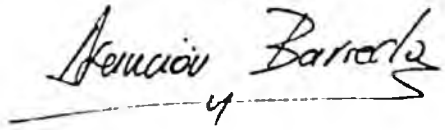
DIRECTOR

Dr. D. Máximo Llaguno Ellacuria

"Modelo de Estimación del Esfuerzo
Basado en la Evolución de la Métrica
del Software".

Asunción Barredo Fuentes



FACULTAD DE INFORMATICA

UNIVERSIDAD DE DEUSTO

JUNIO 1990

INDICE

RESUMEN

El aspecto más importante y a la vez más complejo de un proyecto de software, es probablemente la estimación del esfuerzo requerido para desarrollar el proyecto. Uno de los parámetros más importantes en gran parte de los modelos de estimación del esfuerzo, es el tamaño del proyecto. En este trabajo, presentamos una estimación del tamaño basada en la estimación inicial de métricas de estructuras de datos. La aproximación que presentamos se deriva de la premisa de que los programas se pueden construir estructuradamente, de manera que la mayor parte de las estructuras de datos se desarrollen durante la etapa de diseño del programa. Una vez encontrada la aproximación al tamaño, pasamos a estudiar la evolución de las métricas del software durante el proceso de desarrollo. Análizamos esta evolución para llegar a un conocimiento más claro y conciso del proceso de desarrollo. Hemos identificado y observado varios patrones de evolución de la métrica; y hemos comprobado, que estos patrones dependen en gran manera, de la estrategia de desarrollo empleada en el proceso de construcción de los programas. En base al estudio del comportamiento de la evolución la métrica, hemos realizado un modelo usando funciones simples del tiempo de desarrollo, que predice el esfuerzo. Con los experimentos realizados, hemos conseguido demostrar la capacidad de realización del modelo, descubriendo que la estimación del esfuerzo generada es mejor que las estimaciones subjetivas realizadas por los participantes en los experimentos.

ABSTRACT

Estimating the amount of effort requiered for Software development is probably the most important and difficult aspect of any software project. Most current effort estimation models include project size as one of the most important parameters. In this thesis an early size estimation approach based on the early estimation of data-structure metrics is presented. This approach is derived from the assumption that programs can be constructed in a structured way so that most data structures can be developed during program design. After finding their size estimation approach, we began to study the evolution of software metrics during program development. We studied metric evolution to develop a clearer understanding about the development process. Several distinctive patterns of metric evolution were observed and identified in our research. These patterns of metric evolution were found to hinge significantly upon the development strategy employed in the construction process. An effort prediction model using simple functions of development time were found based on the metric evolution behaviour. The feasibility of this model was demonstrated by our program construction experiment in which the effort estimated generated using this model were significantly better than the subjective estimates made by the experimental subjects.

RELACION DE FIGURAS

RELACION DE TABLAS

CAPITULO 1

INTRODUCCION

En la vida cotidiana, la complejidad que conlleva la realización de una tarea puede diferir en mucho de la complejidad que llevaría el proceso desarrollado para ejecutar esa misma tarea de forma informatizada. Debido a ello, hoy en día es necesario realizar un control exhaustivo sobre los proyectos de desarrollo de software debido a su complejidad. En muchos casos, la dirección de un proyecto de software no difiere de la dirección de proyectos comparables para cualquier otra tecnología [THAY84]. Todos ellos requieren técnicas clásicas de organización: capacidad de planificar, organizar, coordinar, dirigir y controlar. La diferencia entre los proyectos de software y otros tipos de proyectos radica fundamentalmente en la entrega del proyecto ya realizado, sujeto a modificaciones muy habituales en el desarrollo de software, y raras en el resto de tecnologías.

El desarrollo de proyectos de software tiene muchos aspectos únicos que lo caracterizan, vamos a ver algunos de ellos:

- Las decisiones técnicas y estructurales englobadas en el software son casi invisibles hasta su conclusión.

- El desarrollo del software requiere de una complejidad impredecible inicialmente hasta por el profesional encargado del mismo.

- Es especialmente dificultosa la consecución de una normalización de productos de software debido a la heterogeneidad, tanto de máquinas como de software base, existente en el mercado actual.

- En pocas ocasiones, se da por finalizado un proyecto de software, habitualmente experimentará cambios.

- Lo más frecuente en un usario es requerir la adaptación de un producto de software a su necesidad individual.

Estos y otros aspectos del software justifican muchos de los problemas técnicos y de dirección, planteados en nuestros días [THAY84]. Como consecuencia de todo lo aquí indicado, en muchos proyectos de software llevados a cabo actualmente se originan los siguientes problemas típicos [DEMA81]:

- se rebasan los costes,
- se retrasa la planificación realizada,
- la fiabilidad de los procesos es mala,
- los procesos fallan,
- o se produce el fracaso total.

Cualquiera de los problemas típicos mencionados será en general consecuencia de un mal diseño inicial del proyecto.

En una investigación sobre los aspectos más importantes de la dirección de proyectos de software, se identificó la planificación del proyecto (entendiendo por planificación del mismo, el proceso de desarrollo, cálculo de costos y análisis de la implantación), como uno de los problemas más críticos que tienen que resolver los directores de software [THAY81].

