GUIA DE PREGUNTAS 2

Contenido

1. Defina el objetivo de la planificación del proyecto software.

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, coste y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto. Además, las estimaciones deberían definir los escenarios del "mejor caso" y "peor caso" de forma que los resultados del proyecto puedan limitarse. El objetivo de la planificación se logra mediante un proceso de descubrimiento de la información que lleve a estimaciones razonables.

2. Defina ámbito del proyecto software.

La primera actividad de la planificación del proyecto de software es determinar el ámbito del software. Se deben evaluar la función y el rendimiento que se asignaron al software durante la ingeniería del sistema de computadora, para establecer un ámbito de proyecto que no sea ambiguo, ni incomprensible para directivos y técnicos. Se debe delimitar la declaración del ámbito

del software. El ámbito del software describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad. Se evalúan las funciones descritas en la declaración del ámbito, y en algunos casos se refinan para dar más detalles antes del comienzo de la estimación. Dado que las estimaciones del coste y de la planificación temporal están orientadas a la función, muchas veces es útil llegar a un cierto grado de descomposición. Las consideraciones de rendimiento abarcan los requisitos de tiempo de respuesta y de procesamiento. Las restricciones identifican los límites del software originados por el hardware externo, por la memoria disponible y por otros sistemas existentes.

3. Enuncie las cuatro dimensiones de la factibilidad del software.

¿Se puede construir el software de acuerdo al ámbito definido? ¿Es factible el proyecto?

La factibilidad del software tiene 4 dimensiones:

- Tecnología: ¿Es factible un proyecto técnicamente?¿está dentro del estado actual de la técnica?
- Financiación: ¿Es factible financieramente? ¿puede realizarse un coste asumible por la empresa y por el cliente?
- Tiempo: ¿pueden los proyectos adelantarse a los de la competencia?
- Recursos: ¿la organización cuenta con los recursos suficientes para tener éxito?

Tanto el equipo de desarrollo y las demás personas involucradas en el software deben determinar si puede ser construido dentro de las dimensiones especificadas.

4. Enuncie recursos requeridos para desarrollo de software.

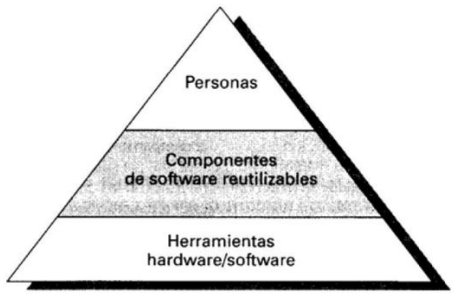

FIGURA 5.2. Recursos del proyecto.

- Herramientas de hardware y software entorno de desarrollo que proporciona la infraestructura de soporte al esfuerzo de desarrollo
- Componentes de software reutilizables bloques de software que pueden reducir drásticamente los costes de desarrollo y acelerar la entrega
- Personal

5. Enuncie las cuatro categorías de recursos de software propuestas por Bennatan.

- Componentes ya desarrollados: El software existente se puede adquirir de una tercera parte o provenir de uno desarrollado internamente para un proyecto anterior.

- Componentes ya experimentados. Especificaciones, diseños, código o datos de prueba existentes desarrollados para proyectos anteriores que son similares al software que se va a construir para el proyecto actual.

- Componentes con experiencia Parcial. Especificaciones, diseños, código o datos de prueba existentes desarrollados para proyectos anteriores que se relacionan con el software que se va a construir para el proyecto actual, pero que requerirán una modificación sustancial.

- Componentes nuevos. Los componentes de software que el equipo de software debe construir específicamente para las necesidades del proyecto actual.

6. Enuncie directrices a tener presente cuando los componentes reutilizables se especifiquen como recurso.

1. **Si los componentes ya desarrollados cumplen los requisitos del proyecto, adquiéralos.** El coste de la adquisición y de la integración de los componentes ya desarrollados serán casi siempre menores que el coste para desarrollar el software equivalente. Además, el riesgo es relativamente bajo.
2. **Si se dispone de componentes ya experimentados, los riesgos asociados a la modificación y a la integración generalmente se aceptan.** El plan del proyecto debería reflejar la utilización de estos componentes.
3. **Si se dispone de componentes de experiencia parcial para el proyecto actual, su uso se debe analizar con detalle.** Si antes de que se integren adecuadamente los componentes con otros elementos del software se requiere una gran modificación, proceda cuidadosamente - el riesgo es El coste de modificar los componentes de experiencia parcial algunas veces puede ser mayor que el coste de desarrollar componentes nuevos.

