Gestión de Proyectos de Software

Alejandro Bedini González

Índice

Capítulo 4

Mediciones en Producto y Proceso de Software

4.1 ¿Por qué medir?

"Medimos para mejorar"
Las mejoras en el proceso de desarrollo de software y sistemas de calidad no pueden ser evaluadas sin un esfuerzo efectivo de medición. Cada organización desea mejorar sus procesos de desarrollo de software debido a que existe un tangible beneficio con la construcción de un mejor software.

A continuación se enumeran las siguientes necesidades de medición:

- ✓ Mejoras en la calidad y productividad.
- ✓ Planificación y estimación de proyectos con alguna precisión.
- ✓ Disposición del personal adecuado, bien utilizado y motivado.
- ✓ Existencia de una adecuada estructura organizacional.
- ✓ Uso de técnicas y herramientas efectivas para el proceso.
- ✓ Obtención de un espacio físico y ambiente de trabajo óptimo.

4.2 ¿Qué es una medición?

Una medición es simplemente una representación desde el mundo real y empírico a una representación matemática, donde puede ser más fácilmente entendible en atributos de entidad y las relaciones entre las otras entidades. El problema real es interpretar el comportamiento matemático y juzgar que significa en el mundo real.

En definitiva, las mediciones nos entregan una descripción cuantitativa de los procesos, productos y recursos claves permitiéndonos entender su comportamiento y resultado.

Los aspectos esenciales de la medición son:

- ✓ *Datos duros*, son cuantificables con poca o sin subjetividad (esfuerzo, volumen documentación, errores detectados, etc.
- ✓ *Datos blandos*, presentan un grado de subjetividad (habilidad y experiencia, presiones de tiempo, satisfacción del cliente, cooperación del cliente, etc.
- ✓ *Datos normalizados*, son usados con propósito comparativo (LOC, PF, CC, PO)

Existen diferentes escalas de dimensionamiento, en las cuales cada una captura mayor información que su predecesora. Ellas son:

- ✓ *Escala Nominal*, la cual ordena ítems por categoría. Un ejemplo de esta escala de medición es cuando catalogamos un lenguaje de programación: C++, Java, entre otros.
- ✓ *Escala Ordinal*, la cual ordena ítems. Un ejemplo es cuando se asigna una severidad a la falla encontrada como menor, mayor, catastrófica.
- ✓ *Escala de Intervalos*, la cual define una distancia desde un punto a otro. Este tipo de escala entrega cálculos no disponibles en la escala ordinal, como el cálculo del significado. Lamentablemente no existe el punto cero absoluto y las relaciones no tienen sentido. Por ello hay que tener cuidado cuando se hacen comparaciones. Por ejemplo en la escala de temperatura Celcius y Fahrenheit, no podemos decir que 30° Celcius es el doble de calor que 15° F.
- ✓ *Escala de Proporción*, esta escala es la que entrega mayor información y flexibilidad, debido a que incorpora el cero absoluto. Mediciones como LOC y número de defectos son mediciones de proporción.

Se podrá decir que una medición es válida si presenta la siguiente condición de representación: si éste captura en el mundo matemático el comportamiento que percibimos en el mundo empírico. Por ejemplo, debemos demostrar que H es una medición de altitud, y si A es más alto que B, entonces H(A) es más alto que H(B). Pero esta prueba debe ser empírica y esto a menudo es difícil de demostrar.

En general se desea medir los siguientes aspectos en Ingeniería de Software:

- ✓ Procesos o tareas a ejecutar (modelado, diseño, prueba).
- ✓ Productos entregados durante el proceso (documentación de diseño, código fuente, registro de pruebas).
- ✓ Recursos que permiten realizar el proceso (personal, computadoras, dinero).

4.3 Atributos internos y externos.

Cada una de estas entidades puede ser medida definiendo sus atributos internos o externos. Un atributo interno es medido directamente desde la entidad. Por ejemplo, una medida interna del código fuente es el tamaño medido por las líneas de código. Un atributo externo es una medida de la entidad con relación a una necesidad externa definida por el ambiente en el cual es desarrollada o utilizada. Por ejemplo, la mantenibilidad del código fuente (medición externa), representa la habilidad de una entidad específica de un producto (código fuente) de alcanzar los requerimientos de acomodarse a los cambios fácilmente.

