UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

Centro Universitario UAEM Texcoco

“Algoritmo para la Asignación Automática de Tareas de los Procesos en las Fábricas de Software”

Tesina

Que para obtener el Título de:

Ingeniero en computación

Presenta:

Refugio Marisol Méndez Vázquez

Director:

M. en C. A. José Sergio Ruiz Castilla

Texcoco, México Noviembre del 2013

DEDICATORIAS

A Dios, por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mis padres, como un testimonio de cariño y eterno agradecimiento por mi existencia, valores morales y formación profesional. Porque sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y porque nunca podré pagar todos sus desvelos, ni aún con las riquezas más grandes del mundo. Por lo que soy y por todo el tiempo que les robé pensando en mí...

A mis queridos hermanos, José Enrique y Wendy, por haber sido los mejores amigos que jamás podré tener en mi vida; y como recordatorio del profundo cariño que siempre les tendré.

A mi esposo Maurilio quien me apoyo y alentó para continuar, cuando parecía que me iba a rendir. El merito de éste trabajo también te pertenece, y nunca terminaré de agradecerte lo bueno que has sido conmigo. Te amo

A mí amada hija Alison, por iluminar mi vida con la suya.
Te amo.

A todas las personas que con su ayuda apoyo y comprensión me alentaron a lograr esta hermosa realidad.

AGRADECIMIENTOS

A la Universidad Autónoma del Estado de México a través del Centro Universitario UAEM Texcoco, por las facilidades brindadas para la realización de esta meta, y por abrirme las puertas de lo que ahora es mi segundo hogar

A mi director de tesina, M. en C. A. José Sergio Ruiz Castilla, por su paciencia y apoyo incondicional mostrado, además por su preocupación infinita, porque día tras día seamos cada vez mejores y por sus conocimientos aportados para la culminación de esta tesina. Gracias.

A mis sinodales: M. en C. A. Rafael Valencia Valencia y M. en C.C. José Jair Vázquez Palma, por sus acertadas recomendaciones, para el mejoramiento de esta tesina y gracias por todo su tiempo invertido en la revisión de la misma.

Contenido

Índice de tablas

Índice de Imágenes

Índice de graficas

Introducción

En las fábricas de software se aplican metodologías de desarrollo de software, las cuales consideran un equipo de trabajo para el desarrollo de los proyectos. En cada proyecto está un líder que tiene a su cargo el equipo de trabajo. Durante la planeación se deben determinar los módulos y sus tareas que se completarán a lo largo del proyecto. Es preciso establecer calendarios de trabajo donde incluyen tareas y los roles de los desarrolladores que deben ejecutar las tareas.

La asignación de tareas es necesaria en los planes detallados de cada módulo. Esta tarea puede ser de forma manual o bien automatizada. En los proyectos grandes pueden ser cientos de tareas que se deben de asignar a los desarrolladores.

Por lo anterior en este trabajo se investigó y escribe en el capítulo I acerca de los equipos de trabajo, con el fin de conocer cómo se forma, sus roles, comunicación y formas de trabajo y de la asignación de tareas que pueden ser automatizadas una vez que están definidas, estimadas y perfiladas. El perfil de una tarea indica qué tipo de tarea es y qué habilidades se requieren para su ejecución. Por otro lado cada desarrollador tiene uno o más perfiles, de los cuales se desprenden los roles. En el capítulo II se habla del uso de arreglos, los tipos de arreglos que existen, cuáles son sus ventajas, el manejo del método Split, la manipulación de los archivos de texto, la escritura de un archivo .txt, actualización de información, y la lectura de archivos.

También se planteó una solución a través de un algoritmo que permite hacer la asignación automatizada de tareas definiendo un perfil para cada tarea así como un perfil para cada desarrollador. Una vez se logró el algoritmo se probó con datos de prueba de un proyecto y un equipo de trabajo, se documentarán los resultados y redactarán las conclusiones.

Antecedentes

Sistema de asignación de tareas a trabajadores polivalentes

El trabajo que presenta Mora y Moreno de fin de carrera, consiste en la creación y posterior desarrollo de un sistema de asignación de tareas a trabajadores polivalentes. Esto es, una herramienta que nos organiza la distribución laboral a lo largo de un día, en jornadas y según el número de trabajadores, de manera que cada uno de estos empleados pueda realizar distintas tareas, según se requiera.

Para poder lograr esto, se ha utilizado Microsoft Access 2003. Este programa se basa en la programación mediante lenguaje VBA (Visual Basic para Access) y consta de herramientas que generan automáticamente distintas utilidades (formularios, subformularios, botones, etc).

Durante el proceso de dicho trabajo se realizaron cinco diseños y el diseño final. El primer diseño permitió la creación de diagramas UML (Lenguaje Unificado de Modelado) donde se realizaron las tablas tentativas a utilizar, todo este primer diseño se enfocó en una tienda de ropa, se crearon 3 formularios (insertar empleado, búsqueda y generar formularios), todo esto con la ayuda de Access 2003 donde se generó la relación de tablas y con la ayuda de VB donde se generan los eventos de la inserción de empleados, la búsqueda y la petición de informes a la BD (Base de Datos) creada.

En el segundo diseño se pretendió mostrar una tabla por pantalla, donde se pudiera observar los diferentes horarios a lo largo del día, separados por horas y clasificados según los nombres de los trabajadores, donde se pueda ver claramente qué tarea le corresponde en cada momento a cada uno, además de mostrar una tabla con el nivel de cada trabajador para dicha tarea. Se aplicó un algoritmo para que estudie si el nivel en cada tarea asignada está por debajo del mínimo: Usando el *Formato condicional,* una de las opciones que ofrece el Access, en la que se escribió la siguiente condición:

([NT1].[Valor]<>4 Y [ctl8h_9h].[Valor]=1) O ([NT2].[Valor]<>4 Y [ctl8h_9h].[Valor]=2)....

Donde NT1 denota el nivel de tarea numero 1 y ctl indica el horario de la tarea

A partir de esto se genera un contador en VB que se encargue de llevar el conteo de número de trabajadores que había para cada horario en cada tarea, para saber así si llegaba al número mínimo requerido. Para ello, lo primero a implementar debía ser este número mínimo deseado de trabajadores para poder compararlo con el número real que hay en cada hora. Creando así una matriz donde se comparan el número de empleados asignados por cada tarea con los que realmente se necesitan. Concluyendo con este diseño se menciona que los valores eran almacenados en variables lo cual hacia que el código fuera enorme y no permitía un estudio eficaz, con lo cual se rehízo todo el diseño pero almacenando en tablas de datos.

El tercer diseño se crea un formulario llamado *general,* el cual consta de 2 subformularios llamados *planning*[1] *y niveles*, en el primero se muestra una tabla con los empleados, horarios y las tareas que han de realizar, así como el nivel mínimo requerido en éstas. En el segundo, se muestra el nivel desarrollado por cada empleado en cada tarea. Se crea un método llamado contadores el cual lleva el conteo de trabajadores por horarios, estos son almacenados en una tabla llamada contadores con el fin de evitar el uso innecesario de variables excesivas. Este diseño permite ver un informe un poco más detallado aunque sigue existiendo un número masivo de variables almacenadas en tablas.

El cuarto diseño pretende solucionar el problema del uso masivo de variables, se utilizan contadores que almacenen el número de empleados que hay para cada tarea, en cada horario establecido según el día. Primera parte se rellenan los contadores con el número de empleados ya registrados en el subformulario general, después seleccionar el día el cual puede ser seleccionado por el usuario mediante un MsgBox (Ventana con mensaje), se recorre la tabla general por medio de un triple bucle, que recorre por tareas *(tx),* franjas *(fx)* y empleados *(trx).*

[1] Planning: Planificacion.

Una vez hallada la posición deseada (haciendo uso de la sentencia *if*), se comprueba que realmente es la tarea que toca (*If (rst1("Tarea") = tx) Then*) se abre la tabla *Contadores* para incrementar el contador que toca, según día especificado, franja y tarea.

En la segunda parte se comprueban los requisitos, los contadores rellenados nos permite observar si el número de empleados es el deseado según el usuario, esto se observa por cada franja, día y tarea, mostrando si el contador correspondiente es menor a la asignada en la columna de requisitos o en su defecto es menor al mínimo impuesto, lo cual se reflejará en pantalla mediante un MsgBox. Se agregaron nuevos elementos en las tablas donde se especificó si al trabajador se le asignó una tarea principal, es decir, si se le contrató para una tarea específica u extraordinaria. Para esto se realizó una tabla llamada *principal* donde se almacenó la relación de todos los empleados y las todas las tareas existentes más un elemento que indicará si la tarea es principal o no.

Este código lo que hace es escoger la tarea seleccionada y almacena una variable, un nuevo bucle empieza a recorrer toda la tabla *principal* buscando la tarea anterior, cuando la encuentra mira si el campo *principal* al ser objeto si/no está activado o desactivado, cuando sube su valor le asigna el mismo al campo principal pero de la tabla *principal (Fig 1).*

Principal : Tabla

	Nombre del campo	Tipo de datos
	Codigo Trabajador	Número
	Tarea	Número
	Principal	Sí/No

Principal : Tabla

	Codigo Trabajac	Tarea	Principal
	1	1	☐
	1	2	☑
	1	3	☐
	1	4	☑
	1	5	☐
	1	6	☑
	1	7	☐
	1	8	☐
	1	9	☑
	1	10	☐
	1	11	☐
	1	12	☑
	4	1	☑
	4	2	☐
	4	3	☐
	4	4	☐
	4	5	☐
	4	6	☑
	4	7	☐
	4	8	☑
	4	9	☐
	4	10	☐
	4	11	☐
	4	12	☑

Tabla 0-1 *Principal* vista diseño y formulario

En el quinto diseño se utiliza un algoritmo que asigna tareas de forma aleatoria a cada trabajador. Se implementa un código que reparte tareas de forma aleatoria, mediante la función $\text{random} = \text{Int}((12 - 1 + 1) * \text{Rnd} + 1)$, a cada uno de los empleados para cada una de las horas. Una vez asignada la tarea se cuenta el núm. de empleados que realizan cada una de ellas y se almacenan los valores en la tabla contadores, ahora solo falta verificar que los empleados cubran los requisitos para dicha y tarea asignada. Si en algún momento el número de empleados es menor al requisito impuesto

$$(\text{If } (\text{rst3}(\text{"Tarea"}) = \text{t}) \text{ And } (\text{rst3}(\text{"Franja"}) = \text{f}) \text{ Then If } (\text{rst3}(\text{"Requisito"}) > \text{c}))$$

se muestra por pantalla un mensaje para que el usuario arregle esto mediante la forma manual.

El diseño final se implementó con la opción de agregar a un nuevo empleado en la BD, se agregó un calendario donde por seleccionar un día específico, y finalmente el proyecto genera informes o reportes.

Dentro de las conclusiones se menciona que el sistema desarrollado puede ser utilizado en cualquier empresa o negocio. En el proyecto se limitaron, los cálculos y algoritmos a un determinado número de tareas tanto en el caso manual como en el automático. Independientemente que el número de tareas se mantenga fijo, se permite al usuario la aplicación de la base de datos añadiendo nuevos empleados. El proyecto no fue aplicado en ningún problema (Mora, y otros, 2007).

Optimización en la asignación de tareas en un sistema de guardería forestal

En el artículo de Pradenas y Azocar se propone el uso de la heurística del vecino más cercano, que resuelve el problema del agente viajero (TSP – Travelling Salesman Problem), para optimizar la asignación de tareas en un sistema de vigilancia y gestión forestal. La solución propuesta es aplicada a la vigilancia de un patrimonio de 27.000 ha, conteniendo 102 predios y 402 nodos. El algoritmo es evaluado en dos épocas del año, invierno y verano, considerando que las salidas de los guardias desde el nodo de pernoctación es: (1) de manera aleatoria o (2) hacia el vecino más cercano.

El objetivo del TSP es visitar una sola vez un conjunto de ciudades obteniendo un costo mínimo.

La solución propuesta en este artículo es dividida en dos problemas a tratar: (1) El *Problema de M vendedores viajeros* donde se dispone de más de un vendedor. Los vendedores salen de un depósito común y regresan al mismo punto. Cada vehículo debe visitar por lo menos un nodo. El objetivo es minimizar la distancia total viajada por los vendedores. Cada vendedor debe viajar a través de un subconjunto de nodos, que incluye un depósito común, y cada nodo debe ser visitado exactamente una sola vez y por un sólo vendedor. (2) el *Problema de programación y ruteo de vehículos* considera un conjunto de tareas (o viajes) caracterizados por un origen, un destino, una duración y un tiempo de inicio fijo. Entre dos tareas sucesivas existe un período de tiempo de tránsito, con duración y costo.

El problema a resolver consiste en formar un programa y secuencias de tareas, donde cada tarea es desarrollada exactamente una vez. Muchos problemas en la administración de distribución de productos o servicios pueden ser formulados como un problema de ruteo de vehículos (VRP, “Vehicle Routing Problem”, el objetivo del VRP es atender a un conjunto de clientes con una demanda por servicio a un costo mínimo a través de rutas que se originan y terminan en un depósito central).

Con esto se busca minimizar el número de vehículos utilizados (*costos fijos*), minimizar la distancia total (tiempo) resolución (*costos variables*), minimizar alguna combinación de las dos primeras (*costos totales*), maximizar la función de utilidad basada en el servicio o conveniencia y maximizar la función de utilidad basada en la prioridad de los clientes.

La solución propuesta para el TSP se divide en 3 clases: procedimiento de construcción de rutas, procedimiento de mejora de rutas y procedimiento compuesto, dentro de las primeras se menciona la heurística del vecino más cercano, el algoritmo de Clark y Wright y los procedimientos de inserción entre otros, dadas las características del problema, el método utilizado será la del vecino más cercano considerando que la primera salida del día es en forma aleatoria a cualquiera de los nodos disponibles para cada guardabosque.

El procedimiento del vecino más cercano contiene básicamente las siguientes etapas:

Paso 1. Comenzar con cualquier nodo en forma aleatoria como el comienzo de la ruta.

Paso 2. Encontrar el nodo más cercano al último nodo incluido en la ruta, unir este nodo a la ruta, si es que son respetadas las restricciones de tiempo disponible, si no regresar al punto 1.

Paso 3. Repetir el paso 2 hasta que todos los nodos estén contenidos en una ruta o se haya completado el tiempo total disponible. El peor de los casos con esta heurística sucede cuando:

$$\frac{Largo\ de\ la\ ruta\ del\ vecino\ más\ cercano}{Largo\ de\ la\ ruta\ optima} \leq 1/2[\,lg\,(n)\,] + 1/2$$

Donde lg denota el logaritmo natural en base 2, y n el número de nodos en la red.

Los resultados obtenidos por este algoritmo evaluado, considerando una salida inicial al nodo más cercano del nodo de pernoctación, y una salida aleatoria desde el nodo de pernoctación[2]; además, se considera la aplicación durante la época de verano e invierno para el área en estudio.

Los resultados para una época de verano en términos generales para la salida aleatoria se dispone de una mayor distancia resolución (2%), un mayor costo total (3%) y un cumplimiento inferior en un 3% de la demanda total (Fig. 2).

Los resultados para la época de invierno nos muestran que para la salida aleatoria se obtuvo una mayor distancia total resolución (6%), un mayor costo total (6%) y un cumplimiento menor (1%) de la demanda total con relación a la salida al nodo más cercano.

