Resumen Parcial 1:

UNIDAD N° 1

¿Qué es el software?

El software es información o conocimiento que aparece en niveles distintos de abstracción y con diferentes representaciones, en general, es un set de programas y la documentación que acompaña. La intención de la Ingeniería del SW es administrarlo correctamente. Existen tres tipos básicos de SW:

- **System Software** → Un sistema operativo
- **Utilitarios** → Winrar
- **Software de Aplicación** → Office

Tipos de Productos:

- **Productos Genéricos**: Son sistemas aislados producidos por una organización de desarrollo y que se venden al mercado abierto a cualquier cliente que le sea posible comprarlos. La organización que desarrolla el SW controla su especificación. Un ejemplo: Word
- **Personalizados o A Medida**: Son sistemas requeridos por un cliente en particular. Un contratista de SW desarrolla el mismo especialmente para ese cliente. Por lo general, la especificación del SW es desarrollada y controlada por la organización que lo compra. Ejemplo: Sistema de control de tráfico aéreo.

Ingeniería de software

Es una **disciplina de la ingeniería** cuya meta es el desarrollo costeable de sistemas de software (este es abstracto, intangible y fácil de modificar). La ingeniería de software comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este después que se utiliza.

Diferencias entre SW y Manufactura:

- El SW es menos predecible.
- El SW tiene un nivel de complejidad inherente al ser intangible.
- El factor más importante en la producción de SW son las personas que están atrás de la misma, es por esto que se lo denomina “peopleware”, el costo más significativo al hacer SW es la gente.
- No hay producción en masa, casi ningún producto de SW es igual a otro.
- No todas las fallas son errores.
- El SW no se gasta ni rompe, necesita continuo mantenimiento y evolución ya que esta inserte en un contexto que cambia constantemente.
- El SW no está gobernado por las leyes de la física.
- La forma de hacer SW es mediante proyectos.

Crisis del Software

Antes se creía que el software era una inspiración divina, que se hacía con inesperadas fuentes de inspiración. La noción de la ingeniería de software surgió en 1968 debido a la crisis de software **causada** por la introducción de nuevas computadoras basadas en circuitos integrados que provoco la posibilidad de desarrollar software más grande y complejos, esto llevo a que los proyectos se atrasaban, otros se cancelaban, se sobrepasaban los presupuestos, como **consecuencia** se generaban software de mala calidad y desempeño pobre que requerían intensas actividades de mantenimiento.

Entonces fue evidente que para crear un software de esta magnitud tomar un enfoque informal no era adecuado, se necesitaban nuevas técnicas y métodos para controlar la complejidad de estos sistemas. Algunos problemas son:

- La versión final no satisface las necesidades de cliente.
- No es fácil extenderlo y/o adaptarlo.
- Mala documentación.
- Mala calidad, la calidad del SW se obtiene a partir de la comparación del mismo con los requerimientos, principalmente funcionales, como se falla en el establecimiento de los mismos, se falla en la calidad.
- Más tiempos y costos que los presupuestados.

El disparador del SW fue Frederick Brooks quién crea el paradigma estructurado, este paradigma introdujo a la disciplina conocimientos importantes como el diagrama de flujo de datos, la máquina de estado, etc.

El manifiesto ágil y el movimiento a procesos ágiles nacen en 2001 donde se juntan personas importantes en el mundo del SW y firman el arreglo de manifiesto ágil poniéndose de acuerdo en algunos principios.

Lean Software Development trae la manifestación “liviana” al software. Esta filosofía tiene bastantes similitudes con la metodología ágil.

Retos que afronta la Ingeniería del software

- De **Heterogeneidad** (consiste en desarrollar técnicas para construir software que pueda operar como un sistema distribuido en redes con distintos tipos de computadoras, con sistema en otros lenguajes, etc.)
- De la **Entrega** (reducir los tiempos de entrega para sistemas grandes y complejos sin comprometer la calidad del sistema)
- De la **Confianza** (desarrollar técnicas que demuestren que los usuarios pueden confiar en el software)

Buen Software

Los atributos reflejan la calidad del software, no están directamente asociados con lo que el software hace, más bien reflejan su comportamiento durante su ejecución y en la estructura y organización del programa fuente y en la documentación asociada.

- **Mantenibilidad** (debe escribirse de tal forma que pueda evolucionar para cumplir necesidades de cambio)
- **Confiabilidad** (no debe causar daños físicos o económicos en el caso de una falla del sistema)
- **Eficiencia** (no debe malgastar los recursos del sistema, como memoria o procesamiento)
- **Usabilidad** (debe ser fácil de usar por el usuario, con interfaz apropiada y su documentación)

Proceso de Software

- Es un conjunto estructurado de actividades y resultados asociados que producen un producto de SW, estas actividades varían dependiendo de la organización y el tipo de sistema, y son llevadas a cabo por los ingenieros de software.
- Debe ser explícitamente modelado si va a ser administrado.
- Un proceso de desarrollo permite usarse con una variedad de ciclos de vida.
- Diferentes tipos de sistemas necesitan diferentes procesos de desarrollo. El proceso de desarrollo se elegirá de acuerdo al tipo de sistema que se vaya a desarrollar, el uso inadecuado del proceso puede reducir la calidad o la utilidad del producto de software a desarrollar o incrementar los costos de desarrollo.
- La IEEE define al proceso como la "secuencia de pasos ejecutados para un propósito dado" y al proceso de SW como un "conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar o mantener SW y sus productos asociados".
- Posee tres patas: La gente (entrenada, motivada, y con habilidad), las Herramientas y Equipos (que nos deben simplificar el trabajo y automaticen los procesos que se puedan automatizar), y los Procedimientos y Métodos.

Existen 4 actividades fundamentales comunes a todos los procesos:

- **Especificación de software:** Los clientes e ingenieros definen el SW a producir y las restricciones sobre su operación.
- **Desarrollo de software:** El SW se diseña y se programa.
- **Validación de software:** El SW se válida para asegurar que es lo que el cliente quiere.
- **Evolución de software:** EL SW se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

Procesos Definidos vs Procesos Empíricos:

- Los procesos **Empíricos** son aquellos en que la gente que trabaja decide como trabajarán el SW en base a sus experiencias.
    - Asume procesos complicados con variables cambiantes. Cuando se repite el proceso se pueden llegar a obtener resultados diferentes.
    - La administración y control es a través de inspecciones frecuentes y adaptaciones.
    - Son procesos que trabajan bien con procesos creativos y complejos.
    - Son los procesos usados en las metodologías ágiles.
- En los procesos **Definidos** se específica de antemano el proceso, y siempre se usa así, lo que logro es que se vuelva repetible y por lo tanto predecible, por lo que puedo estimar fácilmente (costos, tiempo, etc.) y definir indicadores. Aspiran a que se pueda controlar la calidad del producto controlando la calidad del proceso. Está inspirado en la línea de producción.
    - Asume que podemos repetir el mismo proceso una y otra vez, indefinidamente, y obtener los mismos resultados.
    - La administración y control provienen de la predictibilidad del proceso definido.

