**Scrum y métricas ágiles**

**Scrum**

*Es un framework para la gestión ágil de proyectos*

“En un Scrum, hay que actuar como una unidad, no como 8 individuos. Todos tienen un rol. Nunca debemos olvidar que cuando trabajamos juntos como una unidad, el todo es más que la suma de las partes”

El principio que tienen que ver con Scrum es lo que conocemos en otro contexto como sinergia, el todo es más que la suma de las partes y eso es lo que Scrum trata de potenciar en estos grupos de trabajo de 5 a 7 personas

¿Qué es Scrum?

* Scrum es un framework que permite crear su propio proceso para crear nuevos productos
* Scrum es simple puede ser implementado en pocos días, pero puede tomar mucho tiempo perfeccionarlo
* “Scrum no es una metodología, es un camino”
* “Scrum no es un proceso, una tecnica o metodo definitivo, Scrum es un **marco de trabajo** dentro del cual se pueden emplear varios procesos y técnicas”

Este framework nos ayuda a crear un proceso de desarrollo de SW (este proceso puede ser empírico o definido). En particular Scrum está en el contexto de procesos empíricos, para que en el desarrollo de SW se puedan crear nuevos productos

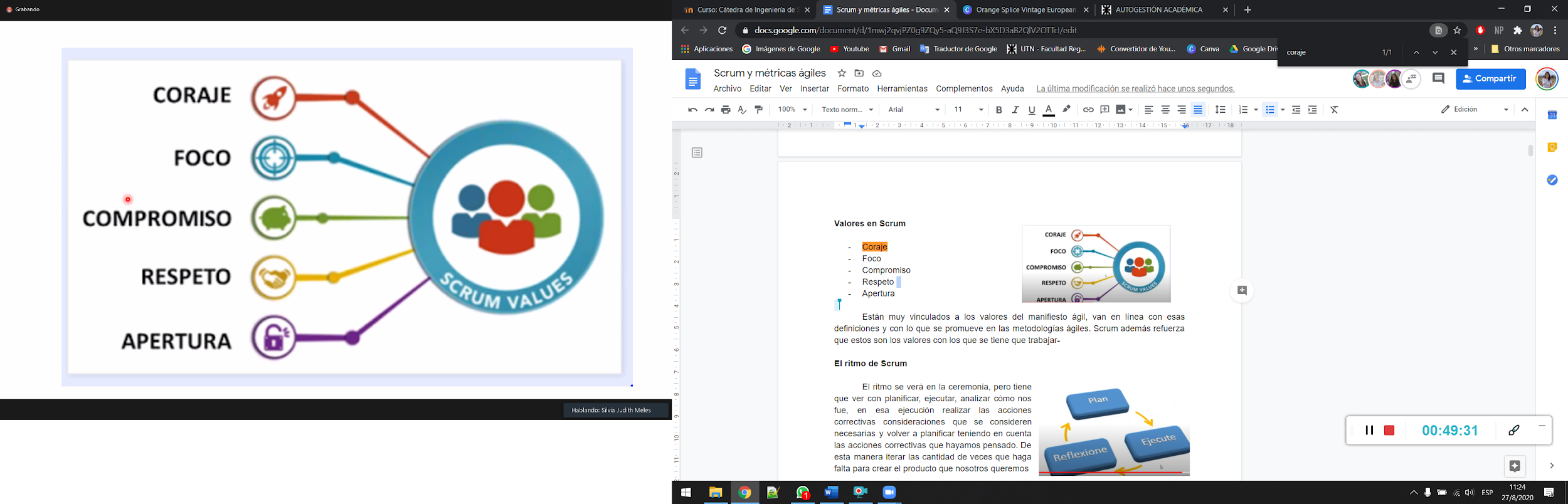
Scrum es un proceso simple, aunque es un proceso empírico trabaja bajo ciertas definiciones concretas, no es que el equipo de trabajo va a ir eligiendo como trabajar, sino que Scrum define lineamientos para trabajar, los cuales son muy claros y si o si deben cumplirse.

*Si bien nació como un framework para el desarrollo de SW, hoy en día se lo utiliza en distintos ambitos*.

**Scrum es empírico**

* Las metodologías rigurosas se basan en métodos definidos,con la idea de línea de ensamble
* El control de Scrum se alcanza con inspecciones frecuentes y correspondientes ajustes

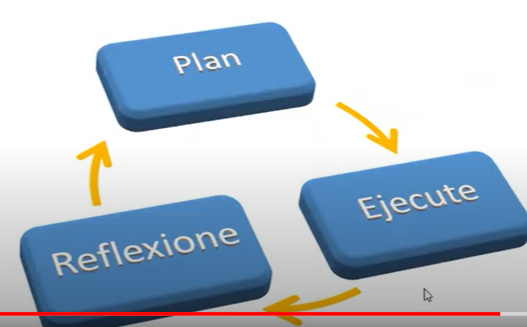
No es una metodología basada en procesos definidos, son estas metodologías basadas en líneas de ensamblado (uno de los componentes de un proyecto de SW). En realidad lo que tiene que ver con el control, por decirlo de alguna manera, está relacionado con la inspección y muchas veces con el control de pares, se usan más este tipo de controles en vez de otros como auditorías.

**Valores en Scrum** 

* Coraje
* Foco
* Compromiso
* Respeto
* Apertura

Están muy vinculados a los valores del manifiesto ágil, van en línea con esas definiciones y con lo que se promueve en las metodologías ágiles. Scrum además refuerza que estos son los valores con los que se tiene que trabajar-

**El ritmo de Scrum**



El ritmo se verá en la ceremonia, pero tiene que ver con planificar, ejecutar, analizar cómo nos fue, en esa ejecución realizar las acciones correctivas consideraciones que se consideren necesarias y volver a planificar teniendo en cuenta las acciones correctivas que hayamos pensado. De esta manera iterar las cantidad de veces que haga falta para crear el producto que nosotros queremos

**Cimientos de Scrum**

* Empirismo
* Auto-organización
* Colaboración
* Priorización
* Time Boxing

*Empirismo***:** se trata de un framework que trabaja bajo las características de un proceso empírico, no se basa en procesos definidos o metodologías tradicionales.

*Auto-organización*: entre los miembros del equipo, como se realiza este control, por este motivo, los equipos de trabajo deben tener características de poder auto-organizarse y esto implica que las personas que conforman estos equipos deben ser autónomas y poder auto-organizarse para cumplir con los objetivos que se plantean

*Colaboración:* entre los miembros del equipo, entre el Product Owner, todos los que participan de la construcción o que tienen algún rol dentro de este framework.

*Priorización:* es muy importante, hace referencia a que nos vamos a concentrar en aquello que definamos con mayor prioridad, sobre eso vamos a hacer foco y vamos a prestar especial atención, lo que quede con menor prioridad lo vamos a ver cuando llegue su momento, cuanto esté dentro de el tope o dentro del momento en que decidamos atenderlo.

*Time Boxing:* tiene que ver con las iteraciones y el trabajo se hace planteando límites de tiempo concretos. Lo que se deja fijo es el tiempo y lo que se va a modificar es el alcance para ver cuánto se va a construir en ese tiempo que está fijo, esto nos sirve para darle una cadencia de entrega a nuestro cliente y para asegurarnos que cada cierto tiempo se cumple con esa entrega. El alcance se va a ir definiendo cada Sprint a Sprint, el cliente sabe que cada cierto tiempo que el equipo de trabajo define como time Boxing el cliente va a recibir algo y eso permite manejar las expectativas.

Este time boxing se extiende más allá de la duración del sprint y de la entrega del cliente, se extiende a la duración de la ceremonia, como trabaja el equipo cuando hay algo para ejecutar. Este concepto se ve reflejado en todos los lineamientos del framework.

