Algoritmo para la asignación automática de tareas a desarrolladores de software utilizando vectores y covarianza

Sergio Ruiz[1], Marisol Méndez[1], Jair Cervantes[1], Joel Ayala[1]

[1]Universidad Autónoma del Estado de México, Centro Universitario UAEM Texcoco, Av. Jardín Zumpango s/n, El Tejocote, Texcoco, Estado de México.

{jsergioruizc@gmail.com, cyber-driva@hotmail.com, chazarra17@gmail.com, joelayala2001@yahoo.com.mx}

Abstract. En la industria del software se desarrollan proyectos que se dividen en módulos luego en actividades y después en tareas. El líder del proyecto debe definir el perfil de las tareas para asignarlas a los desarrolladores de acuerdo a los roles que poseen. La actividad puede ser muy titánica cuando son cientos o miles de tareas. Se propone un algoritmo que tiene la función de asignar automáticamente las tareas a los desarrolladores de acuerdo al perfil de la tarea y el perfil del desarrollador. El perfil se compone de conocimientos y habilidades que requiere cada tarea y que el desarrollador debe poseer en alguno de sus roles. Usando un vector con valores correspondientes a los conocimientos y las habilidades para el perfil de la tarea y otro vector para el desarrollador; el algoritmo asigna tareas sin rebasar el máximo de horas de carga de trabajo para cada desarrollador.

Palabras clave. Perfil de tareas, perfil del desarrollador, roles, asignación automática, distribución de tareas, vectores, covarianza.

1. Introducción

1.1 Perfil de tareas y desarrolladores

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 los largo del proyecto.

La asignación de tareas es necesaria en los planes detallados de cada módulo [5,6]. 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 artículo se escribe acerca de los equipos de trabajo, con el fin de conocer cómo se forma, sus roles, comunicación y formas de trabajo. Se habla 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 tareas 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 [6].

Se propone un algoritmo para la solución de la asignación automatizada de tareas considerando un perfil para cada tarea así como un perfil para cada desarrollador.

1.2 Antecedentes

El trabajo Sistema de asignación de tareas a trabajadores polivalentes que presenta Mora y Moreno, consiste en el desarrollo de un sistema de asignación de tareas a trabajadores polivalentes. Esto es, una herramienta que 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.

A partir de esto se generó un contador que se encarga de llevar el conteo del 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. Se crea un formulario llamado *general,* el cual consta de 2 subformularios llamados *planing 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.

Usando Visual Basic Se pretendió solucionar el problema del uso masivo de variables, se utilizaron contadores que almacenen el número de empleados que hay para cada tarea, en cada horario establecido según el día. Primera parte se rellenaron 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).* 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.

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 *tablaprincipal (Fig 1).*

Table 2. Tabla princial [1].

Código trabajador	Tarea	Principal
1	1	N
1	2	S
1	3	N
1	4	S
1	5	N
2	1	S
2	2	N
2	3	N
2	4	S
2	5	N
3	1	N

Se utilizó un algoritmo que asigna tareas de forma aleatoria a cada trabajador. Se implementa un código que reparte tareas de forma aleatoria, mediante la función *random = Int((12 - 1 + 1) * Rnd + 1)*, a cada uno de los empleados para cada una de las horas. Una vez asignada la tarea se cuenta el número 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 (*If (rst3("Tarea") = t) And (rst3("Franja") = f) Then If (rst3("Requisito") > c)*) se muestra por pantalla un mensaje para que el usuario arregle esto mediante la forma manual [1].

1.3 Propósito de la investigación

El algoritmo propuesto busca ayudar en la toma de decisiones al líder de proyectos en la planeación, distribución y asignación de tareas a los desarrolladores en proyectos de software.

El líder del equipo de trabajo debe conocer todas la habilidades de todos los trabajadores, sin embargo pueden ser muchas se torna imposible tenerlas de memoria. Además los trabajadores se marchan y llegan nuevos.

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. 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?*

El objetivo se enfoca al desarrollo de 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.

Primero se caracterizaron las tareas para tener definido el perfil, se creó una base de datos de tareas, y desarrolladores y se obtuvo una salida con las asignaciones a los desarrolladores.

