

INTRODUCCIÓN

La causa principal de los problemas que se presentan al momento de realizar un proyecto de programación es la falta de planeación.

Para evitar esto, se requiere poner atención en la planeación de las distintas etapas del desarrollo del producto (análisis, diseño, implementación, pruebas y mantenimiento).

Un producto de programación se entiende mejor según se desarrollan el análisis, el diseño y la instrumentación; sin embargo, el proyecto de desarrollo no debe estar supeditado a la disponibilidad de suficiente información para iniciar la planeación preliminar.

En este trabajo se exponen ciertas funciones que han de realizarse al momento de querer hacer una planeación adecuada, dependiendo del tipo de proyecto que se llevará a cabo.

PLANEACIÓN DE PROYECTOS

DEFINICIÓN DEL PROBLEMA, METAS Y REQUISITOS

El primer paso en la planeación de un proyecto de programación es preparar un enunciado breve del problema que se solucionará y de las restricciones que existen en su resolución. El enunciado definitivo del problema debe de incluir una descripción de la situación actual y de las metas que debe lograr el nuevo sistema.

La definición del problema requiere de un entendimiento cabal del dominio del problema y del entorno de éste. Las técnicas para obtener este conocimiento, por parte del planeador, son entrevistas con el cliente, observación de las tareas problemáticas, y desarrollo de las reales. El planeador debe ser muy hábil en las técnicas de definición del problema, ya que distintos representantes del cliente tendrán diferentes puntos de vista, y prejuicios que influirán en la percepción del alcance del problema.

Algunas veces los sistemas se construyen para dar solución a un síntoma, y no a la causa primaria del problema. Esto ocurre cuando el problema se entiende, pero no puede resolverse debido a circunstancias económicas, políticas o sociales, cuando el cliente no es capaz de comunicar el problema real o cuando el planeador no entiende la explicación del cliente sobre el problema.

El segundo paso en la planeación de un proyecto de programación es determinar lo apropiado de una solución computacional. Además de ser eficaz en términos de costo, un sistema computacional debe aceptarse social y políticamente. Para ser eficiente en costo, un nuevo producto de programación debe proporcionar los mismos servicios e información que el sistema antiguo, usando menos tiempo y personal, o proporcionar servicios e información que antes eran inaccesibles. Un sistema que desplace a muchos trabajadores puede ser económica y técnicamente posible, pero inaceptable social o políticamente para el usuario.

Una vez dado el enunciado preciso del problema y la indicación de las restricciones que existen para la solución, se pueden formular metas y requisitos preliminares. Las metas son logros por alcanzar; sirven para establecer el marco de referencia para el proyecto de desarrollo del producto de programación. Éstas se usan para el proceso de desarrollo y los productos finales, y pueden ser cualitativas y cuantitativas.

Los requisitos especifican las capacidades que debe tener un sistema para la solución de un problema. Éstos se establecen para la funcionalidad, el rendimiento, el equipo, la programación en el equipo, la programación y las interfaces con el usuario. Los requisitos pueden establecer también estándares de desarrollo y de control de calidad tanto para el desarrollo como para el producto; deben ser cuantificados siempre que sea posible.

Las metas y los requisitos de alto nivel se pueden expresar en términos de atributos de calidad que el sistema deberá poseer.

Es importante que los criterios de alto nivel para la aceptación del sistema se definan durante la fase de planeación. Cada requisito debe incluir el método que se empleará durante su verificación.

DESARROLLO DE UNA ESTRATEGIA DE SOLUCIÓN

La tendencia de utilizar la primera solución que aparece es un problema importante de la ingeniería de software. Una manera de evitarlo es al desarrollar primero una estrategia de solución. Esta no es un plan detallado de solución, sino un enunciado general sobre la naturaleza de las posibles soluciones.

Una estrategia de solución debe considerar todos los factores externos que son visibles para el usuario del producto, y debe redactarse de tal manera que permita caminos alternos para el diseño del producto.