A diferencia de otros ciclo de vida iterativos e incrementales, por ejemplo en el Proceso Unificado de Desarrollo lo que se hace es partir el alcance y cada una de las iteraciones es un alcance concreto, lo que está fijo es el alcance y lo que modificamos es el tiempo que nos va a llevar cumplir ese alcance.

**Entonces...¿Cómo trabaja Scrum?**

* Equipos pequeños (<10 personas, ideal 7±2)
* Una serie de Sprint ( 30 días)
* Incrementos visibles y usables
* Time boxing
* Auto-organizado
* Colaboración
* Priorización

Equipos pequeños: si es necesario construir un producto más grande en donde con un equipo de 7±2 no sea suficiente, es necesario escalar a un equipo más grande, pero en principio cuando hablamos de framework Scrum los equipos son pequeños.

Sprint: duración fija que le vamos a poner a cada una de las entregas o los trabajos que vamos a hacer, debe durar 30 días y realmente podemos definir 3 semanas, esta duración la define el equipo. A partir de esos sprint obtenemos como resultado incrementos visibles y usables, sirven potencialmente para ser desplegados en producción, independientemente después de la decisión que nosotros tomemos.

**Scrum se siente diferente…**

* Menos tiempo planeado y definiendo tareas
* Menos tiempo creando y leyendo reportes
* Se pasa mas tiempo con el equipo investigando la situación
* Scrum
  + Supone dominios impredecibles
  + No supone un proceso repetible

En Scrum las métricas que se utilizan son mínimas, las mínimas que hacen falta, no hay un conjunto a elegir, a diferencia de los procesos definidos.

Scrum es muy útil para trabajar en *dominios impredecibles*, ya que no hay que definir el alcance completo desde el principio, sino que lo vamos definiendo y priorizando en conjunto con el Product Owner a medida que vamos avanzando, entonces esto permite que toda esta incertidumbre que tenemos a de este dominio la vamos trabajando y la vamos atacando poquito a poco, no nos ocupamos al principio de tratar de lograr mayor certidumbre o mayor predicción sobre todo el dominio completo.

Por otro lado, *no es un proceso repetible*, por eso en parte es que estamos planteando una cuestión disruptiva a los procesos definidos y a los proyectos que que realizan de manera tradicional.

**Pregunta:** Si tengo que estar colaborando todo el tiempo con el Product Owner supongo que tiene que haber cierta transparencia, entonces si el me pide que implementemos alguna tecnología, la cual yo se que mi equipo de trabajo no está capacitado ¿Que hacemos en ese caso?

No tiene nada que ver una cosa con la otra,al decir “dominio impredecible” no estoy hablando de la tecnología, sino que sobre el producto que voy a construir no tengo claridad todavía sobre cómo va a ser en término de requerimientos, no estamos tratando todavía la tecnología. Que la tecnología va a influir posiblemente pero en realidad estamos diciendo es que hay una parte de la que queremos construir que no sabemos va a ser, a eso nos referimos con dominio impredecible.

Después lo que tiene que ver con la tecnología, será parte del trabajo del equipo, encontrar o investigar la forma para poder avanzar con estas cuestiones tecnológicas que no dominamos, pero no necesariamente tiene que ver con un problema esto.

Va más allá de la complejidad esencial. Incluso tiene que ver incluso con lo que ustedes veían del MVP

“¿Qué hacemos con el MVP?”,empecemos a construir este producto, está parte del dominio “¿Y que es todo lo que vamos a ver después?”, no sabemos porque no tenemos mucha claridad para adelante, pero hagamo foco en esto que nos va a permitir retroalimentarnos y poder seguir trabajando más adelante

**Roles en Scrum**

* Scrum Master
* Product Owner
* Equipo de trabajo - Scrum Team

**Pregunta:** ¿Qué recuerdan de cada uno de estos roles?¿Cuáles eran las responsabilidades de cada uno de estos roles?

*Product Owner:* es el experto en el dominio y se encarga principalmente de priorizar los ítems, prioriza en función al valor que le aporta al negocio. El Product Owner es el referente del dominio y tiene que tener la capacidad de decisiones, esto no quiere decir que no pueda consultar con otros expertos del dominio, pero si tiene que quedar claro que debe ser una persona capacitada para tomar decisiones y para definir la prioridad, para donde vamos y que va a hacer el producto.

*Scrum Master:* es el “moderador”, ya que si le decimos líder corremos el riesgo de hacer una relación o una analogía entre un líder del proyecto en los procesos definidos y el Scrum Master y no es lo mismo. La persona que cumple este rol *es uno más del equ*ipo, es el que mejor conoce el framework y ayuda al equipo a salvar los impedimentos, a establecer cómo se debe trabajar.

El concepto es bastante distinto al de líder, ya que este hace entre otras cosas asignar tareas y chequear que estas tareas se cumplan, el Scrum Master NO hace eso.

**Pregunta:** el Scrum Master si se fija que las tareas se cumplan, ¿no por un tema de supervisión, sino por un tema de eficiencia del equipo?

El Scrum Master no tiene un rol de control, en términos de eficiencia para subsanar los impedimentos, esto si es así. Pero la línea es importante, no es lo mismo que sirva como un “facilitador” a pensar que el va a decir cuando hay un retraso.

*Scrum Team:* tiene que estar formado por integrantes motivados. No hay roles dentro del equipo, todos hacen todo, puede ocurrir que algunos sean mejores que otros y a la hora de dividir las user cada uno tiene más habilidad para algunas cosas que para otra, pero todos deben saber hacer todo . Tienen que ser auto-organizados, trabajar de manera colaborativa.

**Entregables**

* Product backlog
* Sprint backlog
* Producto de software potencialmente listo para producción

***Product backlog***

* “Es una cola priorizada de funcionalidades técnicas y de negocio, que necesitan ser desarrolladas”
* El Backlog de producto contiene la lista de requerimientos
* Se listan características, funciones, tecnológicas, mejoras, bugs, etc., que serán aplicadas al producto
* El backlog está incompleto inicialmente, aunque solo se necesita lo suficiente para realizar el primer Sprint de 30 días

Son técnicas y son de negocio, porque no necesariamente acá van las cosas de negocio pero si ser desarrolladas.

Son una lista de requerimientos donde normalmente en esa lista tenemos sobre la lista aquellos ítems que tienen mayor prioridad, en función del valor del negocio la prioridad que se le asignó.

No debe estar definido de manera completa, pero lo que sí se debe definir es lo que está en la parte de arriba de la pila y es lo que vamos a incluir en el próximo Sprint.

Lo demás puede estar en término de épicas, en término de temas que se van a ir refinando conforme van subiendo de prioridad en esa lista. Esto nos ayuda a trabajar en función del cambio y de los dominios que son impredecibles, ya que al hacer foco en lo que está arriba en la pila, lo que está abajo puede ser un recordatorio, que eso en algún momento se tiene que ver, puede ser una épica grande con la cual en algún momento se tendrá que trabajar. Mientras que esto no llegue a la parte de arriba de la pila de prioridad puede quedar expresado en esos términos, sin entrar en mayor detalle.

Los ítems que encabezan la pila debe tener un nivel de detalle suficiente como para que podamos sentarnos a trabajar con ellos.

*Asteroides… ¿Cómo referirnos a un product backlog?*

* Trabajar con un backlog de producto es como jugar con asteroides
* Grandes rocas(epics) se rompen en pedazos más pequeños(stories) hasta que son lo suficientemente pequeñas para ser eliminadas(desarrolladas y entregadas).

Lo más grande es lo que queda abajo y lo más pequeño que es lo que está arriba, son las user que tienen que cumplir el modelo INVEST, todas las User que quedan en la parte de arriba del backlog que son las que vamos a trabajar en el próximo Sprint deberían cumplir con el modelo INVEST.

*Estimaciones de Backlog*

* Las estimaciones son un esfuerzo colaborativo entre las partes
* Las estimaciones son iterativas
* Si un ítem no puede ser debidamente estimado, se debe dividir en el Backlog.
* Las estimaciones sirven de base para la funcionalidad en el Sprint.

