# Contenido

# PRÓLOGO

La Ingeniería del Software es uno de las cuestiones más olvidadas y rechazadas entre la mayoría de estudiantes de Ingeniería Informática. Sin embargo, es una de las temáticas más importantes en el conjunto global de la titulación de Ingeniería Informática, y fundamental para el correcto desarrollo de productos software.

Erróneamente se identifica la creación de software sólo con su programación y prueba, dando de lado el proceso anterior de planificación y diseño, así como de asignación de recursos y de esfuerzo.

Quizá para productos sencillos y de carácter académico, como los realizados por los estudiantes de Ingeniería Informática, la asignación de dinero, tiempo y recursos no sea decisiva para la calidad y corrección del programa, pero en una empresa dedicada al desarrollo software, cuyos productos necesitan una gran cantidad de programadores, y el tiempo de desarrollo y los gastos deben ser prefijados, la estimación y planificación del proyecto son fundamentales e indispensables.

Este es uno de los motivos que han hecho decantarme por este proyecto, ya que además de ser una materia fundamental de cara a las salidas laborales, siempre he sentido especial predilección e interés por ella. Además existe una gran necesidad en las empresas de desarrollo software de estimar y planificar proyectos, siendo este uno de los hechos que más me ha hecho decidirme por su elección.

Como ya he indicado anteriormente, la Ingeniería Software es vital para la correcta elaboración de productos software. Antes de comenzar la tarea puramente de programación y creación del programa es necesario realizar una estimación y una previsión sobre la cantidad de tiempo que llevará su realización así como del esfuerzo y el personal necesario para completarlo. Para llevar a cabo esta estimación existen una serie de modelos que actualmente son

utilizados por las empresas de desarrollo software, y cuyos resultados poseen una calidad dependiente del entorno. Por ello este estudio intentará esclarecer cuáles son los mejores modelos así como mostrar una comparativa entre ellos.

Todos estos modelos y herramientas pueden ser directamente aplicados a la estimación de parámetros como esfuerzo y costo de proyectos de tipo software. Pero en este proyecto, aparte de dar una visión general sobre estos utensilios, se va a proponer una alternativa para la estimación, basada en una técnica muy extendida como es la estimación por analogía.

La estimación por analogía se basa en la selección de proyectos análogos a aquel que quiere ser estimado previamente completados y almacenados en una base de datos. Con los datos de estos proyectos se realizarán una serie de operaciones y ajustes encaminados a determinar cuáles son los más parecidos al proyecto bajo cuestión, para poder ofrecer una estimación de su esfuerzo.

Entre estas operaciones, encontramos principalmente los cálculos de similitud entre proyectos, que en nuestro lugar será la medida del coseno, por su fácil aplicación sobre información vectorial, que es la que almacenaremos en nuestra base de datos, y por los buenos resultados derivados de su aplicación en otros ámbitos.

Por tanto, como principal objetivo de este proyecto se plantea un estudio de los diferentes modelos de estimación, sobre todo de los basados en la analogía, proponiendo una alternativa para realizar la estimación del esfuerzo utilizando para ello algoritmos de clasificación de instancias.

De entre todos los algoritmos existentes para la clasificación, se ha escogido *kNN*, por su sencillez, eficiencia y por los buenos resultados que consigue al ser aplicado en otros ámbitos. Además, es un algoritmo directamente aplicable a nuestro propósito de estimar esfuerzo en proyectos software basándonos en analogías.

Una vez que *kNN* ha preparado el camino, puede aplicarse ya la estimación por analogía. Este es un sencillo método que aporta como valor de

esfuerzo estimado aquel que poseen los proyectos más parecidos al que está siendo sometido a examen. Si embargo, no considerando como definitivo este valor, hemos aplicado un ajuste para tener en cuenta las diferencias existentes entre proyectos. El ajuste seleccionado está basado en un fenómeno estadístico conocido como regresión hacia la media, que en nuestro caso hace que los valores de esfuerzo de los proyectos se muevan hacia la media del conjunto de los más cercanos, por lo que este ajuste desplazará el valor estimado hacia la media del conjunto, según una serie de medidas y correlaciones.

Proponemos además el uso de MMRE como medida del error cometido en las estimaciones hechas. Ésta es una medida comúnmente usada en el ámbito de la Ingeniería Software, que presenta además un cálculo sencillo y sin grandes necesidades de computación, y que es un excelente indicador de la exactitud y precisión con la que nuestra alternativa realiza las estimaciones.

Estas medidas y algoritmos serán evaluados sobre un conjunto de datos reales sobre proyectos software ya completados, almacenados en una base de datos disponible a través de la web, creada y mantenida por la asociación ISBSG (*Internacional Software Benchmarking Standards Group*). En esta base de datos se guardan datos sobre 2047 proyectos de diversos países, con un gran número de características.

Para poder concretar cuáles son los parámetros más adecuados de los algoritmos empleados en la propuesta de estimación presentada, se realizará un estudio de su efectividad considerando muy diversas configuraciones, determinándose finalmente cuál es la más apropiada.

La ejecución de diversas pruebas nos va a permitir realizar todo tipo de comparaciones entre los resultados obtenidos, en función de los parámetros de configuración que poseen los algoritmos empleados. Estas comparativas nos conducirán a la determinación de los valores idóneos que deben tomar estos parámetros.

Mediante la realización de este Proyecto fin de Carrera se pretende por tanto, realizar una comparativa entre los diferentes modelos de estimación en

proyectos software y analizar más en profundidad la técnica de estimación por analogía, complementada con un sencillo ajuste matemático que aporta buenos resultados.

Para completar estos objetivos se debe en primer lugar realizar una profunda búsqueda de bibliografía que muestre el estado actual de la disciplina de la estimación software, y que indique las modificaciones hechas a técnicas ya anticuadas para poder adaptarlas a los nuevos requerimientos del desarrollo software. Además, será crucial la obtención de datos reales sobre proyectos software ya completados, para poder realizar con ellos las pruebas y experimentos diseñados.

Una vez obtenidos los datos necesarios, se procederá a la implementación de la alternativa planteada a lo largo del proyecto, y su posterior ejecución y obtención de resultados.

# CAPÍTULO 1

# REVISIÓN DE LOS MODELOS DE ESTIMACIÓN SOFTWARE

# CAPÍTULO 1

# REVISIÓN DE LOS MODELOS DE ESTIMACIÓN SOFTWARE

Contenidos

Cuanto mayor es la necesidad de utilizar el software en cualquier actividad humana, mayor es también la complejidad y dificultad de implementación que éste adquiere.

Aunque cada vez existan más técnicas que facilitan el diseño y desarrollo de los sistemas, las nuevas exigencias de los usuarios y los nuevos dominios de aplicación generan nuevos problemas. Esto crea la necesidad de prestar cada vez más atención a los procesos de planificación, medición y estimación de diversos parámetros software, antes de comenzar con el desarrollo propiamente dicho.

Esta necesidad convierte la gestión de los proyectos software en una tarea de vital importancia, siendo uno de sus puntos clave la estimación del esfuerzo y el tiempo necesario para la finalización de estos productos. Sin unas buenas estimaciones de estos parámetros, comienzan a surgir imprevistos y problemas durante el desarrollo que imposibilitan la entrega del producto final en el plazo acordado, con sus consiguientes aumentos en los gastos y pérdidas económicas.

En las empresas de desarrollo software se realizan continuamente estimaciones acerca de los productos que desarrollan. Esto hace que surjan cada vez más modelos y técnicas para poder hacer predicciones sobre el tamaño o el esfuerzo necesario para completar estos productos, ya que en dichas organizaciones se adaptan en ocasiones los métodos tradicionales según sus necesidades.

En este capítulo se pretende dar una visión general sobre los distintos métodos de estimación de proyectos software, abarcando multitud de modelos desde los más utilizados a otros más novedosos y actuales, fruto de las nuevas técnicas y entornos de programación.

Previo a este análisis, se explicará cómo se llevan a cabo las estimaciones en las organizaciones, proceso que engloba el cálculo del tamaño del software así como el esfuerzo y tiempo requerido para su creación.

Como cabe esperar, en estos procesos de previsión y estimación intervienen una gran cantidad de métricas y medidas, que serán analizadas y explicadas brevemente, para posibilitar así una mejor comprensión de los modelos.

Por último, y antes de profundizar en cada modelo, se plantean una serie de problemas que afectan a las estimaciones en la actualidad, para dar así una visión más realista de estos procesos.

## 1.1 PROCESO DE ESTIMACIÓN

En la industria en general, es necesario calcular y estimar el esfuerzo y el tamaño del proyecto en etapas muy tempranas del desarrollo del mismo. Sin embargo, si en el ámbito software se hacen las estimaciones en estas fases iniciales, dichas previsiones pueden estar basadas en unos requerimientos erróneos o incompletos, por lo que disminuye mucho su fiabilidad.

El proceso de estimación del coste de un producto software está formado por un conjunto de técnicas y procedimientos que se usan en la organización para poder llegar a una predicción fiable. Éste es un proceso continuo, que debe ser usado y consultado a lo largo de todo el ciclo de vida del proyecto. Se divide en los siguientes pasos [AGA01]:

- Estimación del tamaño.
- Estimación del costo y del esfuerzo.
- Estimación de la programación temporal.
- Estimación de la cantidad de recursos computacionales.
- Asunción de riesgos.
- Inspección y aprobación.
- Redacción de informes de estimación.

- Medición y perfeccionamiento del proceso.

En la ilustración 1 pueden observarse los distintos pasos del proceso de estimación, así como las interacciones existentes entre ellos.

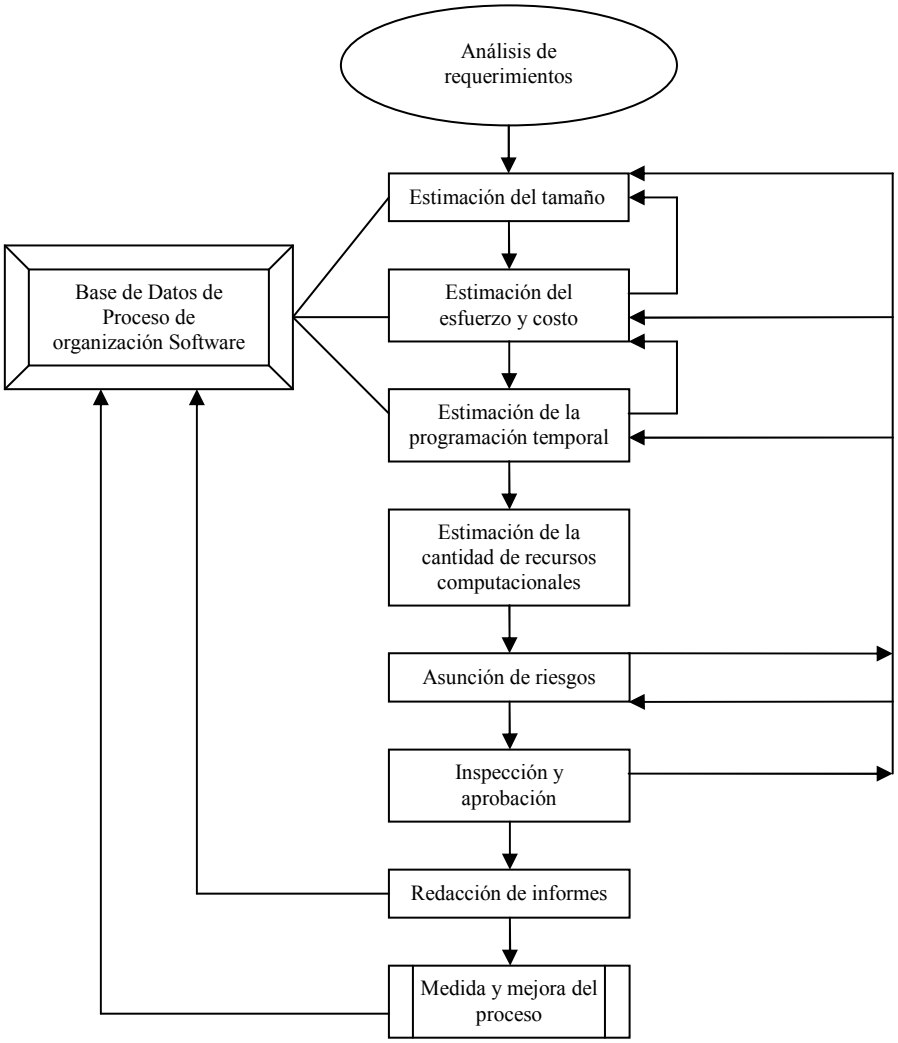


**Ilustración 1: Actividades de la estimación de proyectos software**

Todas estas estimaciones necesarias están basadas en probabilidades debido a la influencia de factores externos de difícil control. Además de estas probabilidades, es necesario recurrir a información histórica, que debe ser fácilmente accesible y disponible para la organización en cualquier momento.

Por desgracia, no siempre se presta la suficiente atención al cuidado de estos datos, por lo que en ocasiones acceder a ellos no resulta sencillo.

La cantidad de esfuerzo y tiempo dedicada a la estimación depende del tamaño del proyecto, del equipo de desarrollo y del objetivo a cumplir. La naturaleza del proyecto y el entorno en el que se desarrolla son factores determinantes en esta tarea, y afectan en gran medida al método de estimación que se utilice.

De una manera general podemos afirmar que existen dos maneras diferentes de estimar el presupuesto y el tiempo para un proyecto software: usando modelos de costo y usando razonamiento basado en similitud. En ambas opciones es necesario recurrir a información histórica y de proyectos anteriores previamente almacenados en bases de datos.

Existen cuatro puntos fundamentales sobre los que se apoya la estimación:

- Las consideraciones y opiniones de los profesionales de la materia, basada en la experiencia y la madurez de los gestores de proyecto, los cuales tendrán que adivinar y predecir el tiempo de realización del proyecto o su coste.

- La participación de expertos, cuyas opiniones no deben ser consideradas y abordadas como las de los profesionales y gestores de proyecto, ya que los expertos no pertenecen a la organización y pueden estar no familiarizados con las prácticas propias de la organización.

- La utilización de factores estándar de tiempos, calculados y establecidos a partir de proyectos anteriores.

- Por último el empleo de fórmulas y funciones, que implica la existencia de datos cuantitativos que representen una buena aproximación a la estimación. Además no deben existir dudas acerca de la fiabilidad y seguridad del predictor usado.

## 1.2. MEDIDAS Y MÉTRICAS DEL SOFTWARE

En el campo de la ingeniería del software, se suele hablar indistintamente de "métricas" y de "medidas", pero sin embargo existen diferencias entre estos términos. Una medida indica cuantitativamente algún atributo de proceso o de producto (extensión, cantidad, dimensiones, capacidad, tamaño, etc.). Una métrica es definida por el *Glosario de estándares del* IEEE (*Institute of Electrical and Electronics Engineers)* [IEE93] como una "medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado".

Si se recopila un sólo tipo de datos, como por ejemplo el número de errores dentro de un componente, se ha establecido una medida. Una métrica de software relaciona de alguna manera las medidas individuales. Siguiendo nuestro ejemplo, podríamos tratar el número de errores encontrados en cada revisión o prueba [PRE98].

Los ingenieros de software, a partir de las medidas, elaboran métricas que les proporcionan información para poder controlar el proceso o el proyecto software. Aquí radica la diferencia entre estos términos.

Hecha esta aclaración, podemos pasar a tratar algunas métricas de proyecto usadas por los ingenieros del software, todas ellas de vital importancia durante los procesos de estimación. Suelen ser recopiladas de proyectos anteriores, de manera que se comparan sucesivamente con los valores actuales de esfuerzo y tiempo, y así se puede supervisar la corrección de las estimaciones realizadas.

Otras métricas de gran utilidad pueden ser el número de páginas de documentación, las horas de revisión, el número de errores detectados, etc. [PRE98].

Referente a las medidas del software, también vitales en la estimación, podemos clasificarlas en directas o indirectas: entre las directas se encuentran las líneas de código, velocidad de ejecución, tamaño de la memoria, etc. y hacen

referencia a atributos cuantitativos del producto; como indirectas se podrían citar la funcionalidad, la calidad, complejidad, eficiencia, fiabilidad, etc.

En general, pueden elaborarse infinidad de métricas a partir de medidas. Por ejemplo el tamaño lo podemos asociar al número de líneas de código o de documentación, al tamaño del equipo de desarrollo, a la cantidad de dinero, etc., y obtener a partir de estas medias multitud de métricas: cantidad de dinero por línea, líneas de código por persona, páginas de documentación por persona, etc.