Se deben considerar varias estrategias de solución antes de elegir alguna, aunque los planificadores deben escoger una o más para poder realizar estudios de factibilidad y de estimados de costo preliminares. La estrategia proporciona un marco de referencia para el diseño y la instrumentación del producto de programación.

Las estrategias de solución se deben crear sin tomar en cuenta aspectos de factibilidad, puesto que no es posible ser creativo y crítico al mismo tiempo. Por lo regular, una idea no lógica produce otras ideas, y éstas pueden ser muy razonables. A menudo, una composición de las ideas de distintos puntos de vista es la mejor estrategia de solución y ésta aparece sólo después de enumerar todas las soluciones obvias. La generación de ideas se da mejor con grupos de personas con experiencia en técnicas de tormenta de ideas.

La factibilidad de cada estrategia de solución propuesta se debe establecer por el análisis de las restricciones de la solución. Éstas establecen las fronteras del espacio de soluciones; el análisis de factibilidad determina si una estrategia propuesta es posible dentro de dichas fronteras. Una estrategia de solución es factible si las metas y requisitos del proyecto se pueden satisfacer dentro de las restricciones de tiempo disponible, recursos y tecnología por medio de esa estrategia.

Las técnicas para determinar la factibilidad de una estrategia de solución comprenden el estudio de casos, análisis del peor caso, simulación y construcción de prototipos. Un prototipo difiere de un modelo de simulación en que aquél incorpora algunos componentes del sistema real. Las instrumentaciones de prototipos suelen tener una funcionalidad limitada, poca confiabilidad y características de operación pobre. Los prototipos se construyen durante la etapa de planeación para examinar aspectos técnicos y para simular despliegues al usuario, formatos y diálogos.

Cuando se recomienda una estrategia de solución, es muy importante documentar las razones por las que se rechazan otras estrategias. Esto da justificación a la estrategia recomendada, y puede prevenir revisiones equivocadas en fechas posteriores.

Una estrategia de solución debe incluir una lista con prioridades de las características del producto. Sin esta guía, un diseñador o programador puede realizar serios errores de juicio, lo que provocaría la insatisfacción del cliente con el producto final.

Las prioridades de las características de los productos son útiles para indicar la manera en que las capacidades pueden irse desarrollando en fases dentro de la evolución del sistema. Muchos ingenieros de programación proponen el desarrollo de sistemas como una serie de mejoras sucesivas hacia un sistema núcleo. Las prioridades del producto son útiles en la planeación de las versiones sucesivas que se construirán.

PLANEACIÓN DEL PROCESO DE DESARROLLO

La planeación del proceso de desarrollo incluye definir un modelo para el ciclo de vida del producto. Este ciclo incluye las actividades requeridas para definirlo, desarrollarlo, probarlo, entregarlo, operarlo y mantenerlo.

Un ciclo de vida permite clasificar y controlar las diferentes actividades necesarias para el desarrollo y mantenimiento del producto.

MODELO DE LAS FASES DEL CICLO DE VIDA.–

Este modelo divide el ciclo de vida del producto en una serie de actividades sucesivas; cada fase requiere información de entrada, procesos y resultados. Las fases son las siguientes: análisis, diseño, instrumentación, pruebas y mantenimiento.

El análisis consta de dos subfases: planeación y definición de requisitos. La planeación incluye la comprensión del problema del cliente, estudio de factibilidad, desarrollo de la estrategia de solución recomendada, determinación de los criterios de aceptación, y planeación del proceso de desarrollo. De la planeación surgen la *Definición del Sistema* y el *Plan de Proyecto*.

La definición de requisitos se refiere a la identificación de las funciones básicas del componente de programación en un sistema de equipo/personal/programación. Se pone atención en las funciones y restricciones bajo las cuales se deben desarrollar. De la definición de requisitos se obtiene una especificación que describe el ambiente de procesamiento, las funciones requeridas de los programas, restricciones de configuración sobre los programas, manejo de excepciones, cambios y modificaciones factibles, así como los criterios de aceptación de producto de programación.