Ejemplos de estas relaciones se muestran en las tablas siguientes:

PRODUCTO		
Entidades	**Atributos Internos**	**Atributos Externos**
Especificaciones	Tamaño, re-uso, modularidad, redundancia, funcionalidad, correctitud	Comprensibilidad, mantenibilidad
Diseño	Tamaño, re-uso, modularidad, acoplamiento, cohesividad, funcionalidad	Calidad, complejidad, mantenibilidad
Codificación	Tamaño, re uso, modularidad, acoplamiento, funcionalidad, complejidad algorítmico, estructuración	Confiabilidad, Usabilidad, mantenibilidad
Datos de prueba	Tamaño, nivel de cobertura	Calidad
PROCESOS		
Entidades	**Atributos Internos**	**Atributos Externos**
Especificación de requerimientos	Tiempo, esfuerzo, número de cambios en los requerimientos	Calidad, costo, estabilidad
Diseño detallado	Tiempo, esfuerzo, número de especificaciones erróneas detectadas	Costo, costo - eficacia
Pruebas	Tiempo, esfuerzo, número y errores encontrados	Costo, costo - eficacia, estabilidad
Vista de Despliegue	Diagrama de Despliegue	Nodo, componente, dependencia, localización.
RECURSOS		
Entidades	**Atributos Internos**	**Atributos Externos**
Personal	Años de experiencia, tasa de trabajo	Productividad, experiencia, inteligencia.
Equipos	Tamaño, nivel de comunicación, estructuración	Productividad, calidad.
Software	Tamaño, costo	Usabilidad, Confiabilidad
Hardware	Precio, velocidad, tamaño de la memoria	Confiabilidad
Oficinas	Tamaño, temperatura, luz	Confort, calidad

Tabla 4.1: Atributos internos y externos.

Atributo	Definición
Acoplamiento	Grado de fuerza de la inter-conexión entre los componentes del sistema.
Calidad	Grado de excelencia.
Cohesividad	Grado en que las entidades dependen o no de las otras entidades.
Correctitud	Grado en el que un entidad satisface las especificaciones y cumple los objetivos del usuario.
Complejidad	Cantidad de diversos elementos que lo componen.
Complejidad algorítmica	Grado de relativa dificultad computacional de las funciones.
Costo	Cuanto cuesta conseguir un objetivo determinado.(Monetariamente)
Costo-eficiencia	Grado en que el costo esta relacionado con la eficencia obtenida.
Confiabilidad	Grado con el que una entidad realiza su función con una precisión requerida.
Confort	Grado de comodidad.
Eficiencia	Cantidad de recursos y código requeridos por una entidad para realizar una función.
Esfuerzo	Nivel de empleo de recursos en la consecución de algún fin.
Estabilidad	Grado de permanencia y equilibrio ante efectos externos.
Estructuración	Grado en que las partes de un todo están en una estructura determinada.
Experiencia	Nivel de práctica o conocimiento de algún tema.
Funcionalidad	Cuan Práctico, eficaz, utilitario es el resultado obtenido.
Inteligencia	Capacidad de la persona sobre el conjunto de funciones que tienen por objeto el conocimiento (sensación, asociación, memoria, imaginación, entendimiento, razón, conciencia).
Mantenibilidad	Esfuerzo requerido para localizar y corregir un error en un programa en funcionamiento.
Nivel de cobertura	Grado en que abarca o contempla cierto ámbito solicitado (en este caso datos de Prueba).
Nivel de comunicación	Grado de comunicación existente entre los distintos desarrolladores.
Productividad	Grado de producción por unidad de trabajo.
Tasa de trabajo	Medida del trabajo que realiza.
Usabilidad	Esfuerzo necesario para aprender, operar, preparar entradas e interpretar la salida de un programa.
Re-uso	Grado en que una entidad se puede utilizar en otras actividades.
Otros Atributos	
Flexibilidad	Esfuerzo requerido para modificar un programa en funcionamiento.
Facilidad de prueba	Esfuerzo requerido para probar un programa (para garantizar que realiza la función deseada)
Interoperatividad	Esfuerzo requerido para aclopar
Portabilidad	Esfuerzo requerido para transferir un programa de una configuración hardware o entorno de software a otro.