Para la salida aleatoria, la proporción entre distancia resolución de traslado y de recorrido propiamente tal dentro de cada nodo, la primera es mayor en un 26%, es decir, se requieren más desplazamientos en llegar a los nodos que la distancia necesaria en su recorrido. Para la salida al nodo más cercano esta proporción baja a un 17%, lo que remarca lo indicado anteriormente con relación a la ubicación geográfica de los nodos de pernoctación de los guardabosques.

El análisis de la demanda para la época de invierno nos muestra que todos los guardabosques satisfacen la demanda en un 100%, con excepción del

[2] Pernoctación: Cada una de las noches que un viajero permanece o está registrado en un establecimiento de alojamiento colectivo o en un alojamiento turístico privado, siendo innecesaria su presencia física.

guardabosque 2 en que se llega a un 92% de cumplimiento, para el caso de la salida aleatoria, y un 93% para la salida al nodo más cercano (Fig. 3).

Guardabosque	Salida aleatoria				Salida al nodo más cercano			
	Distancia total Km	Tiempo total Hr	Costo total M$	Núm. Visitas Unidades	Distancia total Km	Tiempo total Hr	Costo total M$	Núm. Visitas Unidades
1	431	41	97	96	416	40	93	96
2	1.428	146	319	203	1.372	145	305	205
3	444	29	99	50	375	27	84	50
4	241	30	54	42	248	29	56	44
5	625	68	140	98	581	67	130	98
6	622	68	139	98	589	67	132	98
Total	3.791	382	848	587	3.581	375	800	591

Tabla 0-2 Resultado dist-tiem-costo invierno *(Optimizacion en la asignacion de tareas en un sistema de guarderia forestal, 2005)*

Guardabosque	Demanda unid.	Salida aleatoria Visitas Unid.	Cumplimiento Unid	Salida al nodo más cerca Visitas Unid.	Cumplimiento Unid.
1	96	96	100	96	100
2	221	203	92	205	93
3	50	50	100	50	100
4	44	44	100	44	100
5	98	98	100	98	100
6	98	98	100	98	100
Total	607	589	97	591	97

Tabla 0-3 Cumplimiento demnada para epoca de invierno *(Optimizacion en la asignacion de tareas en un sistema de guarderia forestal, 2005)*

Concluyendo que el algoritmo utilizado permite detectar que existen tiempos ociosos de algunos guardabosques que pueden ser corregidos mediante una mejor asignación de los predios bajo su responsabilidad.

El problema debe ser resuelto en forma integral, y no considerar a priori una asignación de predios a cada guardabosque, esta asignación debe ser realizada

por el modelo o se debe mejorar considerablemente la asignación de predios a cada guardabosque, para homologar la carga de trabajo a cada uno de éstos.

Se obtienen mejores respuestas en costos, distancia total resolución y tiempos de recorrido, con la solución que sale desde el nodo de pernoctación al nodo más cercano, con relación a la que tiene una solución aleatoria.

Por condiciones de demanda, la solución para la época de verano concentra una mayor distancia resolución, mayor costo y mayor tiempo de trabajo. Esto se fundamenta en que, durante la época de verano, aumentan los riesgos, dado que todo el patrimonio administrado se encuentra accesible.

El algoritmo propuesto entrega un plan diario de trabajo con una frecuencia preestablecida de verificación del patrimonio, lo que indudablemente produce una mejora sustancial al sistema actual utilizado en la empresa, que se basa en la experiencia del guardabosque, el cual define la ruta diaria a realizar.

Al proporcionar un plan diario de actividades, es factible dimensionar para el área de estudio la cantidad mínima de guardabosques requeridos para el trabajo, lo que permite en un futuro próximo realizar algún nivel de "outsourcing[3]", permitiendo una reducción considerable de los costos de administración. (Optimizacion en la asignacion de tareas en un sistema de guarderia forestal, 2005)

[3] Outsourcing: Es el proceso en el cual una empresa delega una porción de su proceso de negocio a una compañía externa.

Planteamiento del problema

En las fábricas de software y otras industrias se desarrollan proyectos, algunos muy grandes. En los proyectos colaboran decenas o cientos de personas. Existen personas que lideran los grupos o equipos de trabajo que asignan tareas por día, semana o mes. Cuando el proyecto es grande las tareas pueden ser cientos o miles. Cada trabajador tiene al menos una habilidad, pero es importante explotar todas las que posea.

El líder del equipo de trabajo debe conocer todas las habilidades de todos los trabajadores, sin embargo pueden ser muchas y se torna imposible tenerlas de memoria. Además los trabajadores se marchan y llegan nuevos. Al líder se le complicará entonces hacer la asignación de tareas buscando la optimización de la fuerza laboral.

Se propone un algoritmo que almacena el banco de tareas con su perfil y horas estimadas para su realización, por otro lado el conjunto de trabajadores con sus habilidades. Con lo anterior se busca hacer una asignación automatizada a los trabajadores. Dicho algoritmo asignará las tareas a los desarrolladores y el líder del proyecto solo le resta la supervisión de la ejecución de las tareas.

Por lo tanto se plantea la siguiente pregunta investigativa:

¿Cómo asignar automáticamente tareas a un grupo de trabajo aprovechando sus habilidades y considerando los tiempos estimados de las tareas?

Objetivo general

Desarrollar un algoritmo que permite asignar de forma automática tareas a desarrolladores, considerando el perfil de la tarea y del desarrollador para optimizar los tiempos y habilidades de los propios desarrolladores.

Objetivos Específicos

1. Definir la caracterización de las tareas de la fábrica de software.
2. Crear un almacén de datos para la gestión de tareas, desarrolladores y las asignaciones.
3. Probar el algoritmo para obtener resultados

Justificación

La asignación de tareas es crucial en los proyectos de software, porque es la base para lograr las metas de los hitos de cada módulo, así como de la entrega del proyecto completo. Cuando los proyectos son muy grandes se requieren de un equipo de trabajo grande o bien más de un equipo de trabajo. A mayor cantidad de desarrolladores se dificulta la asignación manual por parte del líder del proyecto.

Si la asignación es incorrecta genera tiempos muertos que atrasan el proyecto e incrementan los costos porque se pagan horas hombre en las cuales no hay productividad.

Es importante la estimación de cada tarea con el fin de hacer una asignación más equitativa y evitar efectos de estrés en los desarrolladores. Considerando que los desarrolladores estresados son menos productivos.

Es necesario contar con una explosión de tareas estimadas con algún perfil para asignarse de forma correcta. Por todo lo anterior, se propone un algoritmo que permita disminuir los tiempos muertos, la asignación incorrecta y hacer una planeación detallada más eficiente.

El líder de proyecto podrá dar de alta a los desarrolladores con los roles posibles de acuerdo a las habilidades que tiene cada desarrollador. Por otro lado, registrar las tareas estimadas en horas hombre y el algoritmo se encargará de asignar de forma automática buscando optimizar los tiempos y costos del proyecto.

Cuando la fábrica de software entrega a tiempo los proyectos, con la calidad requerida y no rebasa el presupuesto tendrá mejor imagen y reputación hacia sus clientes. Por otro lado los desarrolladores con una carga de trabajo congruente con los tiempos de su jornada laboral estarán más satisfechos y menos estresados.

Finalmente buscar que la fábrica de software sea más eficiente en el uso de recursos humanos, financieros y del tiempo en el desarrollo de los proyectos.

Metodología

Para lograr el desarrollo del algoritmo se siguió una serie de etapas que se detallan a continuación:

1. Redacción del contenido de los capítulos del marco teórico, a partir de las fuentes posibles como libros, artículos, revistas y otros.
2. Definición de la misión del algoritmo, su alcance, objetivos, uso y su aplicación. Con lo anterior se determinaron las funciones principales del algoritmo.
3. Determinación de los requerimientos funcionales y no funcionales del algoritmo. Los requerimientos son los que los líderes de proyectos requieren en el ámbito de desarrollo de proyectos de software.
4. Se diseñó el modelo arquitectónico del algoritmo. Podrá ser en forma de un diagrama de bloques de alto nivel o bien a través de un pseudocódigo. Incluirá el contenedor de datos y las vistas necesarias para los usuarios.
5. Desarrollo del algoritmo en un lenguaje de alto nivel. Se usó un lenguaje de programación orientado a objetos. Se propone sea en C++ o en Java.
6. Creación del escenario necesario para generar datos de un proyecto y generar una o varias ejecuciones para obtener resultados.
7. Redacción de los resultados y de las conclusiones de acuerdo a los resultados obtenidos en contraste al problema planteado y objetivos.

I. Capítulo I. Equipos de trabajo y Asignación de tareas

I.1 Recursos humanos

El objetivo que tiene la administración de los recursos humanos es mejorar el desempeño y las aportaciones del personal dentro de la organización. (WertherJr., y otros, 1995)

I.2 Equipo de trabajo

Los equipos de trabajo son grupos de individuos que mediante la labor interdependiente[4] cumple con objetivos designados, comunicándose en forma eficaz, y tomando decisiones que afectan dicha obra. Normalmente cuentan con cierto nivel de autonomía y desarrollan procesos para el logro de sus metas. La capacitación continua es un característico de un equipo de trabajo, al igual que la capacitación del líder o el jefe en cuanto a la administración y dirección de un equipo.

Los equipos de trabajo tienen un propósito o meta común y una declaración clara de la misión de dicho propósito. Ellos saben cuáles son los resultados que desean y pueden medir su avance hacia esas metas. Las metas de un equipo están alineadas con las de todo el departamento o la organización, y los miembros del equipo tienen un entendimiento y una noción muy clara de cómo sus esfuerzos cumplen las metas amplias de la organización. (Topchik, 2008)

I.3 Conformación y estructura

Para diseñar y definir una estructura básica de los equipos de trabajo, hay que tener en cuenta los requisitos que resultan más relevantes en el momento de integrarse en una gran organización cuya distribución básica son los equipos o las unidades de trabajo. De esta manera hay que hablar de *flexibilidad, compromiso, crítica complementaria*, *responsabilidad y autoconocimiento*. (S.L., 2007)

- *La flexibilidad*.- Importante en sus diferentes expresiones, como la tolerancia y el manejo del cambio. Necesaria para aceptar la opinión de un miembro del equipo.

[4]Interdependencia:dinámica de ser mutuamente responsable y de compartir un conjunto común de principios con otros

- *Compromiso*.- Mostrado en la aceptación de opiniones y sugerencias de los integrantes del equipo de trabajo.
- *Crítica complementaria*.- Reconoce las aportaciones de cada integrante del equipo de trabajo, además la crítica deberá ser positiva.
- *Responsabilidad*.- Reconocer errores propios y no culpar a terceras personas por ellos.
- *Autoconocimiento*.- Indispensable para integrar un equipo sólido. Identificar fortalezas y debilidades propias.

I.3.1 Fases del proceso de formación de un equipo

Los equipos son entidades que necesitan una evolución y un desarrollo para su formación. En este proceso se identifican cuatro etapas por las cuales pasa el equipo de trabajo. Cada una de las etapas se caracteriza por los siguientes contenidos y atributos:

- *Etapa de formación u orientación:* Los miembros del equipo comienzan a conocerse dejando así de ser solo un conjunto de individuos para ser un equipo de trabajo, establecen la relación de trabajo, las conductas y comportamientos adecuados y conocen la finalidad del trabajo o proyecto. (S.L., 2007)

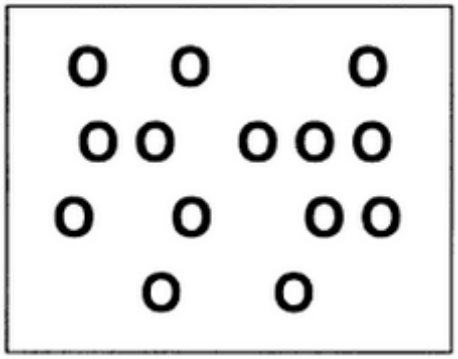


Imagen I-1 El cuadro muestra a un grupo desordenado y no cohesionado de personas en los momentos de toma de contacto *(Porret, 2008)*

- *Conflicto o insatisfacción:* Los miembros aceptan la *existencia* del equipo. La información es más exacta y sistemática, cada miembro quiere imponer sus objetivos y maneras de trabajar y resolver los conflictos y problemas, generando así un ambiente de confusión y tensión entre los integrantes

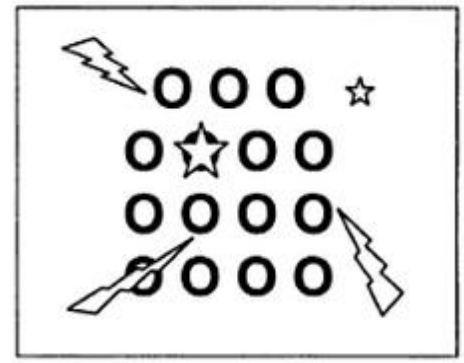


Imagen I-2Grupo no cohesionado en la fase tormentosa de posicionamiento de miembros *(Porret, 2008)*

- *Organización o resolución:* Definición de objetivos, roles y ámbitos de competencia para cada miembro. La información es clara y transparente y es usada para tomar decisiones, existe la posibilidad de manifestar diferencias y superar los conflictos, mejorar el desempeño y satisfacción laboral.

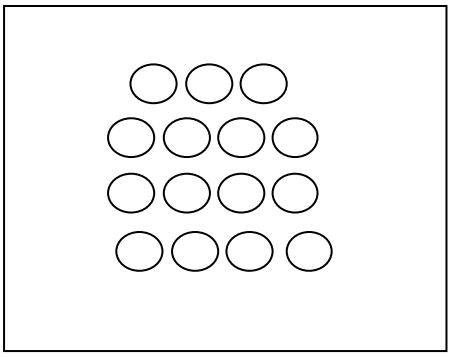


Imagen I-3Grupo en fase de organización de posicionamiento de miembros *(Porret, 2008)*

- *Realización o producción:* Estructura y objetivos definidos, reconocimiento de liderazgo, ejecución de tareas y mantenimiento de equipos, ambiente más cooperativo y satisfactorio. (S.L., 2007)

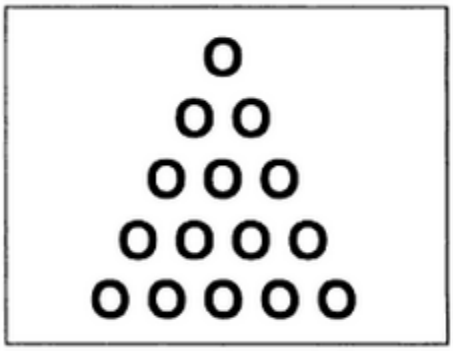


Imagen I-4Grupo ya cohesionado *(Porret, 2008)*

I.3.2 Tipos de equipos de trabajo

Larson y LaFasto (1989) proponen que hay tres tipos básicos de equipos, cada uno de ellos con objetivos diferentes.

