**Algunos conceptos importantes**

* **Medida**: proporciona una indicación cuantitativa de cantidad, dimensiones, capacidad y tamaño de algunos atributos.
* **Medición**: es el acto de determinar una medida. Me proporciona un mecanismo para lograr una evaluación objetiva.
* **Métrica**: Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Nosotros medimos por varias razones, entre ellas para señalar, asi controlamos y supervisamos el desarrollo de un proyecto. Para predecir, estimando proyectos de software, y para evaluar los mismos. También para mejorar.

Ninguna métrica mide personas, son para evaluar proyectos, productos u organización. Las métricas no deben ser utilizadas para retribuir o para castigar a la gente.

**Métricas de software en los diferentes enfoques de gestión**

**Métricas de software en el enfoque tradicional**

El enfoque tradicional es el primero de los 3 enfoques (ágil y Kanban/lean), y se basa en la gestión de proyectos basados en proceso definidos. Es más, completo y consecuente con el pensamiento tradicional

Las métricas son una de las disciplinas de la ingeniería de software, y dicha disciplina es transversal, como SCM, y se deben hacer todo el tiempo. Sin embargo, hay un momento inicial en los proyectos, donde los mismos se planifican y se definen ciertas bases, y entre las cosas que planificamos en ese momento son las métricas que vamos a usar, quién tiene responsabilidad sobre las mismas, cada cuánto se van a tomar, etc.

**Métricas de software**

Las métricas tienen distintos dominios, y este es entendido como el foco de medición, en dónde voy a hacer énfasis para tomar las medidas. Dicho foco puede ser en el proceso, en el proyecto y en el producto:

* **Métricas de proceso**: Son métricas que hacen foco en el proceso que utilizamos para construir el producto, el cual se adapta al proyecto. Son públicas para toda la organización
* **Métricas de proyecto**: tienen que ver con los recursos que se administran en el proyecto como esfuerzo, costo, tiempo, esas son las variables que se miden. Se miden estimaciones, es decir, lo que estimé y el valor real, la plata que estimé y la que necesité realmente, etc.
* **Métricas de producto**: el foco es lo que nosotros construimos, el sistema, la aplicación, el producto de software.

Esta división de dominios se realiza en los 3 enfoques. El enfoque tradicional hace énfasis en los tres dominios, el enfoque ágil y lean no tanto.

**Métricas básicas para un proyecto de software**

Son las mínimas que debemos empezar a medir, si tenemos intención de desarrollar una cultura de medición en la organización. Si no se toman métricas todo queda como una percepción subjetiva (Ej.: me parece que vamos bien, vamos al 33% del desarrollo -cuando esto es imposible de decir con certeza y no tienen ningún sentido-).

Al no medir no tenemos un enfoque cuantitativo de las cosas, por lo que no podemos hablar o tomar decisiones basadas en información, sino en percepción la cual es subjetiva. Las métricas básicas están vinculadas a desarrollar en una organización la capacidad de decisión. Es lo mínimo e imprescindible que se puede medir:

* **Tamaño del producto**: se puede asociar al alcance, implica que tan grande y complejo es el producto que se está desarrollando.
* **Esfuerzo**: se puede asociar a los recursos gastados. La influencia más grande en el esfuerzo necesario para un proyecto es su tamaño.
* **Tiempo (calendario)**
* **Defectos**

Estas métricas básicas tienen cierta relación a la triple restricción.

A la hora de definir las métricas se debe pensar qué le interesaría al desarrollador en particular, a la organización, o al equipo de desarrollo.

Los perfiles más técnicos (**desarrolladores**) apuntan a las métricas de producto, y a las métricas de esfuerzo, porque son compromisos que los developers asumen respecto al cumplimiento de un determinado trabajo. Es decir, les interesa saber métricas como: esfuerzo, esfuerzo y duración estimada y actual de una tarea, % de cobertura del testing unitario, número y tipo de defectos encontrados en el testing unitario.

En los ámbitos **organizacionales**, interesa el dinero, el calendario (el compromiso con el cliente se asume con un calendario), por lo que le interesa saber métricas como tiempo calendario, performance actual y planificación de esfuerzo, defectos en release, etc.

