CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, A.C.

Modelo para la selección óptima de proyectos candidatos en organizaciones desarrolladoras de software usando programación por metas

TESIS

que para obtener el grado de

Maestría en Ciencias con especialidad en Computación y Matemáticas Industriales

presenta

Rosa Virginia Icedo Ojeda

Directores de Tesis

Dr. Carlos Montes de Oca Vázquez

Dr. Francisco Sánchez Sánchez

Octubre de 2004. **Guanajuato, Gto., México.**

Modelo para la selección óptima de proyectos candidatos en organizaciones desarrolladoras de software usando programación por metas

por

Rosa Virginia Icedo Ojeda

Act., Universidad Autónoma de Guadalajara (2002)

Sometida a revisión al Departamento de Ciencias de la Computación
en el cumplimiento parcial de los requisitos para obtener el grado de

Maestro en Ciencias de la Computación y Matemáticas Industriales

en el

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS A.C,

Octubre 2004

Firma del autor.....
Act. Rosa Virginia Icedo Ojeda
Departamento de Ciencias de la Computación
Octubre de 2004

Certificado por.....
Dr. Carlos Montes de Oca Vázquez
Grupo de Ingeniería de Software, Investigador Titular A
Director de Tesis

Certificado por.....
Dr. Francisco Sánchez Sánchez
Grupo de Matemáticas Aplicadas, Investigador Titular A
Co-Director de Tesis

Aceptado por.....
Dr. Salvador Botello Rionda
Coordinador de la Maestría en Ciencias de la Computación

Modelo para la selección óptima de proyectos candidatos en organizaciones desarrolladoras de software usando programación por metas

por

Rosa Virginia Icedo Ojeda

Sometida a revisión al Departamento de Ciencias de la Computación
en Octubre de 2004, en el cumplimiento parcial de los
requisitos para obtener el grado de
Maestro en Ciencias de la Computación y Matemáticas Industriales

Resumen

Existen diversos modelos y métodos para encontrar la selección óptima de proyectos en general, sin embargo, ninguno de los modelos encontrados en la literatura actual resuelve en particular el problema de selección de proyectos de software.

El problema que resuelve esta tesis surge en una ODS en particular establecida en México, pero es general de organizaciones similares. El problema consiste en cómo seleccionar de manera óptima los proyectos candidatos que resulten de mayor rentabilidad económica y optimicen el tiempo de asignación de los ingenieros de software, a la vez que garanticen un flujo de efectivo positivo. Además se requiere que la solución sea fácil y práctica de implementar.

La propuesta de esta tesis para seleccionar proyectos de software es un modelo de optimización de recursos humanos basado en programación por metas que incorpora el valor presente de los flujos de efectivo como medida de la utilidad proporcionada y que considera como restricciones a la disponibilidad de los recursos y el flujo de efectivo. En términos generales el modelo propuesto solo toma en cuenta el recurso humano, no asume incertidumbre y se limita a seleccionar proyectos de software candidatos que sean independientes entre si. Ahora bien, entre sus ventajas se encuentra que considera múltiples periodos de tiempo en el establecimiento de las metas y las restricciones, permite evaluar proyectos heterogéneos, es flexible en sus parámetros de entrada (de tal manera que es posible adaptarlo a las políticas de decisión de cada ODS), y logra combinar dos objetivos de

manera simultánea en una función mono-objetivo. Además, el algoritmo de solución no está limitado a un algoritmo específico y pueden utilizarse diversas técnicas para tal efecto.

Las alternativas exploradas en esta tesis para solucionar el modelo matemático propuesto incluyen la técnica de ramificación y acotamiento, utilizando el lenguaje de modelado LINGO, y la enumeración explicita, incorporada en una herramienta nombrada RQS y construida con el fin de ilustrar el comportamiento del modelo propuesto.

RQS fue programada en *Visual Basic para Aplicaciones* (VBA) y se utilizó una interfaz en *Excel*. Además de encontrar la selección óptima, RQS ofrece ventajas adicionales al tomador de decisiones: alimentación automática de datos, configuración de ciertos valores, resultados gráficos, tablas, listado con los resultados de todas las combinaciones.

Las contribuciones de esta tesis pueden resumirse en los siguientes tres puntos: primero, el planteamiento del problema de selección de proyectos de software candidatos en un modelo de optimización binaria con restricciones que no está limitado a un sólo algoritmo de solución, segundo, el modelo propuesto es lo suficientemente flexible en sus parámetros de entrada para ajustarse a las políticas de decisión de cada empresa, y tercero, la herramienta de decisión RQS.

Agradecimientos

A Dios, el constructor de mi destino.
A mis amigos, por su amistad y comprensión.
A mis padres y hermanos, por su cariño.
A Moisés, por su apoyo incondicional.

A CONACYT por otorgarme una beca,
sin la cuál no hubiera sido posible este sueño.

Un agradecimiento especial a mis directores de tesis,
Dr. Carlos Montes de Oca y Dr. Francisco Sánchez.

A todos los que formaron parte de mi vida
durante mi estancia en Cimat.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Una *organización desarrolladora de software a la medida* (ODS) es una organización cuyo giro principal es el desarrollo de software para otras organizaciones. Normalmente una ODS trabaja en base a proyectos de software. En otras palabras, los ingresos de una ODS los consigue mediante la realización de proyectos de software. Más aún, la utilidad y crecimiento de este tipo de organizaciones depende de los proyectos de software contratados.

Desde un punto de vista de negocios, una ODS funciona como una empresa de servicios. La mayor fuente de sus ingresos es por el servicio que ofrece, es decir el desarrollo de software, la mayor parte de sus egresos es por conceptos de salario, y el recurso más importante lo constituyen los ingenieros de software.

A pesar de que las ODS tienden a especializarse en dominios de aplicación pequeños, no es inusual que acepten proyectos de cualquier dominio; entonces, para una ODS resulta común tener proyectos heterogéneos. Los proyectos de software pueden tener distinto tamaño, duración, tecnología, dominio, requerimientos de recursos humanos, precio, costo y esquema de pago. Hay que resaltar que el esquema de pagos de un proyecto generalmente lo determina el cliente, así que pueden existir tantos esquemas de pago como clientes tenga una ODS.

Los proyectos en una ODS se clasifican en proyectos en curso y proyectos candidatos. Los proyectos en curso son los proyectos bajo contrato y los proyectos candidatos son los proyectos que están aún en negociaciones con el cliente. La información disponible de los proyectos candidatos varía a lo largo del proceso de negociación, inclusive la fecha de inicio, el costo y la fecha de finalización del proyecto pueden cambiar día a día durante este proceso.

Cuando la ODS no tiene los suficientes recursos para aceptar todos los proyectos candidatos, el tomador de decisiones dentro de la ODS debe decidir que subconjunto de los proyectos candidatos seleccionar. Además, debido a que el esquema de pagos de los proyectos en curso y los proyectos candidatos no es constante mientras que lo mayoría de los egresos si lo es, es necesario seleccionar proyectos de software que mantengan un flujo de efectivo acumulado positivo, si no es así, la ODS puede padecer de escasez de efectivo.

Para pequeñas y medianas ODS, seleccionar los proyectos candidatos correctos puede ser vital para su sobrevivencia y crecimiento. Por ejemplo, puede existir una combinación de proyectos que generen mucha utilidad a largo plazo, pero conducen a valles financieros en el corto plazo que pueden provocar la quiebra de la ODS.

En resumen, seleccionar proyectos de software no sólo implica elegir una combinación de proyectos que logre un balance entre el número de proyectos y los recursos disponibles, si no que también engloba diversos aspectos a considerar como: ventana de evaluación común, optimización de recursos, utilidad proporcionada, evaluación de proyectos heterogéneos, impacto en el flujo de efectivo, disponibilidad de recursos, información escasa y cambiante, estimación de cierta información, incertidumbre, y rapidez en la selección.

Ahora bien, el problema que resuelve esta tesis surge en una ODS en particular establecida en México conocida como QS, pero es general de organizaciones similares. El problema consiste en cómo seleccionar de manera óptima los proyectos candidatos que resulten de mayor rentabilidad económica y optimicen el tiempo de asignación de los ingenieros de

software, a la vez que garanticen un flujo de efectivo positivo. Además se requiere que la solución sea fácil y práctica de implementar.

Revisando la literatura actual, se encontró que existen diversos modelos y métodos para encontrar la selección óptima de proyectos en general, entre estos destacan los modelos basados en programación matemática que están combinados con algún tipo de modelo financiero. Sin embargo, ninguno de los modelos encontrados en la literatura actual resuelve en particular el problema de selección de proyectos de software.

Por lo tanto, se definieron 3 objetivos para resolver el problema de esta tesis. El primer objetivo es plantear un modelo matemático que capture la problemática de la selección de proyectos de software en QS y que pueda ser generalizado para una ODS. El segundo objetivo es construir una herramienta de decisión basada en el modelo matemático que sea fácil de usar e interpretar por una persona que no necesariamente conoce la estructura del modelo matemático. Y por último, el tercer objetivo es implementar la solución a la empresa QS.

El modelo matemático propuesto para satisfacer el primer objetivo, es un modelo de optimización de recursos humanos basado en programación por metas que incorpora el valor presente de los flujos de efectivo como medida de la utilidad proporcionada y que considera como restricciones a la disponibilidad de los recursos y el flujo de efectivo. En términos generales el modelo propuesto solo toma en cuenta el recurso humano, no asume incertidumbre y se limita a seleccionar proyectos de software candidatos que sean independientes entre si. Ahora bien, entre sus ventajas se encuentra que considera múltiples periodos de tiempo en el establecimiento de las metas y las restricciones, permite evaluar proyectos heterogéneos, es flexible en sus parámetros de entrada (de tal manera que es posible adaptarlo a las políticas de decisión de cada ODS), y logra combinar dos objetivos de manera simultánea en una función mono-objetivo. Además, el algoritmo de solución no está limitado a un algoritmo específico y pueden utilizarse diversas técnicas para tal efecto.

Las alternativas exploradas en esta tesis para solucionar el modelo matemático propuesto incluyen la técnica de ramificación y acotamiento, utilizando el lenguaje de modelado LINGO, y la enumeración explicita, incorporada en una herramienta nombrada RQS y construida con el fin de ilustrar el comportamiento del modelo propuesto.

RQS fue programada en *Visual Basic para Aplicaciones* (VBA) y se utilizó una interfaz en *Excel*. Además de encontrar la selección óptima, RQS ofrece ventajas adicionales al tomador de decisiones: alimentación automática de datos, configuración de ciertos valores, resultados gráficos, tablas, listado con los resultados de todas las combinaciones. La principal desventaja de RQS es que está limitada a un número máximo de hasta 10 proyectos en curso y 10 proyectos candidatos.

Este documento está organizado de la siguiente forma: en el capítulo 2 se hace una revisión de algunos modelos y métodos para seleccionar proyectos y se presentan algunos trabajos anteriores relacionados, en el capítulo 3 se encuentra el contexto y la descripción del problema de selección de proyectos candidatos de software, los objetivos y la motivación de este trabajo de tesis, en el capítulo 4 se encuentra la formulación del modelo propuesto y los métodos de solución, en el capítulo 5 se describe la herramienta RQS y se presentan algunos resultados ilustrativos del modelo. Finalmente, en el capítulo 6 se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Algunos Modelos y Métodos para Seleccionar Proyectos

Un *Proyecto* se define como "esfuerzo temporal emprendido para crear un producto o servicio único" [35]. Algunas características que tiene un proyecto son: las tareas involucradas no son rutinarias, se requiere de una planeación, se quiere alcanzar objetivos específicos, el proyecto tiene un determinado tiempo de vida, su realización requiere de diversos especialistas, el trabajo se divide por fases, y los recursos que están disponibles para el proyecto son finitos [23].

El problema de seleccionar proyectos se presenta cuando hay muchos proyectos y los recursos son insuficientes [8]. La selección de proyectos es un proceso de decisión dinámico. Este proceso está caracterizado por información incierta y cambiante, oportunidades dinámicas, múltiples metas y consideraciones estratégicas e interdependencia entre proyectos [11].

Actualmente existen diversos modelos y métodos para encontrar la selección óptima de proyectos. La mayoría de las clasificaciones mostradas en [5, 11, 13, 29, 38] coinciden en incluir las tres clasificaciones que se mencionan en este capítulo:

1. *Modelos que usan programación matemática*: se formula un objetivo a maximizar/minimizar bajo un conjunto de restricciones utilizando

programación lineal, programación entera, programación dinámica, programación por metas o alguna otra técnica no lineal.

2. *Modelos financieros o económicos*: relativos al periodo de recuperación de la inversión, valor presente neto, tasa interna de retorno, razón beneficio/costo, entre otros.

3. *Modelos de decisión*: modelos de ordenamiento, cheklist, análisis de jerarquías, enfoques de comportamiento, diagramas de burbujas, árboles de decisión, y teoría de la utilidad.

Las tres primeras partes de este capítulo contienen una breve enumeración de los métodos contenidos en cada una de las tres clasificaciones mencionadas arriba y de sus ventajas/desventajas en la medida de lo posible. Cabe mencionar que si bien es claro separar los modelos de selección, también existen muchos modelos y métodos híbridos, es decir, que son combinación de otros modelos. Se finaliza el capitulo resumiendo trabajos anteriores relacionados a la selección de proyectos.

2.1 Modelos que usan programación matemática

Los primeros modelos de selección de proyectos, alrededor de los 60's y 70's, eran altamente matemáticos y empleaban técnicas como programación lineal, dinámica, y entera. El objetivo que persiguen estos modelos es maximizar/minimizar alguna función objetivo sujeta a un conjunto de restricciones que incluye a los proyectos existentes y a los nuevos proyectos [11].

Este tipo de modelos es muy criticado por varios autores debido a que:

- El método es difícil de entender para personas no expertas en investigación de operaciones [29]

- Requieren una gran cantidad de información. Por ejemplo: información financiera, tiempo de recursos necesarios, y probabilidades de terminación del proyecto [11].
- Históricamente han realizado un inadecuado tratamiento del riesgo y la incertidumbre [11].
- Dentro de aplicaciones de Ingeniería de Software existen limitaciones practicas al formular problemas matemáticamente, ya que es necesario hacer supuestos que son difíciles de justificar [6].

Sin embargo, la programación matemática es una metodología bien establecida [29] que ha permitido en años recientes contar con modelos de selección más prácticos y realistas [13]. Adicionalmente son fáciles de resolver con ayuda de una computadora y no requieren de software muy especializado debido a que las hojas de cálculo incluyen funciones para resolver este tipo de modelos, aunque claro a pequeña escala [22]. Respecto a los beneficios directos en la toma de decisiones, se puede mencionar que los modelos de programación matemática permiten variar los parámetros de entrada y recalcular el algoritmo de manera inmediata, proporcionando información importante respecto a varios escenarios. Mas aún, plantear un problema de programación matemática ayuda a clarificar algunos aspectos del proceso de decisión, como por ejemplo: ¿Qué se está optimizando?, ¿Cuáles son las variables de decisión?, y ¿Cuáles son las variables bajo control? [6].

A continuación se detallan dos tipos de modelos de programación matemática, el modelo de programación lineal y el modelo de programación por metas, por considerarlos los mas populares dentro de la selección de proyectos. Al final de esta sección se mencionan otros tipos de modelos basados en programación matemática haciendo énfasis en los modelos de programación no lineal.

2.1.1 Programación lineal

La *programación lineal* es el estudio de los modelos matemáticos concernientes a la asignación eficiente de los recursos limitados en las actividades conocidas, con el objetivo de

satisfacer las metas deseadas (tal como maximizar beneficios o minimizar costos) [22]. El propósito de la programación lineal es encontrar los valores de $x=(x_1,x_2,...,x_n)$ para maximizar o minimizar funciones lineales de la forma:

Max o Min $F(x)$	(función objetivo)
Sujeto a	
$Ax+s \leq b$	(restricciones lineales)
$x_1,x_2,...,x_n \geq 0$	(no negatividad)

Dónde x_i puede ser continua o discreta. La función *F(x)* mapea un conjunto de valores de x_i a un valor real. El conjunto de restricciones debe de ser lineal y los coeficientes de *A, s* y *b* son conocidos.

La terminología para las soluciones del modelo es la siguiente:

- *Solución*: cualquier conjunto de valores específicos para las variables de decisión $(x_1,x_2,\ldots,x_n)$.
- *Solución factible:* solución en la que todas las restricciones se satisfacen.
- *Solución no factible*: solución en la que al menos una restricción se viola.
- *Solución óptima*: solución factible que proporciona el valor más favorable de la función objetivo, es decir, el valor más grande si la función objetivo debe de maximizarse o el valor más pequeño si la función objetivo ha de minimizarse.

Hay diversas formas de solucionar un problema de programación lineal. El algoritmo más común es el método Simplex desarrollado por Dantzing en 1947: está comprobada su extraordinaria eficiencia, se usa en forma rutinaria para resolver problemas grandes y existe una amplia variedad de paquetes de software para ello [22].

Generalmente los modelos de selección de proyectos exigen que algunas variables de decisión sean enteras [2, 29, 34], o binarias [13, 30]. Para resolver esta exigencia existen los

modelos de *programación entera.* Estos modelos se diferencian de los modelos de programación lineal por el hecho de exigir que los valores $x_1,x_2,...,x_n$ sean valores enteros. Por otro lado, si solo es necesario que algunas de las variables tengan valores enteros entonces el modelo se conoce como de *programación entera mixta.*

A pesar de que los problemas de programación entera tienen un número finito de soluciones, en general es mucho más sencillo resolver problemas de programación lineal que los problemas de programación entera. Uno de los algoritmos mas populares para pequeños problemas de programación entera es la técnica de *ramificación y acotamiento (branch-and-bound)* y las ideas relacionadas con la *enumeración implícita* de las soluciones factibles enteras. Para problemas grandes se pueden utilizar los algoritmos heurísticos ya que son muy eficientes, pero no garantizan que se llegue a una solución óptima. Tres tipos de algoritmos sobresalen: *búsqueda de tabú, simulación de templado* y *algoritmos genéticos* [22].

2.1.2 Programación por metas

Un caso especial de programación lineal es la programación lineal por metas. La programación por metas es una herramienta utilizada por los administradores para resolver problemas con múltiples metas. Ha sido empleada para diferentes fines: seleccionar proyectos [30], seleccionar un sistema de armamento [9], ordenamiento de alternativas [16], esquema de nutrición para pacientes en un hospital [26], diseño de ingeniería [12], y a finanzas, administración y mercadotecnia [33].

La mayor diferencia entre programación lineal y programación por metas (PM) es que el modelo de PM no optimiza el objetivo directamente como en el caso de programación lineal, en su lugar, procura minimizar las desviaciones entre las metas deseadas y los resultados obtenidos [30].

PM ofrece una variedad de modelos que pueden diferir en el criterio de establecimiento de metas, en el criterio de optimización (minimización de la suma pesada de las desviaciones a la meta, minimización de la máxima desviación), en la función de penalización, y en las restricciones (lineales, no lineales, variables continuas, variables discretas).

El modelo estándar de PM basado en la minimización de la suma pesada de las desviaciones a las metas es formulado como [27]:

$$\text{Min } Z=\sum_{i=1}^{r} w_i^+ d_i^+ + \sum_{i=1}^{r} w_i^- d_i^- \tag{2.1}$$

$$\sum_{j=1}^{n} c_{ij} x_j + d_i^- - d_i^+ = g_i \qquad i=1,2,...,r \tag{2.2}$$

$$\sum_{j=1}^{n} a_{kj} x_j \le b_k \qquad k=1,2,...,m \tag{2.3}$$

$$x_1, x_2, ..., x_n \ge 0, \quad d_i^- \ge 0, \quad d_i^+ \ge 0 \tag{2.4}$$

Donde:

n es el número de variables de decisión.

r es el número de metas flexibles

m es el número de restricciones

x_j es la j-ésima variable de decisión.

c_{ij} es el coeficiente de la i-ésima meta para la j-ésima variable de decisión.

a_{kj} es el coeficiente de la k-ésima restricción para la j-ésima variable de decisión.

g_i es el valor de la i-ésima meta.

$d_i^-, (d_i^+)$ es la variable de desviación negativa (positiva) que indica el valor alcanzado por debajo (arriba) de la i-ésima meta.

$w_i^-, (w_i^+)$ es el peso que expresa la importancia de estar por debajo (arriba) de la i-ésima meta.

Si g_i es el valor numérico de la i-ésima meta y $f(x)$ el valor alcanzado por $x=(x_1,x_2,...,x_r)$, entonces es posible establecer 4 tipos de metas [12]:

1) Meta unilateral inferior: $f(x) \le g_i$
2) Meta unilateral superior: $f(x) \ge g_i$
3) Meta bilateral: $f(x) = g_i$
4) Meta establecida en un rango: $g_i^l \le f(x) \le g_i^u$

La función de penalización es la que función que ayuda a determinar los con que se penalizará el exceder o no alcanzar la meta. La figura 2.1 muestra ejemplos de cómo puede construirse una función de penalización. La figura (b) corresponde a la función de penalización utilizada en el modelo estándar de PM basado en la minimización de la suma pesada de las desviaciones ejemplificado anteriormente.

