UNIDAD N° 2
Métricas de Software

Métricas de Software

Definición de Métrica

La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo de software y sus productos para suministrar información relevante a tiempo, así el líder de proyecto junto con el empleo de estas técnicas mejorará el proceso y sus productos

Conceptos importantes

- **Medida:** proporciona una indicación cuantitativa de cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto.

- **Medición:** es el acto de determinar una medida. La medición permite obtener una visión del proceso y el proyecto dado que proporciona un mecanismo para lograr una evaluación objetiva.

- **Métrica:** una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

- **Indicador:** es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto o del producto en sí. El ingeniero en sistemas recopila medidas y desarrolla métricas para obtener indicadores.

¿Por qué medimos?

- **Señalar:** para controlar y así supervisar el desarrollo de un proyecto
- **Predecir:** al estimar proyectos de software
- **Evaluar** : para tener una noción de los costos
- **Mejorar**

Tipos de métricas

Métricas directas o primitivas

Son aquellas que para medir un atributo de una entidad sólo se precisa de dicho atributo, es decir aquellas métricas que podemos tomar directamente sin necesitar de otra métrica.

- Ejemplo: LDC en modulo, la fiebre
- No deberían ser usadas para medir la performance de un individuo

Métricas indirectas

Son aquellas que no miden un atributo directamente, sino que para medir un atributo de una entidad es necesario medir previamente más de un atributo de dicha entidad.

- Son funciones en donde los argumentos son otras métricas, directas o indirectas.
- Normalmente a este tipo de métricas son a las cuales queremos llegar.

Dominio de métricas

Métricas de proceso

Las métricas de proceso se utilizan con **propósitos estratégicos** y tienen como objetivo generar indicadores que permitan mejorar los procesos de software a largo plazo.

Características

- Se recopilan de todos los proyectos, durante períodos largos de tiempo
- Medir los procesos para conocer y mejorar costos, disminuir tiempos mientras se entregan productos con una calidad definida y las demandas de mantenimiento son apropiadas.
- Mejorar continuamente la performance. Para reducir los riesgos se necesita conocer las habilidades de los miembros del equipo de desarrollo y la experiencia del staff en general, y asegurarse que se cuenta con los recursos para competir en un determinado dominio.
- El proceso debería permitir introducir nuevas tecnologías y poder determinar su costo-beneficio.
- Los indicadores de proceso permiten tener una visión profunda de la eficacia de un proceso, determinando que funciona de acuerdo a lo esperado y que no.
- Proporcionan beneficios significativos a medida que la organización trabaja para mejorar su nivel global de madurez del proceso.

Estrategia

Una forma de mejorar cualquier proceso es medir sus atributos específicos, desarrollar un conjunto de métricas significativas con base a dichos atributos y luego emplear las métricas para generar indicadores que conducirán a una estrategia de mejora.

Ejemplos

- Errores previos a releases por proyecto
- Defectos detectados por usuarios
- Esfuerzo realizado
- Tiempos de planificación promedio por proyecto
- Propagación de errores de fase a fase.

Adicional

- Métricas privadas: datos privados para un individuo que ayudan al enfoque del proceso personal. **Ejemplo**: promedio de defectos por módulo o errores durante el desarrollo.
- Métricas públicas: son privadas entre equipos de proyecto, pero públicas para los miembros del equipo y tienen el objetivo de mejorar el rendimiento del equipo de trabajo. **Ejemplo:** defectos informados de funciones importantes que ha desarrollado el equipo, errores encontrados durante revisiones técnicas formales y líneas de código o puntos de función por módulo.

Métricas de proyecto

Las métricas de proyecto se utilizan con **propósitos tácticos**, es decir, tiene como objetivo generar información que sirva como base a líderes de proyectos y al equipo para adaptar el desarrollo de los proyectos y de las actividades técnicas.