Y al **equipo de desarrollo**, le interesa una mezcla entre métricas de producto como de proyecto, tales como: tamaño del producto, duración estimada y actual entre hitos mas importantes, nro. de tareas planificadas y completadas, distribución de esfuerzo, etc.

**¿Debo medir todo?**

Las métricas significan costos, porque hay que definirlas, hay que tomarlas, hay que analizarlas y tomar decisiones en término de ellas. No siempre se puede llegar a tener un alto nivel de definición de métricas. Tampoco se puede medir todo, no se puede crear mil métricas, ya que implican costos y esfuerzo, por lo que es muy importante definir cuáles son aquellas métricas más importantes para el equipo, para la organización y para los desarrolladores

**Métricas y su simplicidad**

**Definición formal de métrica**: es el grado o la presencia de un determinado conjunto de atributos respecto del elemento que quiero medir. Tiene que ser un número y se debe poder medir (cuantificar).

Es decir, una métrica es un número, y se tiene que poder medir. Esto último es muy importante ya que, a veces se expresan métricas que no hay forma de medirlas, o que el costo de medición es muy alto y no conviene medirla. El enfoque de métricas hay que tenerlo lo más simple posible.

También hay que lograr un equilibrio entre el costo de tomar la métrica y el beneficio de hacerlo. Un equilibrio entre certeza y precisión, la precisión es cara y muchas veces es innecesario. Por ejemplo, no tiene sentido medir el tiempo que se dedicará a un proyecto grande en minutos o en segundos, esa precisión es cara y no nos aporta ningún beneficio.

**Métricas Indirectas**

No existe una forma directa de medir o cuantificar la calidad, pero es muy importante intentar medirla y para ello debo elegir otras cosas que me permitan medir la calidad, esas cosas suelen estar relacionadas con los requerimientos. Entonces, para poder medir la calidad voy a usar otras métricas.

La usabilidad tampoco se puede medir directamente, debo elegir otras cosas, como por ejemplo horas de capacitación para que el empelado lo sepa usar, cantidad de clicks que se deben hacer para realizar una función, entonces uso todas estas para obtener un número que me permita medir la usabilidad.

**Métricas de proceso**

Son métricas organizacionales, son públicas y su uso es **estratégico**. Se usan para mejorar los procesos de la organización a largo plazo. Los ingenieros de proceso son los responsables de tomar estas mediciones.

Para poder usar estas métricas, se debe hacer un proceso de despersonalización, es decir, tomo varios proyectos/productos y analizo sus métricas, para luego hacer un promedio de dichas métricas. El objetivo es hacer un trabajo de no vinculación a un producto o proyecto en particular, para poder hablar del comportamiento de la organización. Se junta información de muchos productos, porque no vas a cambiar el proceso porque 1 de 10 proyectos tuvieron problemas.

La idea es mejorar continuamente los procesos de una organización, aumentando su nivel de madurez. Se determinan unos atributos específicos a medir, se desarrollan métricas en torno a dichos atributos y luego se emplean las métricas para generar indicadores que conducirán a una estrategia de mejora.

Asumen que la experiencia del equipo es extrapolable a otro equipo, por ello son importantes las métricas de proceso.

Ejemplo: 4 errores cada 4 millones de líneas sin comentarios. Defectos detectados por usuarios, propagación de errores por fases.

Las métricas de proyecto y de producto son particulares, no generales, como las de proceso. Para armar el conjunto de métricas de la organización, necesitamos saber las necesidades de información de los involucrados, las expectativas que tienen, y en función de eso definimos el esquema de métricas.

**Métricas de proyecto**

Se utilizan con propósitos **tácticos**, es decir, buscan generar información que sirva como base a los líderes de proyecto y al equipo para adaptar el desarrollo de proyectos y de las actividades técnicas. El líder del proyecto es quien debe obtener las métricas del proyecto, él es el responsable del esfuerzo, del calendario, etc.

La idea detrás de estas métricas es mejorar la planificación del desarrollo, evaluar la calidad de los productos en todo momento, y, de ser necesario, modificar el enfoque para mejorar la calidad.