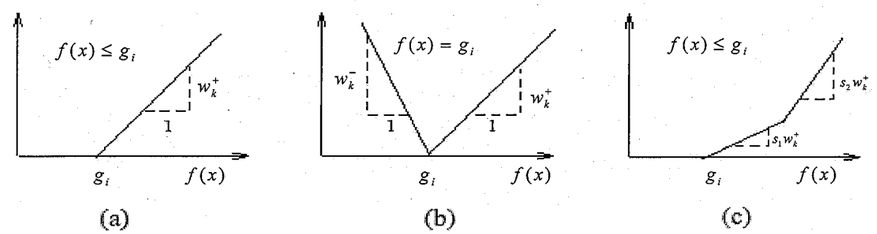


Figura 2.1 Diferentes funciones de penalización.

El modelo estándar de PM presentado en este apartado puede ser generalizado en diferentes formas. El modelo que se presenta a continuación es una generalización que emplea una función de penalización como la presentada en la figura 2.2.

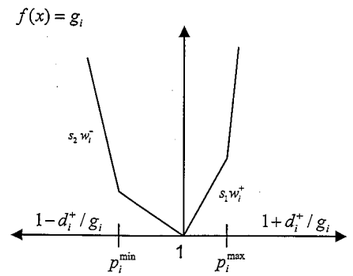


Figura 2.2 Función de penalización para el modelo generalizado de PM.

En este modelo las desviaciones positivas y negativas son normalizadas dividiéndolas entre la meta siempre y cuando sea distinta de cero. En este sentido el valor normalizado 1 significa el cumplimiento del 100% de la meta, los valores menores a 1 indican que la meta no fue alcanzada y los valores mayores a 1 señalan que la meta fue superada. El tomador de decisiones debe de especificar los límites $p_i^{\min}<1$ y $p_i^{\max}>1$ para el primer nivel de la función de penalización. Usualmente los valores de $p_i^{\min}$ y $p_i^{\max}$ varían de 0.80-0.95 y 1.05-1.20 respectivamente.

Las desviaciones positivas y negativas de la meta dentro del intervalo $[p_i^{\min}, p_i^{\max}]$ son penalizadas por el valor marginal de $s_1 w_i^-$ y $s_1 w_i^+$, donde s_1 es el coeficiente para el primer nivel de la función de penalización d_{i1}^- y d_{i1}^+ son las variables que miden las desviaciones. Similarmente d_{i2}^- y d_{i2}^+ son las variables que miden las desviaciones a la meta que están fuera del intervalo $[p_i^{\min}, p_i^{\max}]$ y son penalizadas por $s_2 w_i^-$ y $s_2 w_i^+$. El valor de s_2 es usualmente más grande que s_1.

El modelo generalizado de PM con una función de penalización de 2 niveles basado en la minimización de la suma pesada de las desviaciones a las metas es formulado como [27]:

$$\text{Min } Z=s_1 \sum_{\substack{i=1 \\ g_i\neq 0}}^{r} \frac{w_i^- d_{i1}^- + w_i^+ d_{i1}^+}{g_i} + s_2 \sum_{\substack{i=1 \\ g_i\neq 0}}^{r} \frac{w_i^- d_{i2}^- + w_i^+ d_{i2}^+}{g_i} \qquad (2.5)$$

Sujeto a :

$$\sum_{j=1}^{n} c_{ij}x_j + d_{i1}^- + d_{i2}^- - d_{i1}^+ - d_{i2}^+ = g_i \qquad i=1,2,...,r, \quad g_i \neq 0 \qquad (2.6)$$

$$\sum_{j=1}^{n} c_{ij}x_j = g_i \qquad i=1,2,...,r, \quad g_i = 0 \qquad (2.7)$$

$$\sum_{j=1}^{n} a_{kj}x_j \leq b_k \qquad k=1,2,...,m \qquad (2.8)$$

$$d_{i1}^- \leq g_i - p_i^{\min} g_i \qquad i=1,2,...,r \qquad (2.9)$$

$$d_{i1}^+ \leq p_i^{\max} g_i - g_i \qquad i=1,2,...,r \qquad (2.10)$$

$$d_{i2}^- \geq g_i - p_i^{\min} g_i \qquad i=1,2,...,r \qquad (2.11)$$

$$d_{i2}^+ \geq p_i^{\max} g_i - g_i \qquad i=1,2,...,r \qquad (2.12)$$

$$x_1, x_2, ..., x_n \geq 0, \qquad i=1,2,...,r \qquad (2.13)$$

$$d_{i1}^- \geq 0, \ \ d_{i1}^+ \geq 0, \ \ d_{i2}^- \geq 0, \ \ d_{i2}^+ \geq 0 \qquad (2.14)$$

Para solucionar problemas de PM se utilizan algoritmos basados en los procedimientos de solución de programación lineal, como el procedimiento de solución desarrollado por Schniederjans y Kawk que es un algoritmo modificado del método Simples [33]. En el caso de PM entera los métodos de solución son versiones modificadas de los métodos de solución de programación entera.

Desde el punto de vista computacional hay muchos paquetes disponibles que resuelven los problemas de PM. La desventaja en general de estos sistemas de optimización consiste en la necesidad de preparar la información de entrada en una forma apropiada, así que pequeños cambios en el modelo matemático pueden conducir a cambios sustanciales en la manera de agrupar los datos de entrada [27].

2.1.3 Otros modelos de programación matemática

La programación matemática no sólo abarca problemas de programación lineal, programación entera, programación mixta y programación por metas, si no también programación dinámica, teoría de juegos, modelos de redes, programación no lineal, entre otras [40].

El caso de la programación no lineal es interesante, ya que generalmente se combina con alguna de los otros modelos de programación lineal, por ejemplo es posible construir un modelo de programación por metas no lineal [12]. La *programación no lineal* surge cuando las restricciones o la función objetivo son lo suficientemente no lineales como para aproximar la solución mediante modelos de programación lineal. Las técnicas para resolver problemas de programación no lineal generalmente son más complejas que las técnicas para solucionar problemas lineales. Una función objetivo no lineal puede tener múltiples máximos/mínimos locales esto conduce a que algunos algoritmos no lineales no lleguen a una solución o la solución no sea la óptima [6].

2.2 Modelos financieros o económicos

Los modelos financieros son ampliamente usados y fáciles de interpretar. Entre los más populares se encuentran el periodo de recuperación de la inversión, la tasa interna de retorno (TIR), el análisis costo/beneficio, y el método del valor presente neto (VPN). Otro tipo de modelos financieros son los probabilísticos e incluyen simulaciones como la de Monte Carlo [11]. La principal desventaja del método del periodo de recuperación de inversión es que no toma en cuenta el valor del dinero a través del tiempo, en cambio el VPN si lo hace. De igual manera el análisis costo/beneficio y la TIR, son técnicas ampliamente usadas pero no consideran el riesgo asociado a los proyectos. Debido a esto, los análisis que incluyen probabilidades y riesgo son más atractivos [38]. Finalmente, el uso de técnicas de evaluación financiera en Ingeniería de Software tiene limitaciones ya que estimar los costos y los beneficios de un proyecto puede ser una tarea difícil e imprecisa [1, 14, 36].

Más que seleccionar proyectos, la mayoría de los modelos financieros evalúan proyectos individuales y proporcionan criterios para ordenar los proyectos. Algunos modelos de selección de proyectos combinan los modelos financieros con otros modelos para hacer un ordenamiento de alternativas [29].

Esta sección se organiza en 4 partes: 1) periodo de recuperación de la inversión, 2) método del valor presente neto, 3) tasa interna de retorno, 4) índice de rentabilidad o razón beneficio/costo.

2.2.1 Periodo de recuperación de la inversión

El periodo de recuperación es el tiempo, en años y fracciones de año, que se requiere para recuperar la inversión inicial de un proyecto. Consiste en sumar los flujos de efectivo netos del proyecto hasta recuperar la inversión inicial [1, 17, 21].

Criterio de decisión:

- Si el periodo de recuperación calculado es menor que el periodo de recuperación establecido por la empresa, el proyecto debe aceptarse; de lo contrario, debe rechazarse.

Ventajas del método:

- Es fácil de aplicar e interpretar.
- Es una medida de liquidez del proyecto, ya qué indica que tan rápido se recupera la inversión.
- Es útil para comunicar los resultados a posteriori más que hacer un análisis a priori.

Desventajas del método:

- Ignora el valor del dinero en el tiempo debido a que no descuenta los flujos de efectivo.
- Les da igual peso a todos los flujos de efectivo.
- No toma en cuenta a los flujos de efectivo que ocurren después de que se recupera la inversión.

- Es difícil establecer el periodo de retorno de inversión deseado por la empresa. Este periodo es el que se usa en el criterio de decisión y generalmente es arbitrario y subjetivo.
- Favorece a los proyectos de corto plazo, ya que obliga a rechazar los proyectos con periodos de recuperación superiores al periodo deseado, aunque sean rentables a largo plazo.
- Se enfoca en el tiempo necesario para recuperar la inversión y no en la rentabilidad que esta representa para los accionistas de la empresa. Es decir, no toma en cuenta el verdadero valor del proyecto.

El criterio de decisión, las ventajas y desventajas del método del VPN se obtuvieron de las siguientes referencias: [1, 17, 21].

Ejemplo:

Considérese un escenario con 3 proyectos A, B, y C con igual periodo de recuperación de la inversión pero distintos flujos de efectivo. La tabla 2.1 muestra que todas las opciones producen un periodo de recuperación de la inversión de 2 años, pero el proyecto A tiene ingresos más tempranos que el proyecto B y esto puede significar una ventaja desde el punto de vista de negocios. Mientras que el proyecto C genera mayores ingresos en un plazo mayor.

Flujo de efectivo	Proyecto A	Proyecto B	Proyecto C
F_0	-2000	-2000	-4000
F_1	1000	500	-1500
F_2	1000	1500	5500
F_3	1000	1000	6000
Periodo de recuperación	2 años	2 años	2 años
Flujo neto final	1000	1000	6000

Tabla 2.1 Ejemplo del método del periodo de recuperación de la inversión.

2.2.2 Valor presente neto

El método del valor presente neto (VPN) es por mucho el método financiero mayormente aceptado en el mundo [21]. Antes de dar una definición del VPN es necesario mencionar que es el valor presente.

El *valor presente* es el "valor de hoy de futuros flujos de efectivo estimados" [17]. La fórmula del flujo de efectivo descontado para el valor presente es definida como:

$$VP = \frac{F_k}{(1+D_k)^k} \tag{2.15}$$

Donde:

F_k = Flujo de efectivo en el periodo k.

D_k = Tasa de descuento aplicada en el periodo k.

La tasa de descuento D_k es un factor que cuantifica el valor del dinero en el tiempo y corresponde a la rentabilidad mínima que se le debe exigir al proyecto. También puede entenderse como el costo de los recursos necesarios para financiar el proyecto, es decir, si el proyecto se financia con deuda, la tasa de descuento es la tasa de interés que la empresa paga a sus acreedores [1, 21].

Ahora bien, el VPN de los flujos de efectivo esperados es equivalente al valor de mercado del proyecto y permite conocer la ganancia o pérdida que se obtendría si el proyecto se lleva a cabo [1]. El método del VPN consiste en lo siguiente:

1. Proyectar todos los flujos de efectivo netos que se espera genere el proyecto durante toda su vida.
2. Calcular el valor presente de cada uno de los flujos.
3. Sumar los valores presentes de los flujos netos.

4. Calcular el VPN como la diferencia entre la inversión y las suma del valor presente de los flujos de efectivo netos que se espera obtener.

La fórmula general del VPN es la siguiente:

$$VPN = -I + \sum_{k=1}^{n} \frac{F_k}{(1+D_k)^k} \qquad (2.16)$$

Donde:

I = Inversión inicial requerida (incluye el costo de adquisición de activos fijos, costo de instalación de equipo, inversión en capital de trabajo, entre otros costos necesarios para poner en marcha el proyecto).

n = Número de periodos de vida del proyecto (tiempo de vida).

Criterio de decisión:

- Si el VPN > 0, el proyecto se debe aceptar porque crea valor para la empresa. Cuando el VPN es positivo los ingresos son superiores a los egresos.
- Si el VPN < 0, el proyecto se debe rechazar por que no crea valor para la empresa, debido a que los ingresos no son suficientes para enfrentar las obligaciones.
- Si el VPN = 0, es indiferente aceptar o rechazar el proyecto porque no crea ni destruye el valor para la empresa, ya que genera los ingresos justos para compensar sus gastos.
- Si se trata de elegir entre dos proyectos independientes, se debe elegir el que tenga un VPN mayor.

Ventajas del método:

- Se interpreta fácilmente el resultado en términos monetarios.

- Toma en cuenta el valor del dinero en el tiempo y todos los flujos de efectivo netos que se espera genere el proyecto.
- Vincula las decisiones de aceptación y rechazo de los proyectos con la maximización del valor de la empresa.
- Es sensible al tamaño y la escala de los proyectos.
- Es directamente comparable porque el NPV se expresa en la misma unidad (pesos al día de hoy).
- Permite comparar proyectos con distinto esquema de flujo de efectivo (como los proyectos de la tabla 2.2.1.1).
- Proporciona un criterio para ordenar múltiples proyectos. El proyecto que tenga el mayor VPN es la "mejor" opción.
- Es posible evaluar una combinación de proyectos debido a que posee la cualidad de ser aditivo. Esto es, para un proyecto P y un proyecto Q se tiene que $VPN_{P\,y\,Q} = VPN_P + VPN_Q$.

Desventajas del método:

- La determinación de la tasa de descuento apropiada.
- El VPN depende en gran medida de la tasa de descuento elegida.
- Supone una reinversión total de las ganancias anuales, lo cual no sucede en la mayoría de las empresas.

El criterio de decisión, las ventajas y desventajas del método del VPN se obtuvieron de las siguientes referencias: [1, 4, 6, 17, 21].

Ejemplo:

Se quiere calcular el VPN del proyecto A y el proyecto B. Los flujos de efectivo correspondientes están en la tabla 2.2.1.1, D_k =10% y n=3.

$$VPN_A = -2000 + 1000(1+0.10)^{-1} + 1000(1+0.10)^{-2} + 1000(1+0.10)^{-3} = 486.8$$

$$VPN_B = -2000 + 500(1+0.10)^{-1} + 1500(1+0.10)^{-2} + 1000(1+0.10)^{-3} = 445.5$$

De acuerdo a los resultados, el VPN_A es mayor que el VPN_B, así que se debería escoger el proyecto A. Ahora bien, si $D_k = 30\%$, los resultados son: $VPN_A = -183.9$ y $VPN_B = -272.6$. Como se puede notar, la elección de la tasa de descuento es decisiva para establecer el valor del proyecto debido a que el VPN es inversamente proporcional al valor de la tasa de descuento aplicada [4]. La relación entre el VPN y D_k puede representarse gráficamente como se muestra en la figura 2.3.

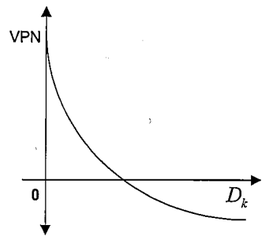


Figura 2.3 Relación entre el VPN y la tasa de descuento.

2.2.3 Tasa interna de retorno

La tasa interna de retorno (TIR) es la tasa de descuento que hace que el VPN sea cero, o bien es la tasa que iguala la suma de los flujos descontados a la inversión inicial [4]. La TIR es una tasa interna de rendimiento porque depende únicamente de los flujos de efectivo que genera el proyecto [1]. Al igual que el VPN y D_k, existe una relación inversa entra la TIR y el VPN: cuando se incrementa la TIR, disminuye el VPN, y cuando disminuye la TIR, se incrementa el VPN.

La TIR se despeja de la siguiente ecuación:

$$VPN = -I + \sum_{k=1}^{n} \frac{F_k}{(1+TIR)^k} = 0 \qquad (2.17)$$

Donde:

I = Inversión inicial requerida.

F_k = Flujo de efectivo en el periodo k.

TIR = Tasa de interna de rendimiento o retorno.

n = Número de periodos de vida del proyecto.

No existe una manera directa de obtener el valor de la TIR. Una alternativa es encontrarla mediante prueba y error, otra es usar tablas de interpolación. Algunos paquetes de software, como *Excel*, incorporan una función que permite calcular ese valor de manera automática [1].

Si ocurre que los flujos de efectivo del proyecto no son convencionales, esto es que además de la inversión inicial existen más flujos negativos, el método de la TIR puede generar tasas de rendimiento múltiples que provoquen que el VPN sea cero. El número máximo de TIRs que pueden existir es igual al número de cambios de signo de los flujos de efectivo y no es posible elegir una de ellas como medida de rentabilidad del proyecto. En estos casos es recomendable utilizar el método del VPN [4].

Criterio de decisión:

- Si TIR > D_k, el proyecto se debe aceptar porque genera flujos de efectivo superiores a los que se requieren para financiarlo.
- Si TIR < D_k, el proyecto se debe rechazar porque los flujos de efectivo que genera son inferiores a los que se requieren para financiarlo.
- Si TIR = D_k, es indiferente aceptar el proyecto.

Ventajas del método:

- Considera el valor del dinero en el tiempo.
- Conduce a decisiones idénticas a las del método de VPN siempre y cuando se trate de flujos convencionales y proyectos independientes.

Desventajas del método:

- Calcular la TIR a prueba y error es difícil y tardado, por lo que se requiere de una calculadora financiera o un programa computacional.
- No es válido cuando para evaluar proyectos con flujos de efectivo no convencionales, ya que en estos casos pueden existir múltiples TIRs, o incluso ninguna.
- Puede conducir a decisiones erróneas si se los proyectos son mutuamente excluyentes.
- No es sensible a la escala y tamaño de los proyectos, así que puede sugerir decisiones erróneas al momento de seleccionar proyectos.

El criterio de decisión, las ventajas y desventajas del método de la TIR se obtuvieron de las siguientes referencias: [1, 4, 17, 21].

Ejemplo:

La tabla 2.2 ejemplifica alguno de los problemas que se tienen con la TIR. En este ejemplo el proyecto D no tiene flujos de efectivo convencionales y esto ocasiona que tenga 2 valores posibles para la TIR. Similarmente el proyecto E tampoco tiene flujos de efectivo convencionales y existe una TIR tal que el VPN sea cero. Los proyectos F y G son muy distintos en cuanto al VPN que proporcionan, sin embrago si solo se sigue el criterio de la TIR mayor, se seleccionaría erróneamente el proyecto F.

Flujo de efectivo	Proyecto D	Proyecto E	Proyecto F	Proyecto G
F_o	-4000	1000	-100	-10000
F_1	25000	-3000	200	15000
F_2	-25000	2500	-----	-----
TIR (%)	25% y 400%	Ninguna	100%	50%
VPN ($D_k = 10\%$)	-1934	339	82	3636

Tabla 2.2 Ejemplo de diferentes TIR.

2.2.4 Índice de rentabilidad o razón beneficio/costo

Existen diversos índices que miden la proporción entre el beneficio y los costos. Dentro del ámbito de Ingeniería de Software es muy popular la tasa de retorno de inversión o ROI por sus siglas en inglés (return of investment) [18, 36]. Esta tasa puede ser definida como: ROI=beneficios/costos o ROI=(beneficios-costos)/costos. Ambas tienen diferente interpretación.

En cambio, el índice de rentabilidad (IR) se define como la razón existente entre la suma de los valores presentes de los flujos de efectivo netos de un proyecto, descontados con la tasa de descuento, y la inversión inicial requerida. El IR se puede interpretar como el valor presente o la rentabilidad obtenida por cada unidad monetaria invertida en el proyecto.

Su fórmula es:

$$IR = \frac{\sum_{k=1}^{n} F_k (1 + D_k)^{-k}}{I} \qquad (2.18)$$

Donde:

I = Inversión inicial requerida.

F_k = Flujo de efectivo en el periodo k.

D_k = Tasa de descuento aplicada en el periodo k.

n = Número de periodos de vida del proyecto.

Criterio de decisión para el IR:

- Si el IR > 1, el proyecto se debe aceptar porque significa que los beneficios que genera son superiores a su costo.
- Si el IR < 1, el proyecto se debe rechazar porque significa que los beneficios que genera son inferiores a su costo.
- Si el IR = 1, es indiferente aceptar o rechazar el proyecto porque sus beneficios son exactamente iguales a su costo.

Ventajas del método del IR:

- Considera el valor del dinero en el tiempo.
- Se recomienda usarlo como información complementaria del NPV. Por ejemplo, puede ayudar a seleccionar proyectos con NPV positivo cuando los recursos de inversión son limitados.
- Se recomienda utilizar el IR para comunicar los resultados a posteriori que hacer un análisis a priori del proyecto.

