ESTIMACIÓN DEL COSTO DE DESARROLLO DE SOFTWARE POR EL MÉTODO DE REGRESIÓN DE LOS MÍNIMOS CUADRADOS

Estimated Cost of Software Development Method of Least Squares Regression

Rubén Alcón López Efraín Silva Sánchez
ralconlo@gmail.com, efrainsilva_777@hotmail.com

RESUMEN

El proyecto pretende determinar el costo de desarrollo de productos software a partir de la identificación de un conjunto de atributos clave (v.gr. tamaño, nivel de calidad, esfuerzo de desarrollo, etc.) en base al uso de las técnicas de la regresión lineal. En el presente artículo se presenta algunas características del producto objeto de estudio y se describen aspectos relacionados al proceso de desarrollo de los mismos; en base a estos conceptos, a un proceso de selección de variables y a un conjunto de datos a elaborarse se planteará la aplicación del método de la regresión lineal.

Palabras Clave: Software, Costo, Estimación, Métricas, Gestión, Control, Proceso Software, Medición.

ABSTRACT

The project aims to determine the cost of software product development from the identification of a set of key attributes (eg size, level of quality, development effort, etc.) based on the use of regression techniques linear. In this paper we present some characteristics of the product under study and describe aspects of the process of developing them; based on these concepts, a variable selection process and a data set to be developed will consider the application the linear regression method.

Keywords: Software, Cost, Estimate, Metrics, Management, Control, Process Software, Measurement.

Los proyectos de desarrollo de software (PDS) se diferencian de los otros proyectos de ingeniería tradicional en la naturaleza lógica del producto software; el software se desarrolla no se fabrica.

En todos los proyectos de ingeniería, la buena calidad se adquiere mediante un buen diseño; en el caso del software, la etapa de construcción incide pobremente en su calidad. Existen diferentes modelos, normas y estándares de calidad orientados a regular los procesos de los proyectos de desarrollo de software y a constituir guías que permiten mejorar la productividad y la calidad del producto software final (CMMI: Capability Maturity Model Integration, ISO/IEC 9126, ISO/IEC 12207 e ISO/IEC 15504); engloban los "conocimientos", proporcionan un marco para implementar procedimientos de aseguramiento de la calidad, proporcionan continuidad y entendimiento entre el trabajo de personas y organizaciones distintas.

En la gestión de proyectos de desarrollo de software se gestiona para lograr alguno de los siguientes objetivos:

- Defectos mínimos
- Máxima satisfacción de usuarios
- Tiempo de respuesta mínimo
- Buen nivel de mantenibilidad
- Buen nivel de extensibilidad
- Robustez
- Alta correctitud

Considerando que en los PDS la relación de requisitos y otras características que tendrá el producto final no es posible definirlas a cabalidad hasta que el proceso de desarrollo ya se inicio; la planificación y consiguiente gestión representan un obstáculo determinante para su éxito. El proceso de estimación del software es un área en el que se están proponiendo métodos y técnicas desde hace 15 años; la mayoría de los procesos existentes, describen cómo aplicar un método simple de predicción, que normalmente está basado en uno o más modelos algorítmicos.

La estimación consiste en determinar el valor de una variable desconocida a partir de otras conocidas, o de una pequeña cantidad de valores conocidos de esa misma variable. Las técnicas de regresión ofrecen una alternativa valiosa para estimar los costos de desarrollo de software.

EL SW COMO BIEN DE CAPITAL

Los bienes de Capital se definen como aquéllos que no se destinan al Consumo, sino a seguir el proceso productivo, en forma de auxiliares o directamente para incrementar el Patrimonio material o financiero (Capital). Si consideramos la función del software dentro de la organización, podríamos decir que el software se constituye en un bien de capital debido a que incorpora conocimientos asociados a los procesos de negocio de la empresa; es decir, incluye un conjunto de conocimientos necesarios y suficientes para realizar determinado tipo de tareas necesarias e imprescindibles en la dinámica de la organización. Las aplicaciones informáticas, sistemas y componentes de software son herramientas o elementos de herramientas que ayudan a producir bienes y servicios que quieren los clientes de la organización.

Las TIC´s tienen gran impacto en la economía actual, por lo que cuando un sistema software tiene problemas en su funcionamiento por: errores no detectados, averías no resueltas o sabotajes; éste, decae en su rendimiento, se interrumpe o se bloquea y las consecuencias económicas pueden ser cuantiosas y podrían afectar el entorno social de la organización.

En resumen, el proceso de desarrollo de nuevos bienes de capital -software- es un proceso de aprendizaje, pues, el conjunto de conocimientos precisos para desarrollar software es algo difícilmente disponible y por ello el proceso de acopiarlo, estructurarlo e implementarlo es un proceso de descubrimiento y de diálogo; es decir, de aprendizaje. Además, este proceso no es un proceso individual, sino un entramado de procesos individuales de muchas personas con diferentes intereses y especialidades, trabajando generalmente en grupos e intercambiando conocimientos y aprendiendo unos de otros.

SPEM (Software Process Engineering Metamodel) es el estándar definido por el OMG (Object Management Group) para la representación de procesos de desarrollo y determina el lenguaje mediante el cual se definen procesos de software. SPEM ofrece un marco de trabajo para el modelado, documentación, presentación, gestión e

intercambio de los procesos de desarrollo de Software y sus componentes, dando una sintaxis y una estructura común para cada aspecto del proceso de desarrollo.