1. *Equipos de resolución de problemas*. Cada miembro debe creer que el equipo será consistente y maduro en su enfoque para afrontar los problemas. Los miembros deben tener un alto grado en el proceso de resolución de problemas que se centra en los temas, en lugar de los puntos de vista o conclusiones predeterminadas.
2. *Equipos creativos. S*on responsables de explorar las posibilidades y alternativas, con el amplio objetivo de desarrollar un nuevo producto o servicio. Para que funcione un equipo creativo, necesita tener autonomía respecto a los sistemas y procedimientos, así como una atmosfera en la que las ideas no sean reprimidas prematuramente.
3. *Equipos tácticos.* Son responsables de ejecutar un plan bien definido. Para ello, debe haber una gran claridad de ideas y una definición de roles sin ambigüedades. (Muchinsky, 2002)

I.4 Líder del equipo

Es posible que nos hayamos preguntado qué es ser “líder” o quizá nos hemos cuestionado la siguiente pregunta “¿el líder se nace o se hace?”, o si alguien nos debe autorizar para ejercer “el poder del liderazgo”. (Hofstadt, y otros, 2006)

Los líderes de equipo sirven de modelos para sus compañeros de equipo. Si los líderes se involucran de manera abierta en el trabajo del equipo, proporcionan y aceptan conductas y feedback[5] de apoyo, es fácil que otros miembros del equipo hagan lo mismo. Los líderes de equipo son imprescindibles y tienen una tremenda influencia sobre los equipos y si los líderes son mediocres, también lo serán los equipos. (Muchinsky, 2002)

La definición de líder se ha relacionado a personas carismáticas, con características extraordinarias, magnetizadoras, capaces de influenciar de manera diferente a otras personas.

El liderazgo es el proceso interpersonal en el cual los administradores tratan de influir en sus empleados para que logren metas de trabajo prefijadas. (Hofstadt, y otros, 2006)

Líder es la persona capaz de ilusionar, movilizar y comprometer a lo seguidores en la realización de un proyecto de cambio, con el fin de alcanzar los objetivos establecidos. (Urcola, 2010)

[5]Feedback: Retroalimentación, conjunto de reacciones o respuestas que manifiesta un receptor respecto a la actuación del emisor, lo que es tenido en cuenta por este para cambiar o modificar su mensaje.

Imagen I-5 Líder de un equipo de trabajo

I.5 Perfil de un administrador de proyecto

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

- *Abstracción:* Entender y comunicar aspectos no tangibles, como visión y misión del equipo de trabajo. Deberá además, poder entender y ver el proyecto completo como una unidad y sus relaciones entre sus partes.
- *Concretización:* Utilizando los recursos e información disponibles, obtener conclusiones y tomar acciones específicas para manejar el proyecto.
- *Organización:* Distribuir eventos y actividades de acuerdo a los recursos y tiempos disponibles para llevar el proyecto al éxito.
- *Liderazgo:* Llevar a un equipo a lograr sus objetivos.
- *Experiencia:* Haber estado en situaciones similares en el pasado.
- *Creatividad:* Ser realista, tomando decisiones y tomando acciones cuando el plan actual no funciona.
- *Persuasión*: Encontrar y desarrollar argumentos para mejorar y ayudar en una situación.

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

- Escuchar y comunicar.
- Tomar decisiones y realizar acciones.
- Trabajar bajo presión. (Fuller, 2003)

I.6 Roles

Cuando un empleado entra en una organización, tiene mucho que aprender: niveles de desempeño esperados, reconocimiento de los superiores, reglas de vestimenta y exigencias de tiempo. Los roles facilitan el proceso de aprendizaje y en general, se definen como las expectativas de los demás sobre la conducta apropiada en una posición especifica.

Scott y sus colegas (1981) numeraron cinco aspectos importantes de los roles:

1. *Son impersonales*. La posición por si misma determina las expectativas, no el individuo.
2. *Los roles están relacionados con la conducta en el trabajo*. Un rol organizacional es la conducta esperada en un puesto concreto.
3. *Los roles pueden ser difíciles de delimitar.* El problema está en definir quién determina las expectativas. Ya que son otras personas quienes definen nuestros roles, ya que las opiniones sobre cuál debe ser nuestro rol son diferentes.
4. *Los roles se aprenden muy rápido y pueden producir grandes cambios conductuales.*
5. *Los roles y los puestos de trabajo no son iguales*; una persona puede desempeñar varios roles en un puesto. (Muchinsky, 2002)

Meredith Belbin define el *rol* como *nuestro modo individual, personal, de comportarnos, de contribuir a la tarea y de relacionarnos con otras personas en el trabajo.*

El conocer nuestros propios roles nos permite comprender nuestra propia identidad, gestionar nuestros puntos fuertes y débiles, proyectar una buena imagen personal y trabajar de la mejor manera posible dentro de un equipo.

Los roles pueden ser diferenciados en *rol funcional* y *rol de equipo*:

- *Rol funcional* consiste en la habilidad, experiencia y conocimientos individuales, necesarios para desempeñar una función concreta dentro del equipo y así contribuir con un logro eficaz para los fines del mismo.
- *Rol de equipo* son las características individuales que definen el comportamiento dentro de un equipo de trabajo. Dentro de este rol se identifican nueve patrones de comportamiento, los cuales a su vez son clasificados en tres categorías:*(Tabla 1.1)*

Categorías	**Patrones**
Roles de acción Orientación de las personas hacia el desempeño de tareas.	Impulsor (IS)
	Implementador (ID)
	Finalizador(FI)
Roles sociales Orientación de las personas hacia las relaciones con otros.	Coordinador (CO)
	Investigador de recursos(IR)
	Cohesionador(CH)
Roles mentales Orientación de las personas hacia el mundo de las ideas.	Cerebro (CE)
	Monitor evaluador(ME)
	Especialista(ES)

Tabla I-1Patrones de comportamiento *(Caldas, y otros, 2010)*

Los roles extremos son el cerebro (ideas), el finalizador (Tareas) y el cohesionador (personas), y los otros roles se sitúan en posiciones intermedias.
El rol más importante es el coordinador, que garantiza la cohesión y buen funcionamiento del equipo, por lo cual se sitúa en una posición central. (Caldas, y otros, 2010)

I.6.1 Los roles en el desarrollo de software

Hay muchos roles que las personas deben desempeñar en los proyectos de desarrollo, e incluso se requerirá que algunas personas desempeñen múltiples roles durante el esfuerzo. Los siete roles son: *programador, cliente, probador, rastreador, entrenador, consultor y gran jefe.*

El rol del *Programador* : Debe contar con excelentes habilidades técnicas para programar, refactorizar y realizar pruebas unitarias al código que escriba, buena disposición para abordar los problemas más difíciles, aprender de otros, compartir el código y el diseño y tener el valor de superar cualquier temor de incompetencia o fracaso al enfrentar nuevos problemas.

El rol del *Cliente*: Debe adoptar nuevas cualidades, algunas muy similares a las del programador conservando la esencia de lo que necesita el sistema, el cliente más apropiado para el equipo es alguien que será usuario del sistema y conoce la funcionalidad de negocios que este requiere, debe aprender a escribir relatos de usuarios, pruebas funcionales para las aplicaciones que generen los programadores y tomar decisiones acertadas. Además de demostrar que tiene valor para enfrentar problemas graves referentes a la programación del proyecto al funcionamiento del sistema.

El rol del *Probador*: a los programadores se les pide realicen pruebas unitarias y de funcionamiento del nuevo código que se haya escrito. El programador también necesita comunicarse con el cliente sobre las pruebas de funcionamiento, realizar pruebas con regularidad, dar mantenimiento a las herramientas de prueba y elaborar informes precisos acerca de los resultados de las pruebas.

El rol del r*astreador*: da seguimiento al progreso general del grupo calculando el tiempo que toman sus tareas y el progreso general hacia sus metas, realiza estimaciones de tiempo, pero también da retroalimentación acerca de las estimaciones del equipo, además funge como memoria del equipo, al dar seguimiento a los resultados de todas las pruebas de funcionamiento, registra los defectos reportados y el nombre del programador a cargo del manejo de cada defecto.

El rol del *Entrenador*: ayuda al equipo de trabajo a conservar la calma, moldea la situación de manera indirecta y solo de vez en cuando necesita retirar con firmeza el mando a un programador errático, volver a encarrilarlo y devolverle las riendas otra vez, se encarga de resaltar las cualidades de todos los miembros del equipo.

El rol del *Consultor*: ayuda al equipo a resolver sus problemas ayudando así a que recuperen la confianza en sí mismos.

El rol del *Gran jefe*: demuestra disposición para apegarse a los valores y principios a los que el equipo se apegue, además tiene la capacidad de señalar un error si el equipo se desvía del camino. Se encarga de conseguir que la comunicación fluya sin convertirse en una ola de rumores. (E. Kendall, y otros, 2005)

I.7 Comunicación

La comunicación une a las personas dentro de un sistema; en las organizaciones fija el índice de valoración que ayuda a determinar el grado de autoridad y su nivel de obediencia. Todo sistema de comunicación se enfrenta con dificultades que pueden considerarse barreras, en términos de organización:

- *Barrera defensiva*: impide la comunicación de abajo hacia arriba por diferencias sociales, temor o miedo al sabotaje.
- *Barrera egoísta*: el individuo no comunica datos interesantes para así poder contar con puntos a su favor y triunfar tal vez sobre los demás.
- *Barrera de indiferencia*: desinterés o ignorancia del superior cuando trata de orientar a su personal hacia el cumplimiento de sus funciones. (Hayes, 2003)

Aunque son muchas las ventajas que presenta una buena comunicación interna en la organización, pues se ha demostrado en algunos estudios que son pocas las empresas en las cuales sus empleados están satisfechos con los sistemas de comunicación existentes. (Palomo, 2010)

En la comunicación y negociación es importante determinar normas que faciliten la interrelación del conjunto de la organización a nivel interno y externo. Así se busca que cada parte haga lo que le interesa al conjunto.

Cabe mencionar que la complejidad de un problema radica en la importancia, urgencia e interés personal del receptor. La urgencia está en función del objetivo del mensaje. Así el interés aumenta o disminuye el impacto del mensaje según como se analice su contenido en función de sus aspectos positivos o negativos. Cualquier mensaje con buenas o malas noticias, atrae la atención e interés del receptor; la dificultad para el emisor se presenta al tratar de impactar con una comunicación complicada. (Ramírez, 2005)

I.7.1 Comunicación en los equipos de trabajo

Una buena comunicación en el equipo de trabajo es fundamental tanto para que el grupo consiga sus objetivos como para reducir el estrés y aumentar la motivación y la satisfacción de sus miembros. Una de las características básicas que define un equipo de trabajo es la *interacción*, es decir, que para que haya un equipo es necesario que los individuos que lo forman se comuniquen y se relacionen entre sí.

La comunicación en los equipos es diferente si el grupo es formal o informal. En los *equipos informales*, la comunicación es producto de las relaciones sociales entre miembros de una organización y surge cuando uno de ellos necesita comunicarse con el otro sin que exista un canal formal para ello.

En los *equipos formales*, la comunicación también lo es, ya que utilizan los medios establecidos para este propósito (informes, escritores, etc.) y se transmiten mensajes cuyo contenido se relaciona con el desempeño del trabajo. En estos grupos la comunicación se establece de acuerdo a los criterios de la política de la

organización, variando desde un modelo de funcionamiento autoritario, donde cada miembro solo las transmiten a los que le suceden en la misma, hasta el modelo de la comunicación participativo, con una estructura menos rígida en la que los miembros pueden comunicarse en todas direcciones.

El modelo de *comunicación autoritario* se da en grupos con una estructura muy jerárquica. Se basa en un sistema burocrático y jerarquizado, donde el jefe es un líder reconocido con capacidad para imponer sanciones. El líder también es la persona que asume todas las decisiones y controla todas las operaciones. Este modelo de comunicación tan centralizado, se puede representar mediante la figura de una estrella, en la que el jefe ocuparía el punto central, el lugar donde convergen todas las informaciones.

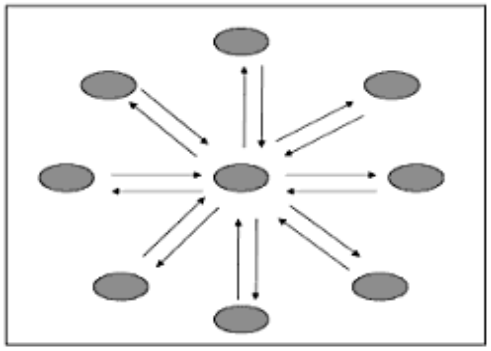


Imagen I-6Modelo de comunicación autoritario *(Hofstadt, y otros, 2006)*

Dentro de este modelo la comunicación puede ser *vertical ascendente* o *descendente*. La comunicación *vertical ascendente* es la que fluye de los miembros que ocupan un nivel inferior en el equipo de trabajo a otro más alto.

Imagen I-7Comunicación vertical ascendente

La comunicación *vertical descendente* es la que fluye de un nivel del equipo a otro más bajo (de jefe a subordinado). Normalmente la comunicación se relaciona con instrucciones e información acerca de la tarea a realizar o retroalimentación sobre el rendimiento del empleado.

Barreras de la comunicación descendente		
Causa	Efecto	Ejemplo
Excesivos niveles jerárquicos	Distorsión de la información	Los directivos no transmiten a sus subordinados la información que han recibido de sus jefes
Error al elegir el canal adecuado	Falta de efectividad	Utilizar el correo electrónico para notificar una modificación urgente, cuando los trabajadores no leen el correo con regularidad o no tienen conexión a internet
Inadecuada cantidad de información	Se desvirtúa el sentido del mensaje	Se informa a los trabajadores de los resultados económicos de la empresa en el ejercicio, dándoles un estudio detallado que solo puede ser útil a especialistas.

Tabla I-2 Barreras de la comunicacion

Imagen I-8 Comunicación descendente

Cabe mencionar que existe la *Comunicación participativa,* que es un tipo de modelo que cualquier equipo de trabajo desea, pues se caracteriza porque es descentralizado y favorece la aportación e implicación de todos en los objetivos, planes y logros, lo que produce una mayor satisfacción entre los miembros del equipo y se otorga una mayor independencia a sus componentes.

La comunicación participativa puede ser de dos tipos:

- *Comunicación entre compañeros de un mismo equipo de trabajo*: posibilita el trabajo y la coordinación dentro del grupo, y también sirve de soporte de la relación, lo que incrementa la satisfacción de los miembros.
- *Comunicación intergrupos*: facilita la coordinación de actividades, resolución de problemas y solución de conflictos entre los distintos equipos. (Hofstadt, y otros, 2006)

Tipo de comunicación	Funciones
Comunicación ascendente	• Clarificar si los subordinados han recibido y comprendido los mensajes que se les han dirigido. • Obtener información para corregir y reevaluar sus objetivos, planes y métodos • Ayuda a tomar decisiones a través de sugerencias, fomenta la crítica sana y reduce las presiones y tenciones emocionales. • Contribuir al conocimiento de las necesidades y expectativas de los miembros del equipo
Comunicación descendente	• Conseguir una mayor coordinación entre los diferentes miembros del equipo • Aumentar su motivación • Mantener informado al individuo para que contribuya a la consecución de los fines y objetivos

Tabla I-3Funciones de la comunicación ascendente y descendente *(Hofstadt, y otros, 2006)*

Las barreras que se pueden dar es la importancia que cada departamento tenga, puede afectar a que sus opiniones sean más tenidas en la comunicación formal, También en cada departamento se puede utilizar un lenguaje más técnico con otros departamentos y puede dificultar la relación entre ellos. (wiksem)

I.8 Asignación de tareas

I.8.1 Tareas

Es la resultante de una serie de factores que se inicia en la necesidad de producir, adquirir o vender algo, y en la elaboración de procesos y métodos con la intervención *hombre-máquina* mediante la asignación de funciones que desembocan en una determinada actividad específica denominada *tarea.* (Ramírez, 2005)

Las tareas son los procesos individuales y complementarios que forman o llevan a la meta del proyecto. El software planificador debe permitir: definir una tarea e identificarla de las demás, incluir en ella los recursos y la cantidad de estos que se necesita para poder llevarla a cabo, la fecha en la que la tarea se inicia y la dicha de finalización. Las tareas pueden ser paralelas con otras, semiparalelas, ligadas inicio-fin y desligadas. Estas ligas entre tareas se conocen como enlaces y existen de diversos tipos como tipos de ligas existen. (RODRIGUEZ, y otros, 2006)

I.8.2 Análisis del trabajo

Es la operación mental que valora los factores antes mencionados con el fin de determinar posibilidades, y así optimizar el balance de fuerzas existentes en función de una operatividad positiva del sistema.

