las cosas que tenemos para hacer.

2- Priorizar lo que es mas importante para el proyecto: Una tecnica util es el MoSCoW (Must *tiene que estar* - Should *debería estar* - Could *podria estar* - Won't *descartado de momento*).

3- Armar las versiones que iremos entregando.

ESTIMACION POR PUNTOS DE HISTORIA PARA UN EQUIPO SCRUM

El gran problema es estimar por horas, hay que pensar en dificultad.

Estimamos para saber con cuantas cosas se puede comprometer un equipo durante un sprint.

Como cada cosa que llega al equipo tiene diferentes tamaños y diferentes dificultades se hace la estimación, para empezar a encontrar la sumatoria de esas dificultades de las cosas que hay que hacer y saber hasta dónde podemos llegar a comprometernos para trabajar en un sprint sin estresarnos y pudiendo dar nuestra máxima capacidad.

1) Lo primero que tenemos que hacer es definir un pivote (uno por proyecto), que es aquella historia de usuario que todo el equipo sabe cómo se realiza y que va a ser la base de la medición para el resto del proyecto.
2) Puntuamos nuestro pivote.

PUNTOS DE HISTORIA vs HORAS

En un equipo no todos tienen las mismas horas para los distintos puntos de historia, para alguien con más experiencia 5 P.H puede llevarle 2-3hs, para alguien que recién esta arrancando podría llevarle 8hs.

No hay que ser egoísta y pensar las historias en horas sino en puntos de historia, así el equipo conoce la dificultad que le llevaría a cada uno.

SCRUM POKER - Estimacion y planeacion del Sprint en Scrum

Este se usa para poder comparar la dificultad de una historia de usuario con otra historia de usuario y no tener que decir cuanto tiempo se demora sino cuan dificil es.

Para iniciar estimar con el scrum poker se necesita que todo el equipo tenga las cartas de estas, hay unas fisicas y hay una aplicacion que se encarga de ello.

*Mientras mas chicos sin los numeros mas juntos van a estar, y mientras mas lejanos, mas separados.
Esto es asi ya que cuando se indica cuan mas dificil es no hay mucha discusion si es 20 veces mas dificil o 21 veces mas dificil.*

Se hace:

- Cada persona del equipo indica cuantos puntos de historia es la historia.
- Habla el que puso el mayor valor y el que puso el menor valor
- Se vuelve a hacer otra votacion sabiendo la opinion de los dos extremos
- Se recomienda que en incertidumbre (mucha gente voto 3 y 5) se deje el mayor

COMO DIVIDIR LAS HISTORIAS DE USUARIO EN TAREAS

En scrum hay un espacio que se llama la retospectiva que es la que nos ayuda a que las cosas se pulan cada vez mas.

En scrum se recomienda que la parte de la planeacion tenga dos partes.

1) Donde te reunis con el cliente y quedas en las cosas que deben hacer.

2) Donde el equipo se queda y distribuye las tareas y en cómo van a realizar lo que ya entendieron que se debe hacer y se comprometieron.

Hay diferentes tipos de distribución:

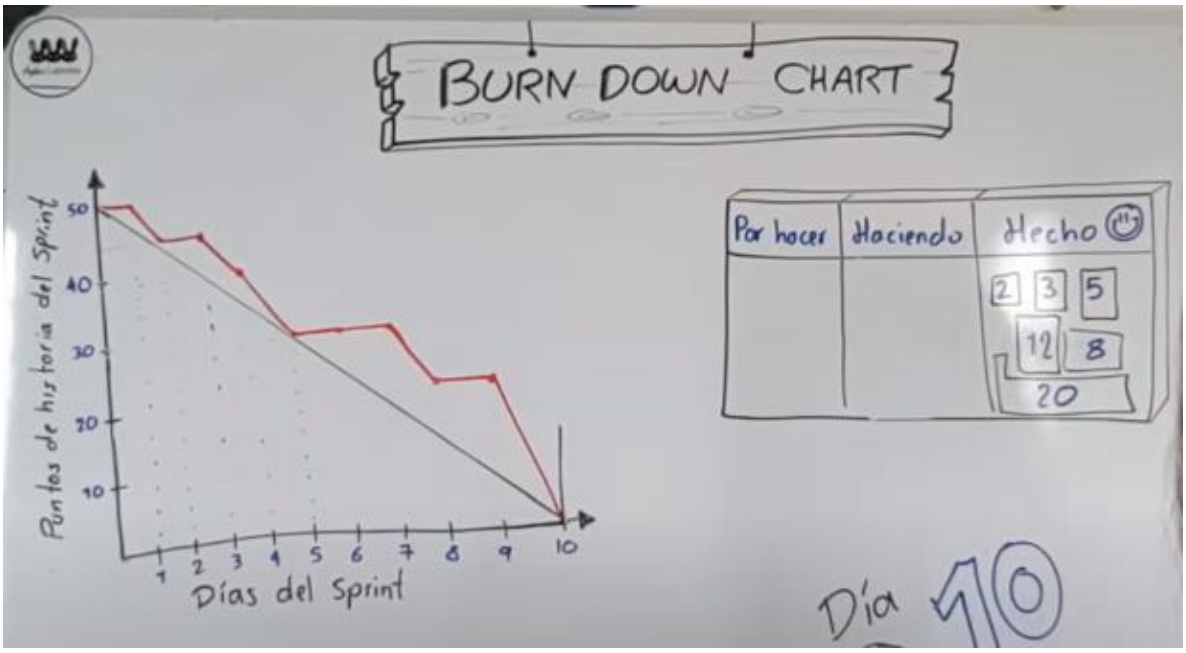
- Individualista: se reparten las historias de usuario y cada uno toma uno y al final se encuentran.
- Equipo: la idea de esta es que se compartan conocimientos así cuando uno no esta no se dependa solo de ese y haya otros compañeros que lo pueden llegar a cubrir.

Cuando se vaya a dividir las historias de usuarios en tareas se debe tener dos cosas en cuenta:

- Requerimientos funcionales del cliente, son esas que se quedó en hacer con el cliente.
- Cosas técnicas que se deben realizar para que salga todo bien, son las que el equipo sabe que tiene que lograr para poder hacer las cosas con calidad.

BURN DOWN CHART

Esta es una gráfica muy importante de Scrum, indica los días del sprint (eje x) y los puntos de historia del sprint (eje y).
Según esta grafica nos ira mostrando si estamos cortos de tiempo o si estamos mas sobrados, cuando estamos por encima de la línea trazada anteriormente vamos atrasados.

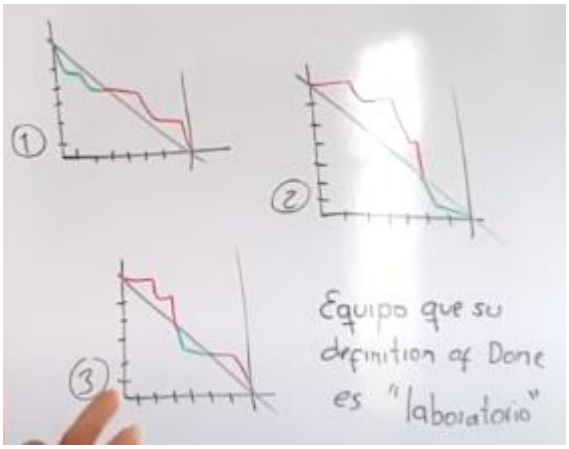


Al analizar la grafica no hay que caer en lo típico de decir, ah si no terminaron las historias de usuario al final del sprint son un equipo malo, no, esto puede significar muchas cosas:

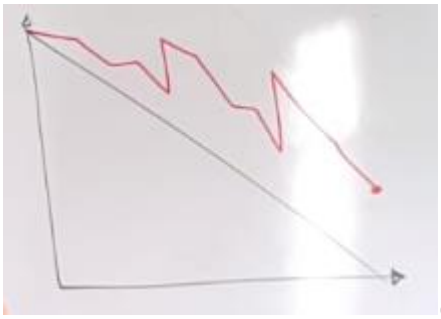
Como que por ejemplo las historias de usuario no estaban claras.