Tabla 4.2: Definiciones de los atributos

En general, las mediciones que se realizan son pocas y simples; para el proceso éstas corresponden a costo y esfuerzo incurridos a lo largo del tiempo necesario en el desarrollo hasta el fin del proyecto; para el producto, líneas de código o puntos de función producidos, páginas de documentación escritas, velocidad de ejecución de los programas, tamaño de memoria requerida y defectos reportados en un período dado; para los recursos, cantidad de personal involucrado, tamaño, experiencia del equipo de desarrollo, costo y disponibilidad de las herramientas utilizadas. De las estadísticas que generan los proyectos realizados se puede obtener algunos indicadores o métricas de calidad y productividad:

Productividad **= KLOC / PM**

KLOC son las Kilo Líneas de Código fuente entregada y PM son las Personas Mes destinados al proyecto.

Calidad **= defectos / KLOC**
Costo **= $ / LOC**
Documentación **= páginas de documentación / KLOC**

En general las mediciones asociadas a las LOC están en permanente discusión, ya que son dependientes del lenguaje de programación, no son fáciles de utilizar en lenguajes no procedurales y su uso en estimaciones requiere de mucho detalle en los requerimientos.

4.4 Atributos de las técnicas de estimación.

Una forma de estimar es utilizar métricas basadas en mediciones pasadas para hacer las estimaciones. También es útil dividir el proyecto en pequeñas unidades o partes para facilitar la estimación. El proceso de gestión de proyecto de software comienza con un conjunto de actividades que, globalmente, se denomina planificación del proyecto. La primera de estas actividades es la estimación. La estimación es una actividad importante que no debe llevarse a cabo de forma descuidada.

Existen técnicas útiles para la estimación de costos, tiempos y recursos. La Planificación es parte fundamental de la planificación y como la planificación del proyecto sirve como guía para una buena ingeniería de software, no es en absoluto aconsejable embarcarse sin esta.

La estimación de costos, tiempos y recursos para el esfuerzo de desarrollo de Software requiere experiencia, acceso a una buena información histórica y coraje para confiar en medidas cuantitativas cuando todo lo que existe son datos cualitativos.
Existen tres puntos que caracterizan al proyecto a estimar:

- ✓ La ***complejidad del proyecto***, que tiene un gran efecto sobre la incertidumbre que es inherente a la planificación. La complejidad, sin embargo, es una medida relativa que se ve afectada por la familiaridad

con anteriores esfuerzos. Para un equipo de desarrollo de SOFTWARE que sólo haya desarrollado aplicaciones no interactivas, una aplicación de tiempo real parece "excesivamente compleja".

- ✓ El ***tamaño del proyecto***, que es otro factor importante y puede afectar a la precisión y eficacia de las estimaciones. A medida que aumente el tamaño o la interdependencia entre distintos elementos del SOFTWARE, éste tiende a crecer rápidamente.
- ✓ El ***grado de estructuración del proyecto***, que también tiene efecto sobre el riesgo de la estimación. En este contexto, la estructuración se refiere a la facilidad con la que las funciones pueden ser compartimentalizadas y la naturaleza jerárquica de la información que debe ser procesada.

El riesgo se mide por el grado de incertidumbre de las estimaciones cuantitativas establecidas para los recursos, los costos y las agendas. El planificador y el cliente deben tener claro que cualquier cambio en los requerimientos del software significa inestabilidad en el costo y en los tiempos.

Un análisis de los requisitos del *software* proporciona la información necesaria para las estimaciones, pero el análisis suele durar semanas o meses. Las estimaciones son necesarias "ahora". Estas estimaciones se hacen sin marco de tiempo limitado, al principio de un proyecto de *software* y deben ser actualizadas regularmente.