Una de las características de este modelo INVEST es que la User tiene que ser estimable y pequeña, en ambos casos necesitamos hacer una estimación.

La técnica que se usaba para estimar es el *Poker Planning* y la unidad que se usaba para estimar eran las story point, los valores razonables que deberían tener la story point es entre 1 y 8,en cambio, si su valor es mayor a 8 significa que ya es grande. Cuando hablamos de que hay que dividir un ítem es cuando su valor es mayor a 8, no podríamos tener en la parte de arriba del backlog para ser incluido en el próximo Sprint un item que tenga por ejemplo un valor de 13, estos estimados que le vamos poniendo, estas story point que le vamos poniendo a cada User, este es uno de los elemento que nos va a hacer falta para definir cuales son los items que vamos a poder incluir en el próximo Sprint, con esto lo que estoy diciendo es cuánto esfuerzo, incertidumbre y cuál es el tamaño de esto que estoy estimando, por otra parte,nos falta poder determinar cuánto es lo que yo puedo hacer.



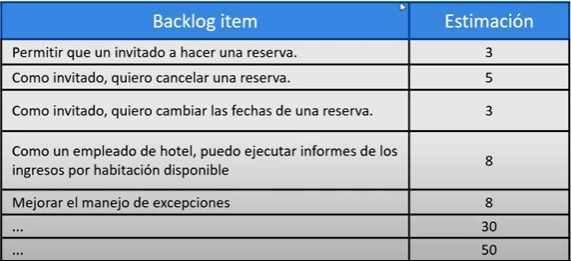
**Product Backlog**

* Los requisitos
* Una lista de todos los trabajos deseados en el proyecto
* Idealmente cada ítem tiene valor para el usuario o el cliente
* Priorizada por el Porduct Owner
* Reprioridaza al comienzo de cada Sprint

En la imagen, a la izquierda tenemos el product backlog, nosotros elegimos una partecita, son las historias o requisitos que están arriba, tienen la mayor prioridad y mayor valor para el cliente,el product Owner es quien define estos valores.

Está lista puede ser repriorizada cada vez que va a empezar un Sprint, es decir, si algo tenía un valor de negocio que estaba más abajo y en el Sprint Planning decido ponerlo más arriba eso también es válido, las prioridades no son inamovibles. Una vez que yo definí cuáles de esos requerimientos voy a hacer en el siguiente Sprint logro el segundo entregable que veíamos en la lista que es el *Sprint Backlog,* es una porción del product backlog que nosotros vamos a construir en el próximo Sprint.

*Ejemplo de Product Backlog*



En este ejemplo se pueden ver los items, los requisitos, la lista de requerimientos y las estimaciones, en este ejemplo falta la prioridad pero en principio la prioridad es el orden en el que aparece, es lo mismo de poner 1,2,3. Este es un ejemplo de un Product Backlog, el primero de los entregables de Scrum.

***Sprint Backlog***

* El equipo determina que debe hacerse para cumplir el objetivo del Sprint.
* El dueño del producto suele asistir
* Se realiza una lista de tareas que toman de 4 a 16 horas para completarse
* El backlog no se modifica durante el desarrollo
* Si el equipo descubre que el backlog no puede realizarse, el Scrum Master y el dueño del producto determina si:
  + Algun item puede ser removido del backlog
  + Alguna funcionalidad puede eliminarse

El Sprint Backlog es una porción del product backlog que nosotros decidimos como equipo hacer en el próximo Sprint, en donde en conjunto con el Product Owner decidimos cuáles son las User que vamos a incluir en este nuevo Sprint.

Para poder construir este Sprint Backlog vamos a necesitar, además de las estimaciones que ya se hicieron en término de story point, hacer una lista de tareas para completar cada una de esas User y teniendo en cuenta que esta lista de tareas para cada una, no debería demorar más de 16 horas ni menos de 4 para poder completarse, es decir, una segunda validación, además de que la User no debe tener una validación mayor a 8, cuando yo la parto en tareas, cada una de ellas tiene que tener la suficiente granularidad como para no llevarme menos de 4 horas ni más de 16.

*Ejemplo de Sprint Backlog*



Se desglosan cada una de las tareas y se asignan las horas de las tareas a los días de la semana, entonces en el Sprint Backlog empieza a aparecer este concepto de tarea justamente para que el equipo o cada una de las personas puedan realizarlas.

Hay que tener en cuenta que por algún motivo hay algún problema y algo no podemos realizar, se puede definir junto con el Product Owner eliminar algún ítem, pero tiene que ser en conjunto si o si con el Product Owner.

Esas tareas que se definen dependen de cada uno de los miembros del equipo.

***Producto de software potencialmente listo para producir***

Parte del producto desarrollado en un sprint en condiciones de ser usado en el ambiente correspondiente.

Cuando termine el Sprint se debe tener un entregable de ese producto que yo pueda desplegarlo en producción. Se define como “potencialmente listo para producción”, porque no necesariamente yo voy a definir que ese incremento vaya a desplegarse en producción pero está construido de tal manera que si se decidiera se puede.

**Reuniones o ceremonias**

**Obligatorias Opcionales**

* Sprint Planning - Story time / Product Backlog
* Daily Refinement
* Review
* Retrospectivas

Todas estas reuniones manejan el concepto de Time Boxing (tienen un tiempo), están planteadas en términos del tiempo.

***Sprint Planning***



Parte de lo que hacemos es tomar el Product Backlog, priorizar (porque por mas que ya tengamos priorizado puede que necesitemos hacer una modificación de esa prioridad o valores de negocios) y elegir cuáles son los requerimientos, requisitos o user de ese Product Backlog que vamos a incluir en el próximo Sprint.

En el próximo Sprint, ¿Qué vamos a hacer?

Por un lado vamos a definir el **objetivo** del Sprint, tenemos que decir cómo vamos a alcanzar ese objetivo, es decir, las users que vamos a elegir. A partir de las user que van a estar incluidas, crear el Product Backlog con las tareas.

Vamos a crear todas las tareas que necesitamos para construir nuestro entregable potencial de las users desglosadas en las tareas, y lograr una estimación del Sprint Backlog en horas.

Como entregable o resultado tenemos el **objetivo** del sprint, todo sprint tiene que tener un objetivo que es el que me permite saber si lo alcanzamos o no cuando terminamos, las **users o los features** que vamos a incluir en el sprint, la **lista de tareas** para poder construir cada user o feature, y las horas de esas tareas.

Una cosa muy importante es la **capacidad del equipo**. Porque nosotros tenemos el Product Backlog, la lista completa, tenemos las que están más arriba. Cómo hago de esas users o features que están arriba del Product Backlog, para saber si en mi Sprint Backlog tomo 3,4,5…? ¿Cómo sé cuáles voy a incluir en mi Sprint? Esto depende de la capacidad del equipo, de cuántas de esas User pueda construir.

Sin la capacidad del equipo no tengo forma de decir cuáles son las features que voy a incluir en el Sprint Backlog.

***Daily meeting - Reuniones de Scrum diarias***

Reuniones de scrum cortas (15-30 min) donde el facilitador es el Scrum Master, se analiza el estado, asiste todo el equipo y se contestan 3 preguntas.

1. ¿Qué se completó desde la última reunión?
2. ¿Qué obstáculos se presentaron?
3. ¿Qué vas a hacer hasta la próxima reunión?

No tiene que ver con un seguimiento de parte del Scrum Master como si fuera un líder de proyecto, sino que tiene que ver con un equipo auto organizado que se compromete delante de sus pares. El concepto es diferente, y si bien el Scrum Master cumple el rol del facilitador, no es un rol de control. Es un rol que hace que el resto del equipo pueda ser ayudado a cumplir con las pautas que necesite.