Es necesario medir el proyecto dado que este debería ser entregado con:

- Las capacidades funcionales y no funcionales requeridas por el cliente/usuario.
- Las restricciones impuestas.
- El presupuesto y tiempo planificado
- Un producto que cumpla ciertas características de despliegue, operacionales y de mantenimiento.

Utilizando las métricas de proyecto

La primera aplicación de las métricas de proyecto ocurre durante la estimación, cuando las métricas recopiladas de proyectos anteriores se utilizan como una base desde la que se realizan las estimaciones de esfuerzo y calendario para el nuevo proyecto.

Finalidades principales

- Las métricas de proyecto se utilizan para mejorar la planificación del desarrollo, generando ajustes que eviten retrasos, reduzcan riesgos potenciales y por lo tanto problemas.

- Las métricas de proyecto se utilizan para evaluar la calidad de los productos en todo momento, y en base a esto, de ser necesario, modificar el enfoque para mejorar la calidad, minimizando defectos, retrabajo y por ende el costo total del proyecto.

- Las métricas del proyecto se consolidan con el fin de crear métricas del proceso que sean públicas para la organización de software como un todo.

Ejemplos

- Esfuerzo/Tiempo por tarea de Ing. De Software
- Errores no cubiertos por hora de revisión
- Fechas de entregas reales vs programadas
- Cambios (cantidad) y sus características

Métricas de producto

Las métricas de producto se utilizan con **propósitos técnicos**, tienen como objetivo generar indicadores en tiempo real de la eficacia del análisis, el diseño, la estructura del código, la efectividad de los casos de prueba y calidad global del software a construir.

Es decir, se deben controlar los artefactos resultantes del proceso de desarrollo (componentes y modelos) para garantizar:

- Que cumplan con los requerimientos del cliente
- Que estén libres de error
- Que cumplen con los criterios de calidad existentes
- Que se realizaron bajo los procedimientos de calidad

Ejemplo

Tamaño del sistema: LDC, CU por Complejidad, Cantidad de User Stories por Complejidad, PF, etc. Otros ejemplos son "cantidad de líneas de código del producto"; "cantidad de métodos de cada clase"; "promedio de métodos por clase".

Factores de calidad McCall:

- Corrección**:** hasta donde satisface un programa su especificación y logra los objetivos propuestos por el cliente.
- Fiabilidad: hasta donde se puede esperar que un programa lleve a cabo su función con la exactitud requerida.
- Eficiencia: cantidad de recursos informáticos y código necesarios para que un programa realice su función.
- Integridad: hasta donde se puede controlar el acceso al software o a los datos por personas no autorizadas.
- Usabilidad: el esfuerzo necesario para aprender a operar con el sistema.
- Facilidad de mantenimiento: el esfuerzo necesario para localizar y arreglar un error en un programa.
- Flexibilidad: el esfuerzo necesario para modificar un programa que ya está en funcionamiento.
- Facilidad de prueba: el esfuerzo necesario para probar un programa y asegurarse que realiza correctamente su función.
- Portabilidad: el esfuerzo necesario para transferir el programa de un entorno Hard/Soft a otro entorno diferente.
- Reusabilidad: hasta donde se puede volver a emplear un programa en otras aplicaciones, en relación al empaquetamiento y alcance de las funciones que realizan el programa.
- Interoperatividad: el esfuerzo necesario para acoplar un sistema con otro.

Métricas básicas

- Tamaño
- Esfuerzo
- Calendario
- Defectos

Mediciones en el software

Medidas directas del proceso

Incluyen costo, esfuerzo, líneas de código, velocidad de ejecución, tamaño de memoria y defectos informados en un período de tiempo establecido, es decir que son medidas fáciles de reunir.

Medidas indirectas

Incluyen funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento, etc.; las cuales son más difíciles de medir.