PROCESO SW (PS)

Una de las principales líneas de trabajo para la mejora de la calidad de los productos software, es el estudio y mejora de los procesos mediante los cuales el software es desarrollado y mantenido -se supone que existe una correlación directa entre la calidad del proceso y la calidad del producto obtenido-. El área de trabajo que aborda esta problemática, dentro del campo de la Ingeniería del Software, es conocida con el nombre de Tecnología de Proceso Software (TPS), o simplemente "Proceso Software"; esta última denominación es la más utilizada. Como disciplina autónoma, la investigación en la TPS comenzó en los años 80 y la primera contribución importante fue la confirmación de que el desarrollo y mantenimiento del software son procesos complejos, que requieren un esfuerzo colectivo y creativo. Por tanto, la calidad de un producto software depende fuertemente de las personas, la organización y los procedimientos utilizados para crearlo, entregarlo y mantenerlo.

El proceso software es un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software.

Un PS hace uso de:

- Tecnología de desarrollo de software: herramientas, infraestructuras y entornos.
- Métodos y Técnicas de desarrollo de software: cómo usar la tecnología.
- Comportamiento Organizacional: la ciencia de las organizaciones y la personas.
- Marketing y economía: cómo cualquier otro producto, el software debe dirigirse a clientes reales y, por tanto, está sujeto a las reglas del mercado.

Por tanto, debemos prestar atención a la compleja interrelación que se produce en un PS entre los diversos factores organizacionales, culturales, tecnológicos y económicos. Un PS es un Proceso de Negocio (PN, proceso para entregar un resultado a un cliente) realizado por una organización para desarrollar y mantener un producto software.

Proceso Software - Naturaleza

La naturaleza de los PS´s generalmente se caracteriza por:

- Su complejidad. Representan excepciones determinados por circunstancias impredecibles; cada uno tiene peculiaridades particulares.
- No son procesos de ingeniería "pura"; se desconoce las abstracciones adecuadas respecto de un problema, dependen demasiado de las personas; el diseño y producción no están claramente separados; los presupuestos, calendarios, nivel de calidad no pueden ser planificados de forma fiable.
- No son (completamente) procesos creativos: algunas partes pueden ser descritas en detalle, algunos procedimientos son impuestos.
- Están basados en referencias subjetivas y descubrimientos que dependen de la comunicación, coordinación y cooperación de terceros dentro de marcos de trabajo predefinidos.

- Avances en su implementación. Los entregables, generan nuevos requerimientos y los costes del cambio del software no inciden en el costo final.
- El éxito del desarrollo final depende de la implicación del usuario y de la coordinación de muchos niveles de la organización (ventas, desarrollo técnico, cliente, etc.).

En el proceso software (PS), los procesos de diferentes proyectos tienden a seguir patrones comunes que pueden capturarse a fin de conseguir homogeneidad en el producto final. El estudio de los procesos de producción de software ha llevado al desarrollo de varios Ciclos de Vida del software que pueden ser empleados en la Ingeniería de Software (IS): ciclo de vida en Cascada, Evolutivo y en Espiral; los modelos de ciclo de vida ayudan a comprender mejor el PS y a organizar el orden de actividades globales de la producción de software.

El objetivo final de la tecnología de proceso software (PS) es lograr que la representación de un proceso pueda ser utilizada para conducir los actuales procesos de desarrollo y mantenimiento del software. Con este fin, surgen varios conceptos:

- Process-sensitive Software Engineering Environment (PSEE). Ambientes de Ingeniería de software con capacidad de gestionar PML´s (lenguajes de modelado de procesos).
- Modelo de Procesos (MP). Representación abstracta de una familia de procesos expresada en una adecuada notación de modelado de procesos (formalismo).

La disponibilidad de un MP (computarizado) proporciona capacidades para:

- Completar procesos. Soporte directo a los desarrolladores vía control de su trabajo, su coordinación con otros, etc.);
- Automatización. Invocación automática de herramientas no interactivas);
- Dirección: soporte indirecto, como información del estado actual del proceso, el significado de los puntos de decisión, etc.;
- Eficiencia: un MP preciso es primordial en el aumento de la efectividad, ya que proporciona una base no ambigua para la comunicación entre los procesos.

Los MP juegan un rol esencial en la supervisión, simulación, validación, verificación y mejora de procesos:

- Supervisión. Permiten una clara comprensión de lo que puede ser observado y por qué.
- Simulación. El comportamiento de un proceso puede ser estudiado al menor coste sin desarrolladores reales o herramientas.
- Validación. La simulación y supervisión, junto con la inspección de modelos, permiten que las propiedades de los modelos sean determinadas.
- Verificación. Para probar formalmente el cumplimiento de las propiedades de interés del proceso.
- Mejora. Revisión y optimización de las capacidades de los procesos para su mejora.

Los MP´s presentan procesos con niveles incrementales de detalle, capturando subprocesos cada vez más pequeños, correspondientes a asuntos cada vez más

detallados: vgr. Ciclo de Vida; MP genérico; MP detallado (customized); Reificable (enactable) o proceso que debe ser ajustado a los requerimientos establecidos en la etapa de definición de requisitos o en estándares de calidad utilizados; Reificado (enacting) o proceso ajustado a exigencias del usuario y/o estándares de desarrollo.

En cada nivel puede distinguirse: 1) La realización (performance) del proceso; resultado de la interacción de procesos, personas y herramientas, 2) la Reificación (enactment) del proceso; relacionado con el acto de conducir el proceso (interpretar con más detalle el MP), 3) la Dualidad de pertenencia a los dominios; las herramientas y las personas al ser condicionadas por el entorno, también pertenecen al dominio de reificación, mientras que un proceso se realiza en el mundo real y es reificado en el mundo abstracto del modelo computarizado; puesto que los computadores son parte del mundo real, se dice que la reificación es parte de la realización.

Las vistas que ofrece un MP pueden ser a nivel de: Actividades, Productos, Recursos, y de Roles. Una vista no puede ser definida sin usar conceptos de otros MP´s.