## 1.3. CLASIFICACIÓN DE LOS MODELOS DE ESTIMACIÓN

Los diferentes modelos de estimación para proyectos pueden ser clasificados de diversas maneras, de entre las cuales se deben destacar las aportadas por dos autores principales [CHA96]:

*Según Basili*

Según este autor existen tres modelos:

- Modelos con una o varias variables estáticas, que se basan en aplicar funciones y constantes a algunas propiedades del proyecto, por ejemplo el LOC (*Lines Of Code*).
- Modelos con varias variables dinámicas, que miden el tiempo del proyecto frente a su costo, usando una distribución obtenida de manera empírica.
- Modelos teóricos con algoritmos prediseñados, que se basan en una hipótesis para realizar una predicción a través de una función obtenida teniendo en cuenta información histórica.

*Según Kitchenham*

Kitchenham propone una clasificación más simple para estos modelos, basada en distinguir entre aquellos que especifican la relación entre varios parámetros de costo, llamados modelos de restricción, y los que predicen el valor de un parámetro de costo, llamado modelos de factor empírico.

Como ejemplo de esta clasificación podemos encontrar entre los modelos de restricción los de Putnam, Jensen, Cocomo, o Parr.

Algunos modelos de factor empírico son Esfuerzo de Cocomo, Wolverton, Softcost, Estimacs, o Price.

*Otras clasificaciones*

Otras maneras más simples de clasificar los modelos de estimación distinguen tres categorías básicas:

- Juicio experto: aunque no es un método en el sentido estricto de obtener resultados de una manera explícita y repetitiva, es una de las prácticas más extendidas, en las que se combinan las opiniones de varios expertos para obtener estimaciones, como por ejemplo el método Delphi.

- Modelos algorítmicos: se basan en fórmulas y parámetros con los que realizan las estimaciones. Son también muy conocidos y empleados en la actualidad, encontrándose en este grupo el modelo COCOMO o puntos de función.

- Analogía: puede ser tomada como una variante del juicio experto, ya que también intervienen las opiniones de los estimadores, pero a diferencia de este tipo, necesita caracterizar claramente el proyecto que se va a estimar y almacenar los proyectos previos para buscar entre ellos el más parecido al actual.

## 1.4. PROBLEMAS PRESENTADOS POR LA ESTIMACIÓN DEL ESFUERZO DE PROYECTOS SOFTWARE

A pesar de la amplia variedad de métodos y paquetes de software que apoyan la estimación para proyectos de desarrollo software, la planificación es aún una tarea muy difícil.

En la práctica, todos los modelos de estimación presentan una serie de problemas:

- Dan unas predicciones muy deficientes cuando se aplican sobre conjuntos de datos independientes entre sí.
- Están basados en apreciaciones subjetivas acerca de las variables de entrada, y por tanto ofrecen resultados muy diferentes cuando se aplican sobre el mismo problema, ya que estas valoraciones están muy influidas por el medio en el que se desarrolla el proyecto y su entorno.
- Estiman el tiempo o costo sólo para las últimas etapas del proceso de desarrollo, olvidándose en la mayoría de los casos las iniciales.
- Usan KLOC (*Thousand of Lines of code)* o KDSI *(Thousand of Delivered Source Instruction)* como factores, para estimar el tiempo o costo inicial de la fase de codificación, constituyéndose como base para la estimación del resto de etapas, a pesar de aquellas recomendaciones que afirman que estos factores no son del todo apropiados.
- El factor KDSI no tiene en cuenta los recursos disponibles para el equipo de desarrollo (como herramientas software), ni las características propias del equipo, como por ejemplo su grado de experiencia. Sin embargo, como resultados de diversos estudios y experimentos, se demuestra cómo la capacidad del equipo de desarrollo es uno de los factores que más afectan al tiempo total de realización del proyecto.

## 1.5. MODELOS DE ESTIMACIÓN SOFTWARE

Hay dos aproximaciones principales para estimar el volumen de un proyecto software: paramétricas y heurísticas [AGA01].

### 1.5.1. Aproximaciones paramétricas

Las estimaciones paramétricas usan como predictor una métrica determinada fácilmente al comienzo del ciclo de vida del software.

Una buena métrica nos asegura una correcta correlación con el esfuerzo del proyecto, y facilita la estimación del mismo.

Las dos métricas más comúnmente usadas son el número de líneas del código fuente (SLOC. *Source Lines of Code*) y los puntos de función (FP, *Function Points*).

Para sistemas dirigidos por datos, domina el uso de puntos de función. Es con sistemas dirigidos por computación con los que es más frecuente el uso de SLOC, ya que la mayor parte del tiempo lo consume el procesador ejecutando operaciones, relegando el papel de los datos y del usuario como fuente de información sobre el tamaño del sistema.

➢ COCOMO

COCOMO (*COnstructive COst MOdel*) fue propuesto por Barry Boehm en 1981, y es un ejemplo claro de micromodelo con tecnología ascendente. Además, es el modelo de estimación de costes más utilizado.

El principal cálculo en el modelo COCOMO es el uso de la ecuación del esfuerzo para estimar el número de personas o de meses necesarios para desarrollar el proyecto. El resto de resultados del modelo se derivan de esta medida.

$$\text{ESFUERZO} = A \times \text{TAMAÑO}^{B}$$

En esta sencilla ecuación, A y B son constantes obtenidas empíricamente y dependientes del modo de desarrollo. La estimación del tamaño del proyecto queda expresada en SLOC, definida con las siguientes reglas:

- Líneas de código creadas por el personal del proyecto (no el creado por generadores de aplicaciones).
- Una instrucción es una línea de código.
- Las declaraciones se toman como instrucciones.
- Los comentarios no se cuentan como instrucciones.

En COCOMO es fundamental el término que hace referencia al modo de desarrollo del proyecto, que influye directamente en el costo y duración del mismo, y que puede ser:

- Modo orgánico: cuando el proyecto es desarrollado en un ambiente familiar y estable, y en el que el producto es similar a otros desarrollados previamente. Son además productos pequeños y poco novedosos, con unos requerimientos definidos y poco rígidos.
- Modo empotrado: para proyectos caracterizados por unos requerimientos y restricciones poco flexibles, que requieren un gran esfuerzo de innovación. También poseen un elevado nivel de complejidad hardware.
- Modo semiacoplado: es un modelo para proyectos que presentan características intermedias entre el orgánico y el empotrado, con un equipo formado por miembros de distintos niveles de experiencia, que trabajan sobre un conjunto de requisitos más o menos flexibles.

COCOMO está definido en términos de tres modelos diferentes: modelo básico, intermedio y detallado, que reflejan el nivel de detalle considerado a la hora de realizar la estimación del coste. El modelo básico provee una estimación inicial poco refinada, el intermedio la modifica utilizando una serie de multiplicadores relacionados con el proyecto y el proceso, y el detallado realiza diferentes estimaciones para cada fase del proyecto. Éste último es el más complejo, y cuenta con más factores que influyen en el software, consiguiendo así mejores estimaciones.

*a) Modelo básico*

Este modelo hace sus previsiones en función del tamaño del proyecto medido en miles de líneas de código (KSLOC). Estas estimaciones de esfuerzo calculadas, se expresan en Personas - Mes (PM), considerando 152 horas de trabajo mensuales, y 19 días de trabajo por mes.

La ecuación básica del esfuerzo queda definida de la siguiente forma [AGA01]:

| MODO DE DESARROLLO | ECUACIÓN BÁSICA DEL ESFUERZO |
|---|---|
| Orgánico | Esfuerzo = 2.4 * $KSLOC^{1.05}$ |
| Semiacoplado | Esfuerzo = 3.0 * $KSLOC^{1.12}$ |
| Empotrado | Esfuerzo = 3.6 * $KSLOC^{1.20}$ |

**Tabla 1: Ecuación básica del esfuerzo en COCOMO (modelo básico)**

Las estimaciones sobre el tiempo son:

| MODO DE DESARROLLO | ECUACIÓN BÁSICA DEL TIEMPO |
|---|---|
| Orgánico | Tiempo de desarrollo = 2.5 * $Esfuerzo^{0.38}$ |
| Semiacoplado | Tiempo de desarrollo = 2.5 * $Esfuerzo^{0.35}$ |
| Empotrado | Tiempo de desarrollo = 2.5 * $Esfuerzo^{0.32}$ |

**Tabla 2: Ecuación básica del tiempo en COCOMO (modelo básico)**

*b) Modelo intermedio*

El modelo intermedio y el detallado, usan un factor de ajuste del esfuerzo (EAF), y los coeficientes de las ecuaciones de esfuerzo varían notoriamente con respecto a los del modelo básico, permaneciendo el cálculo del tiempo sin cambios.

| MODO DE DESARROLLO | ECUACIÓN BÁSICA DEL ESFUERZO |
|---|---|
| Orgánico | Esfuerzo = EAF * 3.2 * $KSLOC^{1.05}$ |
| Semiacoplado | Esfuerzo = EAF * 3.0 * $KSLOC^{1.12}$ |
| Empotrado | Esfuerzo = EAF * 2.8 * $KSLOC^{1.20}$ |

**Tabla 3: Ecuación básica del esfuerzo en COCOMO (modelo intermedio y detallado)**

Este modelo obtiene unos mejores resultados, en gran parte debido al establecimiento de 15 factores de costo, que determinan la duración y el coste del proyecto.

En la siguiente tabla se pueden apreciar los diferentes factores de costo establecidos en COCOMO [AGA01].

| | Factor de costo | | MUY BAJO | BAJO | MEDIO | ALTO | MUY ALTO | EXTR. ALTO |
|---|---|---|---|---|---|---|---|---|
| ATRIBUTOS DEL PRODUCTO | Fiabilidad requerida | RELY | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | - |
| | Tamaño de base de datos | DATA | - | 0.94 | 1.00 | 1.08 | 1.16 | - |
| | Complejidad | CPLX | 0.7 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| ATRIBUTOS DEL COMPUTADOR | Tiempo de ejecución | TIME | - | - | 1.00 | 1.11 | 1.30 | 1.66 |
| | Memoria principal | STOR | - | - | 1.00 | 1.06 | 1.21 | 1.56 |
| | Permanencia de la máquina virtual | VIRT | - | 0.87 | 1.00 | 1.15 | 1.30 | - |
| | Tiempo de ejecución | TURN | - | 0.87 | 1.00 | 1.07 | 1.15 | - |
| ATRIBUTOS DEL PERSONAL | Capacidad del analista | ACAP | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | - |
| | Experiencia en aplicaciones | AEXP | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | - |
| | Capacidad del programador | PCAP | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | - |
| | Experiencia con máquinas virtuales | VEXP | 1.21 | 1.10 | 1.00 | 0.90 | - | - |
| | Experiencia con el lenguaje | LEXP | 1.14 | 1.07 | 1.00 | 0.95 | - | - |
| ATRIBUTOS DEL PROYECTO | Prácticas de programación modernas | MODP | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 | - |
| | Uso de herramientas software | TOOL | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | - |
| | Programación necesaria del desarrollo | SCED | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 | - |

**Tabla 4: Factores de costo en COCOMO**

El Factor de Ajuste del esfuerzo (EAF), se calcula multiplicando los correspondientes factores asociados a cada variable de costo.

Otro de los motivos que justifica que el modelo intermedio obtenga mejores resultados que el básico es la posibilidad de poder dividir el sistema en componentes. Esto implica que valores como SLOC o los factores de costo puedan ser escogidos en función de determinadas partes del sistema, no considerándolo como un todo. De esta manera se pueden hacer estimaciones sobre el personal, el costo y la duración de cada componente, permitiéndose así elaborar distintas estrategias de desarrollo.

*c) Modelo detallado*

La única diferencia existente entre el modelo intermedio y el detallado, es que este último utiliza diferentes multiplicadores del esfuerzo para cada fase del proyecto.

Las seis fases que define COCOMO son: requerimientos, diseño del producto, diseño detallado, codificación y pruebas de unidad, integración y test y por último mantenimiento. Las fases comprendidas entre el diseño del producto y la integración y test son llamadas fases de desarrollo.

➢ COCOMO II

COCOMO II es un modelo que permite estimar el coste, el esfuerzo y el tiempo cuando se planifica una nueva actividad de desarrollo software, y está asociado a los ciclos de vida modernos. Fue desarrollado a partir de COCOMO, incluyendo actualizaciones y nuevas extensiones más adecuadas a los requerimientos de los ingenieros software.

Está construido para satisfacer aquellas necesidades expresadas por los estimadores software, como por ejemplo el apoyo a la planificación de proyectos, la previsión de personal de proyecto, replanificación, seguimiento, negociaciones de contrato o la evaluación del diseño.

COCOMO II proporciona una familia de modelos de estimación muy detallados, y que tiene muy en cuenta el tipo información disponible. Este conjunto de modelos se divide en tres:

- Modelo de composición de aplicaciones, para proyectos de construcción de interfaces gráficas de usuario.

- Modelo de diseño preliminar, utilizado para obtener estimaciones sobre el coste de un proyecto, antes de que esté determinada por completo su arquitectura.

- Modelo post-arquitectura, que es el más detallado, y debe ser usado una vez determinada la arquitectura al completo.

El usar un modelo u otro depende del nivel de detalle del proyecto, de la fidelidad requerida de las estimaciones, y de la definición de los requerimientos y de los detalles de la arquitectura. De una manera general, podemos determinar en qué momentos es más adecuado la utilización de un modelo u otro:

- El modelo de composición de aplicaciones se aplicará sobre todo en las primeras fases o ciclos en espiral.

- El modelo de diseño preliminar es útil para las siguientes fases o ciclos espirales, en los que se incluye la exploración de arquitecturas alternativas o estratégicas de desarrollo incremental.

- Una vez que el proyecto está listo para desarrollar y sostener un sistema especializado, el submodelo de Post-arquitectura proporciona información más precisa de los controladores de coste de entradas y permite cálculos de coste más exactos.

Sendos modelos usan la siguiente ecuación:

$$PM = 2.45 * EAF * Tamaño^{B}$$

Donde EAF *(Effort Adjustment Factor)* es el producto de siete multiplicadores de esfuerzo para el modelo de desarrollo temprano, y diecisiete para el modelo post-arquitectural.

B es un factor que toma valores comprendidos entre 1.01 y 1.26, que viene dado por $B = 1.01 + \sum w_i$, siendo $\sum w_i$ la suma de cinco factores que causan efecto en la economía del proyecto, y que son:

- Existencia de precedentes al proyecto.
- Flexibilidad del desarrollo.
- Resolución de riesgos.
- Cohesión del equipo.
- Madurez del proceso.

Los multiplicadores de esfuerzo para COCOMO II son [AGA01]:

| MODELO DE DESARROLLO TEMPRANO | MODELO POST-ARQUITECTURAL |
|---|---|
| Fiabilidad y complejidad del producto (RCPX) | Fiabilidad requerida del software (RELY)<br>Tamaño de la base de datos (DATA)<br>Complejidad del producto (CPLX)<br>Necesidad de documentación (DOCU) |
| Necesidad de reutilización (RUSE) | Necesidad de reutilización (RUSE) |
| Dificultad de la plataforma (PDIF) | Restricción de tiempo de ejecución (TIME)<br>Restricción de almacenamiento principal (STOR)<br>Estabilidad de la plataforma (PVOL) |
| Capacidad del personal (PERS) | Capacidad del analista (ACAP)<br>Capacidad del programador (PCAP)<br>Continuidad del personal (PCON) |
| Experiencia del personal (PREX) | Experiencia en aplicaciones (AEXP)<br>Experiencia con la plataforma (PEXP)<br>Experiencia con el lenguaje y herramientas (LTEX) |
| Facilidades (FCIL) | Uso de herramientas software (TOOL)<br>Desarrollo distribuido (SITE) |
| Programación (SCED) | Programación requerida del desarrollo (SCED) |

**Tabla 5: Multiplicadores de esfuerzo para COCOMO II**

➢ COCOTS

Los modelos COCOTS *(Constructive Commercial Off-The-Shelf)* se utilizan en proyectos caracterizados principalmente por dos aspectos: su código fuente no está disponible para el desarrollador de la aplicación, y su evolución no es supervisada por el mismo.

El desarrollo de software utilizando COCOTS implica cuatro actividades:

*Valoración de los componentes COTS*

Esta actividad permite escoger productos software COCOTS para ser integrados en el sistema global. Los candidatos viables se determinan en función de una serie de características:

- Requerimientos funcionales: capacidad ofrecida.
- Requerimientos de rendimiento: restricciones de tiempo y tamaño.
- Requerimientos no funcionales: coste, formación, instalación, mantenimiento y fiabilidad.