Hoy en día el concepto empresa es sinónimo de funciones, las cuales son diversas actividades que ésta desarrolla para la consecución de sus fines, aunque lo normal es que este concepto se restrinja a la relación *trabajo humano-producción,* considerando que elevar la productividad no se logra necesariamente con el esfuerzo laboral, sino al hacer más eficiente este esfuerzo.

Una vez conocidos los objetivos, la estructura empresarial se basa en las necesidades funcionales de acuerdo a las actividades rutinarias, determinación de dichas necesidades, su evaluación, su agrupamiento conforme características comunes y su reordenamiento mediante diversas dependencias.

I.8.3 Perfil del puesto

Es el resultado de la descripción de los elementos que lo componen: *tarea-hombre-entono*, es decir, enumeración de las necesidades intelectuales y físicas que demanda el puesto, presentación de los medios auxiliares por utilizar, grados de responsabilidad e iniciativa necesarias, el tipo de decisiones por tomar, tipo de relaciones y de dependencias dentro del sistema, promociones y posibilidades de desarrollo. Su delimitación se basa en:

- *Las tareas del puesto* determinan sus aspectos operativos programando la actividad en forma cuantificable.
- *La carga de trabajo* es la cantidad de trabajo de un puesto cuando intervienen hombre y máquina. La carga es el trabajo del operario con respecto a las maquinas asignadas originando dos variantes, según la asignación sea aleatoria o sincronizada.

Las cargas de trabajo se asignan de acuerdo con las cualidades del individuo, las características del puesto, las condiciones de tipo técnico administrativas y el proceso de producción.

La asignación de *cargas de trabajo con servicio aleatorio* busca la asignación óptima entre la interferencia de máquinas y la inactividad del operario. Dicha optimación permite asignar el número de máquinas indispensable en el que el valor de la interferencia de máquinas más el tiempo improductivo del operario sea mínimo.

Las *cargas de trabajo en acciones sincronizadas* tratan de reducir al máximo la interferencia de máquinas y la inactividad del operario; pero ahora es posible organizar asignaciones sincronizadas sin interferencia de máquinas e inactividad del operario.

El *reparto del trabajo* introduce el concepto de carga vertical de trabajo para dar mayor profundidad a la tarea, así como exigencias de conocimientos y habilidades de alto nivel que dan mayor autonomía y responsabilidad al planificar, dirigir y controlar el proceso de trabajo. La *asignación de tareas* en este caso involucra los conceptos de variedad de materiales y operaciones, autonomía para elegir el método, y una mayor responsabilidad y capacidad del individuo.

Las tareas pueden ser mentales, de relaciones humanas y físicas, comerciales, técnicas o administrativas, con objetivos bien cuantificables y cualificables.

El análisis de la tarea comprende el análisis a priori o análisis global para identificar el puesto, y el análisis a posteriori para comprobar resultado y efecto de las condiciones de trabajo.

El *análisis a priori* comprende la determinación del objetivo general; la descripción de las necesidades en medios, equipo y personal, para cumplir con la finalidad de la tarea; cuantifica las posibilidades de la empresa; implica el estudio del flujo de relaciones personales, de instrucciones recibidas y dadas, y de trabajos recibidos y entregados.

Análisis de condiciones técnicas:

- Proceso del trabajo: diagramas de proceso de operación y de recorrido, diagrama hombre-máquina, de materiales, de ambas manos, etc.

- Carga de trabajo que requiere la tarea: capacidad necesaria, fluctuaciones en la carga, dificultades en su ejecución.

Análisis de las condiciones ambientales:

- Ambiente de trabajo, perturbaciones, seguridad y su localización e interdependencia con las demás tareas del sistema.

El *análisis a posteriori* estudia los resultados desde la puesta en marcha del puesto; compara, mediante un balance de eficacia, las metas propuestas; e identifica las deficiencias de organización, de operatividad de la tarea y condiciones de entorno. Los aspectos por analizar son:

- Análisis de las desviaciones operativas.
- Dificultades del sistema hombre-maquina
- Funcionamiento de los diversos procesos de trabajo
- Etc.

El análisis de tarea las estudia con el objetivo de optimizar su ejecución, determinando alternativas válidas posibles a fin de obtener mejores resultados. Para ello es conveniente hablar en términos de resultados, es decir, de análisis de las áreas de eficacia de la tarea.

El área de eficacia de la tarea al igual que la del trabajador, es la medición de los resultados esperados de ambos para efectos de la tarea y además como elemento de consulta y sugerencia cuando se trata de individuo. Así, la eficacia total del puesto es óptima sí su configuración se fija en función de la producción de la tarea y del trabajador. La eficacia de la tarea debe identificarse con claridad en su conformación y distribución de responsabilidades: si dos personas son responsables de una misma tarea, el puesto de trabajo no está bien delimitado, y el directivo no ha planificado ni controlado las asignaciones.

La descripción de la tarea sirve para precisar su perfil:

- Actividad-resultado-valoración
- Fases-reparto de tiempos-mejora de rendimiento
- Dificultades-soluciones-mejoras-seguridad
- Esfuerzo-mayor o menor-rendimiento
- Nivel de trabajo-status-aplicación esfuerzo
- Calidad-mental/física-aplicación
- Equipos empleados-equipos/herramientas
- Condiciones de trabajo-ambientales
- Coordinaciones-diversos
- Factor humano-calificación

Toda tarea se puede dividir en una serie de elementos analizables bajo el nombre de análisis de procesó, el cual se aplica a los diferentes procesos o etapas por las que pasa un producto. (Ramírez, 2005)

I.9 Calendarios de trabajo

Es el proceso de decidir cómo se organizará el trabajo en un proyecto, como tareas separadas, y cuándo y cómo se ejecutaran las mismas. Se estima el tiempo calendario para completar cada tarea, el esfuerzo requerido y quien trabajara en las tareas identificadas. Al igual se debe estimar los recursos necesarios para completar cada tarea, el tiempo que se necesitará el hardware especializado y cuál será el presupuesto de viajes.

El calendario inicial se utiliza para planear la asignación del personal al proyecto y comprobar el avance de este frente a los compromisos contractuales.

En la calendarización de proyectos dirigidos por un plan se realiza la división del trabajo total de un proyecto en tareas separadas además se estima el tiempo requerido para completar cada tarea. Por lo general las tareas deben durar menos de una semana, pero no más de dos meses. La cantidad máxima de tiempo para

cualquier tarea debe durar alrededor de ocho a 10 semanas. Si tarda más que esto, la tarea debe subdividirse para la planeación y calendarización de proyecto.

El desarrollo de algunas tareas se realiza de forma paralela, con distinto personal que trabaje en diferentes componentes del sistema. Es necesario coordinar las tareas paralelas y organizar las actividades para que la fuerza de trabajo se desempeñe de manera óptima y no introduzca entre las tareas dependencias innecesarias.

Se debe tomar en cuenta que el personal que trabaja dentro del proyecto puede enfermar o cambiar de trabajo, o en su defecto el software o hardware utilizado pueden fallar, lo cual ocasionaría retraso en la entrega del proyecto, por lo cual es recomendable el ampliar el tiempo estimado dentro de la calendarización para enfrentar dichos problemas de manera anticipada o añadir un factor de contingencia, el cual dependerá del tipo de proyecto, los parámetros del proceso y la calidad y experiencia de los ingenieros del software que trabajan en el mismo. (Sommerville, 2011)

I.9.1 Representación del calendario

La planificación de un calendario es una tarea bastante más compleja de los que pueda parecer en un principio. Antes de formalizar un calendario de tareas para el proyecto, es necesario realizar un estudio para estimar el tiempo necesario para cada tarea y las dependencias entre las mismas. (Fernández, 2006)

La representación de proyectos por medio de tablas u hojas de cálculo dificulta la visualización de las relaciones y dependencias entre actividades, por lo cual se han desarrollado representaciones graficas alternativas a los calendarios de proyecto, los cuales son más fáciles de leer y entender.

Existen dos tipos de representación:

- *Graficas de barras o graficas de Gantt* (Henry Gantt), basada en el calendario, las cuales señalan al responsable de cada actividad.
- *Redes de actividad*, diagramas de red que muestran las dependencias entre las diferentes actividades que constituyen el proyecto.

Las actividades de proyecto son el elemento de planeación básico. Cada actividad cuenta con:

- Una duración en días o meses calendario
- Una estimación del esfuerzo, la cual refleja el número de días- hombre o meses-hombre para completar el trabajo.
- Un plazo dentro del cual debe completarse la actividad.
- Un punto final definido. Representa el resultado tangible de completar la actividad.

La definición de hitos dentro de la planeación de un proyecto es importante, puesto que dentro de cada etapa del proyecto puede realizarse una valoración de avance la cual deberá ser documentada dentro de un reporte breve. (Sommerville, 2011)

I.9.2 La asignación de tareas a personal

Es un modelo de gestión de proyectos ideal, una vez utilizadas técnicas como el WBS (work breakdown structure o estructura de desglose de trabajo), disponemos de un conjunto de tareas, que se realizan de forma correcta, nos llevaran a alcanzar nuestro objetivo.

Dada la diferencia de productividad dentro de los proyectos, hay que hacer algunas salvedades [Montesa, 1999]. El tema se puede enfocar desde las vertientes de persona, equipo y tarea, o desde las fases de proyecto: planificación y control. Los enfoques clásicos se centraban en asignar las personas durante la planificación y en base a sus conocimientos para realizar la tarea, pero se han

desarrollado modelos que contemplan los comportamientos y hasta las emociones de los empleados y su forma de gestionarlas.

Dado que lo que nuestro material de partida, además de las tareas, son las personas de que disponemos, es interesante citar las habilidades que se distinguen en estos [Katz, 1974]:

- Habilidades técnicas: conocimientos, su aplicación y la experiencia.
- Habilidades humanas: capacidad de comprender y motivar a los demás, competencia interpersonal.
- Habilidades conceptuales: capacidad mental de análisis y diagnóstico en situaciones complejas.

Como se ve este tipo de taxonomía no es nuevo, pero se ha refinado y nos permite hablar con más propiedad, a la hora de asignar recursos humanos.

I.9.3 La asignación basada en los conocimientos

Dada una tarea, evaluamos los conocimientos, capacidades y experiencias requeridas. Se evalúan estas características en los candidatos disponibles y se asigna aquel que mejor las satisface. Se centran en aspectos cognitivos del individuo.

I.9.4 La asignación basada en el conjunto de tareas asignadas

Los estudios sobre el factor humano en la empresa, aconsejar crea puestos de trabajo que sean variados. Esto también es válido para los proyectos, con la diferencia que esta combinación será especifica en cada proyecto. La ampliación de las tareas, el enriquecimiento del trabajo y la participación directa en el trabajo repercute en la motivación y fidelidad del empleado. Los trabajos que estudian estos aspectos [Rodríguez, 94] identifican los siguientes atributos motivadores del trabajo: Variedad, identidad, trascendencia, autonomía y feedback[6].

[6] Retroalimentación, conjunto de reacciones o respuestas que manifiesta un receptor respecto a la actuación del emisor, lo que es tenido en cuenta por este para cambiar o modificar su mensaje

I.9.5 La asignación basada en la cantidad de personas por tarea.

La duración de una tarea no siempre se reduce al asignar más personas a esta. Esto depende del tipo de tarea, que desde este punto de vista, se han clasificado en cuatro grupos [Brooks, 75]:

- Tareas descomponibles de forma perfecta: hombres y meses son intercambiables, no necesitan comunicarse entre ellos.
- Tareas en-descomponibles: cuando la tarea se asigna a varios trabajadores no se obtendría una duración menor.
- Tareas que requieren comunicación: la tarea puede ser seccionada pero requiere que las personas que realicen cada subtarea se comuniquen para información y productos.
- Tareas complejas: la tarea puede descomponerse, pero las interacciones son complejas y todo el esfuerzo añadido se utiliza en la comunicación.

El hecho de que los implicados en una tarea dispongan de un sistema de comunicación eficiente, reducirá el esfuerzo a realizar para completar un trabajo.

I.9.6 Una visión integrada de la asignación de tareas

Al identificar las tareas de un proyecto, debemos disponer de una lista de verificación, en donde se muestren todos los aspectos mencionados, y no solo los conocimientos requeridos.

De modo que tengamos que reflexionar sobre la dificultad intrínseca de la tarea, vislumbrando la necesidad del que la realice debe estar motivado y que sea capaz de mantener esta motivación cuando surjan problemas.

La complejidad de la tarea a nivel interno, de modo que al asignar varias personas, los perfiles de cada uno de ellos, así como su inteligencia emocional, muestre que tipo de tarea será e incluso tratar de incluir en el grupo de personas con el rol de facilitadores, que si bien no serán muy productivos a niveles técnicos, reducirán el coste global de la tarea, por facilitar la comunicación. (JOM, y otros, 1999)

I.10 Estimación de tamaño de tareas

El equipo de planificación puede optar por varias opciones a la hora de estimar el tiempo de cada una de las tareas. Para ello se pueden utilizar los siguientes métodos.

El primer método es equiparar el tiempo de una tarea al tiempo más probable (TMP) que será necesaria para finalizar la tarea. Una segunda técnica es calcular la media entre el tiempo óptimo (TO) y el tiempo pésimo (TP) de una tarea. El tiempo óptimo es el tiempo mínimo necesario para realizar una tarea si no se producen interrupciones o retrasos, mientras que el tiempo pésimo es el tiempo necesario para finalizar la tarea si todo sale mal. Finalmente el tercer método se basa en la experiencia y consiste en calcular la duración esperada (DE) a través de la siguiente fórmula:

$$\mathrm{DE} = \frac{TO + (4 * TMP) + TP}{6}$$

Donde DE denota duración esperada, TO tiempo optimo, TP tiempo pésimo y TMP tiempo más probable. (Fernández, 2006)

I.10.1 Dependencias entre tareas

En cualquier proyecto formado por actividades y tareas puede encontrarse distintos tipos de dependencias entre ellas. La dependencia más habitual se produce cuando una tarea no puede iniciarse antes de que otra finalice. Por ejemplo, y a nivel de etapas, es necesario finalizar la etapa de planificación de sistemas antes de comenzar la etapa de análisis de sistema. En total existen cuatro tipos de dependencias:

- *Acabar de empezar* (A-E): una tarea que finalizar para que otra pueda empezar
- *Empezar para empezar* (E-E): el inicio de una tarea se produce cuando otra tarea es iniciada.
- *Acabar de acabar* (A-A): dos tareas deben acabar al mismo tiempo

- *Empezar para acabar* (E-A): el inicio de una tarea provoca la finalización de otra tarea.