Las Daily no son para soluciones de problemas, ni para dar un status report al Scrum Master. Se trata de compromisos delante de pares.

***Sprint Demo/review***

Dura más o menos unas 4 horas, el equipo presenta el incremento del producto, el producto potencialmente listo para ponerse en producción.

Participa todo el equipo, se discute acerca de lo que el producto incluye, y se le acepta o no el producto que construyeron. El PO le dice al equipo si eso era o no lo que se esperaba.

Esto es importante porque las user stories que fueron implementadas por el equipo, y son aceptadas, son las que realmente cuentan. Las demás, volverán al Product Backlog para corregir lo que sea necesario.

El PO es el que da el ok de lo que se construyó. Si lo vemos en término de user stories, nos sirven los criterios de aceptación porque son los que nos ayudan a definir cuáles son las cosas que el PO va a mirar para darnos el ok o no de la user que se está implementado. Scrum no sólo se hace con user stories, se pueden usar otras técnicas.

* El equipo presenta el incremento desarrollado a gerentes, clientes, usuarios y al dueño del producto.
* Informal
* Regla de 2 hs de preparación
* No usar diapositivas
* Todo el equipo participa
* Se invita a todo el mundo
* Se reportan los problemas encontrados
* Cualquier ítem puede ser agregado, eliminado, re-priorizado
* Se estima el nuevo Sprint y se asignan tareas

***Sprint retrospective***

Tiene que ver con la retrospectiva y con la idea de que Scrum tiene que ver con planificar, ejecutar y reflexionar. La retrospectiva abarca la parte de la **reflexión.**

Tiene una duración de 1 a 4 hs. Participa todo el equipo (Scrum Master, PO, Equipo, posiblemente clientes y otros), se realiza después de terminar el Sprint y lo que decimos es qué cosas funcionaron en el Sprint y cuáles no. Las que funcionaron para repetirlas y las que no para corregirlas, para generar acciones correctivas.

La retrospectiva no es un espacio para la catarsis, ni para quedarse callado y que una sola persona cumpla para llevar a cabo la ceremonia. La idea es encontrar la manera de que todos los del equipo participen, y es por esto que se usan distintas dinámicas (que lo que se comparta tenga con cierto anonimato) para que todos se animen a aportar y a partir de esto se pueda mejorar los próximos Sprints.

Las acciones correctivas que se generan tienen que ser concretas y tienen que estar en un marco definido (elegir uno o dos), para poder plantearlas para trabajarlas en el próximo Sprint y ver cómo resulta.

***Story time o grooming (Opcional)***

Tiene que ver con poder discutir algunas cuestiones de los ítems del Backlog en términos de prioridad para hacer foco en los criterios de aceptación. Si nosotros hacemos bien algo antes, probablemente cuando lleguemos a la review, como esta discusión la tuvimos antes de empezar a trabajar con la user, no debería haber sorpresas con respecto a lo que tiene que ver con la aceptación, si esos criterios estuvieron bien definidos.

Esta ceremonia es interesante para que la discusión surja antes de la review. En la review está la aceptación del producto. La story time es una ceremonia que se hace antes del desarrollo de las users o features, y después iterativamente en cada Sprint. Sirve para discutir sobre los ítems del Backlog que podamos no tener claridad y determinar cuál es el criterio de aceptación,es decir, cuando desarrollemos está features está user, ¿Cuál es el criterio de aceptación que el PO va a tener a la hora de aceptarla?, es una ceremonia que refuerza este concepto de profundizar en las conversaciones lo que tenemos definido como User, esta ceremonia se realiza antes del desarrollo.

La idea es llegar a la sprint planning con el detalle que me hace falta, más que para la prioridad, para la estimación, porque el problema tiene que ver con la estimación.

Los ítems de alta prioridad son los que están siempre arriba del Backlog, el punto es que teniendo una user, y en términos de estimación, de la complejidad y de lo que el PO espera de la user, tengo que discutirlo en profundidad antes de ponerme a desarrollarlo.

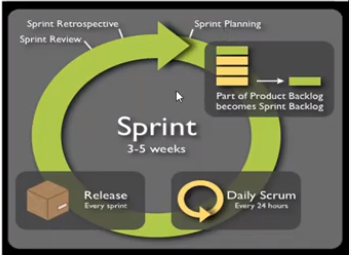
No necesariamente todos los features se van a discutir con el mismo nivel de detalle, porque en algunos va a haber más claridad que en otros.

Se la hace antes de finalizar el Sprint actual y comenzar el próximo.

La idea es que en la planning no trabajemos en esto, nos focalizamos en priorizar que voy a incluir en el Sprint. En la planning ya no deberíamos tener duda de la estimación de las users o features que tenemos incluidas.

**Pregunta:** Si hubiese spikes, ¿Se tratan en esta ceremonia? No necesariamente, porque esto tiene que ver con cosas que va a resolver el Product Owner, si la spike son cosas funcionales si, pero si tiene que ver con cosas técnicas no. La idea es que el PO nos saque toda la incertidumbre o dudas que tenemos.

**Sprint**



El time box sugerido es de 30 días, puede ser menos también, lo ideal es entre 3 y 5 semanas. Al time-box lo define el equipo. La idea es que una vez que ejecutamos o empezamos a ejecutar el Sprint, todo lo que pasa en ese momento no se puede cambiar. No se puede agregar funcionalidad, no se puede cambiar el alcance. Para eso definimos el objetivo del Sprint Backlog para tener en claro que es lo que vamos a hacer en ese Sprint, y no se pueden modificar las reglas.

Todo lo que ocurra luego de terminado el Sprint, nos va a permitir hacer las modificaciones en el próximo Sprint Backlog en función de lo que hayamos aprendido en el Sprint.

**Objetivo del Sprint**

Todo Sprint tiene un objetivo, se define en la planning, es en donde decimos cuál es la declaración de lo que vamos a hacer en el Sprint. Es un objetivo claro a alcanzar.

**Reglas del Sprint**

Tienen que ver con la premisa que Scrum es amigable para los entornos cambiantes y tiene flexibilidad para atender a los cambios siempre y cuando no ocurran durante la ejecución de Sprint. Mientras la Sprint se está ejecutando no hay ni cambios ni interrupciones.

* Foco en la tarea
* Sin interrupciones
* Sin cambios
* El mismo equipo puede descubrir más trabajo a ser hecho

**Elementos de un Sprint**

Reuniones que se ejecutan, las daily. Con el incremento del producto que se obtiene, que tiene que tener un incremento con respecto al producto ya existente, o el producto anterior; y con el compromiso de los miembros del equipo.

* Reuniones de Scrum
* Se produce un incremento usable y viable
* Los incrementos se basan en un producto anterior
* Compromiso de los miembros de la asignación

**Mecánica de los Sprint**

Hacemos foco en los time box, hacemos foco en que se congelan algunas variables del proyecto durante ese tiempo del Sprint. El ambiente se congela, se establecen ciertas reglas con las que vamos a trabajar y mientras se ejecuta el Sprint, eso no cambia.

Puede haber algunos cambios de funcionalidad mínima acordados con el equipo (sacar o agregar una funcionalidad) siempre y cuando se mantenga el objetivo del Sprint. Si se saca una funcionalidad y cambia el objetivo, probablemente implique la cancelación del Sprint, porque si el objetivo no se cumple, no tiene sentido seguir con la ejecución del Sprint.

Cuando ejecutamos el Sprint hay que asegurarse que el objetivo se cumpla.

En un Sprint se congelan 3 o 4 variables de un proyecto:

* Tiempo: 30 días
* Costo: salarios + ambiente
* Calidad: generalmente un estándar organizacional

**Pregunta:** Si tenemos un ambiente de desarrollo y otro productivo, y se reportan errores del productivo, los equipos deben ser diferentes? Si, en principio sí, salvo que dentro de mi Sprint Backlog decida incluir la corrección de un bug. Lo que no puedo hacer es atender los errores en producción mientras generó un nuevo incremento del producto y ya no lo había incluido dentro del Sprint Backlog. La corrección de algo que haya sido incluido en el Sprint en principio podría ser válido.

