Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Sistemas

ADMINISTRACIÓN DE UN PROYECTO DE *SOFTWARE*

Kenneth Eduardo Zea Rodríguez
Asesorado por Ing. Jorge Estuardo Mancilla Tello

Guatemala, enero de 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ADMINISTRACIÓN DE UN PROYECTO DE *SOFTWARE*

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR

KENNETH EDUARDO ZEA RODRÍGUEZ

ASESORADO POR ING. JORGE ESTUARDO MANCILLA TELLO

AL CONFERÍRSELE TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, ENERO DE 2005.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Sydney Alexander Samuels Milson
VOCAL I	Ing. Murphy Olympo Paiz Recinos
VOCAL II	Lic. Amahán Sánchez Alvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIO	Ing. Carlos Humberto Pérez Rodríguez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Ing. Marlon Antonio Perez Turk
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Cesar Augusto Fernandez Caceres.
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

ADMINISTRACIÓN DE UN PROYECTO DE *SOFTWARE*

Tema que me fuera asignado por la Dirección de la Escuela de Ciencias y Sistemas con fecha febrero de 2001.

Kenneth Eduardo Zea Rodríguez

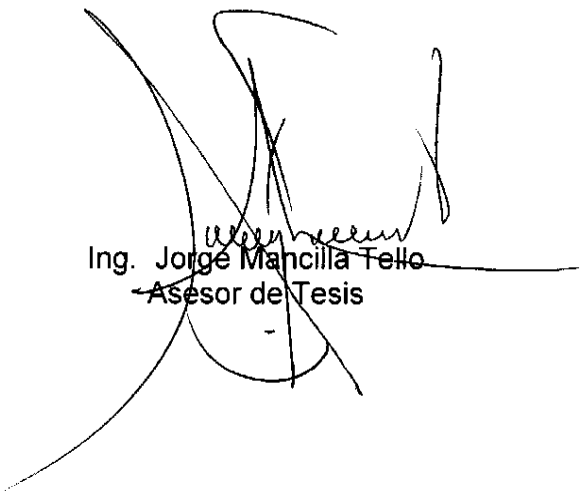
Guatemala, Octubre 8 del 2002

Ing. Carlos Azurdia.
Coordinador Comisión de Tesis
Escuela de Ciencias y Sistemas
Facultad de Ingeniería, USAC.

Ing. Azurdia:

Por medio de la presente hago de su conocimiento que he tenido a bien revisar el trabajo de graduación de **KENNETH EDUARDO ZEA RODRIGUEZ**, titulado "**ADMINISTRACION DE UN PROYECTO DE *SOFTWARE***", por lo cual me permito recomendar dicho trabajo final para la respectiva revisión por parte de la comisión de la escuela de Ciencias y Sistemas.

Sin otro particular me suscribo atentamente.



Ing. Jorge Mancilla Tello
Asesor de Tesis

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Carrera de Ciencias y Sistemas

Guatemala, 23 de Octubre de 2002

Ingeniero
Luis Alberto Vettorazzi España
Coordinador de la Carrera de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Vettorazzi:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **KENNETH EDUARDO ZEA RODRÍGUEZ**, titulado: **"ADMINISTRACIÓN DE UN PROYECTO DE SOFTWARE"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA - INGENIERIA EN CIENCIAS Y SISTEMAS - COORDINACION TESIS

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

El Director de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, con el visto bueno del Revisor y del Licenciado en Letras, del trabajo de graduación titulado **"ADMINISTRACION DE UN PROYECTO DE *SOFTWARE*"**, presentado por el estudiante universitario Kenneth Eduardo Zea Rodríguez, aprueba el presente trabajo y solicita la autorización del mismo

ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Luis Alberto Vettorazzi España
DIRECTOR
INGENIERIA EN CIENCIAS Y SISTEMAS

Guatemala, Enero de 2005.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato
Tels. 24769579 Exts. 101-102-114
Fax: 24760365

Ref. DTG-010-2005

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **ADMINISTRACIÓN DE UN PROYECTO DE SOFTWARE,** presentado por el estudiante universitario: **Kenneth Eduardo Zea Rodríguez,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Sydney Alexander Samuels Milson
DECANO

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, enero de 2005

/lmcb.

AGRADECIMIENTOS

A Dios	Por la vida, por la sabiduría, el conocimiento y la oportunidad de ser una persona de aprendizaje y conocimiento, que pueda ser útil al prójimo y en beneficio de todos, no sólo personal.
A mis padres	Que con tanto sacrificio ven alcanzado un logro que no sólo es mío, sino también es de ellos, y que por su ayuda y comprensión, he logrado cumplir.
A mis hermanas	Ingrid, que me enseño el camino que debía seguir para superarme y con su ejemplo lo he podido lograr. Heidi, que mi logro sea guía para ella, y saber por donde caminar para alcanzar su objetivo.
A mi sobrina	Que mi sacrificio sea un ejemplo para ella en el futuro y guié su camino a seguir.
A mis compañero de clase	Con quienes tanto se aprendió, se rió, se sufrió, se desveló. Y a quienes les tengo un gran aprecio y cariño. En especial a Carlos Yon, Roberto Villate, Ricardo Jiménez, Gerson Villatoro, Ludwing Altan, Mauricio Lone y Mauricio Sánchez.
A mis compañeros de labores	De los cuales tengo gratas experiencias en el trabajo, y que tanto me han enseñado en este largo e interesante camino de trabajo. En especial a Jorge Mancilla, Luis Álvarez, Edgar Cumes, Carlos Recinos.

ÍNDICE GENERAL

GLOSARIO

Administración Suministrar, proporcionar o distribuir algo. Ordenar, disponer, organizar, en especial un proyecto.

Autoridad Potestad, facultad, legitimidad.

Calidad Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.

Concordancia Correspondencia o conformidad de una cosa con otra.

Contingencia Situación que puede suceder o no suceder.

Control Comprobación, inspección, fiscalización, intervención.

Dirección Camino o rumbo que se debe seguir para ejecutar un movimiento.

Estrategias Proceso regulable, conjunto de las reglas que aseguran una decisión óptima en cada momento.

Garantía Cosa que asegura y protege contra algún riesgo o necesidad.

Integración Hacer que alguien o algo pase a formar parte de un todo.

Liderazgo	Situación de superioridad en que se halla una empresa, un producto o un sector económico, dentro de su ámbito.
Motivación	Provocar los intereses propios del individuo.
Normas	Regla que debe seguirse o se deben ajustar las conductas, tareas, actividades.
Organización	Asociación de personas regulada por un conjunto de normas en función de determinados fines.
Planificación	Organización sistematiazada y metódica frecuentemente de gran amplitud, para obtener un objetivo determinado.
Procedimientos	Método para ejecutar algunas cosas.
Proyecto	Disposición que tiene para la realización de un trabajo o para la ejecución de algo importante.
Software	Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
Teoría	Conocimiento especulativo considerado con independencia de toda aplicación.

RESUMEN

El presente trabajo de graduación consta de cinco capítulos, los cuales hacen énfasis en la administración de los proyectos de *software*, agregando además un último capítulo el cual hace referencia a la calidad de un proyecto de ésta índole.

En el capítulo primero son tratados los puntos relacionados con los conceptos básicos que se han de aplicar dentro del trabajo de graduación, indicando de forma general todos aquellos aspectos de administración que han de ser tratados y orientados al desarrollo de *software*.

En el capítulo segundo se tratan los aspectos más importantes y relevante de la organización que deben ser tomados en cuenta para el desarrollo eficaz de un proyecto de *software*, indicando en forma específica todos aquellos puntos posibles por los gerentes o jefes de proyectos para el buen desarrollo de *software*.

En el capítulo tercero se estipulan todos aquellos elementos que se consideran importantes para la aplicación correcta de los lineamientos básicos de la buena integración del personal, de un proyecto de *software*.

En el capítulo cuarto son tratados aquellos puntos y aspectos sobresaliente para poder aplicar en forma correcta una buena dirección dentro del proyecto de *software*.

En el capítulo quinto se tratan los lineamientos principales e importantes que deben de ser tomados en cuenta para el buen control de un proyecto de *software*.

En el sexto y último capítulo son tratados los aspectos y lineamientos que han sido identificados como más importantes sobre la calidad que debe de imperar dentro de un proyecto de *software*.

OBJETIVOS

General

Establecer una guía a seguir para poder implementar en forma correcta el desarrollo de un proyecto de *software* y apoyarse en los lineamientos de la administración tradicional para poder crear un modelo de administración orientado al desarrollo de *software*.

Específicos

1. Aplicar las técnicas de organización, planeación, integración de personal, dirección y control, presentadas por la administración para el desarrollo de los proyectos de *software*.

2. Utilizar y aplicar las técnicas de control de calidad en el desarrollo de proyectos de *software*.

3. Hacer uso de la administración para enfocar el desarrollo de cada una de las etapas de esta, para la elaboración y control de un proyecto de *software*.

INTRODUCCIÒN

La administración ha sido por mucho tiempo el pilar fundamental de toda empresa y a pesar del ambiente tan cambiante en que la administración se desenvuelve, ésta tiene que estar siempre en un punto actualizado y funcional. Si la administración en general ha sido cambiante y variable en los últimos tiempos, también lo ha sido en un porcentaje mayor en las empresas dedicadas al desarrollo del *software*.

Es por esto que el presente trabajo de graduación está enfocado a la forma de guiar un proyecto de *software* a nivel administrativo, tomando en cuenta todas aquellas políticas, ventajas y desventajas que conlleva la realización de un proyecto de esta índole.

Se presentan capítulos, en los cuales son desarrollados los puntos más importantes de la administración tradicional, los cuales están orientados al desarrollo de proyectos de *software*, donde se hace énfasis en la aplicación de la planeación, organización, integración de personal, dirección y control.

Por último fue incluido un capítulo el cual no es parte de una administración tradicional, pero que no podía quedar fuera por el tipo de orientación que le ha sido aplicado a dicho trabajo de graduación, el cual es la calidad de un proyecto de *software*, este indica los lineamiento y reglas a seguir para poder asegurar la calidad en dicho proyecto de *software*.

1. CONCEPTOS BÁSICOS

1.1 Definición de administración

Se dice que la administración, es el proceso de diseñar y mantener un entorno en el que, trabajando en grupos, los individuos cumplan eficientemente objetivos específicos.

Esta definición la podemos aplicar a un proyecto de *software* de la siguiente manera:

a) Cuando se desarrolla un proyecto de *software*, el jefe o los administradores encargados del mismo, deben ejercer las funciones administrativas de planeación, organización, integración de personal, dirección y control.

b) Esta administración se debe aplicar a todo tipo de proyecto, desde el nivel más alto, hasta los niveles menores que este proyecto implica.

c) Se debe aplicar a todo el personal y en todos los niveles del proyecto a ejecutar.

d) La intención de la administración de un proyecto de *software* es mejorar y producir un *superávit* dentro del proyecto en mención.

e) Administrar un proyecto de *software* tiene el fin de producir un *software* en forma productiva, es decir, producir el mismo con eficacia y eficiencia.

Los administradores han dado fe que la buena y útil organización de los conocimientos que se poseen sobre una organización, ayuda y facilita a la mejor administración de la misma. Es por eso que se ha decidido dividir la administración en cinco funciones principales: planeación, organización, integración de personal, dirección y control; que son tomados como base para poder organizar los conocimientos de dichas funciones.

Debe tenerse en cuenta que la administración de un proyecto de *software* se basa y se centra principalmente en un ambiente interno, no se debe de ignorar lo referente a un ambiente externo, con el cual siempre se convive y puede en algún momento con la trayectoria del proyecto distraer la atención de los administradores o siendo aún peor, conllevar a utilizar técnicas de rescate o acoplamiento al mismo.

Todo administrador no puede estar libre del comportamiento externo de algún proyecto y siempre se vera afectado por los elementos que lo componen. Por ejemplo, los factores económicos, tecnológicos, sociales, políticos y éticos, que afectan esta área de operación.

Como un administrador es la persona que asume la responsabilidad de emprender acciones, que permitan a su personal a cargo realizar sus mejores contribuciones al cumplimiento de sus objetivos y tareas asignadas, para el cumplimiento de sus responsabilidades, ya sean estas pequeñas o grandes, siendo este un administrador de proyectos grandes o pequeños, no importando así cual es el fin de este proyecto.

Dentro de un proyecto de *software* pueden existir a lo máximo tres tipos de administradores: administradores de alto nivel, que se ocupan en mayor parte en la planeación y organización; administradores de nivel medio que ocupan más su tiempo entre la organización y la dirección; y administradores de bajo nivel, que se enfocan en la dirección.

Dentro de una organización deben existir cuatro tipos de habilidades:

a) **Habilidades técnicas.** Son aquellas donde la base de la posesión de conocimiento y destrezas tanto en *software* como en *hardware*, son utilizadas en actividades que suponen la aplicación de métodos, procesos y procedimientos.

b) **Habilidades humanas.** Es la capacidad para trabajar con individuos, esfuerzos cooperativos, logro de metas en conjunto, desarrollo de productos, unificando el trabajo de cada una de las personas incluidas en el grupo de trabajo, etc.

c) **Habilidades de conceptualización.** Es la capacidad de percibir el panorama general de un proyecto o una organización, distinguir todos aquellos elementos que sean de importancia para su buen camino, como identificar con facilidad aquellos puntos donde el desarrollo del proyecto conlleve a puntos no fijados desde el principio del mismo.

d) **Habilidades de diseño.** Es la habilidad más importante que debe de desarrollar una administrador de proyectos de *software*. Esta implica la capacidad de resolver problemas en beneficio del proyecto de *software*, ya sean problemas técnicos, conceptuales o de análisis y diseño.

Para ser un administrador eficaz, se deben tener la capacidad de advertir el problema y deducir la solución práctica que se le presenta. Si el administrador se dedica únicamente a identificar y observar el problema, entonces fracasaría. Esto lo obliga a poseer la habilidad de diseñar una solución funcional al problema en respuesta a la realidad que enfrenta en ese momento.

Las habilidades descritas anteriormente se distribuyen entre los tipos de administradores que pueden existir: administradores de alto nivel ocupan más su tiempo en habilidades de conceptualización y diseño, y en habilidades humanas; los administradores de nivel intermedio se enfocan en habilidades de conceptualización, habilidades de diseño, habilidades técnicas y habilidades humanas; en tanto los administradores de bajo nivel se proyectan en mayor parte en habilidades técnicas y habilidades humanas.