En este punto es recomendable crear una tabla en donde aparezcan todas las tareas del proyecto, así como toda la información recopilada hasta el momento de cada tarea. La tabla debe tener como filas todas las tareas, mientras que en las columnas aparece toda la información relacionada con las tareas: un identificador, una descripción, la duración esperada, las tareas con las que existe una dependencia y el tipo de esta dependencia (A-E, E-E, A-A y E-A). (Fernández, 2006)

I.11 Control de tareas

El control de tareas consiste en comparar la actuación real con los estándares para esas tareas, y emprender después la acción apropiada si parecían alguna divergencia significativa. Y puede definirse de la siguiente manera:

El control de tareas es el proceso que sigue para asegurar que unas tareas determinadas efectivamente y con eficacia.

El control de tareas se distingue del control de gestión por las particularidades que se mencionan a continuación:

- El sistema de control de gestión es similar en toda la organización; es esencial para agregar ingresos y gastos de cada una de sus partes integrantes. Cada tipo de tarea requiere un sistema diferente de control de las tareas.
- En el control de gestión, algunos directivos interactúan con otros; en el control de tareas, no intervienen personas o la integración es entre un directivo y un empleado.
- El control de gestión, el punto de enfoque esta en unidades de organización llamadas centros de responsabilidad; el control de tareas, el punto de enfoque esta en las especificaciones que correspondan.

- El control de gestión se refiere a actividades que no están especificadas, y la dirección decide que se ha de hacer sin salirse de las limitaciones generales impuestas por los planes estratégicos; el control de tareas se refiere a tareas que están especificadas y, en cuanto a la mayoría de ellas, se requiere poca ponderación o ninguna con respecto a lo que debe hacerse.
- El control de gestión se centra tanto en la planificación como en la ejecución; el control de tareas lo hace, ante todo en la ejecución. (RAMÍREZ, 2005)

I.12 Asignación automática

I.12.1 Diagrama PERT/CPM

El diagrama PERT (Project Evaluationand Review Technique) es un modelo que consiste en un conjunto de nodos numerados, que representan tareas o actividades, y un conjunto de vectores etiquetados uniendo los nodos, que representan las dependencias. Un diagrama PERT se lee desde siempre de izquierda a derecha, estableciendo de esta manera una secuencia cronológica de eventos/hitos y tareas.

El diagrama CPM (Critical Path Method) constituye un modelo de red muy parecido a Pert. CPM fue creado por J.E. Kelley y M.R: Walker en la empresa DuPont (Kelley, 1961) en la década de los 50 con el objetivo de crear programas de mantenimiento de o plantas químicas de una manera eficiente (Kerzner, 2003). Su desarrollo fue paralelo a Pert, sin tener ninguna relación evidente.

La principal aportación de Pert es el cálculo de las estimaciones de tiempo y como se determina el camino crítico. Para ello, utiliza una combinación de las estimaciones optima, peor y más probable. El cálculo de la estimación de tiempo se basa en suponer que los tiempos de todas las actividades van a seguir una distribución beta.

CPM, sin embargo, se centra en el cálculo del camino del proyecto, esto es, el camino más largo entre el inicio y la finalización del proyecto. Para ello, y suponiendo que están definidas la duración de las actividades y sus dependencias, realiza dos pasadas: una hacia delante para hallar fecha de comienzo más temprana(ES) de cada tarea y otra hacia atrás para hallar la fecha de finalización más tardía (LF) de cada tarea. Este método es posible cuando existe un calendario o hitos que deben cumplirse, y cuando se puede determinar la duración Aproximada de las actividades. En proyectos software la utilización de este método supondría definir detalladamente a nivel de actividad el proceso de desarrollo y definir de manera real la duración de cada actividad.

Inicialmente, ambos métodos fueron desarrollados de manera independiente, aunque en la actualidad, CPM y PERT se utilizan de manera indistinta para la representación de diagramas en red deterministas y, por lo tanto, es adecuado referirse a ellos con las siglas PERT/CPM, donde se comparte una notación común, y una combinación de ambos métodos para la estimación del camino crítico. (Tuya, y otros, 2007)

II. Capítulo II.- Uso de Arreglos y Archivos

II.1 Vectores

Un vector es una estructura de datos con un bloque de memoria contiguo, al igual que un arreglo. Debido a que las localidades de memoria son contiguas, pueden accederse aleatoriamente de modo que el tiempo de acceso de cualquier elemento del vector sea constante. El almacenamiento se maneja en forma automática de modo que en un intento por insertar un elemento en un vector completo, un bloque de memoria más grande se asigna para el vector, los elementos del vector se copian al bloque nuevo y el bloque viejo se libera. El vector es por lo tanto un arreglo flexible, es decir, un arreglo cuyo tamaño puede cambiar dinámicamente. (Drozdek, 2007)

Los arreglos o vectores se consideran elementos de dos tipos primitivos o de referencia. Para hacer referencia a un elemento específico en un arreglo debemos especificar el nombre de la referencia al arreglo y el número de la posición del elemento en el arreglo. (Deitel & Deitel, 2004)

El vector incluye un conjunto amplio de mètodos poderosos para manipular la matriz subyacente y sus elementos. Son tres los constructores de esta clase:

- Vector(int initialCapacity, int capacityIncrement);
- Vector (int initialCapacity);
- Vector ();

Se puede presentar un vector como una matriz de objetos (Object),en primera instancia, no existe nada util en la matriz, si bien se cuenta con la capacidad para almacenar un cierto numerode *Object.* Su tamaño logico se incrementa a medida que se colocan objetos en la matriz, hasta llegar el momento en que ha aumentado hasta ser igual que su capacidad. Despuès es necesario ampliar esta ultima para añadir un nuevo objeto a la matriz. El primero de los tres constructores

antes mencionados hace posible asignar la capacidad inicial y la magnitud de su incremento cuando sea necesario. (Decker, y otros, 2001)

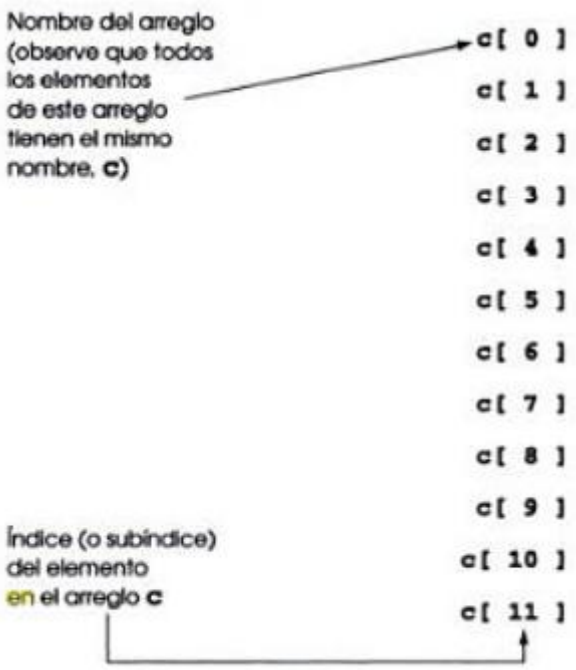


Imagen II-1 Arreglo de doce elementos

Los arreglos pueden guardar cualquier cosa, enteros números flotantes, cadenas, botones, barras de desplazamiento, cualquier objeto de la biblioteca, cualquier objeto creado por el programador. (Bell, y otros, 2003)

En *Java* un arreglo se declara de la misma forma que cualquier otra variable, generalmente al principio de una clase o de un método. Un arreglo es un objeto y, debe crearse mediante el operador *new,* para crear una instancia de la clase A*rray.* P*or ejemplo:*

```
Int [ ] edades;

String [ ] grupo;
```

Imagen II-2 Declaración de arreglos

La variable llamada edades esta lista para hacer referencia a un arreglo de enteros,al crear un arreglo con *new* tiene que indicar su longitud(total de elemnetos que va a contener) y el tipo de objetos a contener. (Bell, y otros, 2003)

```
Edades = new int [6];
Grupo = new String [4];
```

Imagen II-3 Referencia de un arreglo

¿Porque usar vectores y no matrtices?

Al utilizar un vector implica costos minimos de uso de memoria y rendimiento, ademàs de permitir el acceso a numerosos metodos utiles.

II.2 Tipos de arreglos

Los arreglos no tienen que ser de una sola dimensión, ya que java apoya múltiples dimensiones. Estos arreglos se implementan como “arreglos de arreglos”. (Weitzenfeld, 2005)

La manipulacion de un arreglo se debe componer en tres etapas:

- Declaracion de una variable que permite trabajar con un arreglo
- Creacion del arreglo(asignacion memoria)
- Almacenacion y manipulacion de los elementos del arreglo. (Groussard, 2009)

Existen tres tipos de arreglos:

- *Arreglos unidimensionales* (registro o vector): Estructura de datos que permite almacenar un conjunto de datos de un mismo tipo. El acceso a un dato del arreglo se puede hacer por medio de un *índice.* Todo arreglo debe declararse y se le debe reservar memoria para poder ser utilizado.

Algunas características importantes de los arreglos son las siguientes:

- Variable *length* para conocer el tamaño del arreglo (array).
- Uso de corchetes e índices para acceder a los elementos. (Jaramillo, y otros, 2010)

$$Nombre=[X_0,x_1,x_2,x_3\ldots..,x_{i-1}]_i$$

Imagen II-4 Arreglo tipo vector

nombre

X_0,	,x_1	x_2	…..	,x_{i-1}	***i***
0	1	2	…..	i-1	

Imagen II-5 Arreglo tipo registro

Donde:

Nombre = *Nombre del vector*

X_1,X_2… = *Datos almacenados en el vector*

i = *Tamaño del vector*

- *Arreglos bidimensionales* (tabla o matriz): Es un vector de vectores, es decir, una estructura de datos que permite almacenar un conjunto de datos, todos del mismo tipo, en el cual el orden de cada uno de los datos es significativo y en el que se necita especificar dos *índices* para poder acceder a cada uno de ellos.

Nombre =

$X_{0,0}$	$X_{0,1}$	$X_{0,2}$	...	X_{0j-1}
$X_{1,0}$				
$X_{2,0}$				
.				
.				
.				
$X_{i-1,0}$				$X_{i-1,j-1}$

Imagen II-6 Arreglo tipo matriz

Columnas

nombre=

Filas

	0	1	2	...	j-1
0	$X_{0,0}$	$X_{0,1}$	$X_{0,2}$	...	X_{0j-1}
1	$X_{1,0}$				
2	$X_{2,0}$				
	.				
	.				
	.				
i-1	$X_{i-1,0}$				$X_{i-1,j-1}$

Imagen II-7 Arreglo tipo tabla

Donde:

Nombre = nombre de la matriz

$X_{0,0}$…..$X_{1,j-1}$ = Datos almacenados en la matriz

*i*j = Tamaño de la matriz*

- *Arreglos multidimensionales:* Son arreglos que tienen más de dos dimensiones. Por cada dimensión del arreglo se tiene que utilizar un índice para ubicar exactamente a un elemento en particular.

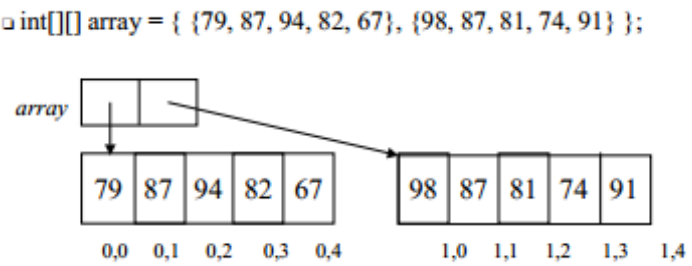


Imagen II-8 Arreglo multidimensional

Ventajas:

- Los datos están almacenados en una sola estructura de datos siendo más fácil el acceso a los mismos.
- Se utiliza un único nombre (nombre del arreglo) para referirse al conjunto de datos almacenados en el arreglo. Estos permite reducir el uso de variables y constantes.
- Los arreglos permiten almacenar datos de entrada y datos utilizados durante el procesamiento.
- Al igual que otras estructuras de datos (como las variables y constantes), los arreglos tienen un tiempo de vida, por lo que pueden ser considerados como globales o locales.

Desventajas:

- Un arreglo no puede almacenar datos de diferente tipo.
- Dependiendo del tipo de arreglo unidimensional o bidimensional, se necesitara uno o dos índices para acceder a los datos almacenados en el arreglo.

- Antes de crear un arreglo es necesario conocer el número exacto de datos que se necesitan almacenar. (Flores, 2013)

II.3 Cadena (clase String)

Una cadena es un conjunto de caracteres, numero y símbolos especiales almacenado en una variable de tipo texto o cadena.

La clase *String* es una clase que viene incorporada en el lenguaje de programación *Java*, incluida en el paquete *Java.lang*, la cual permite declarar y manipular variables de tipo texto o cadena (se crean objetos de la clase *String*). (Flores, 2013)

```
String saludo = “hola”;
```

Imagen II-9 Creación de cadena asignando una literal

```
String cadenA = new String ();
```

Imagen II-10 Creación de una cadena vacía

```
String ()String cadena;
String (char array []);
String (char array [], int desplazamiento, int cuenta);
```

Imagen II-11 Constructores

La clase *String* incluye métodos que permiten examinar los caracteres individuales de una cadena para compararlos, ubicarlos, extraerlos como subcadenas y crear copias de una cadena convirtiendo todos sus caracteres a letras mayúscula o minúscula. Las conversiones se realizan a través del método *toString()*, el cual es un método heredado y definido en la clase *Object.* (Flores, 2013)

Para ejecutaron método de instancia de la clase *String* se necesita colocar el nombre de la variable (tipo texto o cadena), el operador punto y el nombre del método instanciado a ejecutar. Para los métodos de clase se coloca la palabra *String*, el operador punto y el nombre del método de la clase a ejecutar.

```
Saludo = saludo + “que tal”;
```

Imagen II-12El operador + permite una concatenación

II.4 Método Split

El método *Split* divide una cadena en un *Array* de *String* con base en el carácter de partición. La cadena original se divide en cualquier posición que concuerde con una expresión regular especificada. El método Split devuelve un arreglo de cadenas que contiene las subcadenas que resultan de cada concordancia con la expresión regular. (Deitel, y otros, 2004)

```
String cadena =”zero one, two, three four, five+six”;

String result (cadena. Split (“\\,”);

For (int i=0; i<result. Legth; i++)

{

System.out. print (result [i]);

}
```

Imagen II-13 Uso de método Split separando la cadena con comas

```
Zero one
Two
Three four
Five+six
```

Imagen II-14 Resultado del código anterior

II.5 Archivos

Desde el punto de vista de más bajo nivel, se define a un archivo como un conjunto de bits almacenados en un dispositivo, accesible a través de un camino de acceso (*pathname*) que lo identifica, es decir, un conjunto de 0s y 1s que reside fuera de la memoria del ordenador, ya sea en un disco duro, usb, cd, entro otros.

Es diferente manipular números que *Strings*, aunque en el fondo ambos acaben siendo bits en memoria. Al manipular archivos se distinguen dos clases dependiendo del tipo de datos contenidos:

- Archivos de caracteres (texto)
- Archivos de bytes (binarios).

