Capítulo 4: Roles en el desarrollo de software

Versión 1.3

4.1 Introducción

El desarrollo de software es una actividad que, dada su complejidad, debe desarrollarse en grupo. Además, esta actividad requiere de distintas capacidades, las que no se encuentran todas en una sola persona. Por ello, se hace necesario formar el grupo de desarrollo con las personas que cubran todas las capacidades requeridas. Cada una de esas personas aportará al grupo parte del total de las capacidades necesarias para llevar a cabo con éxito el desarrollo.

Por ello, es que cada persona debe tener un rol dentro del grupo, que viene dado por su experiencia y capacidades personales. En este capítulo se describen los roles que tradicionalmente se consideran en el desarrollo de software. Estos roles son administrador de proyecto, analista, diseñador, programador, téster, asegurador de calidad, documentador, ingeniero de manutención, ingeniero de validación y verificación, administrador de la configuración y por último, el cliente. Para cada uno de estos roles, se definen sus objetivos, actividades, interacción con otros roles, herramientas a utilizar, perfil de las personas en ese rol y un plan de trabajo.

Hay que señalar que es posible que no se requieran todos los roles en un desarrollo. Eso dependerá del tamaño y del tipo del desarrollo. Por ejemplo, el desarrollo de un sistema de información de gran tamaño requerirá más roles que uno de menor tamaño. Por otro lado, si el tipo del proyecto está enfocado más hacia la parametrización e integración de sistemas, requerirá algunos roles en menor medida y otros en mayor.

Es posible también que una persona realice las labores de más de un rol al mismo tiempo. Esto, sobre todo en proyectos de desarrollo de software más pequeños. No obstante, es imprescindible que dichas personas conozcan completamente todas sus tareas.

Por otro lado, también es posible la situación de tener más de una persona con un mismo rol en un equipo de desarrollo. Por ejemplo, en sistemas de complejidad mediana hemos utilizado con éxito la fórmula de tener un administrador de proyecto, dos analistas, dos diseñadores, dos programadores y un téster. Eso hace un total de ocho personas en un grupo. Las actividades de documentación y administración de configuración son asumidas por todos los roles. Parte de las actividades de aseguramiento de calidad son asumidas por el téster. El resto de las actividades no son realizadas.

El hecho de que en un grupo de desarrollo no se tengan claro los roles y sus responsabilidades y actividades asociadas, hace que se produzcan problemas. Por un lado, es posible que una o más actividades no estén asociadas a ningún rol, con lo que el proyecto sufrirá. Por otro lado, es posible que una o más actividades estén asociadas a más de un rol. Esto último producirá problemas entre los miembros afectados, lo que también redunda en problemas en el desarrollo del sistema. Por lo anterior, se hace necesario que cada miembro conozca muy bien su rol dentro del proyecto, así como las responsabilidades y actividades asignadas. Eso es lo que se intenta describir en este capítulo.

Es bastante común que, frente a una oferta de una empresa en busca de personal calificado para su equipo de desarrollo de software, nos sintamos atraídos a postular a un rol específico. Por ello, al final del capítulo se entrega, adicionalmente, algunas recomendaciones básicas para postular al cargo que se desee.

4.2 La fábula de la granja

Un día cualquiera, los animales de una granja decidieron hacer una fiesta, con el propósito de pasar un momento agradable. Para organizar la fiesta, se reunieron el mismo día en la mañana. Cada animal debía llevar algo a la fiesta. Como es lógico, a la vaca le pidieron la leche. A la gallina, le tocó llevar los huevos. Y al chancho, el tocino.

En este caso, la vaca y la gallina participan de la fiesta. Sin embargo, el chancho se encuentra involucrado. Su participación le obliga a entregar parte de si mismo como aporte para la fiesta. Al chancho le toca aportar una cuota de sacrificio mayor. Lo anterior muestra la diferencia entre participar en un evento y estar involucrado.

Tomemos esta fábula para caracterizar a los miembros del grupo de un desarrollo de software. ¿Cómo se comportan, en general? ¿Participan o están comprometidos en el proceso de desarrollo de software?

Parece claro que lo deseable, desde el punto de vista del problema completo, es tener integrantes comprometidos. Pero, ¿cómo se obtienen estos miembros comprometidos? ¿Es posible "crear" miembros del grupo comprometidos? ¿Administrador de proyecto comprometido, analista comprometido, diseñador comprometido, programador comprometido, téster comprometido, asegurador de calidad comprometido, documentador comprometido, ingeniero de manutención comprometido, ingeniero de validación y verificación comprometido, administrador de la configuración comprometido y cliente comprometido?

La fábula anterior nos enseña la diferencia entre participar y estar comprometidos en una actividad. Es claro que para tener miembros del equipo de desarrollo comprometidos, es necesario capacitarlos en sus deberes y derechos en el ciclo de vida del desarrollo de software.

Es muy poco probable que un miembro no capacitado pueda estar comprometido con los objetivos del proyecto. Este presentará claras deficiencias en el momento de participar en el proceso. Como ejemplo, se mencionan algunas:

1. Un miembro no capacitado no entenderá el lenguaje técnico utilizado por el resto de los miembros. Muchas veces, entenderá una cosa diferente a la expresada por sus pares. Esto es común debido a diferencias en el lenguaje.
2. Un miembro no capacitado, no conoce el ciclo de vida del desarrollo, ni los problemas que se presentan durante el desarrollo. Sería muy bueno que el miembro pudiera aportar sus conocimientos en su dominio del problema durante todo el ciclo de desarrollo del proyecto. Saber cuando exigir y cuando ceder. Conocer los estándares de desarrollo, de documentación, de aseguramiento de la calidad.
3. Un miembro no capacitado no presupuesta su involucración en el proyecto, sólo su participación. Este solo hecho reduce las posibilidades de éxito del proyecto. También aumenta los tiempos de desarrollo, disminuye la calidad del sistema, aumentan los riesgos de rechazo del sistema por parte de la comunidad de clientes, etc.

Lo anterior sugiere que es necesario contar con “miembros comprometidos” para desarrollar correctamente el proyecto. Aspectos a considerar son los de crear un lenguaje común entre el equipo de desarrollo, así como hacer que entiendan claramente sus deberes y obligaciones.

Por otro lado, los clientes también deben estar comprometidos con el desarrollo. Eso significa que deben conocer el ciclo de vida escogido, cual es su participación en cada una de las fases del ciclo, y la cantidad de esfuerzo y recursos que se espera que pongan en cada momento del proyecto. Es de vital importancia que participen activamente en la etapa de análisis, así como en todas aquellas actividades de revisión y aceptación.

En caso que el cliente no tenga dicha experiencia, se hace necesario que antes de un desarrollo, se los capacite para convertirlos en clientes comprometidos. Lo anterior requiere de trabajo, pero va en la senda correcta del éxito de un proyecto de software.

Es claro entonces que todos los integrantes del equipo de desarrollo debiesen estar comprometidos con el proyecto, incluyendo los clientes. Lo anterior implica trabajar con el equipo completo en torno a las metas a lograr, así como las cualidades y características deseables de cada uno de ellos. Para ello, se requiere entender correctamente las características de liderazgo dentro de un grupo humano.

4.3 Administrador de proyecto

El administrador de proyecto es la persona que administra y controla los recursos asignados a un proyecto, con el propósito de que se cumplan correctamente los planes

definidos. Los recursos asignados pueden ser recursos humanos, económicos, tecnológicos, espacio físico, etc. En un proyecto, siempre debe existir un administrador. No obstante, un administrador puede dirigir más de un proyecto.

El administrador no es dueño de nada, es sólo un administrador temporal de los recursos. Como no es dueño de nada, debe dejarlos en la misma o mejor condición de cómo los recibió. Por ello, el foco de una buena administración debe estar en el control y coordinación de los diferentes eventos y actividades de un proyecto. Adicionalmente, deben crearse las mejores condiciones posibles para que se realicen las actividades.

Una de las preocupaciones principales para los administradores debe ser el tener una visión y misión claras del proyecto, trabajando para que ambas se cumplan. En otras palabras, el foco de un administrador de proyecto debe estar en el bosque más que en los árboles. Sin embargo, debe cuidar cada árbol ya que cada uno de ellos contribuye al bosque.

El rol de administrador de proyecto es un rol muy importante, debido a que sus acciones y decisiones afectan al proyecto completo.

Objetivos

Algunos de los objetivos de un administrador de proyecto son los siguientes:

1. Tener el producto “a tiempo”, “bajo presupuesto” y con los requisitos de calidad definidos.

2. Terminar el proyecto con los recursos asignados.

3. Coordinar los esfuerzos generales del proyecto, ayudando a cada uno de sus integrantes a cumplir sus objetivos particulares. Al final, se cumplirá el objetivo general.

4. Cumplir con éxito las diferentes fases de un proyecto, utilizando herramientas de administración.

5. Cumplir con las expectativas del cliente.

Algunos de los objetivos específicos a cumplir por un administrador de proyecto son los siguientes:

1. Comenzar y terminar cada actividad de acuerdo a lo planificado.

2. Lograr el mejor uso de los recursos disponibles.

3. Observar cada actividad para detectar y resolver inconvenientes.

Actividades y metas

A continuación se muestra el conjunto de actividades a realizar por el administrador de proyecto para lograr cumplir con éxito las metas definidas.

Actividades	Metas
Desarrollo eficiente de reuniones: El administrador de proyecto está a cargo de las reuniones y presentaciones entre el grupo y con los clientes. Debe conducir la reunión usando el protocolo establecido. Debe cuidar de todos los aspectos de la reunión (layout de la sala, luces, sillas, mesas, computadores, proyección y sonido, pizarra, lápices, material que se entrega, etc.). Mida el tiempo y compárelo con lo planificado, para así maximizar la eficiencia. Cada reunión debe ser evaluada. De ser necesario, deben tomarse acciones correctivas. Promueva reuniones de integración (desayunos, comidas, coffee breaks, etc.), donde el grupo pueda intercambiar puntos de vista y experiencias, creatividad y espontaneidad.	Las reuniones logran sus objetivos. Los puntos principales de cada reunión deben ser documentados.
Desarrollo organizacional: Conduzca reuniones y seminarios cuando el grupo determina los próximos puntos: • Visión. • Misión. • Metas. • Objetivos. • Actividades.	Asigne las actividades y el proyecto en un contexto amplio y con sentido para el grupo de trabajo. Ayude al equipo en su desarrollo organizacional.
Administración: Entregar un plan de trabajo general basado en diagramas Gantt y de flujo de actividades, apoyado con el plan de trabajo de cada rol. El plan de trabajo general deberá contener estimaciones de horas-hombre de cada actividad, que permita estimar los recursos humanos requeridos. Desarrollar el contrato junto con el cliente. Realizar actividades de organización, dirección y control. Trabajar con los analistas para estudiar las necesidades de los clientes y los requisitos del sistema.	Coordine todas las actividades.

En general: • Realizar reuniones generales y seminarios de evaluación y planificación. • Realizar reuniones de evaluación con cada rol. • Obtener información sobre el estado el proyecto para el equipo y para el cliente.	

Relación con otros roles

El administrador de proyecto debe relacionarse con todo el equipo de trabajo. Para ello, debe darle apoyo con lo siguiente:

- Una carta de organización de todo el proyecto.
- Un plan de trabajo general.
- Estimaciones de horas-hombre de cada actividad.

El administrador deberá tener una comunicación fluida con cada miembro del equipo para analizar problemas particulares, y si es necesario, tomar acciones correctivas. En particular, el administrador de proyecto deberá apoyar de la siguiente forma a cada rol:

- Analistas: Trabajar con los analistas para estudiar las necesidades de los clientes y los requisitos del sistema.
- Diseñadores: Trabajar con ellos para diseñar la arquitectura del sistema de acuerdo con los recursos asignados al proyecto. El administrador de proyecto requiere la arquitectura del sistema para determinar el plan de trabajo de los demás roles.
- Ténsters: Trabajar con ellos para determinar que tipo de testeo deberá utilizarse, y con que profundidad, de acuerdo con los requisitos de seguridad en el diseño del sistema y de los recursos disponibles. Los resultados de los tests ayudan a determinar el éxito del proyecto, preocupación principal de la administración de proyecto.
- Aseguradores de calidad: La información provista por este rol ayuda a conocer el avance del proyecto. Este rol observa si cada una de las actividades se realiza de acuerdo a las especificaciones planificadas.
- Ingenieros de manutención: Generalmente la manutención utiliza una cantidad muy importante de recursos del proyecto. Por ello, el administrador debe conocer los planes de manutención, y de ser necesario, ajustarlos a los recursos disponibles.