7. Enuncie opciones para realizar estimaciones de costes y esfuerzos.

- Para realizar estimaciones seguras de costes y esfuerzos tenemos varias opciones posibles: Dejar la estimación para más adelante, Es mucho mas fiable realizar una estimación al final del proyecto. Aunque muy poco practica

- Basar las estimaciones en proyectos similares ya terminados. Bastante útil si el proyecto actual es similar a los proyectos anteriores, aunque tampoco es un indicador fiable

- Utilizar técnicas de descomposición de relativamente sencillas para generar las estimaciones de coste y de esfuerzo del proyecto. la estimación del coste y del esfuerzo puede realizarse de una forma escalonada idónea.

- Utilizar uno o más modelos empíricos para la estimación del coste y esfuerzo del software.

8. Enuncie sobre que se apoya la precisión de una estimación del proyecto de software.

En sistemas de este tipo, se describen las características de la organización de desarrollo (por ejemplo, la experiencia, el entorno) y el software a desarrollar. De estos datos se obtienen las estimaciones de coste y de esfuerzo. Obtienen las estimaciones de coste y de esfuerzo. Cada una de las opciones viables para la estimación de costes del software, sólo será buena si los datos históricos que se utilizan como base de la estimación son buenos. Si no existen datos históricos, la estimación del coste descansará sobre una base muy inestable

9. Enuncie los enfoques del problema del tamaño del software propuestos por Putnam y Myers.

Tamaño en «lógica difusa». Este enfoque utiliza las técnicas aproximadas de razonamiento que son la piedra angular de la lógica difusa. Para aplicar este enfoque, el planificador debe identificar el tipo de aplicación, establecer su magnitud en una escala cuantitativa y refinar la magnitud dentro del rango original. Aunque se puede utilizar la experiencia personal, el planificador también debería tener acceso a una base de datos histórica de proyectos para que las estimaciones se puedan comparar con la experiencia real.

Tamaño en punto de función. El planificador desarrolla estimaciones de características del dominio de información

Tamaño de componentes estándar. El software se compone de un número de componentes estándar, El planificador de proyectos estima el número de incidencias de cada uno de los componentes estándar, y utiliza datos de proyectos históricos para determinar el tamaño de entrega por componente estándar.

Tamaño del cambio. Este enfoque se utiliza cuando un proyecto comprende la utilización de software existente que se debe modificar de alguna manera como parte de un proyecto. El planificador estima el número y tipo (por ejemplo: reutilización, añadir código, cambiar código,

suprimir código) de modificaciones que se deben llevar a cabo. Mediante una «proporción de esfuerzo» para cada tipo de cambio, se puede estimar el tamaño del cambio.

10. Indique como “Líneas de Código – LCD” y “Puntos de Función – PF” se puede utilizar durante la estimación del proyecto de software.

LDC como variable de estimación,

la descomposición funcional es absolutamente esencial y, a menudo, se lleva hasta considerables niveles de detalle. Debido a que los datos requeridos para estimar los Puntos de Función son más macroscópicos, en nivel de descomposición al que se llega cuando PF es la variable de estimación es considerablemente menos detallado. También, debe de tenerse en cuenta que mientras que LDC se estima directamente, PF se determina indirectamente mediante la estimación del número de entradas, salidas, archivos de datos, peticiones e interfaces externas, entre otras. Esta técnica trata de definir el tiempo y el costo del proyecto en base a la cantidad de líneas de código se tienen que escribir, cual es el costo por línea y cuantas líneas de código desarrollamos en un mes.

Ventajas

- Facil de calcular
- Base de calculo de modelos de estimación de costos de software existente
- Existencia de literatura al respecto
- Facil de automatizar

Desventajas

- Dependencia del lenguaje de programación
- Difícil de estimar en etapas tempranas de un proyecto
- Difícil de calcular en lenguajes no procedurales

PF como variable de estimación

El método de los Puntos de Función (PF) es un tipo de medida indirecta del software y del proceso por el cual se desarrolla. En lugar de calcular las LDC, las métricas orientadas a la función se centran en la "funcionalidad" o "utilidad" del programa. Los puntos de función se obtienen realizando una relación empírica basada en medidas cuantitativas del dominio de información del software y valoraciones subjetivas de la complejidad del software. se pueden computar sin forzar que la especificación siga un modelo o técnica en particular)

• Primero computamos la cantidad de Puntos Función No Ajustados (UFC)

• Luego se obtienen los FP multiplicando los UFC por un Factor de Complejidad Técnica (TCF)

11. Enuncie las áreas que cubre la jerarquía de modelos de estimación COCOMO II.

Modelo de composición de aplicación. Utilizado durante las primeras etapas de la ingeniería del software, donde el prototipado de las interfaces de usuario, la interacción del sistema y del software, la evaluación del rendimiento, y la evaluación de la madurez de la tecnología son de suma importancia.