Una de las actividades importantes en este aspecto son las reuniones iniciales que se mantienen entre el cliente y el analista. Aunque en un comienzo no se encuentre por dónde empezar e incluso se parezcan mucho más a una conversación entre dos adolescentes, de ésta nacen las bases para un buen proyecto.

En dicha reunión se realiza un conjunto de preguntas, las primeras son:

- ✓ ¿Quién está detrás de la petición para este trabajo?.
- ✓ ¿Quién usará esta solución?.
- ✓ ¿Cuál será el beneficio económico de la solución?.
- ✓ ¿Hay otro origen para la solución?.

El siguiente conjunto de preguntas que se realizan, está orientadas a un mejor conocimiento del problema y del cliente y son las siguientes:

- ✓ ¿Cómo caracterizará el cliente un buen rendimiento que será generado por una solución exitosa?
- ✓ ¿A cuál(es) problema(s) se dirigirá esta solución?
- ✓ ¿Puede describir o mostrar el ambiente en el cual será usada la solución?
- ✓ ¿Hay fuerzas o resultados de rendimiento especiales que afectan la forma en la cual la solución es abortada?

El conjunto final de preguntas se enfoca hacia la eficacia del encuentro:

- ✓ ¿Es Usted la persona correcta para responderlas?
- ✓ ¿Es Usted un contestador oficial?
- ✓ ¿Son mis preguntas relevantes para el problema que tiene?
- ✓ ¿Estoy haciendo muchas preguntas?, ¿Hay algo más que deba preguntarle?

Este conjunto de preguntas buscan lograr una comunicación más expedita, lo cual es esencial para establecer la envergadura del proyecto. Esto es recomendable sólo en su inicio, luego se puede complementar con elementos de la solución de los problemas, negociación y especificación.

4.5 Estimación de costos en el Software.

En todo proyecto existe un presupuesto asignado, el cual debe ser controlado y respetado. Para realizar esto es necesario planificar adecuadamente el qué hacer, por lo cual la primera actividad a realizar es la de estimar el costo del desarrollo, lo que se realiza simultáneamente con la itineración.

Para estimar recursos, costo y tiempo en un proyecto de software se necesita experiencia, acceso a información histórica y hacer uso de métricas cuantitativas cuando existen los datos para ello. Las estimaciones tienen asociado en forma inherente, un grado de riesgo e incertidumbre que incide en la probabilidad de éxito, en la estimación y por ende en el resultado.

Los componentes principales de costos son:

- ✓ Hardware.
- ✓ Entrenamiento.
- ✓ Esfuerzo.

Existen siete técnicas posibles para estimar los costos del Software

- ✓ *Modelos algorítmicos.* Se utiliza un modelo basado en información histórica que relaciona información histórica de costo con alguna métrica del proyecto.
- ✓ *Juicio experto.* El costo es obtenido por consenso de expertos en el desarrollo. La experiencia debe ser en las tecnologías y aplicación a desarrollar.
- ✓ *Estimación por analogía.* Se basa en el desarrollo previo de proyectos similares.
- ✓ *Ley de Parkinson.* El trabajo se expande hasta llenar todo el tiempo disponible.
- ✓ *Precio para ganar.* El costo se estima según el presupuesto disponible.
- ✓ *Estimación top down.* El costo se estima considerando la funcionalidad total del producto y cómo ésta es provista por las subfunciones interactuantes. El costo se basa en las funciones lógicas.
- ✓ *Estimación bottom up.* Se estima el costo de cada componente para luego agregarse en un costo total.

Los factores que afectan a la estimación de un proyecto son básicamente dos: la complejidad del proyecto y el tamaño del mismo.
Al principio, el costo del software constituía un pequeño porcentaje del costo total de los sistemas basados en computadora. Un error considerable en las estimaciones del costo del software tenía relativamente poco impacto. Hoy en día, el software es el elemento más caro de la mayoría de los sistemas

informáticos. Un gran error en la estimación del costo puede ser lo que marque la diferencia entre beneficios y pérdidas. Sobrepasarse en el costo puede ser desastroso para el equipo de desarrollo

Desde un punto de vista ideal, se deben aplicar conjuntamente todas las técnicas apropiadas, usando cada una de ellas como comprobación de las otras. Las técnicas de descomposición utilizan un enfoque de «divide y vencerás» para la estimación del proyecto de software.