La mayoría de los administradores definen que el propósito de su administración es simple: "generación de utilidades", en nuestro caso lo podemos definir de muchas formas:

a) Realizar código reutilizable, el cual puede ahorrar tiempo y esfuerzo para proyectos de *software* posteriores.

b) Desarrollar productos a menor plazo del que había sido planificado, lo cual involucra ahorro de dinero y tiempo a dicho proyecto.

c) Planificación óptima del trabajo a realizar por el personal a cargo, lo cual lo lleva a un costo mínimo con una producción máxima.

Con este concepto, los administradores de proyectos de *software* deben establecer un entorno en el que los individuos puedan cumplir metas grupales en la menor cantidad de tiempo, dinero, materiales e insatisfacción personal o en el que puedan alcanzar en la mayor medida posible una meta deseada con los recursos disponibles.

Como toda industria, las organizaciones dedicadas a los proyectos de *software* deben guiarse y tratar de volverse una empresa de excelencia, las cuales tienen como objetivos:

a) Orientarse a la acción.

b) Promover la autonomía administrativa y el espíritu empresarial.

c) Obtener mayor productividad atendiendo las necesidades de su personal.

d) Concentrarse en el área de actividad que conocen mejor.

e) Poseer una estructura organizacional sencilla y poco personal administrativo.

f) Ser centralizados o descentralizados, según la necesidad.

También es necesario poder definir la productividad, como “la relación producto insumo en un periodo específico con la debida consideración de la calidad”, y aplicándolo al desarrollo de *software* podemos decir que es “la relación entre un sistema de *software* y un equipo de trabajo dentro de un periodo determinado de tiempo, considerando la calidad del trabajo desarrollado”.

Por último definiremos la eficacia como el cumplimiento de las tareas asignadas a una persona o grupo de personas, y eficiencia como el logro de dichas metas por esta persona o grupo de personas con la menor cantidad de recursos posibles.

Las técnicas administrativas indican la manera de hacer las cosas, es decir, métodos para poder obtener un producto de *software* deseado.

Como se ha mencionado anteriormente existen cinco puntos básicos en que se puede dividir la administración de un proyecto de *software*: planeación, organización, integración de personal, dirección y control, aunque éstos sean los más importantes hablando del ambiente interno de un proyecto de s*oftware*, también se deben considerar otros puntos que se dan en el ambiente externo del proyecto, que también son importantes, aunque no sean tratados con gran profundidad en este tema. Dichos puntos son:

a) **Insumos.** Los insumos de un proyecto de *software* pueden ser muchos y de forma variada. Por ejemplo, se puede mencionar el equipo de computo, el cual es manejado completamente en un ambiente externo a la organización encargada del desarrollo de dicho proyecto.

También se puede mencionar las licencias de s*oftware* que son requeridas para los diferentes tipos de productos utilizados en el desarrollo del mismo.

El tipo de cambio del dólar puede ser otro insumo que en algún momento afectaría el desarrollo de un proyecto de *software*, este se da por la constante variación de dicho tipo de cambio y la utilización de la moneda extranjera en una gran cantidad de empresas para la cancelación de sistemas de *software*.

Otro punto externo que puede afectar el proyecto de *software*, y que también puede ser considerado como insumo, es el propio personal utilizado para el desarrollo del proyecto. Tanto la falta de personal capacitado dentro del mercado en el cual se desenvuelve, puede ser una problema externo, hasta la constante competencia que se puede dar entre empresas dedicadas a la misma labor.

b) **Comunicación.** La comunicación es un punto muy importante dentro del desarrollo de un proyecto de *software*. Primero, es utilizado para la integración de funciones administrativas, también es importante entre los administradores que han de llevar a cabo una función específica dentro del proyecto, tanto al mismo nivel como entre niveles, ya sean en forma ascendente o en forma descendente, dependiendo de la necesidad.

Segundo, la comunicación también debe existir hacia el ambiente externo del proyecto de *software*, para establecer con mayor claridad cual es el ambiente en el que se desenvuelve y la forma óptima de adaptarse al mismo.

c) **Revitalización.** La revitalización se puede tomar como un ciclo de *software*, donde el producto final puede ser utilizado como punto de partida para generar

un producto más grande o tomarlo como base para generar un sistema de *software* de características similares.

Podemos decir, que la función de un administrador de un proyecto de *software* es construir una estructura útil para organizar las tareas, conocimientos y funciones que se deben emplear dentro del proyecto y poder así encaminarlos y llevarlos a su cumplimiento. Por esto, existen los cinco puntos administrativos mencionados anteriormente, y que serán descritos a continuación.

1.2 Definición de planeación

Dentro del punto de planeación están implicados la selección de misiones y objetivos que se han de alcanzar dentro de la realización del proyecto de *software*, así como las acciones que se deben ejercer para poder alcanzarlos y realizar la toma de decisiones pertinentes, para seleccionar las acciones futuras a ejercer, tomando como base alternativas propuestas para la misma. Como se explicará en capítulos posteriores existen varias formas y tipos de planes que se pueden ejercer o poner en práctica para alcanzar los puntos propuestos y necesarios que puedan lograr la meta del proyecto de *software*.

1.3 Definición de organización

La organización busca crear y emprender grupos de trabajo que se dediquen a la búsqueda de fines en común para el cumplimiento de las tareas que son necesarias en el cumplimiento del proyecto de *software*, de esta misma forma debe existir una persona encargada de velar por la realización de dichas tareas, por los grupos encargados de las mismas.

Esto también implica que las personas que integran estos grupos de trabajo deben estar consientes del papel que juegan dentro de toda la organización y colaborar con el esfuerzo grupal, teniendo también la autoridad e información necesaria para el cumplimiento de los mismos.

Se puede decir que la organización es la parte de la administración que ayudará a realizar una estructura intencionada, con el fin de definir los papeles que han de ocupar los individuos involucrados dentro de la realización del proyecto de *software*, y debe de la misma forma garantizar la asignación de todas las tareas necesarias que han de realizarse para la culminación de dicho proyecto de *software*, como la responsabilidad de asignar estas tareas al personal mejor calificado para el cumplimiento de las mismas.

1.4 Definición de integración de personal

La integración del personal se encarga de mantener ocupadas todas aquellas plazas que han sido asignadas para el desarrollo del proyecto de *software*.

Esto se lleva a cabo, mediante la identificación o solicitud de la fuerza de trabajo en mención, incluyendo también el reclutamiento, selección, ubicación, ascensos, evaluación, planeación profesional, compensación y capacitación, tanto a los aspirantes a los puestos en mención, como a las personas que ya integran nuestra fuerza de trabajo, con el fin de lograr una eficaz y eficiente realización de las tareas asignadas.

1.5 Definición de dirección

La dirección es la influencia que se tiene en el grupo de trabajo del proyecto de *software* para que contribuya con el cumplimiento de las metas organizacionales y grupales, teniendo estrecha relación con el aspecto interpersonal de la administración del proyecto de *software*. También se debe tomar en cuenta que un buen administrador debe ser un buen líder, y tener la capacidad de motivar y establecer estilos y liderazgo dentro del grupo de personas en el cual trabaja.

1.6 Definición de control

El control consiste en medir y corregir el camino que lleva nuestro proyecto de *software* y hacer que este se pueda apegar a los planes originales, los cuales se deben cumplir.

Este punto también implica la medición del desempeño con base en las metas y los planes previstos en el principio del proyecto, como la detección de desviaciones y la contribución a corregir las mismas.

En pocas palabras se puede decir, que el control facilita el cumplimiento de los planes antes previstos. Estas actividades se relacionan con la medición de los logros.

1.7 Planificación de un proyecto de software

La planificación es la función administrativa básica de la administración. Es la encargada de establecer y seleccionar misiones y objetivos y las acciones para cumplir con los mismos, requiere de la toma de decisiones, esto quiere decir, optar entre diferentes cursos futuros de acción.

Por lo tanto, la planificación dentro de un proyecto de *software*, donde se implica el manejo de grupos de trabajo, es la base fundamental y principal para todo administrador o jefe de proyectos, el cual debe enfocarse a dar a conocer los propósitos, fines y objetivos de todo el proyecto y cual es la función que cada uno ha de desempeñar dentro del desarrollo del proyecto de *software*.

De este principio, el administrador de un proyecto de *software*, debe tener claro que los planes son los que constituyen un método racional para el cumplimiento de los objetivos que significa su proyecto.

1.8 Objetivos de la planificación

Para que un buen administrador pueda definir de forma correcta una planificación, que ha de utilizar a lo largo del proyecto de *software*, son necesarios siete puntos, los cuales son:

a) **Definición de la misión del proyecto.** Este punto identifica la tarea básica por la cual se dio origen al proyecto de *software*, definir de forma clara y precisa cuales son los puntos a cubrir y enfoque teórico práctico del *software*.

b) **Definición de los objetivos del proyecto.** Este punto es utilizado para identificar de forma clara, cual es la finalidad y metas que se pretenden cubrir o ampliar con el desarrollo del proyecto de *software*.

c) **Definición de estrategias de trabajo.** Dentro del contexto de un administrador o jefe de proyectos, se puede definir las estrategias como los objetivos básicos o primordiales a largo plazo que se deben cumplir dentro de dicho proyecto, las acciones que se han de tomar para el cumplimiento y la asignación de recursos necesarios para el mismo.

d) **Definición de políticas de trabajo.** Las políticas son utilizadas por los administradores de proyectos de *software* como enunciados que se utilizan para resolver asuntos o problemas de una forma coherente y estable, definiendo esto como planes o criterios generales que orientan el pensamiento en la toma de decisiones.

e) **Definición de procedimientos de trabajo.** La definición de procedimientos dentro de un proyecto de *software* es una de las tareas más importantes para el administrador o jefe de proyectos, ya que en el mismo se definen los métodos para el manejo de actividades futuras, la secuencia cronológica (se puede definir un plan de trabajo) de todas las actividades en forma exacta, que deben cumplirse para la finalización satisfactoria de dicho proyecto.

f) **Definición de reglas de trabajo.** La definición de reglas de trabajo dentro de un proyecto de *software* es una de las tareas donde la personalidad y el carácter de un buen administrador se comprueba, las reglas que define un administrador son acciones u omisiones que deben ser respetadas y acatadas por todos aquellos que forma su grupo de trabajo.

   Las reglas hacen referencia a una forma exacta de comportamiento o políticas que deben ser cumplidas para el buen funcionamiento del grupo de trabajo.

g) **Definición del programa de trabajo.** Los programas son conjuntos de metas, políticas, procedimientos, reglas, asignación de tareas y definición de los pasos a seguir para llevar a cabo las acciones que han sido definidas en el transcurso de dicha planificación.

   También es importante definir, que dentro del programa de trabajo de un buen administrador de proyectos, deben existir los programas de apoyo o de contingencia, los cuales ayudan con la elaboración y ejecución exitosa de dicho plan, al igual que todas aquellas acciones que no han sido previstas o que por algún motivo no resultan de la forma planeada, y que deben ser atendidas de inmediato, antes que el programa de trabajo sea perdido por dicho administrador.

h) **Elaboración y definición del presupuesto de trabajo.** La formulación de un presupuesto dentro de un proyecto de *software*, normalmente debe ser definido por parte de los administradores de más alto nivel dentro de dicho proyecto.

Es la formulación de un programa de trabajo descrito anteriormente, con la única diferencia que en el presupuesto siempre se habla de cifras o costos del proyecto. Esto quiere decir, hablar de horas-hombre, producción por grupo de trabajo, etc. y cualquier otro término que pueda ser medido.

Dentro de una planificación eficaz y eficiente, existen varios pasos, que deben ser definidos por el administrador del proyecto de *software*, con mucha cautela y énfasis para lograr que su planificación sea el pilar de dicha administración, estos pasos son:

a) **La atención de oportunidades.** Es utilizada por los administradores o jefes de proyectos de *software* como un punto de identificación de oportunidades y debilidades que presenta su grupo de trabajo o la empresa donde se ha de desarrollar el *software*, definiendo dichas oportunidades y debilidades en forma clara e identificar también sus fortalezas para sacar provecho de las mismas, en el desarrollo de dicho proyecto de *software*.

b) **Establecimiento de los objetivos.** El punto principal de una buena administración de proyectos de *software*, está basada en la definición de objetivos en forma descendente, esto implica definir objetivos desde la cima del grupo de trabajo que ha de conformar nuestro desarrollo y posteriormente definir dichos objetivos para cada unidad o grupo de trabajo inferior, hasta cubrir todo el personal involucrado en el mismo.

   Dentro de estos objetivos se debe especificar cuales son los resultados esperados de cada persona o grupo de trabajo, al cual le fue asignada la tarea o tareas correspondientes.

Dicho de otra maneja, los objetivos forman jerarquías dentro del grupo de desarrollo, de la misma forma en que se recolectan resultados, sólo que en forma ascendente.

c) **Definición y desarrollo de premisas.** Definir premisas dentro de un proyecto de *software* es definir el ambiente en el cual se ha de desarrollar el programa elaborado con anterioridad y definir aquellos casos que se supone establece las condiciones donde ha de desarrollarse. Esta premisas también implica informar a todas aquellas personas involucradas en dicho programa, y la forma en que han de ejecutarse dichas acciones.

d) **La determinación de cursos alternativos.** Dentro de un proyecto de *software*, normalmente surgen imprevistos o situaciones no deseadas, las cuales deben ser atendidas en forma inmediata. La forma de atención a estas situaciones imprevistas deben hacerse por medio de la definición de cursos alternativos, especialmente en aquellas tareas en las cuales las acciones no son perceptibles a primer vista.

e) **Evaluación de los cursos alternativos.** Normalmente encontrar cursos alternativos no es el problema, el problema se presenta en definir una lista pequeña de aquellos cursos que son los más apropiados a seguir cuando surge algún imprevisto, y la evaluación de los mismos se realiza examinando las ventajas y desventajas que presenta cada uno de ellos a la situación que se vive en ese momento. Este examen se puede realizar teniendo siempre como base las premisas y metas que fueron definidas dentro del proyecto de *software*.

f) **Selección de un curso alternativo.** Este es uno de los puntos más cruciales para todo administrador de proyectos de *software*, ya que es aquí donde se toman las decisiones y rutas a seguir dentro del proyecto.