Desventajas del método:

- Es necesario estimar la tasa de descuento apropiada para calcular el valor presente de los flujos de efectivo.
- Puede conducir a decisiones erróneas. Por ejemplo el IR de un proyecto con un VPN alto puede ser inferior al IR de un proyecto con un VPN bajo.
- Este método no es válido para proyectos con flujos de efectivo no convencionales o proyectos en los que la inversión se hace periódicamente y no al inicio.
- No es sensible a la escala y tamaño de los proyectos.
- No es aditivo, esto significa que no se cumple que $IR_{A\,y\,B} = IR_A + IR_B$.
- No tiene ventajas adicionales al NPV.

El criterio de decisión, las ventajas y desventajas del método del IR se obtuvieron de las siguientes referencias: [1, 17, 21].

Ejemplo:

Para ejemplificar que el IR no toma en cuenta la escala y tamaño de los proyectos, se calculó el IR de los proyectos F y G de la tabla 2.2.3.1. Los resultados fueron: $IR_F = 1.82$ y $IR_G=1.36$. Esto sugiere escoger el proyecto F, sin embargo el proyecto G tiene un VPN muy superior al del proyecto F.

2.3 Modelos de decisión

Dentro de esta clasificación se encuentra diferentes modelos de decisión, la mayoría de ellos se basan en datos cualitativos y/o de apreciación. En esta sección se mencionan: modelos de ordenamiento, cheklist, análisis de jerarquías, enfoques de comportamiento, diagramas de burbujas, árboles de decisión, y teoría de la utilidad.

- Modelos de ordenamiento: Asignan a cada proyecto un índice basado en algún criterio. Por ejemplo un índice proporcionado por un conjunto de expertos [5].

- Cheklists: Los proyectos son calificados en base a un cuestionario cualitativo. Las preguntas incluyen aspectos relacionados con las ventajas del nuevo producto, que tan atractivo es el mercado, la sinergia con los objetivos que persigue la empresa, entre otras [11].

- Análisis de jerarquías: Estas son herramientas de decisión basadas en comparaciones apareadas de proyectos y criterios [11].

- Enfoques de comportamiento: Son útiles en etapas tempranas cuando solo se tiene información cualitativa. Incluye métodos como el Delphi y Q-Sort [11].

- Diagramas de burbujas: Varios parámetros son graficados uno contra otro en un formato de diagrama de burbujas. Por ejemplo gráficas de beneficios vs. probabilidad de éxito del proyecto [11].

- Árboles de decisión: Proporciona una forma para desplegar visualmente las decisiones posibles, las probabilidades asociadas y los pagos por elegir una rama del árbol. Se basa en la regla de decisión de Bayes [22].

- Teoría de la utilidad: Incorpora al análisis la actitud del tomador de decisiones frente al riesgo. Tiene la propiedad de que el tomador de decisiones se muestra indiferente entre dos cursos de acción alternativos con misma utilidad esperada [22].

2.4 Trabajos anteriores

Existen muchos métodos y/o modelos para seleccionar proyectos u ordenar alternativas con restricciones de recursos (humanos, financieros y materiales). La mayoría de estos combinan varias de las técnicas descritas en las 3 secciones previas. A continuación se describen a grandes rasgos algunos trabajos publicados relacionados con la selección de proyectos en general y posteriormente se mencionan trabajos relacionados con selección en Ingeniería de Software.

Kim y Emery formulan un modelo binario basado en programación por metas para la selección de proyectos y planeación de recursos en una compañía que fabrica controles de avión. El modelo descrito determina que proyectos deben seleccionarse tal que se maximice la utilidad durante un periodo de 4 años, así mismo el modelo desarrolla los planes de la

maquinaria necesaria y estima los requerimientos del personal. La estructura del modelo tiene 7 metas: 5 de ellas son metas definidas por el administrador de la compañía y las 2 restantes son particulares del modelo. Las 5 metas generales son: asegurar que existirá la capacidad de maquinaria para la producción durante los 4 años, asegurar que existirá capacidad del torno para fabricar los componentes, lograr los niveles deseados de utilidad y minimizar el costo de capital [30].

April, Glover y Kelly proponen un software, llamado Optfolio, para encontrar la combinación óptima de proyectos basado en ideas de programación no lineal y simulación de Monte Carlo. Para ilustrar el funcionamiento de Optfolio, se incluyen ejemplos aplicados a las industrias de energía, farmacéutica y tecnologías de la información. Optfolio permite hacer muchas variantes en el planteamiento de los problemas, tomando como base el seleccionar proyectos que maximicen el VPN sujetos a restricciones de presupuesto y límites de riesgo (percentiles de una distribución de riesgo). Tanto los flujos de efectivo como los límites de riesgo se simulan con Monte Carlo [2].

Dickinson, Thornton y Graves describen un modelo de programación entera no lineal para seleccionar proyectos interrelacionados dentro de la compañía Boeing. El objetivo es maximizar el NPV sujeto a restricciones de costos, riesgo, y balance de los proyectos. Dos de las ventajas de este modelo es que funciona para proyectos interrelacionados, y que los proyectos pueden ser evaluados en diferentes periodos de tiempo. Para encontrar solución optima, se creó una aplicación en *Excel* que despliega gráficas y tablas con la información de salida. Adicionalmente, una vez que el modelo encuentra la estrategia óptima permite cuantificar rápidamente pequeños cambios en la información de entrada y despliega de manera inmediata las consecuencias [13].

Kavadias, Loch, y Tapper presentan un método de selección de proyectos, de procesos y de equipamiento para una compañía de diamantes. El método propuesto sigue 2 pasos. Primero, cada proyecto es representado por un árbol de decisión que refleja la flexibilidad futura ante posibles contingencias, entonces los proyectos se ordenan de acuerdo al beneficio

marginal expresado como el valor de la opción por dólar del presupuesto. Al terminar, se verifica si el ordenamiento satisface ciertas condiciones, si esto ocurre se dice que el ordenamiento es el óptimo y el método termina. Si las condiciones no son satisfechas en su totalidad empieza el segundo paso, este consiste en un modelo de programación entera para lograr un uso eficiente de los recursos con el mayor presupuesto disponible [29].

Radulescu y Radulescu exponen diversas variantes de un modelo de selección de proyectos de programación no lineal binaria. El modelo base es maximizar el desempeño o beneficio de los proyectos y minimizar el riesgo siempre que las restricciones de los recursos sean cumplidas. El beneficio se cuantifica mediante la opinión de un conjunto de expertos, mientras que el riesgo es la varianza de las calificaciones otorgadas al proyecto. Un proyecto con alto grado de riesgo es aquel en que la opinión de los expertos fue muy dispersa. Se encuentra la solución óptima mediante la aplicación de un algoritmo heurístico [32].

Por otra parte, en Ingeniería de Software, la selección de proyectos es una decisión a la que constantemente se enfrentan las empresas que tienen recursos limitados. Sin embargo los modelos encontrados en la literatura actual son similares pero no resuelven precisamente el problema de selección de proyectos de software, si bien es cierto, en algunos casos estos modelos pueden adaptarse. A continuación se mencionan algunos de estos modelos.

Erdogmus plantea una evaluación de estrategias alternativas de desarrollo de software basada en NPV. El método construye una jerarquía de 6 niveles con métricas del impacto del riesgo y la rapidez del desarrollo del software. La principal limitante de este modelo es que es válido solo para 2 alternativas mutuamente excluyentes [14].

Smith, Subramanian, Nauss y Beck encuentran la estrategia óptima para el desarrollo de un nuevo sistema con ayuda de un modelo de programación matemática. Este modelo minimiza los costos de: personal de tiempo completo, personal temporal, outsourcing de ciertas aplicaciones del sistema, y outsourcing para el desarrollo de los proyectos de software. El modelo garantiza que el sistema será implementado y operado en ciertas fechas fijas y

también que se cumplen las restricciones acerca de la disponibilidad del personal y de los costos por conceptos de outsourcing. El modelo es creado y solucionado con el paquete comercial SAS/OR [34].

Otros trabajos relativos a ingeniería de software pero con finalidad diferente a seleccionar proyectos que emplean alguna técnica descrita anteriormente, son:

- Modelo de optimización estocástica para minimizar el desarrollo de tiempo y costo de un proyecto de software dado un staff fijo [31].
- Modelo para determinar el número óptimo del staff para desarrollar un proyecto de software tal que el NPV sea máximo [36].
- Métodos de decisión aplicables al diseño de software. En particular la evaluación analítica de estrategia óptima de diseño tomando en cuenta restricciones económicas [37].
- Diversas problemáticas propias de Ingeniería de Software donde se aplican técnicas de decisión multiobjetivo con restricciones para encontrar una solución [6, 19].

En conclusión, los modelos de selección de proyectos basados en programación matemática son muy populares y generalmente están combinados con algún tipo de modelo financiero. En lo referente a modelos de selección de proyectos de software, no se encontraron modelos que resolvieran este problema de manera específica.

Capítulo 3

El Problema de Selección de Proyectos Candidatos de Software

La problemática descrita en este capítulo surge en una organización desarrolladora de software (ODS) a la medida en particular, pero es general de organizaciones similares.

Los *proyectos candidatos* son los proyectos de software que aún se está evaluando la factibilidad de llevarlos a cabo, por lo tanto están en la etapa de negociación y definición con el cliente. En algunas ocasiones la ODS no tiene los recursos humanos suficientes para tomar todos los proyectos candidatos y debe de seleccionar un subconjunto de proyectos de acuerdo a sus limitantes de recursos. El problema es establecer un criterio para seleccionar proyectos candidatos de software que considere el mayor número posible de los siguientes aspectos: ventana de evaluación común, utilidad proporcionada, impacto en el flujo de efectivo, disponibilidad de recursos, optimización de recursos, información escasa y cambiante, estimación de cierta información, evaluación de proyectos heterogéneos, incertidumbre y rapidez en la selección.

El contenido de este capítulo está organizado en 3 secciones: contexto del problema, aspectos a considerar en la selección de proyectos de software, y descripción del problema. En el contexto del problema se definen a las ODS y a los proyectos de software, y se encuentra una descripción general de la ODS de estudio identificada como QS. En la siguiente sección se discuten cada uno de los aspectos a considerar en la selección de

proyectos de software. Se finaliza este capítulo con la descripción del problema, se mencionan los objetivos y motivación de esta tesis.

3.1 Contexto del problema

3.1.1 Organización desarrolladora de software

A las organizaciones que desarrollan software para otras organizaciones se les conoce como *organizaciones desarrolladoras de software a la medida* (ODS) o empresas de outsourcing. Una ODS desarrolla software que se ajusta a las necesidades del cliente que lo solicita, por lo tanto el software es único y de uso exclusivo para quien contrata este servicio. Adicionalmente una ODS puede ofrecer servicios de consultoría respecto a procesos de software o tecnología. Normalmente una ODS trabaja a través de proyectos de software.

3.1.2 Proyectos de software

Un *proyecto de software* es tipo de proyecto especial que cumple con las características mencionadas al inicio del capítulo 2. Se compone de procesos que realizan los recursos humanos con ayuda de la tecnología para producir artefactos de software. Ejemplos de artefactos de software son: el software construido, manuales de usuario, documento de especificación de requerimientos, documento de diseño, entre otros.

En términos generales hay 3 procesos sucesivos que sigue un proyecto de software dentro de una ODS: el estudio de factibilidad, la planeación y la ejecución del proyecto [23].

- Estudio de factibilidad: Es una evaluación del proyecto de software que ayuda a determinar si es viable realizar el proyecto. La evaluación puede comprender tres partes: técnica, económica y estratégica. La evaluación técnica es donde se verifica la disponibilidad de los recursos humanos y tecnológicos para realizar exitosamente el proyecto. La evaluación económica básicamente es un análisis de

costo-beneficio y un pronóstico del flujo de efectivo. Y finalmente se realiza la evaluación estratégica para detectar beneficios y/o perjuicios futuros a la compañía de manera subjetiva, como por ejemplo mejorar la imagen corporativa, aumentar la cartera de clientes, o proyectos que no son redituables pero son entrada a proyectos futuros.

- Planeación del proyecto: Se inicia cuando el cliente acepta la propuesta. La planeación consiste en describir las actividades necesarias para crear los artefactos de software derivados del proyecto. Según el Project Management Institute (PMI) la planificación de proyectos es "la invención y mantenimiento de un esquema de trabajo para ayudar a conseguir los objetivos que motivan la realización del proyecto de software" [35].

- Ejecución del proyecto de software: Se refiere a la ejecución del plan. Es el proceso más largo y se divide en fases de acuerdo al ciclo de vida del proyecto. El ciclo de vida típico de un proyecto de software es el de cascada. Este se basa en el desarrollo secuencial de las siguientes fases: análisis de requerimientos, diseño, codificación, verificación y validación e implantación/instalación (figura 3.1). Existen otros ciclos de vida utilizados como el espiral o el incremental.

Figura 3.1 Modelo de cascada.

Otra forma de clasificar a los procesos que sigue un proyecto de software es el que especifica ISO 12207: adquisición, oferta, desarrollo, operación y mantenimiento. Se considera que los 3 primeros procesos de esta clasificación forman el ciclo de vida de un proyecto.

3.1.3 Tipos de proyectos de software

Una manera de clasificar a los proyectos de software es separar los proyectos de software que se encuentran bajo contrato de aquellos que aun están en proceso de negociación con el cliente. Bajo este contexto, se identifican dos tipos de proyectos:

a) Proyectos en curso

Son los proyectos en los cuales ya se firmó el contrato, pueden estar en cualquier fase de desarrollo (v.gr.: requerimientos, diseño) o podrían no haber empezado aún pero ya se conoce la fecha de inicio. Los proyectos en curso consumen, comparten y liberan recursos durante cada fase de desarrollo. Para cualquier proyecto en curso, la empresa conoce con certeza la información contenida en la tabla 3.1. Esta información es la que permite caracterizar a los proyectos desde el punto de vista económico.

Dominio	Área de conocimiento. Ejemplos de dominios: contabilidad, teoría de inventarios, y seguros.
Tiempo de desarrollo	Tiempo total estimado en horas para desarrollar el proyecto.
Recursos humanos	Número de ingenieros de software clasificados de acuerdo a las habilidades y conocimientos necesarios para realizar las actividades correspondientes al proyecto. Este número puede cambiar de acuerdo a la fase o ciclo en que se encuentre el proyecto de software.
Duración del proyecto	Fecha de inicio y fecha de finalización.
Costo total del proyecto	Costo total del proyecto para la ODS y calendario tentativo de gastos de la empresa ocasionados directamente por este proyecto incluyendo aquellos gastos que son extemporáneos, en el caso de que se tenga crédito con algunos proveedores.
Precio del proyecto	Costo total del proyecto a pagar por el cliente.
Esquema de pagos	El esquema de pagos se determina en el contrato y usualmente se basa en condiciones impuestas por el cliente. Un ejemplo de un esquema de pagos es cuando el precio del proyecto se divide en pagos iguales.

Tabla 3.1. Información en común de los proyectos en curso.

b) Proyectos candidatos

Son los proyectos que se encuentran en la etapa del estudio de factibilidad. Cuando el cliente acepta la propuesta, se formalizan las negociaciones a través de un contrato o de manera verbal si hay mucha confianza y el proyecto cambia su estado a proyecto en curso.

3.1.4 Empresa de estudio

Este trabajo de tesis se basa en un problema real de una organización desarrolladora de software a la medida establecida en México que se conoce como QS.

QS es una empresa mexicana que sigue una filosofía de calidad en todas sus operaciones por medio de control estadístico y mejora continua de procesos. Actualmente emplea la familia de modelos CMM® como marco de referencia para sus operaciones y en particular, el Team Software Process^{SM} (TSP^{SM}) y el Personal Software Process^{SM} (PSP^{SM}) como los procesos de implementación para SW-CMM. Las plataformas de desarrollo son: J2EE, Progress,4J's, C++ y Visual Basic. La tabla 3.2 muestra ejemplos de tecnología que QS utiliza en el desarrollo de sus proyectos.

J2EE	PROGRESS
▪ JSP ▪ Servlets ▪ EJB ▪ Web Services ▪ .NET	▪ Web Speed ▪ Cliente Servidor ▪ ADM2 ▪ Open Client Tool Kit ▪ Web Client

Tabla 3.2 Ejemplo de tecnologías para el desarrollo de proyectos de software

La mayor parte de los ingresos de QS, al igual que otras organizaciones desarrolladoras de software, proviene de los proyectos software. En QS todos los proyectos de software se apoyan en un contrato de trabajo realizado entre la compañía y el cliente. Los clientes de QS son empresas privadas y/o gobierno.

QS tiene 30 ingenieros de software contratados de manera permanente consecuentemente el pago de la nómina de sus ingenieros de software es el gasto mayor y representa alrededor del 70% de los gastos mensuales totales.

Para QS una prioridad de negocios es generar riqueza captando nuevos proyectos que le resulten rentables. Sin embargo, en algunas ocasiones QS no tiene los recursos suficientes para aceptar todos los proyectos en curso, entonces debe de seleccionar un portafolio de proyectos candidatos antes de presentar las propuestas al cliente.

3.2 Selección de proyectos de software

Seleccionar proyectos de software no solo implica elegir una combinación de proyectos que logre un balance entre el número de proyectos y los recursos disponibles, si no que engloba diversos aspectos a considerar como: ventana de evaluación común, utilidad proporcionada, impacto en el flujo de efectivo, disponibilidad de recursos, optimización de recursos, información escasa y cambiante, estimación de cierta información, evaluación de proyectos heterogéneos, incertidumbre y rapidez en la selección. A continuación se describe la importancia de estos factores en la selección de proyectos de software para QS.

1) Ventana de evaluación

Cada proyecto de software tiene una vida delimitada por su fecha de inicio y su fecha de finalización. En otras palabras, la *vida de un proyecto* es la amplitud del proyecto en el tiempo. Cuando se hacen evaluaciones económicas de un conjunto de proyectos es altamente deseable que estas se realicen en un periodo común de tiempo. Ese periodo se denomina *horizonte de planeación* o *ventana de evaluación.* Tener la misma ventana de evaluación para todos los proyectos permite comparar proyectos con diferentes tamaños de vida.

Debido a que un proyecto de software puede tener distintos requerimientos de trabajo durante toda su vida, se vuelve necesario discretizar la ventana de evaluación para tomar en cuenta estos cambios. Lo mas recomendables es elegir intervalos de tiempo pequeños como: una semana, una quincena o a lo más un mes. Aunque en realidad el tamaño de los intervalos de tiempo depende en gran medida en cómo se hace la planeación de los proyectos. Por ejemplo, en QS toda la planeación se basa en semanas.

En resumen, se deben de tomar dos decisiones relativas a la ventana de evaluación: como escoger el tamaño de la ventana de evaluación y el tamaño de los intervalos de tiempo en que se divide la ventana de evaluación.

2) Utilidad

Maximizar la utilidad, es una meta típica de negocios ya que el objetivo de QS es la generación de riqueza a través del servicio de outsourcing de desarrollo de software. Para cuantificar la utilidad de un conjunto de proyectos, es necesario contar con información de los ingresos y gastos derivados de cada proyecto.

3) Flujo de efectivo

El *flujo de efectivo* es igual a los ingresos menos los egresos producidos por un conjunto de proyectos durante un periodo determinado.

Para cuantificar los ingresos es necesario conocer el esquema de pagos de cada proyecto. El *esquema de pagos* especifica la manera (efectivo, cheque o depósito), la cantidad, las fechas y las condiciones del pago del precio total del proyecto por el cliente. Este esquema generalmente se determina en el contrato y usualmente se basa en condiciones impuestas por el cliente. Por lo tanto, pueden existir tantas formas de pago como clientes. Un criterio común es recibir un pago cada vez que se termina una etapa del proyecto y se entregan los

artefactos de software producidos durante esa etapa. Otro esquema es dividir el precio total en pagos mensuales iguales.

Algunos costos que se toman en cuenta para cuantificar los egresos derivados de los proyectos son los costos operativos y el costo de la inversión inicial incluyendo aquellos gastos que son extemporáneos. Ejemplos de costos operativos: honorarios de los ingenieros de software, viáticos, papelería. Ejemplos de costos de inversión inicial: capacitación, licencias de software, manuales, asesoría técnica, equipo, y mobiliario.

Es importante que QS mantenga un flujo de efectivo positivo para prevenir la escasez de efectivo por periodos prolongados o al menos saber que si se tendrá el dinero necesario disponible (liquidez) para afrontar los gastos en el corto plazo.