- Documentadores: El administrador de proyecto tomará como referencia los documentos controlados por los documentadores para elaborar planes y la evaluación del proyecto.

- Clientes: El administrador de proyecto deberá administrar la relación con los clientes, desarrollando una comunicación fluida con éstos, y siendo la cara visible del proyecto.

Herramientas de trabajo

El administrador de proyecto deberá utilizar herramientas que apoyen su trabajo. Algunas de estas herramientas se describen a continuación:

- Herramientas de comunicación, tales como teléfono fijo y móvil, que permita ser localizado y localizar rápidamente a otros miembros del equipo. El correo electrónico y grupos de discusión también son herramientas de uso frecuente.

- Herramientas de administración de proyectos, que permitan definir y modificar diagramas Gantt y de flujo de actividades. Ejemplos de esto son MS Project 2000 y MS Project Central, y Primavera.

- Herramientas que permitan contabilizar el uso de recursos, tal como una planilla de cálculos. Ejemplo de esto es MS Excel.

- Herramientas de presentación, tal como MS Power Point.

- Procesador de texto, tal como MS Word.

- Grabadora de audio.

- Grabadora de video.

Perfil de un administrador de proyecto

El administrador de proyecto deberá tener, al menos, las siguientes capacidades personales para desarrollar adecuadamente su trabajo:

- Abstracción: Entender y comunicar aspectos no tangibles, como visión y misión del equipo de trabajo. Deberá además, poder entender y ver el proyecto completo como una unidad y sus relaciones entre sus partes.

- Concretización: Utilizando los recursos e información disponibles, obtener conclusiones y tomar acciones específicas para manejar el proyecto.

- Organización: Distribuir eventos y actividades de acuerdo a los recursos y tiempos disponibles para llevar el proyecto al éxito.

- Liderazgo: Llevar a un equipo a lograr sus objetivos.

- Experiencia: Haber estado en situaciones similares en el pasado.

- Creatividad: Ser realista, tomando decisiones y tomando acciones cuando el plan actual no funciona.

- Persuasión: Encontrar y desarrollar argumentos para mejorar y ayudar en una situación.

Además, el administrador de proyecto deberá poseer las siguientes habilidades:

- Escuchar y comunicar.

- Tomar decisiones y realizar acciones.

- Trabajar bajo presión.

Plan de trabajo

A continuación se mencionan algunas de las actividades a realizar por el administrador de proyecto, que forman parte de su plan de trabajo.

- Definir y establecer estándares a seguir por el grupo.

- Definir una estructura organizacional y hacer un diagrama organizacional.

- Capacitar al grupo en las metodologías y estándares a utilizar.

- Crear un modelo de ciclo de vida para el proyecto.

- Definir un plan y protocolo para desarrollo de reuniones.

- Definir una agenda de reuniones con cada rol.

- Construir un plan de trabajo específico que contenga diagramas Gantt y de flujo de actividades.

- Definir protocolos para asignar y evaluar actividades. Nótese que durante el proyecto, será necesario redefinir tareas, y con ello, miembros del equipo deberán alterar su carga de trabajo para realizarlas.

- Realizar estimación de horas-hombre por actividad y por persona.
- Realizar reuniones generales para evaluación y planificación.
- Realizar un contrato con el cliente que defina las características y condiciones en que se desarrollará el producto.

4.4 Analistas

La palabra “análisis” se refiere a una característica típicamente relacionada con la inteligencia humana. Esta se refiere a la habilidad de poder estudiar un problema de una complejidad determinada, descomponiendo el problema en subproblemas de menor complejidad. De esa forma, la solución del problema completo se obtiene como la suma de las soluciones de los subproblemas de menor complejidad.

Lo anterior indica que la fase de análisis en un proyecto de construcción de software se refiere a la especificación de un problema como la suma de subproblemas de menor complejidad. Como el experto en el problema es el cliente, se hace necesario trabajar junto a él para realizar la especificación correctamente. Los miembros del grupo que trabajan con el cliente para realizar el análisis y especificación del sistema a construir son precisamente los analistas.

Para que el trabajo de los analistas tenga sentido para todos los integrantes del grupo, se hace necesario ponerse de acuerdo en la forma como se realizará la especificación, así como la forma como el resto del grupo la entenderá. Esto sugiere el uso de un estándar para realizar la fase de análisis del problema. En el caso del estándar de la ESA, el análisis se divide en dos fases: especificación de requisitos de usuario y especificación de requisitos de software. Los analistas deben liderar ambas fases.

Una de las razones más frecuentes del fracaso de un desarrollo de software es la de realizar un análisis pobre. Debido al insuficiente esfuerzo dedicado a conocer y especificar el sistema que desea el cliente, los desarrolladores construyen sistemas que no cuentan con las características que el cliente desea. Ese error se repite una y otra vez, y se debe básicamente a la inexperiencia del grupo de desarrolladores.

Actividades y metas

A continuación se detallan un conjunto de actividades para cada uno de las metas a lograr por los analistas.

Actividades	Metas
Entrevistar al cliente, ayudándole a identificar sus necesidades.	Determinar las necesidades esenciales y no esenciales, así como las que son de segundo nivel.
Verificar si los requisitos especificados son los correctos.	Impedir la introducción de defectos tempranamente en la construcción del sistema.

Definir una estructura básica del sistema que incluya fuentes de información, módulos de procesamiento de información, y resultados esperados.	Construir el documento de requisitos de usuarios.
Realizar el análisis de los requisitos.	Establecer una estructura básica inicial del sistema.
Analizar la estructura básica del sistema.	Establecer interacciones, interrelaciones y sus contextos en dicha estructura.
Generar los diagramas de la arquitectura.	Definir la especificación de la arquitectura del sistema, en forma de un documento técnico comprensible.

En la fase de análisis de requisitos de usuario, los analistas deben identificar las necesidades del cliente, a través de reuniones con el cliente o su representante. En estas reuniones, los analistas deben ayudar al cliente a definir los objetivos del sistema, determinando la información que desea obtener, la información que será suministrada al sistema, las funcionalidad del sistema y el rendimiento requerido. Los analistas deben determinar si cada uno de los requisitos especificados es o no esencial. Luego los analistas deben determinar información adicional requerida, tales como la evaluación de tecnología disponible para el desarrollo y las tecnologías disponibles para el cliente. Debe considerar todos los recursos especiales requeridos, las estimaciones del cliente y sus tiempos límites, así como factores adicionales que puedan ser de interés.

Luego, los analistas deben realizar la especificación de requisitos de software. Esto es, no como una especificación en lenguaje del cliente, sino que como especificación para el equipo de trabajo.

El rol de analista es muy importante, debido a que el éxito del proyecto dependerá de una buena especificación de requisitos. Es claro que los errores detectados en las fases de análisis son más fáciles de detectar y corregir que en fases más avanzadas del desarrollo. Una buena arquitectura debe ser robusta y flexible. Una falla en la estructura puede dar origen al colapso del proyecto en forma parcial o total. Por lo anterior, se hace indispensable realizar una buena detección de las necesidades del cliente y el establecimiento de una buena estructura del sistema desde el principio.

Metodologías de análisis

El analista debe estructurar y especificar el problema del cliente, por lo que se espera que mantengan un contacto estrecho. Durante el período de análisis, el analista se reunirá en forma sistemática con el cliente con el propósito de entender y especificar el problema a desarrollar. Dichas reuniones deben estar planificadas, con fecha de inicio y fecha de término. En cada reunión, los analistas le mostrarán al cliente, como ha ido evolucionando el documento de requisitos de usuario, DRU. El analista deberá asegurarse de que el cliente entiende los conceptos ahí especificados. El analista podrá utilizar prototipos, encuestas, otros sistemas, etc., con el propósito de ayudar a estructurar y definir el problema del cliente. El proceso termina con el proceso de revisión de los requisitos de usuario RU/R, y luego, el hito de aceptación del documento de requisitos de usuario, DRU.

El analista debe además, transformar los requisitos de usuario en requisitos de software, y producir el documento de requisitos de software, DRS. La intervención del cliente en esta etapa es menor, y su trabajo consistirá en resolver los conflictos detectados en los requisitos de usuario por el analista durante el proceso de transformación. El proceso de especificación de los requisitos de software produce el documento de requisitos de software, DRS. Luego, se realiza una revisión formal RS/R y termina con el hito de aprobación del DRS.

Relación con otros roles

El rol de analista debe interactuar con los demás roles en el grupo. A continuación se mencionan algunas de las interacciones.

- Administrador de proyecto: El analista debe interactuar con el administrador de proyecto para estudiar la viabilidad del sistema a desarrollar. Esto es, verificar la realización del sistema con los recursos disponibles. El administrador de proyecto le asignará a los analistas, la agenda con actividades a ser realizadas y sus fechas. Es claro que la asignación de actividades puede ir modificándose durante el proyecto.

- Diseñador: Los diseñadores deben interactuar con los analistas para determinar la factibilidad del proyecto, y establecer los objetivos del sistema para un buen diseño. Los analistas deben permanecer en contacto estrecho con los diseñadores debido a que utilizarán la arquitectura del sistema. Los diseñadores deben poder ayudarle a los analistas.

- Programador: Los analistas son apoyados por los programadores en el entendimiento y especificación de los requisitos de usuario y de software. Además, los apoyan en la construcción de prototipos rápidos.

- Téster: Los analistas participan junto con los ténsters en la revisión de los documentos de análisis de requisitos.

- Asegurador de calidad: Debe revisar los documentos hechos por los analistas.

- Administrador de la configuración: Debe pedir los cambios pertinentes, evitando errores a futuro.

- Documentador: Los analistas deberán entregarles la información que servirá para la documentación del sistema.

Herramientas de apoyo

Resulta innecesario enumerar las herramientas disponibles para apoyar las fases de análisis, debido a que hay muy pocas, las que no son de mucha utilidad. Esto, debido

a que es muy difícil tener una herramienta que detecte las necesidades del cliente. Este trabajo está más bien relacionado con el criterio de los analistas. Estas herramientas administran los requisitos, sus propiedades, y sus cambios.

Por otro lado, existen herramientas que apoyan a los analistas en tareas administrativas y en el manejo de reuniones. Algunos ejemplos de esto son:

- Proyectores de diapositivas.

- Videograbadoras.

- Videocámaras, para grabar las reuniones para análisis posterior.

- Grabadoras de audio.

También debe considerarse herramientas para la creación de documentos, tales como procesador de texto, software para el diseño y dibujo de diagramas, e incluso, sistemas CASE para realizar la estructura general del sistema.

Perfil de un analista

Un analista es una persona con capacidades de comunicación, debido a que deberá tener un contacto estrecho con el cliente. Por lo mismo anterior, debe ser una persona sociable, expresando sus ideas en forma clara en un lenguaje común con el cliente. También debe tener la capacidad de escuchar y entender al cliente. Se espera que los analistas tengan un alto grado de desarrollo de su inteligencia emocional.

Los analistas deben conocer y manejar perfectamente los métodos y las tecnologías de apoyo para realizar las fases de análisis. Además, se espera creatividad, lo que le permitirá establecer diferentes alternativas de modelos para la arquitectura del sistema a construir.

También es importante que los analistas estén muy familiarizados con las técnicas de diseño que se utilizarán en las siguientes fases. Además, se hace necesario que esté familiarizado con los diferentes lenguajes de programación para ayudar a escoger el apropiado para la construcción del sistema.

Plan de trabajo

A continuación se menciona algunas de las actividades a realizar por los analistas, las que forman parte de su plan de trabajo.

- Preparar un documento con preguntas a realizar al cliente durante las entrevistas.

- Determinar las fechas de reunión con el cliente.

- Generar un documento de especificación de requisitos de usuario en base a los acuerdos alcanzados en la primera reunión.
- Presentación del documento de especificación al cliente en la siguiente reunión.
- De ser necesario, realizar las modificaciones al documento de especificación de requisitos de usuario y presentarlas al cliente en la próxima reunión. Repetir esta actividad las veces que sea necesario.
- Estudiar la metodología de diseño.
- Explorar las herramientas CASE a utilizar.
- Generar los diagramas de arquitectura.
- Revisar los diagramas de arquitectura con los diseñadores.
- De ser necesario, realizar las modificaciones a los diagramas.
- Presentar los diagramas de arquitectura finales.
- Construir el documento de requisitos de software.
- Revisar el documento con los ingenieros de aseguramiento de la calidad y cliente, realizando modificaciones de ser necesario.