**Tareas obligatorias**

En un Sprint hay 2 tareas mandatorias

* Reuniones scrum diarias: participan todos los miembros del equipo
* Backlog del Sprint: debe ser actualizado y con los últimos estimados de los desarrolladores

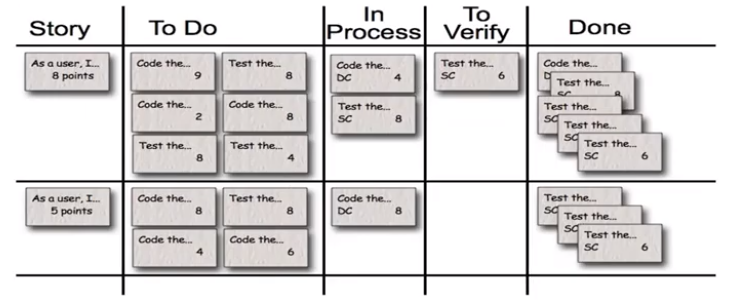
**Cancelación de un Sprint**

* Se puede cancelar un Sprint si las circunstancias hacen que no sea necesario
* El objetivo se vuelve obsoleto
* Las condiciones técnicas o de mercado cambian
* Decidido por el equipo
* Porque no se puede alcanzar el objetivo
* Se encuentra un problema muy grande
* Cancelar un sprint es un costo

**Herramientas de Scrum**

***Taskboard***

Tiene distintos componentes, tiene que ver con la actividad, la estimación, con quién se va a hacer cargo.

Las tarjetas que surgen del Sprint Backlog tienen la **actividad**, tienen el **código del ítem** incluido en el Sprint Backlog, la **persona** que se va a hacer cargo, el **numero de dia de la Sprint**, y el **tiempo estimado** en horas. 

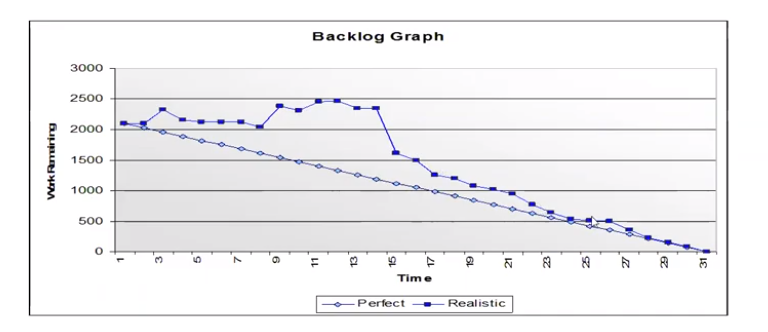
Con este taskboard obtenido en la reunión de la planning, lo podemos dividir en términos de

* To Do: lo que hay que hacer
* In process: lo que está en proceso
* To verify: lo que hay que verificar
* Done: lo que ya está hecho

Los taskboard deben tener suficiente granularidad como para que podamos incluir muchas tareas dentro de lo que tiene que ver con la resolución de cada uno de los items que tenemos que trabajar. Por esto es que hay que descomponer en una granularidad que fuera entre 4 y 16 horas más o menos.

***Gráficos de Backlog***

* La gerencia necesita datos acerca de:
* Progreso del Sprint
* Progreso del Release
* Progreso del Producto
* El Backlog de trabajo es la cantidad de trabajo que queda por ser realizado
* Tendencia del Backlog: trabajo que queda vs tiempo

******

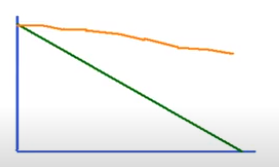
Nos sirven para poder medir el progreso o cuánto nos queda por hacer, dependiendo del gráfico que hacemos y de la mirada que queremos darle. Para eso, los gráficos tienen en uno de los ejes el **trabajo que falta realizar,** y en el otro de los ejes el **tiempo**. Si el trabajo para realizar está en el eje de las Y, en el tiempo que tenemos definido, se supone que los 2000 puntos definidos deberíamos poder consumirlos o quemarlos en los 30 días que es lo que dura el Sprint. Por eso hay una línea perfecta que va desde el trabajo total remanente que tenemos que hacer hasta el ultimo dia, se supone que el ultimo dia tenemos que terminar de hacer todo el trabajo planificado.

Obviamente esto es en un escenario perfecto. La parte realista es lo que realmente trabajamos, en este caso, no se avanzó prácticamente en los primeros días, a medida que se va avanzando, se subsanan los impedimentos, se corrigen las cosas que hay que corregir y esa línea se va asimilando a la progresión lineal de la línea perfecta, llegando al último día del Sprint teniendo el trabajo finalizado que se había planteado. Lo que aparece en color azul es lo realista, lo que realmente nosotros vamos trabajando, en lo realista los primeros días no hemos avanzado, esto pasa porque después se subsanan los experimentos, se corrige lo que se debe corregir y la línea realista se asemeja a la perfecta.

A este gráfico se le llama Burndown Chart, porque nos muestra los puntos quemados o el trabajo realizado, nos dice cuánto nos queda por hacer en función de lo que ya hicimos, y el tiempo restante transcurrido.

**¿Qué pasa si el Burndown chart tiene esta forma?**

***Opción 1***



Significa que se terminó la Sprint y nos quedó trabajo por hacer, no pudimos cumplir con el objetivo. En este caso NO podemos asegurar que el problema haya sido la estimación.

Opciones de problemas

* Se haya estimado mal
* No hubo nadie atento a plantear las correcciones necesarias o subsanar los impedimentos para que se pudieran corregir las dificultades, y esta línea que venía complicada en algún momento se acomode.

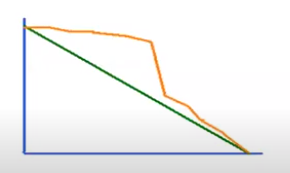
¿Cuál es el resultado de ese Product Backlog? Que no pudimos cumplir.

**Pregunta**: ¿Es culpa del Scrum Master? Depende. Si la estimación fue mal hecha, el problema es de todo el equipo. Si hubo dificultades para resolver los impedimentos, quizás el SM haya tenido parte de responsabilidad en no reaccionar para evitar la situación. Capaz puede pasar que se te vaya gente del equipo, y la capacidad que se determinó inicialmente no es la real.

Uno espera que si vamos viendo o analizando cómo se comporta el Burndown chart, antes del día 30 deberíamos haber cancelado el sprint. No tiene sentido seguir avanzado. El Burndown chart se genera a lo largo del Sprint, y todos los días hay que marcar el trabajo restante que queda para los días siguientes, uno se va dando cuenta de antemano la tendencia, si no llegan se cancela antes de finalizar el Sprint.

* No todo lo incluido en el sprint será entregado en la fecha propuesta. El Backlog no fue modificado para alcanzar la fecha (remover requerimientos o modificar la fecha)
* Posibles causas
* Pobres tecnicas de estimacion
* Mal manejo de riesgos
* Rotación del equipo o renuncias
* El gráfico muestra que el SM y el PO no reaccionaron para evitar la situación. Con el PO también hay que llegar a un acuerdo para cancelar la Sprint.

***Opción 2***

****

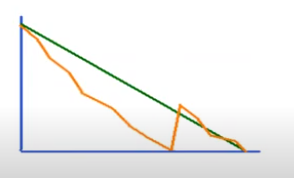
Se cumplieron a tiempo las acciones correctivas. El inicio del gráfico es parecido al anterior, pero en algún punto, hubo acciones correctivas y se pudo resolver. El equipo reaccionó. A lo mejor una de las posibilidades puede haber sido ponerse de acuerdo con el PO y quitar algunos requerimientos mientras se mantenga el objetivo del Sprint. Claramente el equipo hizo algo para poder reaccionar.