Esta valoración se realiza en dos fases. En la primera, se agrupan componentes viables, eliminando aquellos que no se adecuan al contexto actual, y en la segunda se hace una selección final de productos, en función de las características que presentan.

*Adaptación de componentes*

En esta fase se configuran los productos COTS para ser usados en un contexto específico, realizando principalmente inicialización de parámetros, interfaces de usuario, distribución de los informes de salida, instalación de protocolos de seguridad, etc.

Para poder medir y calibrar el esfuerzo empleado en realizar estas labores de adecuación e integración de productos COTS, se asigna a cada tarea de adecuación un valor concreto y establecido según la complejidad de dicha tarea.

El esfuerzo total de la fase de adaptación es una función dependiente del número de productos COTS cuya modificación haya sido necesaria.

La dificultad del proceso de adaptación se basa en los siguientes aspectos:

- Especificación de parámetros.
- Desarrollo de scripts.
- Especificación de informes de entrada o salida, de interfaces gráficas de usuario.
- Inicialización de protocolos de seguridad y acceso
- Fiabilidad de las herramientas de adaptación utilizadas.

Teniendo en cuenta cada uno de estos aspectos, se asignan los niveles de complejidad a cada producto COTS. Sumando los esfuerzos necesarios para cada producto modificado, se estima el esfuerzo total del proceso de adaptación.

*Código de unión*

En esta fase se ensamblan los diferentes productos COTS, para pasar a formar parte de un sistema mayor. Puede ser necesaria la unión de productos COTS entre sí o bien con un sistema de un nivel superior.

Los multiplicadores de esfuerzo utilizados por COCOTS son [AGA01]:

| | |
|---|---|
| FACTORES PERSONALES | Experiencia del responsable de integración con el producto<br>Capacidad personal del responsable de integración<br>Experiencia del responsable de integración<br>Continuidad personal de responsable de integración |
| FACTORES DE PRODUCTOS COTS | Madurez del producto<br>Fiabilidad del distribuidor<br>Complejidad de la interfaz<br>Soporte del distribuidor<br>Formación aportada por el distribuidor |
| FACTORES DE APLICACIÓN Y DEL SISTEMA | Restricciones sobre la fiabilidad del sistema<br>Complejidad de la interfaz de la aplicación<br>Restricciones para el desarrollo técnico de COTS<br>Portabilidad de la aplicación |
| FACTOR DE ESCALA NO LINEAL | Aplicación de Ingeniería Arquitectural |

**Tabla 6: Multiplicadores de esfuerzo para COCOTS**

➢ Puntos de función

El análisis por puntos de función fue desarrollado a mediados de los setenta para intentar superar las dificultades asociadas al uso de las líneas de código como medida del tamaño del software, y para aportar un mecanismo para predecir el esfuerzo necesario para el desarrollo de un proyecto software.

Las medidas realizadas mediante puntos de función son totalmente independientes de la tecnología. Sea cual sea el lenguaje utilizado, el método de desarrollo o la plataforma hardware usada, el número de puntos de función permanece siempre constante.

El análisis por puntos de función puede ser usado también para comparar herramientas, entornos o lenguajes entre sí, bien dentro de una misma organización o entre varias. Esta es una de las grandes ventajas de este tipo de análisis.

Los puntos de función ofrecen una medida de la funcionalidad entregada por la aplicación [PRE06], es decir, las características funcionales aportadas por la aplicación una vez creada. Como este valor no se puede medir directamente, se debe derivar de manera indirecta mediante otras medidas.

Las tareas asociadas a los puntos de función deben ser incluidas como parte del proyecto, y por tanto deben ser planeadas y programadas.

*Principales componentes del análisis por puntos de función*

Desde el momento en que los sistemas computacionales comenzaron a interactuar con otros computadores, fue necesaria la existencia de un límite o frontera alrededor de cada uno de estos sistemas antes de clasificar componentes. Este límite debe ser dibujado desde el punto de vista del usuario, es decir, delimitando el proyecto o aplicación a ser medida y la parte externa de la misma, o dominio del usuario. Una vez que se ha establecido dicha frontera, los componentes pueden ser clasificados, ordenados y tasados.

Los cinco componentes principales son [PRE98]:

- Entradas de usuario (External Inputs o EI): son procesos elementales en los que los datos atraviesan la frontera desde fuera hacia dentro de la aplicación. Estos datos pueden provenir de formularios de entrada por pantalla, o de otras aplicaciones, y puede ser información de control o relativa al negocio. Normalmente, los datos relativos al problema son usados para actualizar ciertos archivos relativos a la lógica interna del programa (Internal Logical Files).

- Salidas de usuario (External Outputs o EO): al contrario que las EI, son procesos que atraviesan la frontera desde dentro hacia la parte externa de la aplicación o sistema. Esta información, que también actualiza ILF's mediante la creación de informes o ficheros de salida, está orientada al programa.

- Peticiones de usuario (External Queries o EQ): son procesos de entrada y salida que solicitan información contenida en los ILF's y en los ficheros de interfaces, sin realizar ningún tipo de actualización de ficheros. Son entradas interactivas que producen la generación de alguna respuesta del software inmediata en forma de salida interactiva.

- Ficheros de lógica Interna (Internal Logical Files o ILF): archivos de datos relacionados entre sí que residen en el interior del sistema, como parte de una gran base de datos o como ficheros independientes.

- Ficheros Externos de Interfaz external Interface Files o EIF): Son datos que residen fuera de la aplicación y son mantenidos por otras diferentes. Son ILF para otras aplicaciones, y son usados sólo como referencia. Engloban todas las interfaces legibles por la máquina que se utilizan para transmitir información a otro sistema.

Una vez que los componentes de la aplicación han sido clasificados en los cinco grupos principales (EI's, EO's, EQ's, ILF's, EIF's), se les califica como

de nivel bajo, medio o alto. Esta es una tarea algo subjetiva, aunque las organizaciones han desarrollado criterios para esta calificación, como por ejemplo las siguientes tablas:

| COMPLEJIDAD DE LAS ENTRADAS | | | |
|---|---|---|---|
| Nº tipos de archivos referenciados | Nº tipos de elementos de datos incluidos | | |
| | 1-4 | 5-15 | ≥ 16 |
| 0-1 | Baja | Baja | Media |
| 2-3 | Baja | Media | Alta |
| ≥ 4 | Media | Alta | Alta |

**Tabla 7: Calibración de la complejidad de las entradas en Puntos de Función**

| COMPLEJIDAD DE LAS SALIDAS Y PETICIONES | | | |
|---|---|---|---|
| Nº tipos de archivos referenciados | Nº tipos de elementos de datos incluidos | | |
| | 1-5 | 6-19 | ≥ 20 |
| 0-1 | Baja | Baja | Media |
| 2-3 | Baja | Media | Alta |
| ≥ 4 | Media | Alta | Alta |

**Tabla 8:Calibración de la complejidad de las salidas y peticiones en Puntos de Función**

| COMPLEJIDAD DE LOS ARCHIVOS E INTERFACES | | | |
|---|---|---|---|
| Nº tipos de archivos referenciados | Nº tipos de elementos de datos incluidos | | |
| | 1-19 | 20-50 | ≥ 51 |
| 0-1 | Baja | Baja | Media |
| 2-5 | Baja | Media | Alta |
| ≥ 6 | Media | Alta | Alta |

**Tabla 9: Calibración de la complejidad de los archivos interfaces en Puntos de Función**

Hecha la clasificación, se calcula la cuenta total mediante la siguiente tabla [PRE98]:

| PARÁMETRO DE MEDICIÓN | | FACTOR DE PONDERACIÓN | | | |
|---|---|---|---|---|---|
| | Cuenta | Simple | Medio | Complejo | |
| Número de entradas de usuario | _____ x | 3 | 4 | 6 | = _____ |
| Número de salidas de usuario | _____ x | 4 | 5 | 7 | = _____ |

| | | | | | |
|---|---|---|---|---|---|
| Número de peticiones de usuario | _____ x | 3 | 4 | 6 | = _____ |
| Número de archivos | _____ x | 7 | 10 | 15 | = _____ |
| Número de interfaces externas | _____ x | 5 | 7 | 10 | = _____ |
| CUENTA TOTAL | | | | | ∑ = _____ |

**Tabla 10: Cuenta total de los puntos de función**

Con esta cuenta total se pueden calcular ya los puntos de función, según la siguiente relación:

$$PF = cuenta\_total \times [0{,}65 + 0{,}01 \times \sum F_i]$$

Donde $\sum F_i$ es la suma de los valores de ajuste de la complejidad, según las respuestas dadas a las 14 preguntas destacadas a continuación [PRE98]:

| EVALUACIÓN DE LOS FACTORES DE COMPLEJIDAD | | | | | |
|---|---|---|---|---|---|
| No influencia | Incidental | Moderado | Medio | Significativo | Esencial |
| 0 | 1 | 2 | 3 | 4 | 5 |

**Tabla 11: evaluación de los factores de complejidad en Puntos de Función**

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2. ¿Se requiere comunicación de datos?
3. ¿Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?
6. ¿Requiere el sistema entrada de datos interactiva?
7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?

8. ¿Se actualizan los archivos maestros de forma interactiva?
9. ¿Son complejas las entradas, las salidas, los archivos o las peticiones?
10. ¿Es complejo el procesamiento interno?
11. ¿Se ha diseñado el código para ser reutilizable?
12. Están incluidas en el diseño la conversión y la instalación?
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14. ¿Se ha diseñado la aplicación para facilitar los cambios y ser fácilmente utilizada por el usuario?

### 1.5.2. Aproximaciones heurísticas

➢ Bottom - up

Esta filosofía de estimación divide el proyecto en pequeñas partes, aplicándole a cada una evaluación directa del esfuerzo. El valor obtenido se normaliza en función del peso y la importancia de cada parte. Posteriormente se irán uniendo estas partes en subsistemas.

La principal ventaja que aporta este sistema, es que las personas que realizan la estimación son las mismas que van a hacer el desarrollo. Sin embargo, tiene el inconveniente de que normalmente no se cuenta el tiempo de estimación, que suele ser importante. Otra desventaja sería que si se descubren errores en las partes finales del proceso, habría que rechazar prácticamente toda la estimación hecha hasta el momento [AGA01].

➢ Top - down

En primer lugar se estiman los costes a nivel de sistema, de una manera general, para posteriormente ir obteniendo los costes de cada uno de los diferentes subsistemas identificados [AGA01].

➢ Juicio experto: técnica Delphi

La técnica Delphi se basa en la obtención de un consenso de un grupo de expertos, que expresan sus opiniones y ofrecen estimaciones sobre el proyecto en cuestión.

Los expertos expresan sus opiniones mediante unos formularios que le son entregados y que rellenan de manera totalmente anónima. Estos cuestionarios contiene cada una de las estimaciones realizadas a nivel personal por cada componente del grupo.

Estos cuestionarios son entregados al coordinador del proceso, quien se encarga de comunicarle a cada experto la opinión de los demás, junto con datos estadísticos sobre todas las estimaciones entregadas.

Una vez que cada estimador posee esta información, vuelven a rellenar los cuestionarios y los entregan nuevamente al coordinador. Este proceso se repetirá hasta que el coordinador encuentre un consenso y una opinión generalizada acerca de la estimación del proyecto.

Con este método, se producen juicios de consenso de una manera rápida y eficaz. Además es un método estructurado para estudiar la anticipación de sucesos futuros, permitiendo la incorporación de factores no racionales, algo muy útil cuando existan variables sociales que puedan tener impacto en la previsión.

Sin embargo, ofrece una serie de desventajas, relacionadas con la elaboración de los cuestionarios y la selección de los expertos. Para salvar la primera dificultad, se debe realizar un diseño mucho más elaborado y detallado de los cuestionarios entregados a los expertos. El segundo inconveniente puede ser solucionado mediante una elección aleatoria de entre todo el universo de

estimadores con experiencia, para evitar así selecciones tendenciosas y dirigidas.

### 1.5.3. Estimación para proyectos orientados a objetos

Lorenz y Kidd [LOR94] diseñaron un modelo de estimación exclusivo para proyectos software orientados a objetos. La importancia de este paradigma de programación hace que sea comentado junto al resto de técnicas de estimación.

Esta técnica se desarrolla siguiendo los siguientes pasos:

- Se hacen estimaciones siguiendo cualquier método propio de aplicaciones convencionales.
- Posteriormente se aplica el modelado de análisis orientado a objetos, desarrollando casos de uso.
- A partir de este modelo de análisis, se calcula el número de clases clave.
- Determinar el tipo de interfaz a implementar, asociándole un multiplicador según la siguiente tabla:

| Tipo de interfaz | Multiplicador |
|---|---|
| Sin IUG | 2.0 |
| Interfaz basada en texto | 2.25 |
| IUG | 2.25 |
| IUG compleja | 3.0 |

**Tabla 12: Multiplicadores para interfaces en estimación de proyectos OO**

- Multiplicar el número de clases clave por el multiplicador, para obtener una estimación del número de clases soporte.
- Multiplicar el número total de clases por el número promedio de unidades de trabajo por clase (15-20 personas-día por clase, según Lorenz y Kidd).

- Comprobar de manera cruzada la estimación basada en clase al multiplicar el número promedio de unidades de trabajo por caso de uso [PRE06].

### 1.5.4. Estimación para desarrollo ágil

Se denomina proyecto ágil de software a aquel que reúne las siguientes características [PRE06]:

- Es muy difícil predecir aquellos requerimientos de software que persistirán y cuáles cambiarán con el tiempo.
- Tienen el diseño y la construcción intercalados, de modo que se prueban los modelos de diseño conforme se crean.
- El análisis, el diseño y la implementación no son predecibles.

La estimación en este tipo de proyectos sigue los siguientes pasos:

- Cada escenario de usuario se considera por separado, y se descompone en un conjunto de funciones y tareas de ingeniería software necesarias para desarrollarlo.
- Cada tarea se estima por separado, con cualquier método, y el volumen mediante LDC o puntos de función.
- Las estimaciones de cada tarea se suman para obtener una estimación de cada escenario.
- El volumen del escenario se traduce en esfuerzo mediante la aplicación de datos históricos.
- Las estimaciones de esfuerzo de cada escenario se suman con el fin de obtener el esfuerzo total de cada incremente de software.

### 1.5.5. Estimación para proyectos de ingeniería web

Los proyectos de ingeniería web adoptan normalmente el modelo de proceso ágil. Por este motivo, es frecuente utilizar una medición de puntos de función modificada en conjunto con los pasos de la estimación en proyectos ágiles.

Para calcular los puntos de función adaptados se utilizan los siguientes valores de dominio de información:

- Por *entradas* se toma cada pantalla o formato de entrada (por ejemplo CGI-beans).
- *Salidas* son cada página web estática o cada guión de página dinámica (ASP, ISAPI, etc.).
- *Tablas* son cada tabla lógica en la base de datos más cada objeto XML, si éste es empleado para almacenar datos en un archivo.
- Las *interfaces* siguen tomándose de igual manera que en puntos de función tradicional-
- *Consultas* son cada interfaz publicada externamente, tales como las referencias externas DCOM o COM.

Estos puntos de función calculados son un indicador razonable del volumen de una aplicación web.

### 1.5.6. Herramientas software de apoyo a la estimación

➢ PRICE-S

El modelo PRICE-S (Programming Review of Information Costing and Evaluation Software) fue desarrollado por RCA PRICE Systems.

Su filosofía se basa en evitar que los conceptos e ideas sobre las que se apoya el modelo son inaccesibles para el usuario, como es el caso de COCOMO y Puntos de función, que son presentados a los usuarios como una caja negra. Por

ello, el usuario de PRICE-S envía sus peticiones a un ordenador que trabaja a tiempo compartido situado en EE.UU, Reino Unido o Francia, y devuelve sus estimaciones inmediatamente [HEE92].

➢Modelo Putnam

Putnam desarrolló este modelo basándose en el trabajo de Norden, quien observó las distribuciones de frecuencia de diversos proyectos de IBM, y analizó cuánta gente se asignaba para el desarrollo y mantenimiento de productos software a lo largo de su ciclo de vida.

Las curvas obtenidas por Norden coincidían muy bien con la curva Rayleigh, descubrimiento totalmente empírico al que no consiguió encontrar una explicación [HEE92].

➢Before You Leap (BYL)

BYL es un paquete comercial, que se basa para sus estimaciones en los puntos de función y en el modelo COCOMO.

En primer lugar, obtiene el total de puntos de función de la aplicación. Esta cantidad la traduce a miles de líneas de código (KSLOC) en función del lenguaje de programación que se vaya a utilizar. Este número de líneas de código se utiliza para realizar la estimación con COCOMO [HEE92].