El diseño de la programación viene después del análisis. El diseño se refiere a la identificación de los componentes de la programación, especificando las relaciones entre ellos, la estructura de la programación, y manteniendo un registro de las decisiones, proporcionando un documento base para la instrumentación. El diseño se divide en estructural y detallado.

El diseño estructural comprende la identificación de los componentes de la programación, su desacoplamiento y descomposición en módulos de procesamiento y estructuras de datos conceptuales y la interconexión entre componentes. El diseño detallado se refiere a cómo empacar los módulos de procesamiento, y cómo instrumentar los algoritmos, las estructuras de datos y sus interconexiones.

En la fase de instrumentación incluye la traducción de las especificaciones del diseño en código fuente, así como su depuración, documentación y pruebas.

Las pruebas del sistema comprenden dos tipos de actividades: pruebas de integración y de aceptación. El desarrollo de una estrategia para integrar los componentes de un sistema de programación requiere una planeación cuidadosa. Las pruebas de aceptación se relacionan con la planeación y ejecución de varios tipos de pruebas para demostrar que el sistema satisface las necesidades establecidas.

Una vez que el sistema ha sido aceptado por el cliente, se entrega para su operación y se inicia la fase de mantenimiento. Las actividades de esta última fase incluyen mejoras de las capacidades, adaptación a nuevos ambientes de procesamiento, y corrección de fallas del sistema.

MODELO DEL COSTO DE UN PROYECTO.–

El costo de un proyecto es la suma de los costos incurridos en cada fase, y éstos, a su vez, incluyen los costos

de realización de los procesos y preparación de los documentos de esa fase, más los costos de verificación de la consistencia de estos productos con los de las fases previas.

El costo de análisis incluye la *Definición del Sistema* y el *Plan de Proyecto*, así como la definición de requisitos.

El costo del diseño es el costo de las actividades propias y la generación de documentos, más el costo de modificar y corregir el análisis y el costo de verificación del diseño contra los requisitos.

El costo de instrumentación es el costo de la conversión, documentación, depuración y pruebas del código fuente, más el costo de la terminación del Manual del Usuario, el plan de verificación, los procedimientos de mantenimiento, y las instrucciones de instalación y entrenamiento, más el costo de modificar y corregir el análisis y el diseño.

El costo de las pruebas del sistema incluyen el costo de planear y llevar a cabo las pruebas, más el costo de las modificaciones al código fuente y a los documentos externos durante ellas, más el costo de verificación de que las pruebas validan adecuadamente al producto.

Por último, el costo del mantenimiento es la suma de los costos de llevar a cabo mejoras al sistema, hacer adaptaciones para nuevas condiciones de operación y encontrar fallas.

MODELO DE PROTOTIPO PARA EL CICLO DE VIDA.–

Este modelo muestra las fuentes de requisitos para el producto, puntos decisivos para continuar o detenerse, y el uso de prototipos. Un prototipo es una representación o un modelo del producto de programación que incorpora componentes del producto real. Por lo regular, un prototipo tiene un funcionamiento limitado en cuanto a capacidades, confiabilidad o eficiencia.

Entre las razones para desarrollar un prototipo se encuentran ilustrar los formatos de entrada de datos, mensajes, informes y diálogos al cliente. También sirve para explorar aspectos técnicos del producto propuesto.

Un prototipo es útil cuando se no se tienen las especificaciones correctas al inicio del ciclo de vida, y por lo mismo, no se cuenta con un modelo apropiado de análisis–diseño–implementación. Una vez que se instrumenta y evalúa una versión del producto, se aclara el proceder hacia la mejora del sistema.

VERSIONES SUCESIVAS.–

Este método es una extensión del método de prototipos en el que se refina un esqueleto inicial del producto, obteniendo así más capacidades.

Mediante este método, las distintas versiones del producto se diseñan antes de cualquier actividad de instrumentación. En este caso, las características de cada diseño sucesivo serán planeadas durante la fase de análisis.