ENTORNOS DE PRODUCCIÓN

Un entorno de producción de software generalmente incluye: un proceso en desarrollo P; es decir, el proceso de producción del mundo real incluyendo actores humanos y herramientas que acompañan a todas las actividades dirigidas al desarrollo y mantenimiento de un producto software, y un modelo de procesos MP que representa y captura el estado actual de las actividades y tiene como fin: guiar, hacer cumplir o automatizar partes del proceso de producción o de mantenimiento. Idealmente el proceso de desarrollo P y el modelo de procesos MP deben estar perfectamente alineados: el estado interno del MP debe ser una fiel representación del actual estado de los asuntos en el mundo real; por lo tanto P es una instancia de PM.

Cualquier PS en el mundo real es un proceso creativo y dinámico que abarca a mucha gente, y no puede ser reducido a la programación de autómatas. Hay diversas razones por las que puede cambiar un PS:

- Puede ser erróneo;
- Ciertos pasos importantes no están previstos;
- El MP puede ser genérico y necesita ser detallado para obtener resultados específicos;
- Las presunciones sobre las cuales se construyó el MP ya no son válidas;
- Las dinámicas políticas, humanas y tecnológicas pueden inducir a su cambio.

Como resultado, el proceso P del mundo real y el MP deben evolucionar de forma conjunta y coherente

Metaproceso.

La evolución de un PS es un proceso completo y se denomina meta-proceso; este incluye:

- Los pasos para cambiar los procesos del mundo real, y los pasos para introducir cambios en el MP.

Su principal función es asegurar que P y MP permanezcan consistentes.

El gestor del proyecto necesitará implicar los servicios del modelador para desarrollar un MP aumentado, validar el nuevo modelo, y decidir cuándo comenzar a realizarlo.

LENGUAJES DE MODELADO DE PS

Un lenguaje de modelado de PS (LMP) expresa los procesos software (PS) en

forma de modelos de procesos software (MP).Todos los elementos de proceso (actividades, roles, recursos, etc.) deben poder describirse.

Los elementos del metaproceso (evolución del proceso) también se deben poder expresar.

Un LMP puede ser:

- Formal: tiene sintaxis y semántica formales.
- Semi-formal: tiene notación formal (normalmente gráfica) pero no tiene semántica formal.
- Informal: sin sintaxis y semánticas formales (lenguaje natural).

El LMP debe adaptarse a estas necesidades del MP.

Hay 6 elementos de proceso (primarios) que debe poder modelar un LMP:

– Actividades.

– Productos.

– Roles.

– Personas.

– Herramientas.

– Soporte para la evolución (al menos del MP): a nivel técnico (p.e., mediante reflexión o interpretación), y a nivel conceptual (mediante un metamodelo asociado).

DESARROLLO DE SOFTWARE – VARIABLES INTERVINIENTES - PROCESO DE CLASIFICACIÓN

Tomando en cuenta las diversas perspectivas existentes en el proceso de desarrollo de software y el impacto de los nuevos modelos de desarrollo generados a partir de la masificación del uso de internet además del informe del Standish Group (1995) se hace necesario establecer una serie de consideraciones acerca de los proyectos de software a fin de establecer el conjunto definitivo de variables a incluirse en la relación final a establecerse.

Los factores que afectan el éxito de los proyectos según Baker, Murphy y Fisher, quienes estudiaron 650 proyectos en los Estados Unidos son los siguientes:

1. Compromiso con el proyecto en el establecimiento de calendarizaciones, presupuestos y objetivo de desempeño técnicos.
2. Frecuente retroalimentación de la organización patrocinadora.
3. Frecuencia de retroalimentación del cliente.
4. Compromiso del cliente, del patrocinador, comprometido en el establecimiento de calendarizaciones, presupuestos y objetivo de desempeño técnicos.
5. Estructura de la organización adecuada al equipo del proyecto.
6. Participación del equipo del proyecto en la determinación de la calendarización y los presupuestos.
7. Entusiasmo del patrocinador.
8. Deseo del patrocinador de crear las capacidades internas.
9. Procedimientos de control adecuados, especialmente en relación con los cambios.
10. Uso con juicio de las técnicas de programación en red.
11. Un mínimo de agencias públicas y de gobierno involucradas.

12. Falta de un gobierno excesivo.

13. Soporte público entusiasta.

14. Falta de impedimentos legales.

Por otra parte los siguientes factores son considerados como críticos en el éxito de un proyecto:

1. Misión del proyecto – Metas y direcciones generales definidas con claridad al inicio del proyecto.

2. Soporte Administrativo de Alto Nivel – Ayuda de la alta dirección para proveer los recursos necesarios y la autoridad y poder para el éxito del proyecto.

3. Auscultación del Cliente – Comunicación, auscultación y escucha activa de todas las partes impactadas.

4. Personal – Reclutamiento, selección y entrenamiento del personal necesario para el equipo del proyecto.

5. Tareas Técnicas – Disponibilidad de la tecnología y experiencia (expertise) necesarias para el cumplimiento del las acciones técnicas específicas.

6. Aceptación del Cliente – El acto de "vender" el final del proyecto a los usuarios finales.

7. Monitoreo y Retroalimentación – Provisión a tiempo y de manera adecuada de información de control en cada una de las etapas del proceso de implementación

8. Comunicación – La provisión de una red apropiada y datos necesarios para todos los actores clave en la implementación del proyecto.

9. Resolución de Problemas – Habilidad de manejar crisis inesperadas y desviaciones del plan.

CICLO DE VIDA DEL SOFTWARE

Un modelo de ciclo de vida define el estado de las fases a través de las cuales se mueve un proyecto de desarrollo de software. El primer ciclo de vida del software, "Cascada", fue definido por Winston Royce a fines del 70. Desde entonces muchos equipos de desarrollo han seguido este modelo. El modelo cascada es restrictivo y rígido, lo cual dificulta el desarrollo de proyectos de software; en su lugar, muchos modelos nuevos de ciclo de vida han sido propuestos, incluyendo modelos que pretenden desarrollar software más rápidamente, o de manera incremental o de una forma más evolutiva, o precediendo el desarrollo a escala total con algún conjunto de prototipos rápidos.