Haber sido demasiado optimistas.

Tener un produc owner demasiado presionador que pide siempre mas de lo que se puede.



esto no quiere decir que sea un buen grupo, todo depende de que considere el equipo como “tarea terminada” si para el equipo terminar las tareas es dejarlas en laboratorio es más sencillo (y NO es agile) “terminar el sprint” pero lo correcto es decir que una tarea está terminada cuando está funcionando.



esto puede pasar cuando en medio del sprint viene por ejemplo el usuario y te pide que le agregues algo, que no puede esperar al próximo sprint. Esto esta muy mal y el product owner no debería permitirlo.

Join Academia

SCRUM PROCESO

Se alimenta de una reunión previa que hace la organización en la cual reúne a las personas con más experiencia en el producto que se quiere y define cual van a ser sus principales características.

De esta reunión salen dos elementos importantes:

- Product Owner o dueño del producto: es una persona que queda empoderada por la organización para tomar las decisiones que considere convenientes, y que tiene clara la visión de lo que el producto aportara a la organización.
- Product Backlog o lista de características del producto.

Eventos de Scrum:

1) Sprint Planning: se divide en dos partes.

   - Refinamiento: se toma las características que se tiene del producto y detallarlas de tal manera que sean claras para todo el equipo de trabajo. Se puede hacer en cualquier momento
   - División por tareas: toma cada una de estas características y la divide en tareas a realizar por el equipo.

2) Sprint Backlog: es la cantidad de tareas a realizar en el tiempo definido para el sprint

3) Scrum Master: el cliente define el tiempo del sprint y sabe cada cuanto va a tener un incremento el producto, el avance revisado en una reunión diaria llamada Daily Meeting cuya duración es de 15 minutos, esta reunión tiene como objetivo ver que se ha hecho desde el día anterior, que dificultades ha tenido y que se va a hacer antes de la próxima reunión.

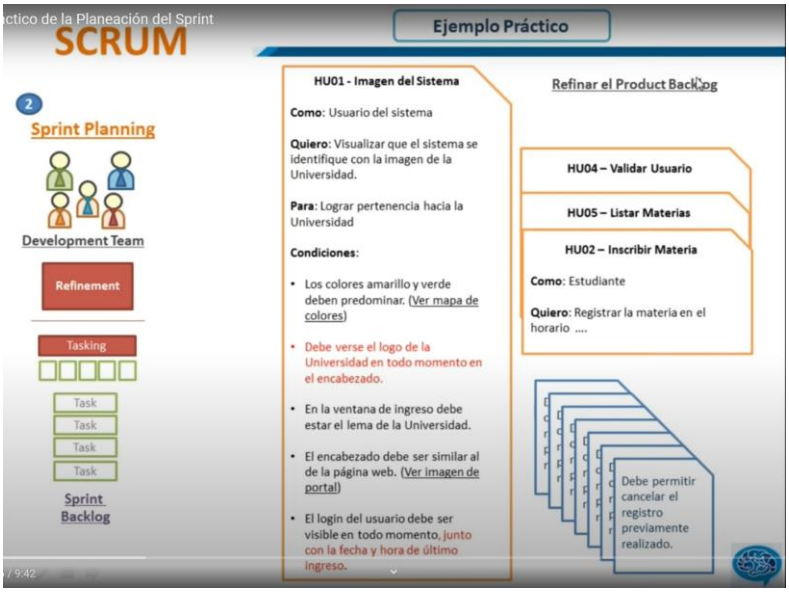
4) Product Increment: es otro de los artefactos importantes para Scrum, el incremento del producto es funcional, es lo que ya venía desarrollado mas eso que estamos desarrollando en este sprint.

5) Sprint Review: tiene como objetivo verificar el avance que se ha tenido en el desarrollo del producto, indicar que falto por cumplir, que se va a hacer a continuación y algunas veces definir si pasa o no a un ambiente productivo.