Dentro de la mayor parte de las organizaciones, la estimación de costos se basa en las experiencias pasadas. Los datos históricos se usan para identificar los factores de costo y determinar la importancia relativa de los diversos factores dentro de la organización. Lo anterior, por supuesto, significa que los datos de costos y productividad de los proyectos actuales deben ser centralizados y almacenados para un empleo posterior.

La *estimación jerárquica hacia abajo* se enfoca primero a los costos del nivel del sistema, así como a los costos de manejo de la configuración, del control de calidad, de la integración del sistema, del entrenamiento y de las publicaciones de documentación. Los costos del personal relacionado se estiman mediante el examen del costo de proyectos anteriores que resulten similares.

En la *estimación jerárquica hacia arriba*, primero se estima el costo del desarrollo de cada módulo o subsistema; tales costos se integran para obtener un costo total. Esta técnica tiene la ventaja de enfocarse directamente a los costos del sistema, pero se corre el riesgo de despreciar diversos factores técnicos relacionados con algunos módulos que se desarrollarán. La técnica subraya los costos asociados con el desarrollo independiente de cada módulo o componente individual del sistema, aunque puede fallar al no considerar los costos del manejo de la configuración o del control de calidad.

En la práctica, ambas técnicas deben desarrollarse y compararse para que iterativamente se eliminen las diferencias obtenidas.

En la Tabla 4.3 se muestran algunas de las variables para medir costos en proyectos TI.

POSIBLES COSTOS DE UN SISTEMA DE INFORMACIÓN	
Costos de elaboración	✓ Costos de consulta ✓ Costos de alquiler o de compra de los equipos ✓ Costos de modificación del emplazamiento de los equipos (aire acondicionado, seguridad, etc.). ✓ Costos del capital ✓ Costos de gestión y de personal
Costos de puesta en marcha	✓ Costos del software del sistema operativo ✓ Costos de la instalación de los equipos de comunicaciones (líneas telefónicas y de datos) ✓ Costos del personal para la puesta en marcha ✓ Costos de contratación de personal y alquileres ✓ Costos de interrupción en el resto de la organización ✓ Costos de gestión necesarios para dirigir la actividad inicial
Costos relacionados con el proyecto	✓ Costos de adquisición del software de aplicación ✓ Costos de las modificaciones del software para que encajen con los sistemas locales ✓ Costos de personal, gastos generales, etc. del desarrollo de aplicaciones internas ✓ Costos de entrenamiento del personal en el uso de las aplicaciones ✓ Costos de recogida de información y procedimiento de instalación ✓ Costos de preparación de la documentación ✓ Costos de la supervisión del desarrollo
Costos del proceso	✓ Costos de mantenimiento del sistema (hardware, software e instalaciones) ✓ Costos de suministros (electricidad, teléfono, etc.) ✓ Costos de depreciación del hardware ✓ Costos del personal involucrado en la gestión de los sistemas de información, operación y actividades de planificación

Tabla 4.3: Variables para medir costos en proyectos TI

4.6 Estimación de recursos

Recursos Humanos: Determinar el personal necesario, la posición dentro de la organización y la especialidad requerida. El número de personas sólo puede ser determinado después de hacer una estimación de esfuerzo.

Recursos de Hardware: Durante la planificación del proyecto de software, se deben considerar básicamente tres categorías de HW; el sistema de desarrollo, la máquina objetivo y los demás elementos de HW del nuevo sistema. Sistemas de desarrollo, es el HW donde se desarrolla el proyecto, máquina objetivo es el HW que utilizará nuestro cliente y el sistema, lo demás serán los periféricos asociados.

Recursos de Software: Se utilizan distintas herramientas que ayudan en el desarrollo del nuevo sistema, estas herramientas tienen diferentes propósitos como: gestión de proyectos, soporte, análisis y diseño, programación, mantenimiento, entre otros.