Dado que un proyecto tiene un esquema de pagos diferentes, los ingresos que recibe QS no son constantes a través del tiempo, en cambio la mayor parte de sus gastos son periódicos y fijos, por ejemplo el pago de la nómina. Por eso, al seleccionar proyectos se debe de verificar que la combinación de proyectos seleccionados no provoque una escasez de efectivo de manera temporal (podría ser durante un mes o mas), ya que QS no tendrá efectivo para cumplir con sus obligaciones. Otra alternativa para combatir la escasez de efectivo es pedir un préstamo, pero QS tendrá que ver disminuidas sus ganancias al pagar los intereses sobre el monto solicitado.

4) Proyectos heterogéneos

Debido a la diversidad de necesidades y características que puede tener un cliente, es común que una ODS tenga proyectos de software heterogéneos.

Las organizaciones como QS manejan diversos proyectos de software al mismo tiempo. Estos proyectos tienen diferentes fechas de inicio y de término, tecnología, tamaño, calendarización de tareas y estrategia de desarrollo entre otras características. Por esta razón

cada proyecto tiene requerimientos de recurso humano diferentes e inclusive no son requerimientos constantes, ya que algunos proyectos de software necesitan diferente número de ingenieros de acuerdo a la etapa en que se encuentre el proyecto.

El problema de tener proyectos heterogéneos es que resulta difícil hacer comparaciones entre ellos, por eso es necesario establecer indicadores que sean sensibles a las variaciones en las características de los proyectos.

5) Disponibilidad de recursos humanos

Cualquier ODS que tenga recursos humanos finitos tiene que cerciorarse de que cuenta con los recursos humanos necesarios antes de presentar una propuesta al cliente. Garantizar que se tendrán los recursos humanos en tiempo y forma para el desarrollo de los proyectos, equivale a tener la gente correcta en el tiempo correcto y es un factor de éxito para realizar el proyecto. Esta verificación no debe de olvidar que los proyectos en curso consumen, comparten y liberan recursos debido a que los requerimientos de recursos humanos son distintos a través del tiempo dependiendo de la fase de desarrollo del proyecto y se hacen en base al tipo de conocimiento que se requiere para el proyecto. Ejemplos de tipos de conocimientos son: tipo de tecnología, tipo de actividad, y tipo de lenguaje de programación.

Una consideración importante es que durante la etapa de factibilidad no es necesario asignar a los ingenieros que trabajarán en los futuros proyectos en curso, basta con cerciorarse que se tendrán los recursos necesarios en tiempo y forma, lo que si es relevante es establecer restricciones para que los ingenieros que dominan más de un tipo de conocimientos no sean contados como dos o más ingenieros disponibles.

6) Optimización de recursos

Para una ODS su principal recurso son los ingenieros de software. En QS los ingenieros de software están capacitados para trabajar prácticamente en cualquier etapa del proyecto de

software siempre y cuando dominen el tipo de conocimiento que requiere el proyecto en un periodo determinado. Así, cada ingeniero puede trabajar durante un periodo de tiempo en un proyecto y al término de este puede incorporarse a otro proyecto.

Una forma de optimizar el uso de los recursos humanos es maximizar el uso de las horas disponibles, ya que cada hora desocupada significa un costo real para QS (principalmente costo de nómina) y no genera ingreso. Por otro lado, el uso de las horas de los ingenieros de software es "limitada". Por ejemplo cada ingeniero tiene un horario normal de 40 horas a la semana y a lo mas puede trabajar 48 horas incluyendo tiempo extra. A pesar de que existe la posibilidad de asignar tiempo extra a los ingenieros, hacerlo constantemente supone fatiga para los ingenieros, por eso es importante no abusar de esta holgura. Así que en este contexto, la expresión "maximizar el uso de las horas disponibles de los ingenieros de software", significa que se desea ocupar por completo las 40 horas a la semana, aunque algunas veces podría asignarse tiempo extra.

7) Información escasa y cambiante

La información disponible relativa a los proyectos candidatos varía a lo largo de las negociaciones e incluso puede ir cambiando día a día.

La información que se maneja durante el estudio de factibilidad es:

- El dominio del proyecto
- Los lenguajes y herramientas de programación
- El número promedio de ingenieros y las habilidades que se requieren
- Duración del proyecto en semanas
- El costo total del proyecto
- Esquema probable de pagos del proyecto

Generalmente, si hay cambios en un dato, este impacta en el valor de los demás.

8) Rapidez en la selección de proyectos

Al ir aumentado las negociaciones con los diversos clientes, la información cambia frecuentemente y se necesita "evaluar" rápidamente el nuevo escenario. Por ejemplo, QS realiza el análisis de factibilidad de un proyecto candidato y presenta la propuesta de trabajo y económica al cliente, pero éste llama por teléfono y pregunta en cuanto se incrementaría el precio si el tiempo de entrega del proyecto se adelanta un mes. Entonces, QS no sólo debe de estimar el nuevo precio, también requiere estimar el incremento en el número de ingenieros y verificar la disponibilidad de recursos en tiempo y forma, entre otras cosas. Esto equivaldría a que QS tiene que hacer una evaluación completa de todos sus proyectos y cerciorarse que los cambios no impactan negativamente, para poder continuar con las negociaciones y contestarle al cliente cual es el nuevo precio.

En conclusión, la selección de proyectos debe de permitir cambios en la información en todo momento, ser rápida, fácil de usar y entendible.

9) Incertidumbre

En QS usualmente son más los proyectos candidatos que nunca llegan a ser proyectos en curso que los que si lo logran. Esto se debe a que algunas veces al realizar el estudio de factibilidad QS determina que no le es viable realizar el proyecto o bien por que el cliente no acepta la propuesta.

Todo esto quiere decir que para cada proyecto candidato se tiene un nivel de incertidumbre que debe tomarse en cuenta al momento de seleccionar los proyectos, mas aún, se puede tener un proyecto candidato que tenga una alta probabilidad de que sea aceptado tanto por la empresa y por el cliente, pero que tenga alta incertidumbre en cierta información, como por ejemplo el esquema de pagos.

10) Estimación

Para poder caracterizar a un proyecto candidato es necesario conocer ciertos atributos cuantificables a través de métodos de estimación y/o predicción.

La estimación se inicia desde el mismo momento que el cliente hace contacto con QS, ya que el cliente proporciona algunas características generales del software que desea. Es muy común que la información proporcionada por el cliente sobre el sistema requerido sea parcial y muy general, no obstante QS debe estimar el tamaño y la duración en horas del proyecto de software candidato.

Los métodos de estimación deben estar perfectamente aplicados y validados para cada empresa en particular.

3.3 Descripción del problema

3.3.1 El problema

QS requiere captar nuevos proyectos para subsistir y/o crecer. Sin embargo, como sus recursos son finitos, no tiene la capacidad de aceptar todos los nuevos proyectos, así que tiene que decidir con muy poca información que nuevos proyectos elegir.

El problema que resuelve esta tesis es como elegir de manera óptima los proyectos candidatos que resulten de mayor rentabilidad económica optimizando el tiempo de asignación de los ingenieros de software y garantizando el flujo de efectivo de la compañía. Y que además sea una forma fácil y práctica de implementar.

Esto es por que desde el punto de vista de negocios, QS funciona similar a una empresa de servicios debido a que:

- Busca generar la mayor riqueza posible captando nuevos proyectos, ya que su mayor fuente de ingresos es por el servicio que ofrece (desarrollo de proyectos).
- Necesita optimizar el uso de sus recursos más importantes que son los ingenieros de software.
- Requiere mantener un flujo de efectivo positivo periódicamente para poder afrontar sus obligaciones recurrentes, como es el pago de la nómina que constituye la mayor fuente de sus egresos.

3.3.2 Objetivos

Este trabajo de tesis propone una solución para la problemática de seleccionar proyectos en QS y para una ODS en general por medio del cumplimiento de tres objetivos:

1) Plantear un modelo matemático para la selección de proyectos candidatos de software que sea válido para una ODS en general y que tome en cuenta el mayor número posible de los siguientes aspectos: ventana de evaluación común, utilidad proporcionada, flujo de efectivo, disponibilidad de recursos, optimización de recursos, información variable, cambiante y escasa, estimación de información, proyectos heterogéneos, e incertidumbre. Además todos los valores de salida deben de ser equiparables.

2) Construir una herramienta tipo DSS (Decision Support System) basada en el modelo matemático que sea fácil de usar e interpretar por una persona que no necesariamente conoce la estructura del modelo matemático, que presente los resultados de forma grafica, que obtenga la información de entrada de manera automática de una base de datos, y que permita la re-evaluación de un portafolio de proyectos de manera instantánea si hay cambios en la información de entrada.

3) Implementar la solución a la empresa QS.

3.3.3 Motivación

Entre las principales motivaciones para realizar este trabajo de tesis se encuentran: aplicar los conocimientos adquiridos para resolver una problemática real, la complejidad del problema que engloba las áreas: Ingeniería de Software, Investigación de Operaciones, evaluación de proyectos, Ingeniería Económica, y programación, y, que la solución propuesta será implantada en una empresa y proporcionará beneficios directos al momento de la toma de decisiones.

Adicionalmente, no se encontró una solución satisfactoria del problema de selección de proyectos en la literatura actual dentro del área de ingeniera de software. Si bien es cierto que la mayoría de los modelos y métodos para seleccionar proyectos se pueden aplicar a diferentes dominios, el principal reto de este trabajo de tesis es adaptar un modelo que capture la problemática descrita.

Capítulo 4

Modelo de Optimización

Este capítulo corresponde al planteamiento de un modelo matemático como parte de la solución al problema de seleccionar proyectos candidatos de software descrito en el capítulo anterior. El primer acercamiento es plantear el problema como un modelo de optimización multi-objetivo, ya que se quiere maximizar la utilidad proporcionada y a la vez maximizar el uso de los ingenieros de software, sujeto a dos restricciones: garantizar el flujo de efectivo y disponibilidad de los ingenieros.

Para maximizar la utilidad se utilizó la función del valor presente neto (VPN) de los futuros flujos de efectivo. Esta técnica financiera es muy popular dentro de la evaluación de proyectos ya que toma en cuenta el valor del dinero a través del tiempo para predecir la utilidad en pesos del día de hoy. Calcular la utilidad de esta manera la hace comparable con la utilidad de otros proyectos a pesar de que tengan tamaños de vida diferentes [14]. Además se incorpora la restricción de garantizar el flujo de efectivo.

El segundo criterio es maximizar el uso de los ingenieros de software, para lograr esto se propone un modelo basado en programación por metas. Este modelo establece metas periódicas acerca del número de horas de trabajo de los ingenieros y penaliza las desviaciones a la meta (número en horas por abajo y/o por arriba de la meta) con costos traídos a valor presente. Es en este modelo donde se verifica que se cumplan las restricciones de disponibilidad de los ingenieros.

La propuesta final para seleccionar proyectos es combinar los dos modelos mencionados arriba en uno solo y tener una función mono-objetivo que incorpora muchos de los aspectos deseados. Haciendo referencia a los tipos de modelos de selección mencionados en el capítulo 2, este modelo es una combinación de un modelo de programación matemática con un modelo de evaluación financiera.

Hay que resaltar que una aportación importante de este trabajo de tesis es en sí el planteamiento del problema en un modelo de optimización binaria con restricciones que no está limitado a un solo algoritmo de solución, ya que pueden aplicarse diversos algoritmos reportados en la literatura para modelos binarios.

Este capítulo está organizado en 2 secciones. La primer parte corresponde a la formulación matemática del modelo e incluye 6 apartados: información de entrada, ventana de evaluación, variables de decisión, modelando la utilidad, optimización de recursos, restricciones y el modelo propuesto. La segunda y última parte es acerca de los métodos de solución del modelo.

4.1 Formulación del modelo

4.1.1 Ventana de evaluación

El método usado en esta tesis para determinar el tamaño de la ventana de evaluación es el de la vida mas larga. Este método fija el inicio de la ventana de evaluación en la fecha actual y termina en la fecha de finalización del proyecto más largo en el eje del tiempo. La ventana de evaluación se representa por el intervalo de tiempo $I=[I_0,I_n]$. La figura 4.1 ilustra con ayuda de un diagrama de Gantt como se obtiene este intervalo.

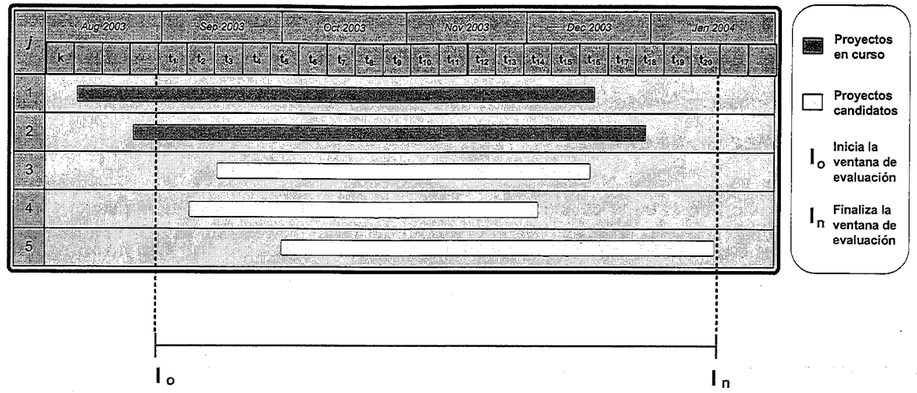


Figura 4.1. Ejemplo de cómo se obtiene la ventana de evaluación.

Para discretizar la ventana de evaluación, el intervalo I se divide en n periodos de tiempo iguales. Si r es el tamaño del periodo de tiempo, entonces los intervalos tienen la siguiente forma: $[I_0,I_0+r],[I_0+r,I_0+2r],...,[I_0+(n-1)r,I_n]$ donde $I_n=(I_0+nr)$. Ahora bien, si etiquetamos a cada periodo con un número natural de acuerdo a la secuencia en el tiempo que siguen, comenzando con 1 y finalizando con n, entonces se forma el conjunto $T=\{t_1,t_2,...,t_n \mid n\geq 2\}$ que contiene a todos los periodos de tiempo en que se dividió la ventana de evaluación. El ejemplo de la figura 5.1 secciona la ventana de evaluación en semanas, en caso de que el intervalo I no sea divisible entre un número entero de semanas es necesario hacer una corrección a I_o y I_n.

4.1.2 Variables de decisión

Sea $P=\{1,...,m\}$ el conjunto de proyectos de software a incluirse en el modelo. Se define a x_p como la variable de decisión individual $\forall p\in P$ tal que:

$$x_p=\begin{cases}1 & \text{si el proyecto } p \text{ es seleccionado}\\ 0 & \text{si el proyecto } p \text{ no es seleccionado}\end{cases}$$

Ahora bien, sea $\bar{x} = (x_1, x_2, ..., x_m)$ el vector que contiene el conjunto de decisiones acerca de seleccionar o no cada proyecto de software tal que $p \in P$.

4.1.3 Información de entrada

La información de entrada es acerca de los proyectos de software y ayuda a establecer los valores de los coeficientes en el modelo de optimización. Se asume que esta información es conocida y/o estimada por la empresa.

Hay que resaltar que la información de entrada debe organizarse de la misma manera en que se dividió la ventana de evaluación.

La información necesaria de cada proyecto de software es:

- Calendarización de las horas necesarias de trabajo por tipo de conocimiento.
- Calendarización de los pagos (cuando y cuanto).
- Calendarización de los costos (cuando y cuanto).

4.1.4 Modelando la utilidad

Como ya se ha mencionado antes en el capítulo 2, el valor presente neto es una forma de medir la utilidad futura, pero en valor de los pesos del día de hoy [21]. Adaptando un poco la notación a la nomenclatura del modelo, la fórmula del valor presente neto (2.15) cambia a:

$$VPN(\bar{x}) = \sum_{t_k \in T} F(\bar{x}, t_k)(1 + D_k)^{-k} \qquad (4.1)$$

Donde:

$VPN(\bar{x})$	=	Valor presente del flujo de efectivo al inicio del periodo t_1.
$F(\bar{x}, t_k)$	=	Valor futuro del flujo de efectivo recibido al inicio del periodo t_k.
D_k	=	Tasa de descuento aplicada para el periodo t_k.
k	=	Número de periodo en el que se recibe el flujo de efectivo

La fórmula (4.1) no menciona la inversión inicial ni tampoco el valor que existe en caja para cumplir las obligaciones recientes de la empresa. Esto es por que se asume que $F(\bar{x}, t_1)$ incluye esas cuentas.

Las razones por las que se utilizó VPN en comparación con otras técnicas financieras [4, 1, 21, 17], son:

- El VPN toma en cuenta el valor del dinero en el tiempo y todos los flujos de efectivo que se espera genere el proyecto
- El VPN vincula las decisiones de aceptación y rechazo de los proyectos con la maximización del valor de la empresa.
- El VPN produce criterios para tomar decisiones cuando se trata de evaluar dos o más proyectos mutuamente independientes.
- El VPN es un método muy conocido en la evaluación financiera de proyectos.
- El VPN es sensible a la escala y tamaño de los proyectos.

4.1.5 Optimización de recursos

El modelo propuesto en esta tesis, además de utilizar el VPN para maximizar la utilidad, se basa en ideas de programación por metas (goal programming) para optimizar el uso de los recursos.

Se utilizó programación por metas porque al establecer metas acerca de las horas de trabajo se cuantifica el número de horas no ocupadas o extras en relación a la meta. Suponiendo que los pesos de penalización se toman como costos, utilizar un planteamiento de programación por metas proporciona la respuesta óptima en base al costo del total de horas no ocupadas y el costo total de horas extras durante la ventana de evaluación.

Definición de las metas

La meta relativa al uso de los ingenieros se expresa en horas totales de trabajo sin tomar en cuenta el tipo de conocimiento requerido por los proyectos. Se establece una meta $\forall t_k \in T$. Por ejemplo, si la empresa tiene 30 ingenieros de software y estos trabajan 40 horas a la semana, entonces la meta es que el trabajo requerido por todos los proyectos sea igual a 1200 horas a la semana. Esto es:

$$N_1(\bar{x},t_k)=H_1(t_k), \quad \forall t_k \in T \tag{4.2}$$

Donde:

$N_1(\bar{x},t_k)$ = Horas de trabajo totales necesarias para los proyectos seleccionados $\bar{x}$ en el periodo t_k.

$H_1(t_k)$ = Horas disponibles del total de ingenieros en el periodo t_k. Aquí se considera una carga normal de horas disponibles. Para QS son 40 horas a la semana por cada ingeniero.

Generalmente $H_1(t_k)$ puede permanecer constante para cualquier t_k, excepto cuando se sabe que algún ingeniero de software saldrá de vacaciones. En el ejemplo de los 30 ingenieros que trabajan 40 horas a la semana $H_1(t_k)=1200$.

Variables auxiliares

La variable que cuantifica la distancia a la que se encuentra la meta se conoce en inglés como la "deviational variable". En este caso está definida por:

$$d_k = N_1(\bar{x},t_k)-H_1(t_k) \tag{4.3}$$

Si ocurre que $d_k = 0$, entonces la meta ha sido alcanzada, pero si $d_k < 0$ entonces la meta no fue alcanzada y en consecuencia existen horas de trabajo que no están siendo

ocupadas. En el caso de que $d_k > 0$, la meta fue sobrepasada y será necesario emplear horas extras de trabajo.

Cabe mencionar que si $d_k < 0$ no puede ocurrir $d_k > 0$ en el mismo periodo t_k. Para poder distinguir que es lo que sucede en cada periodo, definimos las siguientes variables auxiliares:

$$d_k^+ = \begin{cases} d_k & \text{si } d_k > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{4.4}$$

$$d_k^- = \begin{cases} |d_k| & \text{si } d_k < 0 \\ 0 & \text{en otro caso} \end{cases} \tag{4.5}$$

Bajo esas definiciones, las variables (4.4) y (4.5) satisfacen lo siguiente:

$$d_k = d_k^+ - d_k^- \tag{4.6}$$

$$d_k^+ \geq 0, d_k^- \geq 0 \tag{4.7}$$

d_k^+ y d_k^- representan la parte positiva y negativa de la variable d_k. En otras palabras, d_k^+ cuantifica el número de horas extras necesarias en el periodo t_k y d_k^- cuantifica el número de horas no ocupadas durante el periodo t_k. En consecuencia para un mismo periodo, por lo menos una de las dos será mayor a cero, pero nunca ambas al mismo tiempo.

Finalmente, la meta expresada en (4.2) es rescrita como:

$$N_1(\bar{x},t_k)+d_k^- - d_k^+ = H_1(t_k) \quad \forall t_k \in T \tag{4.8}$$

Pesos de penalización

Los pesos de penalización se utilizan cuando la meta no es alcanzada. En este modelo los pesos representan costos e incorporan los efectos del tiempo aplicando una tasa de descuento, similar a como se calcula el VPN.

Sea c_1 el costo por hora extra y c_2 el costo de oportunidad por cada hora no ocupada. En este contexto el costo de oportunidad equivale a la tarifa que hubiera pagado un cliente por cada hora que se tiene desocupada.