Una buena planificación de proyectos, se basa en estimaciones razonablemente adecuadas sobre la cantidad de esfuerzo requerida para completar el proyecto con resultado final satisfactorio. A menos que un proyecto tenga sus puntos clave claramente definidos y unas estimaciones realistas del tiempo y de los costos requeridos para llevarlo a cabo, no existe un camino fácil para que un director de proyectos dictamine si los mismos están o no bajo control. En consecuencia, estimar la cantidad de trabajo requerida para el desarrollo del software, motivo principal de esta tesis, es uno de los aspectos más importantes y aún así más difíciles de cualquier proyecto de software [DEMA82]. Debido a la escalada en el costo de software, a las crecientes dificultades en el manejo y control del desarrollo, y a la aceleración en la demanda del uso de ordenadores, se ha creado una necesidad acuciante de estimadores cualificados del esfuerzo. Por lo tanto, la complejidad del

desarrollo de un proyecto de software debe evalu... mediante métricas objetivas, fiables, válidas y convenientes, y no por la intuición.

1.1 OBJETIVOS

El objetivo primordial de esta tesis es encontrar un modelo de predicción del esfuerzo basado en la definición del comportamiento de la evolución de la métrica. Para lograrlo nos hemos marcado un camino con dos puntos de atención importantes.

El primer paso es la aproximación a la estimación del tamaño, a la que denominamos estimación indirecta. En primer lugar, supondremos que se puede aproximar el tamaño del programa por medio de una función de variables cuantificables, contenidas en dicho programa. En segundo lugar, asumiremos que se puede desarrollar un programa de tal manera, que se puedan medir estas variables, en una primera fase. Así pues, se puede obtener de modo indirecto, la estimación inicial del tamaño, mediante la medición y combinación de estas variables en una fase inicial.

El segundo paso es el comportamiento de la evolución de la métrica del software durante el proceso de desarrollo del programa. Consideramos que una métrica del software es una medida que nos permite cuantificar una cierta propiedad del programa. Las propiedades frecuentemente caracterizadas por la métrica del software incluyen el tamaño, las estructuras de control y las

estructuras de datos. La "evolución de la métrica del software" se ocupa de cómo el valor de la métrica varía con el tiempo de desarrollo desde cero hasta su valor final.

Una de las razones por las cuales es difícil de controlar o predecir el proceso de desarrollo del software es que este proceso nunca ha sido comprendido plenamente. Tradicionalmente, se le ha considerado como un arte y no como una disciplina científica. Es necesario explorar la mayoría de los aspectos del proceso de desarrollo más ampliamente, antes de poder obtener un entendimiento completo del mismo. De acuerdo con esta creencia, cuanto más comprendamos el proceso de desarrollo, más capacitados estaremos para controlarlo.

En este trabajo haremos uso de la evolución de las métricas del software para alcanzar un entendimiento más claro sobre el proceso de desarrollo.

Basándonos en la observación de la evolución métrica, estamos interesados en definir el comportamiento de la evolución de las métricas del software, usando funciones matemáticas simples, que dependan del tiempo de desarrollo. A partir de esta observación, se propone un modelo de predicción del esfuerzo, basado en la definición del modelo de evolución de la métrica.

CAPITULO 2

ANTECEDENTES Y ESTRUCTURA

2.1 DEFINICIONES Y TERMINOLOGIA

En este trabajo usamos indistintamente los términos "programa" y "software", entendidos ambos como "producto de programación", de modo que incluye principalmente, el código fuente, los cuadernos de carga, las especificaciones de diseño, los juegos de ensayo, los controles de calidad, y los informes de problemas encontrados.

Los términos "dirección", "control" y "gestión" de proyectos se utilizan indistintamente.

Entendemos que el ciclo de vida de un proyecto de software está formado por las siguientes etapas [YAU88]:

1.- Definición del Sistema
2.- Análisis
3.- Diseño
4.- Instrumentación
5.- Prueba del Sistema
6.- Mantenimiento

Entendemos que en la Fase de Análisis se realiza el Plan del Proyecto y la definición de los requisitos del mismo. La fase de

diseño está formada por dos etapas, el diseño estructural y el diseño detallado. Dentro de esta última, se realiza la identificación de componentes de la programación, es decir, funciones, flujos de datos, almacenamientos, etc. En la fase de instrumentación se realiza la codificación y depuración de los programas diseñados en la etapa anterior. En la última fase del ciclo de vida inicial, la prueba del sistema, se realiza la integración y adaptación de los procesos desarrollados.

A la largo de la vida de un determinado proyecto de software, la fase de mantenimiento puede originar modificaciones que obligen a repetir totalmente o en parte las fases del ciclo indicadas. En la figura 2.1 mostramos el ciclo de vida de un proyecto de software [YAU88].

A lo largo de esta tesis, entendemos por fase inicial, el final de la etapa de diseño. Consecuentemente, una medición inicial o temprana se entenderá como una medición realizada al final de la etapa de diseño; y una estimación inicial será una estimación realizada al final de la misma.

Adviértase también que en algunas de las figuras que se muestran a lo largo de la tesis, el "principio" y "final" se refieren al comienzo y al fin del proceso de desarrollo de un proyecto de una sola persona en una escala pequeña. Además de esto, el "TIEMPO DE DESARROLLO" representa el esfuerzo de desarrollo y el "principio" se corresponde con el inicio de la fase de diseño (o final de la fase de especificación).