4.5 Diseñadores

Es el encargado de generar el diseño del sistema. Entre sus funciones está:

- Generar el diseño arquitectónico y diseño detallado del sistema, basándose en los requisitos.
- Generar prototipos rápidos del sistema (con analistas y programadores) para chequear los requisitos.
- Generar el documento de diseño arquitectónico de software (DDA), y mantenerlo actualizado durante el proyecto.
- Velar porque el producto final se ajuste al diseño realizado (funciones de tester).

En cada disciplina de la ingeniería, el diseño acompaña el enfoque disciplinado que se utiliza para inventar la solución de un problema, entregando así un camino entre los

requisitos y la implementación. En ingeniería de software, el propósito del diseño es la construcción de un sistema que cumpla con los siguientes aspectos:

- Satisfaga una especificación funcional dada.
- Cumpla con las limitaciones del medio receptor del sistema.
- Cumpla requisitos implícitos y explícitos de rendimiento y uso de recursos.
- Satisfaga criterios de diseño implícitos y explícitos en la forma del artefacto construido.
- Satisfaga restricciones del mismo proceso de diseño, tal como su duración y costo, o las herramientas disponibles para realizar el diseño.

Objetivos

El propósito del diseño es el de crear una estructura interna limpia y relativamente simple, también llamada a veces una arquitectura. Un diseño es el producto final del proceso de diseño. Así, una de las metas en el diseño de software es derivar una arquitectura del sistema. Esta arquitectura sirve como un marco desde el cual se conducen más actividades de diseño detallado.

Las buenas arquitecturas de software tienden a tener algunos atributos comunes. Entre ellos se pueden mencionar los siguientes:

- Se encuentran construidos en niveles de abstracción bien definidos, provistos a través de una interfaz bien definida y controlada, y construida sobre facilidades igualmente bien definidas y controladas en niveles de abstracción inferiores.
- Existe una separación clara de preocupaciones entre la interfaz y la implementación de cada nivel, haciendo posible cambiar la implementación de un nivel sin violar las suposiciones que hicieron los clientes.
- La arquitectura es simple, comportamiento común se obtiene a través de abstracciones y mecanismos comunes.

Actividades y metas

A continuación se describe las actividades y metas a considerar por un diseñador.

Actividades	Metas
Descomposición de subsistemas	Crear una estructura interna del sistema, llamada una arquitectura y la definición de relaciones entre subsistemas.
Definir la administración de acceso a recursos globales	Seleccionar las políticas apropiadas para nombres lógicos, espacio, unidades físicas, y acceso a datos compartidos.
Seleccionar una técnica de administración	Seleccionar el método de almacenamiento apropiado para

de almacenamiento de datos	las estructuras de datos, por ejemplo, estructuras de datos vs. sistemas de archivos vs. SABD.
Interactuar con los programadores	Seleccionar el lenguaje y paradigma apropiado
Asignación de subsistemas a procesadores	Asignar procesos a unidades de procesamiento que sirva como plataforma para la ejecución de subsistemas.
Administración de la concurrencia	Identificar los casos donde la ejecución del sistema incluya múltiples hebras de control.
Selección de estrategias de control	Determina el método apropiado para las líneas de control de ejecución, por ejemplo, procedural vs manejado por eventos vs concurrente.
Administración de condiciones de borde	Asegurarse que los módulos operan apropiadamente en los bordes, establecidos para limitar o restringir procesos, por ejemplo, inicializaciones, terminación, y fallas.

Para evaluar la calidad de la representación del diseño, es necesario establecer criterios técnicos para un buen diseño. A continuación se presenta una lista de criterios que pueden utilizarse.

- Un diseño debe contener una organización jerárquica que haga un uso inteligente del control entre los elementos del software.
- Un diseño debe ser modular. En otras palabras, el sistema debe estar particionado lógicamente en elementos que realizan funciones y subfunciones específicas.
- Un diseño debe contener abstracciones de datos y abstracciones procedurales.
- Un diseño debe conducir a módulos (esto es, subrutinas o procedimientos), que muestren características funcionales independientes.
- Un diseño debe considerar interfaces que reduzcan la complejidad de conexiones entre módulos y con el ambiente externo.
- Un diseño debiese ser construido usando un método repetible, guiado por la información obtenida durante la fase de requisitos de software.

Metodologías de diseño

Existen muchas metodologías de diseño. Describiremos brevemente sólo una clase de ellas, llamado métodos de descomposición. El diseño de un sistema de software implica la descomposición del sistema en partes de menor tamaño, cada una de las cuales puede refinarse en forma independiente. Dentro de los métodos de descomposición, mencionaremos los siguientes dos:

- Descomposición algorítmica: Corresponde al proceso de dividir el sistema en partes, cada una de las cuales representa un pequeño paso de un proceso más grande. La aplicación de métodos de diseño estructurado llevan a una descomposición algorítmica, cuyo foco está puesto en el control de flujo del sistema.
- Descomposición orientada a objetos: Corresponde al proceso de dividir el sistema en partes, cada una de las cuales representa una clase u objeto del dominio del problema. La aplicación de métodos orientados a objetos llevan a descomposición orientada a objetos, en la cual se observa al mundo como una colección de objetos que cooperan entre ellos para obtener la funcionalidad deseada.

  El proceso de diseño orientado a objetos considera el proceso de descomposición orientada a objetos, así como una notación para representar los modelos lógico y físico del sistema en diseño. También se incluyen los modelos estático y dinámico del sistema. Específicamente, la notación incluye diagramas de clases, diagramas de objetos, diagramas de módulos y diagramas de procesos.

Relación con otros roles

Los diseñadores deben relacionarse con otros miembros del equipo de desarrollo. A continuación se describe algunas de las interacciones más relevantes:

- Administrador de proyecto: Los diseñadores trabajan bajo la coordinación del administrador de proyecto para construir la arquitectura del sistema que cumpla con los requisitos bajo restricciones dadas de presupuesto y disponibilidad de recursos humanos. Adicionalmente, el administrador de proyecto utiliza las especificaciones de diseño para planificación y estimación de recursos.
- Analista: Los diseñadores traducen la especificación de requisitos establecida en la fase de análisis de requisitos de software en un modelo de implementación. Dentro de las tareas, los diseñadores deben interactuar con los analistas para determinar requisitos ambiguos del proyecto. Usualmente, los analistas apoyan a los diseñadores, y vice-versa.
- Programador: Los diseñadores crean la especificación de la implementación del sistema para los programadores. Dicho modelo es traducido a código ejecutable por el computador. Los diseñadores apoyan a los programadores en la selección del lenguaje de programación, así como a la interpretación de los documentos de diseño tales como diagramas, cartas, tablas, etc.
- Tester: Los diseñadores deben coordinar esfuerzos con los testers para asegurar que el diseño arquitectónico del sistema de software incluye especificaciones que ayuden en el ejercicio de casos de testeo. Además, debe apoyar a los testers en la verificación de requisitos.

- Asegurador de calidad: Los aseguradores de calidad deben revisar la fase de diseño para asegurarse que el proceso de diseño sigue las normas de calidad especificadas, y cumple con requisitos de rendimiento, diseño y verificación.

- Ingeniero de validación y verificación: Los ingenieros de validación y verificación evalúan el nivel de concordancia entre los requisitos de usuario y el modelo del sistema diseñado, buscando desentendimientos, así como características faltantes o erróneamente implementadas. La relación con los diseñadores es de apoyo.

- Administrador de configuración: Durante el diseño, el administrador de la configuración de software controla los cambios en el diseño y mantiene registros completos de cada cambio y de sus razones.

- Ingeniero de manutención: Los diseñadores deben apoyar al ingeniero de manutención en administrar la evolución de post-venta. Esta evolución incluye arreglo de errores, mejoramiento de la funcionalidad del sistema, y modificación de requisitos.

- Documentador: Los documentadores mantienen los documentos de diseño una vez que el proceso de diseño es completado, haciendo disponibles dichos documentos al resto del equipo de trabajo.

Herramientas de apoyo

Perfil de un diseñador

El perfil de un diseñador debe incluir las siguientes características:

- Para sistemas de tamaño pequeño y mediano, el diseño arquitectónico es realizado por una o dos personas calificadas. Deben mostrar habilidad inusual para sintetizar soluciones construibles por sobre un gran conjunto de restricciones.

- Generalmente son los más capacitados para realizar decisiones estratégicas debido a su experiencia previa en la construcción de sistemas similares.

- No son necesariamente los desarrolladores con más experiencia.

- Deben tener habilidades de programación adecuadas.

- Deben conocer muy bien la metodología de diseño utilizada, así como sus herramientas de apoyo.

Plan de trabajo

El plan de trabajo de los diseñadores incluye las siguientes actividades para el diseño del sistema.

- Organizar el sistema en subsistemas.

- Identificar concurrencias inherentes al problema.

- Asignar los subsistemas a procesos y tareas.

- Seleccionar un administrador de datos.

- Identificar recursos globales y determinar mecanismos de acceso.

- Elegir un enfoque para implementar el control de la ejecución.

- Considerar condiciones de borde.

4.6 Programadores

Los programadores deben convertir la especificación del sistema en código fuente ejecutable utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación.

El éxito del desarrollo de software depende grandemente de conocimiento. Este conocimiento no sólo corresponde a habilidades de programación y de administración de proyectos, sino que a una percepción y entendimiento de los últimos desarrollos de la industria del software. En los mercados actuales, rápidamente cambiantes y altamente competitivos, se hace necesario conocer los últimos desarrollos, quien da soporte, y como pueden beneficiar al proyecto y a la organización. A través de este conocimiento es que la organización genera un camino hacia el éxito futuro.

Objetivos

Uno de los principales objetivos de los programadores durante su trabajo debe ser la de reducir la complejidad del software. Algunos de los beneficios que implican la reducción de la complejidad del programa son:

- Menor cantidad de problemas de testeo.

- Aumento de la productividad de los programadores.

- Aumento de la eficiencia en la manutención del programa.

- Aumento de la eficiencia en la modificación del programa.

Adicionalmente, otros objetivos importantes son:

- Reducir el tiempo de codificación, aumentando la productividad del programador.
- Disminuir el número de errores que ocurren durante el proceso de desarrollo.
- Disminuir el esfuerzo de corregir errores en secciones del código que se encuentran deficientes, remplazando secciones cuando se descubren técnicas más confiables, funcionales o eficientes.
- Disminuir los costos del ciclo de vida del software.

Para alcanzar estos objetivos, es importante escoger las herramientas de desarrollo apropiadas. De eso dependerá en parte poder alcanzar los objetivos, y por lo tanto, el éxito del proyecto.

Es claro que para elegir las herramientas adecuadas, es necesario conocer el ambiente donde el software va a correr.

Actividades y metas

A continuación se especifican algunas de las actividades y metas más relevantes de alcanzar por los programadores.

Actividades	Metas
Explorar los diferentes ambientes en que el sistema puede ser desarrollado	Determinar los lenguajes posibles de usar e identificar las posibles herramientas de desarrollo
Interactuar con los analistas y diseñadores	Seleccionar el ambiente apropiado
Explorar los diferentes lenguajes disponibles para el ambiente seleccionado	Seleccionar el lenguaje apropiado
Interactuar con los diseñadores	Seleccionar el lenguaje apropiado y lenguaje de programación
Explorar diferentes herramientas de desarrollo (compiladores, depuradores, etc.) disponibles para el lenguaje seleccionado	Seleccionar la herramienta de desarrollo apropiada
Explorar los distintos estilos de codificación que pueden ser utilizados en el lenguaje seleccionado	Escoger un estilo de codificación
Realizar la codificación del sistema	Entregar el código ejecutable de acuerdo a las fechas presupuestadas
Interactuar con los ingenieros de testeo	Determinar las formas de realizar el testeo
Apoyar al ingeniero de testeo	Realizar las actividades de testeo en forma rápida, eficiente, sistemática, exhaustiva y confiable, entregando un código utilizable y seguro
Reunirse con otros miembros del equipo de programadores	Conocer el estatus de las actividades de programación, apoyando a sus colegas en caso de requerirlo
Realizar revisiones personales	Mantener el código eficiente y adaptable para ser unido con el código de otros programadores

Interactuar con el administrador de la configuración	Mantener al día el control de la configuración
Realizar los cambios solicitados al código	Mantener el software ejecutable eficiente
Hacer la documentación del código	Entregar la documentación técnica del código fuente

El lenguaje de programación escogido afecta significativamente los costos, confiabilidad y rendimiento del sistema. Ningún lenguaje es ideal para todas las aplicaciones. La elección del lenguaje debe estar basada en la naturaleza de las aplicaciones (tiempo-real, incrustada, procesamiento batch, basado en Web, etc.) y en la importancia de algunos indicadores de calidad (rendimiento versus confiabilidad). La base de datos también debe ser confiable y proteger privacidad e información comercial de usuarios no autorizados.