- A pesar de estar inspirado en la línea de producción, reconoce que al proceso hay que adaptarlo a la realidad del proyecto condicionada por las personas, herramientas, etc.
- Tiene un enfoque distinto al empírico: no se divide el trabajo por tareas, sino que se divide el producto en varias partes y a cada una de esas partes se le realiza todas las tareas.

Ciclos de vida (Modelos de proceso)

- Es la serie de pasos a través de la cual el producto o proyecto progresa. Es el cómo lo vamos a hacer.
- Es una representación del proceso. Grafica una descripción del proceso desde una perspectiva particular, proporcionando sólo información parcial sobre ese proceso.
- NO son descripciones definitivas de los procesos del SW, sino que son abstracciones de los procesos que pueden utilizarse para explicar diferentes enfoques para el desarrollo de SW. Puede pensarse como un marco de trabajo del proceso, que puede ser extendido y adaptado para crear procesos más específicos.
- Especifican: las fases del proceso (requerimientos, especificación, etc.) y el orden en que se llevan a cabo.
- Un ciclo de vida define como se lleva a cabo un proceso, un proceso puede usarse con distintos ciclos de vida.
- El ciclo de vida se elige teniendo en cuenta distintos factores como:

  - Riesgos técnicos
  - Riesgos de administración
  - Fundamentalmente los riesgos ayudan a elegir el ciclo de vida, los riesgos son la probabilidad de que ocurra algo que complique el éxito del proceso. Los riesgos se miran por dos factores: probabilidad de ocurrencia (si no tienen probabilidad asociada no son riesgos, sino problemas); y por impacto (cuánto daño puede hacer el riesgo si se presenta).
  - Tamaño del equipo
  - Ciclo de tiempo requerido
  - Volatilidad de los requerimientos
  - Aspectos del cliente

Tipos de Ciclo de Vida básicos:

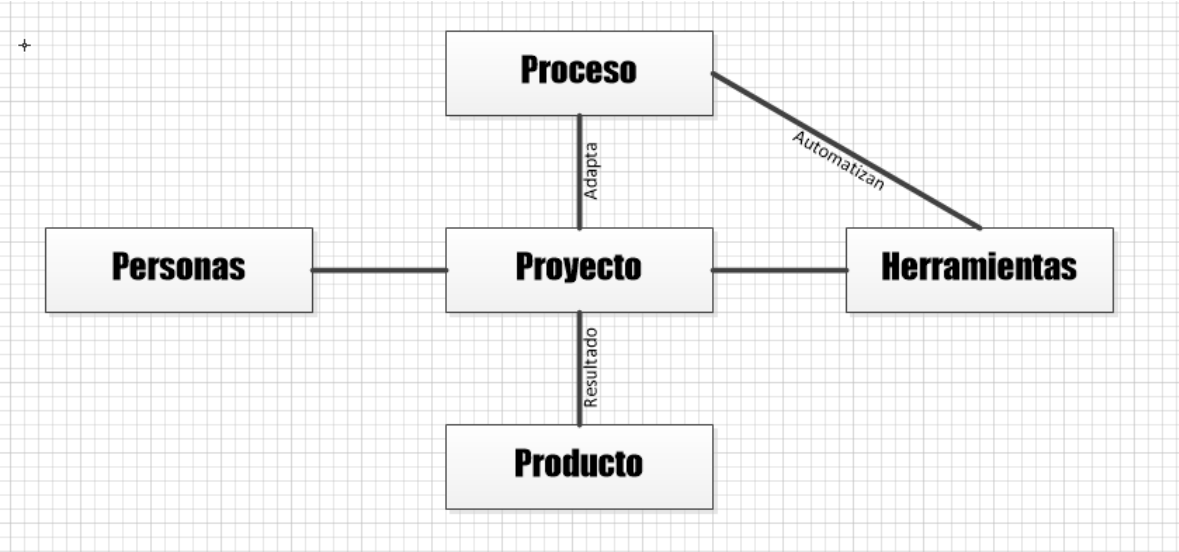
- **Secuencial (Tradicional):** ha fallado en muchos proyectos grandes y complejos, llevando a importantes demoras y sobrecostos, una actividad no inicia hasta que ha terminado la anterior.
  Su enfoque es el de hacer el 100% de una tarea antes de pasar a otra.
  Un caso es el **ciclo de vida en cascada**: ordena rigurosamente las etapas del ciclo de vida del software, para que inicie una debe tener su anterior, cualquier error de diseño detectado en etapas de prueba conduce necesariamente el rediseño y nueva programación del código afectado. Se tiene todo bien organizado y no se mezclan las fases, es bueno para proyectos rígidos y con buena especificación de requerimientos. Sin

resultados o mejoras visibles y rara vez el cliente va a establecer al principio todos los requerimientos.

- **Iterativo/Incremental:** hacer algo una y otra vez, son la tendencia de hoy. Se hace una parte de la tarea, se obtiene retroalimentación y evoluciona el producto de la tarea con cada iteración. Todos los modelos recursivos son iterativos, pero no al revés. Por ejemplo, el **Iterativo:** creado en respuesta a las debilidades del modelo tradicional de cascada, donde se saca ventaja de lo aprendido a lo largo del desarrollo anterior incrementando entregables. Permite mejorar y ajustar el proceso. Desarrollo lento por la necesidad de la retroalimentación del cliente, no es bueno para aquellos proyectos grandes en recursos y largos en el tiempo.
- **Recursivo**: significa que se comienza con algo en forma completa, como una subrutina que se llama a sí misma en un ciclo completo que comienza nuevamente. Por ejemplo, el **Espiral.**

Cuando se inicia un proyecto se debe elegir un proceso y un ciclo de vida para el mismo. Cabe aclarar que proceso y ciclo de vida no son lo mismo, el proceso es el conjunto de actividades.

UNIDAD N° 2



Tanto Proceso, Proyecto como Producto son SW. Nos tenemos que preocupar por todos. Dependiendo del tipo de proceso es que defino a la gestión de proyecto, que puede ser de dos tipos: tradicional o ágil.

Proyecto

Medio por el cual obtenemos un producto o servicio. Genéricamente manejamos recursos. Es una unidad de tiempo a la que se le asignan recursos para lograr un beneficio.