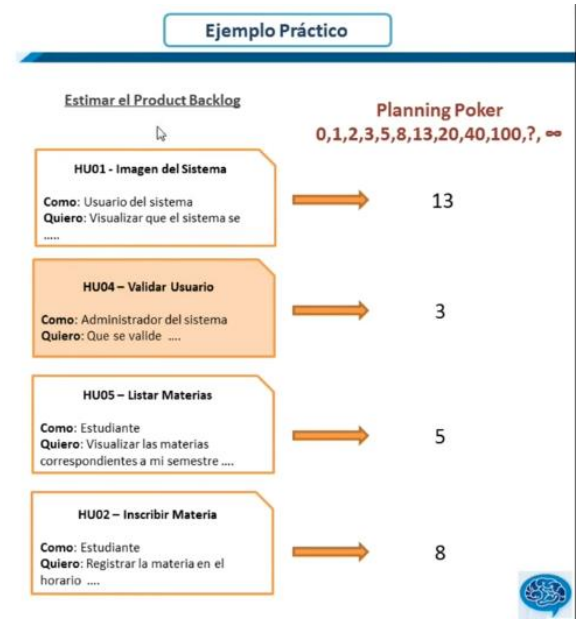
6) Sprint Retrospective: es el evento final y busca revisar como fue el desempeño del equipo en el proceso, verifica que se ha hecho bien y que no se debe seguir haciendo y que se puede mejorar.

EJEMPLO PLANIFICACION DEL SPRING

Ya es tiempo de que el Product Owner con el Scrum Master se reunan con el equipo de trabajo



Tomamos las historias de usuario y las refinamos (recorda que esto se puede hacer cuando sea) y vemos (en rojo) que se agregaron mas condiciones en mayor detalle

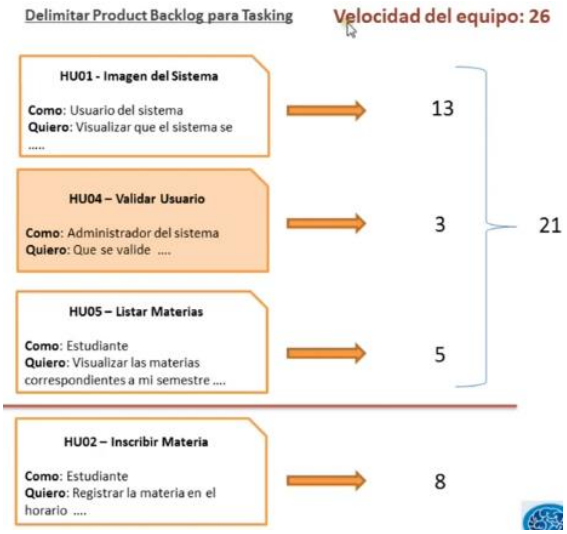


Se elige el pivote, se le da el puntaje y se elige el puntaje de las demás en base a esta.

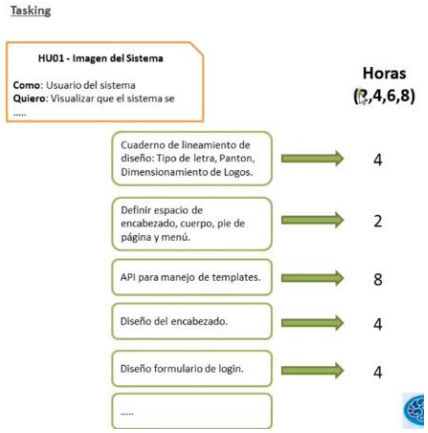
Ahora decidimos la duracion del Sprint (2, 3 o 4 semanas) generalmente aquí el Product Owner ya tiene ese dato por que ya fue acordado con el equipo o por el jefe de área.

Y se tiene que dar la Definicion de Terminado?