➢Estimacs

Es un paquete comercial de estimación que se compone de 9 módulos de diversa aplicación: asunción de riesgos, puntos de función, estimación, etc.

El más valioso de estos módulos es el que realiza las estimaciones de esfuerzo de desarrollo y mantenimiento, usando un método desconocido para los usuarios, quienes se limitan a contestar a 25 preguntas acerca de la complejidad de la organización y del tamaño del producto a desarrollar [HEE92].

➢SPQR-20

SPQR ofrece estimaciones acerca de la duración, costo y esfuerzo de desarrollo de productos software, así como de costes de mantenimiento.

Utiliza puntos de función para establecer el tamaño del programa, y plantea al usuario una serie de preguntas sobre el proyecto a desarrollar.

Su potencia radica en las preguntas que realiza al usuario (entre 10 y 100) y en una enorme base de datos en la que almacena datos sobre proyectos pasados [HEE92].

➢BIS-estimator

Es una herramienta diferente a las demás, basada en el conocimiento. Sin embargo, no se califica certeramente como tal, ya que sus creadores ocultan gran parte de su funcionamiento.

En primer lugar hace una primera estimación basándose en las respuestas que da el usuario a una serie de preguntas. Posteriormente, hace una estimación global del proyecto para cada fase del mismo, haciendo comparaciones con otros proyectos previos indicados por el usuario.

Su principal ventaja radica en el hecho de hacer estimaciones diferentes para cada una de las fases del proyecto [HEE92].

La naturaleza de esta herramienta se parece a la alternativa planteada en este proyecto para la estimación de esfuerzo de proyectos software: a partir de información sobre proyectos completados obtenida de los ingenieros software, se buscan proyectos parecidos al que se quiere estimar, mediante comparaciones y cálculos de distancia.

➢ Costar

Desarrollado por Sofstar Systems, emplea el modelo COCOMO II para desarrollar estimaciones software.

➢ Cost Xpert

Herramienta software que integra múltiples modelos de estimación y una base de datos histórica de proyectos.

➢Estimate Professional

Basado en COCOMO II y el modelo de Slim.

➢Knowledge Plan

Utiliza la entrada de puntos de función como principal controlador para un paquete de estimación completo.

➢ SEER/SEM

Herramienta que proporciona además de estimaciones, análisis de sensibilidad y valoración de riesgos.

## 1.6. COMPARATIVA DE LOS DIFERENTES MODELOS DE ESTIMACIÓN DE PROYECTOS SOFTWARE

Para un gestor de proyecto existen una serie de cuestiones clave cuya respuesta será analizada comparando los diferentes modelos explicados en secciones anteriores del presente documento:

- Qué modelo de estimación se debe usar.
- Qué medida de tamaño de producto debe tomarse: líneas de código o puntos de función.

En general, se puede afirmar que no existe un método que sea mejor que cualquier otro. Todos presentan una serie de ventajas e inconvenientes que hacen que la elección sea difícil. Gran cantidad de parámetros del proyecto y del ámbito en el que se desarrolla hacen que el estimador se decida sobre un método u otro.

Por un lado, los modelos puramente algorítmicos, que precisan de unas entradas concretas y utilizan ciertos factores y variables para producir las estimaciones, son métodos muy objetivos, sujetos a operaciones matemáticas y que producen resultados similares con proyectos igualmente parecidos.

Sin embargo, no prestan ninguna atención a circunstancias excepcionales que puedan ocurrir en torno al desarrollo del proyecto, y rechazan también las opiniones subjetivas de expertos o de desarrolladores doctos en su materia (solamente intervienen para calibrar factores en niveles, tarea en ocasiones muy difícil de realizar).

Esta falta de atención hacia factores externos y personales, se puede compensar con la eficiencia que presentan los cálculos necesarios para la estimación.

Globalmente podemos concluir que los métodos algorítmicos son idóneos en proyectos con escasas alteraciones accidentales y del entorno, así como para equipos de desarrollo estables, que se enfrentan a productos relativamente sencillos y con pocos factores de costo, que faciliten la aplicación del método en cuestión.

Por otro lado, si el estimador se decanta por el juicio experto, tendrá gran cantidad de opiniones subjetivas, y se tendrán en cuenta las circunstancias especiales en las que se crea el producto software, pero es excesiva la dependencia de los expertos, así como de la elección de los mismos y de la comunicación establecida con ellos.

Además, en ocasiones puede ser muy difícil adoptar la postura de un experto y hacer uso de su conocimiento, llegando el momento en el que sus

decisiones se vuelvan demasiado generales e inaplicables a la situación que atraviese el proyecto.

A pesar de estos problemas, esta técnica resulta ideal para las primeras fases de desarrollo del producto, cuando las especificaciones son difusas y deben ser continuamente adaptadas. Las estimaciones hechas en estas etapas son unos muy buenos indicadores del tiempo y del esfuerzo que puede llegar a ser necesitado.

Acerca de los análisis descendentes o top-down, podemos destacar su eficiencia y su capacidad de centrarse en diferentes partes del sistema. Resulta crucial la división que se haga del proyecto, así como las estimaciones hechas para cada unidad. Una mala estimación en alguna parte puede hacer que al sumarse todas se obtengan resultados no muy fiables y que conlleven errores en las previsiones. Por desgracia, realizar esta división no es una tarea fácil.

Por el contrario, las filosofías ascendentes (bottom-up) ofrecen un mayor nivel de detalle y estabilidad, pero pueden caer en el grave error de sobrepasar y desestimar ciertos niveles del sistema, cuyo coste deba ser valorado. La solución a este error pasa por una fuerte supervisión del proceso, que acarrea gran cantidad de esfuerzo y atención.

Si tratamos con un producto fácilmente separable en partes, y se pueden hacer estimaciones cómodas sobre dichas porciones, se aconseja el uso de técnicas descendentes.

Si ocurre que dividir el sistema en partes supone un gran esfuerzo y trabajo a la organización, se debe rechazar el análisis por partes, y optar por filosofías de estimación ascendentes.

No obstante, una de las opciones más recomendada es utilizar varias técnicas simultáneamente, comparando los resultados e iterando los procesos de estimación. La combinación de técnicas dependerá del nivel de detalle a alcanzar. Como posibles combinaciones encontramos, por ejemplo, estimaciones descendentes junto con juicio experto, aplicando analogía si se

dispone de información suficiente, o métodos algorítmicos cuyas entradas sean aportadas por filosofías ascendentes.

En la siguiente tabla pueden observarse de manera resumida todas estas ventajas e inconvenientes comentados anteriormente:

| MODELO DE ESTIMACIÓN | VENTAJAS | INCONVENIENTES | APLICACIÓN IDÓNEA |
|---|---|---|---|
| Modelos algorítmicos | ✓ Entradas y parámetros concretos<br>✓ Objetividad<br>✓ Eficiencia en cálculos | × No prestan atención a circunstancias excepcionales<br>× Rechazan opiniones subjetivas | Proyectos con escasas alteraciones accidentales, con equipos de desarrollo estables y productos sencillos |
| Juicio experto | ✓ Gran cantidad de opiniones subjetivas<br>✓ Consideración de circunstancias excepcionales | × Dependencia de los expertos<br>× Posturas de expertos difíciles de adoptar<br>× Decisiones inaplicables | Primeras fases de desarrollo del producto |
| Análisis descendentes | ✓ Eficiencia<br>✓ Capacidad de centrarse en diferentes partes del sistema | × La división que se haga del proyecto es decisiva y no siempre fácil de determinar | Proyectos con divisiones claras |
| Análisis ascendentes | ✓ Mayor nivel de detalle | × Se pueden desestimar ciertos niveles del sistema<br>× Gran cantidad de esfuerzo y atención | Proyectos muy difíciles de dividir |

**Tabla 13: Ventas e inconvenientes de los modelos de estimación**

Otra de las cuestiones presentadas a los estimadores software hace referencia a la medida del tamaño del proyecto.

Normalmente se identifica SLOC como el mejor indicador de tamaño, fácil de medir (aunque difícil de estimar), e intrínsecamente asociado al lenguaje de programación utilizado. Desgraciadamente, el uso de lenguajes muy avanzados, de programación basada en componentes reutilizables, o basados en objetos, convierten esta medida en algo más inexacta, debido a la existencia de código generado automáticamente, o código ya desarrollado en anteriores proyectos.

En estas situaciones aparece puntos de función como un excelente indicador de tamaño, ya que es totalmente independiente de la tecnología empleada y puede ser calculado fácilmente en fases tempranas del proyecto. Su

desventaja reside en ser más abstracto y no directamente derivable del producto.

Estas indicaciones hacen decantarnos por SLOC cuando se traten proyectos en lenguajes sencillos, y con no demasiadas funcionalidades, y puntos de función para productos con gran cantidad de componentes ya implementados y código no generado manualmente por los desarrolladores.

En la siguiente tabla se esquematizan estas ventajas e inconvenientes citadas:

| MEDIDA DEL TAMAÑO | VENTAJAS | INCONVENIENTES | APLICACIÓN IDÓNEA |
|---|---|---|---|
| SLOC | ✓ Fácil de medir<br>✓ Asociado al lenguaje de programación | × Difícil de estimar<br>× No apropiado con lenguajes OO o basado en componentes | Proyectos con lenguajes sencillos y con no mucha funcionalidad |
| Puntos de función | ✓ Independiente de la tecnología<br>✓ Calculado fácilmente en fases tempranas del proyecto | × es más abstracto que SLOC<br>× No es directamente derivable del producto | Productos con gran cantidad de componentes reutilizados |

**Tabla 14: Ventajas e inconvenientes de SLOC y Puntos de función como indicadores de tamaño**

# CAPÍTULO 2

# ESTIMACIÓN POR ANALOGÍA

# CAPÍTULO 2

# ESTIMACIÓN POR ANALOGÍA

Contenidos

Tal y como se ha expuesto en el capítulo anterior, existe una gran variedad de técnicas y filosofías de estimación, que permiten a las organizaciones prever y calcular la cantidad de esfuerzo y dinero necesarios para llevar a cabo el proyecto software en cuestión.

De entre esta variedad, haciendo una comparativa resaltaba el hecho de no existir un método mejor que cualquier otro, sino que todo es función del entorno y de las características del proyecto, así como del conocimiento que se tiene acerca del mismo.

Para un estimador puede resultarle en ocasiones más cómodo y fácil aplicar un método de estimación que simule el comportamiento humano en la realidad. Este método es denominado estimación por analogía, y se demuestra que obtiene resultados equiparables e incluso mejores que los obtenidos mediante otras formas más extendidas, como el modelo COCOMO o la estimación por puntos de función.

La estimación basada en analogía es el proceso por el cual se localizan uno o más proyectos previos similares al que está siendo desarrollado y se derivan estimaciones a partir de ellos [CHI06].

Es fundamental en este proceso medir el grado de similitud del proyecto bajo examen con respecto a los almacenados en la base de datos de históricos. Así podemos identificar los proyectos más parecidos al actual y estimar a partir de ellos.

No obstante, resulta obvio pensar que es prácticamente imposible que exista uno o varios proyectos exactamente iguales al que queremos estimar, por lo que no se debe utilizar directamente el esfuerzo asociado a estos proyectos. La solución aportada consiste en aplicarle algún tipo de ajuste, en función de la distancia existente entre dicho proyecto obtenido como el más parecido y el proyecto sobre el que vamos a hacer una estimación.

Para ajustar estos valores de esfuerzo de proyectos ya completados existen multitud de técnicas, siendo una de las más apropiadas la búsqueda del

conjunto óptimo de factores a tener en cuenta para obtener la estimación final. Los algoritmos más potentes en este campo son los genéticos, cuyo desarrollo simula el proceso de evolución natural de las especies.

## 2.1. JUSTIFICACIÓN DEL USO DE LA ESTIMACIÓN POR ANALOGÍA

El uso de la analogía para la estimación está apoyado y confirmado por diversos estudios empíricos, que demuestran que se obtienen resultados más exactos y consistentes que los obtenidos con métodos como COCOMO y puntos de función.

Gran cantidad de autores han examinado a fondo esta forma de estimación, obteniendo unos muy buenos resultados. Estos análisis se recogen en [ANG00], [HUA06], [JOR04], o [SHE97]

Además de sus buenos resultados, es uno de los métodos más usados por las organizaciones, tal y como se demuestra en un estudio con más de 600 empresas de la industria del software [HEE92].

## 2.2 PROCESO DE ESTIMACIÓN POR ANALOGÍA

Para poder realizar una estimación sobre un proyecto software utilizando la analogía, es necesario en primer lugar determinar cuál o cuáles de los proyectos ya completados y almacenados en una base de datos son más parecidos al que queremos estimar.

Para realizar esta tarea, debe determinarse el criterio para establecer las distancias entre proyectos. Existen multitud de medidas de similitud entre ejemplares, siendo las más usadas las distancias de Euclides, de Manhattan y de Minkowski. Ésta última viene determinada por la siguiente fórmula:

$$\mathrm{d}(P_{xi}, P_{yi}) = \sqrt[e]{\sum_{i=1}^{m} \left| P_{xi} - P_{yi} \right|^{e}} \ ,$$

donde $P_x$ es el proyecto que está siendo estimado y $P_y$ es un proyecto de la base de datos; $P_{xi}$ es el valor que toma el factor *i* en el proyecto $P_x$, y $P_{yi}$ el valor del mismo factor *i* pero en el proyecto $P_y$; *m* es el número de factores de esfuerzo de los proyectos.

Las distancias de Euclides y de Manhattan pueden tomarse como casos particulares de Minkowski cuando *e* vale 2 y 1, respectivamente. Por tanto, la expresión de la distancia de Manhattan quedaría:

$$\mathrm{d}(P_{xi}, P_{yi}) = \sum_{i=1}^{m} \left| P_{xi} - P_{yi} \right|$$

Y la distancia Euclídea:

$$\mathrm{d}(P_{xi}, P_{yi}) = \sqrt{\sum_{i=1}^{m} \left( P_{xi} - P_{yi} \right)^{2}}$$

Esta última expresión es la más común, ya que representa la distancia geométrica entre dos puntos del espacio Euclídeo, muy usada en Matemáticas.

Mediante esta fórmula de la distancia, podemos obtener ya el conjunto de proyectos más parecidos al que está bajo examen. Lógicamente, este grupo estará formado por los ejemplares con la menor distancia al proyecto en cuestión.

El número de proyectos análogos a considerar puede variar entre ciertos valores. No obstante, debe tenerse en cuenta que escoger un número muy pequeño puede conducir a considerar solamente proyectos muy independientes del resto, y un número demasiado grande disminuye la potencia ofrecida por la analogía, al tomar como similares proyectos que apenas lo son.

Algunos estudios afirman que no existen demasiadas diferencias en la exactitud de las estimaciones al considerar grupos de proyectos análogos de diferentes tamaños [CHI06] [JEF00] [SHE97].

Sí es cierto que un incremento en el número de análogos provoca un aumento del error relativo [CHI06], y los números pequeños deben ser tomados con conjuntos de datos igualmente pequeños.

Una vez obtenido el conjunto de proyectos similares, sólo falta ajustar los valores de esfuerzo de dichos proyectos para poder estimar el del proyecto en cuestión. Existen multitud de técnicas y métodos de ajuste, comentándose a continuación dos de los más frecuentes.

### 2.2.1. Ajuste mediante algoritmos genéticos

Este enfoque pretende escoger el mejor conjunto de características de los proyectos más cercanos al que se quiere examinar, y realizar las estimaciones con ellas. Para seleccionar dicho grupo se deben calcular los errores producidos al estimar con sus componentes, y buscar aquel conjunto que minimice este error en la estimación.

En principio se podrían obtener todos lo posibles conjuntos de características y seleccionar aquel que produzca el menor error en la precisión. Esto resulta fácil para proyectos con pocos factores de esfuerzo, pero por desgracia esta no es la situación más normal. Es frecuente que contengan multitud de características que hacen demasiado ineficiente la aplicación de un algoritmo de fuerza bruta. A este problema se le han buscado muchas soluciones, siendo la más apropiada el uso de algoritmos genéticos.

Un algoritmo genético apropiado para esta selección estaría formado por individuos que representan conjuntos de características de proyectos ya completados. Cada ejemplar tendría asociado un valor de error en la estimación, de manera que con los operadores genéticos y de mutación se sucederían las generaciones, buscando siempre el individuo con menor error.

Una vez identificado, se estimaría el proyecto en cuestión sumándole al valor de esfuerzo del proyecto más parecido el error del individuo más fuerte de la población. Con esto hemos conseguido identificar el conjunto de características más apropiado para tener en cuenta a la hora de estimar, minimizando el error cometido [CHI06].