Características:

1. **Temporalidad:** Debemos poder definir una fecha de inicio y fin. Esta característica diferencia el proyecto de una línea de producción, la cual es una constante, sin fechas.
2. **Resultado único:** Cada salida que se obtiene es diferente a la salida de otro proyecto, aunque trabajemos sobre el mismo producto. Un ejemplo es Word porque voy agregando características.
3. **Asignación de recursos:** Son válidas en el período de tiempo que dura el proyecto, luego se reasignan.
4. **Conjunto de tareas:** Se lleva a cabo el proyecto mediante un conjunto de tareas divisibles, con el enfoque de "divide y vencerás".

➢ Hacemos foco en 1 y 2.
➢ El proyecto es un conductor.

Administración de Proyectos con Procesos Definidos:

La administración de proyectos de SW es necesaria debido a que la ingeniería de SW profesional siempre está sujeta a restricciones organizacionales de tiempo y presupuesto, la buena administración no puede garantizar el éxito, pero la mala sí nos asegura el fracaso del proyecto (SW entregado tarde, costos mayores a estimados, requerimientos que no se cumplen). La podemos definir también como la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de proyecto para satisfacer los requerimientos del mismo. Incluye:

- Identificar los requerimientos.
- Estableces objetivos claros y alcanzables.
- Adaptar las especificaciones, planes y el enfoque a los diferentes intereses de los involucrados.

Y está formada por dos cosas:

- Planificación.
- Seguimiento y Supervisión o Monitoreo y Control.

Planificación:

La planificación tiene como fin determinar de antemano algunas cuestiones. Planificar completamente el progreso del proyecto nos lleva a una administración efectiva del mismo. El artefacto resultante del proceso de planificación es el **plan de proyecto o plan de desarrollo de sw**.

Plan de Proyecto/Plan de Desarrollo de SW:

✓ Un plan, preparado al inicio de un proyecto, debe utilizarse como un conductor para el mismo, por lo que debe ser el mejor posible de acuerdo con la información disponible.
✓ Tiene que poder responder a las preguntas: ¿Qué hacer? (producto/servicio), ¿Cómo hacerlo? (con el proceso) ¿Cuándo hacerlo? (Recurso del tiempo) ¿Quién lo hace? (con los recursos).
✓ Depende del modelo que elijamos.
✓ El que maneja todo es el Líder de Proyecto.
✓ Consta de:

- **Nombre**
- **Objetivo del Proyecto:**
- **Alcance del proyecto:** Es todo el trabajo y sólo el trabajo que ha de hacerse para cumplir con el objetivo del proyecto. Tengo que mirar al proceso. Distinguimos:
    - Alcance del Producto: Todas las características que pueden incluirse en un producto o servicio. El cumplimiento se mide contra el Plan del Proyecto.
    - Alcance del Proyecto: Todo el trabajo y solo el trabajo que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas. El cumplimiento se mide contra la ERS.
- **Proceso + Ciclo de Vida:** El proceso define que tareas se van a realizar y el ciclo de vida me dice cómo voy a hacer las tareas. Por ejemplo, un ciclo en cascada me dice que debo planificar todo de entrada, uno iterativo permite que planifique la idea global y en detalle la primera iteración. El ciclo de vida se elige al principio porque determina como voy a planificar y determinar el proceso.
- **Riesgos:** Los riesgos son la probabilidad de que ocurra algo que complique el éxito del problema. Tiene asociado 2 cosas: Probabilidad (si no la tiene, es un problema, no un riesgo) e Impacto, es decir, cuánto daño me hará si se presenta. Los riesgos con más impacto son, en orden descendente de importancia: el cambio de requerimientos, las estimaciones incorrectas, la gente, y la tecnología.
  Por último, los riesgos se miden con una variable llamada exposición que se calcula multiplicando la probabilidad por el impacto, y es evidente que no podemos determinar todos los riesgos (E = P (Riesgo) * I (Riesgo).
  Los peores riesgos son aquellos que tienen la exposición más alta, se los prioriza ya que son muchos y no se puede ocupar de todos, solo se ocupa y se tienen en cuenta en la planificación los mas importantes.
- **Estimaciones:** Se desarrolla debajo.
- **Calendarización o programación del proyecto:** Es el compromiso, existe en el modo tradicional únicamente. Se establece la lista de tareas, la asignación de personas y una fecha.
- **Definir cómo controlar el proyecto/Monitoreo y Control**
- **Definir cómo medir / Métricas:** No se usa para medir a la gente ni castigar. Nos dan visibilidad desde una visión cuantitativa, permitiendo objetividad. Podemos determinar: Tamaño (producto), esfuerzo (proyecto), tiempo (proyecto), defectos (producto); cuando las juntamos obtenemos métricas de proceso.

Propuesta de servicios: Es el presupuesto, lo que ofrezco. Tiene que contestar las mismas preguntas que el plan (más o menos). Para hacer esto hay que estimar. Incluye la garantía para defectos (6 meses a 1 año), y habla de experiencias (para que elijan).

Métricas:

¿Qué es una métrica?
Es un valor cuantitativo que me da información valiosa para hacer comparaciones y determinar el estado de un proyecto. Una de las cosas que se le puede definir a una métrica es el valor esperado de la misma, que da lugar a los indicadores. Algunos conceptos relacionados:

- Medida: Proporciona una indicación cuantitativa de la cantidad, dimensiones o tamaño de algunos atributos de un producto o proceso.
- Medición: Acto de determinar una medida.

- Indicador: Es una métrica o conjunto de métricas, que proporcionan una visión profunda del proceso de SW, del proyecto de SW o del producto en sí. Los indicadores comparan métricas y me dan una visión del cumplimiento del objetivo.

Clasificación de las Métricas:

- Directas: Se obtienen sin depender de ninguna otra métrica, y cuya forma de medir es un método de medición simple, como un conteo o una suma.
- Indirectas: Se obtienen a través de una función de cálculo, es decir, es decir, las mediciones de dicha métrica utilizan las medidas obtenidas en mediciones de otras métricas, directas o indirectas.

Escalas: Conjunto de valores con propiedades definidas [ISO 14598-1]

- Ordinales: Asumen un orden o ranking. No se asume intervalo equidistante.
- Intervalo: Designa un ordenamiento equidistante (la distancia entre los ítems tiene un significado).
- De ratio: Designa intervalos equivalentes con un punto de cero absolutos.
- Nominal: Designa categorías o atributos.

Confiabilidad y Validez: Las métricas deben ser confiables y validas, esto quiere decir que se encuentren dentro de valores determinados como correctos.
Confiabilidad: Si la misma métrica realizada sobre el mismo concepto varias veces, me da valores diferentes, la métrica es válida pero no confiable.
Validez: Si el resultado de la métrica devuelve un valor fuera de los intervalos designados como correctos, se dice que ésta es no válida.

Dominio de las métricas:

Si lo que se mide se puede medir trascendiendo el proyecto, se trata de una métrica de proceso o producto. Si no se puede, la métrica es de proyecto.