   Normalmente la evaluación de cursos alternativos arrojan dos o más alternativas que son la óptimas para la situación que se vive, y es el administrador del proyecto de *software* el encargado de decidir cual de ellas ha de utilizarse e implementarse.

g) **Formulación de planes de apoyo.** Cuando se ha tomado la decisión del curso alternativo, el siguiente paso del administrador del proyecto es formular planes de apoyo, para asegurarse que el plan elegido sea cumplido y finalizado a cabalidad.

h) **Traslado de planes al presupuesto.** Después de tomada una decisión del curso alternativo a seguir, únicamente queda integrar dicho curso al presupuesto del proyecto de *software,* para que éste pueda ser integrado dentro de gastos y costos del proyecto.

1.9 Estrategias, políticas y premisas de un proyecto de *software*

Existe gran diferencia entre el desarrollo de una buena planificación y la instrumentación y práctica de la misma, por eso, es importante dar énfasis a los puntos que se numeran a continuación para realizar una instrumentación exitosa de la planificación:

a) **Comunicar la planificación a administradores o colaboradores claves.** Desarrollar una buena planificación, también implica hacer saber a todas las personas involucradas en dicha planificación sobre la forma en que ha de afectarlos y como éstos pueden colaborar para lograr dicha planificación con éxito.

b) **Desarrollo de premisas de planeación.** El administrador del proyecto de *software* debe ser el encargado de realizar las premisas cruciales y más importante, y darlas a conocer a todo el personal involucrado, girar instrucciones sobre el desarrollo de los programas creados y toma decisiones que los afecta directamente.

   Se debe tener cuidado de no realizar premisas sobre supuestos básicos de condiciones, ya que esto puede llevar a que las decisiones se basen en supuestos y predilecciones personales o convencionales.

c) **Que los planes de acción contribuyan a los objetivos.** Es tarea del administrador de proyectos de *software* corroborar y estar seguro de que todos los planes de acción que han sido creados e implementados contribuyan en forma directa a los objetivos que fueron identificados y acordados en forma inicial. Esto implica que todas las personas que están involucradas en dicho proyecto conozcan de forma clara los objetivos para que estén seguros que sus acciones en verdad están dando frutos y colaborando con el progreso del mismo.

d) **Revisar regularmente la planificación.** Para que una planificación pueda ser exitosa es necesario evaluarla en forma constante. Esta planificación debe reflejar siempre, la situación actual y el lugar donde se dirige todo el esfuerzo de desarrollo.

Por este motivo, toda planificación puede en algún momento ser obsoleta, por este motivo el administrador del proyecto debe estar siempre atento a las variaciones que ésta pueda sufrir para mantener siempre la óptima planificación dentro de su proyecto, y en forma paralela mantener informado de dichos cambios a todo el personal involucrado en el mismo.

e) **Desarrollo de programas de contingencia.** Como se explicó en el inciso anterior, normalmente toda planificación está expuesta a cambios inesperados dentro de la programación original, por lo cual un buen administrador de proyectos de *software* siempre debe formular y establecer planes que puedan ayudar o disminuir la incertidumbre que se cuenta en la elaboración de su planificación, y la implementación de la misma.

f) **Modificar la estructura organizacional a la planeación.** La estructura de la organización debe ser orientada por parte del administrador de proyectos de *software* a un ambiente donde se permita al personal involucrado dentro del proyecto, cumplir con facilidad sus tareas y tomar las decisiones necesarias para la puesta en práctica de sus obligaciones.

El administrador del proyecto debe dar mayor énfasis en la necesidad de delegar la responsabilidad de una tarea específica a una única persona, la cual ha de ser la encargada del cumplimiento de dicha tarea o la encargada de dirigir a su personal para que esta tarea pueda ser cumplida.

Dentro de este punto, se debe tomar en cuenta la buena definición de puestos organizacionales y del equipo de trabajo de desarrollo, para no caer en redundancia o doble trabajo y no cubrir luego con tareas que fueron obviadas desde el inicio.

g) **Planeación permanente.** Por muy buena que sea una planificación y por mucho que se trate de lograr el cumplimiento de dichos objetivos, siempre se debe tener una actitud de perseverancia y estar constantemente supervisando e insistiendo por que se cumpla con lo programado. Toda nueva tarea debe ser incluida dentro del programa de planeación, ya que esto no dará margen a que tareas que han sido planificadas se dejen en el abandono y que la nueva tarea cuente con el apoyo e importancia que se merece.

h) **Clima empresarial de planeación.** Este punto es de suma importancia para el administrador de proyectos, puesto que él es el encargado de crear un ambiente donde todos se sientan comprometidos al cumplimiento de la planificación y tareas que les han sido encomendadas, y que no reciban ninguna distracción de los problemas que los han de afectar y que no son parte del proyecto de *software*.

1.10 Toma de decisiones dentro del proyecto de *software*

Dentro de la planificación de un proyecto de *software*, como ya se ha mencionado, la toma de decisiones es uno de los puntos más importantes y cruciales, pudiendo decir que la 'toma de decisiones' es la esencia de la planeación. Teniendo en cuenta esta definición se puede determinar a la toma de decisiones como:

a) El establecimiento de premisas claras.

b) Identificación de alternativas.

c) Evaluación de dichas alternativas.

d) Elección de una alternativa, la toma de decisión, propiamente dicha.

Para la toma de una decisión, la cual pueda considerarse como racional o lógica, dentro de los términos que esta definición permite, ya que la incertidumbre en que vive un administrador y a la cual está sujeta sus decisiones, no siempre se pueden considerar lógicas aún más con el paso del tiempo.

Uno de los puntos más importantes para tomar una decisión es la búsqueda de la alternativa óptima o la alternativa que ayude a alcanzar las metas y objetivos que fueron planteados desde un inicio, siempre y cuando estén orientadas a la obtención de las premisas definidas.

Para tomar una buena decisión, se deben considerar dos puntos muy importantes, los cuales son:

a) **El factor limitante.** El cual se define como un 'algo' que se ha de interponer en el camino del cumplimiento de un objetivo deseado. La percepción de dichos objetivos es una situación que permite restringir la búsqueda de alternativas y desechar aquellas en la cual dicho factor limitante es muy grande o apreciable en gran magnitud.

b) **El principio del factor limitante.** Este principio nos indica que la selección del mejor curso de acción, o la mejor decisión a tomar, debe identificar y trascender de los factores que más firmemente se oponen al cumplimiento de una meta u objetivo.

El siguiente punto de un buen administrador de proyectos de *software*, es evaluar y seleccionar aquellas alternativas cuya contribución resulten ideal es para el cumplimiento de los objetivos planteados al inicio del proyecto.

El primer punto que debe evaluar un administrador de proyectos son los factores “cuantitativos y cualitativos”.

El factor “cuantitativo” son todos aquellos factores que pueden ser medidos en términos numéricos, como el tiempo de desarrollo y duración de un proyecto, los costos fijos, etc.

El factor “cualitativo” son todos aquellos factores difíciles de medir numéricamente, por ejemplo, las relaciones interpersonales del grupo de trabajo, reacción o forma de reacción de los usuarios del *software* sobre cambios realizados al mismo, etc.

El segundo punto que debe evaluar un administrador de proyectos, es el “análisis marginal”, el cual implica la medición de producción adicional en el proyecto de *software* sobre costos adicionales de los mismos, es decir, que tanto ayuda a la producción de *software* el incremento de personal o equipo al proyecto. Este punto también puede ser utilizado para la simulación de incremento o disminución de personal, el cual estaría indicando como llegarían a influir estas decisiones dentro del proyecto.

El tercer punto a evaluar por el administrador de proyectos es el “costo-beneficio” de una alternativa. Esta técnica de evaluación trata de identificar en forma clara y sencilla la proporción de beneficios y costos dentro de un proyecto de *software*.

En la toma de decisiones de un administrador de proyectos de *software*, existen tres enfoques diferentes que lo pueden guiar para la toma de la decisión correcta, tomando como base los puntos descritos en la sección anterior. Estos tres enfoques son los siguientes:

a) **La experiencia.** La experiencia para una buen administrador es una de las habilidades y virtudes más importantes que puede poseer, y que normalmente lo ignora. A través de la experiencia se pueden confrontar los aciertos y los errores y sacar provecho de ello. Es de suponer que un administrador normalmente se guiará por este enfoque, ya que mientras más experiencia tenga en el campo del desarrollo de *software* mejor será la capacidad en el punto.

b) **La experimentación.** Otro enfoque de decisión que puede ser tomado por un administrador de proyectos es la experimentación de alguna alternativa, basada en la implementación se puede ver su ejecución y desarrollo de la misma.

   Normalmente se dice que es la única forma en que un administrador puede estar seguro de que un plan es correcto, tomando en cuenta también la incertidumbre de todos los factores intangibles que están involucrados en dicha decisión.

c) **La investigación y análisis.** Este enfoque es una de las técnicas más efectivas que puede ser utilizada por un administrador de proyectos, ya que se enfocan primero en el conocimiento a profundidad del problema. Implica también la relación entre variables internas del proyecto y el posible comportamiento que puedan tener las mismas.

Éste se basa en la resolución de problemas por medio de la división y estudio de los diversos factores cuantitativos y cualitativos que pueden estar involucrados dentro del desarrollo de proyecto. Por último, uno de los pasos más importante de este enfoque lo constituye la elaboración de un modelo que simule el comportamiento de la solución planteada para el problema.

Hay que recordar que la mayoría de las decisiones son tomadas por los administradores de proyectos de *software* en medio de la incertidumbre, por los cuales existen dos métodos que pueden colaborar con dicha toma de decisión en estas condiciones, estos métodos son:

a) **Análisis de riesgo.** Este método propone plantear una decisión sobre la base de la iteración de diversas variables importantes dentro de la decisión a tomar, donde dichas variables poseen una alto grado de incertidumbre, pero también un alto grado de probabilidad de dicho comportamiento propuesto.

b) **Árboles de decisión.** Este método permite determinar al menos las principales alternativas y el hecho de que decisiones subsecuentes pueden depender de acontecimientos futuros. Como cada árbol contiene la probabilidad de ciertos acontecimientos, se puede deducir la probabilidad real que una decisión conduzca a los resultados deseados.

2. ORGANIZACIÒN DE UN PROYECTO DE *SOFTWARE*

2.1 Estructura organizacional del proyecto

Para realizar un trabajo eficaz y eficiente, que es el objetivo primordial de la administración, debe existir por parte del administrador del proyecto de *software* una definición clara y precisa de la forma de organizar su personal y grupos de trabajo para lograr en forma exitosa la finalización del proyecto a su cargo.

Aunque muchas veces se dice que no es necesario la definición de una organización formal, el propósito primordial de dicha organización es crear una conciencia generalizada de la necesidad de cooperación para el logro de los objetivos propuestos dentro del presente proyecto.

Para que exista una 'función organizacional', debe existir y poseer un significado para los individuos involucrados en el mismo, y debe constar de:

a) Objetivos verificables.

b) Una idea clara de los principales deberes o actividades implicadas.

c) Una área discrecional o de autoridad precisa.

También se debe tomar en cuenta el suministro de información, que se considera necesaria e indispensable para su cumplimiento.

De esta definición, podemos decir que la organización de un proyecto de *software* debe constar de:

a) Identificación y clasificación de actividades requeridas.

b) Agrupación de las actividades necesarias para el cumplimiento y alcance de los objetivos planteados.

c) Asignación de cada grupo de actividades, y un administrador o jefe de proyectos, dotado de la autoridad necesaria para supervisar dichas actividades.

d) La estipulación de coordinación horizontal (dentro de un mismo nivel organizacional) y vertical (entre oficinas, divisiones y un departamento), en la estructura organizacional.

Una estructura organizacional debe ser diseñada por el administrador del proyecto para estructurar y determinar quién o quiénes deben realizar, las tareas asignadas y quien será el responsable de que estas tareas den los resultados deseados, y así, eliminar los obstáculos al desempeño que resultan de la confusión e incertidumbre respecto de la asignación de las actividades, y tender redes para la toma de decisiones y comunicación que respondan y sirvan de apoyo a los objetivos buscados.

De la misma forma hemos de definir a una ”organización formal” (la cual es el punto que busca el administrador del proyecto), como la estructura intencional de funciones en un grupo de trabajo formalmente organizado.

Para que dicha organización proceda de forma correcta, el administrador debe generar una estructura que ofrezca las mejores condiciones para la contribución eficaz del desempeño individual de los integrantes, y que busque ha futuro los objetivos y metas grupales de quienes lo integran.

Dicha 'organización formal' debe ser flexible, y por lo tanto permitir discrecionalidad, la utilización de talento creativo y los reconocimientos de gustos y capacidades individuales de las personas que integran dicha organización.

Uno de los puntos más importantes dentro de la organización de un proyecto de *software* es establecer departamentos o grupos de trabajo, los cuales se pueden definir como el área, división o sucursal en particular de una organización sobre la cual un administrador posee autoridad respecto al desempeño de las actividades que se realizan en ella.

Como se ha explicado, el propósito de la organización de un proyecto, es volver eficaz y eficiente la cooperación del personal involucrado dentro del proyecto de *software*, y la razón por la que existen niveles dentro de la organización (llamado también "tramo administrativo"), es porque existe un límite para el número de personas que un administrador, jefe de proyectos o jefe de área, puede supervisar en forma efectiva. Un tramo de administración amplio se asocia con un número reducido de niveles en dicha organización y un tramo estrecho o pequeño se asocia con muchos niveles de organización.

Para determinar un "tramo administrativo" eficaz se debe tener en cuenta que el número de subordinados que un administrador puede supervisar eficazmente depende de la importancia y la dificultad de las tareas de dichos subordinados.