Antes de poner en marcha una versión, la instrumentación puede indicar que se necesita un mayor análisis y diseño.

PLANEACIÓN EN LA ESTRUCTURA ORGANIZACIONAL

Existe una estructura de actividades durante el tiempo de vida de un producto de software: planeación, desarrollo, servicios, publicaciones, control de calidad, apoyo y mantenimiento. Esta estructura puede ser

modificada u organizada de diferente forma.

Los métodos para organizar estas tareas pueden ser los formatos de proyecto, el funcional y el matricial.

ESTRUCTURA DEL PROYECTO.–

+ Formato de Proyecto:

Este tipo de formato implica la integración de un equipo de programadores que lleven a cabo el proyecto de principio a fin y que realicen la definición, diseño, instrumentación, prueba y revisiones del producto, así como el desarrollo de los documentos de apoyo.

Algunos miembros del equipo pueden permanecer durante la instalación y el mantenimiento, mientras otros pueden ir a nuevos proyectos sin dejar la responsabilidad del mantenimiento.

Al terminar un proyecto, se le asigna uno nuevo a los miembros del equipo.

+ Formato Funcional:

En este esquema un equipo distinto de programadores realiza cada fase del proyecto, y los productos, pasan de un equipo a otro conforme el producto va evolucionando.

Una variación común del formato funcional comprende tres equipos: uno de análisis, otro de diseño e instrumentación, y un tercero de pruebas y mantenimiento. En este esquema un grupo de apoyo realiza las publicaciones, mantiene las instalaciones y proporciona el entrenamiento. Los miembros de los equipos se rotan periódicamente para proporcionar desarrollo profesional y evitar el tedio de la superespecialización.

El formato funcional requiere de más comunicación entre equipos, pero también permite que el personal se especialice en ciertas áreas y que la documentación sea más clara a causa de las necesidades de comunicación.

+ Formato Matricial:

En organizaciones matriciales, cada función de las descritas tiene su propia administración y un equipo de gente dedicada exclusivamente a dicha función.

Cada proceso de desarrollo tiene un administrador, que organizacionalmente es miembro de la planeación o del desarrollo, y que genera y revisa los documentos y puede participar en el diseño, instrumentación y pruebas del producto.

Cada grupo funcional participa en todo proyecto, por lo que cada uno de los grupos trabaja en uno o más proyectos bajo la supervisión del administrador correspondiente.

ESTRUCTURA DEL GRUPO DE PROGRAMACIÓN.–

Todo equipo de programación debe tener una estructura interna, la óptima estructura de un proyecto depende de su naturaleza y la del producto. Las estructuras básicas son el grupo democrático, en el que todos los miembros participan en todas las decisiones; el grupo con jefe de programadores, en el que otros miembros del equipo apoyan y auxilian al jefe; por último, el grupo jerárquico que combina aspectos de los dos anteriores. Pueden existir variaciones de las tres estructuras.

+ Grupo Democrático:

Las metas y decisiones se definen por consenso. Los productos (requisitos, diseño, código fuente, manual del usuario, etc.) se discuten abiertamente y son examinados con libertad por todos los miembros.

Se nombra un líder que ocupa la primera posición entre los demás. El liderazgo no suele rotar porque el funcionamiento mejora cuando un solo individuo se responsabiliza de la coordinación de las actividades y de tomar las decisiones finales en situaciones en las que

no se logra consenso.

Las ventajas de un grupo de este tipo incluyen la oportunidad de todos los miembros de contribuir en las decisiones, aprender uno de otro, y la satisfacción que produce trabajar en un ambiente bien comunicado y sin presiones.

Las desventajas de esta estructura son la cantidad de comunicación necesaria para tomar decisiones, el requisito de que todos los miembros trabajen juntos y la falta de autoridad y responsabilidad que puede ocurrir, lo que producirá menos iniciativa.

+ *Grupo con Jefe de Programación:*

Estos grupos son muy estructurados. El jefe diseña un producto, instrumenta partes críticas de él, y toma las decisiones técnicas importantes; también asigna el trabajo de los programadores, y éstos, escriben el código, lo depuran, documentan y prueban.