Un modelo de ciclo de vida de software es una visión de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas.

Un modelo de ciclo de vida del software:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo, y
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

Así, los modelos por una parte suministran una guía para los ingenieros de software con el fin de ordenar las diversas actividades técnicas en el proyecto, por otra parte suministran un marco para la administración del desarrollo y el mantenimiento, en el sentido en que permiten estimar recursos,

definir puntos de control intermedios, monitorear el avance, etc.

SELECCIÓN DE MODELOS DE CICLO DE VIDA

Los modelos, suministran una guía con el fin de ordenar las diversas actividades técnicas en el proyecto de desarrollo de software e intentan suministrar un marco para la administración en el desarrollo y el mantenimiento. Aunque todos ellos son compatibles unos con otros, un proyecto puede decidir cuáles enfoques son más útiles en situaciones especiales. V.gr. Criterios a considerar:

- Madurez de la aplicación (relacionado a la probabilidad que muchos requerimientos comenzarán a conocerse sólo después del uso del sistema).
- Complejidad del problema y de la solución.
- Frecuencias y magnitudes esperadas de los cambios de requerimientos.
- Financiamiento disponible, y su perfil como una función del tiempo.
- Acceso de los desarrolladores a los usuarios.
- Certeza de requerimientos conocidos.
- Tolerancia al riesgo.
- Planes y presupuestos críticos
- Grado de lentitud de construcción dentro de los planes y presupuestos.

Uno de los factores que más influyen en el proceso de desarrollo de software y que prácticamente acompaña a toda aplicación es el hecho de que en su mayoría, no hay forma de tener todos los requerimientos corregidos antes del desarrollo del software. Muchas veces los requerimientos emergen a medida que la aplicación o partes de ella están disponibles para experimentación práctica. En todos los casos, el trabajo comienza con la determinación de objetivos, alternativas y restricciones, paso que a veces se llama recolección preliminar de requisitos.

GESTIÓN DEL PROYECTO DE SOFTWARE

Para que un proyecto software se lleve a cabo con éxito es necesario comprender el ámbito del trabajo a realizar, los riegos que podría enfrentarse, las tareas que se llevarán a cabo, las etapas que comprende el proyecto, el coste del proyecto, y el plan a seguir. Este conocimiento lo proporciona la gestión del proyecto de software; esta actividad empieza antes del inicio del trabajo técnico, continúa a medida que el desarrollo del producto software evoluciona desde un nivel conceptual y finaliza en el momento en que concluye el proyecto.

Luego de establecer el ámbito y los objetivos del proyecto de desarrollo de software; debe considerarse soluciones alternativas e identificar las restricciones técnicas y de gestión; sin esta información no hay manera de: realizar estimaciones de coste razonables, identificación realista de las tareas del proyecto o establecer un plan de trabajo adecuado que proporcione una indicación significativa del proyecto. Esta tarea –establecer el ámbito- a cargo del cliente y el responsable del proyecto generalmente; identifica los fines globales del proyecto sin establecer a priori la forma en la que se llegará a alcanzarlos. El ámbito, identifica las funciones primordiales que debe llevar a cabo el software e intenta limitar esas funciones de manera cuantitativa a fin de disponer de referencias que permitan controlar y hacer seguimiento del proceso de desarrollo.

La gestión del proyecto de software es el primer nivel del proceso de ingeniería de software, porque cubre todo el proceso de desarrollo. Para conseguir un proyecto de software fructífero se debe comprender el ámbito del trabajo a realizar, los riesgos en los que se puede incurrir, los imprevistos a enfrentar, los recursos requeridos, las tareas a llevar a cabo, el esfuerzo (costo) a consumir y el plan a seguir. La gestión es el conjunto de Actividades y tareas ejecutadas por una o más personas con el propósito de planificar y controlar las actividades de otras personas para alcanzar un objetivo o completar una actividad que no puede ser realizada por ellas actuando independientemente.

ACTIVIDADES DE LA GESTIÓN

- Planificación: Predeterminación de tareas necesarias y de un curso de acción para alcanzar los objetivos del proyecto.
- Organización: Arreglo de las relaciones entre las unidades de trabajo del proyecto para el cumplimiento de objetivos y el otorgamiento de responsabilidad y autoridad para obtener esos objetivos.
- Staffing: Selección y entrenamiento de personas para puestos en la estructura del proyecto.
- Dirección: Acción de dirigir y crear una atmósfera que apoye y motive a la gente para alcanzar los resultados finales deseados.
- Control: Establecimiento, medición y evaluación del desempeño de las actividades a determinados periodos de tiempo y en función de objetivos planeados para su ejecución.

HABILIDADES PARA LA GESTIÓN

- Habilidades Técnicas. Conocimiento y pericia en actividades que involucran métodos, procesos y procedimientos; implica trabajar con herramientas y técnicas específicas.
- Habilidades Humanas. Habilidad para trabajar con gente, del esfuerzo cooperativo, del trabajo en equipo, de la creación de un ambiente donde la gente se sienta segura y libre de expresar sus opiniones.
- Habilidades Conceptuales. Habilidad para ver la "Imagen Completa", para reconocer los elementos relevantes de una situación, y para entender las relaciones entre los elementos.
- Habilidades de Diseño. Habilidad para resolver problemas de manera que beneficie a la empresa –desarrollar software-.

PLANIFICACIÓN DE UN PROYECTO DE INGENIERÍA DE SOFTWARE

La planificación de un proyecto de software no difiere de la planificación de cualquier proyecto de ingeniería. Se identifica una serie de tareas del proyecto, se establecen interdependencias entre las tareas, se estima el esfuerzo asociado con cada tarea, se hace la asignación del personal y de otros recursos, se crea una red de tareas y se desarrolla una agenda de fechas. La planificación temporal para proyectos de desarrollo de software puede verse desde dos perspectivas bastante diferentes:

- En la primera, la fecha final de lanzamiento del sistema basado en computadora ya ha sido (irrevocablemente) establecida. La organización encargada de desarrollar el software se ve forzada a distribuir el esfuerzo dentro del marco prescrito.