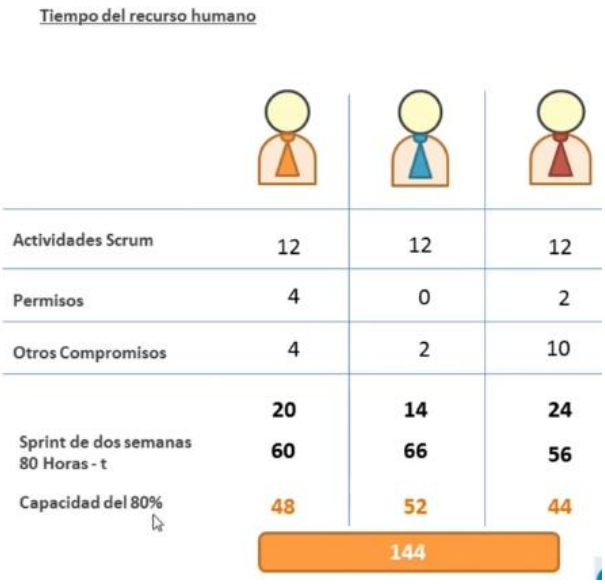
Ahora hay que determinar la velocidad del equipo



Ahora hacemos el Tasking: dividir las historias de usuario en tareas:



Y se le asigna un estimado de horas que llevara concluir cada tarea, deben ser un numero par y no pueden superar las 8 horas, si es asi debe subdividirse en mas tareas.



Suponemos que el sprint va a ser de dos semanas, y que nuestras tareas de scrum van a ser de 12 horas por semana, se restan los permisos que pudo haber pedido la persona, también se restan los compromisos que puede tener esa persona con otros proyectos.
se supone que son 8 horas por dia, 5 dias por semana, dos semanas por sprint = 80 horas. A eso le restamos las horas de cada persona.
y por ultimo se recomienda hacer la capacidad del 80 %

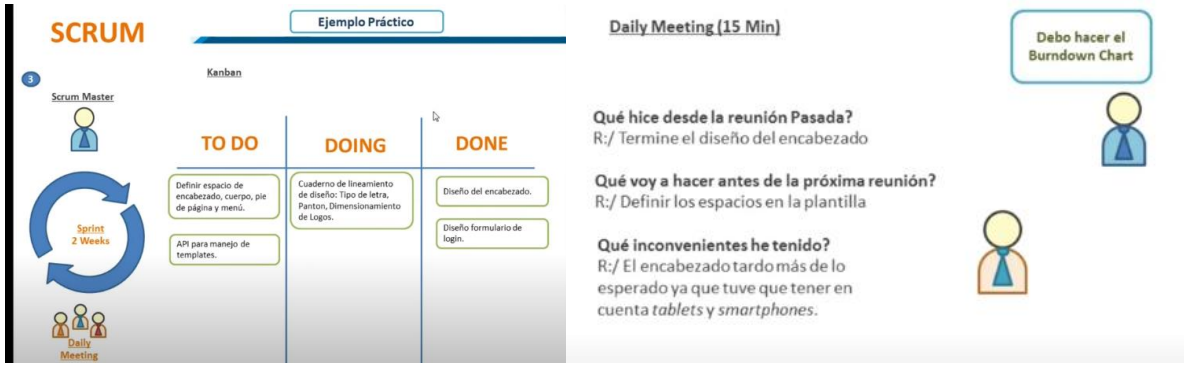
Finalmente, está el compromiso del sprint, donde se toman las tareas a realizar (viendo su prioridad y dependencias) y se decide cuantas se cree que podrán completar en el sprint según las horas estimadas.

Una vez terminado todo esto pasamos a la ejecución del sprint

SCRUM MASTER

El Scrum Master es el rol encargado para hacer que el equipo se vuelva altamente efectivo por medio de una correcta utilización del marco de trabajo Scrum.