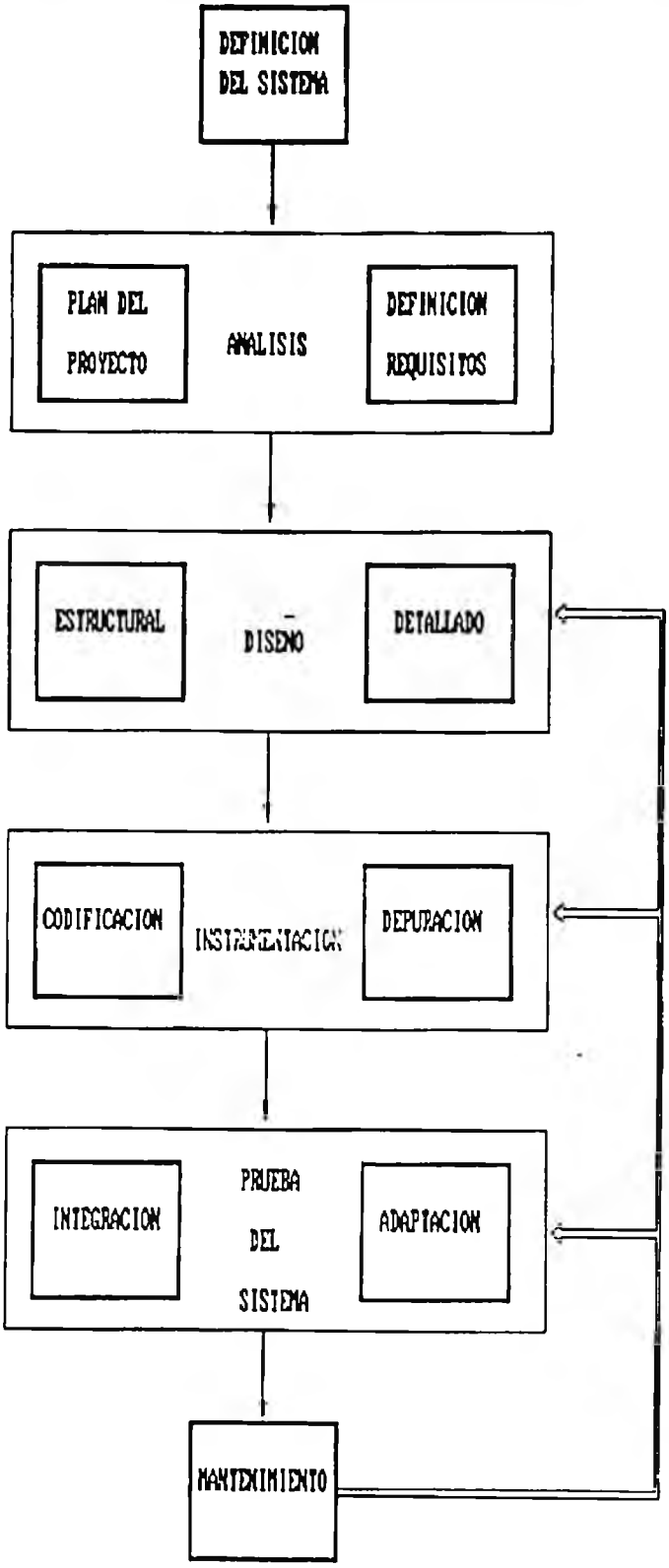


Figura 2.1

Ciclo de Vida de un Proyecto de Software

El número de líneas de código fuente contenidas en un programa se abrevia, en la literatura sobre lás métricas del software, como LOC (lines of code, y seguiremos usando las siglas inglesas), y se define como el número total de líneas de programa, eliminando líneas de comentario y líneas en blanco. La notación S representa el tamaño real del programa en LOC, y la notación S' representa el tamaño estimado del programa en LOC.

En la Figura 2.2 mostramos la importancia de la complejidad del software en relación con otros componentes del mismo [LICH87].

2.2 TRABAJOS PREVIOS SOBRE LA ESTIMACION DEL ESFUERZO

Las labores previas en este área de investigación pueden caracterizarse por el gran número de modelos de estimación del esfuerzo propuestos [MOHA81]. Estos incluyen el modelo Doty [DOTY77], el modelo Walston-Felix, [WALS77], el modelo Putnam SLIM [PUTN78], el modelo RICA PRICES [FREI79], y por último el modelo COCOMO [BOEH81]. En todos estos modelos se indica que el esfuerzo en el desarrollo del software esta funcionalmente relacionado con:

- Los atributos de los productos (e.j. tamaño y complejidad).

- Los atributos del personal (e.j. capacidad y experiencia de los programadores).

- Los atributos del hardware (e.j. tiempo de ejecución, y necesidades de almacenamiento).

- Los atributos orientados al proceso (e.j. necesidades de ejecución y grado de uso de las herramientas de software).

La mayoría de los modelos propuestos se obtuvieron a través del análisis de los datos de desarrollo del proyecto, recopilados después de la finalización del mismo. Es decir, estos modelos son realmente modelos analíticos de "después del hecho" que se usan de modo primario, para identificar factores que han influenciado de modo significativo el esfuerzo de desarrollo de un proyecto ya completo. Los factores que están relacionados de manera fundamental con el esfuerzo de desarrollo del proyecto, se han identificado habitualmente como parámetros tipo y el modelo ha sido debidamente calibrado utilizando información histórica. Así, y de acuerdo con cada modelo, el esfuerzo de desarrollo de un proyecto finalizado, se puede explicar basándose en los parámetros definidos en dicho modelo.