- El segundo enfoque de la planificación temporal del software asume que se han estudiado unos límites cronológicos aproximados pero que la fecha final es fijada por la organización del software. El esfuerzo se distribuye para hacer un mejor uso de los recursos y la fecha final se define después de un cuidadoso análisis del elemento software.

  Desafortunadamente la primera perspectiva se encuentra bastante más a menudo que la segunda.

  Principales problemas en la planificación de un proyecto de ingeniería de software:

- Requerimientos incorrectos e incompletos.
- Muchas especificaciones de requerimientos son inestables y sujetas a cambios mayores.
- La planificación no se lleva a cabo por la creencia errónea de que es una pérdida de tiempo y los planes cambiarán de todos modos.
- La planificación de costos y plazos no es actualizada y se basa en necesidades de mercadeo y no de los requerimientos del sistema.
- Los costos y plazos no son re estimados cuando los requerimientos del sistema o el ambiente de desarrollo cambia.
- Es difícil estimar el tamaño y complejidad del proyecto de software de modo de realizar una estimación de costos y plazos realista.
- No se manejan factores de riesgo.
- La mayoría de las organizaciones de desarrollo de software no recolectan datos de proyectos pasados.
- Las compañías no establecen y/o no disponen de políticas o procesos de desarrollo de software.

ORGANIZACIÓN DE UN PROYECTO DE INGENIERÍA DE SOFTWARE

Involucra desarrollar una estructura organizacional efectiva y eficiente para asignar y completar las tareas del proyecto y establecer las relaciones de autoridad y responsabilidad entre ellas. Los principales problemas en la organización de un proyecto de ingeniería de software incluyen los siguientes:

- Dificultad en la determinación de la mejor estructura organizacional para una organización y/o ambiente particular (por ejemplo tipo proyecto, funcional o matriz) para gestionar el proyecto.
- Mala definición de tareas del proyecto y consiguientes problemas en las tareas de la estructura organizacional del proyecto.
- Dificultades de integración del personal a la estructura organizacional del proyecto.
- Acción dual de muchos líderes de equipo que buscan desarrollarse tanto técnicamente como en la gestión de su equipo de trabajo.

DIRECCIÓN DE UN PROYECTO DE INGENIERÍA DE SOFTWARE

Dirigir un proyecto de ingeniería de software consiste en aquellas actividades de gestión que involucran aspectos interpersonales y de motivación por medio de las cuales el personal del proyecto entiende y contribuye a alcanzar los objetivos del proyecto. Una

vez que los subordinados son entrenados y orientados, el jefe de proyecto tiene una responsabilidad continua por clarificar sus asignaciones, guiándolos hacia la mejora de la productividad, y motivándolos a trabajar con entusiasmo y confianza hacia las metas del proyecto.

Los principales problemas en la dirección son:

- Fallas para tener una comunicación efectiva entre las entidades del proyecto y aquellas que no pertenecen al proyecto.
- El dinero no es un motivador suficiente para los desarrolladores de software.
- Las compañías y los jefes no poseen las técnicas y herramientas apropiadas para motivar a los ingenieros de software.
- Los clientes y gerentes no reconocen el impacto potencial en el software causado por un aparentemente cambio trivial, por ejemplo, ellos creen que es "sólo un problema simple de programación".

CONTROL DE UN PROYECTO DE INGENIERÍA DE SOFTWARE

Controlar es el conjunto de actividades de gestión utilizadas para asegurar que el proyecto va de acuerdo a lo planificado. El desempeño y los resultados se miden contra los planes, se notan las desviaciones, y se toman acciones correctivas. El control, es un sistema de retroalimentación que provee información acerca de cuán bien va el proyecto. El control responde las preguntas:

¿Está el proyecto en itinerario?

¿Está dentro de los costos?

¿Existen problemas potenciales que causen retrasos en alcanzar los requerimientos dentro del presupuesto y plazo?

Principales problemas en el control

- Muchos métodos de control de proyectos de desarrollo de software confían en los gastos del presupuesto para medir el "progreso" sin considerar el porcentaje de trabajo que lo acompaña.
- La visibilidad del progreso en un proyecto de software es difícil de medir.
- La calidad no es requerida, monitoreada o controlada.
- A menudo los estándares para el desarrollo de software no están escritos o, si lo están, no se aplican.
- El cuerpo de conocimiento llamado métricas de software (usadas para medir productividad, calidad, y progreso de un producto software) no es gestionado adecuadamente.

PROBLEMAS Y ERRORES COMUNES EN LA GESTIÓN DEL RECURSO HUMANO

A continuación aparecen algunos de los errores clásicos relacionados con las personas.

- Motivación débil. La motivación probablemente tiene mayor efecto sobre la productividad y la calidad que ningún otro factor.
- Personal mediocre. Después de la motivación, la capacidad individual, así como las relaciones como equipo, probablemente tienen la mayor influencia en la productividad.