Un archivo de *texto* es aquel formado exclusivamente por caracteres y que, por tanto, puede crearse y visualizarse usando un editor. Las operaciones de lectura y escritura trabajaran con caracteres. Por ejemplo, los archivos con código java son archivos de texto.

En cambio un archivo binario ya no está formado por caracteres sino que los bytes que contiene pueden representar otras cosas como números, imágenes, sonido, etc.

Existen dos métodos básicos de acceso a la información contenida en un archivo:

- Secuencial
- Acceso directo

El modo secuencial la información del archivo es una secuencia de bytes (caracteres) de manera que para acceder al byte (*carácter*) *i-èsimo* se ha de haber accedido anteriormente a los i-1 anteriores.

El método de accesos directo permite acceder directamente a la información de byte *i-èsimo*. Un ejemplo de acceso directo son los vectores (*arrays*). (Gimeno, y otros, 2010/2011)

Un *archivo* es un conjunto de información relacionada lógicamente, como un texto y etiquetas de formato en un documento HTML o los bytes de un programa compilado. (Decker, y otros, 2001)

Los archivos se utilizan para almacenar datos de forma permanente y que estos persisten después de la finalización del programa.

La clase *File* y otras del paquete *java.io* permite gestionar el acceso archivos. La clase *File* permite el acceso, no al contenido del archivo, si no a sus propiedades generales como permisos, tamaño, localización, etc. Se suele utilizar para manejar el archivo como un todo sin usar los datos que contiene. (Duran, y otros, 2007)

Las clases FileInputStream y FileOutputStream permiten utilizar un archivo como fuente de bytes o como destino para escribirlos. Dichas clases son subclases de InputStream y OutputStream, por lo tanto comparten caracteristicas primitivas de lectura y escritura de sus clases madre.

Un FileInputStream es un InputStream por herencia, de modo que puede usarse como argumento real en el constructor de DataInputStream. Al wrapping un FileInputStream en un DataInputStream, es posible utilizar los metodos de esta ultima clase para la lectura de tipos primitivos en un archivo. (Decker, y otros, 2001)

II.6 BufferedReader/ BufferedWriter

El concepto de buffering queda muy bien explicado en el siguiente párrafo extraído del libro HeadFirst Java:

Si no hubiera buffers, sería como comprar sin un carrito: debería llevar los productos uno a uno hasta la caja. Los buffers dan un lugar en el que dejar temporalmente las cosas hasta que está lleno. Por ello has de hacer menos viajes cuando se usa carrito.

Cualquier operación que implique acceder a memoria externa es muy costosa, por lo que es interesante intentar reducir al máximo las operaciones de

lectura/escritura que se realizan sobre los ficheros, haciendo que cada operación lea o escriba muchos caracteres.

Además, eso también permite operaciones de más alto nivel, como la de leer una línea completa y devolverla en forma de cadena. (Gimeno, y otros, 2010/2011)

Esta subclase de la superclase raíz *Reader* decora los objetos *Reader* añadiéndoles la posibilidad de usar *buffers*, lo que mejora la eficiencia.

Esta clase incorpora un nuevo método: public string readLine() throws IOException, que permite leer líneas de texto desde un archivo de texto.

Siempre que se vaya a procesar el contenido de archivos de texto es conveniente usar esta clase. Para ver un ejemplo de su uso, se considera un archivo sagan.txt, ubicado en el directorio donde se ejecutara la clase LeerArchivo. Su contenido se muestra a continuación:

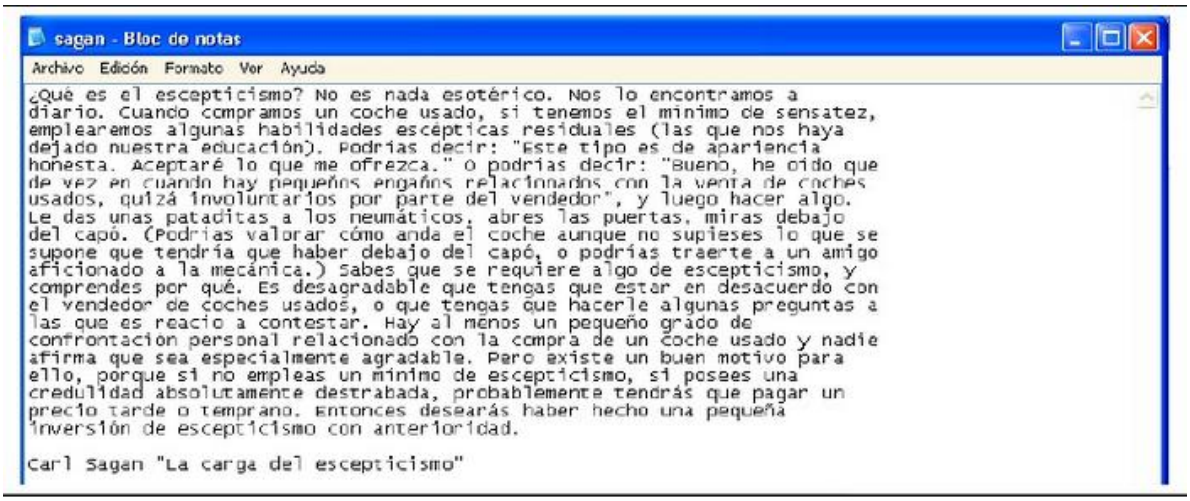


Imagen II-15 Archivo Sagan.txt

Esta subclase de la superclase raíz *Writer* decora los objetos *Writer* añadiéndoles la posibilidad de usar buffers, lo que mejora la eficiencia. Esta clase incorpora un nuevo método: *public String newLine() throws IOException,* que permite introducir saltos de línea.

Esta clase incorpora un nuevo método: *public String newLine() throws IOException,* que permite introducir saltos de línea. (Abiàn, 2012)

II.7 Lectura de un archivo

La subclase *java.io.InputStreamReader* de *Reader* modela un flujo de entrada basado en bytes como un flujo de caracteres, también de entrada. Los bytes se leen de un InputStream y se convierten en caracteres, de acuerdo con la codificación especificada en el constructor.

La única subclase de *InputStreamReader* es *FileReader*, que permite leer archivos de texto usando la codificación por defecto del sistema. Esta clase proporciona una interfaz de flujos de caracteres para leer archivos de texto usando la codificación por defecto. Uno de sus constructores es este:
Public FileReader(String nombreArchivo) throws FileNotFoundException
Este constructor crea un objeto *FileReader* que lee, usando la codificación por defecto del sistema, del archivo denotado por *nombreArchivo*. (Abiàn, 2012)

```
File file = new File(dir. archvo);
bufferedReader is = new Bufferedreader(new FileReader(file));
String s= is.readLine();
…
Is.close();
```

Imagen II-16 Código de lectura de una archivo

Se instancia un archivo *file,* especificando ubicación dentro del sistema *dir* y su nombre *archivo,* seguido se instancia un objeto de tipo *FileReader,* el cual se conecta al archivo *file* y en la misma línea se instancia el objeto *is* de tipo *BufferedReader(*permite conectarse a través de un búfer de lectura*).* La llamada *is.readLine()* realiza la lectura de una línea completa y la guarda en una variable *s* de tipo *String, a*l finalizar este se cierra mediante la llamada *is.close()* (Deitel, y otros, 2004)

II.8 Escritura de un archivo

La subclase *java.io.OutputStreamWriter* de *Writer* modela un flujo de salida basado en bytes como flujo de caracteres, también de salida. Proporciona métodos de escritura (*public void write(); public void writer(char[] array, int offset, int longitud); public void write (String cadena, int offset, int longitud)*) que permite escribir en un *OutputStream* los bytes que resultan de codificar los caracteres, de acuerdo con la codificación especificada en el constructor.

La única subclase de *OutputStreamWriter* es *FileWriter*, que permite escribir en archivos de texto mediante la codificación por defecto del sistema. Esta clase proporciona una interfaz de flujos de caracteres para escribir en archivos de texto mediante la codificación por defecto. Uno de sus constructores tiene esta forma:

Public FileWriter (String nombreArchivo, boolean anyadir) throws IOException

Este constructor crea un objeto *FileWriter* que escribe en el archivo al que se refiere *nombreArchivo* usando la codificación por defecto del sistema. El parámetro añadir indica si los datos deben añadirse a un archivo ya existente (*true*) o si debe eliminarse el archivo que ya existía (*false*). (Abiàn, 2012)

```
BufferedWriter os = new BufferedWriter (new FileWriter(file));
Os.write(s);
…
Os.close();
```

Imagen II-17 Código de escritura de un archivo

Se instancia un archivo *file*, se hace otra instancia a un objeto de tipo *FileWriter*, el cual se conecta al archivo, después en la misma línea se instancia el objeto *os* de tipo *BufferedWriter* el cual permite la conexión a través de un búfer de escritura. La llamada *os.write(s)* hace una escritura de una cadena referida por la variable *s* de tipo *String,* para finalizar la escritura se llama a *os.close().* (Deitel, y otros, 2004)

II.9 Excepciones

Las excepciones son un mecanismo que permite a los métodos indicar que algo "extraño" ha sucedido que impide su correcto funcionamiento, de manera que quien lo ha invocado puede detectar la situación errónea. En este caso el método lanza (*throw*) una *excepción*. Cuando esto sucede, en vez de seguir con la ejecución normal de instrucciones, se busca hacia atrás en la secuencia de *llamadas* si hay alguna que quiera *atraparla* (*catch*). Si ninguna de las *llamadas* decide *atraparla*, el programa acaba su ejecución y se informa al usuario del error que se ha producido la excepción y que nadie ha tratado.

Muchas de las excepciones que existen en Java, por ejemplo, dividir por 0, son *excepciones en tiempo de ejecución* (*runtime exceptions*) y no obligan a que el programador las trate explícitamente.

En java, existe otro tipo de excepciones, las denominadas *excepciones comprobadas* (*checked exceptions*), que obligan al programador que dentro del código de un método invoca una instrucción que puede lanzarla a:

- Atrapar dicha excepción (colocando dicha instrucción en un bloque try-catch)
- Declarar en la cabecera del método que dicho método puede lanzar esa excepción (usando una declaración throws).

```
Try{
Código que abre y trata el fichero
}catch (IOExcepction ex){
Código que trata el error
}
```

Imagen II-18 Uso de Try-Catch

La idea intuitiva de esta construcción es: *intenta* (*try*) ejecutar esas instrucciones y, en caso de reproducirse un error en el tratamiento de los archivos (se ha lanzado una *IOException*), *atrapa* (*catch*) ese error y ejecuta el código de corrección. (Gimeno, y otros, 2010/2011)

La palabra clave *try* se usa antes de un bloque de instrucciones que podrá generar una excepción, dicho bloque va seguido de una o más clausuras *catch.* Si no se generan excepciones durante la ejecución de un bloque *try*, se omite la ejecución de las clausulas *catch* que le correspondan, cuando ocurre una excepción, el control pasa inmediatamente del bloque *try* a la cláusula *catch* apropiada. Las instrucciones del bloque *catch* se ejecutan en siguiente termino, y después el control se transfiere a la primera instrucción que sigue al bloque *try.* (Decker, y otros, 2001

```
Try{
BufferedReader is = new Buffered Reader (new FileReader(file));
leerArchivo(is);
is.close();
}
Catch (IOException e){
System.out.print("Error Lectura Registro:"+ e);
System. Exit(1);
}
Finally {
System.out.print ("Lectura Archivo: "+ file);
}
```

Imagen II-19 Código de excepción

El bloque *try* abre la conexión al archivo de lectura *file*, llama al método de lectura, *leerArchivo* y cierra posteriormente la conexión al archivo. El bloque *catch* define el manejo de la excepción, *IOException,* la cual se pasa como argumento único dentro del bloque. (Weitzenfeld, 2005)

III. Capítulo III Desarrollo de la metodología

El algoritmo se desarrolló en el lenguaje de programación Java dentro de la plataforma *NetBeans*, el cual ofrece el uso de un entorno gráfico.

Se programaron tres clases para otorgarle la funcionalidad requerida al proyecto, la Clase inicial [main()] llamada *Asignación*, la clase para el manejo del entorno gráfico y control de eventos llamada *CargaEmpleados* y una última clase para el apoyo de funciones u operaciones específicas llamada *Utilidades*.

La clase principal Asignación contiene las definiciones primordiales de la aplicación, como lo son las rutas de los archivos de uso y la definición de los Arreglos a utilizar. Además de mandar a realizar las operaciones de llenado inicial de los arreglos y ejecución del entorno gráfico.

La clase llamada *CargaEmpleados* es la parte gráfica de la aplicación, además de ser la que contiene las operaciones principales del proyecto como lo son: definición de empleados, definición de tareas, postulación de empleados a partir de una tarea y asignación de una tarea a un empleado. Se hace uso del constructor de dicha clase para inicializarla con los arreglos a usar (Empleados, Tareas) y con las rutas de los archivos para poder hacer uso de ellos.

```
String [] Empleados;
String [] Tareas;

String ArcEmp="";
String ArcTar2="";
String ArcTar="";
```

Imagen IIII-1 Arreglos dentro del constructor

La última de las clases denominada *Utilidades* contiene funcionalidades específicas para el mejor manejo del programa, dichas funcionalidades están definidas en los métodos de la clase en los cuales se realizan las operaciones de lectura de un archivo asignando su contenido a un arreglo (*LeerArchivo*), reescribir un archivo a partir de un arreglo (*ReescribirFile*), guardar una línea en un archivo

(*GuardaLinea*), conversión de *string* a entero (*ConvierteEn*)y control de *logs* de nuestro programa (*GuardaLog*).

III.1 Archivos para el manejo de datos de empleados y tareas

Al iniciar el programa el código carga los archivos E*mpleados.txt* y *Tareas.txt* a los arreglos de String *Empleados* y *Tareas* respectivamente, esto instanciando la clase *Utilidades* y haciendo referencia al método *LeerArchivo**,*** posterior al llenado de los arreglos se invoca la clase C*argaEmpleado**s*** la cual es la parte gráfica del código y dentro de la cual se realizan las operaciones principales. Para la inicialización del objeto *CargaEmpleados* se envía como parámetros los Arreglos (*Empleados y Tareas*) y las definiciones de las rutas de los archivos de uso.

```
Utilidades uti=new Utilidades();
String [] Empleados=uti.LeerArchivo(ArcEmp);
      String [] Tareas=uti.LeerArchivo(ArcTar);
new CargaEmpleados(Empleados,Tareas,ArcEmp,ArcTar,ArcTar2).setVisible(true);
```

Imagen IIII-2 Codigo que carga los archivos e invoca al Frame

El arreglo de Strings *Empleados* se encarga de almacenar la información de los empleados para la manipulación de los mismos dentro del programa. El diseño del arreglo es de la siguiente manera: (tabla III.1)

Posición	Perfil del Empleado : Horas del Empleado : Nombre del empleado
0	**00011110:40:Julian Pérez López**
1	**11111111:25:Lorenzo Rivas Quiles**
2	**11100011:15:Angelica Suarez Roldan**
.	
.	
N	**Perfil N:Horas N:Nombre N**

Tabla III-1 Diseño del arreglo Empleados

Al momento de ocupar *n* elemento del arreglo, se hace uso del método *Split* para obtener los elementos de la información del empleado. (Tabla III.2))

Posición	Elemento
0	Perfil del Empleado
1	Horas del Empleado
2	Nombre del empleado

Tabla III-2 Diseño de los elementos de empleado

El arreglo de Strings *Tareas* se encarga de almacenar la información de las Tareas para la manipulación de las mismas dentro del programa. El diseño del arreglo es de la siguiente manera: (tabla III.3)

Posición	Perfil de la Tarea : Horas de la Tarea : Nombre de la Tarea
0	**11100000:15:sistemas**
1	**11100000:15:sistemas**
2	**00001100:40:Default**
.	
.	
N	**Perfil N:Horas N:Nombre**

Tabla III-3 Diseño del arreglo Tareas

Al momento de ocupar n elemento del arreglo, se hace uso del método *Split* para obtener los elementos de la información de la Tarea.