El Modelo Constructivo de Costos, COCOMO (Constructive Cost Model) [BOEH81], es un modelo de costos por algoritmos descrito por Boehm. En estos tipos de modelos, se parte del hecho de que todo proyecto o sistema está formado por la unión de n subsistemas; la estimación se realiza mediante la suma de los

costos de cada uno de los subsistemas pertenecientes al sistema, de modo que esta técnica es de tipo jerárquica hacia arriba (Down-top). El procedimiento para la estimación de costos usando COCOMO incorpora los siguientes pasos:

1.- Identificar todos los subsistemas y los módulos del producto.

2.- Estimar el tamaño de cada módulo y calcular el tamaño de cada subsistema y de todo el sistema.

3.- Especificar los factores multiplicadores de módulos para cada uno; éstos son: la complejidad del producto, la capacidad de programación, la experiencia en máquinas virtuales y la experiencia en lenguajes de programación.

4.- Calcular el esfuerzo para cada módulo, así como el tiempo de desarrollo.

5.- Especificar los once multiplicadores restantes de cada subsistema.

6.- De los pasos 4 y 5, calcular el esfuerzo y el tiempo de desarrollo estimados para cada subsistema.

7.- Del paso 6, calcular el esfuerzo y el tiempo de desarrollo para todo el sistema.

8.- Efectuar un análisis de sensibilidad sobre la estimación, estableciendo comparaciones para diversos factores.

9.- Sumar los otros componentes en el costo de desarrollo, como la planificación y el análisis, que no se hayan incluido antes.

10.- Comparar la estimación con otra obtenida a partir de la técnica DELFI [HELM66], identificando y corrigiendo las diferencias de estimación.

Tal vez el aspecto que reporte mayor beneficio de los modelos de estimación por medio de módulos, reside en la atención puesta en la recolección y análisis de esos datos históricos.

2.3 ESTIMACION DEL TAMAÑO

Según nos indica Boehm [BOEH81], los modelos de estimación del esfuerzo más corrientes se basan en factores que se pueden determinar de forma adecuada, en una fase inicial del proceso de desarrollo del proyecto, con la excepción del factor del tamaño del proyecto. Se considera que la estimación inicial del tamaño del proyecto es algo realmente difícil. Por lo tanto, un paso esencial para la estimación del esfuerzo es la estimación del tamaño. Podemos utilizar un modelo de estimación del esfuerzo para predecir el tamaño de un proyecto. Si estamos en este caso, necesitamos determinar los valores de todos los parámetros del

modelo, para poder obtener una estimación del esfuerzo inicial basada en dicho modelo.

Desafortunadamente, todos los modelos del esfuerzo propuestos hasta ahora, usan el tamaño del proyecto como uno de los parámetros más importantes [MOHA81, BOEH84, RAMA88]. En estudios realizados por Wolverton y Boehm [WOLV74, BOEH81], el tamaño del proyecto incluía alrededor del 50% de la variación posible en el esfuerzo del proyecto. En el metamodelo propuesto por Bailey y Basili [BAIL81], el primer paso en la construcción de un nuevo modelo del esfuerzo, que se usará en determinadas circunstancias, es determinar la relación existente entre el tamaño y el esfuerzo. De esta manera, aunque el tamaño no es el único factor que determina el esfuerzo del proyecto, su estimación inicial es definitivamente el aspecto más dificil e importante en una estimación inicial de dicho esfuerzo. Por esta razón, se ha reconocido que la estimación del tamaño del software es uno de los resultados más importantes en la investigación de estimaciones del esfuerzo [BOEH84, LICH87, RAMA88]. Uno de los puntos clave de nuestra investigación se encuentra centrado en esta característica. Intentamos desarrollar las técnicas para la estimación del tamaño del programa en una fase inicial del proceso de desarrollo.

En la industria, la construcción del software consiste en una fase de diseño seguida por la codificación y verificación de los diversos componentes del mismo (lo que algunos denominan

implantación), seguido por la integración de los componentes más importantes dentro del producto final. Nuestra investigación se ha llevado a cabo en un ambiente universitario, donde la norma son los pequeños proyectos de programación unipersonales. Para este tipo de proyectos, la integración no suele ser necesaria y el diseño y la ejecución son las actividades dominantes, susceptibles de estudio y análisis.

En este trabajo, los resultados y las observaciones que presentaremos, se ocupan predominantemente del proceso de desarrollo de software para estos pequeños proyectos. Cuando posterioemente hablemos de la estimación inicial del tamaño y del esfuerzo, queremos dejar claro que este punto inicial, se produce siempre después de que se haya realizado el trabajo de diseño total o bien una parte substancial del mismo.

Las cifras del tamaño y del esfuerzo que nos ocupan no incluyen los efectos de una fase de integración, por lo tanto nuestros resultados se aplican esencialmente a esta visión de micromodelo, más que a una visión de macromodelo privativa de la mayoría de los modelos (como el Doty, Walston y Felix, SLIM, COCOMO, y otros). Reconocemos que la generalización de nuestro trabajo a un ambiente industrial, donde el diseño y la investigación son partes significativas de la actividad, requerira un estudio posterior. De cualquier modo, tal y como sugerimos más adelante, creemos que la comprensión de una parte del ciclo de vida del software, nos ayudará a extender nuestro conocimiento a otros componentes.