Este tipo de ajuste de las estimaciones es costoso en términos de computación y tiempo, y no ha sido empleado en la alternativa propuesta en este proyecto porque pensamos que con algoritmos y técnicas más simples se pueden alcanzar resultados equiparables. Por ello, se ha implementado una técnica más eficiente y rápida, basada en un fenómeno estadístico explicado a continuación.

### 2.2.2. Ajuste por "regresión hacia la media"

La "regresión hacia la media" es un fenómeno estadístico ligado a aquellas variables cuyo coeficiente de correlación es inferior al 100 %. Fue descubierto por Sir Francis Galton en el siglo XIX, cuando al medir la altura de padres e hijos observó que los padres muy altos tenían hijos que no lo eran tanto, y los padres de menor altura tenían hijos más altos que ellos.

La regresión hacia la media provoca que aquellas muestras que presentan características extremas dentro de una población (*outliers*), se correspondan con otros individuos de características opuestas y también extremas, tendiéndose siempre al mantenimiento de la media del conjunto.

Existen estudios que demuestran que la regresión hacia la media es muy frecuente en situaciones de la vida cotidiana, pero que en realidad pasa desapercibida para la mayoría de la gente.

Centrándonos en la estimación de proyectos software por analogía, el fenómeno de la regresión hacia la media aporta un ajuste del esfuerzo asociado a aquellos proyectos que se han seleccionado como los más parecidos al que se pretende estimar.

La relevancia de este tipo de ajuste está en parte debida a la aleatoriedad ligada de manera intrínseca a cualquier proyecto de desarrollo software. Siempre existen eventos de difícil previsión, o incluso totalmente imprevisibles, que tienen consecuencias drásticas en el proceso de desarrollo, convirtiendo en erróneas aquellas estimaciones que en principio se tomaron por fiables.

Esta aleatoriedad provoca también que aquellos proyectos que presentan características similares no tengan valores de esfuerzo y tiempo igualmente parecidos, por lo que surge la necesidad de ajustar estos valores de los proyectos análogos al que se va a estimar.

Antes de realizar un ajuste por regresión hacia la media (RTM), deben valorarse dos importantes aspectos: la exactitud de la estimación hecha a partir de los proyectos análogos, y las situaciones extremas que pueden presentar sus características.

Si el modelo de estimaciones es muy seguro, no se realizarían ajustes en los valores de esfuerzo de los proyectos análogos, por muy diferentes que sean del resto de proyectos considerados. Si por el contrario el modelo no es muy fiable, es razonable buscar un valor estimado que se aproxime a la media del conjunto. La dimensión de este ajuste, depende de la naturaleza de los proyectos seleccionados como análogos: será mayor cuanto más extremos sean los valores de sus características.

El método de ajuste aplicado se basa en la fórmula desarrollada por Galton:

$$\text{Pr}\,oductividad = PCA + (M - PCA) \times (1 - c)\,,$$

siendo PCA la productividad de los proyectos análogos, M la media del conjunto y c la correlación entre la productividad de los proyectos análogos y el actual [JOR03].

Esta fórmula tiene en cuenta la distancia a la media de los proyectos considerados análogos (M - PCA) y la correlación histórica entre los valores de productividad del proyecto estimado y los más similares (c).

La aplicación de este ajuste no es una tarea sencilla. Su uso requiere: que los proyectos incluidos en el cálculo de la correlación *c* sean representativos para la exactitud de la estimación, que el grupo de proyectos similares tiendan hacia el mismo valor medio *M*, y que dicho valor sea conocido.

De todas formas el ajuste RTM *(Regression Toward the Mean)* es muy aconsejable, sobre todo para proyectos que presenten unos valores muy extremos de productividad. Incluso en situaciones en las que la correlación no sea muy buena, debe aplicarse este ajuste.

## 2.3. VENTAJAS E INCONVENIENTES DE LA ESTIMACIÓN POR ANALOGÍA

Como se ha citado en anteriores ocasiones, la estimación por analogía es uno de los métodos más utilizados en las organizaciones de desarrollo software. Esto hace que existan multitud de estudios en los que se analiza la exactitud y fiabilidad de sus estimaciones. Estas investigaciones nos pueden ayudar a encontrar ventajas e inconvenientes planteadas por el uso de la estimación por analogía.

Entre sus ventajas podemos citar las siguientes:

- El método de razonamiento empleado es similar al realizado por la mente humana. Las búsquedas por analogía son familiares para el estimador, que se siente cómodo usando esta alternativa.

- Es un modelo muy apropiado para aquellas situaciones en las que el dominio es difícil de modelar. Se sabe que existen multitud de factores que influyen en la cantidad de esfuerzo necesaria para completar un proyecto software. Sin embargo, lo que no siempre

se sabe es cómo interactúan estos factores entre sí, o a qué factores se les debe otorgar mayor peso e importancia. Es en estas situaciones cuando la estimación por analogía ofrece unos muy buenos resultados sin necesidad de tener un modelo claro o de saber exactamente cómo el esfuerzo está relacionado con otros factores de proyecto.

- Puede ser usado teniendo un conocimiento parcial del proyecto que se quiere estimar. La información que más importa es la de proyectos ya completados y almacenados que se usan de conjunto de búsqueda para encontrar los más parecidos al que está siendo estimado.

- Puede suavizar problemas debidos a la calibración. La estimación basada en la analogía proporciona buenas estimaciones incluso cuando se utilizan datos procedentes de otra organización, siempre y cuando el proyecto análogo sea encontrado en el conjunto de datos que se use para la estimación, y se midan las variables siguiendo un criterio uniforme y consistente en todo el conjunto, sea cual sea su procedencia.

- Mitiga problemas asociados con los *outliers* (elementos extremos). La existencia de proyectos muy diferentes al resto no afecta en las estimaciones hechas por analogía, si el proyecto a estimar no es en sí un *outlier*. Si por el contrario tratamos con un proyecto que es un *outlier*, ya que contiene valores muy distintos de sus características, al menos se encontrarán proyectos similares también con características extremas, pudiendo completarse la estimación sin problemas.

Entre las principales desventajas de este método de estimación podemos citar:

- El proyecto seleccionado como análogo puede no ser apropiado para la estimación, ya que algunos de sus factores que afectan al

esfuerzo hayan cambiado radicalmente con el tiempo. Por estos motivos, el mantenimiento y actualización de las bases de datos de proyectos ya completados afecta profundamente al proceso de estimación.

- Un estimador puede usar su juicio experto para excluir aquellos proyectos inapropiados, pero también puede utilizarlo para escoger uno similar a ciegas, sin justificación alguna, existiendo la posibilidad de que también sea inapropiado. Esta característica es propia del razonamiento humano, en el que se observa la tendencia a encontrar evidencias que confirmen nuestras opiniones, y rechazar situaciones contrarias.

- Normalmente en la estimación por analogía se usa la distancia de Euclídes como criterio de selección de proyectos análogos, otorgando el mismo peso a cada factor. A pesar de que los resultados obtenidos son excelentes, no hay nada que indique que este método de pesado sea el idóneo, existiendo la posibilidad de que algunos factores deban tener un mayor peso a la hora de calcular las distancias. Como alternativas la asignación de pesos encontramos el juicio experto, los coeficientes de correlación o los algoritmos genéticos.

- Una vez determinado el proyecto análogo, al estimador se le plantea la cuestión de ajustar su valor de esfuerzo. De nuevo el abanico de posibles ajustes es muy amplio, por lo que la elección de uno u otro se vuelve complicada, sobretodo al tener en cuenta las diferentes características del proyecto que se quiere estimar y del análogo. Además, se pueden escoger como análogos 1, 2 e incluso 3 proyectos, ya que los resultados obtenidos en diversas experimentaciones afirman que apenas influye la elección de este número

Resumiendo, la exactitud y precisión de las estimaciones hechas por analogía se apoya en 4 factores: la selección adecuada de un proyecto análogo, la

estrategia utilizada para encontrarlo, el tratamiento de las diferencias existentes entre el análogo y el proyecto a estimar para realizar predicciones, y la fiabilidad del conjunto de datos usado.

Todas estas ventajas e inconvenientes del uso de la estimación por analogía, se muestran de manera abreviada en la tabla 15.

## 2.4. HERRAMIENTAS PARA LA ESTIMACIÓN POR ANALOGÍA

Hasta ahora se ha seguido un enfoque teórico para explicar la técnica de estimación por analogía de proyectos software. Por otra parte, se ha comentado cómo este método es uno de los más extendidos y utilizados en la actualidad por las empresas de desarrollo software. Como apoyo a estas afirmaciones, se presentan a continuación las herramientas software más utilizadas para los procesos de estimación.

| VENTAJAS | INCONVENIENTES |
|---|---|
| - El método de razonamiento es similar al empleado por la mente humana<br>- Es apropiado para situaciones con un dominio difícil de modelar.<br>- Se puede aplicar sin necesidad de conocer completamente el proyecto a estimar<br>- Suaviza problemas asociados a la calibración.<br>- Tolera la existencia de *outliers*. | - El proyecto seleccionado como análogo puede no ser apropiado para la estimación.<br>- El estimador según su juicio personal puede escoger proyectos análogos no recomendables, o excluir aquellos que sí sean válidos.<br>- Existe controversia sobre cuál es el mejor método de cálculo de similitud.<br>- No existe un mecanismo de ajuste del esfuerzo del proyecto análogo mejor que otro. |

**Tabla 15: Ventajas e inconvenientes de la estimación por analogía**

### 2.4.1. ESTOR

ESTOR es una herramienta de razonamiento basada en casos, utilizada para la estimación de esfuerzo en proyectos software. Fue desarrollada en 1992 por Mukhopadhyay, Vicinanza y Prietula y maneja como métricas los puntos de función y algunos parámetros del modelo intermedio de COCOMO [MUK92].

Los casos utilizados por esta herramienta son proyectos software y cada uno se representa mediante los valores de sus medidas. Las métricas que utiliza son puntos de función y algunos parámetros utilizados con el modelo intermedio de COCOMO. Se recupera un sólo caso como análogo, basándose en el valor de los puntos de función del proyecto a estimar. La solución inicial aportada por ESTOR es el valor de esfuerzo para el proyecto análogo. Sin embargo, para tener en cuenta las diferencias existentes con el proyecto que se va a estimar, se le aplican una serie de ajustes tomando como guía un conjunto de reglas derivadas de las opiniones de un experto.

### 2.4.2. ANGEL

ANGEL (ANaloGy SoftwarE tooL) es una herramienta que también realiza estimaciones basándose en la analogía de proyectos software aunque, en comparación con otras herramientas, dota de una mayor libertad al usuario para configurar a su medida el proceso de estimación.

En primer lugar, el estimador puede escoger cualquier conjunto de datos que esté disponible para realizar la búsqueda de los proyectos análogos. Una vez establecido el conjunto de datos, se seleccionan las variables que se van a tener en cuenta a la hora de buscar analogías. Este conjunto de variables lo puede determinar el usuario, o bien se puede ordenar a ANGEL que encuentre el mejor subconjunto de características a tener en cuenta, minimizándose así los errores y el ruido.

Tras estos pasos ANGEL devuelve un conjunto de proyectos análogos al que se está estimando, ordenados por distancias. El valor de esfuerzo ofrecido es la media de esfuerzo del conjunto de análogos, de modo que si el usuario decide utilizar grupos de un elemento, se dará su valor de esfuerzo como estimación, y se decide que sean por ejemplo 4, se calculará la media de sus esfuerzos.

## 2.5. CRITERIOS DE EVALUACIÓN

Una vez hechas las estimaciones y desarrollados los proyectos, se deben tomar medidas de la exactitud del modelo de estimación empleado. Para esta labor, existen tres magnitudes fundamentales: el MMRE (*mean magnitude of relative error)*, MdMRE (*median magnitud of relative error)*, y Pred(0,25).

La fórmula general del MMRE es:

$$MMRE = \frac{1}{n} \times \sum_{x=1}^{n} \frac{\left| A_x - \hat{E}_x \right|}{A_x},$$

que calcula la media de los errores cometidos siendo $A_x$ el valor actual del esfuerzo para el proyecto que se está estimando, y $\hat{E}_x$ el esfuerzo estimado con el error ya añadido.

Un valor aceptable para esta medida ronda el 25 %, que indica que como media la exactitud de los modelos de estimación aplicados es menor a un 25 % [CHI06].

Otro criterio muy usado el Pred(l), que representa el porcentaje de MRE (*magnitude of relative error*) menor o igual al valor *l*, de entre todos los proyectos. Su fórmula es muy simple:

$$\Pr ed(l) = \frac{k}{n},$$

Siendo *n* el número total de observaciones y *k* el número de observaciones en las que MRE es menor o igual al valor *l*. Normalmente, *l* suele tomar valor 0.25, indicando por tanto Pred(0.25) el porcentaje de proyectos con un MMRE menor o igual al 25 %.

# CAPÍTULO 3

# ESTUDIO DE ALGORITMOS DE CLASIFICACIÓN PARA LA ESTIMACIÓN

# CAPÍTULO 3

# ESTUDIO DE ALGORITMOS DE CLASIFICACIÓN PARA LA ESTIMACIÓN

Contenidos

En el presente capítulo se explicará con detalle la implementación del modelo de estimación por analogía propuesto en el proyecto, basado en la utilización de algoritmos de clasificación de instancias previos a la estimación, y en el ajuste de las estimaciones por regresión hacia la media.

Anteriormente fueron revisados varios ajustes que pueden aplicarse a las estimaciones de proyectos software hechas basándose en la analogía. De entre ellos, para la implementación de este sistema ha sido escogido el ajuste por regresión hacia la media por su sencillez y por los buenos resultados que ofrece en este tipo de estimaciones.

Sin embargo, previo a este ajuste se realizan una serie de pasos que se apoyan en medidas de similitud y en algoritmos de selección de elementos análogos a uno determinado. En nuestra propuesta se ha utilizado como medida de similitud entre proyectos la medida del coseno, muy apropiada para una representación vectorial de los datos. De entre todos los algoritmos de selección de análogos, ha sido implementado el *kNN*, o *k* vecinos más cercanos.

Existen multitud de posibilidades de implementación para este tipo de sistemas, pero se ha creído conveniente la utilización de algoritmos y medidas sencillas y de gran potencia y precisión, con las que se pretenderá alcanzar unas buenas estimaciones, con bajas cotas de error.

Además, los buenos resultados conseguidos por estas medidas de similitud y estos algoritmos de clasificación en otros ámbitos han hecho decantarnos por esta alternativa, pretendiendo alcanzar resultados equiparables a los obtenidos en anteriores investigaciones, con un bajo costo computacional y temporal, gracias a la eficiencia intrínseca de las operaciones y técnicas empleadas.

Como la evaluación de los resultados se hará en función de los errores cometidos al estimar, no es necesario escatimar a la hora de escoger un lenguaje de programación, ya que la eficiencia y el tiempo no van a ser valorados. Por estos motivos, buscándose la potencia y la comodidad de la programación, se ha

escogido el lenguaje Java, que además aporta una buena precisión en todos los cálculos matemáticos requeridos para el desarrollo de este estudio.

En definitiva, la propuesta de este proyecto se basa en la utilización de un algoritmo de clasificación como *kNN* previo a la estimación de esfuerzo en proyectos software, con el objetivo de obtener los ejemplares más parecidos al proyecto objeto de estudio. Una vez obtenido este conjunto de análogos, se realizará una primera estimación según dichos elementos, y este valor calculado será posteriormente ajustado mediante regresión hacia la media, con una simple operación matemática.