Dominio de métricas

- **Producto**: privadas para el individuo y pueden ser combinadas para formar métricas de proyecto.
- **Proyecto**: públicas para un equipo de software, estas se consolidan para formar métricas de proceso.
- **Proceso**: públicas para toda la organización.

Principio de medición

- **Formulación**: La derivación de medidas y métricas apropiadas para la representación del software que se considera.
- **Recolección**: El mecanismo con que se acumulan los datos necesarios para derivar las métricas formuladas.
- **Análisis**: El cálculo de las métricas y la aplicación de herramientas matemáticas.
- **Interpretación**: La evaluación de las métricas en un esfuerzo por conocer mejor la calidad de representación.
- **Retroalimentación**: Recomendaciones derivadas de la interpretación de las métricas del producto transmitidas al equipo del software.

Métricas orientadas al tamaño

Las métricas orientadas al tamaño provienen de la normalización de las medidas de calidad y/o productividad considerando el tamaño del software que se haya producido.

Posibles variables a medir

- Líneas de código: poco aceptadas dado que depende mucho del lenguaje de programación, como así también de la capacidad lógica del programador.
- CU por complejidad
- User Stories por Complejidad
- Cantidad de Puntos de Función
- Nº de páginas web
- Errores, defectos, personas.

Ejemplos de métricas

- Errores por KLDC (miles de líneas de código)
- Defectos por KLDC, Esfuerzo por KLDC
- Páginas de documentación por KLDC.
- Errores por persona/mes
- LDC por persona/mes
- Costo por página de documentación.

Métricas orientadas a la función

Ya que la funcionalidad no se puede medir directamente, se debe derivar a partir de otras medidas directas, también conocidas como "puntos de función". Las métricas de software orientadas a la función emplean como valor de normalización una medida de la funcionalidad que entrega la aplicación.

Los **puntos de función** se definen a través de una tabla donde se indican algunas variables multiplicado por su complejidad y se suman los resultados, una vez concluido esto, se suman y multiplican algunos coeficientes resultantes de una serie de preguntas.

Variables para calcular puntos de función

- Número de entradas de usuario (pe: peticiones).
- Número de salidas por usuario (pe: informes, pantallas, mensajes de error, etc.).
- Número de archivos.
- Número de interfaces externas (pe: archivos de datos de cinta o disco).

El PF al igual que LDC es controversial. Los partidarios afirman que el PF es independiente del lenguaje de programación, y que se basa en datos que son más probable que se conozcan temprano en la evolución de un proyecto, lo que hace al PF más atractivo como enfoque de estimación.

Los detractores afirman que el método requiere cierta prestidigitación en cuanto a que el cálculo se basa en datos subjetivos más que objetivos y que el PF no tiene significado físico directo, es solo un número.

Métricas de Calidad

La calidad de un sistema, aplicación o producto es tan bueno como:

- Los requisitos que describen el problema,
- El diseño que modela la solución,
- El código que conduce a un programa ejecutable,
- Las pruebas que ejercitan el software para detectar errores.

Aunque existen muchas medidas de la calidad del software, la corrección, la facilidad de mantenimiento, la integridad y la facilidad de uso ofrecen indicadores útiles para el equipo del proyecto.

Medidas de calidad

Corrección

Grado en el que el software lleva a cabo su función requerida. La medida más común de corrección es "defectos por KLDC", donde defecto se define como la falta verificada de conformidad de requisitos.

Facilidad de mantenimiento

Es la facilidad con la que se puede corregir un programa si se encuentra un error, se puede adaptar si su entorno cambia o mejorar si el cliente desea un cambio de requisitos, y es necesario utilizar métricas indirectas para definirlo, como ser "tiempo medio de cambio".

Integridad

Mide la capacidad de un sistema para resistir ataques contra su seguridad (ataques en programas, datos y documentos), y para medir la integridad se tienen que definir 2 atributos adicionales: amenaza y seguridad.

- **Amenaza**
  Probabilidad de que un ataque de un tipo determinado ocurra en un tiempo determinado.