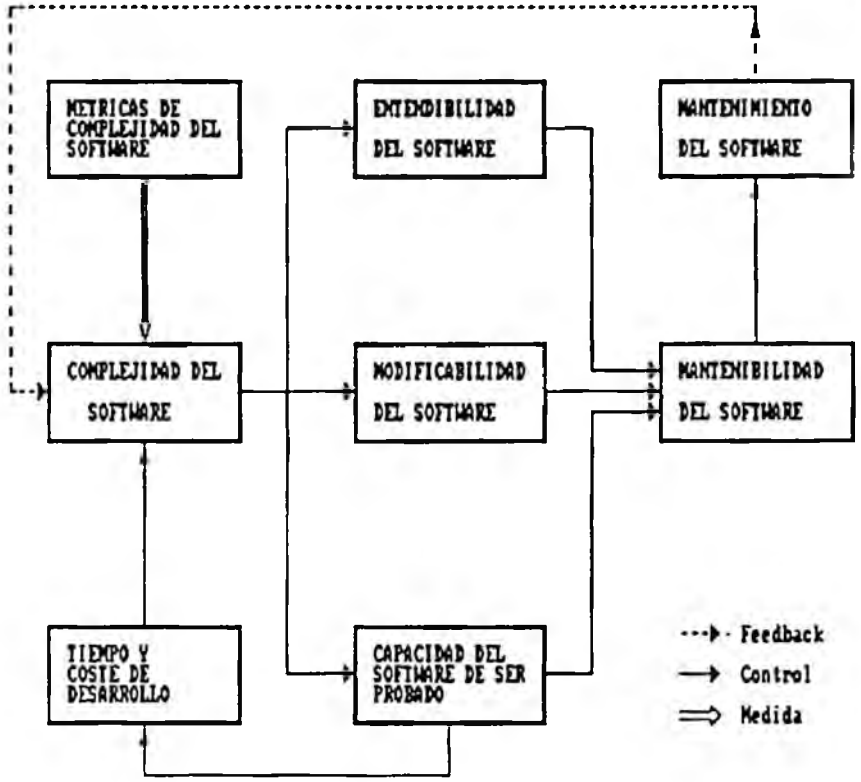


Figura 2.2

La Complejidad del Software

2.4 PROCESO DE INVESTIGACION

2.4.1 Estimación Indirecta del Tamaño

La aproximación a la estimación del tamaño emprendida en esta investigación es lo que llamamos "aproximación indirecta". En primer lugar, vamos a suponer que se puede aproximar el tamaño del programa como una función de variables medibles contenidas en dicho programa. En segundo lugar, asumiremos que podemos desarrollar el programa de tal manera, que estas variables se puedan medir en una primera fase. Así pues, está claro que se puede obtener la estimación inicial del tamaño de modo indirecto, mediante la medición y combinación de estas variables en una fase inicial. Detalles de dicha aproximación, asi como resultados de las investigaciones empíricas se describen en los capítulos tercero y quinto de este trabajo.

2.4.2 Evolución de la Métrica del Software

El segundo tema abordado en esta tesis es la evolución de la métrica del software durante el desarrollo del programa. Como ya hemos indicado en el Capitulo 1, consideramos que una métrica del software es una medida que nos permite cuantificar una cierta propiedad del programa.

Las propiedades frecuentemente caracterizadas por la métrica del software incluyen tamaño, estructuras de control y estructuras de datos. Las métricas familiares de sofware que

caracterizan estas propiedades del programa son para el tamaño, las "líneas de código" [JONE78, DUNS84a], y la longitud de programa de Halstead [HALSA77]; para las estructuras de control, "el número ciclomático" de McCabe [MCCA76]; y para las estructuras de datos, el "contador de operando único" de Halstead [HALS77]. Estas métricas (y otras como ellas) han introducido pautas cuantitativas para la estimación de la complejidad, el coste y la fiabilidad del software finalizado.

Supongamos que los valores de una cierta métrica se pueden medir periódicamente durante el proceso de desarrollo. Cuando alineamos los valores métricos sucesivos frente al tiempo de desarrollo correspondiente transcurrido, tenemos una "línea de evolución" que nos da el comportamiento de la evolución de dicha métrica a lo largo del tiempo. Por ejemplo, la Figura 2.3 contiene la línea de evolución del tamaño del programa en líneas de código, (LOC), medidas durante un proceso real de desarrollo de uno de nuestros experimentos. El eje horizontal representa el tiempo de desarrollo en horas; el eje vertical representa el tamaño del programa en LOC. El tiempo total de desarrollo de este programa fue aproximadamente de 24 horas y el tamaño final del programa fue de 344 LOC excluyendo comentarios. Se tomaron seis mediciones. Esta figura muestra que, para este programa, el tamaño se ha incrementado aproximadamente de forma lineal con el tiempo de desarrollo.