En la ilustración 2, podemos observar el proceso seguido por la alternativa de estimación propuesta:

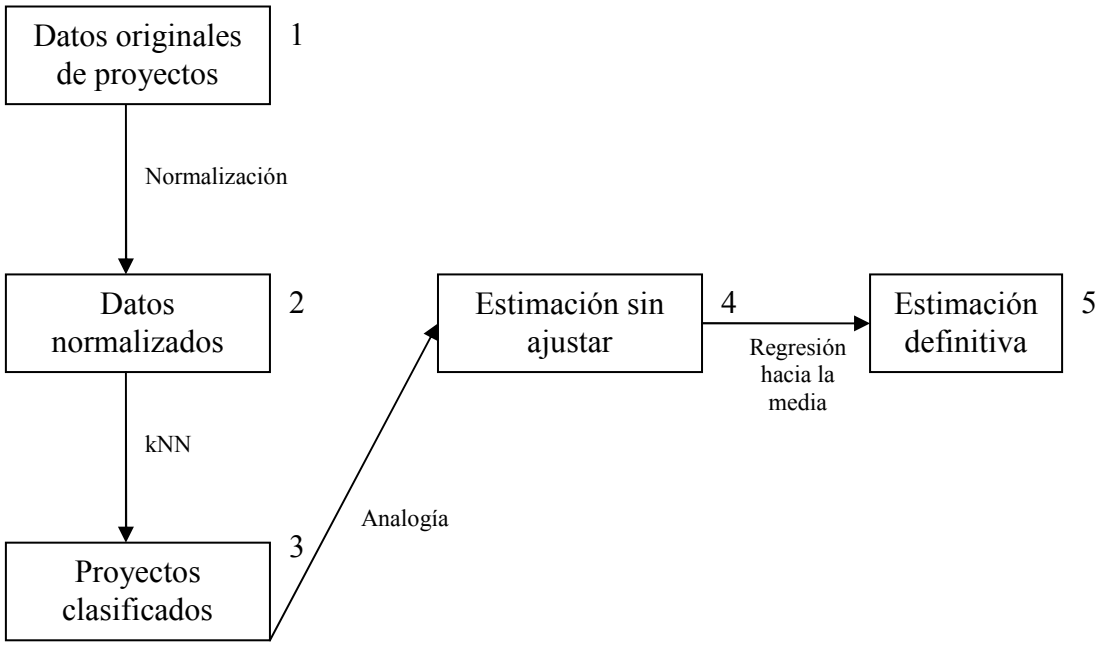


**Ilustración 2: Diagrama de secuencia de la estimación por analogía**

## 3.1. REPRESENTACIÓN DE LOS DATOS

Los datos necesarios para entrenar y realizar las pruebas del modelo de estimación hacen referencia a proyectos software, donde cada elemento está determinado por un identificador de proyecto, posee una serie de parámetros cuantificados numéricamente, y además tiene asociado un esfuerzo total de desarrollo, que será el parámetro sobre el que se realizarán las estimaciones.

Esto hace que los proyectos puedan ser representados mediante una clase, con un identificador de proyecto, un esfuerzo y un vector de valores que representan cada una de las características disponibles de dicho proyecto.

Esta representación de los parámetros de los proyectos nos va a facilitar mucho los cálculos, ya que pueden aplicarse fácilmente medidas de similitud y cercanía entre elementos en un espacio vectorial de $n$ dimensiones, siendo $n$ el número de parámetros conocidos de cada proyecto. Además, para el ajuste que se hará para la estimación es necesario conocer la correlación entre algunas características, por lo que de nuevo la representación vectorial se presenta como idónea para dicha tarea.

En un principio no se disponía de datos reales con los que probar el modelo de estimación planteado, por lo que se decidió generar de manera aleatoria los datos de cada proyecto a utilizar. Con esto conseguimos obtener de una manera rápida la información con la que entrenar y probar el sistema, pero surge un problema: no existe relación entre las características de los proyectos y sus esfuerzos, ya que son todos números aleatorios. Esta situación obliga a utilizar datos reales de proyectos software, que serán igualmente almacenados y tratados.

### 3.1.1. Descripción de los datos utilizados

Los datos sobre proyectos software ya completados han sido obtenidos de una organización internacional, la ISBSG (*Internacional Software Benchmarking Standards Group*), que pone a disposición un repositorio de

datos sobre 2047 proyectos software. La información de cada uno de estos proyectos ha sido voluntariamente proporcionada por sus responsables a la ISBSG, de manera totalmente gratuita.

Para cada proyecto software se han considerado los siguientes atributos:

- Identificador de proyecto: la ISBSG asigna a cada proyecto completado un identificador único.
- Valor ajustado de puntos de función: este valor final de puntos de función ha sido ajustado según el valor del atributo explicado a continuación.
- Valor de ajuste de los puntos de función: es un número utilizado para ajustar la cuenta de puntos de función de cada proyecto.
- Esfuerzo total del proyecto expresado en horas. Este es el valor que será estimado por nuestra alternativa.
- Método de conteo del esfuerzo: este parámetro indica la manera en que se ha tomado el número de horas empleadas para desarrollar el proyecto (conteo diario, porcentaje de horas dedicadas al proyecto, tiempo de desarrollo sólo).
- Nivel del recurso: este parámetro indica qué parte de la organización ha sido la que ha empleado el número de horas indicado en el parámetro del esfuerzo total para completar el proyecto (equipo de desarrollo, cómputo, usuarios finales, etc.).
- Tamaño máximo de los equipos involucrados en el desarrollo del proyecto.
- Tipo de desarrollo: puede ser nuevo, de mejora o hecho a partir de otro proyecto anterior.
- Plataforma de desarrollo: toma valores como PC, *main frame* o plataforma intermedia.

- Tipo de lenguaje empleado: de primera generación, segunda, tercera, etc.
- Lenguaje de programación.
- Utilización de alguna base de datos.
- Utilización de herramientas *Upper CASE*: herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.
- Utilización de herramientas *Lower CASE*: herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación.
- Utilización de herramientas *Coger CASE* no generadoras de código.
- Uso de herramientas CASE ya integradas en el sistema.
- Uso de alguna metodología específica de programación.
- Tiempo de desarrollo transcurrido, expresado en meses.
- Tiempo en el que el proyecto estuvo detenido por cualquier razón, expresado en meses.
- Número de defectos muy graves detectados en el primer mes de uso del producto.
- Número de defectos graves detectados en el primer mes de uso del producto.
- Número de defectos leves detectados en el primer mes de uso del producto.
- Indica si el proyecto es parte de un paquete software.

- Esfuerzo empleado en la planificación del proyecto, expresado en horas.
- Esfuerzo empleado en la especificación del proyecto, expresado en horas.
- Esfuerzo empleado en la construcción del producto, expresado en horas.
- Esfuerzo empleado en la fase de test del producto, expresado en horas.
- Esfuerzo empleado en la implementación del producto, expresado en horas.
- Entradas de usuario, para los puntos de función.
- Salidas de usuario.
- Peticiones de usuario.
- Ficheros de lógica interna.
- Ficheros externos de interfaz.
- Puntos de función añadidos.
- Puntos de función cambiados.
- Puntos de función eliminados de la cuenta total.
- Número de líneas de código.
- Suma total de defectos encontrados.
- Suma total de esfuerzo por fases.
- Esfuerzo de aquellos proyectos que no han completado todas sus fases.
- Número de puntos de función sin ajustar.

- Fiabilidad otorgada por la organización a la cuenta de los puntos de función sin ajustar.

Sin embargo, en la base de datos original de ISBSG existen otros atributos que no han sido incluidos en este estudio, al considerar que no influyen en el valor de esfuerzo necesario para completar el proyecto, como por ejemplo el país donde se realizó el proyecto, su fecha de inicio y finalización o el ámbito de la empresa desarrolladora.

Los datos utilizados han sido normalizados según la expresión indicada en el siguiente punto, antes de ser tratados por el algoritmo de clasificación. Además, como existen atributos desconocidos en varios proyectos, se ha asignado un valor -1 para dichas situaciones, de manera que el cálculo de las distancias y de las estimaciones no tenga en cuenta aquellas características desconocidas en algún proyecto.

### 3.1.2. Normalización de los datos

Un factor fundamental que debe ser tenido en cuenta a la hora de almacenar los datos, es la necesidad de que estén expresados en el mismo intervalo, es decir, que cada característica de los proyectos esté expresada en un rango igual para todos ellos.

Por ello, a los datos se les debe aplicar primeramente una normalización según la siguiente expresión:

$$x' = \frac{x - \bar{x}}{\sigma},$$

donde $x$ es el valor que toma un parámetro en un determinado proyecto, $\bar{x}$ es la media de ese parámetro y $\sigma$ es su desviación típica.

Con esto conseguimos que las medidas de distancia empleadas sean mucho más coherentes y no se den como parecidos aquellos proyectos que en realidad no lo son.

En el siguiente ejemplo podemos apreciar la necesidad de la normalización:

"*Supongamos dos variables, una de ellas expresada con valores en torno a 1000, y la otra expresada con valores próximos al 5. Estos valores son la media de cada variable.*

*Sin aplicar normalización, un elemento de características (1012, 5) será muy parecido a otro formado por (1014, 7), sin darle importancia a la diferencia existente entre el 5 y el 7. esto hace que se le de un mayor peso o importancia a la primera característica en lugar de a la segunda.*

*Al aplicar normalización, la diferencia existente entre el valor 5 y el 7 de ambos elementos se agudiza, por lo que se aumenta la distancia existente entre ellos."*

La aplicación de esta normalización, concluye la etapa 2 de la ilustración 2, que muestra el proceso del modelo planteado.

## 3.2. ALGORITMO kNN

k-Vecinos más cercanos (del inglés *k Nearest Neighbours)* es uno de los algoritmos más utilizados para clasificar instancias. Está basado en la búsqueda de los *k* elementos más parecidos a uno dado-

En nuestra propuesta, dicho algoritmo se encargará de seleccionar los proyectos más parecidos al que se quiere estimar, para obtener a partir de ellos un valor aproximado de esfuerzo para el proyecto bajo análisis.

Esta selección de elementos parecidos implica la necesidad de utilizar una determinada medida de similitud entre instancias, que en este sistema será la medida del coseno, explicada posteriormente.

Para alcanzar nuestro objetivo de encontrar los *k* proyectos más parecidos a uno dado, el algoritmo *kNN* implementado sigue los siguientes pasos:

- Se selecciona un proyecto.
- Para cada uno de los proyectos restantes:
    - Se calcula la distancia que lo separa del proyecto elegido.
    - Esta distancia se introduce en un mapa ordenado, siendo la clave el valor del coseno.
- Se escogen los *k* elementos con mayor valor de similitud.

Al final de este algoritmo, obtendremos una matriz *n* x *k*, donde *n* es el número de proyectos evaluados, y las *k* columnas contienen los proyectos que más se le parecen.

Con esta matriz podemos comenzar a realizar las estimaciones basadas en la analogía *(fin de la etapa 3, ilustración 2)*.

### 3.2.1. MEDIDA DE SIMILITUD

Existen multitud de medidas empleadas para determinar el grado de cercanía entre elementos de un espacio vectorial, pero dada la naturaleza de los datos y su representación, se ha escogido razonadamente la medida del coseno.

Este coeficiente indica el valor del coseno formado por dos elementos del espacio vectorial como medida de similitud entre ambos. Su expresión es bien sencilla:

$$s(x,y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}},$$

donde $x_i$ e $y_i$ son el valor que adquiere la propiedad *i* en los proyectos *x* e *y*, cada uno de *n* características.

El valor que alcanza esta medida en caso de máxima similitud es de 1, no pudiendo rebasa nunca este valor. El caso contrario ocurre cuando el valor del coseno vale 0, lo que implica la máxima diferencia entre proyectos.

Para aquellas características expresadas mediante valores numéricos, la aplicación de la fórmula no requiere ningún tratamiento especial. Sin embargo, si una variable queda expresada mediante letras, tomándose cada letra como identificadora de un tipo o clase, se debe definir una "función de distancia", que determine cuando dos características son iguales. En nuestro sistema, esta función es bien sencilla:

$$d(A,B) = \begin{cases} 1 & si \quad A = B \\ 0 & si \quad A \neq B \end{cases}$$

Con el valor ofrecido por la fórmula del coseno para un determinado proyecto con respecto a los demás, podemos seleccionar los que más se le parecen.

## 3.3. ESTIMACIÓN BASADA EN LA ANALOGÍA

Para poder realizar una estimación del esfuerzo necesario para completar un proyecto software, se necesita primero conocer los esfuerzos de los *k* proyectos determinados como más cercanos por el algoritmo anterior.

La estimación por analogía en primer lugar aporta un valor estimado que es igual al esfuerzo del proyecto más cercano, en nuestro sistema a la media de, como máximo, los 3 más cercanos *(fin de la etapa 4, ilustración 2)*.

Este es un valor indicativo del esfuerzo del proyecto, que debe ser ajustado para tener en cuenta las diferencias existentes con sus vecinos más cercanos. Para realizar este ajuste, existen gran cantidad de técnicas y procedimientos, que pasan por redes neuronales, algoritmos genéticos, redes bayesianas, etc.

Nuestro alternativa implementa un ajuste sencillo y matemáticamente muy simple, pero con unos resultados muy buenos, sobretodo al ser aplicado sobre proyectos homogéneos y expresados vectorialmente. El ajuste utilizado se basa en un fenómeno estadístico y se denomina *"ajuste por regresión hacia la media"*, y fue explicado ampliamente en el anterior capítulo del presente documento.

La expresión que adopta el ajuste es la siguiente:

$$Esfuerzo = ECA + (M - ECA) \times (1 - c)\,,$$

siendo *ECA* el esfuerzo asociado a los proyectos más cercanos dentro de los análogos, *M* la media del conjunto y *c* la correlación existente entre los valores reales de esfuerzo y los predichos por la analogía.

Este sencillo ajuste no pierde vista la existencia de elementos con características extremas, y procura ofrecer valores que tiendan a mantener la media del conjunto.

En las primeras implementaciones de nuestro sistema de estimación, se tomó *M* como la media de todos los proyectos existentes, observándose la tendencia a predecir valores muy parecidos en general para todos los proyectos, y haciendo que en multitud de ocasiones el error cometido fuese muy grande. Por eso, en posteriores revisiones y mejoras se limitó el cálculo de la media a los

*k* vecinos más cercanos, con lo que se disminuyó considerablemente el error obtenido en las estimaciones.

# CAPÍTULO 4

# DISEÑO DE LAS PRUEBAS Y RESULTADOS OBTENIDOS

# CAPÍTULO 4

# DISEÑO DE LAS PRUEBAS Y RESULTADOS OBTENIDOS

Contenidos

A lo largo de este capítulo, se realizarán diversas pruebas de estimación siguiendo el modelo propuesto a lo largo de este documento.

Los resultados obtenidos en cada ejecución serán analizados y comparados para determinar cuáles son los valores que deben tomar los parámetros de los algoritmos de clasificación y de las técnicas de estimación utilizadas.

Con una base de datos tan grande como la que disponemos para la realización de este proyecto, no se pueden ir realizando pruebas con parámetros elegidos sin un análisis y planificación previa, ya que podríamos no conseguir resultados aceptables, y realizar experimentos que aporten información de escaso valor.

En este tipo de sistemas en los que se realizan predicciones acerca del valor que tomarán ciertos parámetros, es muy importante el entrenamiento al que se somete el programa, y los resultados obtenidos se ven con frecuencia afectados por los parámetros del algoritmo empleado, en nuestro caso por el número *k* de vecinos seleccionado.

De entre todas las técnicas existentes para el entrenamiento de este tipo de modelos, se ha escogido la validación *hold-out*, que se basa en la división de la base de datos en dos conjuntos disjuntos, llamados conjunto de test y de entrenamiento. El primero de ellos tiene menor tamaño, y contiene los elementos que serán evaluados, utilizando para realizar dicha evaluación los contenidos en el segundo grupo.

Esta técnica presenta como principal ventaja la eficiencia, ya que otras técnicas, como la validación cruzada (*cross validation*), requiere que sean evaluados todos los ejemplares de la base de datos, lo que acarrea un gran esfuerzo computacional, sobre todo en situaciones en las que existe una cantidad considerable de ejemplares a examinar. Además, en *hold-out* el conjunto de test se crea aleatoriamente, realizándose una buena evaluación del modelo si se ejecuta varias veces.

Los tamaños de los conjuntos de test y de entrenamiento también adquieren gran relevancia, por lo que deben ser combinados de diversas formas para obtener multitud de resultados, y poder escoger así la configuración que alcanza menores cotas de error en las estimaciones.

## 4.1. PARÁMETROS DE CONFIGURACIÓN DE LAS PRUEBAS

El conjunto de todos los datos de proyectos software será dividido en dos conjuntos disjuntos de diferente tamaño. Uno de ellos será el conjunto de test, formado por proyectos cuyo valor de esfuerzo será estimado. Para poder realizar esta estimación, será necesaria una "base de información de proyectos" con los que trabajar para seleccionar los vecinos más cercanos y hacer el ajuste por analogía. Este grupo de proyectos utilizados para tal efecto es el denominado conjunto de entrenamiento.

Se deben por tanto diseñar pruebas con distintos tamaños del conjunto de test y de entrenamiento, siendo preferible un menor tamaño del de test, para poder tener así muchos proyectos para entrenar y realizar las estimaciones.