- Removiendo los inpedimentos.
- Busca motivación.
- Tiene capacidad inata de motivar.
- Se vale de las restrospectivas para identificar riesgos y como prevenirlo (el team solo tiene 1 tipo q sabe hacer x tarea, esto es un riesgo)
- Tiene q saber potenciar las habilidades de cada uno.
- Mantener al equipo unido, y que seamos todos amigos :3
- Debería haber 1 scrum master por equipo, por que sino no tienen tiempo de conocer al equipo
- Tiene que estar muy de la mano con un Product Owner, es muy común que tal vez esta no tenga tiempo, bueno en tonces a quien le pregunto, como saco la información, tiene que mover ficha o buscar otro product Owner



Es el Scrum Master el que dice lo de hacer el Burndown Chart

Ante una eventualidad, el equipo de desarrollo vera si lo puede incluir en el sprint actual, sino le diran al Scrum Master que vea donde lo puede incluir, el cual se comunicara con el Product Owner

PROCESO UNIFICADO

El Proceso Unificado es un proceso de desarrollo de software: “conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software”.

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, junto con el Lenguaje Unificado de Modelado (UML).

UP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Dirigido por Casos de Uso y basándose en estos los desarrolladores crean una serie de modelos de diseño e implementación. De este modo los casos de uso no solo inician el proceso de desarrollo, sino que le proporcionan un hilo conductor.

Principales elementos dentro de UP:

¿Quién?: Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo

¿Cómo?: Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

¿Qué?: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

¿Cuándo?: Secuencia de actividades realizadas por trabajadores que produce un resultado de valor observable.

Características:

- Unifica los mejores elementos de metodologías anteriores.
- Preparado para desarrollar grandes y complejos proyectos.
- Orientado a Objetos.
- Utiliza el UML como lenguaje de representación visual.

Ventajas:

- Coste del riesgo a un solo incremento.
- Reduce el riesgo de no sacar el producto en el calendario previsto.
- Acelera el ritmo de desarrollo.
- Se adapta mejor a las necesidades del cliente.

Estructura del ciclo de vida:

- Dirigido por los casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. Se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

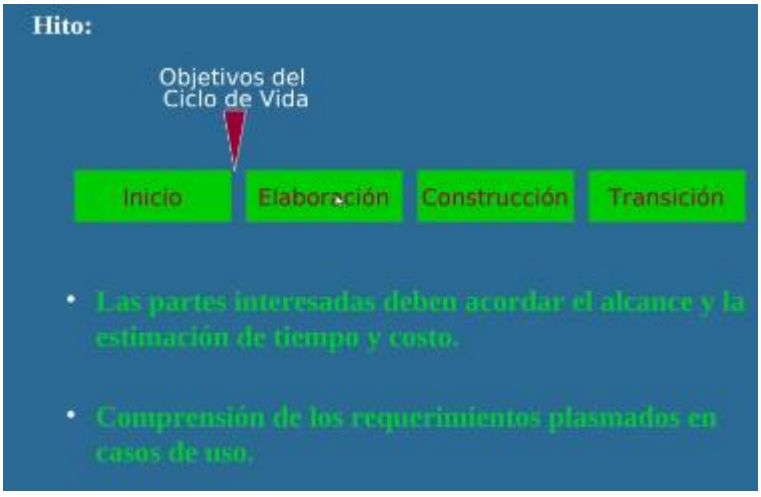
- Iterativo e incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones deben estar controladas. Esto significa que deben seleccionarse y ejecutarse de una forma planificada.

Fases del desarrollo en UP: El desarrollo en UP es a través de fases, Cada fase se subdivide en iteraciones. En cada iteración se desarrolla en secuencia un conjunto de disciplinas o flujos de trabajos. Cada fase finaliza con un hito (entrega de sistema, cumplir determinado objetivo, etc).

Fase de Inicio:

El objetivo de esta fase es ayudar al equipo de proyecto a decidir cuáles son los verdaderos objetivos del proyecto. Por resultado se pueden obtener: mayores requerimientos, boceto inicial, objetivos generales, versión muy preliminar, modelo de negocio, etc.