Los estilos de codificación incluyen los nombres de variables, la forma de hacer comentarios en el código fuente, el diseño y escritura de rutinas y módulos, la creación de tipos de datos, la selección y control de estructuras y la organización de bloques de instrucciones. A veces, algunos de estos factores son determinados por la sintaxis y el paradigma de programación utilizados, existiendo estándares para lo anterior. Típicamente, un grupo de programadores, o una empresa, utiliza el mismo estándar con el propósito de que todos los programadores puedan acceder fácilmente a todo el código.

Las modificaciones hechas al código deben ser solicitadas por el administrador de configuración. Otras veces, las modificaciones son solicitadas por el ingeniero de testeo directamente al programador. En algunos casos, algunas modificaciones no requieren que se completen los formularios de cambio debido a que los cambios solicitados no son relevantes o no requieren aprobación del comité de cambios (por ejemplo, no modifican los requisitos de usuarios). Es importante destacar que los programadores no deben realizar cambios al software solicitados directamente por el cliente.

Las reuniones realizadas entre programadores son muy importantes. En ellas se mantiene al día al equipo sobre el estatus de la codificación, los problemas que tienen las otras personas, la forma en que otras personas abordaron una tarea específica, nuevas necesidades internas y cambios al código, y hacer revisiones al código de otros programadores.

Metodología

Existen diferentes metodologías para realizar las actividades de programación. Sin embargo, todas ellas muestran un patrón común. Este patrón consiste en la exploración de herramientas y lenguajes, la determinación del estilo de programación, el desarrollo de herramientas utilitarias y rutinas comunes para administrar la entrada, salida y errores, la codificación y depuración del sistema, la escritura de la documentación técnica, y para todo el equipo de programadores, realizar revisiones personales periódicas y reuniones.

En la exploración de las herramientas y lenguajes, se debe considerar el tipo de aplicación y su naturaleza, con el fin de determinar el lenguaje apropiado. Para realizar

esta selección, se debe considerar la metodología utilizada en las actividades de diseño así como el paradigma de programación del lenguaje.

El desarrollo de las herramientas utilitarias y rutinas comunes debe realizarse antes de la entrega del documento que contiene el diseño arquitectónico del sistema. Estos utilitarios ayudan al programador a realizar la codificación más rápida debido a que puede focalizarse sólo en la codificación de los factores especificados en el documento de diseño en vez de ocupar tiempo en la codificación de otras rutinas que son necesarias en todo programa. En estas funciones utilitarias, el programador debe construir algunas rutinas que considere necesarias, dependiendo del tipo de aplicación. Algunas herramientas de desarrollo han integrado herramientas que contienen rutinas comunes para cada programa, así como rutinas para ambientes específicos. Este factor debe ser considerado en la selección del lenguaje y herramientas de desarrollo.

Un factor muy importante en la construcción de un sistema es el paradigma de programación utilizado. Uno de los paradigmas muy utilizados es el orientado a objetos, el cual tiene varias ventajas sobre otras metodologías de programación.

Así mismo, existen varias técnicas de diseño y análisis orientado a objetos. Dicha elección es vital para la elección del lenguaje de programación. Por ello, si la metodología de diseño utilizada es orientada al objeto, se sugiere utilizar un lenguaje orientado a objetos.

En un equipo de programación, la revisión personal consiste en inspeccionar el código escrito por otro programador, con el propósito de evaluarlo. La evaluación incluye buscar errores de diseño, programación, estilo y documentación. De esa forma, es posible mantener el código en forma más eficiente durante las actividades de programación, disminuyendo los posibles problemas durante las actividades de integración.

Las reuniones que se realizan son de carácter informativo para determinar el estatus de la codificación.

Relación con otros roles

Los programadores deben relacionarse con otros miembros del grupo del proyecto. Dentro de éstos, se encuentran los siguientes:

- Administrador de proyecto: El programador debe entregar un reporte con los resultados de las actividades de programación cuando el administrador lo solicite. Debe además. Ayudarle al administrador en la estimación de tiempos y costos de las actividades de programación.

- Analista: Deben interactuar con los analistas para determinar el ambiente apropiado para el sistema.

- Diseñador: El rol de programador depende mucho del rol de diseñador, debido a que debe utilizar herramientas adaptadas a la metodología utilizada en las actividades de diseño. El diseñador también le ayuda al programador a seleccionar el lenguaje de programación adecuado.

- Téster: El programador debe interactuar con el téster para determinar una forma apropiada de construir los tests y de testear los programas. El programador debe estar presente durante el testeo de código, cuando situaciones no esperadas suceden o es necesario realizar pequeñas modificaciones al código.

- Administrador de configuración: El programador debe entregar la última versión del diseño al administrador de configuración. El programador debe pedir la última versión del diseño al administrador de configuración, debiendo atender los diferentes pedidos de cambio del código. El programador puede solicitar cambios en otras partes del sistema a través del administrador de la configuración. La petición se realiza llenando el formulario correspondiente y enviándoselo al administrador de configuración.

- Ingeniero de manutención: El programador tiene mucha influencia en el rol de manutención, debido a que si el código está claro, será fácil de mantener. Dependiendo de las metodologías y herramientas empleadas, será más fácil o más difícil mantener los sistemas.

- Asegurador de calidad: El asegurador de calidad debe verificar la calidad del sistema construido. El programador deberá entregarle su plan de trabajo al asegurador de calidad.

- Documentador: El programador debe proveer la documentación técnica del código al documentador.

Herramientas de apoyo

Existen muchas herramientas para ayudar al programador a desarrollar el sistema. Éstas consisten principalmente en ambientes de desarrollo integrados adaptados para un lenguaje de programación. Estos ambientes pueden compilar y ejecutar un programa usando comandos para ello. La mayoría de estos ambientes también proveen herramientas para depurar el programa. Algunos ambientes también proveen herramientas de scheduling.

Perfil de un programador

El perfil del programador requiere conocimiento en varios ambientes, pudiendo ayudarle a los analistas y diseñadores a elegir el apropiado. Debe tener experiencia en el desarrollo de aplicaciones en el ambiente seleccionado.

Debe conocer diferentes lenguajes de programación disponibles para el ambiente seleccionado, y debe tener experiencia en el lenguaje de programación seleccionado. Las herramientas utilitarias desarrolladas en proyectos previos pueden ser útiles en el proyecto actual. Es preferible que el programador tenga conocimientos en diferentes paradigmas de programación y estilos.

Debe además, conocer perfectamente las técnicas de diseño utilizadas por el diseñador. También es deseable que el programador tenga conocimiento en varias metodologías de diseño.

Las bases de datos son una herramienta muy poderosa en un proyecto. Los programadores deben tener experiencia en bases de datos. De ser posible, es preferible que los programadores tengan experiencia en el tipo de proyecto que se desea realizar.

Plan de trabajo

El plan de trabajo de los programadores debe contener, al menos, las siguientes actividades:

- Explorar los diferentes ambientes de desarrollo.
- Explorar los diferentes lenguajes disponibles para el ambiente.
- Explorar las diferentes herramientas de desarrollo (compiladores, bases de datos, depuradores, etc.) disponibles para el lenguaje seleccionado.
- Explorar sistemas ya construidos de los cuales, el nuevo sistema será parte.
- Elegir el estilo de programación.
- Programar las herramientas utilitarias y rutinas comunes.
- Codificar y depurar.
- Testear.
- Realizar revisiones personales y reuniones.
- Escribir la documentación técnica.

4.7 Téster

El desarrollo de un sistema de software requiere la realización de una serie de actividades de producción. En dichas actividades existe la posibilidad de que

aparezcan errores humanos. Dichos errores pueden empezar a aparecer desde el primer momento del proceso. Por ejemplo, los requisitos del sistema pueden ser especificados en forma errónea o imperfecta. Por ello, el desarrollo de software considera una actividad que apoye el proceso de detección y eliminación de los errores y defectos del sistema en construcción. El objetivo del rol de téster es precisamente realizar dichas tareas.

El téster es el encargado de asegurar la calidad de cada uno de los productos (documentos, prototipos, etc). Entre sus tareas están:

- Construir y aplicar los planes de prueba unitarios, de módulo, de sistema, y aceptación parcial, manteniéndolos actualizados durante el proyecto.
- Velar por la completitud, y exactitud (no ambigüedades) de todos los documentos del proyecto.
- Coordinar las inspecciones, y/o caminatas.
- Velar por la adhesión al estándar adoptado para el desarrollo.
- Velar por la calidad del producto final (cumplimiento de los requisitos).

Objetivos

El objetivo principal de la labor de téster es el de diseñar tests que en forma sistemática, permita eliminar diferentes clases de errores, realizando esto con la mínima cantidad de tiempo y esfuerzo.

Los objetivos específicos en la labor de un téster son los siguientes:

- Aplicar métodos para diseñar casos de tests efectivos.
- Construir buenos casos de tests que tengan altas probabilidades de encontrar errores aún no descubiertos.
- Demostrar que las funciones del sistema parecen estar funcionando de acuerdo a sus especificaciones.
- Proveer una buena indicación de la confiabilidad del software y algunas indicaciones de la calidad del software.

Actividades y metas

Las actividades y metas a cumplir por los ténsters se describen a continuación.

Actividades	Metas

Participación en el proceso de especificación del sistema	Prevenir errores en las etapas tempranas del desarrollo
Interacción con el diseñador	Realizar tests al diseño, obteniendo índices de medición
Realizar los tests, apoyado por los programadores	Realizar diferentes tests, obtener una buena interpretación de ellos, y realizar los ajustes pertinentes
Informar sobre los resultados obtenidos	El grupo de desarrollo es informado sobre los progresos y resultados obtenidos

Metodología

Los ténsters deben utilizar una metodología que en forma sistemática, organizada y estructurada, les permita detectar y corregir, los errores y defectos introducidos en el proceso de desarrollo del sistema. Típicamente, se utilizan dos técnicas para ello: el test de la caja blanca y el test de la caja negra.

Los tests de caja blanca corresponden a un método de diseño de casos de tests que utilizan la estructura de control del diseño procedural para derivar los casos de tests. Usando este método, el téster puede derivar casos de tests para lo siguiente:

- Asegurarse que todos las trayectorias independientes en un módulo han sido visitadas al menos una vez.
- Ejercitar todas las decisiones lógicas en sus lados verdadero y falso.
- Ejecutar todos los loops en sus límites y sobre sus límites operacionales.
- Ejercitar estructuras de datos internas para asegurar su validez.

Por otro lado, los tests de caja negra se focalizan en los requisitos de usuario del sistema. De esa forma, este tipo de tests permite que los ténsters generen conjuntos de datos de entrada que ejercitarán completamente los requisitos del sistema. Los tests de caja negra no son una alternativa a los tests de caja blanca. Más aún, corresponde a un enfoque complementario que posiblemente descubra una clase diferente de errores.

Los tests de caja negra tratan de encontrar errores en las siguientes categorías:

- Funciones incorrectas o faltantes.
- Errores de interfaces.
- Errores en estructuras de datos o acceso a bases de datos externas.
- Errores de rendimiento del sistema y errores de inicialización y terminación.

Los tests de caja blanca son realizados tempranamente en el ciclo de vida del desarrollo. Los tests de caja negra se utilizan en etapas más tardías del ciclo. Debido a que la metodología de caja negra ignora las estructuras de control del programa, la atención se focaliza en el dominio de la información.

Relación con otros roles

Los distintos miembros del grupo de trabajo deben relacionarse con los ténsters. En cada rol, la actividad de téster juega una parte importante. A continuación se menciona algunas actividades relacionadas con otros roles:

- Analista: Participar en la revisión de los documentos de requisitos de usuario y de software.
- Diseñador: Coordinarse con el grupo de diseñadores para garantizar que el diseño arquitectónico del producto de software incluye las especificaciones que facilitan el ejercicio de los casos de tests. Además, debe coordinarse con los diseñadores en la verificación de los requisitos. Por último, debe participar en las revisiones técnicas del diseño.
- Programador: El téster debe trabajar con el programador para realizar las siguientes actividades: revisión de código; elección del mejor tipo de tests para aplicar al código; tests de los métodos; tests de integración; tests de regresión.
- Validación y Verificación: El téster debe coordinarse con este rol en la ejecución de los diferentes casos de tests, de acuerdo con las necesidades del cliente.
- Administrador de configuración: El administrador de la configuración debe proveerle al téster de la última versión de documentos desarrollados por los otros roles (analista, diseñador, programador).