1.4 Ventajas posibles de la asignación automática

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 [7]. 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 generar *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 [7].

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.

2. Desarrollo del algoritmo

Utilizando la covarianza que es un valor que indica el grado de variación conjunta entre dos variables aleatorias. Es el dato básico para determinar si existe dependencia entre dos variables.

El estimador insesgado de la covarianza denotado *Sxy* de dos variable aleatorias *x* e *y* es:

$$Sxy=\frac{1}{(n-1)}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) \quad (1)$$

El criterio es:

Sxy > 0 Existe dependencia directa positiva. Esto es, a grandes valores de x corresponden grandes valores de y

Sxy=0 No existe una relación lineal entre las dos variables

Sxy<0 Existe una dependencia negativa. Esto es, a grandes valores de x corresponden valores pequeños de y.

En el uso de la formula en forma directa se tiene un sesgo si el trabajador está capacitado en un área que no es requerido por la tarea, por lo que antes de utilizar la covarianza, se requiere realizar el operando lógico AND a los dos vectores (trabajador[i] AND trabajo[i]). De esta forma, se escoge al trabajador con el valor más alto en la covarianza (ya que es el trabajador con mayor dependencia con respecto al trabajo a realizar) [2,3,4].

Table 3. Tabla de correlación.

Tarea/trabajador 1,1,1,1,0,1,0,1,1,1	**7/elementos que coincide el trabajador**	**Sxy**
0,0,0,0,1,0,1,0,1,0	7/0	0
1,0,0,0,1,0,1,0,1,0	7/1	.033
1,0,0,0,1,0,1,1,1,0	7/2	0.066
1,1,0,0,1,0,1,1,1,0	7/3	0.1
1,1,0,0,1,1,1,1,1,0	7/4	0.13
1,1,0,0,1,1,1,1,1,1	7/5	0.16
1,1,1,0,1,1,1,1,1,1	7/6	0.2
1,1,1,1,1,1,1,1,1,1	7/7	0.23
Tarea/trabajador 1,0,1,0,0,1,0,0,0,1		
0,1,0,1,1,0,1,1,1,0	4/0	0
1,1,0,1,1,0,1,1,1,0	4/1	0.066
1,1,0,1,1,0,1,1,1,1	4/2	0.13
1,1,1,1,1,0,1,1,1,1	4/3	0.2
1,1,1,1,1,1,1,1,1,1	4/4	0.26

El algoritmo se desarrolló en el lenguaje de programación Java con el IDe *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.

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*).

Se utilizaron 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*argaEmpleados* 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.

Table 4. Perfil del empleado.

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:	N Horas	Nombre N

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.

Table 5. Perfil de las tareas.

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 N

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.