**Pregunta:** Se quedaron hasta tarde para llegar a cumplir. No, no es la idea, el problema con esto tiene que ver con lo sustentable que es en el tiempo.

¿Puedo tener un equipo que trabaja siempre el doble de lo que debería? No, porque en algún momento el equipo se quema. Más allá de si alguien insiste en que el equipo trabaje o no más horas de la cuenta, tiene que ver con que a lo largo del tiempo esto no rinde. Se puede hacer una excepción.

* Este BC es mucho mejor que el anterior
* Al comienzo de la Sprint hubo problemas, pero en el medio el equipo de trabajo reaccionó y tomó medidas correctivas. Por la línea se nota que el PO quito requerimientos para mantener la fecha de release.
* También pone en evidencia un mejor manejo de riesgos lo que facilitó aumentar la velocidad en la segunda parte del Sprint.

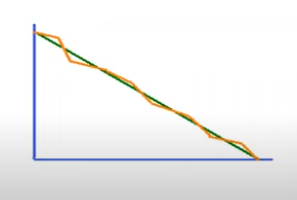
***Opcion 3***

****

Se la complicaron cuando lo habían logrado, o encontraron algún problema, agregaron funcionalidad. Se estimó mal, terminamos antes, y agregamos cosas hasta que termine el Sprint. Claramente hay un problema con la **estimación**. La línea sube porque lo más probable es que se hayan agregado requerimientos para que no queden días sin hacer nada.

* Este BC también tiene características negativas.
* Este no es un ejemplo de desarrolladores con las mejores habilidades técnicas. Demuestra pobres tecnicas de estimacion.
* El PO y el SM deberían haber agregado más requerimientos para ese release.

***Opción 4***

****

Escenario ideal, la línea nunca es perfecta, pero casi. Hicimos todo lo que prometimos, prácticamente vamos en línea con lo lineal.

* Ideal
* Buenas tecnicas de estimacion
* Buena velocidad de desarrollo
* Buen manejo de impedimentos
* Todos los requerimientos prometidos son los entregados
* Buen manejo de la asignación de tareas (tareas cortas, reporte diario)
* Capaz demasiado bueno para ser verdad…

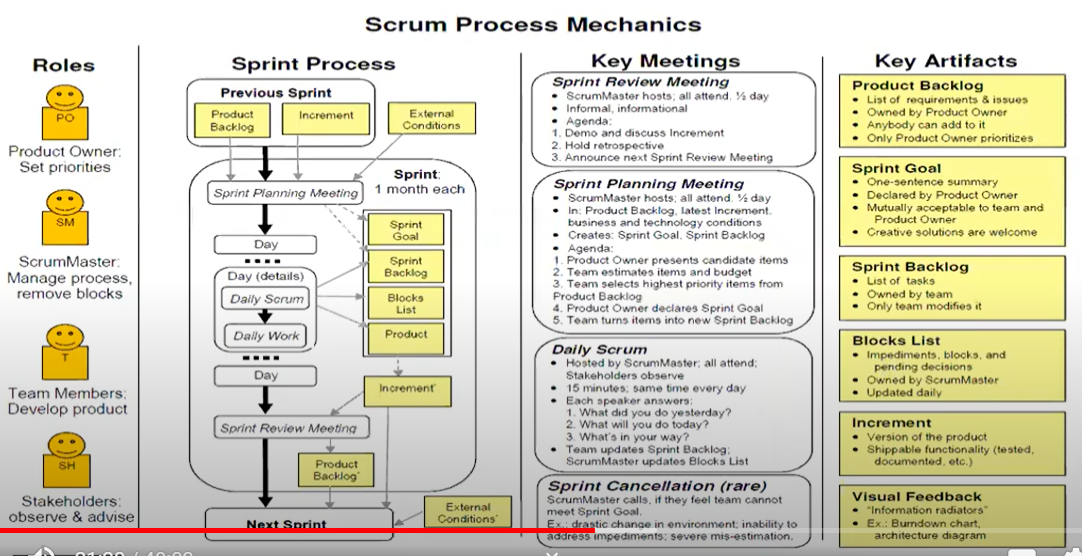
Cuando definimos la capacidad del equipo, lo que definimos es cuantas horas el equipo va a tener disponibles. Si tengo 4 personas, y esas personas trabajan 8 horas por 5 días a la semana. De las 8 horas que van a trabajar, como planteó esfuerzo, realmente cuántas de esas 8 horas son realmente productivas (se para para ir al baño, para almorzar, etc), claramente en equipos altamente productivos nunca se calcula más de un 70% de las horas que tenemos.

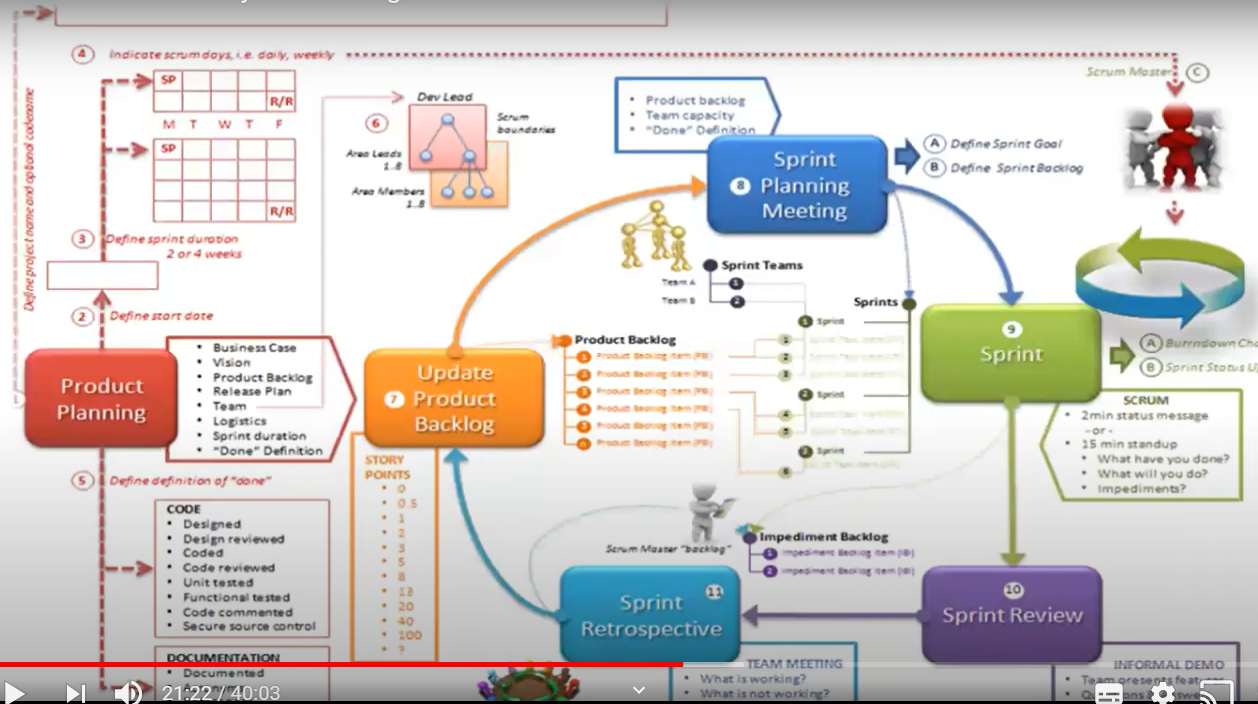
Cuando definimos la capacidad nos comprometemos a hacer las tareas de acuerdo a la capacidad que definimos, acá empieza a jugar la estimación. Cómo hicimos la estimación en función de la realidad del tiempo que nos va a llevar. Nos tenemos que asegurar que definimos la cantidad de horas que tiene disponible el equipo en función de lo más cercano a la realidad.