- Métricas de Producto: Miden los atributos del producto.
- Métricas de Proceso: Son métricas organizacionales y públicas, es decir que las puede ver cualquier miembro de la organización a diferencia de las de proyecto, las cuales solo las pueden ver los miembros del equipo. En éstas, no se muestra la información de un proyecto, sino un promedio de los proyectos.
- Métricas de Proyecto: Métricas privadas para un equipo, que muestran información relativa al proyecto. Una métrica de proyecto se puede transformar en una de proceso estableciendo intervalos de tiempo y determinando promedios. No todas las métricas de proyecto convienen ser llevadas a métricas organizacionales.

“Se busca identificar métricas que sirvan para la gestión del proyecto y no para el testing”.

Métricas Básicas: Según el enfoque tradicional estás métricas deben ser definidas si o sí.

- Tamaño (de Producto).
- Esfuerzo (de Proyecto).
- Tiempo calendario: La cantidad de días que me llevó realmente, no la que estimé. (de Proyecto).
- Defectos (de Producto).

Si bien el avance no es una métrica básica, es una métrica fundamental para el proyecto.

¿Quién las hace?
La mayoría de las métricas vinculadas al proyecto son tomadas por el Líder de Proyecto, pero todo el equipo puede tomar distintas métricas. El que toma las métricas es quién tiene más a mano la información necesaria para hacerlo, pero éste no siempre es el mismo a quien necesita la métrica.

¿En qué situaciones concretas se hace el seguimiento de un proyecto?
El principal responsable en un equipo de proyecto es el Líder de proyecto. Para determinar el seguimiento utiliza información que le da el resto del equipo, esta información es utilizada para determinar métricas y se obtiene a través de reuniones de equipo que puede ser de diferentes tipos:

- Periódicas: El período entre las reuniones es el establecido en el Plan de Proyecto, en la sección de controles.
- Frente a hitos: Un hito es un acontecimiento importante que se da en un momento puntual de tiempo.

¿Por qué son importantes?
Las métricas no resuelven problemas, las personas los resuelven. Las métricas proveen información para que las personas puedan tomar mejores decisiones. Las métricas dan visibilidad a los problemas.
Las métricas nos ayudan a obtener datos para planificar y estimar. Nos ayudan también al seguimiento del proyecto comparando la información de las métricas con lo que habíamos planificado. Nos proporcionan datos concretos de diferentes situaciones.
Sirven para predecir, estimar y prevenir riesgos y actuar en función de sus valores.

¿Cómo elegimos que métricas tomar?
Debemos considerar que la métrica sea útil para la gestión del proyecto, que sean factibles de ser tomadas. Tienen que ser eficientes y tomadas en el menor tiempo posible y que den la mayor cantidad de información posible y debe tratarse de que puedan ser automatizadas.
En el enfoque tradicional todo el proceso de métricas se planifica, tiene asociado un tiempo, recursos, etc. Pero siempre tratando de que no se lleve mucho tiempo y esfuerzo en el proceso de métricas.
GQM (General Question Metric): Este enfoque plantea que las métricas son útiles en la medida que estén vinculadas con un objetivo claro.

No debemos hacer:

- No debemos medir las personas, porque la métrica pasa a depender de lo que la persona informa lo cual puede no ser cierto y la métrica deja de ser verídica.
- Nunca alcanza con una sola métrica.
- No ignorar los datos obtenidos de las métricas.
- No usar métricas como castigos.

Estimaciones:

Las estimaciones son valores numéricos, cuantitativos que nos ayudan a predecir, se realizan al comenzar el proyecto y puede existir incluso una Pre-estimación antes de arrancarlo.

No es lo mismo que planificar ya que el plan es un compromiso que asumo, y para asumirlo tengo que estimar. Hay una probabilidad involucrada porque tengo que trabajar con la INCERTIDUMBRE (falta de información), es decir NO es precisa.

¿Por qué estimamos?: Estimamos para predecir completitud y para administrar los riesgos.

Si se dan cambios en los requerimientos reestimo, cualquier factor que cambie algo debe obligarme a reestimar. Puedo usar como variable de ajuste al Alcance, especialmente cuando no puedo modificar la fecha de finalización.

¿Cuándo se estima? La pre estimación se realiza cuando se hace la propuesta de servicio, y luego se vuelve a estimar cuando se realiza el plan. Cuando empieza el proyecto también se vuelve a estimar, ante cambios en los requerimientos, cambios en el personal, es decir cuando aparece un factor que "cambia las reglas del juego" se tiene nueva información por la tonto se reestima.

¿Quién estima?: El encargado de estimar suele ser el líder, aunque puede hacerlo también el gerente. Dentro del proyecto si es un líder autoritario realiza solo las estimaciones, si es más democrático las realiza con colaboración del equipo.

¿Qué estimamos?:

1. **Tamaño:** Es una estimación del producto que apunta a contar los CU por complejidad (aunque pueden existir otros criterios). Es lo primero que estimo.
2. **Esfuerzo:** Se mide en horas persona lineal, son una bolsa de horas necesarias de forma lineal (asumiendo que va a trabajar una sola persona a la vez).
3. **Tiempo**: Se estima la gente disponible, horas por día de trabajo, días de la semana, etc. NUNCA estimamos el tiempo de programación, ya que estoy dejando afuera al testeo, relevamiento, etc.
4. **Costo:** Se estima al último porque depende de todas las estimaciones anteriores.
5. **Recursos críticos:** Como ancho de banda necesario, licencias de SW, etc.

<u>Técnicas para estimar:</u>

- **Basado en la experiencia:** Pueden ser individuales o grupales.
  - **Datos Históricos:** se compara un nuevo proyecto con uno pasado, se utilizan para evitar las discusiones entre desarrolladores y el cliente, se recolectan datos de tamaño (LOC, PF, CU), esfuerzo, tiempo, defectos y luego se calibran (convierte las cuentas a estimados, x LOC por mes, LOC a esfuerzo), solo se puede usar en organizaciones que tengan la cultura de guardar información.
  - **Juicio experto:** es el más usado, el 75%, usa la fórmula de (Optimismo + 4 Habitual + Pesimismo) /6
    - **Puro:** un experto estudia las especificaciones y hace su estimación, se basa en los conocimientos del experto, quien se supone que tiene el conocimiento para estimar. Tiene dos problemas esta técnica: la dependencia de la organización en una sola persona, por lo que se trata que el conocimiento sea organizacional, no individual; el segundo problema es el de que el experto tiene que ser experto en todo ya que el software tiene muchas variables, el cliente, tecnología, etc. Lo cual es muy difícil de encontrar.
    - **Delphi:** Se junta gente experta en distintas áreas. Se les dan las especificaciones a un grupo de personas que intentan estimar lo que

costara el desarrollo tanto en esfuerzo como en duración, las grupales suelen ser mejores que las individuales. Luego le remiten sus estimaciones individuales al coordinador, se reparten las estimaciones anónimamente, se discute, se revisa sus propias, se envía al coordinador nuevamente. Se repite el proceso hasta que la estimación **converge de forma razonable.**