Una vez obtenido el perfil de la tarea y la duración de la misma en horas, se invoca el método B*usca* 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);
for(int i=0; i<Empleados.length; i++)
   boolean enc = false;
  char [] Buscar = new char [8];
String [] PEmpleados = Empleados[i].split(":");
PEmpleados [0].getChars(0,8,Buscar,0);
for( j=0; j<8;j++){
  if (Tarea[j]= = '1'){
    iIf (Buscar[j] = = '1')
      enc = true;
else{
   enc = false;
break;
}
}
}
```

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.*

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.*

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

3. Resultados y discusión

El algoritmo "asignación de tareas" puede permitir optimizar la distribución de tareas a cada empleado de una manera automatizada.

Con *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. Fig. 1.

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, con el botón *Regresar* podemos de nuevo acceder al Menú principal. Fig. 2.

Finalmente *Informe*, muestra un reporte detallado de la asignacion de empleados para cada tarea, dentro de una tabla la cual cuenta con 6 columnas encabezadas con:

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.

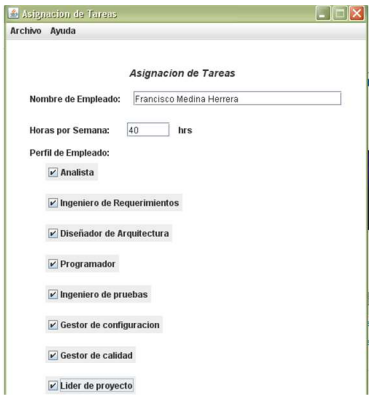

Fig. 2. Captura del perfil del desarrollador.

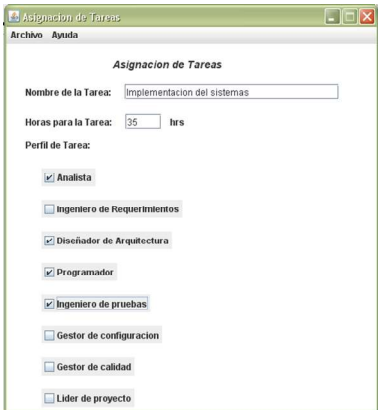

Fig. 3. Captura del perfil de las tareas.

Cabe mencionar que el líder del proyecto es el que usa y toma la decisión en función de los resultados obtenidos. En la Fig. 4 se muestra la gráfica de las tareas asignadas a cada desarrollador.

NomTarea	PerfilTarea	HoraTarea	IdEmpleado	PerfilEmple...	HorasRes
Lider de pr...	00000001	40	0	11111111	0
Desarrolla...	11101000	40	1	11111111	0
Administra...	10000010	40	2	11111111	0
Diseñador	11100100	15	3	11111111	25
Analista	11100000	12	3	11111111	13
Digitador	01111100	30	4	11111110	10
Analisis de...	10010000	20	5	11111110	20
Diseño de ...	10110000	30	6	11111110	10
Implement...	10111000	35	7	11111100	5
Prueba del ...	00111100	25	8	11111100	15
Evaluacion ...	10000000	10	3	11111111	3
Desarrollar...	00010000	35	9	11111000	5

Fig. 4. Informe de la asignación automática de tareas.

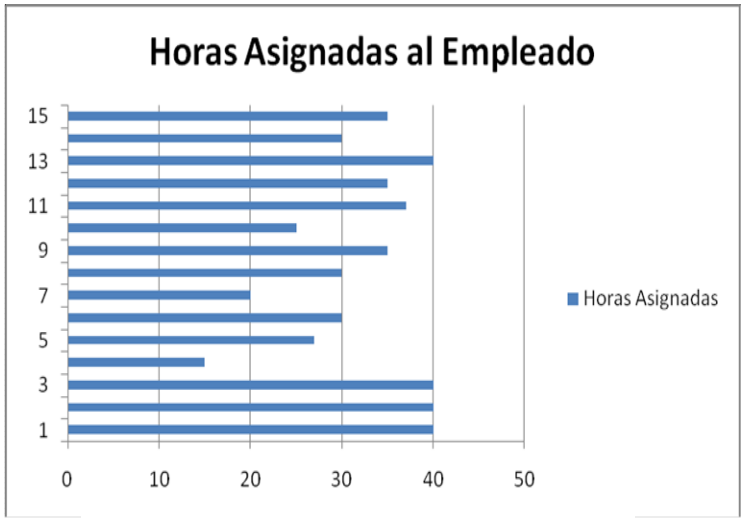

Fig. 5. Gráfica de las horas asignadas al empleado.

4. 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.

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.

5. Referencias

1. Mora, A. Moreno, I.:Desarrollo de un sistema de asignación 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. 2007.
2. Infante, S., Zárate, G.: Métodos estadísticos, un enfoque interdisciplinario. Ed. Trillas. 1984.
3. Shledon, R., Introducción a la estadística. Edición segunda, Ed. Reverté. España. 2007.
4. Box, G., Hunter, J. S., Hunter, W. G.: Estadística para investigadores. Segunda edición, Ed. Reverté. España. 2008
5. Sommerville, I. Ingeniería de software. Séptima edición. Ed. Pearson Educación. España. 2011.
6. Sánchez, S., Sicilia, M. A., Rodríguez D.: Ingeniería de software un enfoque práctico SWEBOK. Ed. Alfaomega, España. 2012.
7. Caper, J.: Estimación de costos y administración de proyectos de software. Segunda edición. Ed. McGraw Hill. 2008.