HITO:



Fase de elaboración:

Tiene como principal finalidad completar el análisis de los casos de uso y definir la arquitectura del sistema, además se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. La fase de elaboración finaliza con el hito de la Arquitectura del Ciclo de

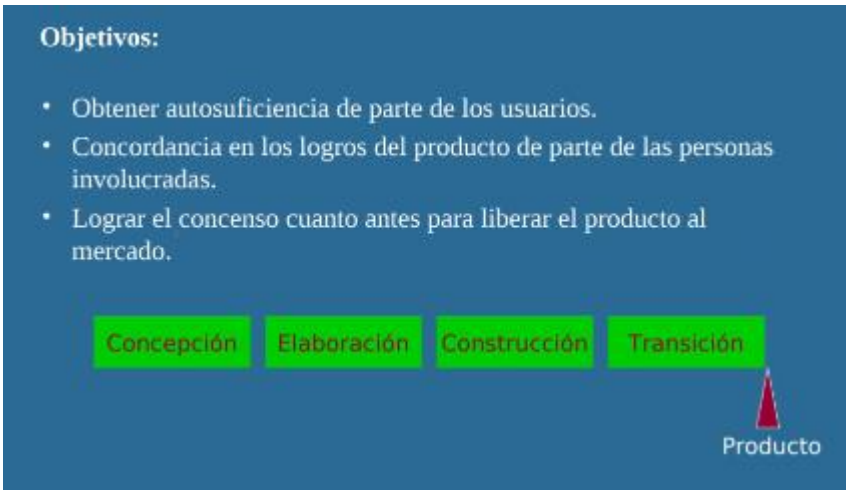
Vida. Este hito se alcanza cuando el equipo de desarrollo llega a un acuerdo como: casos de uso que describen la funcionalidad del sistema, la línea base de arquitectura, los riesgos, el plan del proyecto.

Fase de Construcción:

Está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto.
En la construcción se crea el producto. La línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de esta fase, el producto contiene todos los casos de uso implementados, sin embargo, puede que no esté libre de defectos. Al final de la fase se obtiene: el software, casos de test, manuales de usuarios.

Fase de transición:

Se inicia con una versión “beta” del sistema y culmina con el sistema en fase de producción. Las iteraciones en esta fase continúan agregando características al software. Sin embargo, las características se agregan a un sistema que el usuario se encuentra utilizando activamente. La fase de transición finaliza con el hito de Lanzamiento del Producto.



METODOLOGÍA ÁGIL

Estos métodos buscan un justo equilibrio entre ningún proceso y demasiado proceso, proporcionando simplemente suficiente proceso para que el esfuerzo valga la pena.

A diferencia inmediata con las metodologías ingenieriles es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. Son más bien orientados al código.

Las metodologías ágiles son adaptables en lugar de predictivos.

Estas metodologías constituyen un nuevo enfoque, simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

Los 4 valores / puntos de Agile:

- Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas.
- Desarrollar software que funciona más que conseguir una buena documentación, implica minimalismo respecto del modelado y la documentación del sistema.
- La colaboración con el cliente más que la negociación de un contrato. El cliente es parte del equipo de desarrollo (además in-situ)
- Responder a los cambios más que seguir estrictamente una planificación.

Metodología ágil	Metodología tradicional
Orientada a proyectos pequeños	Proyectos de cualquier tamaño
Problemas de escalabilidad en proyectos grandes	Problemas de adaptación a proyectos pequeños
Equipos pequeños (<10 personas)	Efectivas con equipos grandes y/o dispersos
Proyectos de corta duración	Proyectos de cualquier duración
Poca documentación (artefactos)	Mucha documentación
Pocos roles y más genéricos. Roles no intercambiables	Más roles y más específicos. Los roles no se intercambia
Más flexibilidad en el contrato	Contrato prefijado
Cliente parte del equipo	Cliente informado mediante reuniones con la dirección
La arquitectura se redefine y mejora a lo largo del proyecto	Arquitectura prefijada
Énfasis en los aspectos humanos: el individuo y el trabajo en grupo	Énfasis en la definición del proceso: roles, actividades y documentación
Se esperan cambios en el proyecto	No se esperan cambios importantes en el proyecto
Poco control de cambios	Control de cambios estricto y aprobado