Además de estos factores, también el administrador del proyecto o jefe que este a cargo del tramo administrativo debe contar con cualidades personales que ayuden a dicha tarea y siendo la principal cualidad, la reducción del tiempo que se debe de dedicar a los subordinados.

a) Tramos administrativos estrechos

Escasa o nula capacitación, mala o inadecuada delegación de autoridad, planes de operación ambiguos, objetivos no verificables, técnicas de comunicación deficientes o inapropiadas, interacción ineficaz o ineficiente entre el administrador y sus subordinados, reuniones ineficientes, administradores incapacitados, tareas complejas, negativa de los subordinados a tomar responsabilidad y riesgos razonables.

b) Tramos administrativos amplios

Capacitación completa de los empleados, delegación de tareas claras, planes bien definidos, objetivos verificables, técnicas de comunicación apropiadas, adecuada estructura organizacional, interacción eficaz y eficiente entre el administrador y su grupo de trabajo, reuniones eficientes, administradores competentes, tareas sencillas, disposición de los subordinados a asumir su responsabilidad y tomar riesgos.

No cabe duda que la forma óptima y deseada por un administrador de proyectos de *software* es un “tramo administrativo” plano, el cual no siempre se puede obtener por importantes restricciones. Esto se puede dar ya que un administrador simplemente puede tener más subordinados de los que puede dirigir eficientemente, aunque éste delegue autoridad, formule planes eficientes y adopte técnicas eficientes de control y comunicación.

También se ve afectado por el crecimiento del proyecto de *software*, lo cual implica mayor cantidad de personal a su cargo, lo que ha de obligarlo a incrementar en número de niveles en dicho ‘tramo administrativo’.

En la mayoría de los casos la solución óptima es la reducción de ‘tramos administrativos’ y la reducción de niveles organizacionales, aunque no siempre el sistema de administración o simplemente la necesidad del proyecto de *software* lo permita, por lo tanto la forma inversa sería la adecuada, lo importante es balancear en su totalidad los costos que implica la utilización de una u otra técnica, no sólo en los costos financieros del presente proyecto, sino la moral y el desarrollo personal de cada una de las personas involucradas en dicho proyecto.

También es importante para el administrador de un proyecto de *software* poseer un ‘espíritu emprendedor’, el cual debe reflejar una actitud innovadora y un cambio orientado a metas, para la utilización del potencial de una empresa.

Es de suma importancia crear un ambiente para el eficaz y eficiente cumplimiento de metas grupales, así como promover oportunidades para que los subalternos emprendedores utilicen su potencial de innovación y superación. Finalmente, los emprendedores o innovadores necesitan cierto grado de libertad y autoridad para que puedan culminar o cumplir con sus objetivos y metas.

Dentro de los procesos a los cuales está sujeto un administrador de proyectos de *software*, y sobre todo a lo atento que debe estar para utilizar dichos procesos en beneficio de sus objetivos y metas.

La “reingeniería”, la cual podemos definir como el recomienzo o reinicio del proyecto de *software*, es decir, “si hoy volviera a iniciar este proyecto de *software* con el conocimiento y tecnología que se posee actualmente, ¿Cómo sería?, ¿Qué haría?, ¿Qué no haría?”.

También se puede definir la reingeniería como el proceso fundamental y rediseño radical de todos aquellos procesos administrativos y empresariales que son utilizados para la obtención de drásticas mejoras en las medidas críticas y contemporáneas de desempeño.

Este proceso es de suma importancia al administrador de un proyecto de *software*, ya que se puede utilizar esta técnica cuando sea conveniente. Por ejemplo, cuando es el enfoque prestado al presente proyecto, se implementen luego cambios que se consideran necesarios y no aplicar otras técnicas más drásticas que sean consideradas para mejorar los procesos actuales.

2.2 Autoridad dentro del proyecto

Antes de entrar en detalle sobre la autoridad de un proyecto de *software* y la forma correcta que se debe manejar la misma, se definirá dos conceptos muy importantes, los cuales son:

a) **Poder.** Es la capacidad de individuos o grupos de inducir o influir en las opiniones o acciones de otras personas o grupos.

b) **Autoridad.** Es el derecho propio de una puesto (y de la persona que lo ocupa), a ejercer discrecionalidad en la toma de decisiones que afectan a otras personas.

Dentro de la definición de poder, también se puede ampliar un poder que influye mucho dentro de un proyecto de *software*, el cual es el poder de "experiencia", que poseen aquellas personas que pueden influir dentro del actuar de su personal o de sus colegas y el cual debe ser controlado muy de cerca por los administradores de proyectos de *software*, para que no se mal entienda dicho poder que pueden poseer alguna personas y así evitarse problemas mayores en el transcurso de dicho proyecto.

Dentro de este concepto de poder y autoridad, un administrador de proyectos de *software* debe considerar el uso del *empowerment*, que autoriza a los administradores, analistas o programadores de dicho proyecto, el poder para tomar decisiones sin tener que requerir la autorización de sus superiores.

Esto quiere decir que quienes se hallan directamente relacionados con la tarea que hay que cumplir, sean los más indicados para tomar una decisión al respecto, siempre que éstos tengan las aptitudes necesarias para resolver dicha tarea.

Esta técnica de *empowerment*, implica que todos los empleados del proyecto que utilicen esta técnica acepten la responsabilidad de las acciones que fueron utilizadas para resolver la tarea que tenían asignados. En forma conceptual podemos definir esta técnica de autoridad como:

a) El poder es proporcional o igual a la responsabilidad que posee cada uno de los implicados dentro del proyecto de *software*.

b) Si el poder es mayor que la responsabilidad, el resultado obtenido en esta técnica de autoridad sería una conducta autocrática para el personal que la aplique, ya que ellos no serían responsables de sus acciones.

c) Si la responsabilidad es mayor que el poder, el resultado de autoridad sería la frustración del equipo de desarrollo del proyecto, ya que ellos carecerían de poder necesario para desempeñar las tareas de la cual son responsables y participes.

Una de las técnicas más importantes y que tienen más éxito dentro de la administración de un proyecto de *software* es la descentralización de la autoridad. Esta técnica conlleva varios pasos que serán analizados a continuación:

a) Su naturaleza.

b) Tipos de centralización.

c) Descentralización como filosofía y política.

Otro punto que es importante y que influye en la forma en que la autoridad se distribuye y maneje dentro de la administración del proyecto de *software*, es la delegación de autoridad, que se manifiesta cuando un superior concede discrecionalidad a un subordinado para la toma de decisiones. Esta delegación de autoridad conlleva varios pasos los cuales se describen a continuación:

a) La determinación de los resultados esperados de un puesto.

b) La asignación de tareas a ese puesto.

c) La delegación de la autoridad para el cumplimiento de dichas tareas.

d) La responsabilidad de la persona que ocupa ese puesto con respecto al cumplimiento de dichas tareas.

Dentro de un proyecto de *software* es imposible separar las diversas partes de este proceso, ya que sería injusto esperar que una persona cumpla ciertas metas sin dotarlo de autoridad para que el pueda lograrlo, como también el delegarle autoridad a una persona sin indicarle cuales son los resultados finales en función de los cuales se le habrá de utilizar.

También la delegación de autoridad resulta ineficaz dentro de un proyecto de *software*, cuando los administradores o jefes de proyecto no conocen los principios y naturaleza de los mismos, porque se resisten o son incapaces de aplicarla. Existen varios pasos que son necesarios para realizar una delegación de autoridad eficiente, éstos son:

a) **Receptividad.** Una de los atributos que debe de poseer un buen administrador de proyectos de *software* que delega autoridad, es el conceder a otras personas la práctica de sus ideas. Ya que la toma de decisiones nunca será la misma, puesto que un subordinado puede tomar decisiones diferentes a las que tomaría un administrador de proyectos o jefe de proyectos.

b) **Disposición de ceder.** Para efectuar una delegación de autoridad eficiente, un administrador o un jefe de proyectos debe estar conciente en la necesidad que tiene de ceder autoridad y toma de decisión a sus empleados. Ya que uno de los grandes errores de los administradores que son ascendidos dentro de un proyecto de *software* es tomar decisiones que ya no le competen o corresponde.

   De este modo, los administradores de proyectos realizaran mayores contribuciones al proyecto de *software*, si se dedican a realizar el trabajo que les corresponde y por el cual poseen el puesto de administradores o jefes de proyecto, y delegar las tareas, a pesar de que ellos podrían realizarlas mejor que los subordinados.

c) **Disposición de permitir que los demás cometan errores.** Dentro de la delegación de autoridad que realiza un administrador de proyectos de *software*, siempre debe de estar presente permitir que los subordinados comentan errores, y tomar esto como una inversión del desarrollo personal de su equipo de trabajo.

d) **Disposición de confiar en los subordinados.** Para conseguir una eficiente y eficaz delegación de autoridad, el administrador del proyecto de *software* debe confiar en sus subordinados, por este motivo es importante que los administradores elijan a personas debidamente preparadas para asumir este tipo de responsabilidad.

e) **Disposición a establecer y aplicar controles amplios.** Cuando un administrador de un proyecto de *software* delega autoridad, debe estar dispuesto a buscar y utilizar medios para la obtención de retroalimentación que le sirva para cerciorarse efectivamente de que se hace uso correcto de la autoridad que se delegó y así prestar apoyo a las metas y planes del proyecto.

Normalmente dentro de proyectos de *software* se pueden tener administradores que gusten o sean contrarios a la descentralización de la autoridad, y que normalmente sean ellos los que tomen las decisiones. Aunque muchas veces el temperamento o personalidad de un administrador o jefe influya en el grado de delegación de autoridad, también existen otros factores que afectan esta decisión, por ese motivo se presentan las siguiente sugerencias prácticas que facilitan la delegación de autoridad exitosa:

a) **Defina asignaciones y delegue autoridad en vista de los resultados esperados.** Esto implica al administrador del proyecto de *software* delegar la suficiente autoridad para que sus subordinados puedan cumplir sus metas y tareas de una manera eficiente.

b) **Seleccione a cada persona de acuerdo con el trabajo por realizar.** El administrador del proyecto o jefe de proyecto debe estar conciente de que no todas las personas están capacitadas a realizar todos los trabajos, por lo tanto es necesario seleccionar de forma adecuada a cada persona para que realice una tarea en particular.

c) **Mantenga abiertas las líneas de comunicación.** El administrador debe establecer una línea directa de comunicación entre él y el subordinado al cual ha delegado la tarea y tiene la responsabilidad del cumplimiento de la misma. Esto se realiza con el fin de que dicho subordinado tenga la información necesaria para tomar decisiones y pueda interpretar correctamente la autoridad que se le ha delegado.

d) **Establezca los controles adecuados.** El administrador del proyecto está obligado a establecer controles que favorezcan realmente la delegación de la autoridad y que deben ser relativamente amplios y estar diseñados para mostrar de forma correcta las irregularidades en el cumplimiento de los planes asignados y así no interferir en las acciones diarias de los subordinados.

e) **Recompense la delegación eficaz y la exitosa asunción de autoridad.** Un buen administrador de proyectos, siempre debe de estar permanentemente atento a los medios para premiar la delegación de la autoridad.

Aunque muchas veces esta recompensa se puede expresar en carácter monetario, también la discrecionalidad y prestigio es a menudo uno de los incentivos más apreciables por parte de los subordinados.

En resumen, se puede decir que existen varios elementos fundamentales que deben ser tomados en cuenta para la buena organización de una empresa, los cuales son:

a) La estructura debe ser un reflejo de los objetivos y metas que deben de ser alcanzados en el proyecto de *software*. Dado que las actividades que se han de realizar dentro de dicho proyecto, siempre deben ser enfocadas y derivarse de los mismos.

b) Dicha estructura debe ser reflejo de la autoridad que se presenta en el actual proyecto de *software*.

c) La estructura organizacional de un proyecto de *software* también debe responder a las condiciones en que dicho proyecto se encuentra en un determinado momento.

Por último, se puede decir que no existe una estructura organizacional única, y que ésta debe ser determinada en forma efectiva dependiendo de la situación que prevalece en ella.

3. INTEGRACIÒN DE PERSONAL DE UN PROYECTO DE *SOFTWARE*

3.1 Selección del recurso humano

Definición de integración de personal. Esta función administrativa consiste en ocupar y mantener los puestos o cargos dentro de una estructura organizacional. Esto quiere decir, que se debe mantener los puestos asignados dentro de la definición organizacional de la empresa, con personal altamente calificado para cada una de las áreas que éste contenga.

Esto se logra mediante la identificación de los requerimientos de fuerza de trabajo, el inventario de personas disponibles y el reclutamiento, selección, contratación, ascenso, evaluación, planeación de carreras, compensación y capacitación o desarrollo de los candidatos, como de los empleados en funciones a fin de que puedan cumplir eficaz y eficientemente sus tareas.

Podemos aplicar esta definición a nuestro proyecto de *software* diciendo que tendremos una organización definida para la elaboración y culminación de dicho proyecto, de este punto, debemos decidir y obtener al personal más calificado para que pueda ocupar cada uno de los puestos que han sido asignados.

Factores que influyen en el número de personal requerido. No solamente se da por el tamaño del proyecto, sino también por la complejidad de la estructura de la organización, los planes que se tienen de hacerlo más grande, el índice de rotación de personal que se ha de tener dentro del proyecto de *software.*

Por lo tanto se puede decir que esta cantidad de personal no sigue ley alguna.

También está cantidad esta determinada por las aptitudes que exige cada puesto a fin de que sea posible elegir al personal más indicado.

Factores que influyen en la integración de personal. Entre los más importantes se encuentra el factor externo, como las actitudes imperantes en la sociedad, al igual que las leyes y reglamentes de trabajo, las condiciones económicas y las condiciones de oferta y demanda.

El factor interno es otro factor que afecta la integración de personal dentro de un proyecto de *software*, entre los puntos que afectan este factor se puede mencionar: las metas del proyecto de *software* (haciendo referencia al tamaño del proyecto, tiempo de desarrollo dentro de la empresa contratista, etc.), las tareas que se deben realizar dentro del proyecto, la tecnología utilizada (ya que comúnmente se pueden definir como extremos, si se tiene una tecnología muy avanzada talvez habrá necesidad de obtener más personal, lo mismo sucederá si se tiene una tecnología muy obsoleta), la estructura interna, los tipos de personas empleadas, la demanda y oferta del personal dentro del proyecto de *software*.

Ahora se estudiará dichos factores con más detalle:

a) **Ambiente externo.** Los factores que existen en el ambiente externo influyen en la integración del personal del proyecto de *software*. Éstas están dictadas por restricciones u oportunidades educativas, socioculturales, político-legales y económicas.