**Beneficios de Scrum**

* Se gestionan los cambios de requerimientos
* Se incorpora la visión de mercado
* Los clientes ven incrementos que refinan los requerimientos en un tiempo razonable
* Mejores relaciones con el cliente

**Resumen de Scrum**





**¿Para quién es Scrum?**

“Scrum **no es para todos**, sino para aquellos que tienen que trabajar con sistemas funcionando con la complejidad de tecnologías inestables y el surgimiento de requerimientos”

Este concepto es muy importante. **No** es para todos los proyectos, ni para todos los equipos. Cuando vimos los cimientos dijimos las características que debían tener los miembros del equipo, ser auto organizados, que todos puedan hacer todo. No todos los equipos pueden configurarse de esta manera y no todos pueden aplicar Scrum. No en todos los proyectos resulta de utilidad.

**Métricas Ágiles**

*Regla de Oro Ágil sobre Métricas*

* La medición es una salida, no una actividad
* Una filosofía minimalista sobre las Métricas: Medir la que sea necesario y **nada más**

A diferencia de lo que puede pasar con las métricas tradicionales, en las métricas ágiles hay una mirada minimalista. Las métricas tradicionales son muchas, uno las puede elegir en función de lo que necesite en el proyecto y apuntan a la parte del monitoreo y control (las actividades del monitoreo y control nos permiten detectar problemas y generar las acciones correctivas cuanto antes). Cuando hablamos de las métricas ágiles, hablamos de medir solamente lo estrictamente necesario y nada más. Cuando nos referimos a “lo estrictamente necesario” tenemos en cuenta una definición que viene del Manifiesto Ágil que dice que *la principal medida de progreso es el software funcionando*. Este criterio es el que nos va a servir de base para definir cuáles son las métricas que nosotros vamos a utilizar. Asi como cuando hablábamos de la documentación y decíamos que no es que no hay documentación, sino que solamente se trabaja con la necesaria. Aca con las métricas es lo mismo, no es que no tengamos métricas sino que tenemos una mirada mínima sobre las métricas con las que vamos a trabajar.

**Dos principios ágiles que guían la elección de las Métricas**

“Nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software valioso”

“El Software trabajando es la principal medida de progreso”

El otro lineamiento que viene del Manifiesto Ágil es satisfacer al cliente por medio de entregas tempranas y continuas.

**Capacidad**

Capacidad es una estimación de cuánto trabajo puede completarse en un periodo de tiempo dado.

Basado en la cantidad de tiempo ideal disponible del equipo.

Se puede medir en:

* Esfuerzo (horas)
* Puntos de Historia (Story Points)

Es un poco tramposa porque en realidad la vamos a definir en términos de estimación. Es loco porque la métrica justamente es tomar una medición de algo que ya ocurrió. La capacidad se define como una métrica pero está basada en una estimación.

**¿Qué significa la capacidad?** Es cuánto trabajo puede completarse en un periodo de tiempo dado. El periodo de tiempo dado que nos interesa ahora es el *Sprint*. Buscamos definir cuánto trabajo podemos hacer en el Sprint.

**¿Para qué nos sirve saber esto?** Para la Planning, porque si no tengo la capacidad, es decir, no definí cuánto trabajo puedo hacer en el Sprint, ¿Cómo voy a saber cuales son las features que voy a incluir en el Sprint? Cuando corto de esa lista de prioridad? Corto en función del trabajo que puedo hacer.

Lo ideal sería poder definir esta capacidad en función de Story Points porque como las features o Users las tenemos estimadas en Story Points, si quiero decir cuánto trabajo voy a poder hacer lo ideal sería poder definirlo en Story Points para cuando empiece a recorrer la lista de lo que voy a incluir en el Sprint Backlog hago la comparación entre la suma de los Story Points versus la capacidad. ¿Qué dificultad nos podríamos encontrar? Tener un equipo inexperto. Porque en realidad el Story Point es una medida que surge de la experiencia que tiene el equipo y que es propia para ese equipo.

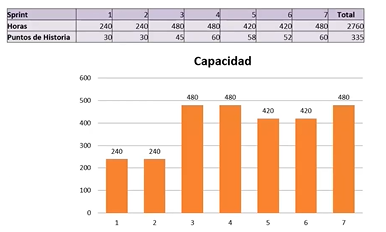
Lo que se suele hacer es medir el esfuerzo en **horas** que es más tangible y más fácil.

**Horas de Trabajo Disponibles por dia** (WH) x **Dias Disponibles Iteración** (DA) = Capacidad

WH x DA = Capacidad

*Ejemplo*

* Equipo de 8 miembros
* 4 miembros disponibles los 2 primeros sprints
* 1 miembro se casa en sprints 5 y 6 (no va a estar disponible)
* 6 horas de trabajo



Más o menos queda la capacidad así en horas.

Normalemer la capacidad se ratifica en la Planning porque nos va a servir para armar un plan pero debemos reafirmarla antes de empezar el Sprint que esa capacidad que definimos se mantiene. Si se enferma una persona en el medio hay que corregir la capacidad que habíamos estimado para los distintos Sprints. Tratamos de convertir esa capacidad en alguna medida de puntos de historia para después poder trabajarlo con el equipo y poder hacerlo dentro de la Planning. No necesariamente hay una relación lineal entre las horas y los puntos de historia, porque justamente hablábamos que para poder definir los puntos de historia íbamos haciendo un aprendizaje en términos del equipo.

*Consideraciones*

* Individuos calculan capacidad realista
* Aplicar estimaciones honestas a sus tareas
* Considerar una capacidad máxima de 50%-70%
* Comprender la capacidad a largo plazo con la velocidad y los puntos de historia
* ¿Cuál es el promedio de finalización de un punto de historia para un equipo/individuo?

Es importante tener una capacidad realista del cálculo, es decir, no decir trabajamos 8 horas entonces son 8x5 = 40 horas semanales porque sabemos que no son todas horas reales de trabajo.

Poder empezar a trabajar y modelar esa capacidad en el tiempo en función de lo que realmente va pasando. De la experiencia que vamos adquiriendo.

Hay una palabra clave: **velocidad**.

**Pregunta:** ¿Sería correcto en los primeros Sprints dejar un margen hasta acumular experiencia necesaria? No, la idea no sería que dejemos margen, la idea sería que lo hagamos lo más a conciencia que se pueda sabiendo que los primeros Sprint nos vamos a equivocar. No importa. Probablemente la velocidad sea cero, es decir, vamos a haber estimado cosas que no vamos a poder hacer. Los primeros Sprint son de aprendizaje, pero para poder hacer ese aprendizaje tengo que tratar de ser lo más honesto posible, trabajar en función de lo que realmente creo.

**Velocidad**

Para comprender la capacidad, poder ajustarla e ir puliendo la estimación aparece otra medida que es importante que es la velocidad.

Velocidad es…

* Una observación empírica de la capacidad del equipo para completar el trabajo por iteración
* Comparable entre iteraciones para un equipo dado y un proyecto dado

NO es…

* Una estimación
* Un objetivo a alcanzar
* Comparable entre equipos
* Comparable entre proyectos

A diferencia de la capacidad que la trabajamos en la Sprint Planning, la velocidad si es concretamente una medida, no es una estimación, es concretamente una medición, una observación. La obtenemos concretamente en la ceremonia de la **review**.