- **Seguridad**
  Probabilidad de que se pueda repeler el ataque de un tipo determinado.

Facilidad de uso

Significa "amigable al usuario", se mide en función de:

- Habilidad intelectual requerida para aprender el sistema.
- Tiempo requerido para ser moderadamente eficiente en el uso del sistema.
- Aumento neto de la productividad media cuando alguien usa el sistema de manera medianamente eficiente.
- Valoración subjetiva de la disposición de los usuarios hacia el sistema.

Integración de Métricas

La gestión de alto nivel puede establecer objetivos significativos para la mejora del proceso de desarrollo de software, evaluando las medidas de productividad y calidad del mismo. Si el proceso por puede ser mejorado, se producirá un impacto directo en los resultados que genere dicho proceso, pero para establecer estos objetivos de mejora, se debe comprender el estado actual de desarrollo de software, por ende la medición se utiliza para establecer una línea base del proceso a partir de la cual se pueden evaluar las mejoras.

Establecimiento de una línea base

Una línea base está conformada por datos recolectados de múltiples proyectos desarrollados anteriormente y tiene como objetivo obtener beneficios tanto a nivel de proceso, como de proyecto y de producto (estratégico, táctico y técnico).

Características

- Pueden ser datos muy simples o muy complejos
- Deben ser datos exactos (sin conjeturas)
- Los datos deben reunirse del mayor número de proyectos posible.
- Las métricas deben ser consistentes y las aplicaciones de las mismas deben ser semejantes para trabajar en la estimación.

Métricas Ágiles

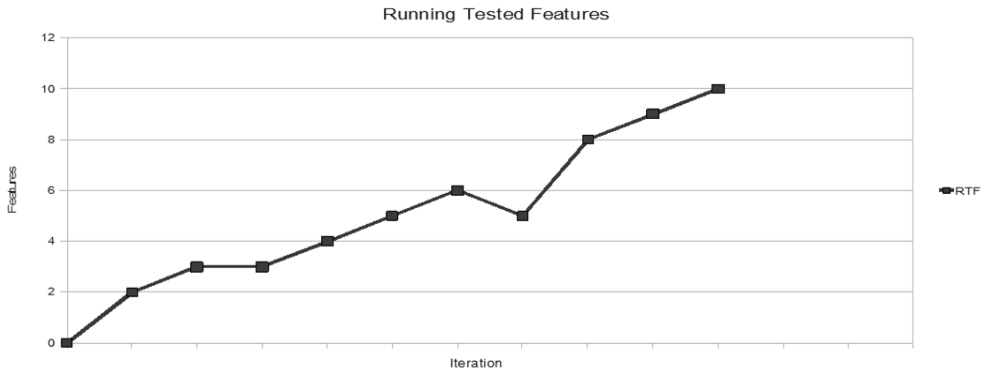
Regla de Oro Ágil sobre Métricas

En enfoques ágiles nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software valioso, es decir el software funcionando es la principal medida de progreso, por lo cual **no se debe tomar a las métricas como una actividad, sino como lo que son, una salida.** Esto quiere decir que se debe medir lo que sea necesario y nada más, es decir lo que agregue valor para el cliente.

Running Tested Features (RTF)

El software funcionando es la mejor medida de progreso y es una medida directa de los resultados entregados, desglosando el nombre de esta métrica encontramos:

- **Running:** funcionalidad entregada en un producto
- **Tested:** la funcionalidad desarrollada paso las pruebas de aceptación establecidas
- **Features:** dado que incluye aspectos determinados por el cliente.



Funciones de la métrica

- Principio
  Software funcionando es la mejor medida de progreso.

- Informacional
  Es una medida directa de los resultados entregados

- Diagnóstico
  Si es llano o disminuye en el tiempo, es un indicador de problema

- Motivacional
  Los miembros del equipo naturalmente quieren ver un incremento en RTF.