Una propuesta para formar los pesos de penalización es la siguiente:

$$w_k^+ = c_1(1+D_k)^{-k} \quad (4.9)$$

$$w_k^- = c_2(1+D_k)^{-k} \quad (4.10)$$

Donde:

w_k^+ = Peso de penalización aplicado cuando la meta es excedida.
w_k^- = Peso de penalización aplicado cuando la meta no es alcanzada.
D_k = Tasa de descuento aplicada para el periodo t_k.
k = Número de periodo en el que se recibe el flujo de efectivo

Cabe destacar que los pesos definidos de esta manera producen una función de penalización como la mostrada en la figura 4.2:

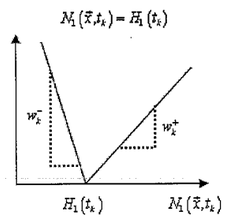


Figura 4.2 Función de penalización.

Costos

Finalmente, la manera de obtener los costos por horas no ocupadas y por horas extras durante todo la ventana de evaluación es mediante las ecuaciones 4.11 y 4.12.

$$\text{costo horas extras}(\bar{x}) = \sum_{t_k \in T} w_k^+ d_k^+ \quad (4.11)$$

$$\text{costo horas no ocupadas}(\bar{x}) = \sum_{t_k \in T} w_k^- d_k^- \quad (4.12)$$

Entonces, para optimizar el uso de los recursos humanos es necesario minimizar estos costos. Hay que destacar que ambos costos están en valor presente neto, así que son directamente comparables con el NPV de la ecuación 4.1.

4.1.6 Restricciones

Las restricciones generales del modelo son:

a) Los proyectos seleccionados deben garantizar que el flujo de efectivo será positivo durante cierto periodo de tiempo
b) Los ingenieros de software necesarios para el desarrollo de los proyectos estén disponibles en las fechas requeridas y tengan los conocimientos solicitados.

En este apartado estas restricciones se traducen en ecuaciones, para posteriormente incorporarlas en el modelo propuesto para seleccionar proyectos de software.

Garantizar el flujo de efectivo

Que el flujo de efectivo sea positivo durante cierto periodo de tiempo, es equivalente a que la empresa podrá afrontar sus obligaciones sin tener que pedir préstamos o realizar

inyecciones de capital, lo cual significa que los ingresos de los proyectos serán suficientes para cubrir sus gastos.

Esta restricción se expresa como:

$$\mathrm{CF}(\bar{x},t_k) \geq 0 \quad \forall t_k \in T \tag{4.13}$$

Y su fórmula es:

$$\mathrm{CF}(\bar{x},t_k) = \mathrm{CF}(\bar{x},t_{k-1}) + F(\bar{x},t_k)(1+D_k)^{-k} \tag{4.14}$$

Donde:

$\mathrm{CF}(\bar{x},t_k)$ = Flujo de efectivo acumulado desde el periodo t_1 hasta t_k.

$\mathrm{CF}(\bar{x},t_0)$ = Es el efectivo que tiene la empresa al inicio de la ventana de evaluación, por ejemplo dinero en caja o bien en el banco. Es la misma para cualquier $\bar{x}$.

$F(\bar{x},t_k)$ = Flujo de efectivo durante el periodo t_k, es igual a los ingresos menos los egresos correspondientes a $\bar{x}$.

D_k = Tasa de descuento aplicada para el periodo t_k.

k = Número de periodo.

Disponibilidad de recursos

Antes de plantear las restricciones impuestas por la disponibilidad de recursos, se mencionan los supuestos en los que estas se basan:

- Los ingenieros son clasificados por tipo de conocimientos.
- Cada proyecto requiere cierto número de horas de trabajo por tipo de conocimiento.
- Un ingeniero puede trabajar en un proyecto solo si domina el tipo de conocimiento que requiere el proyecto en un periodo t_k determinado.
- Cada ingeniero puede trabajar al menos durante un periodo en un proyecto y al término de este puede incorporarse a otro proyecto.

- No se es posible obtener un mayor número de ingenieros de software del que trabaja en la compañía
- Un ingeniero de software no puede exceder 48 horas de trabajo a la semana.

Sea L el conjunto de conocimientos que se requiere para desarrollar un proyecto de software. Un ejemplo de un criterio para formar L es el usado por QS. Esta empresa clasifica a los ingenieros por sus conocimientos de las tecnología de desarrollo, de esta manera L es el conjunto formado por las tecnologías de desarrollo especificadas en la tabla 3.1 (JSP, Servlets, EJB, Web Services, .NET, Web Speed, cliente-servidor, ADM2, Open Client Tool Kit y Web Client). Ahora bien, asumiendo que todos los ingenieros de software que dominan JSP también dominan Servlets y EJB, no es necesario tener 3 distintas categorías bastaría con tener solo una, ya que la finalidad es clasificar a los ingenieros de acuerdo a sus conocimientos.

Sea S el conjunto de todos los subconjuntos de L excepto el conjunto vacío, construir este conjunto ayuda a establecer restricciones para que los ingenieros que dominan más de un tipo de conocimientos de L no sean contados como dos o más ingenieros. Si la cardinalidad de L es q, entonces la cardinalidad de S es $2^q - 1$. Por ejemplo, si $L = \{\text{Java, VB, C}\}$ entonces $S = \{\text{Java, VB, C, Java}\cup\text{VB, Java}\cup\text{C, VB}\cup\text{C, Java}\cup\text{VB}\cup\text{C} \mid S^* \in S\}$.

Para satisfacer las restricciones impuestas por los ingenieros, es necesario verificar todas las $(2^q - 1)$ ecuaciones que se derivan de la siguiente ecuación:

$$N_2(\bar{x},S^*,t_k) \leq H_2(S^*,t_k) \quad \forall S^* \in S \quad \forall t_k \in T \tag{4.15}$$

Donde:

$N_2(\bar{x},S^*,t_k)$ = Horas de trabajo totales necesarias para los proyectos seleccionados $\bar{x}$ en el periodo t_k para el conjunto de conocimientos que comprende S^*.

$H_2(S^*,t_k)$ = Horas disponibles del total de ingenieros en el periodo t_k que dominan

al menos un tipo de conocimiento contenido en S^*. Aquí se considera el máximo número de horas disponibles. Para QS son 48 horas a la semana por cada ingeniero.

4.1.7 El modelo propuesto

Este apartado contiene: los supuestos en que se basa el modelo, la función objetivo, las restricciones del modelo y la nomenclatura empleada en el modelo.

Los supuestos en los que se basa este modelo son:

- Se considera que cada proyecto de software es independiente entre si.
- Se asumen condiciones normales del mercado. Esto es que dos proyectos similares en tamaño y tiempo de desarrollo, tienen un precio cercano o bien se utiliza la misma tarifa para asignarle el precio a ambos proyectos.
- Cada ingeniero de software puede trabajar h horas a la semana sin exceder $h_{\max}$ horas a la semana ($h < h_{\max}$).
- No se es posible obtener un mayor número de ingenieros de software del que trabaja en la compañía.
- Se considera que no es posible obtener inversiones ni prestámos en el corto plazo.
- Cada ingeniero puede trabajar al menos durante un periodo t_k en un proyecto y al término de este puede incorporarse a otro proyecto.
- No se es posible obtener un mayor número de ingenieros de software del que trabaja en la compañía

La función objetivo a maximizar es:

$$\max_{\arg \bar{x}} Z = \text{VPN}(\bar{x}) - \sum_{t_k \in T} w_k^+ d_k^+ - \sum_{t_k \in T} w_k^- d_k^- \qquad (4.11)$$

Sujeta a:

i) Garantizar que el flujo de efectivo sea positivo $\forall t_k \in T$.

$$\text{CF}(\bar{x}, t_k) \geq 0 \quad \forall t_k \in T \qquad (4.13)$$

$$\text{CF}(\bar{x}, t_k) = \text{CF}(\bar{x}, t_{k-1}) + F(\bar{x}, t_k)(1 + D_k)^{-k} \qquad (4.14)$$

ii) Garantizar que existan los recursos humanos con los conocimientos necesarios.

$$N_2(\bar{x}, S^*, t_k) \leq H_2(S^*, t_k), \quad \forall S^* \in S \qquad (4.15)$$

iii) Restricción impuesta por la meta de maximizar el número de horas disponibles de los ingenieros.

$$N_1(\bar{x}, t_k) + d_k^- - d_k^+ = H_1(t_k), \quad \forall t_k \in T \qquad (4.8)$$

iv) Restricciones clásicas de no negatividad.

$$x_p = \{0,1\} \quad \forall p \in P \qquad (4.16)$$

$$d_k^+ \geq 0 \text{ y } d_k^- \geq 0 \quad k = 1, 2, \ldots n, \; n \geq 2 \qquad (4.10)$$

Donde:

$\text{CF}(\bar{x}, t_k)$ = Flujo de efectivo acumulado desde el periodo t_1 hasta t_k.

$\text{CF}(\bar{x}, t_0)$ = Es el efectivo que tiene la empresa al inicio de la ventana de evaluación. Es la misma para cualquier $\bar{x}$.

$F(\bar{x}, t_k)$ = Flujo de efectivo durante el periodo t_k.

$N_2(\bar{x}, S^*, t_k)$ = Horas de trabajo totales necesarias para los proyectos seleccionados $\bar{x}$ en el periodo t_k para el conjunto de conocimientos que comprende S^*.

$H_2(S^*, t_k)$ = Horas disponibles del total de ingenieros en el periodo t_k que dominan al menos un tipo de conocimiento contenido en S^*. Aquí se considera el máximo número de horas disponibles incluyendo horas extras.

$N_1(\bar{x}, t_k)$ = Horas de trabajo totales necesarias para los proyectos seleccionados $\bar{x}$ en el periodo t_k.

$H_1(t_k)$ = Horas disponibles del total de ingenieros en el periodo t_k. Aquí se considera una carga normal de horas disponibles sin incluir horas extras.

4.1.8 Extensión del modelo

El planteamiento del modelo propuesto supone que c_1 y c_2 son constantes, aunque en la práctica estos costos pueden ser obtenidos de una función de costo. Por ejemplo en algunas empresas el pago de horas extras depende del número de horas acumuladas por una persona en una semana. Otro ejemplo es cuando la compañía empieza tener pérdidas económicas por el alto número de horas desocupadas. Este límite se conoce como *punto de equilibrio*.

Una idea puede ser tomar c_1 y/o c_2 de manera tal que el peso de penalización dependa del valor de d_k^- y d_k^+ respectivamente. Ahora bien, los pesos de penalizaciones definidos en las ecuaciones (4.9) y (4.10), pueden ser modificados por:

$$w_k^+ = c_1(d_k^+, t_k)(1+D_k)^{-k} \quad (4.17)$$

$$w_k^- = c_2(d_k^-, t_k)(1+D_k)^{-k} \quad (4.18)$$

Donde:

$c_1(d_k^+, t_k)$ = Función de costo por una hora extra obtenida en base al d_k^+ en el periodo t_k.

$c_2(d_k^-, t_k)$ = Función de costo de oportunidad por una hora desocupada obtenida en base al valor de d_k^- en el periodo t_k.

La dificultad de que los pesos de penalización dependan de una función que a su vez depende de $\bar{x}$ es que el modelo de optimización puede convertirse en un modelo de optimización no lineal. Esto complica el modelo y produce mayores dificultades al momento de buscar un algoritmo que proporcione una solución óptima. Un recurso que se puede optar, es plantear un modelo generalizado de programación por metas similar al que se planteó en la sección 2.1.2 para obtener un modelo lineal, entonces la función de penalización se definen por segmentos.

Un ejemplo de una función de penalización con 4 segmentos es la que se muestra en la figura 4.3:

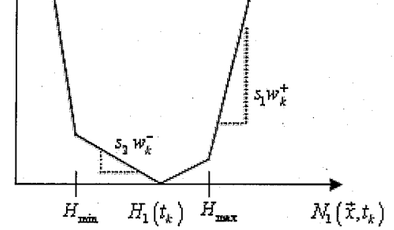


Figura 4.3 Función de penalización de 4 lados.

Donde w_k^+ y w_k^- se definen como en las ecuaciones (4.9) y (4.10) respectivamente. Mientras que s_1 es el coeficiente de penalización del primer nivel y s_2 es el coeficiente del segundo nivel. Usualmente el valor de s_2 es varias veces más grande que el de s_1.

H_{min} es el número mínimo de horas desocupadas en el primer nivel y H_{max} es el número máximo de horas extras en el primer nivel. Un ejemplo para establecer el valor de H_{min} es tomar el punto de equilibrio de la empresa.

Finalmente, una extensión del modelo de optimización propuesto en la sección 4.17 puede ser utilizando la función de penalización mostrada en la figura 4.2, o bien, una función de

penalización con tantos segmentos como sea necesario, aunque esto significa que el número de restricciones y variables auxiliares aumenta.

4.2 Métodos de solución

Tanto el modelo propuesto para seleccionar proyectos de software como el modelo alternativo son modelos binarios y lineales. Debido a que tienen un número finito de posibles soluciones en este tipo de modelos, resulta natural considerar algún tipo de procedimiento de enumeración para encontrar una solución óptima. Uno de los algoritmos más populares de programación entera es la técnica de ramificación y acotamiento (branch-and-bound) y las ideas relacionadas con la enumeración implícita de las soluciones factibles [22].

Entre las ventajas que ofrece la técnica de ramificación y acotamiento se encuentra la flexibilidad que proporciona en cuanto al diseño de un algoritmo específico para que se adapte a la estructura particular de un problema [22]. Actualmente, muchos sistemas comerciales que resuelven problemas de optimización binarios utilizan esta técnica, como por ejemplo LINGO/LINDO y SAS/OR [25, 34, 22]. La técnica de braunch-and-bound es aplicable a problemas enteros en general, sin embargo para problemas que son puramente binarios existen algoritmos especializados derivados de la enumeración implícita [39].

Una propiedad fundamental de las técnicas derivadas de la enumeración implícita es que excluyen un gran número de soluciones, ya que estas técnicas no enumeran en su totalidad a las posibles soluciones, pero aseguran de que entre las soluciones ignoradas no se encuentra la solución optima. Algunos algoritmos basados en enumeración implícita que resuelven problemas binarios fueron propuestos por: Glass, Balas, Glover, Lawler-Bell, Geoffrion, Lemke-Spielberg, Balintfy, y Healy [20].

Existen otros tipos de algoritmos basados en algebra booleana que en la práctica solo funcionan bien si el modelo binario tiene muchas restricciones y pocas soluciones factibles. Williams describe el algoritmo de Granot y Hammer [39]. Desarrollos recientes para resolver problemas binarios de gran escala son los derivados del algoritmo de ramificación y cortadura de Hoffman y Padberg [22].

Por otro lado, hay diversos procedimientos de solución para los modelos binarios de programación por metas, muchos de ellos basados en procedimientos de solución para modelos de programación binaria. Estos procedimientos incluyen también técnicas de branch-and-bound y numeración implícita. Resolver un modelo binario de programación por metas, puede ser significativamente diferente en complejidad y requerir diferente esfuerzo computacional que los procedimientos de branch-and-bound aplicados a modelos de programación entera, incluso para problemas pequeños. Un recurso para disminuir el esfuerzo computacional es creando restricciones adicionales para forzar que la solución sea binaria y tener un menor número de ramificaciones [33].

Otra opción para resolver problemas binarios basados en programación por metas es utilizar algún software comercial. Jablonsky ha publicado algunos artículos donde explica como se hace aplicación sencilla basada en Excel que internamente manda llamar al solver de LINGO para resolver este tipo de problemas [26, 27].

En conclusión, existen numerosos algoritmos para encontrar la solución óptima. Al momento de elegir el algoritmo de solución, se debe tomar en cuenta la dimensión y estructura del modelo binario, así como las ventajas y/o desventajas de cada uno. En el caso de que sean pocos los proyectos a seleccionar, por ejemplo a lo mas 10 proyectos, entonces la enumeración completa es una buena opción [32, 33], ya que es muy sencilla de implementar incluso en hojas de cálculo como *Excel*.

Capítulo 5

Herramienta y Resultados

Al final del capítulo anterior se mencionaron brevemente las opciones que se tienen para solucionar el problema planteado para seleccionar proyectos de software. En este trabajo de tesis en particular, se eligieron dos formas de encontrar la selección óptima de proyectos: la enumeración explícita y la técnica de ramificación y acotamiento. La enumeración explícita se incorporó en una herramienta tipo DSS (Decision System Support) programada en *Visual Basic para Aplicaciones* (VBA) e interfaz en *Excel*, que se nombró RQS ver1.0, mientras que para utilizar la técnica de ramificación y acotamiento se usó el software comercial LINGO.

Además de encontrar la selección optima, la herramienta RQS ofrece muchas ventajas adicionales al tomador de decisiones: alimentación automática de datos, configuración de ciertos valores, resultados gráficos, tablas, listado con los resultados de todas las combinaciones, recalcular inmediatamente si hay cambios en la información, y hace un archivo con los datos de entrada como los requiere LINGO. La principal desventaja de la herramienta es que está limitada a un número máximo de 20 proyectos distribuidos de la siguiente manera: 10 proyectos en curso y 10 proyectos candidatos.

En el caso de LINGO, el número total de proyectos puede ser mayor a 20 y el límite máximo depende de la licencia contratada. La desventaja de emplear LINGO, es que sólo se tendrán resultados para la selección óptima y esto último puede no ser lo más adecuado dentro de una ODS en general, las razones se explicarán mas adelante.

Este capítulo está organizado en 3 secciones: herramienta, resultados y solución con LINGO. Dentro del apartado de herramienta se describe la plataforma, la configuración, los datos de entrada, el procesamiento de los datos, el algoritmo empleado y el tipo de resultados obtenidos. En el apartado de resultados se analizan a 2 ejemplos, las soluciones respectivas y su interpretación, así como la importancia de conocer los resultados de todas las combinaciones. Finalmente, en la solución con LINGO, se explica brevemente como se hace el modelado del problema, se resuelven 2 ejemplos y se verifica que los resultados óptimos son los mismos que RQS.

5.1 Herramienta

Con la finalidad de cumplir con el objetivo 2 de esta tesis mencionado en el apartado 3.3.2, se construyó una pequeña herramienta en VBA y usando *Excel* como interfaz. Se decidió utilizar *Excel* porque:

- Ofrece flexibilidad para manipular los resultados. Por ejemplo, imprimir ciertas tablas.
- La familiaridad de cualquier tomador de decisiones con su manejo, debido a que es una hoja de cálculo muy popular.
- Incluye el editor de VBA, así se pueden realizar modificaciones y/o adecuaciones fácilmente y sin necesitar de un ambiente de desarrollo especial como Java. Inclusive, muchas personas tienen nociones del manejo de macros en VBA sin ser necesariamente programadores.

La herramienta se nombró RQS, está empaquetada en el ambiente de Excel y contiene su propia barra de herramientas. RQS tiene toda la funcionalidad de Excel siempre y cuando la barra de herramientas estándar esté visible. La figura 5.1 muestra la pantalla de inicio de RQS.

Figura 5.1 Pantalla de inicio de RQS.

RQS es una pequeña herramienta tipo DSS (Decision Support System). Un DSS es un sistema que ayuda a las personas a tomar decisiones basado en datos que son tomados de diversas fuentes. Un DSS no es una aplicación de información, como una base de datos acerca de los recursos o un programa que muestra gráficamente una figura con las ventas del mes, es una combinación de diversos recursos trabajando juntos [10]. En los siguientes párrafos se mencionan las características de RQS.

Datos de entrada

Los datos de entrada están conformados principalmente por información de los proyectos en curso y los proyectos candidatos. RQS se alimenta de 3 fuentes distintas: 1) archivos de texto, 2) conexión con una base de datos, y 3) configuración manual de ciertos valores.

Los archivos de texto contienen información sobre los proyectos en curso y los proyectos candidatos. Estos archivos deben seguir un formato específico para que RQS interprete correctamente los datos proporcionados. La información que se requiere por cada proyecto es:

- Nombre o identificador del proyecto
- Fecha de inicio
- Fecha de término
- Tipo de proyecto (candidato o en curso)
- Ingresos y egresos del proyecto por cada semana de vida del proyecto (también pueden ser extemporáneos).
- Horas planeadas por tipo de conocimiento para cada semana de vida del proyecto.

RQS supone que el lenguaje de programación es el tipo de conocimiento requerido por los proyectos. La figura 5.2 muestra un ejemplo de la información de entrada para un proyecto en particular.