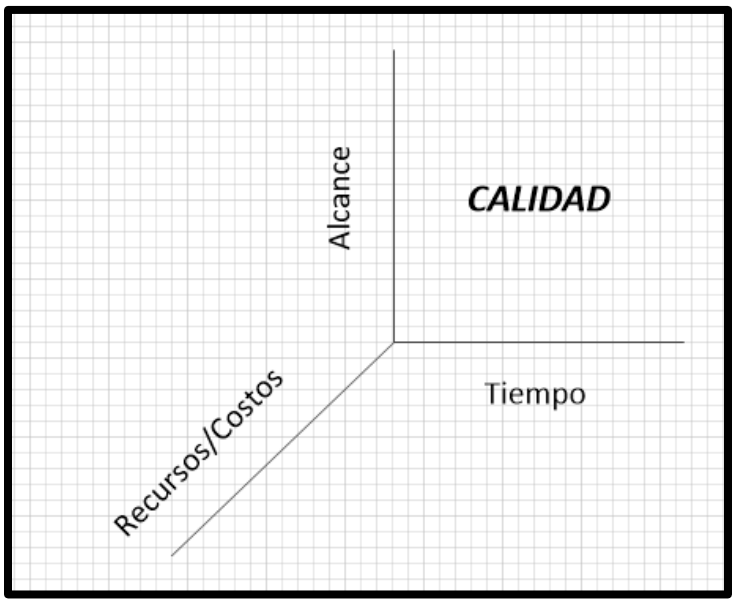
    - **Analogía:** consiste en comparar las especificaciones de un proyecto, con las de otros proyectos, se compara usuarios, complejidad, tamaño, tecnologías, etc.
- **Basado en el mercado:** Se estima en función de lo que se cree que el cliente puede pagar. Es una decisión comercial, de negocio, no tiene nada que ver con lo que vale realmente el software.
- **Basado en los componentes:** Se divide en dos: **De Proceso:** Descompongo en las tareas a hacer, calculo cuánto tiempo necesito para cada tarea luego se suma y se obtiene el esfuerzo y en consecuencia el costo.
  De Producto: Se desagrega en CU módulos o subsistemas, se calcula el tiempo que se necesita por cada componente de software, se suma el de todos los componentes y se calcula el costo.
  Se puede dar el uso de la técnica mediante la mezcla de ambos.
- **Basado en los recursos:** Analizo cuántos recursos dispongo y distribuyo respecto a ello. Se utiliza normalmente en el gobierno.
- **Basado en algoritmos:** Utiliza fórmulas de cálculo, por lo que automatizo bastante. No garantiza, cómo ninguna técnica lo hace, la estimación.

Triple Restricción:

La triple restricción es un concepto que consiste en el equilibrio que se debe mantener para lograr el éxito del proyecto. Se puede fijar como máximo dos de estas restricciones, pero debe quedar como mínimo una libre que se pueda ajustar para mantener el equilibrio en el proyecto. La variable de ajuste es la CALIDAD, conviene negociar el alcance (menos cosas, pero bien hechas) y el tiempo (aumentarlo), los recursos son más o menos fijos.

Las tres restricciones están vinculadas y condicionadas entre sí.

Tratando con procesos definidos, puedo trabajar con mi cliente en la definición del alcance y luego queda fijo, entonces parto con los alcances fijos y de ellos derivo el tiempo y los recursos.



- NO puedo determinar de antemano ninguno de ellos.

Modelado Ágil

Es un conjunto de principios y prácticas para modelado y análisis de requerimientos, que complementa a la mayoría de los métodos de desarrollo iterativos e incrementales (los métodos agiles son un subconjunto de los modelos iterativos)

Manifiesto Ágil

Fundaciones sobre las que hoy en día se basan las metodologías agiles, los métodos con pensamiento ágil deben tener los siguientes valores:

- **Individuos e interacciones por sobre procesos y herramientas** (los roles son intercambiables, se centra en los individuos no los roles)
- **Software funcionando por sobre documentación detallada** (se documenta aquello que agregue valor al producto, lo útil se documenta, si una documentación para el proyecto tiene valor agregado se debe hacer en forma creciente)
- **Colaboración por sobre negociación con el cliente** (el cliente se vuelve alguien importante en el proyecto, no se negocia, se colabora en forma conjunta y así surgen los requerimientos emergentes)
- **Responder a cambios por sobre seguir un plan**

Existen 12 principios:

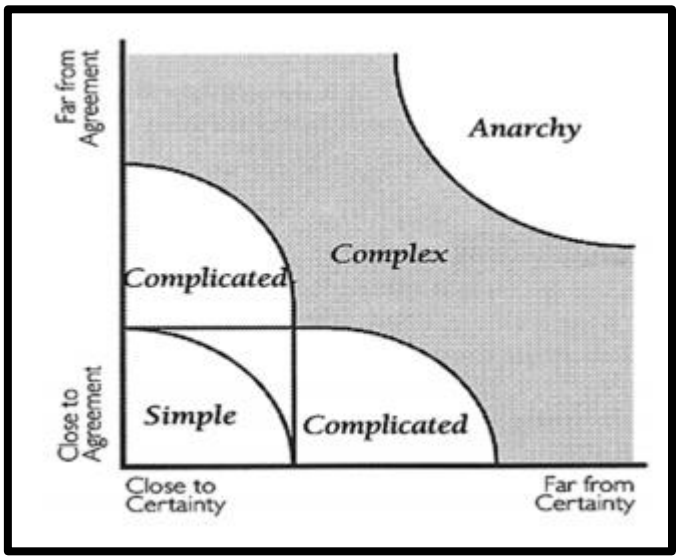
1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos SW funcional frecuentemente, entre dos semanas y dos meses, con preferencia al período de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El SW funcionando es la medida principal del progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para, a continuación, ajustar y perfeccionar su comportamiento en consecuencia.

¿Qué significa ser ágil?

- Balance entre ningún proceso y demasiado proceso. La diferencia inmediata es la exigencia de una menor cantidad de documentación**, pero sigue existiendo**.
- Los métodos ágiles son adaptables en lugar de predictivos.
- Son orientados a la gente en lugar de al proceso.

¿Cuándo lo usamos?



Podemos determinar como el eje de las ordenadas a la Tecnología, y el de las abscisas a los Requerimientos. Entonces si nuestro problema cae en la región de “Complex” es susceptible de resolverse por Agile. En “Anarchy” normalmente está la investigación, y el Mantenimiento en “Simple” (No siempre, pero la mayoría de los casos, es lo deseable).