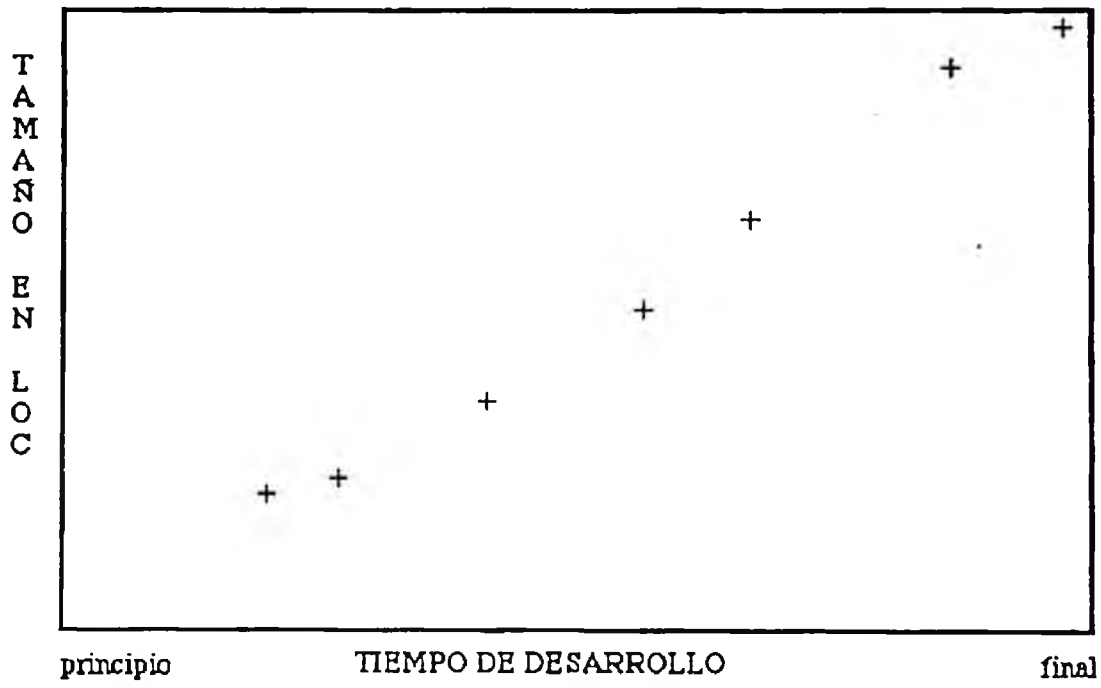


Figura 2.3

Evolución del Tamaño

En este trabajo, haremos uso de la evolución de las métricas del software para alcanzar un mayor entendimiento acerca del proceso de desarrollo. La línea de evolución métrica es una representación gráfica de dicho proceso y nos puede revelar algunos aspectos importantes en el mismo. Por ejemplo, si observamos que una métrica no sigue su curva típica, esta puede ser una señal de que el proceso necesita algún grado de atención. En este estudio, observamos e identificamos varios modelos distintivos de la evolución. Los resultados se presentan en el Capítulo 6.

2.4.3 Modelo de Predicción del Esfuerzo

En base a la observación de la evolución métrica, estamos interesados en definir el comportamiento de la evolución de las métricas del software, usando tiempos de desarrollo de funciones simples. Proponemos un modelo de predicción del esfuerzo basado en la definición del modelo de evolución métrica. Las ideas básicas sobre el estudio de evolución métrica se presentan en el Capítulo 4. Los resultados de la investigación empírica sobre la evolución de la métrica se pueden encontrar en los Capítulos 6 y 7. El modelo de predicción del esfuerzo basado en la definición del comportamiento de evolución de la métrica, aparece en el Capítulo 7.

2.5 ESTUDIOS EMPIRICOS SOBRE EL PROCESO DE PROGRAMACION

Programar es una tarea humana muy compleja. Ningún estudio del proceso de programación puede estar completo sin la investigación empírica. Una manera natural de validar un modelo propuesto para el proceso de programación, es considerar las estadísticas de desarrollo actual desde el punto de vista de proyectos de software a gran escala. En cualquier caso, los proyectos a gran escala se verán afectados generalmente por otro tipo de factores (como entorno de programación, características del proyecto, comunicaciones, etc) que causarán confusión, y que serán difíciles de controlar. Cualquiera de estos factores incontrolables puede encubrir los efectos de los factores que realmente están siendo estudiados. Adicionalmente, los investigadores de la estimación del esfuerzo se enfrentan con un serio problema como lo es la falta de mecanismos para recopilar datos que se producen durante el proceso de desarrollo del proyecto. Actualmente, las herramientas C.A.S.E. (Computer-aid Software Engineering), ayudan a solucionar gran parte del problema, ya que permiten automatizar muchos aspectos del proceso de desarrollo. La dificultad de emplear estas herramientas, reside en el hecho de que llevan consigo la utilización de ciertas metodologías de diseño [MYNA90].

Habitualmente, la finalidad de un proceso de desarrollo es obtener un producto terminado de gran calidad, tan pronto como sea posible. Por comparación con esto, el mantenimiento y almacenamiento de información parace ser una tarea de mucha menor

envergadura, que incluso se puede ignorar. Así y debido al común énfasis que se da a una retribución económica rapida durante el desarrollo del software, muchas estadísticas cruciales producidas en el mismo desarrollo se pierden durante el proyecto. Como resultado, varios estudios de modelos de estimación de proyecto [MOHA81, BOEH82, THEB83] se basan en estadísticas de desarrollo final, opuestas a las estadísticas producidas durante el proceso de desarrollo [THAY84].

Otro método atractivo para validar los modelos propuestos de procesos de programación, es el conducir experimentos controlados que impliquen tareas de programación que no sean triviales. Nuestra aproximación ha tenido que conducir experimentos controlados que impliquen tareas de programación, tan grandes como sea posible, con programadores estudiantes con la mayor experiencia posible. Hay que tener en cuenta que las restricciones económicas y de tiempo limitan severamente la magnitud de tales experimentos.