Sistema contable PyMES					
Inicia	Termina	Candidato	Horas por Lenguaje		
1-Sep-03	28-Dec-03	Sí			
Semana	Ingresos	Egresos	Total	Progress	J2EE
1-Sep-03	$ 236,600	$ 56,000	160	160	0
8-Sep-03	$ -	$ 3,000	160	160	0
15-Sep-03	$ -	$ 2,000	160	160	0
22-Sep-03	$ -	$ 3,000	160	160	0
29-Sep-03	$ 236,600	$ 2,000	160	160	0
6-Oct-03	$ -	$ 3,000	160	0	160
13-Oct-03	$ -	$ 2,000	160	0	160
20-Oct-03	$ -	$ 3,000	160	0	160
27-Oct-03	$ 236,600	$ 2,000	160	0	160
3-Nov-03	$ -	$ 3,000	160	160	0
10-Nov-03	$ -	$ 2,000	160	160	0
17-Nov-03	$ -	$ 3,000	160	160	0
24-Nov-03	$ 236,600	$ 2,000	160	160	0
1-Dec-03	$ -	$ 3,000	160	0	160
8-Dec-03	$ -	$ 2,000	160	0	160
15-Dec-03	$ 236,600	$ 2,000	160	0	160
22-Dec-03	$ 397,040	$ 6,000	160	0	160

Figura 5.2 Ejemplo de información de entrada de un proyecto en particular.

Otro tipo de información que también se lee de archivos de texto, es la referente a ingresos y egresos de la organización desarrolladora de software (ODS) que son independientes de los proyectos de software.

RQS se conecta a la base de datos que contiene información de cuantos y cuales ingenieros conocen los lenguajes de programación necesarios para los proyectos y extrae sólo la información relativa a los lenguajes necesarios para los proyectos. La tabla 5.1 muestra un ejemplo del tipo de información recopilada, el 1 indica que si lo domina y el 0 que no.

Ingenieros por lenguaje		
Ingenieros	Progress	J2EE
aarmada	1	1
agiron	1	1
apallares	1	0
apineda	0	1
arangel	0	1
arivas	0	0
azalazar	1	1
bcoss	1	1
cdelatorre	1	1
cguerrero	0	1
dgarcia	0	1
Total	6	10

Tabla 5.1 Lenguajes de programación e ingenieros que los dominan.

Para configurar ciertos valores de manera manual y que no se ingresan por ninguno de los dos medios anteriores, RQS despliega una ventana como la mostrada en la figura 5.3.

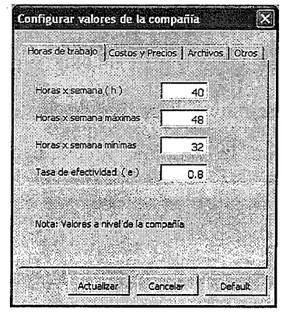


Figura 5.3 Ventana de configuración de valores de RQS.

Esta ventana de configuración se divide en 4 apartados: horas de trabajo, costos y precios, archivos, y otros.

- Dentro del apartado de horas de trabajo se solicitan: el número de horas de trabajo a la semana, el número de horas de trabajo máximas en una semana tomando en cuenta tiempo extra, y el número de horas de trabajo mínimas sin que la ODS tenga pérdidas económicas.

- Dentro del apartado de costos y precios se requieren: el costo por una hora extra de trabajo de un ingeniero, el precio promedio por cada hora de trabajo de un ingeniero, la tasa de descuento anual, y el saldo inicial (efectivo) en bancos y caja.

- En el apartado de archivos se configuran los nombres de los archivos de texto y la base de datos.

- Finalmente, en el apartado de otros se introducen algunos parámetros del modelo.

Procesamiento de la información

Una vez que toda la información de entrada fue proporcionada, RQS hace un procesamiento y ordenamiento de los datos con la finalidad de dejarlos listos para que sean entrada del modelo propuesto en el capítulo anterior. El procesamiento de la información incluye algoritmos para: determinar el tamaño de la ventana de evaluación, ajustar la información de los proyectos en la ventana de evaluación, obtener las matrices de las restricciones, entre otros.

Adicionalmente, en esta etapa se prepara la información que será entrada para resolver el modelo con la técnica de ramificación y acotamiento incluida en el software LINGO.

Algoritmo utilizado

RQS utiliza enumeración explicita para encontrar la combinación óptima de proyectos de software candidatos. El algoritmo está formado por los siguientes 3 pasos:

1) Se listan todas los posibles valores del vector de decisiones $\bar{x}$

   - $x_p = 1$ cuando p corresponde a un proyecto en curso.
   - $x_p = \{0,1\}$ cuando p corresponde a un proyecto candidato.
   - Si se tienen v proyectos candidatos, existen 2^v alternativas diferentes.

2) Se calcula el valor de la función objetivo Z (4.11) para cada posible alternativa de $\bar{x}$. En el momento en que alguna restricción (4.8, 4.10, 4.15, 4.17, 4.18) es violada, la alternativa se considera no factible y se procede a continuar con la siguiente alternativa.

3) Se ordenan las alternativas de $\bar{x}$ de mayor a menor valor obtenido en Z siempre y cuando sean factibles. El vector $\bar{x}$ que produce el valor más grande Z es la solución óptima.

Tipo de resultados

Al finalizar el algoritmo anterior, RQS muestra una pequeña ventana resumiendo los resultados obtenidos en la evaluación. Esta ventana contiene información acerca de: número de proyectos en curso, número de proyectos candidatos, total de alternativas posibles, total de soluciones factibles y total de soluciones no factibles.

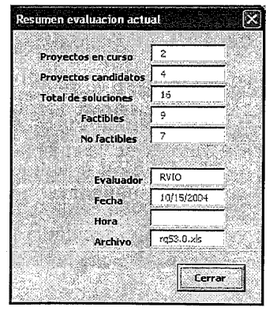


Figura 5.4 Resumen de la evaluación actual.

Una vez que se encuentra la solución óptima, RQS muestra el impacto de $\bar{x}$ en el número de horas totales por arriba (4.4) y/o por debajo (4.5) de la meta y en el flujo de efectivo acumulado (4.16), para cada semana del periodo de evaluación. Esto lo hace auxiliándose en gráficas como las mostradas en la figura 5.5 En la gráfica de las horas no ocupadas y las horas extras, la línea superior indica el máximo de horas de trabajo y la línea inferior el mínimo aceptable de horas no ocupadas para que la empresa no tenga perdidas económicas.

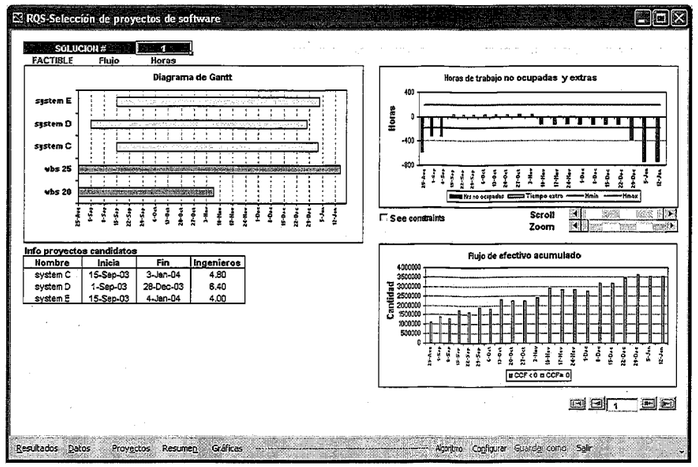


Figura 5.5 Ventana de resultados.

La ventana de resultados también contiene un diagrama de Gantt con los proyectos en curso y los proyectos candidatos seleccionados.

Además, RQS permite recorrer cada alternativa factible de mejor a peor, y conocer el impacto de la selección en ciertas variables de la misma manera que hace con la solución óptima. Inclusive si se desea ver las soluciones no factibles y observar las causas gráficamente también es posible, ya que RQS recalcula toda la ventana de evaluación para esa alternativa en particular.

Las soluciones no sólo se muestran de forma gráfica, sino que también se encuentran en tabulados sin formato en una hoja llamada "datos". Por ejemplo, en la figura 5.6 se muestra el área donde se encuentra un listado con todas las alternativas posibles y los valores que se obtuvieron al calcular la función objetivo Z. Si la función Z tiene un valor negativo significa que la alternativa de $\bar{x}$ violó por lo menos una restricción, y por tanto es una solución no factible.

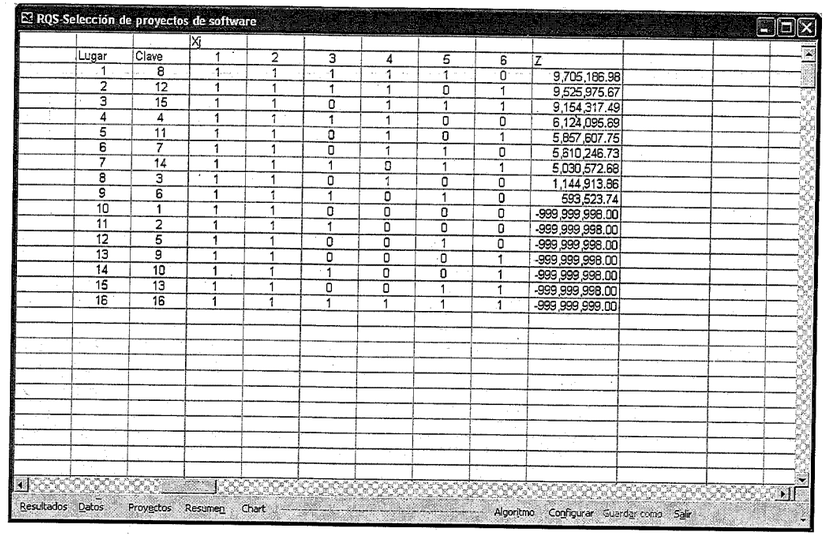


Figura 5.6 Ventana de la hoja "datos" con ejemplo de un tabulado de resultados.

Alcances y limitaciones

RQS es una herramienta pequeña de aproximadamente 600 kb. Está construida para Excel 2000 en adelante y Windows Xp. Toda la programación es interna y fue realizada en VBA. No es totalmente portable, por ejemplo puede tener problemas en otras versiones de Windows.

La capacidad de RQS soporta a lo más 10 proyectos en curso y 10 proyectos candidatos, y 10 lenguajes de programación distintos, excepto que aún no es posible configurar cuales serán y en este momento RQS tiene por default una lista con 9 lenguajes.

RQS permite correr el algoritmo de manera inmediata y hace todo el procesamiento necesario de la información. De esta manera pequeños cambios en los datos son fáciles de manejar por el usuario, que simplemente mueve un valor y RQS calcula inmediatamente los cambios.

RQS "prepara" de manera automática los datos de entrada que requiere el modelo por el software Lingo para ser resuelto utilizando la técnica de ramificación y acotamiento. Mas adelante, en la sección 3.3, se explicará un poco más este punto.

RQS es una primera versión de lo que puede convertirse en una herramienta DSS más poderosa, con mayor funcionalidad y más robusta, inclusive se puede llamar a las librerías del software LINGO de manera interna para aprovechar algoritmos más rápidos.

5.2 Resultados

En esta sección se presentan resultados obtenidos con ayuda de la herramienta RQS con el fin de ilustrar el comportamiento del modelo y de la herramienta. Los datos de entrada son simulados, es decir no son de proyectos reales, sin embargo si cumplen con los supuestos mencionados en el apartado 4.1.7.

El escenario base contiene 2 proyectos en curso y 4 proyectos candidatos. En el escenario alternativo se realizan pequeñas variantes para ejemplificar el impacto en la selección óptima. Además, se comparan distintos casos no factibles y se hace una discusión acerca de las causas y su posible interpretación.

Escenario base.

Se consideran 2 proyectos en curso denotados por A y B, y 4 proyectos candidatos denotados por C, D, E y F.

Para simplificar se asume que un proyecto se realiza en un lenguaje de programación, aunque RQS soporta hasta 10 lenguajes para un mismo proyecto. Además, las necesidades de recursos humanos se toman constantes a través del tiempo, por ejemplo: el proyecto A requiere de 160 horas por cada semana de su periodo de vida.

A nivel de empresa, se manejan 2 lenguajes: Progress y J2EE. La empresa tiene 24 ingenieros, de los cuales 15 dominan Progress y 20 dominan J2EE, por lo tanto 9 ingenieros dominan ambos lenguajes.

Se consideraron 40 horas disponibles a la semana por cada ingeniero o 48 si se incluyen horas extras. El límite mínimo de horas a ocupar para que la empresa no tenga pérdidas económicas se estableció en 32 horas a la semana.

Los proyectos se resumen en la tabla 5.2 mientras que la figura 5.7 se muestra el diagrama de Gannt respectivo.

Proyectos escenario base						
Proyecto	A	B	C	D	E	F
Candidato	No	No	Sí	Sí	Sí	Sí
Lenguaje	Progress	J2EE	Progress	J2EE	J2EE	Progress
Duración	17	16	15	15	18	17
Ingenieros	4	6	6	5	4	7
Esquema pagos	mensual	mensual	mensual	mensual	10% inicio, 90% final	variable

Tabla 5.2 Resumen de todos los proyectos del escenario base.

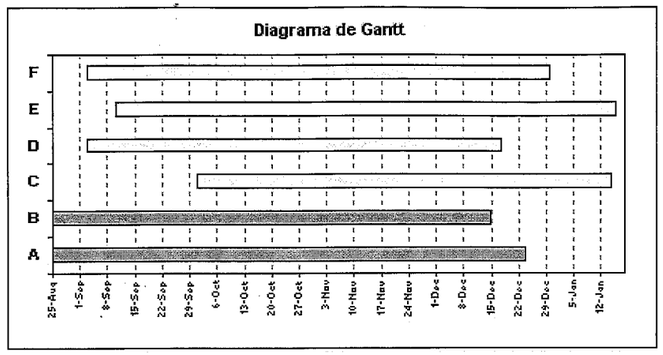


Figura 5.7 Diagrama de Gantt con todos los proyectos del escenario base.
Los proyectos A, B corresponde a proyectos en curso y los proyectos (C, D, E, y F) son candidatos.

Los resultados obtenidos de este escenario base se resumen en las tablas 5.3 y 5.4. Ahora bien, lo primero que resalta es que las 3 primeras opciones deciden tomar a los proyecto D y F, así que la diferencia en el resultado lo hace el seleccionar a C, E o ninguno.

Resumen evaluación	
Proyectos	**6**
En curso	2
Candidatos	4
Alternativas	**15**
Factibles	7
No factibles	8

Tabla 5.3 Resumen de la evaluación del escenario base.

Lugar	Factibles				Z
1	C	D	---	F	100
2	---	D	E	F	89
3	---	D	---	F	86
4	C	---	E	F	75
5	C	---	---	F	72
6	C	D	E	---	67
7	C	D	---	---	61

Tabla 5.4 Resumen de los resultados factibles.
Z es el valor normalizado de la función objetivo.

Analizando las gráficas de las 3 primeras opciones (figura 5.8), vemos que el flujo de efectivo acumulado tiene un mejor comportamiento para la solución 1 respecto a la 2 y 3. En cambio si se hace un análisis respecto a la total de horas utilizadas, resulta que la solución óptima requiere de 12 semanas de trabajo extra de todos los ingenieros, la solución 2 requiere solo del tiempo extra de 8 ingenieros durante 15 semanas, y la solución 3 no ocupa a 2 ingenieros durante 16 semanas.

Otro aspecto a considerar es que la solución óptima tiene altas exigencias en el uso de horas extras, inclusive el tiempo extra necesitado es equivalente a 5 ingenieros de tiempo completo adicionales durante 3 meses. Bajo esta perspectiva pudiera ser que la solución 2 sea más conveniente en términos de negocios.

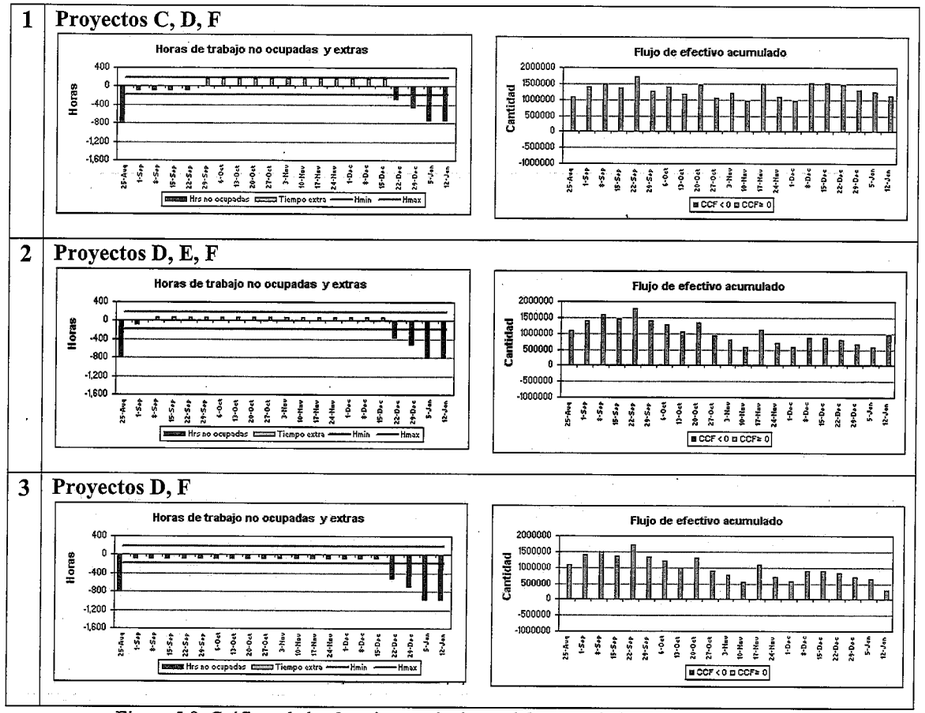


Figura 5.8 Gráficas de las 3 mejores soluciones del escenario base. Todas son factibles.

Las siguientes 4 opciones factibles son las mostradas en la figura 5.9. Nuevamente se observa que la cantidad del flujo de efectivo acumulado va decreciendo conforme decrece el lugar ocupado por la alternativa. En cuanto a horas extras y/o desocupadas, la alternativa 4 necesita que 15 ingenieros trabajen tiempo extra, la alternativa 5 mantiene un ingeniero desocupado, la alternativa 6 obliga que 5 ingenieros trabajen tiempo extra, y la alternativa 7 no ocupa a 3 ingenieros. Todas estas cifras son para el mismo periodo de 12 semanas.

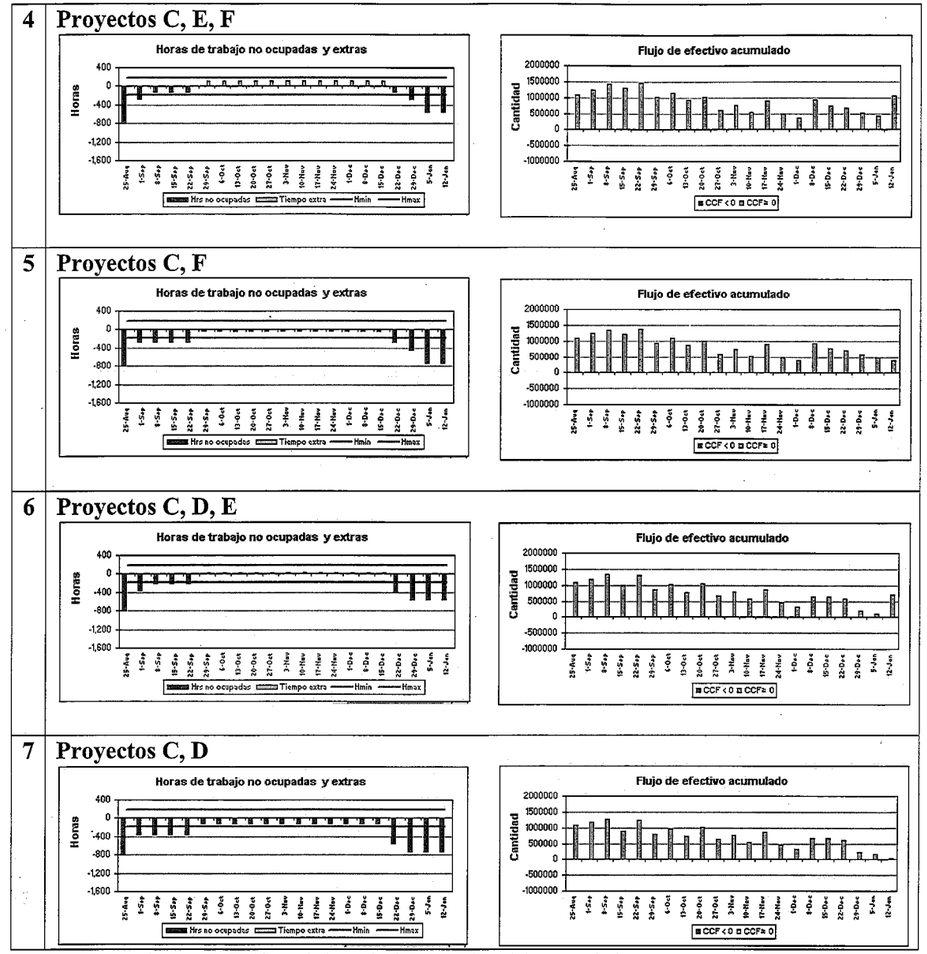


Figura 5.9 Gráficas de las soluciones 4 a la 7 del escenario base. Todas son factibles.