Por ejemplo, si dentro del proyecto se utiliza tecnología de punto o reciente, lo mismo obliga a la necesidad de tener que contratar a personal con educación y experiencia mucho más avanzada y formal. También la influencia sociocultural afecta al entorno del proyecto de *software*, ya que el mismo rige la forma de aptitud y carácter de cada personal dentro del proyecto, haciendo que éste se involucre más y más.

Otro factor importante son las condiciones económicas (hasta la competencia), la cual hace más exigente la oferta y la demanda entre el personal con el que se cuenta y también con el personal que se busca para que forme parte del proyecto de *software*. Otra restricción con que se cuenta es la legal y política, la cual exige a las empresas de desarrollo de *software* el cumplimiento de ciertas leyes y normas establecidas en ciertos niveles organizacionales.

b) **Ámbito interno.** Existen varios factores internos que afectan la integración del personal, entre estos podemos mencionar:

- Promoción interna. La promoción interna originalmente se suponía de la ascensión dentro de puestos del grupo de desarrollo del proyecto, lo cual enfatiza y ayuda a la promoción del personal capacitado, para que poco a poco, con experiencia y esfuerzo alcancen metas y puestos mucho mayores.

Dentro de este punto, se debe tomar en cuenta y manejar con mucho cuidado la forma y política que se ha de seguir para evaluar el ascenso que se ha de ofrecer ya que pueden existir o a partir de éste surgir envidias y rivalidades dentro de las personas que están buscando dicho ascenso.

La promoción interna, no solamente tiene aspectos positivos dentro de la moral de los empleados, sino también el compromiso a largo plazo dentro del grupo de desarrollo, y también puede conservarse el personal que llegue a ser jefes de proyecto en potencia.

- Políticas de competencia abierta. En este punto, el administrador del proyecto de *software*, debe decidir si los beneficios de la política de promoción interna son mayores que su deficiencia. Dado esto puede recurrir a ofrecer las vacantes dentro de su grupo de trabajo, tanto al personal que está a su cargo, como al personal externo, y evaluarlos con criterios honestos y justos para obtener a la persona indicada en su grupo de desarrollo.

  Aunque hay que tomar en cuenta que en esta política el personal interno tiene la ventaja que conoce la forma de trabajo del grupo de desarrollo, con lo que no cuenta el personal externo que busca ocupar la vacante dentro del proyecto de *software*.

- Correspondencia entre personal y puesto. Un buen grupo de desarrollo de *software*, no sólo está marcado por la tecnología, computadoras, *software*, etc. que se ha de utilizar dentro del proyecto, sino dentro del personal que ha de estar involucrado dentro del mismo, por lo tanto se debe tener mucho cuidado cuando se seleccione las personas que han de ocupar la plaza vacantes dentro del grupo de desarrollo, poniendo mayor énfasis en la obtención del personal interno o externo más calificado, y obtener así la persona más indicada para ocupar el puesto en ese momento o en un futuro.

  Es importante dar a comprender y despertar dentro del administrador del proyecto de *software* la responsabilidad que éste tiene con el hecho de identificar los requisitos y analizar el puesto que se necesita dentro de su grupo de trabajo.

- Identificación de requisitos del puesto. Para identificar de forma correcta los requisitos que se necesitan en un puesto, el administrador del proyecto de *software* debe responder a algunas interrogantes como:

  a) ¿Qué se deberá hacer en este puesto?

  b) ¿Cómo se deberá hacer?

  c) ¿Qué conocimientos, actitudes y habilidades se requieren?

  d) ¿Este puesto debe ejercerse de otra manera?

El administrador del proyecto de *software* debe ser muy cauteloso y tratar de no cometer ciertos errores al definir e identificar un puesto, ya que algunas veces se pueden asignar puestos faltantes de tareas y responsabilidades dentro del grupo de desarrollo, lo cual conlleva a un empleado a aburrirse, sentirse insatisfecho, o de forma contraria no debe sobrecargarse en sus tareas ya que sentiría tensión, frustración y pérdida de control.

A continuación se describirán algunas habilidades y características personales que debe evaluar un administrador de proyectos de *software* para poder asignar personal a un puesto vacante:

a) Capacidades analíticas y de solución de problemas

b) Deseo de desarrollo y trabajo

c) Habilidades de comunicación y empatía

d) Integridad y honestidad

e) Antecedentes de desempeño como buen empleado

- Procesos y técnicas de selección de personal. Este proceso de los lineamientos al administrador de un proyecto de *software*, para la instrumentación de la técnica de selección de personal que debe aplicar.
-

Para que dicha selección arroje buenos resultados, éstos deben de ser válidos y confiables, es decir, deben de reflejar efectivamente los datos que se desean medir, o dicho de otra forma es el grado en el que los datos predicen el éxito de un candidato. Y el ser confiable indica la precisión y consistencia de dichas medidas.

Para obtener una evaluación correcta y que de como resultado la persona óptima para el puesto vacante dentro del grupo de desarrollo, el administrador del proyecto de *software* debe evaluar los siguientes pasos:

a) **Establecer los requerimientos vigentes o futuros del puesto.** Estos incluyen requerimientos como estudio, conocimiento, habilidades y experiencia.

b) **El candidato debe llenar una solicitud.** Este paso se puede omitir si el aspirante al puesto es miembro del grupo de desarrollo.

c) **Entrevista preliminar.** Este paso es utilizado para identificar a los candidatos más prometedores.

d) **Pruebas de aptitud.** Éstas son utilizadas, si es el administrador del proyecto lo considera necesario.

e) **Entrevista.** Normalmente es la vía que se le da más importancia dentro de la evaluación de personal, aunque no siempre arroja los resultados más confiables o verídicos de un aspirante, ya que depende de los entrevistadores, cada uno puede interpretar la información obtenida por este medio en diferente manera.

Para mejorar esta técnica el entrevistador (ya sea el administrador o jefe de proyectos, o alguien con un rango mayor dentro de la empresa de desarrollo), debe, en primer lugar, saber qué se busca en la entrevista, debe analizar y estudiar los resultados alcanzados por dicha entrevista. En segundo lugar, debe estar preparado para plantear las preguntas más indicadas. En este caso se pueden presentar tres tipos de entrevistas:

- No estructurada. Donde el entrevistador no tiene un itinerario definido para la entrevista.

- Semiestructurada. Donde el entrevistador sigue un patrón de entrevista, pero no explícita o específica.

- Estructurada. Donde el entrevistador hace preguntas previamente formuladas.

Un tercer medio para la mejora de la selección es realizar múltiples entrevistas con diversos entrevistadores.

f) **Pruebas.** El objetivo de las pruebas es obtener información sobre los aspirantes a la plaza vacante, para evaluar sus habilidades y destrezas. Dichas pruebas se pueden calificar de la siguiente manera:

  - Pruebas de inteligencia. Estás medidas sirven para medir la capacidad intelectual de los individuos y probar su memoria, agilidad mental, identificación de problemas, etc.

- Pruebas de habilidad y aptitud. Persigue el descubrimiento de intereses, habilidades poseídas y potencias para la adquisición de nuevas habilidades.

- Pruebas vocacionales. Están orientadas a ocupar las habilidades más convenientes del candidato.

- Pruebas de personalidad. Están diseñadas para revelar las características personales de los candidatos.

Hay que tomar en cuenta que a pesar de la gran diversidad de métodos y pruebas que existen para la selección de personal, no existe ningún medio perfecto para dicha selección. La experiencia ha demostrado que aunque se utilicen los medios de selección más cuidadosos, nunca se cubren los desperfectos en lo que se refiere a la previsión de desempeño.

3.2 Evaluación del desempeño

La evaluación del desempeño se trata de una de las claves principales para la correcta conducción de la integración de personal. Es la base para determinar quién es susceptible de ser ascendido a un puesto más alto.

- Selección de criterios de evaluación. Por medio de la evaluación de personal se puede medir el desempeño en el cumplimiento de metas y planes de las personas que están involucradas en el proyecto de *software*.

Una de las formas más aceptables para evaluar el desempeño de un integrante del equipo de desarrollo de *software* es la 'verificación de metas'.

Esto implica la fijación de metas inteligentes, planea los programas necesarios para alcanzarlos y logra su efectiva consecución. Aunque muchas veces hay que tomar en cuenta que hay factores que intervienen y que escapan del control de los desarrolladores de *software* o del mismo jefe de proyectos.

Como se dijo anteriormente no solamente podemos evaluar al personal por el cumplimiento de sus metas, ya que muchas veces hay casos que salen del control de los mismo, por lo tanto es importante evaluar al personal del proyecto de *software* por el grado de comprensión y ejecución de las tareas que les han sido asignadas.

- Evaluación del personal en base a objetivos verificables. Este es uno de los métodos más prácticos que se pueden utilizar para la evaluación de personal, éste consta del establecimiento y cumplimiento de objetivos verificables. Consta de dos pasos:

  a) **Proceso de evaluación.** En este primer paso las personas encargadas de la supervisión (podría ser el administrador del proyecto o jefe de proyecto), determinan con base en los objetivos que le fueron impuestos al personal a evaluar, el desempeño y cumplimiento de los resultados esperados en forma correcta.

Para realizar una evaluación correcta, el evaluador debe considerar varios punto: si las metas eran razonablemente alcanzables, si los factores internos o externos facilitaron o dificultaron el cumplimiento de los objetivos.

b) **Tipos de revisión.** Existen tres tipos de revisión para la evaluación del desempeño, los cuales son:

- Evaluación exhaustiva formal. Debe realizarse por lo menos una vez al año o de preferencia un poco más frecuente.

- Evaluación periódica o de avances. Estas revisiones pueden ser breves e informales, pero permiten identificar problemas o barreras que impiden un desempeño eficaz.

- Evaluación permanente. Esta evaluación es una supervisión inmediata, que se da entre el administrador o jefe de proyectos y el subordinado, a fin de ser posible emprender en forma inmediata acciones correctivas para impedir fallas que se puedan convertir en problemas de grandes dimensiones.

- Desarrollo profesional por medio de la evaluación del desempeño. En una evaluación de desempeño deben de identificarse tanto las fortalezas, como las debilidades de cada una de las personas que integran el equipo de desarrollo, lo cual se puede tomar como un punto de partida para el desarrollo profesional del individuo, es decir, que sea capaz de utilizar sus fortalezas para superar sus debilidades y aprovechar las oportunidades profesionales que se le presentan. Este desarrollo profesional esta dado por:

a) Elaboración de un perfil personal.

b) Desarrollo de metas profesionales.

c) Análisis del ambiente: amenazas y oportunidades.

d) Análisis de fortalezas y debilidades personales.

e) Desarrollo de opciones profesionales estratégicas.

f) Prueba de congruencia y elección de estrategias.

g) Desarrollo de objetivos profesionales y planes de acción.

h) Desarrollo de planes de contingencia.

i) Instrumentación del plan profesional.

j) Supervisión del progreso.

3.3 Cambio de personal

Las fuerzas o condiciones de cambios se pueden dar por condiciones externas, del interior del grupo de desarrollo de *software* o por los mismos integrantes de dicho grupo.

Cambios que influyen en el grupo de desarrollo de *software*. Existen motivos o tendencias que obligan al cambio de personal o evolución del mismo dentro del grupo de desarrollo, entre estos se puede citar:

a) La educación debe ser continua. Esto involucra que el aprendizaje de por vida es una necesidad, donde las empresas deben poner especial atención.

b) Aumento al personal con mayor conocimiento y pocas habilidades, dando lugar a la capacitación en conocimiento y aptitudes conceptuales y de diseño.

c) La empresa de desarrollo de *software* debe estar conciente del constante cambio que se presenta en este ámbito, lo cual obliga a dicha empresa a impartir los propios cursos de capacitación.

Para el proceso de cambio existen tres etapas, las cuales son:

a) **Descongelamiento.** Es generada por motivaciones de cambio, ya que la gente se siente insatisfecha con la situación que se vive dentro de la empresa de desarrollo, lo cual es probable que advierta la necesidad de un cambio, aunque hay que tomar en cuenta que en algunos casos los propios miembros de la empresa crean perturbaciones capaces de inducir al cambio.

b) **Movimiento o cambio.** Este movimiento o cambio puede ocurrir por la asimilación de nueva información, nuevos proyectos para el grupo de desarrollo en una perspectiva diferente.

c) **Recongelamiento.** Se da cuando el cambio se estabiliza, lo que implica la identidad y valores de las personas involucradas en dicho cambio.

Dentro de los cambios que pueden surgir en la empresa de desarrollo de *software* puede que exista resistencia esto se puede dar porque:

a) Lo desconocido produce temor e induce resistencia. Por ejemplo una reorganización o evaluación de personal puede causar temor dentro de los empleados de dicha organización.

b) El desconocimiento de la razón de cambio produce resistencia. Puede darse que las personas que serán directamente afectados no le parezca la necesidad de realizar dichos cambio, lo cual puede llevar a la resistencia del mismo, si no existe un conocimiento claro de lo que ha de realizarse.

c) El cambio también puede terminar en la reducción de poder dentro del grupo o empresa de desarrollo, por parte de algunos de los empleados de dicha empresa.

Estos problemas de resistencia al cambio se pueden reducir de alguna manera como:

a) Participación del personal del grupo de desarrollo o empresa de desarrollo.

b) La comunicación de los cambios a realizar y las razones por las que se realizan dichas acciones.

El conflicto dentro de una empresa o un grupo de desarrollo se puede dar entre individuos o grupo; en una persona solamente; entre un individuo y su grupo de desarrollo; dentro de diferentes grupos de desarrollo de aplicaciones; etc. Más que tacharlo de malo, se puede decir que dicho conflicto puede tomarse como beneficioso, ya que es capaz de provocar la presentación de un asunto desde diferentes perspectivas.

Puede que existan muchas fuentes donde se generan dichos conflictos, esto puede surgir porque las tareas que se realizan son muy independientes y pueden generar fricciones entre diferentes grupos de desarrollo o dentro del mismo personal.

Estos conflictos pueden darse también por la forma de delegar autoridad dentro del proyecto de *software*, ya que pueden existir estilos de liderazgo diferentes o existir preferencia entre los superiores por algún o algunos subordinados que se encuentran en el mismo nivel organizacional dentro de la empresa o proyecto de *software*.