- Empleados problemáticos incontrolados amenaza la velocidad de desarrollo.
- Comportamientos Heroicos. Algunos desarrolladores de software ponen un gran énfasis en la realización de hazañas en los proyectos.
- Añadir más personal a un proyecto retrasado. Este error puede quitar productividad a los miembros del equipo existente.
- Oficinas repletas y ruidosas. Los desarrolladores consideran sus condiciones de trabajo como insatisfactorias; 60 por 100 indica que no tienen suficiente silencio ni privacidad.
- Fricciones entre los clientes y los desarrolladores. Pueden presentarse de distintas formas.
- Expectativas pocos realistas. Una inspección del Standish Group marcó las expectativas realistas como uno de los cinco factores principales necesarios para asegurar el éxito de los proyectos de software (Standish Group, 1995).
- Falta de un promotor efectivo del proyecto. Soportar muchos de los aspectos del desarrollo, necesita un promotor del proyecto de alto nivel, además de una planificación realista, el control de cambios y la introducción de nuevos métodos de desarrollo.
- Falta de participación de los implicados. Todos los principales participantes del esfuerzo de desarrollo de software deben implicarse en el proyecto.
- Falta de participación del usuario. La inspección de Standish Group descubrió que la razón número uno de que los proyectos de Sistemas de Información tuviesen éxito es la implicación del usuario (Standish Group, 1995).
- Orientación Política del Gestor. Las orientaciones políticas asociadas al desarrollo de proyectos de software son: Los "políticos" están especializados en la "gestión", centrándose en las relaciones con sus directivos; los "investigadores" se centran en explorar y reunir la información; los "aislacionistas" crean fronteras para el proyecto que mantienen alejados a los que no son miembros del equipo y los "generalistas" hacen un poco de todo.
- Discrepancias en la planificación. Muchos problemas del desarrollo del software se deben a expectativas irreales acerca de la conclusión del proyecto en las fechas previstas en la planificación.

PROBLEMAS CON EL PROCESO

Los errores relacionados con el proceso malgastan el talento y el esfuerzo del personal. A continuación se muestran algunos de los peores errores relacionados con el proceso.

- Planificación excesivamente optimista. Se infravalora el alcance del proyecto, minando la planificación efectiva y reduciendo las actividades críticas para el desarrollo.
- Gestión de riesgos insuficiente. Algunos errores no son lo suficientemente habituales como para considerarlos clásicos.
- Fallos de los contratistas. La realización de partes de un proyecto cuando tienen

impacto en el proyecto total.

- Planificación insuficiente. Si no planificamos para conseguir un desarrollo rápido, no podemos esperar obtenerlo.
- Abandono de planificación bajo presión. Los equipos de desarrollo hacen planes y rutinariamente los abandonan cuando se tropiezan con un problema en la planificación.
- Pérdida de tiempo en el inicio difuso. El "inicio difuso" es el tiempo que transcurre antes de que comience el proyecto.
- Escatimar tiempo en las actividades. Los proyectos se aceleran intentando acortar las actividades "no esenciales", y puesto que el análisis de requerimientos, la arquitectura y el diseño no producen código directamente, son los candidatos fáciles.
- Diseño inadecuado. Un caso especial de escatimar en las actividades iniciales, es el diseño inadecuado. Proyectos acelerados generan un diseño indeterminado, no asignando suficiente tiempo para él y originando un entorno de alta presión que hace difícil la posibilidad de considerar alternativas en el diseño.
- Escatimar en el control de calidad. En los proyectos "urgentes" se suele eliminar las revisiones del diseño y del código, la planificación de las pruebas y realizar sólo pruebas superficiales.
- Convergencia prematura o excesivamente frecuente. Bastante antes de su fecha de entrega programada, hay un impulso para preparar el producto para la entrega y completar tareas relacionadas a su habilitación y uso eliminando las funciones que no van a estar listas a tiempo y demás.
- Omitir tareas necesarias en la estimación. Si no se cuenta con datos de proyectos anteriores es posible olvidar las tareas menos visibles; pero que deben añadirse en algún momento. El esfuerzo omitido suele aumentar el plan de desarrollo en un 20 o 30 por 100.

ERRORES EN LA DEFINICIÓN DEL PRODUCTO

A continuación se muestran los errores clásicos relacionados con la forma en la que se define el producto.

- Exceso de requerimientos. Algunos proyectos tienen más requerimientos de los que necesitan, desde el mismo inicio; la eficiencia se fija como requisito más de lo que es necesario, y puede generar una planificación del software innecesariamente larga.
- Cambio de las prestaciones. Los proyectos sufren como media sobre un 25 por 100 de cambios en los requerimientos a lo largo de su vida (Jones, 1994). Este cambio puede producir un aumento en el plan de al menos un 25 por 100, resultando fatal para los proyectos.
- Desarrolladores meticulosos. La nueva tecnología puede dar lugar a veces a probar nuevas prestaciones de lenguajes o entorno, crear una propia implementación de alguna utilidad -vista en otro producto- la necesite o no su producto.
- Tiras y aflojas en la negociación. Un retraso en el plan de un proyecto que progresa más lento de lo esperado y al que además se le añade tareas completamente nuevas presenta una

situación curiosa -rezago-.

- Desarrollo orientado a la investigación. Los planes de desarrollo de software son razonablemente predecibles; los planes en la investigación sobre software ni siquiera son predecibles teóricamente.

ERRORES EN EL USO DE LA TECNOLOGÍA

El resto de los errores clásicos están relacionados con el uso correcto o incorrecto de la tecnología moderna.

- Síndrome de la panacea. Se confía demasiado en las ventajas de tecnologías que no se han usado antes (generadores de informes, diseño orientado a objetos, lenguajes de programación, etc.) y se tiene poca información sobre lo buenas que serían en un entorno de desarrollo concreto.
- Sobreestimación de las ventajas del empleo de nuevas herramientas o métodos. Las organizaciones mejoran raramente su productividad a grandes saltos, sin importar cuántas nuevas herramientas o métodos empleen o lo bueno que sean.
- Cambiar de herramientas a mitad del proyecto. Es un viejo recurso que funciona raramente.

MEDICIÓN Y MÉTRICAS

Los datos sobre costes, planificación y productividad en los proyectos de desarrollo de software hacen posible disponer de datos que sirven para que en la gestión del proyecto se realicen mejoras y se reduzca costes y niveles de incertidumbre en la planificación. Si además de datos de coste y de planificación se registra datos históricos; como el tamaño de los programas en líneas de código, el factor de complejidad o cualquier otra medida, se tendrá las bases para la planificación de proyectos futuros.