Cuando RTF no está bien

- Es cero desde el inicio en algunos sprints
- Comienza muy rápido y luego cae.
- Se comporta como un "yoyo"
- Cae demasiado rápido

Capacity

Es una estimación de cuanto trabajo puede completarse en un período de tiempo basado en la cantidad de tiempo ideal disponible del equipo. Es teórica, y resulta de gran utilidad al momento de estimar, derivando de esta la velocidad.

Como se puede medir

- Esfuerzo (horas hombre)
- Puntos de Historia (Story Points)

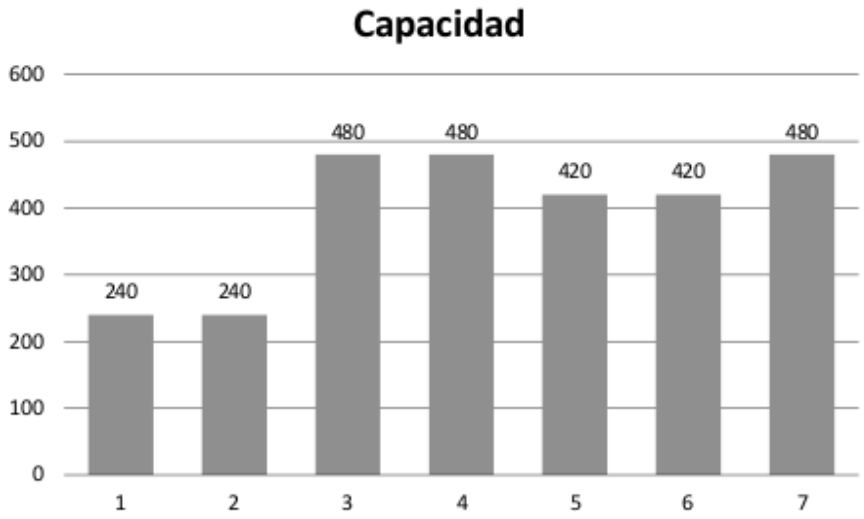
Cálculo de la capacidad

Horas de Trabajo Disponible por día (**WH**) x Días Disponibles Iteración (**DA**) = Capacity

Ejemplo

- Equipo de 8 miembros
- 4 miembros disponibles los 2 primeros sprints
- 1 miembro se casa en sprint 5 y 6
- 6 horas de trabajo

Sprint	1	2	3	4	5	6	7	Total
Horas	240	240	480	480	420	420	480	2760
Puntos de Historia	30	30	45	60	58	52	60	335



Consideraciones adicionales

- Individuos deben calcular 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
- Conocer promedio de finalización de un punto de historia para equipo/individuo

Velocity

La velocidad es una observación empírica de la capacidad del equipo para completar el trabajo por iteración, comprobable entre iteraciones de un equipo dado. Es importante aclarar, que la velocidad no se estima, se calcula. Se lo suele definir como "cuanta velocidad tengo con mi equipo para llevar algo a producción".

En definitiva, la velocidad está determinada por la cantidad de Storie Points que el Product Owner acepto, y se toma en la Sprint Review.

Consideraciones

- Solo cuenta el trabajo completado
- No es una estimación
- No es un objetivo a alcanzar
- No es comparable entre equipos
- No es comparable entre proyectos
- Me sirve para observar empíricamente mi capacidad de trabajo pero no para estimar mis objetivos.
- En Scrum baja la velocidad cuando se corren ciclos de testing y se encuentran bugs y tengo que hacer bugfixing.
- Estas dos métricas se mezclan en el Burn-downchart, acá puedo proyectas a mitad del sprint si llego o no.

Unidad de medida

Cómo planea el equipo	Unidad de Medida
Compromiso con las historias	Historias
Tamaño relativo (puntos)	Puntos de Historia
Estimación (horas ideales)	Horas ideales

Ejemplo