La forma en que se pueden manejar los conflictos varía y existen varios métodos para contrarrestarlos.

a) Gira en torno a relaciones interpersonales.

b) La segunda se basa en cambios estructurales.

c) La tercera se puede definir como allanamiento o distensión (donde se destacan los puntos de acuerdo y las metas comunes y se resta importancia a los desacuerdos).

d) La cuarta opción puede ser la coacción, la cual es la imposición de las opiniones propias a las demás, lo que puede conllevar resistencia.

e) El último método es la negociación, que busca acuerdos parciales entre opiniones o demandas.

f) Modificación de la conducta.

g) Intervención de personal de más alto rango.

- Desarrollo organizacional. El desarrollo organizacional (DO) es un enfoque sistemático, integrado y planeado para elevar la eficacia de una empresa. Su diseño está enfocado a la solución de problemas que nublan la eficiencia operativa en todos los niveles. Estos problemas se dan comúnmente por la falta de cooperación, excesiva descentralización o deficiente comunicación.

  Algunas veces se recurre con propósitos de desarrollo a la formación de equipos, consultoría de procesos, enriquecimiento de puestos, modificación de la conducta organizacional, diseño de puestos, manejo de tensiones, planeación de la trayectoria profesional y administración de objetivos.

- Proceso del desarrollo organizacional. El desarrollo organizacional, como ya se dijo, es el proceso de enfoque situacional o de contingencia para elevar la eficacia de la empresa. Los pasos que se han de seguir para lograr el desarrollo dentro de la empresa de *software* está delimitado a seis:

  a) Identificación del problema.

  b) Diagnóstico de la organización.

c) Retroalimentación.

d) Desarrollo de estrategias de cambio.

e) Intervenciones.

f) Medición y evaluación.

Las últimas tres fases concluyen el ciclo de vida del desarrollo organizacional, estas deben convertirse en cierta forma como proceso continuo, cuyo propósito es volver más efectiva a la empresa de desarrollo.

4. DIRECCIÒN DE UN PROYECTO DE *SOFTWARE*

4.1 Motivación

La motivación se puede definir en términos genéricos como una amplia serie de impulsos, deseos, necesidades, anhelos y fuerzas similares. Esto significa que los administradores o jefes de proyecto motiven a su grupo de desarrollo para realizar cosas con las que se espera satisfacer dichos impulsos y deseos e inducir a este grupo a actuar de determinada manera.

Dentro del ámbito de la motivación, existen los motivadores, los cuales inducen a los miembros del grupo a alcanzar un alto desempeño, y también pueden ser vistos como las recompensas o incentivos que intensifican el impulso de trabajo. Por este motivo el jefe de proyectos debe intensificar la motivación, ya sean personales o no, con el establecimiento de condiciones favorables a ciertos impulsos.

La motivación, como ya se dijo es el impulso y esfuerzo para satisfacer un deseo o meta, esta motivación, lleva a un sentimiento llamado “satisfacción”, el cual se refiere al gusto que se experimenta cuan se cumple con algo propuesto.

Por lo tanto es de sumo cuidado para el administrador o jefe de proyectos controlar tanto la motivación, como la satisfacción de su personal, tomando en cuenta que es un gran riesgo que algún miembro de su grupo de desarrollo disfrute de una alta satisfacción en su trabajo, pero que al mismo tiempo cuente con un bajo nivel de motivación para la realización del mismo, o viceversa.

Existen varias teorías que ayudan a aumentar la motivación dentro del personal de un proyecto de *software*, estas son:

a) **Teoría de la jerarquía de las necesidades.** Es una de las teorías más conocidas y concluye que una vez satisfechas las necesidades que abajo se mencionan, estas dejan de ser motivadoras.

- Necesidades fisiológicas. Como alimento, agua, calor, sueño, etc.

- Necesidades de seguridad. Son necesidades para librarse de riesgos físicos y laborales, la propiedad, alimentos, etc.

- Necesidades de asociación o aceptación. Es dada por la necesidad de pertenencia y el ser aceptado por los demás.

- Necesidades de estimación. Son las necesidades de poder, prestigio, categoría y seguridad en uno mismo.

- Necesidades de autorrealización. Se trata de llegar a ser lo que se es capaz de ser, optimizar el propio potencial y realizar algo valioso.

b) **Teoría motivacional de la expectativa.** Esta teoría indica que el grupo de desarrollo se sentirá motivado a realizar cosas en favor del cumplimento de una meta, sí está convencida del valor de ésta y si comprueba que sus acciones contribuirán efectivamente a alcanzarla. Esta teoría se puede deducir de forma matemática de la siguiente forma:

Fuerza = Valencia X Expectativa

Donde la ‘fuerza’ es la intensidad de la motivación del grupo de desarrollo, la ‘valencia’ es la intensidad de la preferencia por un resultado y la ‘expectativa’ es la probabilidad de que cierta acción en particular conduzca al resultado deseado.

c) **Teoría de la equidad.** Es un factor importante en la motivación de un grupo de desarrollo, el hecho que cada uno de los integrantes del mismo perciban como justa o no justa la estructura de recompensa. Esto quiere decir, si existe igualdad el lo que se refiere a juicios subjetivos de la recompensa, en comparación a la recompensa obtenida por él y por los demás. Esta teoría esencial se puede describir así:

$$\frac{\text{Resultados de una persona}}{\text{Insumos de una persona}} = \frac{\text{Resultados de otra persona}}{\text{Insumos de otra persona}}$$

Dando como resultado el equilibro entre relación/insumos de una persona y de la otra.

Si un miembro del grupo de desarrollo considera que ha sido recompensado de manera no equitativa, entonces podrá sentirse insatisfecha, reducirá su calidad y cantidad de trabajo, y lo más seguro es que resulta abandonando el proyecto de *software*.

a) **Teoría del reforzamiento.** Esta teoría indica que un grupo de desarrollo de *software* puede ser motivado median el adecuado diseño de condiciones de trabajo y el elogio de su desempeño. También ayuda a determinar las causas de las acciones de los empleados y después emprender cambios para eliminar áreas problemáticas y obstáculos al buen desempeño y desarrollo del proyecto de *software*. Luego se fijan metas con la ayuda del grupo de desarrollo y participación de los mismos, proponiendo una retroalimentación sobre resultados.

b) **Teoría de las necesidades sobre la motivación.** Existen tres tipos de necesidades, y son de especial importancia para la buena administración de un proyecto de *software*, ya que permiten que nuestro grupo de desarrollo funcione adecuadamente.

- Necesidad de poder. Indica que las personas que tienen gran necesidad de poder, se dedican a ejercer influencia y control. Estos integrantes del grupo de desarrollo buscan posiciones de liderazgo, con frecuencia son conversadores, empeñados, francos, obstinados, exigentes, les gusta enseñar y hablar en público.

- Necesidades de asociación. Los integrantes con necesidad de asociación pueden disfrutar enormemente que se les tenga estimación y tienden a evitar los rechazos de un grupo en general.

- Necesidad de logro. Estos integrantes del proyecto de *software* tienen un intenso deseo de éxito e igualmente de fracaso y son realistas frente a los riesgos.

c) **Técnicas motivacionales especiales.** Existen tres técnicas especiales de motivación:

- Dinero. El dinero nunca debe ser pasado por alto como motivador, ya sea en forma de salario, bono, compensación, etc., ya que éste puede dar categoría y poder. Primero puede que sea más importante para una persona que empieza una carrera o familia, que quién ya lo tiene; segundo, ver el dinero como la facilidad de encontrar personal adecuado para nuestro equipo de desarrollo; tercero, tener el cuidado de remunerar correctamente a nuestro equipo de desarrollo conforme la remuneración en el exterior; cuarto, la compensación equitativa dentro del grupo de desarrollo.

- Participación. Esta técnica está enfocada en el sentido que a todo el personal de desarrollo se le tome en cuenta, dentro del grupo de trabajo. También la participación es un buen tipo de reconocimiento, y apela a la necesidad de asociación y aceptación, y genera a los individuos una sensación de logro.

- Calidad de vida laboral. Esta motivación se enfoca al compromiso de la empresa de desarrollo, de proporcionar un promisorio avance en el terreno del enriquecimiento del puesto a cada uno de los empleados de desarrollo con los que se cuenta.

4.2 Liderazgo

Se puede definir como liderazgo el arte o proceso de influir en las personas que se esfuercen voluntaria y entusiastamente en el cumplimiento de metas grupales. Lo ideal se daría no sólo alentar al grupo de desarrollo, sino despertarles la disposición a trabajar y hacerlo con ahínco y seguridad en sí mismos.

- Componentes del liderazgo. Existen cuatro componentes importantes dentro del liderazgo, los cuales son:

  a) Capacidad de hacer un uso eficaz y responsable del poder.

  b) La capacidad para comprender que los seres humanos tienen diferentes motivación en diferentes momentos y situaciones.

  c) La capacidad de inspirar a los demás.

  d) La capacidad de actuar en favor del desarrollo.

- Conducta y estilos de liderazgo. Existen tres teorías de estilos de liderazgo, los cuales son:

  a) **Estilo basado en el uso de la autoridad.** Este estilo se basa en hacer uso de la autoridad por parte de los administradores de proyectos de *software*. De los cuales existen tres estilos básicos:

  - Autocrático. Que impone y espera cumplimiento, es seguro y conduce por medio de la capacidad de retener u otorgar premios y castigos.

- Democrático o participativo. Esta definido por la consulta a los miembros del grupo de desarrollo respecto a acciones y decisiones y alienta su participación.

- Liberal. Utiliza muy poco el poder y le concede a los miembros del grupo de desarrollo gran independencia en el sentido de sus tareas y obligaciones.

b) **La rejilla administrativa.** La cual se ocupa sobre el interés que poseen los administradores de proyectos de *software*, tanto por las metas que se deben alcanzar dentro del proyecto, como los intereses del personal a cargo para el cumplimiento de dichas metas. Este método puede brindar cuatro posibles estilos de liderazgo, los cuales son:

- Administración empobrecida. Que se da cuando el jefe de proyectos se interesa poco en su grupo de desarrollo y se involucran en forma mínima en sus funciones y labores.

- Administración en equipo. Son los jefes de proyectos que ponen en sus acciones la mayor dedicación posible tanto a las personas de su grupo de desarrollo como en el logro de las metas del proyecto.

- Administración de club campestre. Son los jefes de proyectos que casi nunca se preocupan por las metas a alcanzar dentro del proyecto y se ocupan más por la situación de su grupo de desarrollo.

- Administración autocrática. Son los jefes de proyectos dedicados únicamente a las metas por alcanzar en el proyecto de *software* y dedican una mínima parte al estado de su grupo de desarrollo.

c) **Liderazgo continuo.** El liderazgo continuo trata de unificar los diferentes estilos de liderazgo que puedan existir dentro de los administradores de proyectos de *software*. Esto conlleva un liderazgo en el cual se encuentra involucrado el administrador, los integrantes del grupo de desarrollo y la situación en que se encuentra el grupo de desarrollo.

En resumen, se puede decir que existen dos tipos de líderes dentro de un proyecto de *software*, los cuales son:

a) **Líderes transaccionales.** Son líderes que identifican las necesidades de su grupo de desarrollo, para que puedan cumplir con los objetivos y las metas de los proyectos de *software*.

b) **Líderes transformacionales.** Son líderes que articulan su visión a la inspiración de su grupo de desarrollo.

4.3 Grupos y toma grupal de decisiones

Podemos definir a los grupos como un conjunto de personas encargada de un asunto. Por esta razón, se puede decir que un grupo es el encargado de un asunto específico dentro del proyecto de *software*, para darle solución o seguimiento a dicho asunto.

Ventajas de un grupo. Existen varias ventajas en el uso de grupos, que promueven la participación éstas son:

a) **Deliberación y criterios grupales.** Es una ventaja tener un grupo para deliberar y formar criterios.

b) **Temor al exceso de autoridad en una sola persona.** Esto conlleva a no centralizar la autoridad en una persona dentro del proyecto de *software*.

c) **Representación de grupos o personal interesado.** Esto es la participación y representación del personal involucrado dentro del desarrollo del proyecto de *software*.

d) **Coordinación de departamentos, planes y políticas.** Es importante la formación de un grupo, ya que con esto se pueden coordinar departamentos de desarrollo o entre el departamento al cual se desarrolla el proyecto de *software* y el grupo de desarrollo, tomando en cuenta los planes y políticas de cada uno.

e) **Transmisión y compartimiento de información.** Estos grupos son utilizados también para la transmisión y compartimiento de la información.

f) **Consolidación de la autoridad.** Esta ventaja se da cuando no existe una forma clara de delegación de autoridad dentro del proyecto de *software*, aquí es donde se han de solucionar dichas discrepancias.

g) **Motivación mediante la participación.** Estos grupos permiten una amplia participación de todo el personal involucrado o representado, para el momento de la toma de decisiones.

Desventajas de los grupos. Aunque los grupos tienen muchas ventajas, no dejan de presentar algunas desventajas, las cuales son:

a) Alto costo en tiempo y dinero

b) Negociación del mínimo común denominador

c) Indecisión

d) Tendencia a la auto-destructividad

e) Dispersión de la responsabilidad

4.4 Comunicación

La comunicación se debe aplicar en todas las fases de la administración de un proyecto de *software*, pero en forma particular e importante en la función de dirección.

- Tipos de comunicación. Dentro de la comunicación de un proyecto de *software* existen varios tipos de comunicación, los cuales se describen a continuación:

  a) **Comunicación descendente.** Es la comunicación que fluye de personas de alto rango a personas de niveles inferiores dentro de la jerarquía del proyecto de *software*. Este tipo de comunicación se puede dar por medio de discursos, reuniones, por teléfono, etc.

  b) **Comunicación ascendente.** Es la comunicación que se circula de los niveles inferiores de la jerarquía organizacional del proyecto de *software* hacia los niveles superiores.

  c) **Comunicación cruzada.** Este tipo de comunicación influye en el flujo de información horizontal (entre personal del proyecto de un mismo rango) y en forma diagonal (entre personal de diferentes rangos pero sin relación directa dentro del proyecto de *software*).

  d) **Comunicación escrita.** Es una comunicación utilizada tanto para mensajes formales como informales, y tiene la ventaja de registro y protección legal, aunque tienen la desventaja de generación de mucho papeleo, y puede tener expresiones que no reflejen lo que en realidad se desea o necesita dar a entender.

e) **Comunicación oral.** Esta comunicación puede ocurrir de frente a frente o por medio de discursos o conferencias, ya sea formal o informalmente.

f) **Comunicación no verbal.** Esta comunicación se da por medio de expresiones faciales o movimientos corporales.