El desarrollo eficiente del Proceso Software, demanda unos conocimientos básicos sobre las medidas o métricas del software. El término "Métricas del Software" comprende muchas actividades, todas ellas relacionadas de alguna manera con la idea de mejorar la calidad del software. Una Métrica se define como: Evaluación del grado en el cual un producto o proceso posee un atributo determinado (extensión, cantidad, dimensiones, capacidad o tamaño) (IEEE, 1993). Las métricas pueden ser:

Métricas directas e indirectas

Una métrica es **directa** si se puede medir directamente el atributo y su valor no depende de la medida de otros atributos (longitud del código fuente, duración del proceso de prueba, número de defectos...). Una métrica es **indirecta** si comprende la medición de varios atributos; es decir, si deriva de otros atributos (productividad, estabilidad de requisitos, densidad de defectos en un módulo, etc.).

Métrica objetiva y subjetiva

Una métrica es **objetiva** si su valor no depende del observador y es **subjetiva** en caso contrario.

La experiencia indica que una métrica se usa únicamente si es intuitiva y fácil de calcular. Si se requiere múltiples conteos y se han de utilizar complejos cálculos, la métrica no será ampliamente utilizada.

Las métricas del software abarcan muchas actividades y son múltiples las razones que justifican su uso:

- Estimación de coste y esfuerzo.
- Modelos y medidas de productividad.

- Recolección de datos.
- Modelos y evaluación de la calidad.
- Modelos de fiabilidad.
    - Están incluidas en la mayoría de los modelos de calidad.
    - La especialización de los modelos de fiabilidad permite aumentar el entendimiento y control de los productos.
- Evaluación del rendimiento.
    - Valoración del rendimiento incluye características observables como tiempos de respuesta y características internas como eficiencia de los algoritmos.
- Métricas estructurales y de complejidad.
    - Para realizar predicciones sobre atributos de calidad (fiabilidad, facilidad de mantenimiento,...) se pueden medir atributos estructurales sobre representaciones del software que están disponibles antes que el código.
- Gestión mediante métricas
    - La realización de gráficos basados en diferentes medidas a lo largo del proyecto permite conocer el estado del mismo.
- Evaluación y comparación de métodos y herramientas
    - Las investigaciones cuidadosas, con análisis y mediciones controladas sobre una herramienta o método permiten hacerlos más productivos para situaciones particulares.

Por otra parte la Medición se entiende como: Proceso objetivo y empírico por el que se asignan números o símbolos a atributos de entidades del mundo real con objeto de describirlas. La posibilidad de medir es el fundamento de las disciplinas científicas y de ingeniería.

Sin la medición es muy difícil evaluar y experimentar las técnicas y los métodos de ingeniería del software. No se puede controlar lo que no se puede medir y no se puede predecir lo que no se puede medir.

La medición contribuye a superar algunos problemas habituales en el desarrollo del software:

Problema	**Medir ayuda a**
Requisitos incorrectos	Desarrollar requisitos verificables expresados en términos medibles.
Toma de decisiones	Proporciona evidencia cuantificable para apoyar las decisiones.
Falta de control	Hacer más visible el desarrollo e identificar problemas anticipadamente.
Exceso de gasto	Realizar predicciones de coste y plazo justificables.

Costes de mantenimiento	Recomendar determinadas estrategias de prueba e identificar los módulos problemáticos.
Evaluación de nuevos métodos	Valorar los efectos en la productividad y la calidad.

Objetivos del proceso de medición:

- **Gestión** durante el proceso de Ingeniería del Software, más concretamente;
- **Predicción**: estimación de los atributos que tendrá una entidad que no existe aún (coste de un proyecto, esfuerzo necesario).
- **Evaluación**: comprobación del cumplimiento de ciertas características por una entidad que ya existe (calidad del diseño, fiabilidad del software, etc.).
- **Experimentación** en Ingeniería del Software.

Comparando los valores obtenidos en el proceso de medición entre ellos y con los estándares aplicados en la organización, es posible establecer criterios acerca de la calidad del software o de los procesos del software.

EL PROCESO DE MEDICIÓN

El primer paso de la medición es identificar los atributos o entidades a medir. Estos pueden ser de tres tipos:

- **Productos**: componentes, entregas o documentos resultantes de una actividad de proceso.
- **Procesos**: atributos de actividades relacionadas con el software.
- **Recursos**: entidades requeridas por una actividad de proceso

Dentro de cada clase anterior se puede distinguir:

- **Atributos internos**: Son aquellos que pueden ser medidos examinando el proceso, producto o recurso mismo. Los atributos internos describen los productos de software de forma que dependen únicamente del producto mismo.

El producto puede ser descrito en función de su **tamaño.** Se puede definir un conjunto de atributos para medir el tamaño del software:

- **Longitud:** tamaño físico del producto.
- **Funcionalidad:** funciones que proporciona el producto al usuario.
- **Complejidad** (de tiempo o espacio): recursos necesarios (de tiempo o memoria de ordenador) para implementar una solución particular.

Las **propiedades estructurales** del software son atributos internos relacionados con la calidad del producto. Los tipos de medidas estructurales son:

- **Flujo de control**: secuencia en que se ejecutan las instrucciones.
- **Flujo de datos**: seguimiento de cómo los datos se crean y se manejan por un programa.
- **Estructura de los datos**: organización de los datos independiente del programa.

Los principales productos – del proceso de desarrollo - que resulta útil medir son: la

especificación, el diseño y el código.

Medidas de tamaño (Longitud del código, funcionalidad ...)