Otro parámetro del sistema que se debe tener en cuenta es el número de vecinos más cercanos que se seleccionarán para cada proyecto del conjunto de test y poder así realizar las estimaciones. Este parámetro tomará obviamente valores menores que el tamaño del conjunto de entrenamiento, ya que no se pueden seleccionar más vecinos de los que hay en dicho conjunto.

Se observará en los experimentos que para valores muy altos del número de vecinos seleccionados se introduce mucho ruido en las estimaciones. Además, dada la naturaleza del ajuste hecho a la estimación, que tiene en cuenta la media de esfuerzo de los $k$ vecinos, este ruido en teoría se debe ver incrementado.

Por el contrario, valores muy pequeños del número de vecinos pueden hacer que el valor estimado del esfuerzo sea prácticamente igual que el de sus análogos, perdiendo de vista las diferencias existentes entre ellos.

Un tercer parámetro que será combinado con los anteriores es el número de proyectos más parecidos de entre los *k* seleccionados por *kNN*, con los que se calculará una primera estimación sin ajustar, a la que se le aplicará posteriormente el ajuste por regresión hacia la media.

En la tabla 16 se pueden ver reflejados los valores que tomarán estos tres parámetros de configuración de las pruebas, que al combinarlos entre sí se forman 36 experimentos en total.

## 4.2. DESARROLLO DE LAS PRUEBAS

Una vez establecidos los parámetros de configuración de las pruebas, comenzará el desarrollo de las mismas. En primer lugar, se dividirá el conjunto de todos los proyectos en dos grupos: el de test y el de entrenamiento. Para ello se seleccionarán aleatoriamente *n* proyectos, siendo *n* el tamaño del conjunto de test. El conjunto de entrenamiento será creado incluyendo en él todos aquellos proyectos que no formen parte del grupo de test.

| TAMAÑO DEL CONJUNTO DE TEST | NÚMERO DE VECINOS SELECCIONADOS | NÚMERO DE MÁS CERCANOS DE ENTRE LOS K SELECCIONADOS | NOMBRE DEL EXPERIMENTO |
|---|---|---|---|
| 10 % | k=5 | 1 | Prueba 1 |
| | | 2 | Prueba 2 |
| | | 3 | Prueba 3 |
| | k=10 | 1 | Prueba 4 |
| | | 2 | Prueba 5 |
| | | 3 | Prueba 6 |
| | k=20 | 1 | Prueba 7 |
| | | 2 | Prueba 8 |
| | | 3 | Prueba 9 |
| | k=50 | 1 | Prueba 10 |
| | | 2 | Prueba 11 |
| | | 3 | Prueba 12 |
| 20 % | k=5 | 1 | Prueba 13 |
| | | 2 | Prueba 14 |
| | | 3 | Prueba 15 |
| | k=10 | 1 | Prueba 16 |
| | | 2 | Prueba 17 |

| | | | |
|---|---|---|---|
| | | 3 | Prueba 18 |
| | k=20 | 1 | Prueba 19 |
| | | 2 | Prueba 20 |
| | | 3 | Prueba 21 |
| | k=50 | 1 | Prueba 22 |
| | | 2 | Prueba 23 |
| | | 3 | Prueba 24 |
| 30 % | k=5 | 1 | Prueba 25 |
| | | 2 | Prueba 26 |
| | | 3 | Prueba 27 |
| | k=10 | 1 | Prueba 28 |
| | | 2 | Prueba 29 |
| | | 3 | Prueba 30 |
| | k=20 | 1 | Prueba 31 |
| | | 2 | Prueba 32 |
| | | 3 | Prueba 33 |
| | k=50 | 1 | Prueba 34 |
| | | 2 | Prueba 35 |
| | | 3 | Prueba 36 |

**Tabla 16: Pruebas diseñadas**

Con esto realizado, se comenzará a seleccionar para cada elemento del conjunto de test, los *k* elementos más parecidos a él provenientes del conjunto de entrenamiento.

Una vez determinados estos proyectos, se realizará una estimación de su esfuerzo mediante analogía, realizando posteriormente el ajuste por regresión hacia la media. Cuando se tengan hechas las estimaciones de cada proyecto del conjunto de test, se procederá al cálculo del error cometido, a través de la siguiente expresión:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|E_R - E_E\right|}{E_R},$$

donde $E_R$ es el esfuerzo real de proyecto, $E_E$ el esfuerzo estimado y *n* el número de proyectos del conjunto de test.

Este MMRE (Mean Magnitude of Relative Error), es el error cometido por el experimento, que nos servirá para determinar al final cuál es la mejor configuración de los mismos. Es una medida muy común para mostrar la eficiencia de las estimaciones en Ingeniería Software.

Como la creación de los conjuntos de test y entrenamiento es aleatoria, cada experimento será realizado 20 veces, obteniéndose al finalizar todas las ejecuciones la media de los MMRE obtenidos en cada iteración.

Otra medida que será tenida en cuenta a la hora de realizar las ejecuciones, es el número de ocasiones en los que el ajuste por regresión hacia la media realizado a cada estimación ha conseguido un valor de esfuerzo más cercano al real que el aportado inicialmente teniendo en cuenta sólo los elementos más parecidos.

## 4.3. RESULTADOS OBTENIDOS

Antes de comenzar a comentar los resultados obtenidos en las ejecuciones, es conveniente mostrar los porcentajes de tamaño de los conjuntos de test y de entrenamiento considerados para realizar las pruebas, así como el número de proyectos incluidos en cada caso.

En la tabla 17 se puede observar la distribución de estos tamaños, según los porcentajes asignados a cada uno de los conjuntos.

Los conjuntos de test y de entrenamiento creados para cada prueba no poseen ningún elemento en común, y engloban la totalidad de los proyectos de la base de datos, en total 2027.

| PORCENTAJE CONJUNTO TEST / TRAINING | NÚMERO DE ELEMENTOS CONJUNTO TEST | NÚMERO DE ELEMENTOS CONJUNTO TRAINING |
|---|---|---|
| 10 % - 90 % | 203 | 1824 |
| 20 % - 80 % | 406 | 1621 |
| 30 % - 70 % | 608 | 1419 |

**Tabla 17: Tamaños de los conjuntos de Test y Entrenamiento**

En la tabla 18 pueden observarse los resultados de las pruebas, con los MMRE obtenidos y los porcentajes de mejoras en las estimaciones gracias al ajuste por regresión.

| PRUEBA | MMRE | ESTIMACIONES HECHAS | MEJORAS POR REGRESIÓN | % MEJORAS |
|---|---|---|---|---|
| 1 | 1.4240353 | 4060 | 2684 | 66.1084% |

| | | | | |
|---|---|---|---|---|
| 2 | 0.9248598 | 4060 | 2441 | 60.1232% |
| 3 | 0.88683033 | 4060 | 2281 | 56.1823% |
| 4 | 1.2652854 | 4060 | 2836 | 69.8522% |
| 5 | 0.9283263 | 4060 | 2477 | 61.0099% |
| 6 | 1.1535473 | 4060 | 2352 | 57.9310% |
| 7 | 1.0953348 | 4060 | 2719 | 66.9704% |
| 8 | 1.1732101 | 4060 | 2523 | 62.1429% |
| 9 | 0.84652203 | 4060 | 2329 | 57.3645% |
| 10 | 0.9617287 | 4060 | 2565 | 63.1773% |
| 11 | 1.0202768 | 4060 | 2387 | 58.7931% |
| 12 | 1.0323083 | 4060 | 2300 | 56.6502% |
| 13 | 1.07788 | 8120 | 5549 | 68.3374% |
| 14 | 1.024119 | 8120 | 4891 | 60.2340% |
| 15 | 0.950191 | 8120 | 4535 | 55.8498% |
| 16 | 1.2807004 | 8120 | 5533 | 68.1404% |
| 17 | 1.0464857 | 8120 | 4905 | 60.4064% |
| 18 | 0.93076813 | 8120 | 4755 | 58.5591% |
| 19 | 0.9554926 | 8120 | 5375 | 66.1946% |
| 20 | 0.958801 | 8120 | 4844 | 59.6552% |
| 21 | 1.0361177 | 8120 | 4699 | 57.8695% |
| 22 | 1.2096382 | 8120 | 5224 | 64.3350% |
| 23 | 1.0122854 | 8120 | 4759 | 58.6084% |
| 24 | 1.0188601 | 8120 | 4475 | 55.1108% |
| 25 | 1.2203498 | 12160 | 8257 | 67.9030% |
| 26 | 0.9988824 | 12160 | 7360 | 60.5263% |
| 27 | 0.95995396 | 12160 | 6786 | 55.8059% |
| 28 | 1.1595623 | 12160 | 8353 | 68.6924% |
| 29 | 1.034171 | 12160 | 7596 | 62.4671% |
| 30 | 1.0339931 | 12160 | 7085 | 58.2648% |
| 31 | 1.1660955 | 12160 | 8113 | 66.7188% |
| 32 | 0.93949234 | 12160 | 7278 | 59.8520% |
| 33 | 0.89622515 | 12160 | 7021 | 57.7385% |
| 34 | 1.1606127 | 12160 | 7949 | 65.3701% |
| 35 | 1.0452673 | 12160 | 7209 | 59.2845% |
| 36 | 0.95805824 | 12160 | 6908 | 56.8092% |

**Tabla 18: Resultados de las pruebas**

En la tabla 19 se observan los valores medios obtenidos de MMRE y de mejoras por regresión hacia la media conseguidos con todas las ejecuciones

| | |
|---|---|
| MEDIA TOTAL DE MMRE | 1.049618561 |
| MEDIA TOTAL DE MEJORAS | 61.3622% |

**Tabla 19: Media de los resultados obtenidos**

Con estos resultados a la vista, podemos obtener un gran número de conclusiones. Primeramente podemos hacer referencia a los porcentajes de mejoras alcanzados por el ajuste por regresión hacia la media. Una primera estimación nos ofrece el valor de esfuerzo de los proyectos más parecidos dentro del conjunto de los *k* vecinos encontrados. Recordemos que según las pruebas diseñadas, se pueden escoger 1, 2 ó 3 proyectos para realizar esta primera estimación, basada tan sólo en obtener la media de esfuerzo de estos elementos.

A este dato aportado se le aplica el ajuste por regresión hacia la media, y se comparan las diferencias existentes entre los dos valores estimados (uno sin ajuste y otro ajustado) con el valor real de esfuerzo del proyecto en cuestión.

Pues bien, de este conteo de mejoras se deduce fácilmente la conveniencia de usar este tipo de ajuste, no sólo por su buen porcentaje de resultados, sino por su eficiencia y sencillez, al estar basado en una simple operación matemática y un cálculo del coeficiente de regresión lineal entre dos variables.

En esta expresión del ajuste por regresión hacia la media, interviene la media de esfuerzo del conjunto. Este valor puede ser interpretado como la media de todos los proyectos contenidos en la base de datos, o como la media de los *k* proyectos seleccionados por el algoritmo *kNN*.

En las primeras ejecuciones de las pruebas se tomaba como valor de media para el ajuste la media de toda la base de datos, observándose unos valores muy altos del MMRE, y extremadamente bajos del porcentaje de mejora en la estimación. Esto provocó la sustitución de este valor por el de la media del conjunto devuelto por *kNN*, apreciándose una notable mejora en todos los sentidos. Los malos resultados obtenidos por las estimaciones considerando la media de toda la base de datos no merecen ser reflejados en este documento, por la escasa información que aportan.

Con respecto a los errores cometidos, medidos según MMRE, podemos concluir la obtención de unos resultados aceptables y comparables con los alcanzados por la literatura en este ámbito [CHI06] [JOR03].

Existen muchas diferencias en las cotas de error alcanzadas en las estimaciones en función de los parámetros de la prueba, por lo que a continuación se estudiará la influencia de cada uno de estos factores en los resultados de las mismas.

El primer parámetro de configuración que salta a la vista es el tamaño del conjunto de test y del de entrenamiento. Se han considerado valores

pequeños para el conjunto de test, aportando cada uno de ellos unos resultados parecidos, como podemos apreciar en la tabla 20.

| PORCENTAJE CONJUNTO TEST / TRAINING | MMRE | MEJORAS POR REGRESION |
|---|---|---|
| 10 % - 90 % | 1.05935543 | 61.3588% |
| 20 % - 80 % | 1.041778269 | 61.5445% |
| 30 % - 70 % | 1.047721983 | 61.5538% |

**Tabla 20: Resultados obtenidos en función del tamaño del conjunto de Test**

Como norma general, las variaciones del MMRE y de los porcentajes de mejora no varían mucho en función del tamaño del conjunto de test. Las medias se mantienen más o menos parecidas, aunque si tuviéramos que decantarnos por una configuración quizá fuera conveniente hacerlo por la segunda, que ofrece un tamaño del 20 % para el grupo de test, y que ofrece los valores más bajos de MMRE, siendo las mejoras por regresión cercanas a las mejores, alcanzadas con un 10%.

Una vez determinado que el tamaño óptimo para el conjunto de test ronda el 20 %, debemos analizar el resto de parámetros de configuración de las pruebas. Para ello la tabla 20 no nos sirve de mucho, por lo que para conseguir este objetivo puede interesarnos más analizar los datos agrupados por dichos factores:

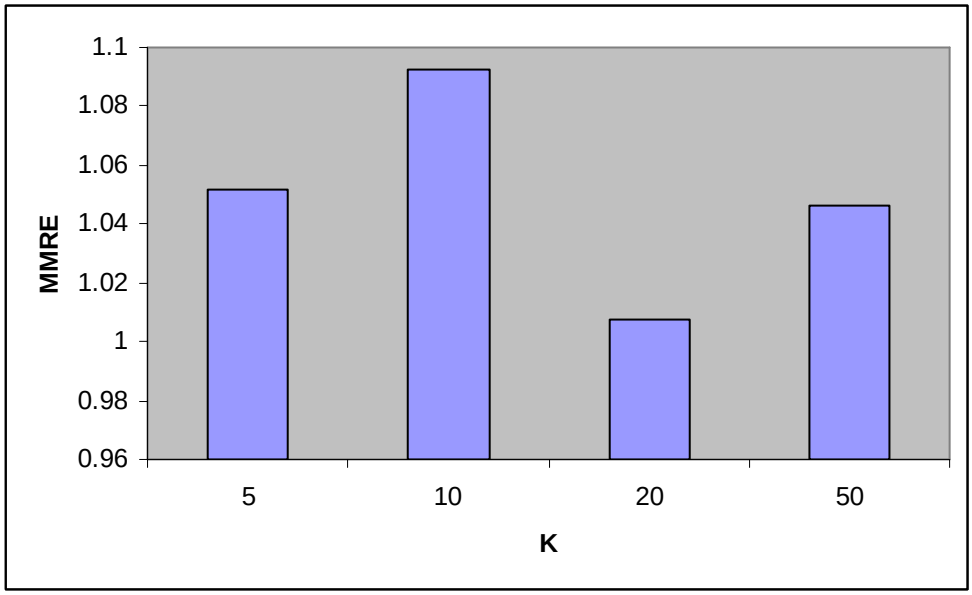


**Ilustración 2: MMRE en función de k**

Parece claro que *k* no debe tomar valores extremos, ni muy pequeños ni muy grandes. Esto se explica por dos motivos. En primer lugar, valores muy pequeños hacen que el ajuste por regresión hacia la media funcione peor, ya que toma como media la de un conjunto reducido. Con valores muy grandes, se introduce demasiado ruido en la estimación, y asigna a proyectos que presenten unas características extremas valores de esfuerzo demasiado cercanos a la media.

Por estos motivos, y tal y como se aprecia en la ilustración 2 el valor idóneo de *k* para el algoritmo *kNN* es 20.

Esta decisión queda apoya por la ilustración 3, en la que se observa un elevado nivel de mejoras en la regresión con un valor de *k=20*.

Veamos ahora lo que ocurre con el número de proyectos más cercanos a considerar dentro de los *k* seleccionados. Para ello podemos servirnos de la ilustración 4, que, muestra una clara predilección por valores altos, como el 3. La respuesta a esta predilección radica en que apenas existen proyectos en la base de datos con características extremas. Esto hace que la estimación inicial dada por los más parecidos sea bastante buena de por sí, viéndose aún mejorada por el ajuste por regresión.