Posición	Elemento
0	Perfil de la Tarea
1	Horas de la Tarea
2	Nombre de la Tarea

Tabla III-4 Diseño de los elementos de la Tarea

III.2 Métodos

Una vez invocada la clase *CargaEmpleados* existen cuatro métodos primordiales para el programa, los cuales son:

- Método para dar de alta los empleados dentro del *Arreglo Empleados* y archivo de uso.
- Método para la definición de las tareas dentro del *Arreglo Tareas* y del archivo de uso.
- Método para buscar a partir de una tarea los empleados que cumplen con el perfil de la misma.
- Método para asignar la tarea al empleado elegido para realizarla.

Al dar de alta a los empleados la parte esencial son sus perfiles para esto se crean 8 variables de tipo *String* llamadas *p1,p2,p3,p4,p5,p6,p7,p8* inicializadas en “0”, las cuales si son seleccionadas su valor cambia a “1”, estos elementos son concatenados para formar el perfil del empleado, adicional a esto se obtiene las horas y nombre del empleado, esta información es almacenada dentro del *Archivo Empledos.txt* con ayuda del método *GuardaLinea* de la clase *Utilidades.* Cabe destacar que si no es asignado un nombre u horas el programa asigna sus defaults.

Posterior al almacenamiento en el archivo, el programa carga nuevamente su contenido en el arreglo Empleados.

```
p1=p2=p3=p4=p5=p6=p7=p8="0";
    if(jCheckBox9.isSelected())
    p1="1";
    if(jCheckBox10.isSelected())
    p2="1";
    if(jCheckBox11.isSelected())
    p3="1";
    if(jCheckBox12.isSelected())
    p4="1";
    if(jCheckBox13.isSelected())
    p5="1";
    if(jCheckBox14.isSelected())
    p6="1";
    if(jCheckBox15.isSelected())
    p7="1";
    if(jCheckBox16.isSelected())
    p8="1";

Empleado=p1+p2+p3+p4+p5+p6+p7+p8+"";
```

Imagen IIII-3 Código que crea el perfil del empleado

```
11111000
01111100
00011110
11111111
11100011
```

Imagen III-4 Perfil creado de la selección antes mencionada

```
uti.GuardaLinea(ArcEmp,Empleado+":"+horas+":"+nombre);
 Empleados=uti.LeerArchivo(ArcEmp);
```

Imagen IIII-5 Código que guarda los datos del empelado y carga nuevamente el archivo

Una vez registrados los empleados, la siguiente operación a realizar es dar de alta tareas, para almacenar el perfil de la tarea y así asignarla a un empleado, este perfil se guarda en la las 8 variables de tipo String llamadas *p1,…….,p8* inicializadas en "*0"*, las cuales si son seleccionadas su valor cambia a *"1"* de no ser así permanecen en "*0"*, dichos elementos son concatenados para formar el

perfil de la tarea, adicional a esto se obtiene las horas y nombre de la tarea, esta información es almacenada dentro del *Archivo Tareas.txt* con ayuda del método *GuardaLinea* de la clase *Utilidades*. Cabe destacar que de no ser establecido un nombre u horas el programa asigna sus defaults.

```
p1=p2=p3=p4=p5=p6=p7=p8="0";
if(jCheckBox1.isSelected())
p1="1";
if(jCheckBox2.isSelected())
p2="1";
if(jCheckBox3.isSelected())
p3="1";
if(jCheckBox4.isSelected())
p4="1";
if(jCheckBox5.isSelected())
p5="1";
if(jCheckBox6.isSelected())
p6="1";
if(jCheckBox7.isSelected())
p7="1";
if(jCheckBox8.isSelected())
p8="1";
Tarea=p1+p2+p3+p4+p5+p6+p7+p8+"";
```

Imagen IIII-5 Código que crea el perfil de la tarea

```
11100000
00001100
00001100
01100000
```

Imagen III-6 Perfil creado de la selección antes mencionada

Una vez obtenido el perfil de la tarea y la duración de la misma en horas, se invoca el método *Busca* que recibe como parámetros el perfil de la tarea y las horas, realizando aquí el análisis para la selección de los posibles perfiles de empleados que puedan realizar dicha tarea.

El perfil de la tarea es asignado al arreglo *Tarea*, se tiene un ciclo el cual corre según tantos empleados se tienen en el arreglo *Empleados*, los perfiles de los empleados son almacenados en el arreglo *Busca* para así comparar el perfil de la tarea contra el perfil del empleado para determinar que empleado cumple con las

habilidades requeridas. Si el empleado cumple con el perfil, la última comparación es que el empleado tenga las horas suficientes para cubrir la tarea, en caso afirmativo el Empleado es postulado para llevar a cabo la tarea.

```
char [] Tarea=new char[8];
      Cadena.getChars(0, 8, Tarea,0);
    System.out.println("Total de arreglo Em: "+Empleados.length);
      for(int i=0;i<Empleados.length;i++){ //numero de emplados
       boolean enc=false;
           char [] Buscar=new char[8];
           String [] PEmpleados=Empleados[i].split(":");
  PEmpleados[0].getChars(0, 8, Buscar, 0);
       System.out.println("I: "+i+"PerfilE: "+PEmpleados[0]+ "PErfilT: "+Cadena);
          for(int j=0;j<8;j++){
             if(Tarea[j]=='1'){
                if(Buscar[j]=='1')
                   enc=true;
                else{
                   enc=false;
                   break;
        }
     }
}
```

Imagen III-7 código que realiza la selección de posibles empleados

Una vez que el método *Busca* realiza la selección de los posibles empleados, se invoca el método *Casar* el cual recibe como parámetros la posición del empleado, el perfil de la tarea, las horas y el nombre de la tarea, realizando la asignación de la tarea a la primer posición de los empleados postulados, obteniendo así la pareja *Tarea-Empleado* la cual se almacena en el arreglo *Empleados*, añadiendo a dicho arreglo el perfil del empleado, las horas libres del empleado y el nombre del empleado, todo esto sobrescribiendo el archivo *Empleados.txt* con ayuda del método *ReescribirFile* de la clase *Utilidades.*

```
int EmplOk=uti.ConvierteEn(EmplPost[0]);
int horaTa=uti.ConvierteEn(Horas);
String PEmpleado []=Empleados[EmplOk].split(":");
int horaEm=uti.ConvierteEn(PEmpleado[1]);
horaEm=horaEm-horaTa;
```

Imagen III-1 Código donde se crea la pareja tarea-empleado

```
Empleados[EmplOk]=PEmpleado[0]+":"+horaEm+":"+PEmpleado[2];
uti.ReescribirFile(ArcEmp,Empleados);
```

Imagen III-8 Código que permite la actualización del archivo

Ya obtenida la asignación, el String *DetTarea* guarda el nombre de la tarea, el perfil de la tarea, las horas de la tarea, la posición o Id del empleado, perfil del empleado y horas restantes del empleado, esta información se almacena dentro del archivo *TareasAsignadas.txt* con ayuda del método *GuardaLinea* y cargando nuevamente el archivo con el método *LeerArchivo,* ambos métodos pertenecientes a la clase *Utilidades.*

```
String DetTarea=NomTar+":"+Tarea+":"+Horas+":"+EmplOk+":"+Empleados[EmplOk];
uti.GuardaLinea(ArcTar, DetTarea);
Tareas=uti.LeerArchivo(ArcTar);
```

Imagen III-9 Código que almacena y carga el archivo Tareas Asignadas

A partir de la información almacenada se crea una tabla para mostrar un informe detallado de las asignaciones realizadas, para realizar esta tabla se utiliza el modelo (*TableModel*) del objeto "*JTable*." El cual tendrá un objeto que contendrá los siguientes campos: nombre de la tarea, perfil de la tarea, horas de la tarea, el Id (posición) del empleado, perfil empleado, horas restantes del empleado.

```
DefaultTableModel temp = (DefaultTableModel) jTable1.getModel();
Object nuevo[]= {NomTar,Tarea,
Horas,EmplOk,PEmpleado[0],horaEm+"",PEmpleado[2]};
temp.addRow(nuevo);
```

Imagen III-10 Código que imprime en tabla

Cabe mencionar que la información de la tarea es almacenada dentro del archivo *Tareas.txt* utilizando el método *GuardaLinea* de la clase Utilidades.

```
uti.GuardaLinea(ArcTar2,Tarea+":"+Horas+":"+NomTar);
```

Imagen III-11 Código que almacena dentro del archivo

IV. Resultados

El algoritmo “asignación de tareas” permite optimizar la distribución de tareas a cada empleado de una manera automatizada. En la siguiente ventana se muestra un menú de opciones, “*cargar nuevo empelado*”, “*Cargar nueva tarea*” y finalmente “*Informe*”**.**

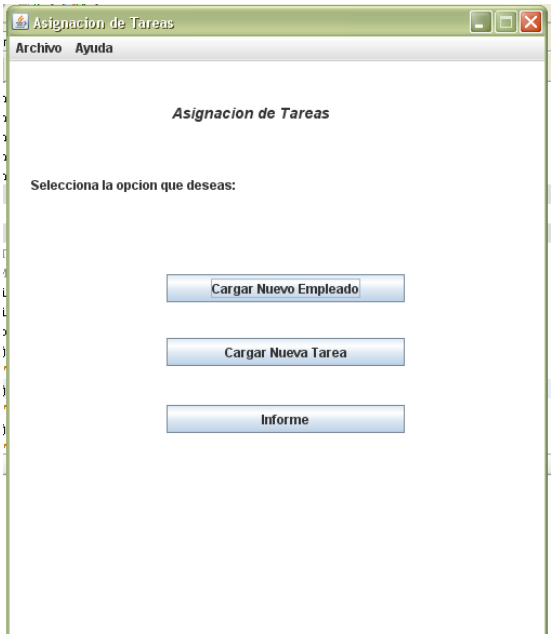


Imagen IV-1 Ventana de menú

Donde el botón *Carga nuevo empleado*, permite almacenar el nombre del empleado, las horas por semana en las que labora el perfil del empleado, esto por medio de una selección de perfiles, las cuales describan el entorno donde se desenvuelve más el empleado. El botón *Cargar* guarda los datos del empleado dentro del archivo *Empleado.txt*, con el botón *Regresar* volvemos al menú principal. (Ver imagen. IV-2)

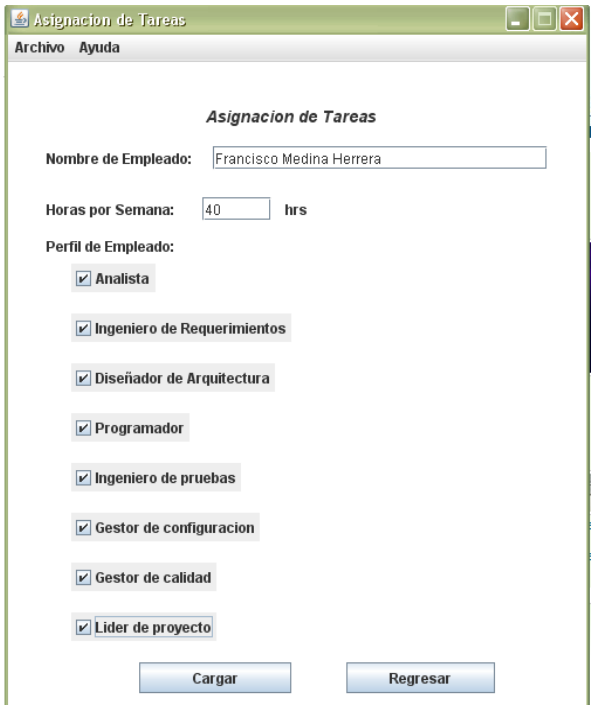


Imagen IV-2 Ventana de captura de empleados

Regresando al menú principal podemos ahora seleccionar la opción *Cargar Nueva Tarea,* con esta opción registraremos el nombre de la Tarea, las horas que necesita la tarea para su ejecución y finalmente la selección de los perfiles necesarios con los que debe contar el empleado para realizarla, al presionar el botón *Cargar* se guarda la tarea dentro de memoria y se trabaja con ella y con el botón *Regresar* podemos de nuevo acceder al Menú principal. (Ver imagen IV-3)

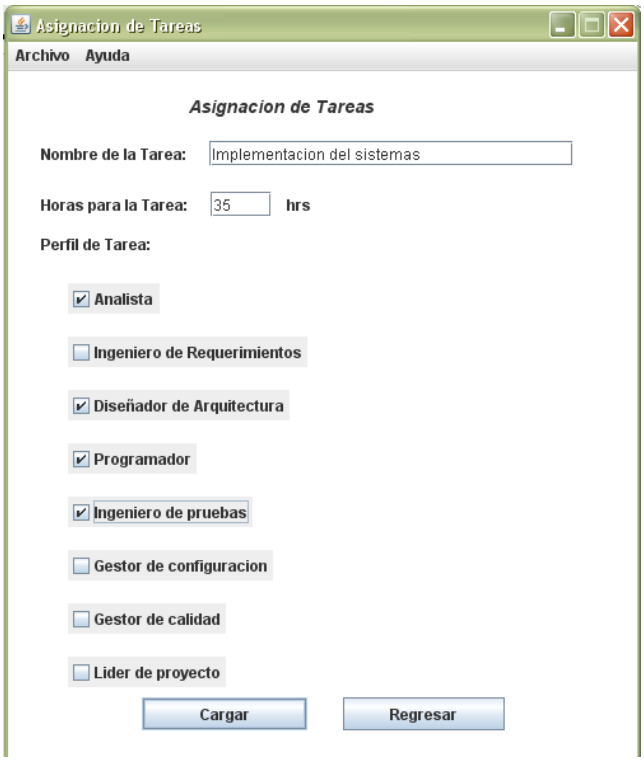


Imagen IV-3 Ventana de captura de tareas

Y finalmente el boton de *Informe*, muestra un reporte detallado de la asignacion de empleados para cada tarea, dentro de una tabla la cual cuenta con 7 columnas encabezadas con (ver imagen IV-4):

NomTarea: Indica cual es el nombre de la tarea registrada.

PerfilTarea: Muestra cuales son las caracteristicas solicitadas por la misma

HoraTarea: Son las horas que dicha tarea necesita para su realizacion.

IdEmpleado: El numero identificar del empleado asignado a la tarea

PerfilEmpleado: Muestra las caracteristicas del empleado, las cuales deben ser iguales o mayores a las solicitadas por la tarea.

HorasRes: son las horas restantes del empleado, con lo cual permite asignar mas de una tarea a dicho trabajador.

NomEmp: Indica el nombre del trabajador al cual se le asigno dicha(s) tarea(s).