Lugar	No factibles				Z
8	---	---	E	F	NA
9	C	---	E	---	NA
10	---	D	E	---	NA
11	---	---	---	F	NA
12	---	D	---	---	NA
13	C	---	---	---	NA
14	---	E	---	---	NA
15	---	---	---	---	NA
16	C	D	E	F	NA

Tabla 5.5 Resumen de los resultados NO factibles.
El valor de Z no aplica (NA) para estas opciones.

La figura 5.10 muestra las gráficas para las 9 soluciones no factibles. A continuación de hace un breve análisis de algunas de ellas que resultan interesantes.

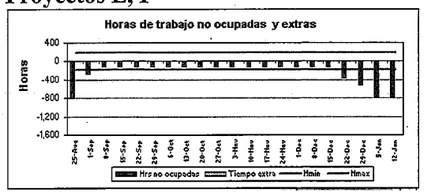
- Alternativa 8 (proyectos E y F). El flujo de efectivo acumulado es negativo durante pocas semanas y en una cantidad pequeña. Probablemente con un pequeño ajuste en el esquema de pagos, esta alternativa podría ser factible. Esto también puede aplicar en caso de proyectos con altas utilidades pero que debido a la mala planeación del esquema de pagos, existan unas pocas semanas en que no se satisface el flujo necesario. Con ayuda de RQS es posible detectar estos casos.

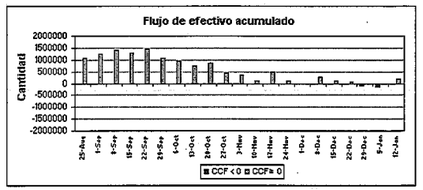
- Alternativas 9 a la 13. Tienen pequeños problemas al final de la ventana de evaluación con el flujo de efectivo acumulado. Sin embargo las horas están muy cercanas al limite de mínimo de horas desocupadas antes de que la compañía tenga perdidas económicas, sin embargo, tampoco son tan malas opciones suponiendo que se tendrán suficientes ingenieros disponibles para aceptar nuevos proyectos futuros que se presenten.

- Alternativa 14 y 15. Las horas no ocupadas sobrepasan por mucho el mínimo deseado y el flujo de efectivo es altamente negativo durante muchas semanas.

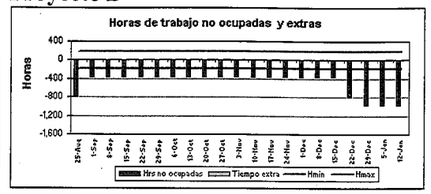
- Alternativa 16. Esta considera todos los proyectos candidatos, obviamente no tiene problemas con el flujo de efectivo, inclusive es la alternativa que le da mas a ganar a la empresa, sin embargo, no tiene los recursos humanos suficientes. La gráfica sugiere 5 ingenieros de software de tiempo completo adicionales a los 24, para que los proyectos se realicen satisfactoriamente y no se empleen horas extras.

8	**Proyectos E, F**
9	**Proyectos C, E**
10	**Proyectos D, E**
11	**Proyecto F**





12	**Proyecto D**
13	**Proyecto C**
14	**Proyecto E**
15	**No elegir ninguno**
16	**Proyectos C, D, E, F (todos)**



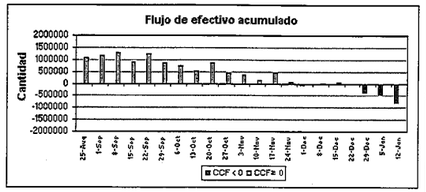


Figura 5.10 Gráficas de las soluciones no factibles del escenario base.

Cabe hacer la aclaración que las figuras 5.8, 5.9, y 5.10 incluyen también a los proyectos en curso, A y B, a pesar de que únicamente se mencionan los proyectos candidatos seleccionados.

Escenario alternativo.

Se considera el mismo escenario base, con el único cambio de que el proyecto E ahora tiene pagos mensuales iguales. Anteriormente el esquema era pagar el 10% al inicio y el 90% al final del proyecto.

Los resultados obtenidos son los siguientes:

Resumen evaluación	
Proyectos	**6**
En curso	2
Candidatos	4
Alternativas	**16**
Factibles	8
No factibles	8

Tabla 5.6 Resumen de los resultados del escenario alternativo.

Lugar	Factibles					Z	Lugar anterior
1	---	D	E	F		100	2
2	C	D	---	F		94	1
3	C	---	E	F		86	4
4	---	D	---	F		81	3
5	C	D	E	---		79	6
6	---	---	E	F		72	No factible
7	C	---	---	F		68	5
8	C	D	---	---		58	7

Tabla 5.7 Resumen de los resultados factibles del escenario alternativo.

*El lugar anterior es respecto al escenario base.

Algo que resalta inmediatamente es que la combinación de proyectos D, E, y F, es la selección óptima. Esto ocurre porque en el escenario base, el proyecto F tenía un esquema de pagos que impactaba negativamente en el flujo de efectivo acumulado, y hacer el cambio a pagos iguales resultó favorable para el flujo de efectivo.

De manera similar ocurrió con la combinación de proyectos E y F, que en el escenario base era una alternativa no factible por que en pocas semanas no se cumplían las restricciones de flujo de efectivo positivo. Y que en este escenario alternativo es factible.

Debido a que no se hicieron cambios en las horas de trabajo requeridas, las gráficas correspondientes son iguales. Las únicas gráficas que cambiaron son las del flujo de efectivo acumulado donde aparece el proyecto E.

Otras consideraciones.

Este apartado resume las diversas consideraciones al momento de usar e interpretar los resultados de RQS que no se mencionaron en los dos escenarios anteriores.

Algunos parámetros que el tomador de decisiones decide al momento de utilizar el modelo y que afectan la respuesta son:

- La tasa de descuento elegida. La tasa de descuento tiene una relación inversa con el cálculo del valor presente neto (VPN), si la tasa aumenta el NPV disminuye.

- El saldo inicial en bancos o efectivo disponible. Esta cantidad debe de ser la cantidad actual al momento de la evaluación.

- Los costos promedio por hora extra y/o hora desocupada. Si ambos costos son iguales, el modelo no tendrá preferencia entre tiempo extra y horas desocupadas. Lo usual es que el costo de tiempo extra sea mayor al costo de horas desocupadas, ya que desde un punto de vista radical es preferible tener ingenieros desocupados a quienes se les puede conseguir un nuevo proyecto, que saturar a los ingenieros con trabajo extra y poner en riesgo el desarrollo de todos los proyectos por exceso de fatiga.

- Las horas máximas de trabajo a la semana. Si se consideran 40 horas normales y 48 horas incluyendo horas extras equivale a asignar trabajo a 1 ingeniero que deben realizarlo 1.2 ingenieros. Por ejemplo, si son 24 ingenieros, el trabajo asignado es para cerca de 30 ingenieros tiempo completo.

Conclusiones.

En conclusión, se ha demostrado que:

- El modelo es lo suficientemente flexible en sus parámetros de entrada para ajustarse a las políticas de decisión de cada empresa.

- RQS siempre encuentra (si existe) el óptimo, ya que hace una enumeración explícita de las opciones.

- Existen muchas ventajas al listar todas las alternativas que sólo encontrar la solución óptima, ya que se pueden identificar casos interesantes como que la solución óptima no es precisamente la mejor decisión de negocios, o que es necesario cambiar el esquema de pagos de un proyecto.

Por todo lo anterior, la sugerencia es tomar a RQS como un evaluador de escenarios más que como una herramienta que da la decisión correcta, así el tomador de decisiones podrá anticipar ciertos resultados en caso de que la decisión de aceptar proyectos no sea totalmente de la empresa y dependa de factores externos.

5.3 Solución con LINGO

LINGO es un lenguaje de modelado matemático diseñado en particular para formular y resolver una amplia variedad de problemas de optimización, que incluyen problemas de programación lineal, programación entera y programación no lineal [22]. Se puede obtener una versión de estudiante en www.lindo.com.

Los modelos construidos en LINGO contienen 3 secciones básicas:

1. Definición de conjuntos: en esta sección se definen los conjuntos, el número de sus elementos y sus atributos.
2. Cuerpo del modelo: corresponde a la formulación matemática del modelo. El primer renglón corresponde a la definición de la función objetivo.
3. Sección de datos: pueden definirse directamente o bien obtenerse de una fuente externa. La función @OLE permite recuperar y colocar datos en una hoja de cálculo de Excel. De manera similar, la función @ODBC es una función de conexión para transferir datos de y a una base de datos [27].

La finalidad de resolver el modelo propuesto con LINGO es demostrar que no es necesario tener una herramienta similar a RQS para encontrar la solución óptima, además para resaltar la importancia del planteamiento del modelo, y finalmente, dar una alternativa para cuando se tienen muchos proyectos candidatos y sólo se quiere conocer la solución óptima.

Sin embargo, al momento de desarrollar estas pruebas, no fue posible modelar por completo el planteamiento del modelo debido principalmente a los límites de la versión de estudiante que no permiten un número mayor de 150 restricciones.

La estrategia para reducir el número de restricciones fue separar de manera intuitiva el modelo propuesto en el capítulo 4 en dos modelos: uno que minimiza el costo de las horas extras y las horas desocupadas en la ventana de evaluación e incorpora las restricciones de los recursos humanos, y otro que maximiza la utilidad verificando que el flujo de efectivo acumulado siempre sea positivo. Además se adecuó RQS para que ejecutar tres tipos de evaluaciones: una con el modelo de utilidad, otra con el modelo de recursos humanos, y el tercero con el modelo propuesto que combina ambos modelos. Así, se encontró la solución óptima con RQS tomando el modelo de recursos humanos y después se utilizó LINGO para verificar que la respuesta fuera exactamente la misma.

El modelo en LINGO consta sólo de 28 líneas de código, y es capaz de leer los datos de entrada y depositar los resultados en una hoja de Excel. LINGO utiliza la técnica de ramificación y acotamiento para encontrar la solución óptima. RQS prepara los datos de manera tal que el modelo en LINGO los pueda interpretar. La figura 5.11 contiene a la ventana con el modelado en LINGO.

LINGO Model - GP4RVIO-6.1

```
!Modelo de programacion por metas que minimiza los costos de horas desocupadas
y horas extras e incorpora las restriccione de recursos humanos;
MODEL:
  SETS:
    var/1..6/:x;
    constr/1..63/:b;
    objetivo/1..21/:wmin, wplus, goal, dplus1,dminus1,opt,pminus,pplus;
    mata(constr,var):A;
    matc(objetivo,var):C;
  ENDSETS
    ! Función objetivo ;
    min = @SUM(objetivo(i): (wmin(i)*dminus1(i)+wplus(i)*dplus1(i)));
    ! Restricciones de las metas;
    @FOR(objetivo(i)|goal(i)#NE#0: @sum(var(j):c(i,j)*x(j))+dminus1(i)-dplus1(i)=goal(i));
    ! Restricciones de los recursos humanos;
    @FOR(constr(i): @sum(var(j):a(i,j)*x(j)) <= b(i));
    ! Variables de decision binarias;
    @FOR(var(j):@BIN(x));
    ! Se fijan los valores para los proyectos en curso;
    @FOR( var( J) | J #LE# 2: x( J)=1);
    ! Horas totales ocupadas;
    @FOR(objetivo(i): opt(i)= @sum(var(j):c(i,j)*x(j)));
  DATA:
    A = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    b = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    C = @OLE('C:\prevalQS\datosPruebaLingo.xls','CH');
    goal = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    pminus = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    pplus = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    wmin = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    wplus = @OLE('C:\prevalQS\datosPruebaLingo.xls');
    @OLE('C:\prevalQS\datosPruebaLingo.xls')=opt;
    @OLE('C:\prevalQS\datosPruebaLingo.xls')=x;
    @OLE('C:\prevalQS\datosPruebaLingo.xls')=dplus1;
    @OLE('C:\prevalQS\datosPruebaLingo.xls')=dminus1;
  ENDDATA
END
```

Figura 5.11 Ventana de LINGO con el modelo de recursos humanos.

Ejemplo #1:

Se tomaron los datos de entrada del escenario base. Los resultados obtenidos con RQS están en la tabla 5.8. La selección óptima de proyectos es D, E y F, esto es resolviendo el modelo de recursos humanos que no toma en cuenta el flujo de efectivo.

Lugar	A	B	C	D	E	F	Z
1	1	1	0	1	1	1	33
2	1	1	0	1	0	1	34
3	1	1	1	1	1	0	37
4	1	1	1	0	1	1	38
5	1	1	1	0	0	1	38
6	1	1	1	1	0	1	39
7	1	1	0	0	1	1	39
8	1	1	1	1	0	0	51
9	1	1	0	1	1	0	52
10	1	1	1	0	1	0	57
11	1	1	0	0	0	1	60
12	1	1	0	1	0	0	73
13	1	1	1	0	0	0	78
14	1	1	0	0	1	0	79
15	1	1	0	0	0	0	100
16	1	1	1	1	1	1	No factible

Tabla 5.8 Soluciones del escenario base resolviendo el modelo de recursos humanos.

Para LINGO, el modelo de recursos humanos con los datos del escenario base tiene las siguientes dimensiones: 106 restricciones lineales, 67 variables, y 4 variables enteras. La solución óptima en LINGO coincidió completamente con la solución óptima proporcionada por RQS. A continuación se muestra una parte del reporte de solución generado por LINGO:

```
Global optimal solution found at iteration:          70
Objective value:                                     33
                     Variable           Value
                        X( 1 )       1.000000
                        X( 2 )       1.000000
                        X( 3 )       0.000000
                        X( 4 )       1.000000
                        X( 5 )       1.000000
                        X( 6 )       1.000000
```

LINGO proporciona los valores de las variables d_k^+ y d_k^- en forma de tabulados o gráficamente, esto permitió verificar que tanto LINGO como RQS llegaron a respuestas idénticas, las figuras 5.12 y 5.13, así como la tabla 5.9 muestran los valores respectivos. En el modelo de LINGO, la variable d_k^+ se llama dplus1, mientras que dminus1 es la variable que corresponde a d_k^-.

k	DPLUS1(k)	DMINUS1(k)
1	0	800
2	0	80
3	80	0
4	80	0
5	80	0
6	80	0
7	80	0
8	80	0
9	80	0
10	80	0
11	80	0
12	80	0
13	80	0
14	80	0
15	80	0
16	80	0
17	80	0
18	0	360
19	0	520
20	0	800
21	0	800

Tabla 5.9 Valores de las horas extras (dplus1) y horas no ocupadas (dmin1), obtenidas con LINGO.

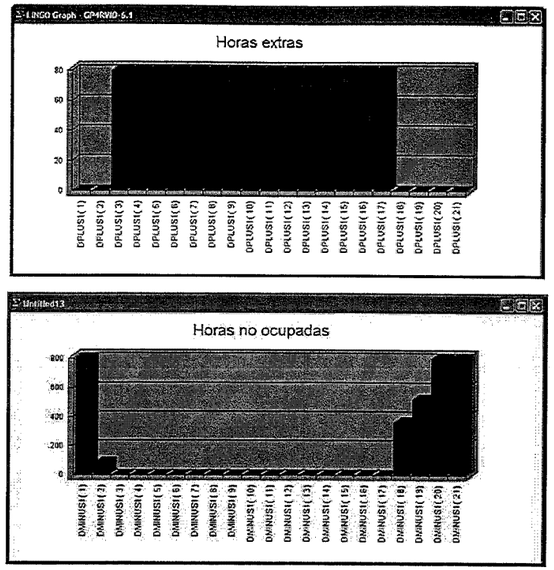


Figura 5.12 Gráficas de dplus1 y dmin1.

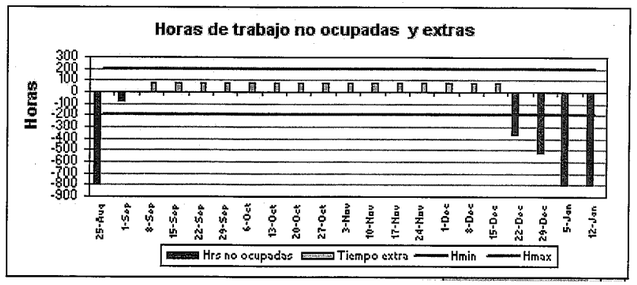


Figura 5.13 Soluciones del escenario base resolviendo el modelo de recursos humanos en RQS.

Ejemplo #2:

Este ejemplo evalúa 2 proyectos en curso y 10 proyectos candidatos. La versión de estudiante de LINGO tiene capacidad hasta 30 proyectos candidatos y RQS solo soporta 10

proyectos candidatos, así que es un buen ejemplo para probar los límites de RQS y la rapidez de LINGO.

Los parámetros de entrada son los mismos que en el escenario base. Es decir, se tomaron 2 lenguajes: Progress y J2EE, tasa de descuento anual del 10%, 24 ingenieros en total, 15 que dominan progress y 20 dominan J2EE, entre otros.

Los proyectos en curso se denotan por A y B. El proyecto A es para 4 ingenieros que dominen Progrees y el proyecto B es para 6 ingenieros que dominen J2EE. La información de los 10 proyectos candidatos (C al L) está en la tabla 5.10 y el diagrama de Gantt con todos los proyectos está en la figura 5.14.

Nombre	Inicia	Fin	Ingenieros	Lenguaje
C	1-Oct-03	15-Ene-04	6.00	Progress
D	3-Sep-03	18-Dic-03	5.00	J2EE
E	10-Sep-03	16-Ene-04	4.00	J2EE
F	3-Sep-03	30-Dic-03	7.00	Progress
G	21-Oct-03	25-Dic-03	3.00	Progress
H	3-Sep-03	23-Oct-03	8.00	Progress
I	3-Nov-03	16-Ene-04	4.00	J2EE
J	3-Sep-03	10-Dic-03	3.00	Progress
K	7-Oct-03	30-Dic-03	4.00	J2EE
L	3-Sep-03	5-Dic-03	5.00	Progress

Tabla 5.10 Información de los proyectos candidatos.

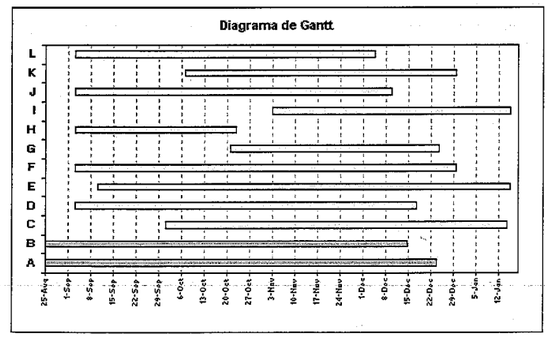


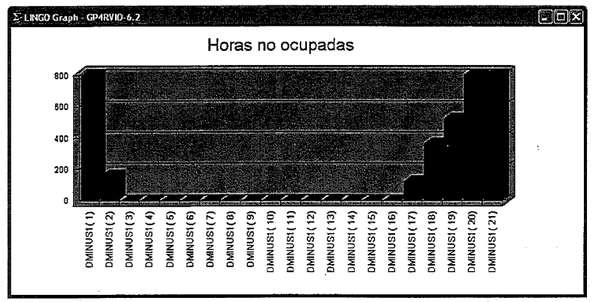
Figura 5.14 Diagrama de Gantt con los 12 proyectos del ejemplo #2.

La solución óptima proporcionada por RQS fue seleccionar los proyectos E, F y J, con un valor de la función objetivo de 24 y como se esperaba, LINGO obtuvo la misma respuesta. Lo interesante de esta selección es que se alcanza exactamente la meta de trabajo durante 14 semanas de 21 que tiene la ventana de evaluación, esto significa que no hay horas extras ni horas desocupadas en esas 14 semanas.

Igual que en el ejemplo #1, se obtuvieron algunas gráficas de RQS y LINGO para verificar que los valores reportados fueran exactamente los mismos.

k	DPLUS1(k)	DMINUS1(k)
1	0	800
2	0	160
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	120
18	0	360
19	0	520
20	0	800
21	0	800

Tabla 5.11 Valores de las horas extras (dplus1) y horas no ocupadas (dmin1), obtenidas con LINGO y RQS para la solución óptima (proyectos E, F y J).



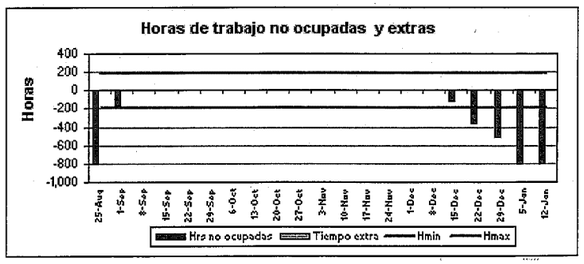


Figura 5.15 Valores de las horas no ocupadas para la solución óptima. Gráfica superior obtenida con LINGO y gráfica inferior con RQS.