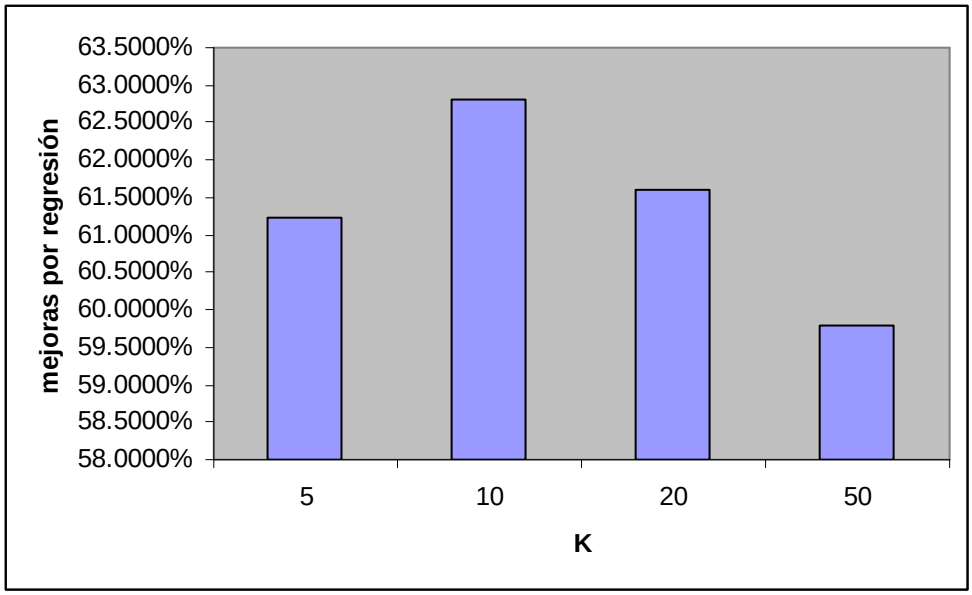


**Ilustración 3: Mejoras por regresión según k**

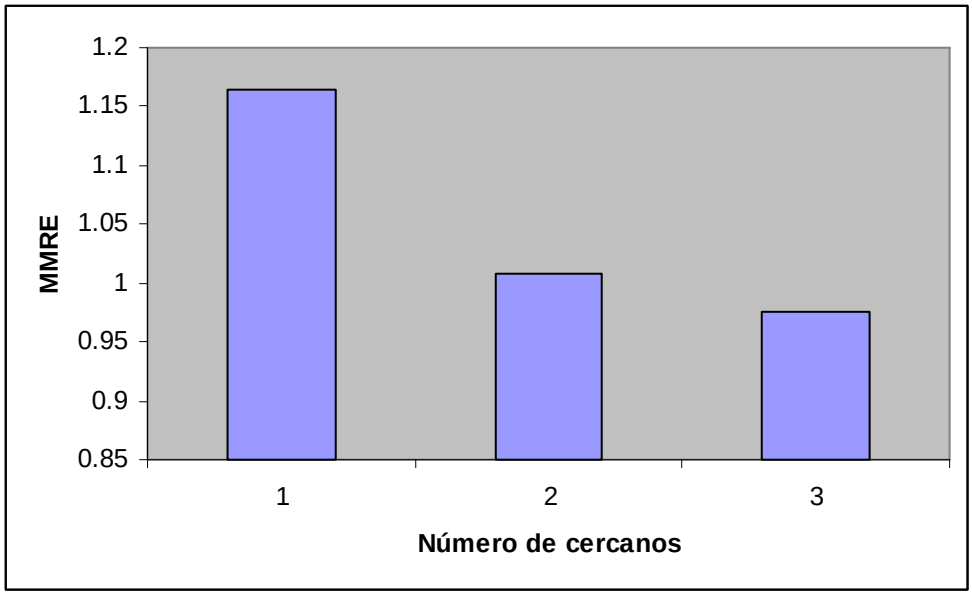


**Ilustración 4: MMRE según el número de proyectos cercanos**

Todas estas conclusiones nos han ido encaminando a la obtención de la configuración más apropiada de la alternativa para la estimación de esfuerzo de proyectos software planteada en este trabajo.

Esta configuración se corresponde con un tamaño del conjunto de test del 10 % del total de la base de proyectos, un número *k* para el algoritmo *kNN*

de 20, y la consideración de 3 proyectos como los más parecidos para realizar la estimación inicial previa al ajuste por regresión.

Efectivamente, la prueba configurada con estas características es la que consigue el menor valor de MMRE de todas. Esta es la prueba número 9.

# CONCLUSIONES

La realización de un trabajo de la envergadura de un proyecto fin de carrera aporta a cualquier estudiante un gran número de conclusiones sobre la labor realizada. La exposición de estas ideas es una tarea difícil y compleja, ya que involucra concentrar en unas líneas todo lo aprendido a lo largo de meses de dedicación.

No obstante, el recordar y comentar las conclusiones obtenidas constituye en sí una aportación más a la formación del autor, que cerrará así con brillantez la etapa de desarrollo del proyecto.

La dificultad encontrada por cualquier estudiante para extraer las conclusiones finales, se ve acentuada aún más si cabe si éste pertenece a una carrera técnica con gran implicación e importancia en la sociedad actual, o si ha desarrollado un proyecto acerca de un tema de gran trascendencia o futuro.

En mi caso, al pertenecer a una titulación con grandes salidas laborales y de enorme implicación en la sociedad, me veo obligado a realizar un profundo análisis que debe comprender desde el momento en que opté por cursar esta titulación, hasta la elección de este proyecto fin de carrera. Completado este análisis, aportaré ya una serie de conclusiones más técnicas y derivadas directamente de los resultados obtenidos de las ejecuciones del modelo planteado en el proyecto.

Cierto es que opté por estudiar Ingeniería Informática por lo llamativas y variadas que eran sus salidas laborales, aunque también sentía una especial curiosidad e intriga por la Programación. Sin embargo desconocía todo el proceso que rodea a la codificación de un producto software, pensando solamente en su creación mediante un determinado lenguaje de programación.

Cierto también es que me agradó conocer y practicar diversas técnicas de programación y construcción de programas, pero seguía ignorando la labor realizada fuera de los lenguajes.

Llegado el momento de aprender ciertas nociones básicas de Ingeniería Software, me percaté de la enorme importancia que ésta adquiere en la creación de programas y productos informáticos, por lo que no tuve ninguna duda a la hora de escoger una temática para mi proyecto fin de carrera.

Recuerdo ahora con mucha ilusión el comienzo de este trabajo, sentando sus bases y delimitando sus objetivos; y es ahora cuando de verdad comprendo el papel tan importante que juega la Ingeniería Software en la creación de productos totalmente funcionales y de calidad.

Sin perder de vista el mundo laboral, cada vez más cercano para cualquier estudiante que se encuentre elaborando su proyecto fin de carrera, las empresas desarrolladoras de Software convierten en totalmente imprescindible la aplicación de métodos y técnicas de Ingeniería Software, ya que su esencia y supervivencia dependen de la entrega de programas fiables y correctamente desarrollados.

Estas empresas pueden dedicar absolutamente todo su esfuerzo en desarrollar excelentes programas, pero surge la necesidad de medir y controlar este esfuerzo, para no acabar en situación de pérdida de dinero, tiempo, empleados, etc.

Ante esta situación se debe actuar, y la mejor manera pasa por realizar predicciones sobre diversas características del proyecto software previas al desarrollo del mismo. Una de las predicciones más importantes que se hace es la del esfuerzo, para evitar desembocar en situaciones similares a las comentadas anteriormente.

Razonadamente pienso que no conozco todos los secretos de la Ingeniería Software, pero al menos me he dado cuenta de que uno de sus momentos más delicados y dificultosos sucede a la hora de realizar las estimaciones del esfuerzo necesario para completar un determinado proyecto.

Esta decisión conlleva un gran riesgo a cualquier organización que deba tomarla, por lo que debían existir técnicas que ayudaran a realizarla y que

indicaran de alguna manera la probabilidad de errar al dar como predicción un cierto valor.

Efectivamente son muchas las técnicas existentes para la estimación. Creo que algunas son demasiado antiguas, y necesitan una adaptación a los modelos actuales de desarrollo software. Otras por el contrario son muy novedosas, y responden con precisión a las dudas planteadas por los estimadores. En general, de entre la multitud de formas de realizar predicciones, se debe elegir la que mejor se adapte a la información disponible y a las características del proyecto.

No debe surgir el temor de no encontrar un modelo de estimación que se ajuste a lo necesitado. Al revés, debe contemplarse siempre la posibilidad de combinar técnicas y crear nuevos modelos y alternativas, como ha pretendido hacer quien escribe estas líneas. Puede escogerse un modelo, y utilizar sus salidas como entrada para otro; o puede aplicarse un sistema que tome las ecuaciones y operaciones de otro; o se puede, de una manera más atrevida, emplear algoritmos que en principio están destinados a otros propósitos para mejorar modelos ya establecidos.

Esta última alternativa ha sido la escogida en este proyecto, donde a un método robusto y fiable como la estimación por analogía, se le ha aplicado un algoritmo de clasificación de instancias, cerrando este proceso de predicción un ajuste también conocido y utilizado, aunque de naturaleza estadística y matemática.

Si ya de por sí tenía una visión de la estimación como un proceso importante y de enorme impacto en cualquier empresa de desarrollo, ahora más aún lo comprendo, al haber trabajado y comprobado desde una posición privilegiada las dificultades que esta tarea conlleva.

De todas formas, no debemos pensar que la estimación sea un proceso excesivamente complicado. Efectivamente es muy delicado por las repercusiones que puede tener en la actividad normal de una empresa, pero como proceso algorítmico e informático puede llegar a ser bastante sencillo. De

hecho, la alternativa que propongo no utiliza complicados ajustes ni operaciones; tan sólo se basa en la aplicación de un mecanismo de clasificación de elementos, y en el ajuste de la estimación inicial mediante una muy sencilla operación matemática.

Sobre estos algoritmos empleados he obtenido también varias conclusiones. La primera es que creo que, al igual que ocurre con los modelos de estimación, existen multitud de métodos de clasificación de instancias. Debemos escoger siempre el que más se ajuste a la estructura de la información que poseamos, y el que veamos que puede alcanzar mejores resultados en función de los ámbitos en que suele ser ejecutado. Poniendo como ejemplo la alternativa a la estimación que planteo en este proyecto, conocemos de antemano que vamos a realizar las predicciones de esfuerzo basándonos en las analogías existentes entre proyectos. Si queremos aplicar un algoritmo de clasificación, de poco nos va a servir aquel que prediga la clase a la que pertenece un elemento. Nos va a venir mejor un método que asigne a un proyecto determinado un conjunto de ejemplares parecidos, además ordenados por su valor de distancia.

Estos motivos expuestos son los que hicieron que se escogiera como algoritmo de clasificación *kNN*, ya que se amolda perfectamente a nuestros requerimientos, y sabemos además que ofrece unos buenos resultados, muy coherentes con las medidas de similitud que utiliza. La elección de esta medida de similitud no presentó muchos problemas, ya que desde el momento que supe que los datos de los proyectos estaban expresados en forma vectorial, pensé en la medida del coseno como algo natural y propio, de fácil implementación, eficaz y robusta.

Me resulta muy llamativa la sencillez con la que *kNN* crea los conjuntos de proyectos análogos a uno dado. Además, el poder disponer de potentes estructuras de datos como las ofrecidas por el lenguaje de programación utilizado lo convierte en un proceso más transparente si cabe, y más fácil de entender por cualquier persona. Sólo basta con proporcionarle las medidas de

distancia del elemento que queremos clasificar con el resto, y el algoritmo ya se encarga de crear los conjuntos de parecidos.

Ya con estos conjuntos creados, el modelo planteado realiza las estimaciones y las ajusta por el mecanismo de regresión hacia la media. A la vista de los resultados obtenidos, podemos concluir que la regresión hacia la media es un ajuste sencillo que en la mayoría de las ocasiones consigue mejorar la estimación inicial aportada por la analogía. Existen una gran cantidad de ajustes posibles, pero considero que la elección hecha en nuestro modelo ha sido acertada, al ser palpable la sencillez con la que pueden ser alcanzados unos buenos resultados, y al ser muy adecuado para la información con la que hemos trabajado. No obstante, pueden aplicar se en trabajos futuros otro tipo de ajuste a las estimaciones, y poder así comparar qué mecanismo resulta mejor.

A lo largo de las ejecuciones, salta a la vista una cierta aleatoriedad a la hora de realizar las estimaciones, ya que las predicciones iniciales se realizan en función de los proyectos más parecidos, pudiendo ser estos 1, 2 ó 3. Las pruebas han demostrado que la probabilidad de error disminuye cuanto mayor es el número de vecinos con los que se calcula una estimación inicial. Además, los valores de error obtenidos varían en función del conjunto de test con el que se trabaje, debido a esta aleatoriedad con la que se eligen los miembros de los conjuntos de test y de entrenamiento.

Por estos motivos expuestos anteriormente se propone para investigaciones futuras la implementación de modelos que tengan en cuenta más proyectos análogos para realizar las estimaciones iniciales. En nuestro modelo, se han tomado como máximo 3, al ser éste el valor más frecuentado en la literatura.

Puede resultar también interesante aplicar diferentes pesos a los atributos de los proyectos, en función de su importancia o influencia en el valor final de esfuerzo; o incluso realizar una selección de características, en lugar de hacer los cálculos teniéndolas todas en cuenta. Todas estas posibles ampliaciones, me hacen darme cuenta de que éste ha sido un proyecto innovador por su temática, y que acepta multitud de modificaciones futuras, lo

cual me alegra, ya que eso significa que este trabajo ha sido una aportación no sólo a mi formación académica, sino también a la disciplina de la estimación en Ingeniería Software.

# REFERENCIAS Y BIBLIOGRAFÍA

# REFERENCIAS

[AGA01] Agarwal, R. y Kumar, M. “Estimating software projects”, Software engineering Notes, 26(4), 60-67, 2001.

[ANG00] Angelis L, y Stamelo, I. “A simulation tool for efficient analogy based cost estimation”, Empirical Software engineering 5, 35-68, 2000.

[BAR06] Barranco, M. J., y Martínez, L. "Un Sistema de Recomendación Basado en Conocimiento con Información Lingüística Multigranular" SIGEF XIII, 30th November- 2nd December, 2006, Hammamet, Túnez, pp. 645-664.

[CHA96] Chatzoglou, P. D. y Macaulay, L. A. “A review of existing models for project planning and estimation and the need for a new approach”, International Journal of Project Management, 14(3), 173-183, 1996

[CHI06] Chiu, N. H. y Huang, S. J., “The adjusted analogy-based software effort estimation based on similarity distances”, The Journal of Systems and Software, 80(2007), 628-640, 2006.

[HEE92] Heemstra, F. J. “Software cost estimation”, Information and Software engineering, 34(10), 627-639, 1992.

[HUA06] Huang, S. J., y Chiu, N. H., in press. “Optimization of analogy weights by genetic algorithm for software effort estimation”, Information and Software Technology, 2006

[IEEE93] IEEE Standard Glossary of Software Engineering Terminology, 1993.

[JEF00] Jeffery, R. y Ruhe, M. “A comparative study of two software development cost modeling techniques using multi-organizational and

company-specific data", Information and software Technology, 42, 1099-1016, 2000.

[JOR03] Jørgensen, M., y Indahl, U. "Software effort estimation by analogy and regresión toward the mean", The Journal of Systems and Software 68 (2003), 253-262, 2003.

[JOR04] Jørgensen, M. y Sjøberg, D. I. K. "The impact of customer expectation on software development effort estimates", Internacional Journal of Project Management, 22, 317-325, 2004.

[LOR94] Lorenz, M. y Kidd, J. Object-oriented Software Metrics, Prentice-hall, 1994.

[MUK92] Mukhopadhyay, T. y Vicinanza, S. "Estimating the feasibility of a case-based reasoning model for software effort estimation", MIS Quarterly 16(2), 155-171, 1992.

[PRE98] Pressman, R. "Ingeniería del Software. Un enfoque práctico", McGraw Hill, 1998.

[PRE06] Pressman, R. "Ingeniería del Software. Un enfoque práctico", McGraw Hill, 2006.

[SHE97] Shepperd, M. y Schofield, C. "Estimating software project effort using analogies", IEEE Transactions on Software engineering 23 (12), 736-743, 1997.

# BIBLIOGRAFÍA

Gramajo, E. y García-Martínez, R. “Combinación de alternativas para la estimación de proyectos software”, Centro de Actualización Permanente en Ingeniería de Software.

Varas, M. “Una experiencia con la Estimación del Tamaño del Software”, disponible en Internet (http://www.inf.udec.cl/revista/edicion1/mvaras.htm).

“Métricas, estimación y planificación en Proyectos de Software”, disponible en Internet (www.willydev.net).

“Estimación del software”, Colección de Apuntes Asignatura “Planificación de Sistemas informáticos”, Universidad de Jaén.

Repositorio de datos del *International Software Benchmarking Standards Group*, disponible en Internet (www.isbsg.org).