Modelo de fase de diseño previo. se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca

Modelo de fase posterior a la arquitectura. se aplica en la etapa de desarrollo, después de definir la arquitectura del sistema, y en la etapa de mantenimiento

12. Defina la Ecuación del Software.

$$E = (LOC * B^{0.333} / P)^3 * (1 / t^4)$$

Donde,

E	Esfuerzo en hombres-año.
t	Duración del proyecto en años.
B	Factor especial de destrezas. Para programas pequeños *B* vale 0.16, para programas intermedios vale 0.28, para programas mayores vale 0.39.
P	Parámetro de productividad, para un software de tiempo real, *P* vale 2,000, para comunicaciones vale 10,000, para software científico vale 12,000 y para aplicaciones comerciales de sistemas vale 28,000.

Es importante señalar que la “La Ecuación Del Software"
(1) una estimación del tamaño (en LDC)

(2) una indicación de la duración del proyecto en meses o años (E).

13. Enuncie directrices a seguir ante la posibilidad de adquirir productos de software catalogados como caros. De un ejemplo de árbol de decisión para apoyar la decisión desarrollar-comprar.

En muchas áreas de aplicación del software, a menudo es más rentable adquirir el software de computadora que desarrollarlo. Los gestores de ingeniería del software se enfrentan con una decisión desarrollar o comprar. Para productos de software más caros, se pueden aplicar las directrices siguientes:

- 1 desarrollo de una especificación para la función y rendimiento del software deseado. Definición de las características medibles, siempre que sea posible.
- 2 estimación del coste interno de desarrollo y la fecha de entrega.
- 3ª Selección de tres o cuatro aplicaciones candidatos que cumplan mejor las especificaciones.
- 3b Selección de componentes de software reutilizables que ayudarán en la construcción de la aplicación requerida.
- 4 desarrollo de una matriz de comparación que presente la comparación una a una de las funciones clave. Alternativamente, realizar el seguimiento de las pruebas de evaluación para comparar el software candidato.
- 5 Evaluación de cada paquete de software o componente según la calidad de productos anteriores, soporte del vendedor, dirección del producto, reputación, etc.
- 6 Contacto con otros usuarios de dicho software y petición de opiniones.

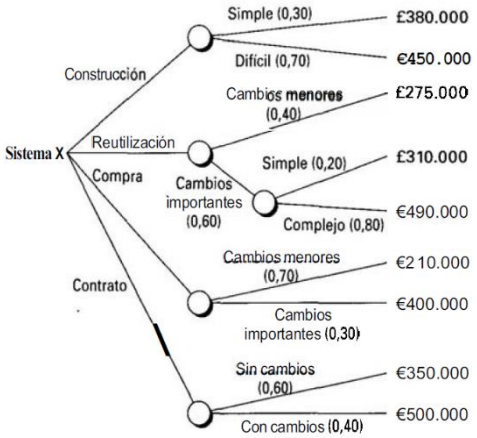

FIGURA 5.6. Árbol de decisión para apoyar la decisión desarrollar-comprar.

(1) construir el sistema *X* desde el principio

(2) reutilizar los componentes de experiencia parcial existentes de para construir el sistema

(3) comprar un producto de software disponible y modificarlo para cumplir las necesidades locales (4) contratar el desarrollo del software a un vendedor externo.

Es necesario Considerar muchos criterios -no sólo el coste-durante el proceso de toma de decisiones. La disponibilidad, la experiencia del desarrollador/vendedor/ contratante conformidad con los requisitos, la política «local», y la probabilidad de cambios son sólo uno de los pocos criterios que pueden afectar la última decisión

14. Enuncie las seis funciones genéricas de las herramientas automáticas de estimación.

1. Dimensionamiento de las entregas del proyecto. Se estima el «tamaño» de uno o más productos de software. Los productos incluyen la representación externa del software (por ejemplo, pantallas, mes), el software en sí (por ejemplo, KLDC), su funcionalidad y la información descriptiva (por ejemplo, documentos).

2. Selección de las actividades del proyecto. Se selecciona el marco de trabajo del proceso adecuado y se especifica el conjunto de tareas de ingeniería del software.

3. Predicción de los niveles de la plantilla. Se especifica el número de personas disponibles para realizar el trabajo. Esto es muy importante, puesto que la relación entre las personas disponibles y el trabajo (esfuerzo previsto) no es muy lineal.

4. Predicción del esfuerzo del software. Las herramientas de estimación utilizan uno o más modelos que relacionan el tamaño dé las entregas del proyecto con el esfuerzo necesario para producirlas.

5. Predicción del coste del software. Dado los resultados del paso 4, los costes pueden calcularse asignando proporciones del trabajo a las actividades del proyecto señaladas en el paso 2.

6. Predicción de la planificación del software. Cuando se conoce el esfuerzo, los niveles de la plantilla y las actividades del proyecto, se puede realizar un borrador de la planificación asignando el trabajo a través de actividades de ingeniería del software basadas en modelos recomendados para la distribución del esfuerzo

los resultados obtenidos por las herramientas de estimación se deben usar sólo como «punto de partida» para la obtención de estimaciones-no como única fuente para la estimación-.