Ahora bien, estos resultados en LINGO demuestran que RQS efectivamente encuentra el resultado óptimo con los valores correctos. La sugerencia es utilizar RQS si el número de proyectos a seleccionar es pequeño, en caso contrario utilizar LINGO.

Capítulo 6

Conclusiones y Trabajo Futuro

El problema de selección de proyectos de software candidatos resuelto en esta tesis surgió en QS, una organización desarrolladora de software (ODS) a la medida que está establecida en México, D.F., debido a esto, la definición de problema capta algunas particularidades y limitaciones propias de QS, sin embargo la solución propuesta es generalizada a organizaciones similares.

El primer objetivo de esta tesis fue plantear un modelo matemático para la selección de proyectos de software en QS que pudiera ser extendido a una ODS en general. Se requería que el modelo tomará en cuenta el mayor número posible de los siguientes aspectos: ventana de evaluación común, optimización de recursos, utilidad proporcionada, flujo de efectivo, disponibilidad de recursos, información variable y escasa, estimación de información, proyectos heterogéneos, e incertidumbre,

Como resultado del primer objetivo, se planteó un modelo de optimización de recursos humanos basado en programación por metas que incorpora el valor presente como medida de la utilidad, y que considera como restricciones la disponibilidad de los recursos humanos y el flujo de efectivo. Una característica importante es que todas las variables de decisión son binarias y corresponden a la decisión de seleccionar o no un proyecto, por lo tanto, en términos generales el modelo propuesto es un modelo de optimización binaria con restricciones.

De los aspectos mencionados en el primer objetivo, el modelo propuesto no toma en cuenta la incertidumbre y la estimación de la información. La incertidumbre inherente de los

proyectos candidatos no se incluye en el modelo, ya que se asume que la información de los proyectos es estática. Por otro lado, el modelo propuesto tampoco incluye o sugiere métodos de estimación de la información, por lo tanto, la validez del proceso de estimación del tamaño, la duración, y los recursos necesarios del proyecto quedan fuera del alcance de esta tesis al asumir que toda la información de entrada necesaria para resolver el modelo debe ser conocida y/o estimada por otros métodos. El resto de los aspectos fueron incluidos en el modelo propuesto. A continuación se explica como se agregaron al modelo propuesto y, las ventajas y desventajas de hacerlo de esa manera.

Como primer paso se definió una ventana de evaluación común para comparar proyectos con diferentes tamaños de vida y se discretizó esta ventana en *n* periodos de tiempo iguales. De esta forma el modelo adquirió la capacidad de considerar múltiples periodos de tiempo en el planteamiento de las metas y la verificación de las restricciones, es decir, el modelo capta los cambios en la información y por tanto permite evaluar proyectos heterogéneos. Una vez que se definió la ventana y la información de entrada se organizó de manera tal que coincidiera con los periodos en que se discretizó la ventana, se procedió a plantear la parte del modelo que optimiza los recursos.

De manera puntual, la propuesta para optimizar los recursos es minimizar el costo de las horas no ocupadas y el costo de las horas extras de los ingenieros durante la ventana de evaluación para los proyectos seleccionados. Para lograr lo anterior se formuló una función objetivo basada en programación por metas: las metas se refieren al número deseable de horas de trabajo de los ingenieros para cada periodo de la ventana de evaluación y los pesos de penalización representan los costos por una hora no ocupada o por una hora extra, traídos a valor presente. De esta forma, la interpretación económica en la optimización de recursos al seleccionar proyectos se volvió la principal cualidad del modelo propuesto, y aunado a esto, su principal limitante resultó ser que únicamente optimiza el recurso humano. Con esto, una parte esencial del modelo estaba ya plasmada, sin embargo aún faltaba incorporar la utilidad.

La utilidad proporcionada por los proyectos se cuantificó con el valor presente neto (VPN) y se aprovechó el hecho de que la función objetivo del modelo que optimiza los recursos humanos representa costos a valor presente para acoplar el VPN en dicha función objetivo. Gracias esto, se logró tener una función mono-objetivo, a pesar de que el problema original era multi-objetivo por que perseguía optimizar los recursos y maximizar la utilidad al mismo tiempo. Sin embargo, el modelo heredó la desventaja de que el resultado depende en gran medida de la elección de la tasa de descuento.

Por otro lado, el flujo de efectivo se incorporó al modelo en forma de restricciones. Estas restricciones garantizan que los proyectos seleccionados proporcionarán siempre un flujo de efectivo acumulado positivo y se establecen para cada periodo en que se dividió la ventana de evaluación. Incluir estas restricciones al modelo resulta indispensable porque una ODS no debe quedarse sin efectivo para cumplir las obligaciones contraídas con anticipación, sin embargo, estas restricciones se basan en dos supuestos que no siempre se cumplen: que toda la ganancia es reinvertida, y que los ingresos y egresos de la ODS siempre se realizan al inicio de cada periodo de la ventana. Derivado de lo anterior, las restricciones relativas al flujo de efectivo son muy estrictas, por ejemplo si ocurre que en un periodo el flujo de efectivo acumulado es negativo pero relativamente muy pequeño, el modelo determina que la opción no es factible por violar al menos una restricción, cuando en la práctica la cantidad realmente es insignificante o bien las obligaciones no tienen que pagarse necesariamente el día en que inicia el periodo y esto proporciona cierta holgura a la ODS. Una manera de suavizar la restricción es permitir que el tomador de decisiones determine el límite inferior del flujo de efectivo y no asumir que necesariamente ese límite es cero.

Otro de los aspectos que también se incorporó al modelo propuesto en forma de restricciones es la disponibilidad de los recursos. Nuevamente, los únicos recursos considerados son los recursos humanos, esto es porque para una ODS su principal recurso es el recurso humano. Aquí, un supuesto importante es asumir que los requerimientos de recurso humano se especifican en términos del tipo de conocimiento y la cantidad de horas necesarias, sin embargo, se requiere tener especial cuidado debido a que generalmente un

ingeniero domina más de un tipo de conocimiento. Así que la línea que se siguió para verificar la disponibilidad de recursos fue establecer restricciones en base al tipo de conocimiento para que los ingenieros no sean contados como dos o más.

Una vez hecho el planteamiento del modelo es necesario elegir el algoritmo de solución. Como se había mencionado antes, el modelo propuesto es un modelo binario con restricciones. Afortunadamente existen numerosos algoritmos para encontrar la solución óptima de este tipo de modelos. En esta tesis se exploraron dos opciones: la técnica de ramificación y acotamiento, y la enumeración explícita. Específicamente, se uso la técnica de ramificación y acotamiento que acompaña al software de modelado LINGO, mientras que la enumeración explícita se incluyó en una herramienta especialmente construida como parte del segundo objetivo de esta tesis. Ambas estrategias de solución resultaron acertadas.

Concretamente, el segundo objetivo que perseguía cumplir esta tesis era construir una herramienta tipo DSS (Decision Support System) basada en el modelo matemático. Y se alcanzó completa y satisfactoriamente mediante la herramienta RQS.

Además de encontrar la selección óptima de proyectos, la herramienta RQS ofrece muchas ventajas adicionales al tomador de decisiones: alimentación automática de datos, configuración de ciertos valores, resultados gráficos, tablas, listado con los resultados de todas las combinaciones, y permite recalcular inmediatamente si hay cambios en la información. La principal desventaja de la herramienta es que está limitada a un número máximo de 20 proyectos distribuidos de la siguiente manera: 10 proyectos en curso y 10 proyectos candidatos.

Para validar los resultados de RQS se corrieron algunos ejemplos tanto en RQS como en LINGO. El lenguaje de modelado LINGO usa la técnica de ramificación y acotamiento para desplegar solo la selección óptima de proyectos y su impacto en ciertas variables, así que se compararon las soluciones óptimas de RQS y de LINGO, resultando que eran idénticas en todos los casos.

A pesar de que LINGO es un software que es fácil de utilizar y eficiente en proporcionar la solución óptima, construir RQS permitió explorar el comportamiento del modelo propuesto y sugirió nuevas cualidades y/o alternativas del modelo (algunas se discutirán cuando se hable del trabajo futuro). En lo referente al modelo propuesto actual, RQS demostró que el modelo es lo suficientemente flexible en sus parámetros de entrada para ajustarse a las políticas de cada ODS. Adicionalmente, como RQS lista todas las combinaciones posibles, se pudo observar que algunas veces existían casos interesantes dentro de las soluciones no factibles o bien que la selección óptima podía no ser precisamente la mejor decisión de negocios.

En resumen, la principal ventaja de utilizar enumeración explicita en RQS es que también funciona como un evaluador de escenarios, permitiendo al tomador de decisiones anticipar y/o conocer ciertos resultados futuros o bien tomar decisiones derivados de estos.

Posteriormente, el tercer objetivo trazado al inicio de esta tesis era implementar la solución propuesta en QS, haciendo la aclaración que la solución incluye el modelo propuesto y la herramienta RQS. En este sentido, la dinámica fue ir presentando los avances del modelo a través de la herramienta y QS a su vez daba información que servía como retroalimentación para el modelo. Hasta este momento, no se ha logrado implementar por completo la herramienta en QS, si bien todo está dispuesto para empezar a utilizarla, falta hacer las conexiones entre las bases de datos reales. A pesar de esto, se cuenta con la aprobación y validación del modelo y la herramienta por parte de QS.

De los 3 objetivos de esta tesis listados, se puede decir que en general fueron cubiertos, más sin embargo, durante la evolución de la tesis se identificaron varios aspectos que no fueron incluidos y que quedaron como trabajo futuro.

En relación al modelo de optimización propuesto, el trabajo futuro va encaminado a hacer extensiones al modelo. Algunas de estas extensiones deben considerar lo siguiente:

- Incluir la interdependencia entre proyectos de software, ya que el modelo propuesto es válido solo para seleccionar proyectos independientes.

- Añadir otro tipo de recursos a la optimización y a las restricciones de disponibilidad de recursos, por ejemplo distintos recursos materiales, o bien, establecer mayor nivel de finura en los recursos humanos, por ejemplo adicionalmente al tipo de conocimiento establecer grados de conocimiento.

- Tomar en cuenta la incertidumbre debido a que el modelo propuesto es determinista.

- Asignar los recursos a los proyectos seleccionados para cada periodo de la ventana de evaluación. A pesar de que el modelo verifica la disponibilidad de los recursos es deseable que también asigne los recursos.

Gracias a que RQS utiliza la enumeración explicita es posible analizar todas las soluciones factibles y no factibles, esto también permite al tomador de decisiones detectar algunos casos interesantes, sin embargo RQS no cuenta con un mecanismo que detecte, clasifique y maneje de manera apropiada los casos interesantes. Antes de incorporar un mecanismo, es necesario definir los casos interesantes. Por ejemplo, cuando una combinación de proyectos es altamente rentable y tiene un buen comportamiento en las metas establecidas de uso de recursos humanos, pero en un solo periodo tiene flujo de efectivo acumulado negativo, el modelo propuesto lo considera no factible siendo que puede ser una buena oportunidad para la ODS. Otro ejemplo es cuando una combinación de proyectos requiere un número grande de horas extras por mucho tiempo, tal vez le resulte conveniente a la empresa contratar más ingenieros de manera temporal o permanente. Un último ejemplo es cuando se requiere un número muy alto de horas extras durante solo un periodo, que con atrasar alguno de los proyectos en su fecha de inicio el comportamiento de esa combinación es el mejor. Evidentemente pueden existir muchos tipos de casos interesantes, es por eso que se deben de

definir de manera conjunta con el tomador de decisiones e incluir en RQS para ampliar su poder de análisis e importancia como herramienta de decisión.

Y para finalizar en lo que respecta al trabajo futuro, queda probar el modelo propuesto con valores reales de QS y hacer una evaluación a retrospectiva del modelo y de la herramienta, una vez que ya tenga tiempo de estarse usando en QS.

En resumen, las contribuciones de esta tesis pueden resumirse en los siguientes tres puntos: primero, el planteamiento del problema de selección de proyectos de software candidatos en un modelo de optimización binaria con restricciones no está limitado a un sólo algoritmo de solución, ya que pueden aplicarse diversos algoritmos reportados en la literatura para modelos binarios. Segundo, el modelo propuesto es lo suficientemente flexible en sus parámetros de entrada para ajustarse a las políticas de decisión de cada empresa, y tercero, la herramienta que integra el modelo de selección de proyectos propuesto y que ofrece muchas ventajas al listar todas las alternativas que sólo encontrar la solución óptima.

Resolver el problema de selección de proyectos de software en QS tuvo motivaciones teóricas y motivaciones prácticas. Dentro de las motivaciones teóricas destacan que aunque el problema de selección de proyectos en general ha sido resuelto de diversas maneras, no se encontró una solución específica para seleccionar de proyectos de software, además de que la solución del problema englobaba distintas áreas de conocimientos. En relación a las motivaciones prácticas se puede mencionar que para pequeñas y medianas ODS el seleccionar correctamente los proyectos de software candidatos resulta vital para su sobrevivencia y crecimiento, además de que es posible obtener beneficios inmediatos en la toma de decisiones de la ODS.

Bibliografía

[1] M.C. Alemán, and E. Zavaleta, *Modelos financieros en Excel,* Editorial CECSA, México, 2003.

[2] J. April, F.Grover, and J.P. Nelly, "Optfolio- A Simulation Optimization System for Project Portfolio Planning", *Winter Simulation Conference*, ACM Press, 2003, pp. 301-309.

[3] J.L. Arthur, and A. Ravindran, "PAGP, A Partitioning Algorithm for (Linear) Goal Programming Problems", *ACM Transactions on Mathematical Software*, vol. 6, no. 3, September 1980, pp. 378-386.

[4] G. Baca, *Evaluación de proyectos*, McGraw-Hill, México, 2001.

[5] S. Banerjee, and W.J. Hopp, "The Project Portfolio Management Problem," *Department of Industrial Engineering and Management Sciences*, Northwestern University, June 2001.

[6] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[7] S. Butler, P. Chalasani, S. Jha, O. Raz, and M. Shaw, "The Potential of Portfolio Analysis in Guiding Software Decisions", *First Workshop on Economics-Driven Software Engineering Research (EDSER-1), affiliated with the 21st Int. Conf. on Software Engineering (ICSE'99)*, IEEE Computer Society, USA, 1999.

[8] J.K. Christiansen, and S.C. Nielsen, "Management of Innovation and Improvement: Portfolio Management Approaches and their Application", *SAM/IFSAM VIIth World Congress,* Sweden 2004.

[9] R. Chen, J.Z. Shyu, and G. Tzeng, "Selecting a Weapon System Using Zero-One Goal Programming", XVII-th Multiple Criteria Decision Making International Conference, Canada, 2004.

[10] P.Y. Chu, Y. Hsu, and M. Fehling, "A Decision Support System for Project Portfolio Selection", *Computer in Industry*, vol. 32, no. 2, 1996, pp. 141-149.

[11] R.G. Cooper, S.J. Edgett, and E.J Kleinschmidt, "New Product Portfolio Management: Practices and Performance," *Journal of Product Innovation Management*, vol. 16, no. 4, July 1999, pp. 333-351.

[12] K. Deb, "Non-linear Goal Programming Using Multi-Objective Genetic Algorithms", *Journal of the Operational Research Society,* vol. 52, no. 32, 2001, pp. 291-302.

[13] M.W. Dickinson, A.C. Thornton, and S. Graves, "Technology Portfolio Management: Optimizing Interdependent Projects Over Multiple Time Periods", *IEEE Transactions on Engineering Management*, vol. 48, no. 4, November 2001, pp. 518-527.

[14] H. Erdogmus, "Comparative Evaluation of Software Development Strategies based on Net Present Value", *First Workshop on Economics-Driven Software Engineering Research (EDSER-1), affiliated with the 21st Int. Conf. on Software Engineering (ICSE'99)*, IEEE Computer Society, USA, 1999.

[15] H. Erdogmus, "Valuation of Complex Options in Software Development", *First Workshop on Economics-Driven Software Engineering Research (EDSER-1), affiliated with the 21st Int. Conf. on Software Engineering (ICSE'99)*, IEEE Computer Society, USA, 1999.

[16] Z. Fan, G. Hu, and H. Li, "A Goal Programming Method for Ranking Alternatives Based on Fuzzy Preference Relation", *Journal of Northeastern University (Natural Science)*, vol. 20, vo. 6, 1999, pp. 651-653.

[17] J. Favaro, "A Comparison of Approaches to Reuse Investment Analysis", *Proc. 4th Int. Conf. on Software Reuse*, IEEE Computer Society, 1996, pp. 136-145.

[18] J. Favaro, W. Strigel, and H. Erdogmus, "Return on Investment in the Software Industry", *IEEE Software*, vol.21, no. 3, May/Jun 2004, pp. 18-22.

[19] K.S. Gregory, C.L. McGowan, "Forestry as an Alternative Metaphor for Software", *Software Engineering Economics and Declining Budgets*, Springer –Verlag, 1994, pp. 95-115.

[20] R.L. Gue, and K.C. Cain, "Analysis of Algorithms for the Zero-One Programming Problem", *Communications of the ACM archive*, ACM Press, vol. 11, no. 12, 1968, pp. 837 - 844.

[21] W. Harrison, D. Raffo, and N. Eickelmann, "Adapting Financial Measures: Making a Business Case for Software Process Improvement", *Software Quality Journal*, vol. 8, no. 3, 1999.

[22] F.S. Hillier, and G.J. Liberman, *Investigación de Operaciones,* McGrawHill, Séptima edición, México, 2002.

[23] B. Hughes, M. Cotterell, *Software Project Management*, Third edition, Mc Graw Hill, England, 2002.

[24] J. Jablonsky, "Multicriteria Linear Programming with Spreadsheets", *Conference MME'97*, Czech Republic, 1997.

[25] J. Jablonsky, "Mathematical Programming with LINGO", *Conference University Education Focused on Economic Management*, Czech Republic, 1996.

[26] J. Jablonsky, "Integer Goal Programming Applications using Modeling Systems and Excel", *Proc. Conf. MOPGP'00*, Poland, 2000, pp.195-201.

[27] J. Jablonsky, "Software support for Generalized Goal Programming Models", *Proc. Conf. MME 2000*, Czech Republic, 2000.

[28] P. Jalote, *CMM in Practice*, Addison Wesley, 2000, USA, pp. 37-44.

[29] S.K. Kavadias, C. H. Loch, and U. A. Staffan, "When to Use Marginal Benefits to Maximize Project Portfolio Value", Set. 2001; http:\\www.insead.fr/~loch/articles/GemStone_Port.pdf.

[30] G.C. Kim, and J. Emery, "An Application of Zero-One Goal Programming in Project Selection and Resource Planning: a case study from the Woodward Governor Company", *Computers and Operations Research*, vol. 17, no. 14, 2000, pp. 1389-1408.

[31] F. Padberg, "Scheduling Software Projects to Minimize the Development Time and Cost With a Given Staff", *8th Asia-Pacific Software Engineering Conference*, IEEE Computer Society, 2001, pp. 187-194.

[32] C.Z. Radulescu, M. Radulescu, "Project Portfolio Selection Models and Decision Support", *Studies in Informatics and Control: with Emphasis on Useful Applications of Advanced Technology*, vol. 10, no. 4, 2001.

[33] M. J. Schniederjans, *Linear Goal Programming*, Petrocelli Books, USA, 1994.

[34] L.D. Smith, A. Subramanian, R.M. Nauss, and R. Beck, "Developing Outsourcing Strategies for MIS (Managament Information Systems): A Mathematical Programming Approach.", *34th Annual Hawaii Int. Conf. on Systems Sciences*, IEEE Computer Society, 2003, pp. 250-258.

[35] L.W. Smith, *Project Management Body Of Knowledge (PMBOK)*, Project Management Institute, 2000.

[36] R. Solingen, "Measuring the ROI of Software Process Improvement", *IEEE Software*, vol.21, no. 3, May/Jun 2004, pp. 32- 38.

[37] A. Stoica, "A decisional Framework in Software Design", *First Workshop on Economics-Driven Software Engineering Research (EDSER-1), affiliated with the 21st Int. Conf. on Software Engineering (ICSE'99)*, IEEE Computer Society, USA, 1999, pp. 1-6.

[38] T. Thompson, "Project Selection Analysis Techniques for Program Managers", *2nd International Workshop on Engineering Management for Applied Technology (EMAT'01)*, IEE Computer Society, 2001, pp. 25-28.

[39] H.P. Williams, *Model Solving in Mathematical Programming*, Editorial Wiley, England, 1993.

[40] W.L. Winston, *Introduction to Mathematical Programming, Applications and Algorithms*, Second edition, International Thompson Publishing, USA, 1995.