Es importante darse cuenta de los beneficios y limitaciones de los estudios empíricos conducidos dentro de un entorno académico. Como parte negativa, destacamos que las generalizaciones derivadas de tales estudios pueden ser bastante limitadas. Debido a algunos controles artificiales, los resultados que se derivan de las condiciones de "laboratorio" pueden no ser aplicables directamente a situaciones en el mundo "real". De cualquier modo, como parte positiva vemos que los

controles rigurosos permiten a los investigadores realizar sus estudios con todo detalle y así, ver los efectos de los factores manipulados experimentalmente e identificar las posibles relaciones de causa-efecto entre ellos, y la importancia relativa de cada uno de estos factores. Así, tal y como ocurre en otras disciplinas científicas, se pueden usar los resultados obtenidos de experimentos bien controlados, para alcanzar un mejor entendimiento de un factor aislado que esté siendo probado. Adicionalmente, las percepciones obtenidas de estos estudios pueden formar una base firme para estudiar proyectos a gran escala que impliquen factores adicionales. Por lo tanto, creemos que un estudio académico es el paso esencial antes de realizar estudios empíricos industriales posteriores.

Durante los últimos años se han conducido varios experimentos controlados que han recogido datos empíricos a lo largo del proceso de programación. Una característica común, compartida por estos experimentos, es que sólo pequeños programas, que no requieren más de cinco horas para ser finalizados podían utilizarse en este proyecto. Para dar un paso adelante en nuestra investigación, realizamos dos experimentos de elaboración de programas, que implicaban programas con tamaños del orden de quinientas líneas de código. El tiempo total para completar un programa fue de sesenta horas a lo largo de un período de dos semanas. El primer experimento, que se discutirá en el Capítulo 3 y al que nos referiremos como EXPER-1, fue un estudio exploratorio. Después de análizar los resultados de este experimento, especulamos con la posibilidad de que algunos de los

resultados experimentales estuviesen confundidos con algunas condiciones experimentales incontroladas. Debido a esto, llevamos a cabo el segundo experimento, que se discute en los Capítulos del 4 al 7, y al que llamamos EXPER-2, fue realizado como mejora del primer experimento. Nuestra esperanza era que EXPER-2 confirmase además algunos de los hallazgos descubiertos en el primer experimento.

2.6 DISEÑO DE LA TESIS

Esta tesis está dividida en ocho capítulos. La tabla 2.1 resume los temas abordados así como los capítulos correspondientes. Tal y como ya se ha señalado, el estudio de la estimación del tamaño realizado en esta tesis está cubierto en los Capítulos 3 y 5. Las ideas básicas del estudio de la evolución métrica se explican en el Capítulo 4. Los resultados de la observación y definición de la evolución métrica del modelo se indican en el Capítulo 6. Los resultados de la estimación inicial del esfuerzo basados en la evolución de la métrica estan en el Capítulo 7, donde damos paso a nuestro modelo de predicción del esfuerzo. Finalmente, el sumario y la conclusión aparecen en el Capítulo 8, acompañados de una serie de sugerencias para futuros trabajos.

Tabla 2.1

Diseño de la Tesis

Trabajo de Investigación	Capítulos
Estimación del tamaño indirecto	3 y 5
Evolución de la métrica del software	4,6, y 7
Modelo de predicción del esfuerzo	7
Estudio empirico del proceso de programación (EXPER-1)	3
Estudio empírico del proceso de programación (EXPER-2)	4,5,6, y 7

CAPITULO 3

ESTUDIO PRELIMINAR DE LA ESTIMACION DEL TAMAÑO

Tal y como se ha discutido en el Capítulo 2, podemos decir que la obtención de una estimación indirecta del tamaño en una fase inicial es, basicamente, hacer una aproximación al tamaño como función de alguna característica o propiedad medible del programa. Para realizar el presente estudio, hemos de seleccionar una característica que pueda ser determinada en una fase inicial del proceso de desarrollo.

A lo largo de las investigaciones realizadas para realizar estimaciones del tamaño, los investigadores han propuesto diversas características de programa que se pueden usar con ese fin.

En este capítulo, empezamos por repasar brevemente las aproximaciones realizadas. Después discutiremos un experimento piloto realizado en la Universidad de Deusto durante el verano de 1988, que denominaremos EXPER-1, y concluiremos con lo aprendido en este experimento y como ha influenciado en nuestro trabajo posterior.

3.1 TRABAJO PREVIO SOBRE LA ESTIMACION DEL TAMAÑO

En esta sección vamos a comentar algunas de las métricas del software ya mencionadas en el capitulo anterior.

3.1.1 La Ciencia del Software de Halstead

Halstead propuso varias métricas que se calculaban con facilidad en base a propiedades simples de obtener a partir del código fuente [HALS77]. Estas propiedades, reflejadas en La Ecuación del Esfuerzo de Halstead incluyen:

n_1 = Número de operadores únicos

n_2 = Número de operandos únicos

N_1 = Número total de operaciones

N_2 = Número total de operandos

Los operandos incluyen variables, constantes y series de caracteres. Un operador es cualquier cosa que tiene efecto sobre los operandos. El vocabulario del programa n, incluye a n_1 y n_2, es decir $n = n_1 + n_2$.

En las Figuras 3.1 y 3.2 ilustramos dos programas sencillos realizados en PASCAL y los valores asociados de N_1, N_2, n_1, y n_2.