User Stories:

El principio asociado con los requerimientos ágiles es el principio número 6: “El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara”.

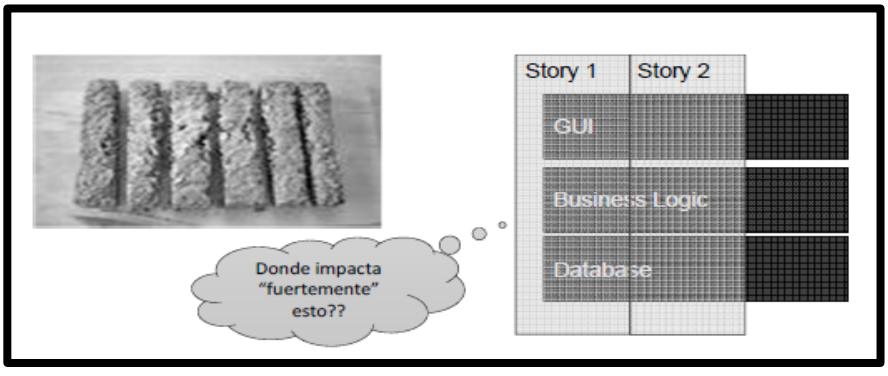
Son técnicas de requerimientos ágiles. Los requerimientos ágiles se basan en que no es necesario perder tiempo documentando porque la forma más eficiente de comunicación es cara a cara, no se necesita en detalle especificado un documento escrito y firmado, porque se tiene al cliente en las conversaciones cara a cara. Este enfoque combate la presión de la necesidad de encontrar todos los requerimientos juntos al principio, los agilistas dicen que esto no es necesario ya que los requerimientos van cambiando constantemente. No se hacen especificaciones de antemano completas, se hacen especificaciones HOC que tiene que ver con especificar solamente lo que voy a consumir ahora,

Podemos decir que son multipropósito ya que:

- Son una necesidad del usuario
- Son una descripción corta del producto
- Son un ítem de planificación.
- Son un token para una conversación.
- Son un mecanismo para diferir una conversación.

Existen dos tipos de requerimientos: Requerimientos de usuario y Requerimientos de SW.
Las User Stories se encuentran dentro de los Requerimientos de usuario, por lo que el cliente sigue teniendo la posibilidad de involucrarse y comprometerse, mientras que los cu se encuentran en los requerimientos de sw.
La forma de expresarnos con las Stories es con “Como **<<quién realiza la acción>>** yo quiero **<< qué acción que realizara el sistema>>** de forma tal que **<<para qué es necesaria la actividad (valor de negocio)>>**”

- Cortan verticalmente a la arquitectura de SW.



Product Backlog:

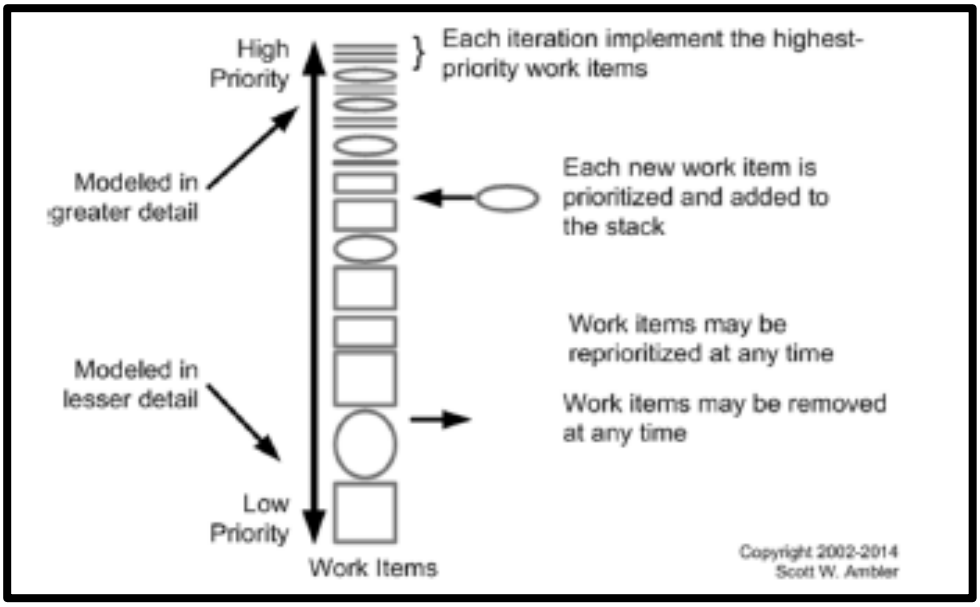
El encargado de administrar las User Stories es el Product Owner, el representante del negocio quien define hacia dónde va a ir el producto, y que va a tener el mismo. El mismo las maneja en un Product Backlog, que es un contenedor de las cuestiones que se espera que el producto posea en un momento de tiempo determinado (expresado en Stories) y se encuentra en forma de **PILA PRIORIZADA** (con las Stories más importantes en la cima, y con la menor granularidad de las mismas en la cima, las épicas y temas se encuentran en el fondo).

Nunca tiene la totalidad de las características del producto como si se espera que tenga la ERS, además la ERS contiene requerimientos de SW.

Granularidad de las User Stories:

Según cuánto abarque nuestra Story tendremos:

- User Story: Tiene tamaño manipulable en el contexto de una sola iteración.
- Épica: User Story grande que se puede descomponer.
- Tema: Agrupador de épicas o de users.



Del Backlog priorizado se extraen las User Stories más importantes según lo que el Product Owner considere (ya que él es el que la administra), y se las envía a la iteración a través del criterio de READY que debe cumplir al menos con el modelo de INVEST, luego el equipo puede determinar otros criterios para decir que cumplen con READY, una vez elegidas las users van a formar parte de la iteración y no se pueden cambiar hasta que finalice la misma.

- **I**ndependent**:** Calendarizables e implementables en cualquier orden.
- **N**egotiable**:** Es el contexto de una User, es decir, qué voy a permitir que el cliente me defina el qué quiere, pero no el cómo.
- **V**aluable**:** Debe tener valor para el cliente. Es el para qué que vemos en la User. En base a esto se prioriza.

- **E**stimable**: Se** debe poder estimar la User, su tamaño. Si no se puede estimar se convierte en un SPIKE (una incertidumbre que tiene tal nivel de cosas que no conocemos, que no podemos estimarla).
- **S**mall**:** Deben ser consumidas en una iteración, por lo tanto, deben ser lo suficientemente pequeñas para que se pueda terminar en una iteración, cuando decimos terminar, nos referimos al criterio de DONE, el cual responde a la pregunta de cuando un trabajo está terminado, para esto existe un Check List que nos ayuda a definir el criterio de DONE, además el equipo puede definir otros aspectos a tener en cuenta en el criterio. Cabe aclarar que existe un criterio de DONE para cada User, y un criterio de DONE para el producto entero, y finalmente existe un criterio de DONDE por encima del de producto, que es el DONE para la Release.
- **T**esteable**:** Deben poder probarse, implementarse.