Medidas de diseño

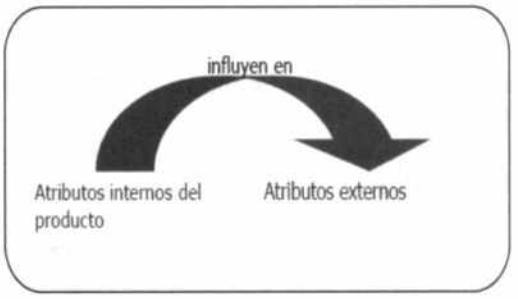
- Acoplamiento: Grado de interdependencia entre módulos
- Cohesión: Grado en el que los componentes locales de un módulo colaboran para realizar una tarea concreta
- Modularidad
- Otros relacionados al diseño

Medidas de Complejidad (Estructuras de datos, estructura lógica ...)

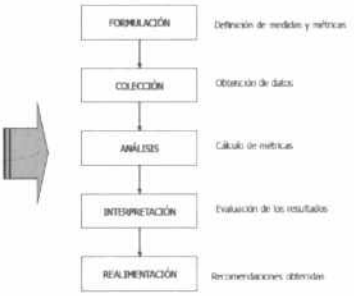
Otros atributos internos

- **Atributos externos**: se miden con respecto a cómo el proceso, producto o recurso se relaciona con su entorno. Dependen del comportamiento del producto en un entorno determinado, v.gr.
    - Portabilidad
    - Fiabilidad
    - Usabilidad
    - Facilidad de mantenimiento
    - Escalabilidad

Generalmente, el interés de conocer el valor de un **atributo interno** es que pueda dar información sobre algún **atributo externo** de interés para el observador.



V.gr. El **número de requisitos** de una especificación de requisitos de sistema puede ayudarnos a predecir el **esfuerzo** asociado al proyecto.



Proceso de medición de Roche

Estimación

Una estimación se define como: Una evaluación tentativa o un cálculo preliminar o un cálculo preliminar del costo de un proyecto. La estimación también se entiende como juicio basado en unas impresiones u opiniones. La gestión exitosa de proyectos de software pasa por tres etapas básicas. Primero se estima el tamaño del producto, luego el esfuerzo necesario para construir un producto con este tamaño y por último se estima la duración cronológica del proyecto.

En muchos casos, las estimaciones se hacen valiéndose de la experiencia pasada como única guía. Si un nuevo proyecto es bastante similar en tamaño y función a un proyecto pasado, es probable que el nuevo proyecto requiera aproximadamente la misma cantidad de esfuerzo, que dure aproximadamente el mismo tiempo y que cueste aproximadamente lo mismo que

el trabajo anterior. Pero si el proyecto es totalmente distinto, la experiencia no será suficiente.

DISCUSIÓN

Los conceptos y métodos descritos brevemente en este artículo, servirán como base para encarar el objetivo final del trabajo: Estimación del Costo de Desarrollo de Software por mínimos cuadrados. Considerando el conjunto de factores determinantes para el éxito de los PDS se completará la tarea *Determinación de Relación de Esfuerzo* en base a la cual se estimará el costo de desarrollo de SW a través de los métodos de la regresión lineal.

En estadística, la regresión lineal o ajuste lineal es un método matemático que modela la relación entre una variable dependiente Y, las variables independientes Xi y un término aleatorio ε. Este modelo puede ser expresado como:

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

Y_t : variable dependiente, explicada o regresando.

$X_1, X_2, \cdots, X_p$: variables explicativas, independientes o regresores.

$\beta_0, \beta_1, \beta_2, \cdots, \beta_p$:

parámetros, miden la influencia que las variables explicativas tienen sobre el regresando.donde, β_0 es la intersección o término "constante", las $\beta_i \ (i > 0)$ son los parámetros respectivos a cada variable independiente, y p es el número de parámetros independientes a tener en cuenta en la regresión. La regresión lineal puede ser contrastada con la regresión no lineal (De wikipedia).

En esta investigación las técnicas de regresión lineal y ajuste por mínimos cuadrados se utilizarán para representar la relación de costo -modelo lineal que representa el costo de desarrollo de software- y la estimación de los parámetros de la relación. El proceso de estimación por mínimos cuadrados se realizará a partir de un conjunto de datos –muestra- de tamaño n relativos al costo de proyectos de una clase y complejidad predefinidas.

v.gr. Si CTDS es el costo total de desarrollo del producto y par_j son los parámetros y $CRRT_i$, CX, PRD, OTR son variables que determinan el costo ($CRRT_i$: Con restricción en el tiempo de entrega; CX: Complejidad X; PRD: Presupuesto de desarrollo; OTR: Otra variable relacionada con el proceso de desarrollo de software), el modelo de regresión lineal será:

$CTDS = par_1 * CRRTi + par_2 * CX + par_3 * PRD + \ldots + par_k * OTR + error\ de\ ajuste$

CONCLUSIONES

La regresión lineal múltiple es una alternativa que permite considerar el conjunto de variables críticas que determinan el costo final del proyecto de desarrollo de software; en este sentido, la identificación de este conjunto de variables, es una tarea importante que merece bastante atención. Los resultados a obtenerse en el proyecto dependerán de la muestra de costos de desarrollo de productos software considerado.

BIBLIOGRAFIA

Marcela Varas C.; Año 2000; Gestión de Proyectos de Desarrollo de Software; Departamento de Ingeniería Informática y Ciencias de la Computación Facultad de Ingeniería, Universidad de Concepción..

Standish Group Report; 1995; Presentación: Medición del Software.

Iván Darío Páez Anaya; Mayo 2012;Estudio Empírico del Estado Actual de la Estimación de Software en Pymes de Colombia, Universidad EAFIT.

David Cerrillo; Mayo 1999; Planificación y Gestión de Sistemas de Información. ESTIMACIÓN DEL SOFTWARE, UNIVERSIDAD DE CASTILLA-LA MANCHA.

Pablo Szyrko, Diego Rubio, Definición de un metamodelo para la validación de procesos de software organizacionales basa_ dos en modelos estándares, Depto de IS de Inf. Universidad Tecnológica Nacional. Argentina.