Perfil de un téster

El perfil de un téster debe considerar las siguientes características:

- Ser un buen programador en el lenguaje seleccionado, y tener experiencia en el desarrollo de sistemas.
- Conocer bien la metodología de diseño utilizada.
- Ser sistemático en las revisiones de código y resultados de los tests.
- Tener una personalidad agresiva para buscar errores en el código y documentos del proyecto.

- Debe además tener una personalidad alegre, debido a que debe relacionarse con gran parte de los miembros del equipo de desarrollo.

Plan de trabajo

El plan de trabajo del téster debe incluir, al menos, las siguientes actividades:

- Participar en la revisión de los requisitos del sistema.
- Construir un plan de testeo.
- Coordinarse con los diseñadores para incluir el test del diseño en el documento.
- Ejecutar los tests de bajo nivel.
- Ejecutar los tests de mediano nivel.
- Ejecutar los tests de alto nivel.
- Construir la documentación del proceso de tests.

4.8 Aseguradores de calidad

En la actualidad, los factores dominantes en la administración de proyectos de software son los tiempos y costos de desarrollo. Existen buenas razones para ello. Los tiempos y costos de desarrollo son con frecuencia, muy grandes. Por ello, la administración se ha concentrado en tratar de resolver dichos problemas. Sin embargo, existe un gran peligro en esto. En la medida que crece la presión por cumplir con las fechas estipuladas, y reducir los costos, es la calidad del producto la que sufre. Cuando se acelera el desarrollo de un sistema que está atrasado, generalmente se corta todo lo que no se considere "esencial", usualmente cortando las actividades de verificación y testeo, resultando en un producto de calidad reducida.

Se hace necesario encontrar una nueva ecuación para el desarrollo de software. No debe ser simplemente "producto de software = a tiempo + dentro de los costos". Debiese ser "producto de software = calidad + a tiempo + dentro de los costos". Para ello, debe existir el convencimiento individual y de la gerencia de considerar la calidad como una meta final, junto con el cumplimiento de plazos y costos. Como se mencionó antes, la calidad corresponde a un conjunto de atributos a cumplir por el desarrollador. Típicamente, dichos atributos se encuentran definidos en la forma de un estándar, el que debe cumplirse.

Actividades y metas

A continuación se presentan las actividades y metas a cumplir por los aseguradores de calidad.

Actividades	Metas
Revisar los documentos de requisitos de usuario y de software	Asegurarse que la especificación de requisitos es una representación correcta y completa de las expectativas del cliente, y que es suficientemente clara para el equipo de desarrollo, especialmente para los diseñadores.
Revisar el plan de administración del proyecto	Asegurarse que el plan es creado y se cumple.
Revisar el plan de testeo	Asegurarse que el plan se crea, que es adecuado al proyecto específico, y que se sigue en cada fase del ciclo hasta que se entrega el producto.
Revisar la fase de diseño arquitectónico	Asegurarse que los diseñadores seleccionaron la metodología apropiada y que el producto final cumple con los requisitos de rendimiento, diseño y verificación.
Revisar la fase de diseño detallado	Asegurarse que el software producido cumple con los requisitos especificados y con los atributos de calidad impuestos.
Revisar las políticas de control de cambios, control de errores y control de la configuración	Asegurarse que se realizan monitoreos de errores en cada fase del desarrollo y que se respaldan las líneas bases haciendo que el producto no se pueda perder.
Revisar la documentación	Asegurarse que la documentación cumple con el estándar utilizado durante el desarrollo del producto de software.

Metodología

De entre las actividades del Asegurador de Calidad, la más importante es la de participar en las revisiones técnicas formales (RTF). Si estas revisiones están bien conducidas, son la forma más efectiva de encontrar, revelar y corregir errores mientras aún es barato encontrarlos y arreglarlos.

El estándar ESA incluye revisiones en las fases RU/R, RS/R, DA/R y DD/R. No obstante, las RTFs son especialmente requeridas en la fase de diseño arquitectónico. Esto, debido a que las actividades de diseño introducen entre el 50 y 65% de todos los errores durante el proceso de desarrollo. Se ha demostrado que las RTFs descubren del orden del 75% de los errores de diseño.

Los objetivos de las RTFs son: 1. descubrir errores en funciones, lógica e implementación en cualquiera de las representaciones del software; 2. verificar que el software bajo revisión cumple con los requisitos; 3. asegurarse que el software ha sido representado de acuerdo al estándar en uso; 4. alcanzar software que es desarrollado en forma uniforme; y 5. hacer el proyecto más manejable.

Una RTF es una reunión entre tres a cinco personas. Cada una de ellas ha realizado una preparación de antemano de no más de dos horas, y su duración no debe tampoco sobrepasar las dos horas. La RTF se focaliza en un producto pequeño del software, tal como una porción de los requisitos, el diseño detallado de un módulo, o el listado de código fuente de un módulo.

Los participantes de una RTF son el productor (la persona que desarrolló el producto a revisar), un encargado de la revisión que evalúa el producto genera copias de material,

y lo distribuye a dos o tres revisores para que se preparen de antemano. Uno de los revisores toma el rol de documentador de los aspectos más relevantes aparecidos durante la revisión.

Al final de la revisión, los participantes deben decidir si: 1. aceptar el producto sin modificación posterior; 2. rechazar el producto debido a errores serios; o 3. aceptar el producto con errores menores que deben ser corregidos, pero no se requiere una revisión posterior.

Relación con otros roles

A continuación se analiza la relación del asegurador de calidad con los otros roles:

- Administrador de proyecto: El asegurador de calidad revisa el plan de administración de proyecto, para asegurarse que se crea y que se sigue.

- Analista: El asegurador de calidad revisa la especificación de requisitos de usuario y de software, para asegurarse que es una representación correcta y completa de las expectativas del cliente, y que es suficientemente clara para todos en el grupo de desarrollo, especialmente para el diseñador.

- Diseñador: El asegurador de calidad revisa la fase de diseño arquitectónico, para asegurarse que el diseñador seleccionó la metodología apropiada y que el producto final de esta fase cumple con requisitos de rendimiento, diseño y verificación.

- Programador: El asegurador de calidad revisa la fase de diseño detallado, para asegurarse que el código producido cumple con la especificación de requisitos establecida y que cumple con los atributos de calidad en uso.

- Téster: El asegurador de calidad revisa el plan de testeo, para asegurarse que es creado, que es adecuado para el proyecto específico, y que se aplica en cada fase del proceso de desarrollo hasta la entrega del producto.

- Documentador: El asegurador de calidad revisa la documentación, para asegurarse que corresponde con el software desarrollado, y que cumple con el estándar en uso.

- Administrador de configuración: El asegurador de calidad revisa los registros de cambios, errores y de configuración, para asegurarse de que los cambios han sido implementados apropiadamente, y que las líneas bases son almacenadas y que el producto no se puede perder.

Perfil de un asegurador de calidad

El asegurador de calidad debe ser una persona con mucha experiencia en proyectos de desarrollo de software, con conocimientos suficientes sobre técnicas que aseguren la calidad de un producto de software. Lo anterior lo hace capaz de negociar con la calidad del producto, y ocasionalmente, modificar el criterio de los desarrolladores.

Plan de trabajo

El plan de trabajo de un asegurador de calidad es extenso, ya que debe estar involucrado en todas las fases del desarrollo del software.

4.9 Administrador de configuración

La administración de la configuración es una disciplina que tradicionalmente se aplica al desarrollo de sistemas de hardware, al desarrollo de elementos de hardware o sistemas de hardware/software. La administración de la configuración de software corresponde a la administración de la configuración aplicada a un sistema, o a partes de un sistema, predominantemente correspondiente a software. Su aplicación, en conjunto con otras disciplinas, lleva al desarrollo de sistemas en forma ordenada y estructurada.

La administración de la configuración es una disciplina que aplica dirección y vigilancia técnica y administrativa a:

- Identificar y documentar las características funcionales y físicas de items de configuración.

- Auditar los items de configuración para verificar cumplimiento de especificaciones, control de interfaces y documentos, así como otros requisitos adicionales que pueda definir el contrato.

- Controlar cambios a los items de configuración y su documentación relacionada.

- Registrar y reportar información necesaria para administrar items de configuración en forma efectiva, incluyendo el estatus de cambios propuestos y el estatus de implementación de cambios aprobados.

- Mantener el repositorio del proyecto actualizado con las últimas versiones de todos los entregables del proyecto.

- Administrar el software utilizado para el control de versiones.

- Definir y controlar perfiles de acceso a los archivos del proyecto.

- Velar por la completitud y exactitud del repositorio del proyecto.

Los problemas de software más frustrantes son frecuentemente ocasionados por una pobre administración de la configuración. Los problemas son frustrantes debido a que requieren tiempo para arreglarlos, y usualmente ocurren en el peor momento, y son totalmente innecesarios.

La administración de la configuración ayuda a reducir estos problemas coordinando los productos de muchas personas que trabajan en un proyecto común. Sin ese control, su trabajo va a producir conflictos con frecuencia, resultando en problemas como los descritos a continuación:

- Modificaciones simultáneas. Cuando dos o más personas trabajan separadamente en el mismo programa o documento, el último en realizar los cambios puede fácilmente destruir el trabajo del otro.

- Código común. En grandes sistemas, cuando se modifican funciones comunes de un programa, es necesario notificarlo a todos los miembros del grupo. Sin una administración de código efectiva, no hay forma de estar seguro de encontrar y alertar a cada uno de los miembros del equipo.

- Versiones. Muchos de los grandes programas son desarrollados en releases evolucionarios. Con uno siendo utilizado por el usuario, otro en testeo, y un tercer en desarrollo, los arreglos de errores deben ser propagados entre ellos. En sistemas de gran tamaño con varios releases activos en forma simultánea y muchos programadores trabajando en arreglo de errores y mejoras, los conflictos y confusión son bastante frecuentes.

Estos problemas conducen a confusión y pérdida de control, y pueden hacer perder una gran cantidad de tiempo. La clave es tener un sistema de control que conteste a las siguientes preguntas:

- ¿Cuál es mi configuración de software actual?

- ¿Cuál es mi estatus?

- ¿Cómo controlo cambios en mi configuración?

- ¿Cómo le informo al resto de mis cambios?

- ¿Qué cambios han sido hechos a mi software?

- ¿Los cambios realizados por otros afectan mi software?

La administración de configuración de software, es una disciplina que identifica la configuración de un sistema en puntos discretos en el tiempo, con el propósito de controlar sistemáticamente los cambios a esa configuración, manteniendo integridad y trazabilidad de la configuración a través del ciclo de vida del sistema. Para entender los

elementos involucrados en la disciplina de la administración de la configuración de software, es necesario formular un concepto de software. Digamos que software es información que es:

- estructurada con propiedades lógicas y funcionales;

- creada y mantenida en varias formas y representaciones durante su ciclo; y

- ajustada para procesamiento de máquina en su estado completamente desarrollado.

El hecho de que, en general, el software es fácil de modificar tiene implicaciones importantes para la administración de la configuración de software. Esta susceptibilidad al cambio, esta plasticidad. Es la razón primaria para disciplinar el proceso de desarrollo de software. Así, el software puede existir en dos formas básicas:

- Una forma no ejecutable: documentación tal como la especificación de funciones a ejecutar, cartas de flujo que diagraman la lógica de las funciones a ejecutar, etc.

- Una forma ejecutable: consiste en secuencias ejecutables de información directamente accesibles por una suite dada de maquinaria computacional.

Para los propósitos de la administración de la configuración de software, estas dos formas de software son equivalentes. La primera forma es necesaria para describir el software, facilitando la comunicación del usuario con el computador. La segunda forma es necesaria para ejecutar el software en un computador.

Los elementos que componen la administración de configuración de software son:

- Identificación de la configuración. Corresponde a una disciplina para identificar la configuración de un ítem, documentando sus características funcionales y físicas.

- Auditoria de configuración. Provee los mecanismos para determinar una línea base formalmente establecida.