- Desventajas de la comunicación. Normalmente las fallas en la comunicación pueden representar síntomas de problemas más profundos, los cuales se detallan a continuación:

a) Falta de planeación

b) Supuestos confusos

c) Distorsión semántica

d) Mensajes deficientemente expresados

e) Escucha deficiente y evaluación prematura

f) Deficiente retención

g) Sobrecarga de información

5. CONTROL DE UN PROYECTO DE *SOFTWARE*

5.1 Sistemas de control

Como se explicó anteriormente, el control es la función administrativa en la cual el administrador del proyecto de *software* a de medir y corregir el desempeño, a fin de garantizar que se ha de cumplir con los objetivos del proyecto y que dichos planes elaborados han de ser alcanzados.

Las técnicas y sistemas de control que ha de utilizar un administrador de proyectos de *software*, son básicamente los mismos, siendo estos asuntos de dinero, procedimientos de oficina o desarrollo, moral y motivación a sus empleados, etc. en cualquier momento y cualquiera que sea el área a aplicar el control, siempre se ven implicados tres pasos, los cuales son:

a) **Establecimiento de normas.** Las normas de un control eficaz y eficiente están dadas sencillamente por criterios de desempeño, los cuales están establecidos por la percepción de señales con el fin de establecer en que forma están marchando los planes del proyecto.

b) **Medición del desempeño.** Este paso debe ser tomado por el administrador del proyecto de *software* como un punto de previsión, con el fin de poder detectar desviaciones de los planes originales antes que estos ocurran y tomar las acciones apropiadas para los mismos.

Si las normas establecidas, han sido ideadas en forma adecuada y se dispone de medios para determinar con toda precisión lo que cada una de las personas del grupo de desarrollo hace, la evaluación a realizar es facilitada en gran medida por el administrador del proyecto de *software*.

También se deben considerar aquellas tareas o trabajos los cuales no pueden ser medidos con gran exactitud, y algunas tareas que no son tan técnicas y por lo tanto también se dificulta la evaluación de los mismos.

c) **Corrección de las desviaciones.** Las normas que han sido establecidas por el administrador del proyecto de *software* deben reflejar en forma precisa los diversos puestos de una estructura organizacional definida al inicio de éste, y si el desempeño a medir tiene gran correspondencia con ellas, entonces se puede decir que la corrección de las desviaciones que se presenten en dichas normas han de ser fáciles de corregir.

Este punto se dificulta un poco cuando el control y corrección se aplica a labores individuales o grupales dentro del proyecto de *software*, en la cual el administrador debe saber exactamente donde aplicar las medidas correctivas.

Cuando se realiza una corrección de desviaciones que se han presentado, es cuando el control se vuelve dentro del proyecto una función administrativa.

Un administrador de proyecto de *software* puede realizar correcciones por medio del rediseño de sus planes o la modificación de sus metas.

También puede ser corregido ejercitando su función de organización, ya sea resignando o aclarando los deberes de cada integrante del proyecto de *software*. También puede corregir estas desviaciones utilizando la ayuda de personal adicional, mediante la selección y capacitación de sus subordinados o recurriendo a la reintegración del personal. Por último, un administrador de proyectos de *software* puede corregir estas desviaciones por medio de una mejor dirección.

En este punto, es necesario definir los puntos críticos de control, el cual indica que 'para ser eficaz, éste implica particularmente la atención a los factores críticos para la evaluación del desempeño con base en los planes'.

Existen diferentes tipos de normas para los puntos críticos, estos son dictados por los objetivos, metas, programas, actividades, políticas, procedimientos, presupuestos, etc., se convierten en normas con base en las cuales es posible medir el desempeño real o esperado. Aunque normalmente se suele utilizar los siguientes tipos:

a) **Normas físicas.** Estas normas son medidas no monetarias dentro del proyecto de *software*, en la cual se utilizan materiales, se emplea la fuerza de trabajo, se prestan servicios o se producen bienes. También reflejan calidad.

b) **Normas de costo.** Estas normas, son medidas monetarias comunes, como las físicas. Son atribuidas a valores monetarios a aspectos operacionales del proyecto de *software*.

c) **Normas de capital.** Estas normas son muchas en la vida real, pero normalmente la aplicable a un proyecto de *software*, se puede decir que son las de medidas monetarias aplicadas a objetos físicos. Este aspecto tiene más que ver con el dinero invertido al proyecto de *software*, y por lo tanto se relaciona bastante con los balances generales. Tanto así como con la rentabilidad del proyecto y el rendimiento del mismo.

d) **Normas de ingresos.** Las normas de ingreso son utilizadas como resultado de la aplicación de valores monetarios a la venta. Este punto puede ser aplicado dentro del proyecto de *software* únicamente, si se está pensando en realizar un producto final, que ha de ser mercadeado por la empresa que desarrolla dicho *software*.

e) **Normas de programas.** Son utilizadas para el cumplimiento de normas establecidas, para el mejor desempeño y control del proyecto de *software*.

f) **Normas intangibles.** Son aquellas que no pueden formularse en forma física o monetaria y que son difíciles de establecer. Existen muchas normas de este tipo, cuando llegada una fecha puede que no exista el inicio de dichas tareas, por lo tanto el costo y productividad de los mismo aún son un misterio o solamente son estimaciones que se establecen en el proyecto.

5.2 Técnicas de control

Como se ha explicado con anterioridad, el control, en su naturaleza y su propósito básico es invariable, normalmente se han utilizado varias técnicas para tecnificar dicha práctica, las cuales deben estar siempre presentes para todo buen administrador de proyectos de *software*. Dichas técnicas son las siguientes:

- Presupuesto. Este es una de las técnicas más amplias de control para un proyecto, aunque hay que tomar en cuenta que no sólo depende del presupuesto en sí, también están implícitas otras técnicas y métodos esenciales y a su vez no son presupuestales.

Como se definió anteriormente el 'presupuesto' es la formulación en términos numéricos de planes para un periodo futuro determinado. Esto es utilizado para crear planes en términos numéricos y a su vez dividirlos en las partes de la organización del proyecto de *software*, lo cual permite delegar autoridad con mayor facilidad y sin la pérdida del control. En fin, el hecho de convertir los planes a números, es una forma de obligar a todo administrador de proyectos de *software* o jefe de proyectos a tener especial cuidado con el dinero que se gasta, quién lo gasta y cómo o por qué lo gasta.

Es por eso, que a continuación se mencionan cuatro tipos de presupuestos que son utilizados en el ámbito de la administración de proyectos de *software*, los cuales son:

a) **Presupuesto de ingresos y gastos.** Este presupuesto se basa en la proyección de ventas del *software* en un periodo determinado, o en algunos casos, los ingresos que se pretende que genere el proyecto el cual se está implementando, contra los gastos de operación que genera dicho desarrollo.

b) **Presupuesto de tiempo, espacio, materiales y producto.** Como se dijo anteriormente, los presupuestos son elaborados gracias a la conversión de los planes a números, los cuales normalmente son expresados en dinero, pero muchas veces esta necesidad no es tan grande, y es mejor tener que convertir dichos planes en cantidades (no necesariamente dinero).

   Este tipo de presupuesto se puede crear por ejemplo, con la elaboración de trabajo productivo por horas, semanas, meses, por persona, por grupo de trabajo, etc.

c) **Presupuesto de gasto de capital.** Este tipo de presupuesto se centra únicamente en el gasto de capital que se presenta por la manutención del proyecto de *software*. Ya que normalmente los gastos de inversión, que pueden ser tomados aquí, son difíciles de recuperar, puesto que realizan a largo plazo.

d) **Presupuesto de efectivo.** Este tipo de proyecto, no es más que la proyección de los ingresos y los egresos que ha de sufrir nuestro proyecto, dentro de un tiempo determinado. También es utilizado para hacer proyecciones de la utilización del efectivo en obligaciones de la empresa, etc.

- Análisis de la red tiempo-eventos. Otra técnica de control que existe, muy diferente a la técnica de presupuesto, es la llamada técnica de evaluación y

revisión de programas o análisis de la red tiempo-eventos. Para esta técnica de control existen dos métodos, los cuales se numeran a continuación:

a) **Gráficas de Gantt.** La gráfica de Gantt es utilizada para visualizar de forma general las tareas que debe seguir un grupo de desarrollo, o mejor dicho, las tareas que han sido identificadas por el jefe de proyectos y que han de ser cumplidas por el jefe de desarrollo para alcanzar la culminación de dicho proyecto.

Esta técnica indica que la gráfica debe ser vista como una serie de planes de apoyo interrelacionados, para que los individuos puedan comprender y seguir para la culminación eficiente de sus tareas.

Dicha técnica también es de gran importancia dentro de un proyecto de *software*, porque refleja de forma sencilla y rápida el avance o atraso que sufre un proyecto, los costos o ahorros que esta sufriendo el mismo, de forma un tanto indirecta, puede indicar la efectividad de las personas involucradas en el proyecto; evaluando de forma general, el cumplimiento y avances del mismo.

b) **Técnicas de evaluación y revisión de programas (PERT).** La técnica de PERT, es una técnica basada en la ruta crítica que ha de seguir un proyecto de *software* para su culminación. Esta técnica es una de las más utilizadas en los proyectos, además es un sistema de análisis de redes tiempo-evento, en el que se identifican diversas tareas a cumplir dentro de un proyecto de *software*, y para cada uno de ellos se establece un periodo planeado.

Estos eventos se establecen y se organizan en una red que revela la relación entre ellos. El sistema de PERT, también se basa en la adjudicación de tiempos para cada una de las tareas que han de ser culminadas, y normalmente se elaboran con tres tipos de estimación de tiempo.

El 'optimista', la cual es establecida como un tiempo extraordinario para la culminación de alguna tarea específica. El 'probable', que es el tiempo que ha de necesitarse en condiciones normales para el cumplimiento de las tareas. Y el 'pesimista', que son las estimaciones de tiempo adjudicando mala suerte y complicación para el cumplimiento de las tareas. El último paso de la técnica PERT, es la identificación de la ruta crítica, la cual es la secuencia de eventos de mayor duración, y que no exista un tiempo de inactividad entre una y otra tarea. La técnica de PERT, tiene cinco ventajas importantes, las cuales se describen a continuación.

Primero, obliga a los administradores de proyecto de *software* a planear; segundo, obliga a los administradores de proyectos de *software* a planear bajo la línea de organización de la empresa; tercero, dirige la atención a eventos críticos, que de alguna forma han de necesitar corrección; cuarto, hace posible un control de previsión; y quinto, hace que los análisis y correcciones se centren al nivel organizacional indicado, en un momento oportuno.

6. CALIDAD DEL PROYECTO DE *SOFTWARE*

6.1 Conceptos de calidad

La calidad es definida como una característica o atributo de algo, que puede ser comparado con estándares conocidos.

No obstante, dentro de un proyecto de *software*, tanto la aplicación de la calidad como el control de la misma, son de gran importancia, por lo tanto es necesario definir cuales son los tipos de calidad que existen, y que serán descritos a continuación:

a) **La calidad de diseño.** Que está definida por las características que aplica un desarrollador de *software* para un artículo específico. Definido también, por el grado de material, tolerancia y especificación de rendimiento, donde todos contribuyen a la calidad del diseño.

b) **La calidad de concordancia.** Este tipo de calidad se determina por el grado en el cumplimiento de las especificaciones de diseño durante la realización. Esto conlleva la definición, que mientras más alto sea el grado de concordancia, mayor será la calidad alcanzada en la misma.

- Control de la calidad. El control de la calidad dentro de un proyecto de *software*, es una de las tareas más complejas y de vital importancia para un administrador de proyectos de *software*, el cual consta de una serie de inspecciones, revisiones y pruebas utilizados a lo largo del ciclo de desarrollo de la aplicación, para asegurar que cada producto entregado cumple con los requisitos que le han sido asignados.

Una de las técnicas más utilizadas para desarrollar e implementar el 'control de calidad' es la combinación de la medición y reglamentación de los procesos que afinar el producto de trabajo que ha sido creado, y con ello evitar las fallas que se tienen dentro de las especificaciones de desarrollo.

Las actividades de control que se realiza dentro de un proyecto de *software* pueden ser manuales, automáticas o la combinación de ambas, lo cual implica la utilización de alguna herramienta automática y la inspección humana.

Un punto clave para realizar e implementar un buen control, es la definición exacta y correcta de todas las especificaciones mensurables de un producto, en las que se pueda comparar los resultados de cada proceso.

- Garantía de calidad. La garantía de calidad, también llamada 'aseguramiento de la calidad', consiste en la auditoria y las funciones de información de una gestión de calidad.

Ésta debe de proporcionar la gestión de informar y proporcionar los datos necesarios sobre la calidad de un *software*, por lo que se adquiere una visión más profunda y segura de la calidad del *software* está cumpliendo a plenitud con los objetivos planeados para el mismo.

Los datos que son proporcionados por la garantía de calidad, son únicamente la identificación de los problemas, y es responsabilidad del administrador de un proyecto de *software* afrontar y aplicar recursos necesarios para resolver dichos aspectos de calidad.

- Coste de calidad. El coste de calidad, está definido por todos aquellos costes que son acarreados para encontrar la calidad de un *software*, y todas las actividades relacionadas con la obtención de dicha calidad.

Normalmente se realizan estudios para saber cual será el coste de la aplicación de calidad en un momento determinado identificar de la misma manera las oportunidades de reducir dichos costos y proporcionar una base normalizada para la comparación de dichas características.

Dicha base de normalización o coste de calidad tiene un precio, el cual es necesario para que el administrador del proyecto de *software* pueda evaluar el lugar en donde hay oportunidades de mejorar nuestros procesos.

Por este motivo, los costes de calidad se pueden dividir en costes:

a) **Costes de prevención.** Los cuales incluyen:

Planificación de la calidad
Revisiones técnicas formales
Equipo de pruebas
Formación

b) **Coste de evaluación.** Son utilizadas para obtener una visión más profunda de las condiciones en que se encuentra el *software* desarrollado en un momento en particular. Algunos ejemplos de costes de evaluación son:

Inspección en el proceso y entre procesos
Calibrado y mantenimiento del equipo
Pruebas

c) **Costes de fallos.** Los costes de fallo son errores que surgen en el envío del *software* a un cliente. Existen dos costes de fallo:

Los costes de fallos internos. Se produce cuando se encuentra algún error en el *software* antes de ser enviado al cliente.