Ejemplos: fechas de entrega reales vs programadas, cantidad de cambios y sus características, esfuerzo por tarea.

**Métricas de producto**

Se utilizan con propósitos técnicos, y tienen como objetivo generar indicadores en tiempo re al de la eficacia del análisis, del diseño, de la estructura del código, la efectividad de los casos de prueba y de la calidad global del software a construir. Las métricas que tienen que ver con el producto, son responsabilidad de quienes hacen el trabajo (son quienes la obtienen, ejemplo defectos son tomados por los testers, )

Es decir, buscan controlar que los artefactos resultantes del proceso de desarrollo cumplan con los requerimientos del cliente, que estén libres de errores, que cumplan con los criterios de calidad existentes y que se hayan realizado bajo los procedimientos de calidad.

Las métricas de producto son:

* **Tamaño del producto**: se puede asociar al alcance, implica que tan grande y complejo es el producto que se está desarrollando. En un inicio, el tamaño del producto se medía en las líneas de código sin comentar, pero es una métrica poco aceptada ya que depende mucho del lenguaje de programación, como de la experiencia del programador. Luego se evolucionó, y el tamaño del producto se estimó en función de los requerimientos, y para medir el tamaño del producto en función de ellos requerimientos se miden casos de uso (CU por complejidad), features, clases y puntos de historia (US por complejidad).
* **Esfuerzo**: se puede asociar a los recursos gastados. La influencia más grande en el esfuerzo necesario para un proyecto es su tamaño.
* **Tiempo (calendario)**
* **Defectos:** Los defectos de un proyecto de software es una función del esfuerzo y del tamaño del proyecto. Algunas métricas pueden ser, cantidad de defectos por punto de función. Generalmente proyectos más grandes tienden a producir más defectos por la línea de código
  + **Cobertura:** nos da confianza para liberar el producto de SW. Hay que tratar de que sea la mayor posible. Me indica que tanto de nuestro código está cubierto por los tests. Algunas métricas pueden ser % de clases cubiertas, % de líneas de código cubiertas por test, etc. No indica si los test son buenos, solo si cierta parte del código estuvo corriendo durante una prueba.
  + **Defectos por severidad:** tanto los defectos como los riesgos se pueden analizar según su severidad. Cantidad de defectos según su severidad, o % de defectos de severidad X
  + **Densidad de defectos:** una de las posibles métricas es la densidad de defectos, la cual se puede medir en cantidad de errores por líneas de código (para analizar esta métrica es importante tener en cuenta el tamaño del proyecto). Cuantos defectos se encuentran por unidad de testeo.

**Métricas en ambientes ágiles**

Tienen una visión mas simplificada y minimalista. Pero en enfoques ágiles, la experiencia no se puede extrapolar, y en enfoques tradicionales dicen que sí. Esto hace que cambie el enfoque respecto a lo que es valioso medir. A esto se le suma el principio de que el software funcionando es la principal medida de progreso.

De los tres dominios que habla la gestión tradicional, en ambientes ágiles lo único que importa es medir el producto. Las métricas en los enfoques ágiles no se esconden, están visibles.

**Regla de Oro Ágil sobre Métricas:** En enfoques ágiles nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software funcionando, el cual es la principal medida de progreso. Por lo cual, no se debe tomar a las métricas como una actividad, sino, como una salida. Se debe medir lo que sea necesario y nada más, solo lo que agregue valor, tanto para el equipo como para el cliente.

**Velocity**

Gráfico, Gráfico de barras

Descripción generada automáticamenteUna de las métricas mas importantes es la velocidad, que es la cantidad de puntos de historia que se completaron en la iteración anterior y fueron aceptados por el PO. Es una observación empírica que mide producto, no se estima, se calcula. Se obtiene a hecho consumado, sumo los SP de las US que el PO me acepta. No cuentan las historias parcialmente terminadas.

No es comparable entre equipos ni entre proyectos, ya que la experiencia no es extrapolable. Es la principal medida de progreso del equipoi

Se miden con gráficos en barras, y sirve para medir la estabilidad y sostenibilidad del equipo. Es probable que el primer sprint sea 0, si el equipo se está conociendo. La idea es buscar una estabilización, que el gráfico sea uniforme. Estos gráficos son permanentes, mientras dura el proyecto.