- Control de configuración. Es la ejercitación de procedimientos establecidos para clasificar, aprobar o reprobar, liberar, implementar y confirmar cambios aprobados a especificaciones y líneas base.

- Contabilidad del estatus de configuración. Contabilidad de configuración es el registro y reporte de datos relacionados con la identificación de la configuración, estatus de aprobación de cambios propuestos y estatus de implementación de cambios aprobados durante todas las fases del proyecto.

Objetivos

El objetivo principal de la administración de configuración de software es la administración efectiva del ciclo de vida del sistema de software y la evolución de su configuración. En otras palabras, corresponde al establecimiento y manutención de los productos de software del proyecto a través del ciclo de vida del software. La administración de configuración no es una tarea independiente, existe para apoyar el desarrollo y manutención del producto de software. Es necesario utilizar el criterio al aplicar técnicas de administración de configuración. Por un lado, muy poca administración de la configuración y es posible perder productos, necesitando trabajo adicional para rehacerlos. Por otro lado, mucha administración de la configuración y la organización nunca producirá un producto, debido a que estará muy ocupada moviendo papeles.

Actividades y metas

A continuación se muestran parte de las actividades y metas del administrador de configuración.

Actividades	Metas
Preparar el Plan de Administración de la Configuración de Software de acuerdo al estándar en uso	Se planifican las actividades de administración de la configuración de software.
Las tareas iniciales son identificar las líneas base que serán usadas en el proyecto, y los items que serán parte de cada línea base. Este proceso se llama identificación de la configuración. En el sentido de la administración de la configuración, una línea base es un documento, o un conjunto de ellos, formalmente designados y fijados en el tiempo. Por ello, una línea base es un repositorio oficial del producto, y contiene la versión más reciente.	Se identifican las características funcionales y físicas de los items de configuración.
Para alcanzar esta meta, el control de la configuración de software provee el mecanismo administrativo para precipitar, preparar, evaluar y aprobar o reprobar el procesamiento de propuestas de cambio. Incluye 3 actividades básicas: • Establecer y diseñar la documentación requerida para formalmente precipitar y definir un cambio propuesto a un sistema de software (Propuesta de Cambio de Ingeniería, PCI). • Formar un cuerpo organizacional para formalmente evaluar y aprobar o reprobar un cambio propuesto a un sistema de software (Cuerpo de Control de Cambio, CCC). • Realizar los procesos para controlar cambios a un sistema de software.	Se controlan los cambios.
Realizar RTFs y/o auditorias de configuración de software. Éstas son los medios para asegurar que el cambio se implementó correctamente.	Asegurarse que los cambios se implementan apropiadamente.
Registrar y reportar información requerida para administrar items de configuración eficientemente, incluyendo el estatus de los cambios propuestos y el estatus de implementación de los cambios aprobados.	Los grupos afectados y personas son informados del estatus y contenido de las líneas bases.
Mantener un registro de cómo evolucionó el sistema y donde está el sistema en cualquier instante respecto a su línea base y acuerdos escritos. Esta actividad es la Contabilidad del Estatus de la Configuración de Software, y provee los medios para seguir la historia del ciclo de vida del sistema de software.	Verificar cual es la configuración de software actual y cual es su estatus.
Auditar los items de configuración. Auditar la configuración de software provee los mecanismos para determinar el grado en el cual el estado actual del sistema de software refleja el sistema de software dibujado en la línea base y documentación de requisitos. También provee los mecanismos para establecer formalmente una línea base.	Verificar cumplimiento de especificaciones, documentos de control de interfaces, y otros requisitos de contratos.

Metodología

La administración de configuración de software es una actividad paraguas que es aplicada a través del proceso de ingeniería de software completo. Dichas actividades son desarrolladas para: identificar el cambio, controlar el cambio, asegurarse que el cambio se implementó adecuadamente, y reportar cambios a personas que puedan estar interesadas. Los items que componen toda la información producida como parte del proceso de ingeniería de software se le llama en forma conjunta, configuración de software. Los cambios pueden ocurrir en cualquier momento, y por cualquier razón.

Relación con otros roles

El administrador de la configuración de software se relaciona con todos los integrantes de su equipo de una o más de las siguientes formas:

- Si los roles producen un ítem de configuración de software que ha sido identificado y puesto en el repositorio de administración de la configuración de software.
- Si pertenecen al Cuerpo de Control de Cambios.
- Si solicitan un cambio de ítem de la configuración de software.
- Si la persona debe implementar un cambio.

Si uno de los eventos anteriores ocurre, estas personas deben realizar las actividades correspondientes de acuerdo al Plan de Administración de Configuración de Software. Además, deben saber que la administración de la configuración de software es un punto central en cualquiera de esas actividades.

Específicamente, la relación con cada uno de los roles es:

- Administrador de proyecto: Su plan será parte del repositorio de administración de la configuración de software. Además, él es parte del CCC.

- Analista: Los ítems de configuración de software producidos por este rol son el DRU y el DRS.

- Diseñador: Los ítems de configuración de software producidos por este rol son el DDA y el DDD. Por lo tanto, el diseñador es parte del CCC.

- Programador: Los ítems de configuración de software producidos por este rol son parte del DDD y el código del sistema (fuente y ejecutable). Este rol es parte del CCC.

- Téster: El único ítem de configuración de software producido por este rol es el Plan de Testeo.

- Asegurador de calidad: El ítem de configuración de software producido por este rol es el Plan de Aseguramiento de la Calidad.

- Validación y verificación: El ítem de configuración de software producido por este rol es el Plan de Validación y Verificación. Este rol es parte del CCC.

- Documentador: El único ítem de configuración de software producido por este rol es su plan.

- Ingeniero de manutención: Este rol produce los siguientes items de configuración de software: Plan de Manutención, y además es responsable de implementar los cambios debido a cambios pedidos al software después de la fase de diseño detallado.

Perfil de un administrador de configuración

Las personas en este rol deben poder manejar tres elementos: actividades administrativas, auxiliares (registrar eventos), y técnicas. El administrador de configuración debe disponer de los recursos para hacer efectiva una solución. Estos recursos pueden ser experiencia, mano de obra o autoridad. Idealmente, debiesen ser las tres. Un administrador efectivo tiende a influir el resultado de un evento en forma positiva y productiva.

Un administrador de configuración puede obtener experiencia y crear un buen perfil de su rol de la siguientes formas:

- Recibir entrenamiento. No significa que deba recibir entrenamiento exhaustivo, pero si debe conocer las funciones principales de la administración de la configuración de software y sus técnicas.

- Debe estar familiarizado con la mayoría de las áreas de administración funcional, y el conocimiento de los programas necesariamente debe llevarlo a construir un sistema de administración de la configuración adecuado.

- Debe tener en mente que es como la conciencia del programa, o policía, exigiendo revisión completa y decisiones oficiales antes de que se realicen cambios a lo contratado.

- Debe mantener los principios de la administración de configuración visibles y aplicarlos como se definió.

Plan de trabajo

En el plan de trabajo del administrador de configuración figuran las siguientes tareas:

- Planificar las actividades principales de la administración de configuración de software, y como se realizarán durante el ciclo de vida del software.

- Escribir un Plan de Administración de Configuración de Software.

- Elaborar las plantillas necesarias para solicitar un cambio: Propuesta de Cambio de Ingeniería.

- Diseñar y elaborar el repositorio central para mantener los items de configuración de software identificados.

- Estudiar las herramientas que serán usadas para acceder al repositorio.

- Una vez que el proyecto empiece, debe realizar las cinco actividades principales durante el ciclo de vida del software:

    - Identificación de configuración de software.

    - Control de versiones.

    - Control de cambios de software.

    - Auditorias de configuración de software.

    - Contabilidad del estatus de configuración de software.

4.10 Ingeniero de validación y verificación

Una de las metas necesarias de tener en cuenta en toda organización de desarrollo de software que se considere exitosa es que el software que evoluciona, continúe satisfaciendo las expectativas de los usuarios durante dicho proceso. Para lograr esta

meta, es necesario aplicar prácticas de ingeniería de software durante la evolución del producto. La mayoría de estas prácticas tratan de crear y modificar software de forma de maximizar la probabilidad de satisfacer las expectativas de sus usuarios. Otras prácticas tratan de asegurar que el producto cumplirá con las expectativas de dichos usuarios. Estas últimas son ampliamente conocidas como la validación y verificación de software (V&V).

Validación se refiere al proceso de evaluación del software al final de su proceso de desarrollo para asegurarse que está libre de fallas y cumple con sus requisitos. Una falla se define como un comportamiento incorrecto del producto. Validación se refiere al proceso de determinar si el producto en una determinada fase del proceso de desarrollo cumple con los requisitos establecidos en la fase anterior.

V&V es una ayuda para determinar que los requisitos de usuario han sido implementados correcta y completamente, y existen trazas a los requisitos del resto de las fases. El objetivo principal del proceso de V&V es el de analizar y testear el software en forma completa durante el desarrollo para determinar que el software ejecute sus funcionalidad correctamente, asegurarse que no ejecute funciones no definidas, y proveer información sobre su calidad y confiabilidad. En otras palabras, las personas dedicadas a V&V deben evaluar cuan bien el software está cumpliendo con sus requisitos técnicos y sus objetivos de seguridad y confiabilidad relativos al sistema. También asegura que los requisitos de usuario no están en conflicto con ninguno de los estándares o requisitos aplicables a otros componentes del sistema. Las tareas de la V&V de software son analizar, revisar, demostrar y testear todas las salidas del desarrollo de software.

El proceso de V&V produce un Plan de Validación y Verificación de Software (PVVS), planes individuales y reportes para tareas, reportes con resúmenes, reportes con anomalías y un reporte de validación y verificación de software (RVVS) al final. El proceso de V&V contiene actividades de administración de V&V y actividades técnicas de V&V de software.

Es necesario entender que el proceso de V&V tiene sus limitaciones. El objetivo central de los enfoques de V&V de software es el de asegurarse que el producto está libre de fallas y cumple con las expectativas de sus usuarios. Sin embargo, existen muchas limitaciones teóricas y prácticas que hacen que este objetivo no sea posible de obtener para el caso de muchos productos. Aunque no entraremos en detalles, estas limitaciones se deben a:

- Fundamentos teóricos.
- No resulta práctico probar todos los datos.
- No resulta práctico probar todos los caminos posibles en el software.
- No es posible realizar una prueba de correctitud absoluta.

Objetivos

El objetivo principal del proceso de V&V de software es el de analizar y testear en forma completa el software durante el desarrollo para determinar que ejecuta su funcionalidad correctamente, asegurarse que no ejecuta funciones no intencionalmente definidas y proveer información sobre su calidad y confiabilidad.

Dependiendo de las actividades de V&V respecto a la funcionalidad y rendimiento del software, es posible identificar algunos objetivos:

- Correctitud: En que grado el producto está libre de fallas.
- Consistencia: En que grado el producto es consistente consigo mismo y con otros productos.
- Necesidad: En que grado lo que hay en el producto es necesario.
- Suficiencia: En que grado el producto es completo.
- Rendimiento: En que grado el producto satisface los requisitos de rendimiento.

Estos objetivos proveen un marco en que es posible determinar la aplicabilidad de varios enfoques y técnicas de V&V.

Actividades y metas

A continuación se describen las principales metas y actividades del proceso de V&V de software.