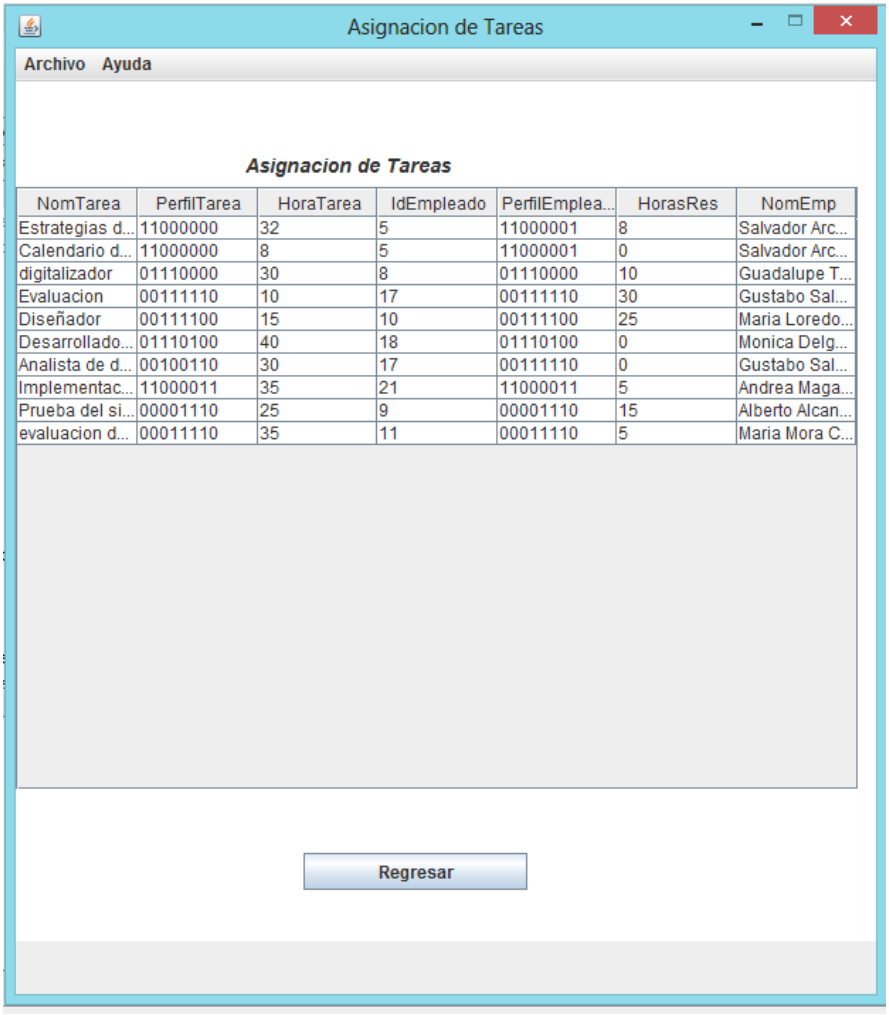


Imagen IV-4 Tabla de informe final

En la imagen IV-4 se muestra un ejemplo del informe de asignacion para el cual se insertaron 10 tareas a desempeñar dentro de una empresa de software, cabe mencionar que a algunos empleados se les fue asignada mas de una tarea, pues el fin del algoritmo es optimizar el tiempo de los empleados.

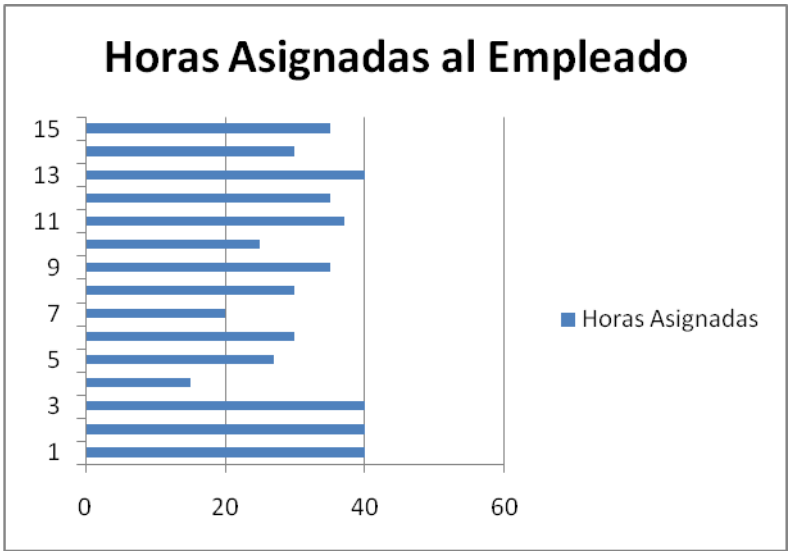
Cabe mencionar que el líder del proyecto es el que usa y toma la decisión en función de los resultados obtenidos.

Gráficas

La tabla 5 muestra los resultados obtenidos en cuanto a Horas Asignadas por Tarea a cada empleado, los resultados son mostrados dentro de una gràfica de barras.

Horas Asignadas al Empleado	
ID_Empleado	Horas Asignadas
0	40
1	40
2	40
3	15
3	27
4	30
5	20
6	30
7	35
8	25
3	37
9	35
0	40
18	30
10	35

Tabla 5Horas tareas asignadas a empleados



Grafica 1 Horas de tareas asignadas a empleados

La tabla 6 muestra las horas libre con las que cuenta cada empleado, permitiendo asi la asignacion de otra tarea, los resultados son mostrados dentro de una gráfica.

Horas Restantes del Empelado	
ID_Empleado	Horas Restantes
0	0
1	0
2	0
3	25
3	13
4	10
5	20
6	10
7	5
8	15
3	3
9	5
0	0
18	10
10	5

Tabla 6 Horas restantes del empleado

Grafica 2 Horas restantes del empleado

La tabla 7 muestra la cantidad de tareas asignadas a cada empleado, los resultados son graficados

Cantidad de tareas Asignadas por Empleado	
ID_Empleado	Cantidad Tareas
0	2
1	1
2	1
3	3
4	1
5	1
6	1
7	1
8	1
9	2
18	1
10	1

Tabla 7 Numero de tareas por empleado

Grafica 3 Cantidad de tareas asignadas/ empleados

Conclusiones

El método utilizado resolvió el problema planteado. El uso de archivos almacenó los datos y permitió la manipulación sin contratiempos. El uso de archivos permite que la aplicación corra en una computadora del usuario sin la necesidad de la conexión a un servidor de bases de datos. No se probó pero puede funcionar hasta en un dispositivo móvil.

El problema planteado se resolvió hasta donde se definió. Permite al líder o gerente de proyectos asignar las tareas. Permite a su vez el monitoreo de horas laborables semanales de cada desarrollador y también las horas libres. Sin embargo quien toma la decisión final de la asignación puede ser el propio líder o gerente.

Los objetivos se cumplieron porque se culminó el algoritmo y si logra asignar las tareas a los desarrolladores respetando el perfil de la tarea y del desarrollador. Se creó el almacén de datos; en este caso archivos de datos, que funcionan de forma correcta.

El objetivo no contempla una comparación con otra norma. En el estado del arte no se encontró otro algoritmo que pueda compararse.

La aportación de esta investigación se resume a un algoritmo basado en archivos, creado en Java para la asignación de tareas a desarrolladores de forma automática, como apoyo a los líderes o gerentes de proyectos de software.

Trabajos futuros

Después de culminar el trabajo de investigación se detectó que existen al menos tres trabajos futuros para complementar un sistema de asignación automática de tareas a desarrolladores.

El primero es un algoritmo que registra las tareas terminadas y libera al trabajador de dichas tareas para asignar nuevas tareas y usar la herramienta durante todo el periodo de desarrollo.

El segundo es un algoritmo que revisa los tiempos programados y los tiempos reales con el fin de evaluar la productividad de los programadores. O bien para que el líder o gerente de proyectos logre afinar su sentido de estimación para cada tarea.

El tercero consiste en otro método que incluya un valor de escala del perfil en lugar de 0´s y 1´s ya que tanto la tarea como el desarrollador podrían tener cierto nivel del perfil y ni es 0 ni es 1.

Trabajos citados

Bell, Douglas y Parr, Mike. 2003. *Java para estudiantes.* Mexico : Pearson Educaciòn, 2003.

Beuchot, Mauricio. 1999. *Heurística y Hemenéutica.* México : Universidad Nacional Autónoma de México, 1999.

Cairò, Osvaldo y Guardati, Silvia. 2002. *Estructuras de datos.* Mèxico : McGraw-Hill, 2002.

Calancha, Niefar Abgar. 2012. Breve aproximación a la técnica del Árbol de Decisiones. [En línea] 2012. http://niefcz.files.wordpress.com/2011/07/breve-aproximacion-a-la-tecnica-de-arbol-de-decisiones.pdf.

Caldas, Ma. Eugenia, y otros. 2010. *Formación y orientación laboral.* Madrid : Editex, 2010.

Deitel, Harvey M. y Deitel, Paul J. 2004. *Còmo programar en Java.* Mexico : Pearson Educacion Mèxico, 2004.

E. Kendall, Julie y E. Kendall, Kenneth. 2005. *Analisis y diseño de sistemas.* México : Pearson educación, 2005.

Fernández, Vicenç. 2006. *Desarrollo de sistemas deinformacion: una metodología basada en el modelado.* Barcelona : UPC, 2006.

Fuller, David. 2003. Apuntes de Taller de Ingeniería de Software . [En línea] 2003. http://www.ganimides.ucm.cl/ygomez/descargas/Sist_inf2/apuntes/2009/Roles_desarrollo_software.pdf.

Galindo González, Carlos. monografias.com. [En línea] http://www.monografias.com/trabajos75/busqueda-heuristica/busqueda-heuristica.shtml.

Hayes, Nicky. 2003. *Dirección de equipos de trabajo, una estrategia para el éxito.* Madrid, españa : Paraninfo, 2003.

Hofstadt, Carlos J. y Gómez, José Ma. 2006. *Competencias y habilidades profesionales para universitarios.* Madrid : Diaz de Santos, 2006.

JOM, ANDRÉS y MR, ADAM. 1999. XVII CONGRESO NACIONAL DE INGENIERIA DE PROYECTOS; INFORMÁTICA, TELECOMUNICACIONES Y MULTIMEDIA. *ASIGNACIÓN DE PERSONAS A TAREAS EN PROYECTOS*

INFORMÁTICOS: ASPECTOS HUMANOS. [En línea] MURCIA, 1999. [Citado el: 14 de 10 de 2012.] http://www.unizar.es/aeipro/finder/INFORMATICA,TELECOMUNICACIONES/FD01.htm.

Mora, Alejandro y Moreno, Irene. 2007. Desarrollo de un sistema de asignacion de tareas a trabajadores polivalentes. [En línea] 27 de 03 de 2007. [Citado el: 24 de 08 de 2012.] http://www.upcommons.upc.edu/handle/2099.1/4044.

Muchinsky, Paul M. 2002. *Psicología aplicada al trabajo.* s.l. : Thomson Learnig, 2002.

Optimizacion en la asignacion de tareas en un sistema de guarderia forestal. **Pradenas, Lorena y Azocar, Leandro. 2005.** 2, Chile : Bosque, 2005, Vol. 26.

Palomo, María Teresa. 2010. *Liderazgo y motivación de equipos de trabajo.* España : ESIC, 2010.

Porret, Miquel. 2008. *Recursos Humanos, dirigir y gestionar personas en las organizaciones.* España : ESIC EDITORIAL, 2008.

Ramírez, César. 2005. *Seguridad industrial, un enfoque integral.* Mexico : Limusa, 2005.

RAMÍREZ, RAMÓN. 2005. *Gestiòn del desarrollo de sistemas de telecomunicacion e informaticos.* MADRID, ESPAÑA : THOMSON, PARANINFO S.A., 2005.

Ramos Cruz, Carlos. scribd, Heurística. [En línea] http://es.scribd.com/doc/36818832/Heuristica.

Rich, Elaine y Knight, Kevin. 1994. *Inteligencia Artificial.* s.l. : Mc. Graw Hill, 1994.

RODRIGUEZ, NURIA y MARTÍNEZ, WILLIAM. 2006. *PLANIFICACIÓN Y EVALUACIÓN DE PROYECTOS INFORMÁTICOS.* SAN JOSÉ : EUNED, 2006.

S.L., Publicaciones Vértice. 2007. *Coordinación de equipos de trabajo.* Malaga : Publicaciones Vértice, 2007.

Sommerville, Ian. 2011. *Ingeniería de software.* México : Pearson Educación, 2011.

Topchik, Gary S. 2008. *Gerente por primera vez: Como desarrollar a tu equipo.* s.l. : Grupo Nelson, 2008.

Tuya, Javier, Ramos, Isabel y Dolado, Javier. 2007. *TÉCNICAS CUANTITATIVAS PARA LA GESTIÓN EN LA INGENIERIÁ DEL SOFTWARE.* LA CORUÑA, ESPAÑA : NETBIBLO, S.L., 2007.

Urcola, Juan Luis. 2010. *Dirigir personas: fondo y formas.* Madrid : ESIC, 2010.

VIZCAINO, PAULA ANDREA. 2008. APLICACIÓN DE TÉCNICAS DE INDUCCIÓN DE ÁRBOLES DE DECISIÓN A PROBLEMAS DE CLASIFICACIÓN MEDIANTE EL USO DE WEKA (WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS). [En línea] 2008. http://es.scribd.com/doc/69923922/8/CONSTRUCCION-DE-ARBOLES-DE-DECISION.

Weitzenfeld, Alfredo. 2005. *Ingenierìa de software orientada a objetos con UML Java e Internet.* s.l. : Thomson, 2005.

WertherJr., William y Davis, Keith. 1995. *Administración de personal y recursos humanos.* Mexico : Mc Graw-Hill, 1995.

Texcoco, México a 3 de Septiembre de 2013

M. EN C. JUAN MANUEL MUÑOZ ARAUJO
SUBDIRECTOR ACADÉMICO DEL
CENTRO UNIVERSITARIO UAEM TEXCOCO.
PRESENTE:

COPIA

AT'N M .EN P.P. ANTONIO INOUE CERVANTES
RESPONSABLE DEL DEPARTAMENTO DE TITULACIÓN.

Con base en las revisiones efectuadas al trabajo escrito titulado **"Algoritmo para la Asignación Automática de Tareas de los Procesos en las Fábricas de Software"** que para obtener el título de **Licenciado en Ingeniería en Computación** presenta la sustentante **Méndez Vázquez Refugio Marisol**, con número de cuenta **0114654** respectivamente, se concluye que cumple con los requisitos teórico -metodológicos por lo que se otorga el voto aprobatorio para su sustentación, pudiendo continuar con la etapa de digitalización del trabajo escrito.

ATENTAMENTE

M. en C. José Jair Vázquez Palma
Revisor

M. en C. A Gerardo Rafael Valencia Valencia
Revisor

M. en C. A José Sergio Ruiz Castilla
Director

TITULACION
RECIBIDO
Por
Texcoco, Méx. a 3 de 9 del 2013

C.C.P Sustentante Méndez Vázquez Refugio Marisol
C.C.P. Director M en C.A José Sergio Ruiz Castilla
C.C.P Titulación M .en P.P. Antonio Inoue Cervantes