**Capacidad**

Es una métrica de proyecto, ya que mido el compromiso de trabajo (esfuerzo) en un determinado sprint. Habla del equipo y se estima al inicio, es una de las actividades iniciales del sprint planning, es la cantidad de horas ideales a las que se puede comprometer el equipo durante el sprint. Me hace falta para poder armar el PB, saber a cuánto producto me puedo comprometer en el sprint. Hay que hacerlo siempre, en cada sprint, ya que puede variar de sprint a sprint.

Se puede medir en esfuerzo (horas hombre ideales) en el caso de equipos menos experimentados, o en puntos de historia si el equipo ya tiene experiencia trabajando juntos.

Se puede calcular como las horas de trabajo disponible por día por los días disponibles de la iteración.

**Running Tested Features (RTF)**

Gráfico, Gráfico de líneas

Descripción generada automáticamenteEs parecido a la velocidad, también mide producto. La traducción es características testeadas que están funcionando.

Cuantas piezas de producto fueron terminadas, testeadas y están en ejecución. Es una medida de cantidad de características testeadas y funcionando. Pero el problema de la cantidad es que no es muy representativa si no se tiene en cuenta la complejidad de las características. Se basa en el principio de software funcionando como medida de progreso.

También tienen que estar presentes las métricas de defectos, no la ponen explícitamente como una métrica en los enfoques ágiles, pero se deben tomar.

**Métricas en Kanban**

Como Kanban busca mejora de los procesos, las métricas que hay son de proceso. En Kanban puro todo es flujo continuo, no existen proyectos.

En Kanban yo mido el proceso, y para medirlo lo que hago es medir las características que pasan por el mismo. Es decir, mido el comportamiento del proceso en función del tiempo que se demoran las características en pasar por todo el proceso y llegar al cliente.

Las dos métricas más importantes de Kanban son el Lead Time y el Cycle Time

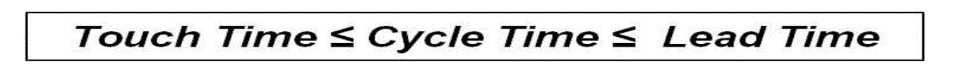
**Pantalla de video juego

Descripción generada automáticamente con confianza mediaLead time** es de vista o perspectiva del cliente, es la más importante ya que me permite saber la satisfacción del cliente, mide desde el momento en que el cliente me pide (punto de compromiso) algo hasta que se lo entrego (punto de entrega). Incluye el tiempo en el que una unidad de trabajo está en el backlog o en colas de acumulación. Se suele medir en días de trabajo y me determina el delivery rate o tasa de entrega.

**Cycle time:** saco la columna de backlog y comienzo a contar el tiempo desde que comienzo a accionar sobre el ítem de trabajo. Es la métrica que registra el tiempo que sucede entre el inicio y el final del proceso, para un item de trabajo dado (se suele medir en días de trabajo o esfuerzo). Es la medición más mecánica de la capacidad del proceso y me permite determinar la tasa de terminación

Me quedo solo con el tiempo de trabajo del ítem, es una vista interna por lo que le sirve al equipo de trabajo.

**Otras métricas de Kanban**

**Touch Time**: es más chico que el cycle time. Saca los tiempos de las columnas de acumulación. Solo mide las columnas en los que las personas están efectivamente haciendo algo. Es la suma de todos los tiempos durante los cuales se está trabajando activamente sobre un elemento de trabajo (excluyendo las colas de acumulación)

**Eficiencia del ciclo de proceso**, es el touch time dividido elapsed time (es lo mismo que lead time). Lo que hace para calcular la eficiencia es dividir el touch time sobre el tiempo transcurrido, mientras más cercano a 1 sea, mejor nos va, tenemos más eficiencia. Si yo lo toqué 10 horas y estuvo 1000 horas de lead time, no estoy siendo muy eficiente.

Tengo que medir defectos en todos lados, incluso en Lean.