Actividades	Metas
Administración de V&V de software	El proceso requiere ser administrado y realizado en forma completa durante todo el proceso de desarrollo de software.
Planificación	Planificar y mantener el proceso de V&V.
Coordinación	Coordinar e interpretar rendimiento y calidad del esfuerzo del proceso de V&V.
Reportar	Reportar discrepancias a la brevedad posible al usuario o al grupo de desarrollo.
Monitoreo	Identificar tempranamente caminos de problemas, focalizando las tareas de V&V en ellos.
Evaluación de resultados	Proveer una evaluación técnica del rendimiento del software y sus atributos de calidad en cada revisión de software.
Evaluación del impacto del cambio	Determinar el impacto completo de los cambios de software propuestos.
Monitoreo del progreso técnico de V&V y calidad de resultados	Durante cada actividad de V&V, se debe revisar las tareas planificadas de V&V, incluyendo nuevas para mantenerse focalizado en las funciones críticas de rendimiento y calidad del software.
Examinar documentación temprana del	Muchas veces los llamados documentos conceptuales, para

proyecto	verificar que el sistema que se construirá no es sólo posible, pero que además utiliza las reglas, convenciones, algoritmos y prácticas apropiadas al dominio de aplicación del sistema.
V&V de los requisitos de usuario	Realizado para asegurarse que los requisitos de usuario especificados son correctos, completos, consistentes, fieles, legibles y es posible testearlos, y que además, satisfacerán los requisitos de software.
V&V de los requisitos de software	Realizado para asegurarse que los requisitos de software especificados son correctos, completos, consistentes, fieles, legibles, es posible mapearlos desde los requisitos de usuario y es posible testearlos, y que además, satisfacerán los requisitos de diseño.
V&V del diseño de software	Provee seguridad de que los requisitos de software no son mal interpretados o implementados en forma incompleta.
V&V del código	Asegurarse que se utilicen los estándares en uso, que usualmente consideran programación estructurada, reuso de código, adopción de estándares y estilos de programación, etc.
Administración de tests	Una actividad importante en la actividad de V&V es la de asegurarse que se consideren y planifiquen todos los testeos requeridos para el sistema.
V&V de la transferencia	Asegurarse que se consideren todos los detalles de la instalación y puesta en marcha del sistema, así como la migración y/o poblamiento de sus datos y posibles problemas de eficiencia que empiezan a aparecer.
V&V de la operación y manutención	Cuando se realiza un cambio en el software, en necesario repetir todas las actividades de V&V pertinentes para asegurarse que nada queda fuera.

Relación con otros roles

El ingeniero a cargo del proceso de V&V de software debe interactuar con otros roles. A continuación se detallan las interacciones más relevantes:

- Analista: El ingeniero de V&V tiene la responsabilidad de verificar que el analista especifica correctamente los requisitos de usuario y de software. Debe además, verificar la documentación producida en diferentes fases del desarrollo del sistema.

- Diseñador: El ingeniero de V&V debe evaluar el nivel de concordancia entre los requisitos de usuario y el modelo diseñado del sistema, buscando errores de interpretación en los requisitos, y características faltantes o mal concebidas.

- Programador: El ingeniero de V&V debe verificar la correctitud del proceso de traducción de diseño de software a su implementación en código. La verificación de código es la última oportunidad de encontrar y eliminar errores pudiesen causar costos innecesarios y atrasos por realizar actividades de testeo sobre código pobre.

- Téster: Se requiere coordinación con este rol en la ejecución de los casos de tests, de acuerdo con las necesidades del cliente.

- Ingeniero de manutención: El ingeniero de V&V requiere realizar chequeos periódicos para asegurarse que la integridad del sistema se mantiene, que los cambios que afectan su operación han sido documentados, y los operadores han recibido entrenamiento en procedimientos nuevos o que han cambiado.

4.11 Documentador

Durante el proceso de desarrollo de software, se genera una gran cantidad de documentación. Dicha documentación debe ser almacenada en el repositorio del proyecto. La documentación sirve, entre otras cosas, para conocer la historia del proyecto. Hay que destacar que los documentos no se escriben al final del proyecto, sino que se van generando junto con las diferentes fases del proyecto. A medida que el proyecto va avanzando, los documentos deben ir siendo modificados para mantener el estado de los documentos a la par con el estado de desarrollo del proyecto. Por lo anterior, debe pensarse que los documentos van evolucionando, para mostrar el estado más reciente de desarrollo del proyecto. Sin embargo, el objetivo principal de la documentación es de actuar como medio de comunicación entre los miembros del equipo, incluyendo el cliente. Además, durante el proyecto, la documentación sirve también para reducir la distorsión de ideas, ayudar al control del proyecto, almacenar la lógica de las decisiones tomadas y hacer visibles, en forma temprana, tanto las capacidades como las limitaciones del sistema.

Se suele clasificar la documentación en dos categorías [Sommerville]:

- Documentación de procesos. Los documentos pertenecientes a esta categoría registran la información del proceso de desarrollo y manutención del sistema. El objetivo de esta documentación es hacer “visible” el proceso de desarrollo, manteniendo información sobre:

    - Planificación y control de procesos (planes, calendarios, estimaciones).

    - Reportes sobre recursos utilizados durante el desarrollo.

    - Estándares a ser utilizados en las diferentes fases.

    - Registro de ideas y estrategias a ser consideradas por el equipo.

    - Lógica de las decisiones de diseño.

    - Detalles de la comunicación diaria entre los gerentes y el equipo de desarrollo.

- Documentación de producto. Estos documentos describen el desarrollo del producto desde los puntos de vista técnico (documentación de sistema) y usuario del sistema (documentación de usuario), siendo el usuario un usuario

final o un usuario administrador del sistema. Estos dos tipos de documentación tienen las siguientes características:

- La documentación de sistema incluye los documentos desde la especificación de requisitos hasta el plan de testeo de aceptación final. Esta documentación es usada durante la fase de manutención y debe ser actualizada cada vez que se realizan cambios al sistema.

- La documentación de usuario debe contener una vista general de los servicios prestados por el sistema, como usarlo, un manual de referencia que permita manejar los errores y facilidades prestadas por el sistema, un documento que describa el proceso de instalación del sistema y un manual de administración.

La calidad de la documentación generada es de gran importancia, debido a que la utilidad del sistema se degrada si no hay información adecuada de cómo usar o entender sus características. Para obtener esta calidad, la documentación del proyecto debe seguir estándares. Estándares que gobiernan el proceso, aseguran el intercambio satisfactorio de información, y determinan como será cada uno de los documentos.

Los estándares de desarrollo de documentos definen el proceso a seguir para producir documentos de alta calidad. Estos estándares definen la forma de crear, de refinar y de realizar la producción final de los documentos. Es claro que estos estándares deben ser lo suficientemente flexibles como para permitir ser usados en distintos tipos de documentos.

Además, es necesario considerar estándares que permitan el intercambio de documentos en caso que se requieran ser intercambiados o transmitidos en forma electrónica. El uso de estos estándares permiten la recreación del documento con el mismo formato que el utilizado en el otro extremo.

Por último, los estándares de documentos especifican la apariencia que todos los documentos deben tener. Dichos estándares tienen el propósito de mantener la consistencia de los documentos, los que incluyen como identificarlo, su estructura y presentación, indicaciones para actualizarlo y líneas generales para un buen estilo de escritura.

Objetivos

El objetivo principal del rol de documentador es el de mantener la información generada durante el proceso de desarrollo. Como objetivos específicos se tienen los siguientes:

- Permitir el almacenamiento y recuperación de la documentación de los procesos y productos más recientes durante el desarrollo, manteniendo así la información al día.

- Mantener la consistencia en la apariencia y estructura de los documentos, facilitando su almacenamiento, recuperación e intercambio, no permitiendo el almacenamiento de documentos con formatos diferentes.

- Asegurarse que los cambios que necesitan hacerse en el sistema serán reflejados en la documentación correspondiente.

- Elaborar, almacenar y permitir la recuperación de las actas y registros generados durante las reuniones de revisión, los que constituyen parte del proceso de documentación.

- Construir el manual de usuarios del sistema, MUS, que contempla los aspectos de uso del sistema.

Actividades y metas

Las actividades de un documentador también son muchas. A continuación se muestran algunas de ellas, ordenadas por metas.

Actividades	Metas
El documentador debe diseñar y construir un repositorio de información compartido, donde se almacenará la documentación.	Tener un repositorio central que permite almacenar, recuperar y mantener la documentación del proyecto.
Mantener el repositorio de información. El documentador debe agregar todos los nuevos documentos generados y remplazar los documentos que fueron modificados en el proceso de desarrollo.	Tener accesible y organizada la última versión de todos los documentos generados durante el proceso de desarrollo, en un repositorio común.
Especificar el formato que será usado para elaborar la documentación. El formato especificado debe contemplar al menos lo siguiente: • Estructura del documento. • Tipos de letra y colores a usar en cada documento. • Distribución de los elementos en el documento (texto, imágenes, dibujos, logos, etc.). • Características de las figuras, imágenes y dibujos consideradas en el documento.	Será posible integrar en un documento único general, todos los documentos almacenados en el repositorio, sin necesidad de cambiar sus formatos o estructura.
Asegurarse que los documentos mantienen el estándar de documentación definido para el proyecto antes de incluirlos en el repositorio.	Toda la información almacenada en el repositorio tendrá el formato definido, y se ajustará al estándar de documentación en uso.
Durante las reuniones de revisiones, el documentador elaborará las actas de la reunión. Estos documentos serán usados luego por el ingeniero de validación y verificación.	Es necesario mantener la historia del proyecto en el repositorio. Al término del proyecto, el repositorio contendrá toda la información histórica del proyecto.
Elaborar el manual de uso del sistema, MUS.	El usuario final debe disponer de un MUS, que le permita operar el sistema correctamente, conociendo sus funciones, y administrando los errores que puedan aparecer durante su ejecución.

Metodología

La metodología de trabajo del documentador está enfocada a realizar las acciones que le permita cumplir con sus objetivos principales y específicos. Al inicio, debe establecer los formatos usados en los documentos del proyecto, definiendo las estructuras de los documentos:

- Información de identificación de cada documento (nombre, tipo, autores, versión, etc.).
- Información que facilite la recuperación de la información (resúmenes, palabras claves).
- Enfoques para estructuras capítulos, secciones y subsecciones, y su numeración respectiva.
- Índices y numeración de páginas.

El estándar ESA define plantillas para cada uno de los documentos. En el Anexo XXX se encuentran dichas plantillas, las que deben ser utilizadas en caso de seguirse dicho estándar. En caso de querer construir su propia metodología de documentación, se sugiere revisar [Brockmann] y [Foehr et al.].

Hay que señalar que no es poco frecuente encontrar gente que piensa que lo importante de un documento es su contenido, menospreciando su forma. Es correcto pensar que el contenido es vital. Sin embargo, se olvidan que un documento es un instrumento de comunicación, y como tal, debe trabajarse su forma para no tener ambigüedades, inconsistencias, y malas interpretaciones. Lo anterior sugiere la necesidad de trabajar las formas del lenguaje con el propósito de producir documentos con mayor capacidad comunicacional. Por otro lado, se hace vital cuidar la ortografía y la gramática de todos los documentos del proyecto. En el Anexo XXX se incluyen algunas sugerencias para esto.

Relación con otros roles

El rol de documentador está relacionado con los otros roles, debido a que el objetivo principal de la documentación es el de servir como medio de comunicación entre los miembros del equipo. El documentador recibe los documentos generados durante las diferentes fases, y se espera que los miembros del equipo puedan acceder a esta documentación en el momento que quieran.

Existe una relación estrecha con los otros roles, como se describe a continuación:

- Administrador de proyecto: La información del repositorio ayuda al administrador a realizar los planes, agenda y presupuestos del proceso de desarrollo de software.

- Administrador de configuración: El administrador de configuración y documentador comparten documentación a través del repositorio. Cuando se autoriza un cambio de un documento, debe ser reflejado en el sistema de administración de documentos. El propósito de esto es evitar inconsistencias y permitir al equipo de desarrollo el acceso a la versión más reciente de cada documento.
- Validación y verificación: El documentador mantiene una relación con el ingeniero de V&V a través del registro elaborado durante las reuniones de revisión. Este registro, así como las minutas elaboradas son utilizadas durante el desarrollo de los planes y agendas de las reuniones de revisión.
- Asegurador de calidad: A los aseguradores de calidad se les ha confiado que garanticen la calidad del producto a ser desarrollado. Toda la documentación del proyecto es parte del producto. Por ello, parte del trabajo del asegurador de calidad es de garantizar la calidad de la documentación generada.
- Ingeniero de manutención: La documentación generada durante el desarrollo del producto es usado durante la fase de operación y manutención del sistema. Si durante esta fase se realizan cambios, será necesario reflejarlos en la documentación correspondiente para mantener la consistencia de la documentación con la operación del sistema.

Herramientas de apoyo

El documentador requerirá herramientas para la elaboración de documentos (minutas, documentos de acuerdos, manual de usuario) y herramientas de apoyo para la elaboración y administración del repositorio de documentos. Dentro del primer conjunto de herramientas se encuentran las siguientes:

- Editor de texto, que permite elaborar documentos fácilmente y almacenarlos en diferentes formatos, incluyendo HTML. Debe incluir una herramienta para el chequeo ortográfico y de gramática.
- Navegador Web y editor HTML, que permita editar y publicar documentos HTML, proveyendo diversas funciones.

Por otro lado, las herramientas de apoyo para la elaboración y administración del repositorio de documentos deben proveer el acceso a los documentos. Estas herramientas dependerán de la decisión del repositorio en uso durante el desarrollo del proyecto. Hoy en día existen muchas herramientas de manejo de repositorios de documentos, con interfaces Web.