Un bibliotecario de programas mantiene en un lugar accesible los listados, documentos de diseño, planes de prueba, etc. El programador de apoyo sirve como consultor del jefe en problemas técnicos; establece la relación con el cliente, el grupo de publicaciones y el de control de calidad, y puede realizar algún análisis, diseño e instrumentación bajo la supervisión del jefe.

El jefe es auxiliado por un gerente administrativo que se encarga de los detalles.

Las ventajas de esta estructura son las decisiones centralizadas y la reducción en las trayectorias de comunicación; sin embargo, su eficacia es muy sensible a las capacidades técnicas y administrativas del jefe, lo cual puede producir desmoralización en los programadores subordinados.

+ *Grupo Bajo Jerarquía Administrativa:*

Esta estructura ocupa un punto intermedio entre los grupos democráticos y de jefe. En un grupo jerárquico, el líder del proyecto asigna tareas, asiste a revisiones y recorridos, detecta áreas de problemas, balancea las cargas de trabajo y participa en actividades técnicas.

Esta estructura limita las trayectorias de comunicación en un proyecto, permitiendo comunicación real cuando se requiere; es particularmente útil para el desarrollo de productos de programación con jerarquía, puesto que cada subsistema se puede asignar a un equipo distinto.

Si un producto jerárquico va a tener tres subsistemas, es razonable tener tres equipos cada uno jerárquico, y que el líder de cada uno informe al líder del proyecto.

Una desventaja importante de esta estructura es que los programadores técnicamente son competentes, tienden a ser promovidos hacia posiciones administrativas. Por desgracia, la promoción del mejor programador puede tener un doble efecto negativo: perder un buen programador y crear un mal administrador.

ADMINISTRACIÓN POR OBJETIVOS.—

Al inicio de un proyecto, cada miembro del mismo debe escribir la descripción de su participación basándose en su percepción del proyecto; estas descripciones se revisan con el líder del proyecto y se modifican en su caso, de manera que para todo el equipo queden claras sus responsabilidades y participación.

La administración por objetivos es una técnica que está popularizándose en las organizaciones que desarrollan productos de programación; en su uso, los empleados establecen sus propias metas con la ayuda del supervisor, y participan en el establecimiento de los objetivos del supervisor; así pueden evaluarse logros concretos y explícitos.

La administración por objetivos se puede aplicar a todos los niveles de la organización y puede considerar tanto objetivos relacionados con productos, como terminación del diseño o de las pruebas, u objetivos personales como capacitación o adquisición de nuevas habilidades. Tanto los objetivos relacionados con el trabajo como los personales, se acuerdan entre el supervisor y el subordinado, se formalizan y se evalúan al final de un periodo acordado.

La administración por objetivos recibe críticas acerca de su complicación y burocracia. Algunos problemas comunes en su uso son el establecimiento de demasiados objetivos y la utilización de periodos de evaluación muy largos.

Los objetivos para los proyectos de programación deben ser específicos, pocos en número y factibles en un lapso breve; además deben redactarse de modo que su logro pueda definirse con claridad.

CONCLUSIONES

Cuando se realiza un proyecto de programación, es muy importante la fase de planeación. Se necesita una planeación cuidadosa del proceso de desarrollo y del producto final.

Todo proyecto de desarrollo debe tener una estructura organizacional, una de equipos, y un mecanismo para asignar y evaluar el trabajo.

Es relevante el hecho de que existan varios modelos de ciclo de vida, ya que se puede utilizar el más adecuado, dependiendo de la naturaleza del trabajo que se está desarrollando.

Es importante tomar en cuenta las actividades adicionales como la planeación de la administración de la configuración, control de calidad, validación y verificación externas, así como herramientas y técnicas que dependen de cada fase.

BIBLIOGRAFÍA

FAIRLEY Richard. Ingeniería de Software.

Ed. McGraw Hill/Interamericana de México,

México, 1988,

390 pp.