Los costes de fallos externos. Se asocian a los defectos encontrados una vez ya fue enviado el *software* al cliente.

6.2 Garantía de calidad para el *software*

Todo buen desarrollador de *software*, sabe que mientras más calidad exista en una aplicación, ésta resulta más benéfica para el usuario.

Es por esto, que se define la calidad del *software* como:

> *La concordancia entre los requisitos funcionales y el rendimiento explícito establecido, con la aplicación de estándares de desarrollo bien documentados, y con las características que se espera de todo software desarrollado profesionalmente.*

Para realizar una buena aplicación de calidad, nos vemos en la necesidad de definir tres puntos importantes de la calidad del *software*.

a) Los requisitos del *software* son la base de la medida de la calidad. La falta de concordancia que puede existir con los requisitos puede ser una falta de calidad.

b) Los estándares especificados para el desarrollo de *software* es un conjunto de criterios de desarrollo que guía la forma en que se aplica el buen desarrollo de un *software*.

c) Por último, se debe considerar el alcanzar tanto los requisitos explícitos de los estándares que definen la calidad, poder cumplir con los requisitos implícitos.

- Actividades de la garantía de calidad de *software*. La garantía de calidad se divide en dos grandes tareas, la primera, que debe ser realizada por el administrador del proyecto de *software*, la cual realiza el trabajo técnico de garantía de calidad, y el trabajo del grupo de garantía de calidad del *software*, los cuales están encargados de la supervisión, mantenimiento, análisis e informes de la calidad.

El administrador del proyecto de *software* ha de afrontar la calidad, aplicando métodos técnicos y sólidos, para la realización de revisiones técnicas formales, y llevando a cabo pruebas de *software* planificadas.

El grupo de trabajo de garantía de calidad de *software*, trata de ayudar al equipo de desarrollo, para obtener un producto final de alta calidad. Para esta tarea, se pueden sugerir varias actividades que ayudan a obtener la calidad dentro de un proyecto de *software*, dichas actividades son:

a) Establecimiento de un plan de garantía de calidad. Este conlleva la elaboración de un plan de desarrollo durante la planificación del proyecto y es revisado por todas las partes involucradas e interesadas. Las actividades que son realizadas por el grupo de garantía de calidad son guiadas por dicho plan.

   Dentro de este plan, normalmente se deben definir las siguientes tareas: evaluaciones a realizar, auditorias y revisiones a realizar, estándares que se pueden aplicar al proyecto, procedimientos para información y seguimiento de errores, documentos producidos por el grupo de garantía de calidad.

b) Participación en el desarrollo de la descripción del proceso de *software*. El proceso que es escogido por el grupo de desarrollo de *software*, es evaluado y considerado si cumple con las reglas y política de la empresa, reglas y estándares de impuestos externamente, y reglas que hayan sido establecidas de forma interna.

c) Revisión de las actividades de desarrollo de *software* para verificar su ajuste al proceso de desarrollo. El grupo de garantía de calidad identifica, documenta y sigue la pista de las desviaciones desde el proceso y verifica que se hayan hecho las correcciones indicadas.

d) Auditoria de los procesos de *software* designados para verificar el ajuste con los definidos como parte del desarrollo del *software*.

El grupo de garantía de calidad revisa los procesos seleccionados dentro del *software* desarrollado, identifica, documenta y sigue la pista de las desviaciones que fueron identificados en dichos procesos, verifica posteriormente que se hayan realizado las correcciones indicadas e informa en forma periódica los resultados que se han obtenido con dichas garantías de calidad, al administrador o jefe de proyectos.

e) Asegurar que las desviaciones del trabajo y el desarrollo del *software* se documenten y se manejen de acuerdo con un procedimiento establecido.

   Los defectos que son encontrados en el plan del proyecto, descripción de procesos y los estándares no aplicados en forma correcta, deben ser documentados y manejados en una forma y procedimiento establecidos por el administrador del proyecto de *software*.

f) Registrar lo que no se ajuste a los requisitos e informar a sus superiores. Los requisitos o procedimientos que no se ajustan a los estándares de garantía de calidad, deben ser reportados al administrador del proyecto de *software* y darle seguimiento hasta que éstos sean cumplidos a cabalidad.

6.3 Revisiones de software

Las revisiones de *software*, pueden ser consideradas como un filtro por parte del administrador de un proyecto de *software*. Dicha revisiones deben ser aplicadas en varias etapas del proyecto, y esto se hace para identificar defectos que pueden ser eliminados.

Las revisiones de *software* son utilizadas para purificar las actividades de desarrollo de *software* que provienen del resultado del análisis, diseño y la codificación.

El objetivo primario de las revisiones técnicas del *software,* es encontrar errores durante el proceso de forma, que se conviertan en defectos después de la entrega del *software* al cliente. El beneficio de estas revisiones es el descubrimiento de errores al principio para que no se propaguen al paso siguiente del desarrollo del *software*.

6.4 Revisiones técnicas formales

Una revisión técnica formal es una actividad de la garantía de calidad del *software* que es realizada por el administrador del proyecto, y que busca cumplir con los siguientes objetivos:

a) Descubrir errores en la función, lógica o implementación de cualquier representación del *software*.

b) Verificar que el *software* que esta siendo revisado alcanza sus requisitos.

c) Garantizar que el *software* ha sido representado de acuerdo con ciertos estándares que fueron definidos.

d) Conseguir el desarrollo de un *software* en forma uniforme.

e) Hacer que todo proyecto de *software* sea más manejable.

Las revisiones técnicas formales son realmente una clase de revisión que incluye recorridos, inspecciones y torneos de revisiones, y que se lleva a cabo mediante una reunión y sólo podrá tener éxito si es bien planificada, controlada y atendida.

Una revisión de técnicas formales consta de varios pasos, los cuales se describen a continuación:

a) **La reunión de revisión.** Cualquier reunión que se elija para implementar una revisión técnica formal, debe contener por lo menos las siguientes tres restricciones:

Debe convocarse para la revisión entre tres y cinco personas.

Se debe preparar por adelantado al personal convocado.

La duración de la reunión debe ser menor a dos horas.

El proceso de revisión técnica forma se inicia cuando la persona encargada del desarrollo del *software* lo finaliza y solicita al administrador del proyecto de *software,* la revisión técnica formal, entonces dicho administrador contacta al grupo de persona revisores, realiza un documento relacionado con el *software* a evaluar y lo envía con anticipación, para que éstos puedan familiarizarse con el *software* que deberán de evaluar. Cada uno de los revisores no tendrá más de dos horas para realizar la evaluación.

Al final de la revisión formal técnica dichos revisores deberán: Uno, aceptar el producto sin posteriores modificaciones; dos, rechazan el producto por los serios errores encontrados, y después de su modificación se ha de someter de nuevo a otra revisión; tres, aceptar el

producto provisionalmente, que implica la corrección de errores menores, sin la necesidad de otra revisión.

Registro e informe de la revisión. El registro e informe de revisión es un documento realizado por alguno de las personas que conforman el grupo de revisión, en donde se apuntan las correcciones que se han de realizar al *software* desarrollado, este documento por lo menos debe contar con las siguientes respuestas:

¿Fue revisado?
¿Quién lo revisa?
¿Qué se descubrió y cuales fueron las conclusiones?

Este informe es un sumario en página simple y que tiene dos propósitos: Uno, identificar áreas problemáticas dentro del *software*; dos, servir como lista de comprobación de puntos de acción que guíe al desarrollador para realizar sus correcciones.

b) **Directrices para la revisión.** Nunca se deben establecer directrices de antemano para la elaboración de una revisión técnica formal, pero se debe contener y contemplar por lo menos los siguientes puntos:

Revisar el producto, no al productor.

Fijar una agenda y mantenerla.

Limitar el debate y las impugnaciones.

Enunciar áreas de problemas, pero no intentar resolver cualquier problema que no se ponga de manifiesto.

Tomar notas escritas.

Limitar el número de participantes e insistir en la preparación anticipada.

Desarrollar una lista de comprobación para cada producto que haya de ser revisado.

Disponer recursos y una agenda para las revisiones técnicas formales.

Llevar a cabo un buen entrenamiento de todos los revisores.

Repasar las revisiones anteriores.

6.5 Fiabilidad del software

La fiabilidad del *software* se define como ‘la probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y durante un tiempo específico’.

La fiabilidad del *software*, a diferencia de otros controles de calidad de un proyecto de *software*, puede ser medido con relación a datos históricos o de desarrollo.

Medidas de fiabilidad y de disponibilidad. Considerando los fallos de nuestro *software*, se puede definir la medida de fiabilidad como ‘el tiempo medio entre fallos’, como:

Tiempo medio entre fallos = tiempo medio de fallo +
tiempo medio de reparación

Además de una medida de fiabilidad, se debe definir una medida de disponibilidad del *software*. Esta disponibilidad, es la probabilidad de que un programa funcione de acuerdo a los requisitos solicitados en un momento dado, el cual podemos definir de la siguiente manera:

Disponibilidad = tiempo medio de fallo /
(tiempo medio de fallo + tiempo medio de reparación) * 100%

Análisis de riesgo y seguridad del *software*. Se define como la actividad de garantía de calidad de *software* que se centra en la identificación de los riesgos potenciales que pueden producir un impacto negativo en el *software* y hace que falle el sistema completo.

Si se pueden identificar los riesgos en forma pronta, es mucho más fácil para el administrador del proyecto de *software* especificar características para eliminar o controlar riesgos potenciales.

Como parte del proceso de seguridad de *software*, el administrador del proyecto, debe identificar los riesgos y clasificarlos por su importancia y su grado de riesgo.

Existen dos técnicas de análisis para asignar la gravedad y probabilidad de ocurrencia, los cuales son:

a) **Análisis del árbol de fallos.** Este método construye un modelo gráfico de las combinaciones secuenciales y concurrentes de los sucesos que pueden conducir a un estado del sistema o suceso peligroso. Mediante un árbol detallado de fallos desarrollados, es posible observar las consecuencias de una secuencia de fallos interrelacionados que ocurren en diferentes componentes.

b) **Lógica de tiempo real.** Este modelo se construye mediante la especificación de los sucesos y las acciones correspondientes. Este modelo suceso-acción se puede analizar mediante operaciones lógicas, para probar las valoraciones de seguridad de los componentes del sistema y su temporalización.

En este punto hay que establecer una gran diferencia, la fiabilidad y la seguridad del *software* están bastante relacionadas. La fiabilidad es un análisis estadístico para determinar la probabilidad de que ocurra un fallo en el *software*, y la seguridad del *software* es examinar los modos según los cuales los fallos producen condiciones que pueden llevar a accidentes.

6.6 Plan de garantía de calidad de software

El plan de garantía de calidad de *software* es un mapa para institucionalizar dicha garantía.

El plan desarrollado por el administrador y el grupo encargado del control de calidad, sirve como plantilla para actividades de garantía de calidad instituidas para cada proyecto específico.

Este plan de garantía de calidad, consta de varios puntos o incisos, que se describen a continuación, y que enmarcan un plan de garantía de calidad de *software* estándar, aunque no único, dichos puntos son:

a) Propósito del plan

b) Referencias

c) Gestión

d) Documentación

e) Estándares, prácticas y convenciones

f) Revisiones y auditorias

g) Prueba

h) Información sobre problemas y acción correctora

i) Herramientas, técnicas y metodología

j) Control de códigos

k) Control de medios

l) Control de distribución

m) Recopilación de registros, mantenimiento y retención

n) Formación

o) Gestión de riesgos

Los primeros dos incisos hacen referencia al alcance del documento e indican que esas actividades del desarrollo del *software* son abarcadas por la garantía de calidad.

La gestión del plan, describe la situación de la garantía de calidad del *software* dentro de la estructura organizativa.

La documentación, describe cada uno de los productos de trabajo producidos en el desarrollo del *software*.

Los estándares, prácticas y convenciones, lista todos aquellos estándares y/o prácticas que se aplican durante el proceso de desarrollo de *software*.

La sección de revisiones y auditorias del plan, identifica las revisiones y auditorias que se han de llevar a cabo por parte del equipo de desarrollo de *software*.

Las pruebas se utilizan para referenciar el plan y los procedimientos de prueba del *software*.

La sección de información sobre problemas y acción correctora, define los procedimientos para informar, hacer seguimiento y resolver errores y defectos del *software*.

El resto del plan, contiene garantía de calidad, identifica herramientas y métodos de configuración de *software*, enfoque de gestión de contratos, métodos para reuniones, etc.

CONCLUSIONES

1. La utilización de la metodología de la administración tradicional para el desarrollo de productos y proyectos de *software* es altamente efectivo dentro del ámbito comercial, porque ambos tipos de administración son claramente similares al momento de ser aplicados.

2. El crear un ambiente similar de trabajo, al desarrollar productos de *software* ayuda de gran manera al personal a cargo de la administración de dicho producto, por poder utilizar las técnicas administrativas al personal y proyecto en general. Teniendo en cuenta siempre las variantes y modificaciones que esta aplicación puede estar sufriendo durante el periodo de desarrollo del mismo.

RECOMENDACIONES

1. Todo proyecto de *software*, sin importar el tamaño del mismo, se considera de vital importancia para la empresa que lo desarrolla, por lo mismo se debe tener en cuanta la aplicación de todas aquellas herramientas que hagan posible el desarrollo en forma correcta de dicho proyecto.

2. Es esencial que los puntos citados en el presente trabajo de graduación sean tomados en cuenta, y más importante aún, aplicados desde el inicio del desarrollo del proyecto de *software*.

BIBLIOGRAFIA

1. Kendall, Kenneth E. y Julie E. Kendall. **Análisis y diseño de sistemas.** 3a. ed. México: Ed. Prentice Hall Hispanoamérica, S.A., 1997. 914 pp.

2. Koontz, Harold. **Administración, una perspectiva global.** 11a. ed. México: Ed. McGraw Hill, 1998. 796 pp.

3. Pressman, Roger S. **Ingeniería de *software*, un enfoque práctico.** 4a. ed. México: Ed. McGraw Hill, 1998. 581 pp.