Perfil del documentador

El documentador debe ser una persona ordenada, con capacidades de mantener una gran cantidad de información en forma ordenada y accesible. Todo el contenido de los documentos debe ser organizado en forma clara. Esta claridad debe ser consecuencia del formato en que se presenta la información. El documentador debe poseer capacidades para no sólo apoyar su trabajo con tecnología, sino que además, deberá diseñar y construir el repositorio para la documentación del proyecto. Esto incluye, al menos, la definición del modelo de datos, definición de las interfaces, definición de los perfiles de acceso de los usuarios, y la definición del protocolo social de uso de los documentos.

El documentador también debe tener creatividad para presentar la información y aptitud de expresión para escribir. Hay que señalar que debe conocer y utilizar el procesador de texto definido para el proyecto en toda su potencialidad, utilizando funcionalidades como estilos, corrector sintáctico y gramatical, y control de versión.

Plan de trabajo

Las actividades del documentador se dividen en dos grupos. El primer grupo se refiere a actividades que se realizan durante el ciclo de vida del proyecto. Entre estas actividades se encuentra la elaboración de las actas de reuniones, el almacenamiento de los documentos generados durante el proyecto, manutención del repositorio de información, y manutención del sitio Web del proyecto. El segundo grupo se refiere a actividades que se realizan sólo una vez, al principio o al final del proyecto. Entre estas actividades se encuentra la elaboración de la documentación, el diseño e implementación de la base de datos para la documentación, y la elaboración de los perfiles de acceso a la documentación para cada uno de los miembros del equipo.

A continuación se detallan las actividades principales del documentador:

- Elaboración de los formatos de la documentación:
    - Determinar los tipos de documentos que se desarrollarán durante el proyecto.
    - Definir su formato (.doc, .rtf, .html, .txt, .ppt, .xls, .mpp, etc.).
    - Elaborar una plantilla para cada documento, en el formato definido.
    - Publicar las plantillas.
- Diseño y elaboración del repositorio central usado para almacenar información:
    - Análisis de las características deseadas del repositorio.
    - Estudio de las herramientas que serán usadas para elaborar el repositorio.

    - Elaborar el diseño lógico de la base de datos.
    - Construir el repositorio.
    - Incluir un puntero entre la base de datos y el sitio Web del proyecto, en caso que sean diferentes.
- Elaboración de las actas de reunión y documentos de acuerdo:
    - Asistir a reuniones y escribir las actas.
    - Elaborar los documentos de acuerdo de las reuniones.
    - Elaborar las actas finales de cada reunión.
    - Incluir las actas en el repositorio.
- Almacenamiento de los documentos generados:
    - Verificar formato de los documentos.
    - Almacenar los documentos a medida que se van produciendo en el ciclo de desarrollo.
- Manutención del repositorio con la documentación, actualizando la información a medida que se produce.
- Manutención del sitio Web del proyecto durante todo su ciclo de vida.
- Elaboración del manual de usuario del sistema.

Es claro que parte de la labor del documentador se simplifica si el proyecto se ajusta a un estándar de documentación, como es el caso del estándar ESA PS-05-0. En dicho caso, las plantillas de documentos ya vienen definidas. En el Anexo XXX se muestran dichas plantillas.

Por otro lado, hoy en día se suele crear un sitio Web donde se maneja, en forma conjunta, la información pública y privada del proyecto. En ese sitio se mantiene el repositorio de todos los documentos del proyecto. La información pública es toda aquella información del desarrollo del proyecto que se desee publicar con fines de divulgación. En el caso privado, el acceso al repositorio está restringido a los miembros del equipo de desarrollo. En el Capítulo 5 se muestra un ejemplo de la estructura y funcionalidad de un sitio Web para administrar la documentación del proyecto.

4.12 Ingeniero de manutención

Hace ya casi 30 años, el proceso de manutención de software se caracterizaba con el término “iceberg” [Canning], ya que los problemas que eran visibles eran una pequeña parte de la enorme cantidad de problemas potenciales y costos que se escondían debajo de la superficie.

La manutención es la última fase del proceso de desarrollo de software. Sin embargo, la manutención toma una parte importante del presupuesto destinado al desarrollo. En general, dichos costos son subestimados o simplemente ignorados. Por otro lado, a medida que se desarrollan más programas, la cantidad de esfuerzo y recursos dedicados a la manutención crecerá. Al final, algunas empresas pueden llegar a encontrarse con la barrera de la manutención, no pudiendo abordar nuevos proyectos debido a que todos sus recursos están dedicados a mantener programas antiguos.

Sólo el 20% del trabajo de manutención es usado arreglando errores. El 80% restante se utiliza adaptando sistemas existentes a cambios en su ambiente externo, realizando mejoras pedidas por usuarios, y realizando reingeniería del sistema para usos futuros.

Objetivos

Los objetivos a cumplir por un ingeniero de manutención son los siguientes:

- Modificar el software para adaptar nuevas funciones o modificar algunas funciones existentes.
- Modernizar el software por medio de cambios al sistema.
- Asegurarse de que el equipo de desarrollo esté informado de los errores encontrados en el sistema.

Actividades y metas

A continuación se presentan las actividades para cada una de las metas a cumplir por el ingeniero de manutención:

Actividades	Metas
Manutención correctiva	Diagnóstico y corrección de errores encontrados durante el uso del programa.
Manutención adaptiva	Adaptar el sistema a los cambios que pueden producirse en el hardware, sistema operativo, periféricos y herramientas de trabajo.
Manutención perfectiva	Satisfacer la demanda de recomendaciones por parte de los usuarios del sistema. Realizar cambios al sistema para mejorar el proceso de manutención o confiabilidad del sistema.

Metodología

La metodología a usar por el ingeniero de manutención debe considerar los siguientes aspectos:

- Establecer y coordinar la organización de manutención.
- Definir los procesos de evaluación e información.
- Definición de una secuencia estándar de eventos para cada requisito de manutención.
- Establecer un sistema de registro e información de las actividades de manutención.
- Definición de las actividades de revisión y evaluación.

Relación con otros roles

El ingeniero de manutención debe mantener relaciones con varios roles. Entre ellos se consideran los siguientes:

- Administrador de proyecto. Debe supervisar y controlar los cambios requeridos, utilizando los estándares del proyecto.
- Analista. Debido a que probablemente será necesario adaptar o perfeccionar el sistema durante el tiempo, los analistas requerirán determinar nuevos requisitos.
- Diseñador. Es necesario involucrar al diseñador para rediseñar las partes del sistema que requieren ser corregidas, agregadas o perfeccionadas.
- Programador. La naturaleza de las actividades requeridas para cubrir la fase de manutención está estrechamente ligada con el rol de programador. Durante el uso del sistema es probable que se detecten errores, siendo necesario comunicárselos al equipo de desarrollo para su reparación.
- Téster. Debe testear las partes que han sido modificadas para chequear si están adecuadas para ser liberadas como parte del sistema.
- Asegurador de calidad. Es responsable de asegurar que el resultado del producto es de la calidad definida por el estándar en uso.
- Validación y verificación. Es responsable de revisar que el cliente quede completamente satisfecho con el producto entregado.

- Documentador. Es responsable de documentar los cambios aprobados por el Comité de Control de Cambios durante la fase de manutención.

Herramientas de apoyo

El ingeniero de manutención debe utilizar para realizar su labor, una herramienta que le permita capturar requisitos de manutención, y además controlar la organización de la manutención dentro del proyecto. Es deseable que dicha herramienta esté integrada al ambiente de trabajo utilizado por el resto del equipo, compartiendo el mismo repositorio común. Todos los requisitos de manutención, así como la información operacional del proceso de manutención, deben ser almacenados en el sitio del proyecto para su revisión por todos los miembros del equipo de desarrollo.

Perfil del ingeniero de manutención

El perfil del ingeniero de manutención, como es el caso de otras formas de administración, requiere de la creación y preservación de una atmósfera adecuada que permita llevar a cabo las actividades de la mejor forma posible. Existen aspectos que son comunes a la mayoría de las actividades de administración. No obstante, se espera que el ingeniero de manutención tenga visión para poder predecir las actividades de manutención a futuro.

Plan de trabajo

Las actividades de manutención se llevan a cabo durante todo el proceso de desarrollo del proyecto. Existen actividades en cada fase del proceso de desarrollo, desde manutención preventiva a correctiva. El proceso de organización de la manutención requiere trabajar con todo el equipo de desarrollo, organizando y controlando el buen desempeño de sus actividades.

Algunas de las actividades a realizar por el ingeniero de manutención se describen a continuación.

- Planificación y coordinación de la organización de manutención durante el proyecto.
- Definición de procedimientos de evaluación e información.
- Desarrollo de herramientas para administrar los requisitos de manutención.
- Definición de una secuencia estándar de eventos para cada requisito de manutención.
- Definición de un sistema de registro e información de las actividades de manutención y organización de la manutención.

- Definición y uso de enfoques de revisión y evaluación.

4.13 Cliente comprometido

Es frecuente escuchar que los términos “cliente”, “usuario” y “usuario final” se utilizan como sinónimos, lo cual puede provocar confusión. Un cliente es aquella persona responsable de llevar a cabo el buen desempeño del proyecto, por parte de la empresa que contrata el desarrollo, también llamada mandante. El cliente debe representar los derechos y asumir los deberes de dicha empresa ante el equipo de desarrollo. Por lo tanto, el cliente debe estar presente en todas las fases del desarrollo del producto, y realizar todas las actividades que se esperan de él, tales como la aceptación provisional y final del producto.

Por otro lado, los usuarios corresponden a las personas que están operando día a día un sistema de software. Es la persona que conoce el problema, y utiliza la herramienta computacional para apoyar su trabajo. Un cliente y un usuario no siempre son lo mismo, ya que es posible que el cliente no opere el sistema de información.

Un usuario final generalmente se refiere a aquella persona que utiliza el sistema, pero que es desconocida o no identificable. Generalmente pasa esto en sistemas de información de uso masivo, tales como los sistemas de atención bancaria por Internet, sistemas de apoyo al comercio electrónico, etc.

En el desarrollo de software nos interesa tener clientes comprometidos. Es vital para el éxito del proyecto. Un cliente comprometido es aquel que participa en todas las etapas del proyecto, compartiendo deberes y responsabilidades.

Entre sus tareas están:

- Liderar el proyecto de software cuando la organización así lo requiere.
- Debe conocer las distintas etapas y roles en la construcción de software.
- Definir los objetivos del proyecto negociando con sus clientes las características que le afecten.
- Definir y priorizar requisitos.
- Revisar y aprobar documentos en forma responsable.
- Difundir el estado del proyecto al resto de su ámbito de trabajo.
- Entregar los recursos necesarios para la realización del proyecto.
- Escribir o participar en la elaboración del manual de usuario del sistema (MUS).

- Determinar y alertar del impacto del proyecto en otras áreas de la organización.

- Realizar la capacitación del sistema a sus usuarios.

- Construir el plan de pruebas de aceptación del sistema y aplicarlo al final del proyecto, aceptando o rechazando la entrega.

Relación con otros roles

Un cliente comprometido se relacionará principalmente con el administrador de proyecto y con los analistas en forma frecuente. No obstante, también necesitará relacionarse con otros roles con el propósito de controlar, monitorear y aceptar el sistema. A continuación se detalla dichas relaciones:

- Administrador de proyecto. El cliente comprometido llevará toda la relación formal del proyecto a través de su administrador. En otras palabras, toda la comunicación formal entre las dos partes del proyecto se hace a través de estas dos personas. El cliente deberá además, cuidar la relación con el equipo de desarrollo.

- Analista. El cliente comprometido participará en reuniones sistemáticas de análisis, dirigidas por los analistas. Deberá además participar en las RTFs de análisis, tales como las fases de revisión RU/R y RS/R.

- Diseñador: El cliente comprometido debe participar en las fases de revisión DA/R y DD/R junto a los diseñadores.

- Programador. El cliente comprometido debe participar en la fase de revisión DD/R junto a los programadores.

- Téster. El cliente debe preparar el plan de aceptación parcial y plan de aceptación definitiva, con apoyo del téster. Además, deberá trabajar con el téster para apoyar la construcción del plan de testeo.

- Asegurador de calidad. El cliente comprometido debe apoyar al asegurador de calidad en su búsqueda de diferencias entre los requisitos de usuario y las funcionalidades implementadas en el sistema.