Confirmación:

Es el conjunto de pruebas de aceptación, se escribe detrás de la carta y lo hace el Product Owner ya que es el que vive la situación de negocio. Debemos elegir la menor cantidad de pruebas, pero que éstas me cubran la mayor cantidad de casos en el negocio.

La confirmación es la identificación de las pruebas de aceptación. Mientras que los criterios de aceptación nos ayudan a escribir pruebas de aceptación. Estos criterios se escriben en la sección de "Notas" en las cards. Las pruebas son en sí las reglas de negocio. La tendencia "agilista" es realizar las pruebas lo antes posible.

Estimaciones:

Las estimaciones se colocan en el User Story a través de un Story Point, en el extremo superior derecho de las Card, el Story Point es una unidad de medición de las User Stories que me permite estimarlas, mide la complejidad, el esfuerzo y la incertidumbre asociada a esa User y esto conforma una especie de tamaño asociado al User Point. Hay una tendencia a estimar inicialmente las Users en el proceso de priorización del Product Backlog, y revisar ésta cuando se empieza a trabajar en la iteración.

Requerimientos no funcionales:

Las User Stories apuntan a las características funcionales del SW.
Los requerimientos no funcionales no se pueden implementar sin el contexto de un requerimiento funcional.
Se dice que los requerimientos no funcionales son restricciones, no Users, donde una restricción puede afectar a una sola User o a todas. Se las escribe en Cards de restricción y que se sabe tienen peso 0 (cero).
También se los llamó Tech Stories, pero éstas son peligrosas ya que luego el que prioriza en el Product Backlog es el Product Owner, quién no entiende y no sabe cómo priorizar las mismas.
Una forma de manejar los requerimientos no funcionales es integrarlos a lo funcional, esto se puede hacer cuando los mismos están asociados a una sola User en particular, escribiéndolos junto con los Criterios de Aceptación (en la sección "notas"); y cuando están asociados a todas las Users, se los incluye en cada una de las Users a las que afecta.

Spikes

Son un tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una User Story u otra faceta del proyecto.
Surgen cuando se aplica el INVEST Model a una User y generalmente cuando se llega a la parte de Estimable. Al no poder estimar, no se puede saber si es pequeña o no, si es testeable o no, por lo que deja de cumplir con el criterio INVEST. Por lo que dejan de ser Users para convertirse en SPIKES.

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 resolverla interacción con el usuario para alcanzar el beneficio esperado.

Se clasifican en:

Técnicas:

- Son las incertidumbres relacionadas a la tecnología.
- 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.

Funcionales:

- Incertidumbres acerca del negocio.
- Utilizadas 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.

Algunas User Stories requieren de ambos tipos de Spike.

Lineamientos para Spikes:

- Estimables, demostrables, y aceptables.
- La excepción, no la regla.
- Toda historia tiene incertidumbre y riesgos.
- 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 el spike sea pequeño y sencillo y sea probable encontrar una solución rápida en cuyo caso, spike e historia pueden incluirse en la misma iteración.

SCRUM:

Es un framework o marco de trabajo para hacer gestión de proyectos ágil. Propone una forma de cumplir con los principios del manifiesto. Es un conjunto de lineamientos, buenas prácticas vinculadas a la Gestión de Proyectos Ágil. Hace foco en cómo gestionar el proyecto, asume que las personas involucradas saben construir el producto.

Cimientos:

- Empirismo: Se basa en la teoría de control de procesos empíricos, asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce.
- Auto-organización: Los equipos son auto-organizables, no hay un jefe y cada uno asume la responsabilidad de hacer un trabajo y sabe cómo hacerlo.
- Time-boxing: Las actividades que SCRUM tiene que ejecutar (llamadas ceremonias) dentro de las cuales la más importante es la iteración llamada SPRINT, que es un período fijo de tiempo en el cual tenemos que generar un producto entregable al cliente y funcionando. La recomendación es que dure hasta 30 días, pero los tiempos pueden fijarse de ante mano y no se pueden modificar.
- Priorización: Es importante y el cliente debe hacerlos para que se comprometa y haga cargo de lo que pidió.
- Colaboración: El equipo debe colaborar.

Equipo SCRUM/Roles:

Está formado por 3 roles, y todos están involucrados:

- **Dueño de producto o Product Owner:** Es el responsable de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. Es la única persona responsable de gestionar la Lista del Producto (Product Backlog). Siempre se encuentra del lado del negocio y maneja la economía y el presupuesto. No es un perfil técnico.
- **Scrum Master:** es parte del scrum teames el encargado de realizar funciones de coordinación, es **un líder de servicio**, funciona como mediador entre el Product Owner y el Equipo. A su vez también es el nexo entre el equipo y los propios jefes de la organización. Además, también realiza tareas vinculadas a la construcción el producto, controla que las ceremonias de Scrum se respeten.
- **Scrum Team:** Son profesionales que realizan el trabajo de entregar un Incremento de producto “Done” que potencialmente se pueda poner en producción al final de cada Sprint. Son auto-organizados ya que nadie les indica cómo convertir User Stories en Incrementos. Posee de 5 a 9 miembros, teniendo preferencia por el número 7.

Eventos/Ceremonias de Scrum:

Existe una ceremonia denominada Story Time que es en donde se genera y mantiene al Product Backlog, Scrum no la considera dentro de sus ceremonias ya que asume que el Product Backlog ya existe.

Todos los eventos son bloques de tiempo (time-boxes), de modo tal que todos tienen una duración máxima. Una vez que comienza un Sprint, su duración es fija. Las ceremonias son:

1. Sprint Planning: Durante el mismo se planifica el trabajo a realizar. Se crea mediante el trabajo colaborativo de Equipo Scrum completo. Se manifiesta a través del artefacto SPRING BACKLOG, ya que toma la porción del Product Backlog que se decidió que entraba. Debo determinar y estimar la capacidad del equipo para poder saber qué es lo que entrará del PB al Spring, para ello puedo medir en horas ideales (el tiempo que realmente se trabaja) o Story Points. La ejecución del SPRINT genera un incremento del producto/Product Increment (porción de SW entregable).
2. Daily Meeting: Son reuniones obligatorias diarias organizadas por el SCRUM MASTER cuyo propósito es que el equipo se sincronice. Duran entre 15 y 30 minutos (3 minutos por persona) y cada miembro debe contestar a 3 preguntas que sirven para dar visibilidad y exponer los compromisos:
    a. ¿Qué hiciste?
    b. ¿Qué vas a hacer?
    c. ¿Tuviste algún problema?
3. Spring Review: Mostramos el producto, lo que alcanzamos a hacer. El Owner decide si lo acepta o no. Se calcula la velocidad del equipo en Story Points sumando los SP de cada User **aceptada**. El foco se hace en el producto y la mejor métrica del progreso es el SW funcionando. Las user "casi" terminadas no cuenta, vuelven al Product Backlog con la prioridad que el Product Owner defina.
4. Sprint Retrospective: El foco es mejorar el proceso, por lo que se toma un tiempo para reflexionar, realizando una inspección y adaptación. Es recomendado que la coordine otra persona que no sea el Scrum Master, y que esta persona tenga experiencia, ya que servirá como mediador. No puede durar menos de 2 horas. La técnica básica utilizada es START (Las cosas que acordamos como equipo que hay que comenzar a hacer), STOP (Acuerdos para determinar prácticas que no se realizarán o seguirán realizando, un ejemplo es llegar tarde), y CONTINUE (Lo que debemos seguir haciendo).

✓ El Scrum Master está en todas las ceremonias, el Product Owner solo en review y planning (aunque se podría invitar a las otras).

Artefactos de SCRUM

- **Product Backlog:** es una lista priorizada del producto, dicha priorización es realizada por el Product Owner. Contiene requerimientos de usuario en distintos niveles de granularidad, al tope del Product Backlog se encuentran las User de mayor prioridad para el usuario.
- **Sprint Backlog:** este artefacto es la salida del Sprint Planning, contiene una porción del Produt Backlog que se va a llevar a cabo en la iteración, para determinar qué tanto se va a realizar en el Sprint se debe tener en cuenta la capacidad del equipo (medida en horas ideales o story points).
- **Product Increment:** Hace referencia al incremento en el producto conseguido al final de cada iteración, software funcionando.

Tablero Scrum:

User	To do	Doing	Done	Done Done

El formato es como el visto en la tabla anterior, las User se suelen descomponer en tareas para enviarlas a la iteración, en el “To Do” se colocan las tareas que se harán durante la misma, en el “Doing” las tareas que se están realizando, en el “Done” las realizadas y en el “Done Done” las ya aceptadas por el Product Owner. Siempre recordemos que lo importante es el SW funcionando, así que tener las tareas solas no sirve de mucho, hay que saber elegir qué ingresará en la iteración.

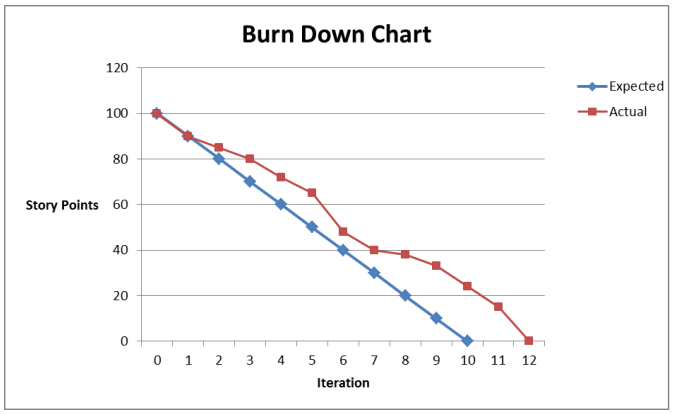
- El tablero se coloca en un lugar visible para todos.

<u>Charts de Micromanaging (dentro de Daily Meetings):</u>

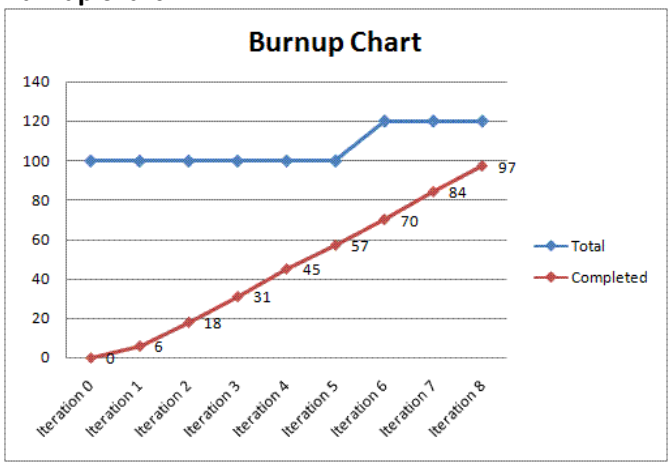
Tenemos distintos tipos:

- **Burndown chart:** Presenta una curva ideal (expected) la cual es imposible de lograr pero nos sirve para comparar con lo que estamos haciendo. Nos informa cuántos story points vamos “quemando” a lo largo de la iteración, por cada día (daily meeting). SI las User son muy grandes, parecerá una escalera donde los escalones son bastante grandes, si no los mismos serán más pequeños.
- **Burndown chart:** Es un gráfico propio del PRODUCTO, muestra los SP acumulados en el tiempo.
- **Gráfico de velocidad:** Podemos observar la velocidad con la que trabaja el equipo. En el eje de las ordenadas tenemos los SP, y en el de las abscisas la velocidad por cada sprint.

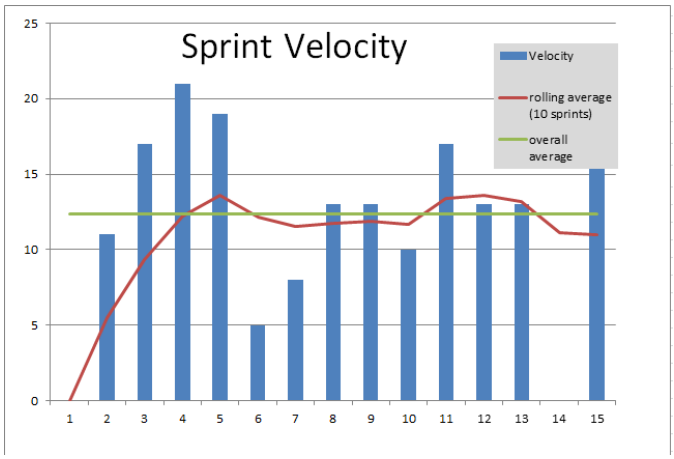
- **Burndown chart:**



- **Burnup chart:**



- **Gráfico de velocidad:**



<u>Métricas Ágiles:</u>

- Capacidad: Se estima, mide cuánto trabajo puede hacer el equipo. Se mide en Story Points o en Horas Ideales.
- Velocidad: Es métrica pero NO se estima, se calcula al final de todo.
- RTF (Running Testing Features) o RTU (U de Users): No es muy utilizado, usa las características hechas o features.