**¿Cómo vamos a medir la velocidad?** Lo que hacemos es sumar la cantidad de puntos de historia de las features o de las Users terminadas que han sido probadas por el Product Owner. Si hubiera dicho que mi trabajo a realizar eran 25 story points y resulta que mi Product Owner, de los 25 puntos de historia me aprobó solamente 2 Users que suman 10 puntos de historia, la velocidad es de 10. Lo que realmente mi Product Owner me aceptó en la review, sumo los puntos de historia y eso me da la velocidad. ¿Por qué decimos que la velocidad retroalimenta la capacidad? Porque en función de lo que me pasó, en función de la cantidad de puntos de historia que realmente fueron aceptados al final del Sprint, me va a servir de indicador a la hora de hacer mi nueva planning, cuánto es realmente el trabajo que yo puedo realizar. Si yo dije que mi capacidad eran 25 puntos de historia y resulta que solamente pude hacer 15 y, supongamos que el PO me aprobó los 15 pero solamente pude hacer 15, cuando plantee de nuevo la capacidad en la siguiente Planning tendré que preguntarme si son 25 o en realidad son los 15 que obtuve. Lo que nos permite la velocidad es hacer los ajustes necesarios en la capacidad para que justamente vayamos aprendiendo a medida que los Sprints van corriendo.

Al principio, en los primeros Sprints probablemente nuestra velocidad sea cero. Probablemente no tengamos ningún valor de velocidad que nos resulte satisfactorio. No importa, porque es parte del aprendizaje. El equipo tiene que conocerse y nuestro objetivo es que a medida que los Sprint transcurran el equipo logra una velocidad estable que se puede predecir.

Uno espera que en los primeros Sprint la velocidad no sea la que vamos a lograr después y que en algún momento del tiempo la velocidad que nosotros logremos sea más o menos siempre la misma, salvo que algún miembro del equipo se vaya o no este trabajando. Si se logra una velocidad estable quiere decir que justamente la experiencia del equipo lo enriqueció y sirvió para justamente obtener esta velocidad estable. La capacidad se vuelve más fácil, porque ya puedo predecir que es lo que va a pasar.

Es importante tener en cuenta en ese contexto que, como los story points son una medida propia del equipo, lo mismo pasa con la velocidad. No es comparable entre equipos. No puedo decir que este equipo tiene una velocidad de 25 story points y este otro una de 35 entonces el de 35 es mejor. Son valores que no son comparables. Tampoco son comparables entre proyectos que ejecute el mismo equipo. La velocidad nos sirve cuando trabajamos comparando lo que se hace en un Sprint con respecto al otro, pero no es una medida relativa que nosotros podemos hacer comparaciones entre equipos ni podemos decir: “nuestro objetivo como equipo es alcanzar esta velocidad”, no sirve para esto. La velocidad es una medida que nos permite llegar a ese nivel en donde nosotros podemos predecir o nos puede servir para predecir después que es lo que vamos a poder hacer en el próximo Sprint en función de esa velocidad estable que nuestro equipo, a medida que fue trabajando juntos y fue trabajando de una manera autoorganizados fue logrando.

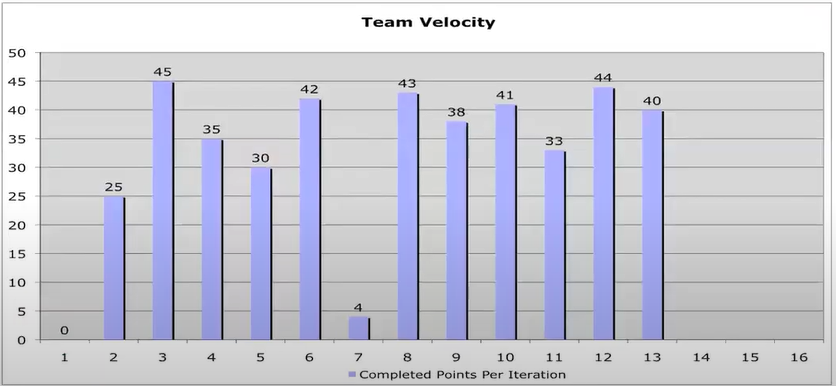
Sería para un mismo equipo o para un mismo proyecto. Nos permite retroalimentar esa medición de la capacidad que no deja de ser una estimación.

**Pregunta:** Cuando se habla de que el PO aprueba o no las User Story, ¿Qué pasa cuando no las aprueba? Porque esa User fue desglosada en tareas en el Sprint y fueron realizadas y se marcó como que está terminada la tarea, ¿Se vuelve al inicio?

A la User hay que volver a meterla en el Product Backlog. Dependiendo del motivo por el cual no se aprobó hasta podría ir al principio del Product Backlog o quizás no entra en el próximo Sprint y queda más abajo. Hay una premisa que tiene que ver con lo que se plantea en Scrum que dice que a medida que se van haciendo entregas y que el PO puede visualizar el producto funcionando también puede darse cuenta de cosas que no se había dado cuenta cuando a lo mejor la feature se definió en términos teóricos. No necesariamente lo que no se acepta es responsabilidad del equipo, también puede tener que ver con que el PO se dió cuenta que no era eso lo que quería, que así como está planteado no le sirve. También hay un trabajo ahí que tiene que ver con decir “esta feature tiene que volver al Product Backlog a volver a ser priorizada y reformulada en función del motivo por el cual no se aceptó”. Un ejemplo podría ser que había un criterio de aceptación que no lo pusimos pero estaba, al omitirlo es algo que no tuvimos en cuenta cuando construimos la User y resulta que hay algo que nuestro PO supuso que se iba a hacer de una manera o que iba a funcionar de una manera y ahora lo visualiza y no. Ese feature tendrá que volver a ir al Product Backlog, tendrá que ser priorizado de nuevo, tendrán que incluirse cuando hablemos los criterios de aceptación que falten y cuando se enuncien las tareas para ejecutar por supuesto que el trabajo que ya tenemos hecho no lo vamos a hacer de nuevo. Tenemos en cuenta a la hora de listar las tareas de que lo que ya está hecho nos sigue sirviendo.

**Pregunta:** la velocidad sería una medición que sirve para refinar la capacidad del equipo pero no significa que se tenga que modificar la capacidad de todos los Sprint a causa de los cambios de velocidad?

Esta perfecto. La capacidad es una estimación. Estamos tratando de predecir cuánto trabajo vamos a poder hacer en el próximo Sprint. No necesariamente lo único que vamos a mirar para eso es la velocidad. Por ejemplo, si venimos con una velocidad estable y resulta que un miembro del equipo se va de vacaciones, la velocidad nos va a servir para determinar la capacidad, pero en ese Sprint la capacidad se va a ver disminuida porque tenemos un miembro del equipo menos. La velocidad sirve como input para refinar la capacidad pero además nos sirve para poder ver cuál es la medida del trabajo que el equipo va realizando.



El gráfico muestra la velocidad posible de un equipo. Además de servirnos de retroalimentación para la capacidad, también nos permite ver cómo trabaja el equipo, que es lo que logra el equipo. Otras de las cosas que nosotros deberíamos poder ver es que nuestro equipo en algún momento debería tener una capacidad más o menos estable, en el gráfico sería alrededor de 40. No solamente nos permite servirnos de entrada para la capacidad, sino además ver cómo va el equipo. En el primero tuvo 0, se ve lo que dijimos de poder ir aprendiendo.

**Unidad de Medida de la Velocidad**



*¿Qué cuenta para la velocidad?*

Solo cuenta el trabajo completado para la velocidad

La unidad de medida tiene que ver con los Story Point y solo cuenta el trabajo completado. Si nosotros desglosamos las tareas, hicimos la mitad de una feature, no cuenta para la velocidad, solamente cuentan las historias terminadas. No una parte.

**Pregunta:** En la review donde el PO puede decir que una User no era apta y descartarla, ?No se debe hacer eso antes? No hay una instancia intermedia donde pueda ya ver esos resultados intermedios y corregirlos?

No, lo que sí puede ocurrir es que si el equipo tiene dudas llame al PO para subsanarlas. Pero no hay una instancia intermedia. El Sprint dura poquito tiempo, tampoco hay demasiado espacio para hacer otra demo. Lo que sí está habilitado es que si tenemos dudas a la hora de desarrollar la User lo llamemos al PO y le preguntemos.