El Programa 1 tiene 11 operadores diferentes, son; "BEGIN END", "readln", "()", ",", ";", ":=", "+", "*", "-", "writeln", y ".". Estos 11 operadores se usan un total de 23 veces. Por lo tanto, $n_1 = 11$, y $N_1 = 23$. Existen 5 operandos distintos que se identifican en la instrucción VAR, y se usan 18 veces a lo largo del programa. Por lo tanto, $n_2 = 5$, y $N_2 = 18$.

```
PROGRAM 1(input,output);

VAR
   a,b,c,d,m:integer;          N = N  + N  = 23 + 18 = 41
BEGIN                               1    2
   readln(a,b,c,d);
   a:=a+a;                     n = n  + n  = 11 + 5 = 16
   b:=b+b;                         1    2
   c:=c*d;
   m:=a+b-c;
   writeln(m)
END.
```

Figura 3.1

Programa 1

El Programa 2 tiene 12 operadores diferentes, son; "BEGIN END", "IF THEN ELSE", "readln", "()", ",", ";", ">", ":=", "+", "*", "writeln", y ".". Estos 12 operadores se usan un total de 25 veces. Por lo tanto, $n_1 = 12$, y $N_1 = 25$. Existen 5 operandos distintos que se identifican en la instrucción VAR, y se usan 22 veces a lo largo del programa. Por lo tanto, $n_2 = 5$, y $N_2 = 22$.

```
PROGRAM 2(input,output);

VAR
   a,b,c,d,m:integer;          N = N  + N  = 25 + 22 = 47
BEGIN                             1    2
   readln(a,b,c,d);
   IF a>b THEN                 n = n  + n  = 12 + 5 = 17
      IF b>c THEN                  1    2
         m:=a+b
      ELSE
         m:=b+c
   ELSE
      m:=b+c+d;
   m:=m*m;
   writeln(m)
END.
```

Figura 3.2

Programa 2

La longitud de programa N, Ecuación de Longitud de Halstead, se define como el número total de apariciones en el programa de todos los operadores y operandos:

$$N = N_1 + N_2 \qquad (3.1)$$

Smith demostró [SMIT80] que S (el tamaño del programa en LOC) y N estaban muy relacionados. En el estudio realizado para demostrarlo, se trabajó con módulos de programas realizados en PL/S y en Ensamblador. Los coeficientes de correlación que se obtuvieron entre S y N fueron 0,952 para PL/S y 0,985 para Ensamblador. Además, se mostró que una línea de código fuente en Ensamblador tenía un promedio de 4,7 operadores y operandos, y

una línea de código fuente en PL/S tenía un promedio de 5,1. Estos resultados sugieren que N se puede convertir en S mediante la relación S = N/c, donde c es una constante de valor aproximado a 5,1 para PL/S, y a 4,7 para ensamblador.

Halstead supuso que N es una función que dependía unicamente de n_1 y n_2 y que podía ser estimada antes de que N_1 y N_2 estuvieran disponibles. La fórmula que propuso para estimar N basándose en el vocabulario del programa, es la siguiente:

$$\text{Longitud Estimada} = N' = n_1 \log n_2 + n_2 \log n_2 \qquad (3.2)$$

La ecuación (3.2) se denomina "Ecuación de longitud de Halstead".

No podemos justificar la relación entre tamaño de programa y vocabulario aludiendo únicamente a bases teóricas. De hecho, es fácil construir un programa patológico para hacer de N' una pobre predicción de N. Durante los últimos años, se ha ido acumulando una cantidad substancial de evidencia empírica que apoya la validez de esta relación para una serie de lenguajes [SHEN83, LICH87, WEYU88, ARME89]. Se ha comprobado que la métrica N' es un estimador aceptable de N cuando se aplica a un amplio abanico de programas [CHRI81]. En un estudio realizado sobre 1637 módulos en lenguaje ensamblador procedentes de una serie de productos comerciales sobre una amplia gama de aplicaciones, el error relativo entre N y N' era menor del 6%, aunque este error era

mucho mayor para algunos módulos individuales [SMIT80]. El estudio de dichas observaciones sugiere que podemos acercarnos al tamaño de programa por medio de N, que es una función de operadores y operandos únicos.

3.1.2 El Módelo de Tamaño de Itakura y Takayanagi

Itakura y Takayanagi han realizado un modelo de estimación de tamaño basado en datos elementales de entrada/salida [ITAK82]. De acuerdo con dicho modelo el tamaño del programa puede predecirse utilizando la siguiente fórmula:

$$S = 105 + 8X_1 + 1{,}3X_2 + 8X_3 + 3{,}3X_4 + 131X_5 + \sum_{i=1}^{X_5}(4{,}6X_{6i} + X_{7i} + 80X_{8i} + 61X_{9i}) + X_{10}$$

Donde:

S = Número de líneas en código fuente

X_1 = Número de ficheros de entrada

X_2 = Número de campos de entrada

X_3 = Número de ficheros de salida

X_4 = Número de campos de salida

X_5 = Número de impresos

X_{6i} = Número de campos horizontales en el impreso i.

X_{7i} = Número de campos verticales en el impreso i

X_{8i} = Número de procesos de cálculo en el impreso i

X_{9i} = Número de clasificaciones en el impreso i

X_{10} es una función de X5 y X6i.

Todos los coeficientes del modelo estaban determinados por expertos que habían escrito antes programas similares. Por lo tanto, si utilizamos este modelo, podremos estimar el tamaño tan pronto como podamos estimar, con precisión razonable, los valores de los parámetros del modelo.

Este modelo se ha validado utilizando 38 programas que podemos englobar en el ámbito de la gestión, de tamaño entre pequeño y mediano, pertenecientes a una organización de proceso de datos y realizados en lenguaje Cobol. Al final de la fase de diseño se determinan los valores de los parámetros del modelo en base a los documentos especificados y a los documentos diseñados [ITAK82].

Utilizando este modelo de estimación, el error relativo entre la suma de los tamaños reales de los programas y la suma de los tamaños estimados, que se obtuvieron al final de la fase de diseño, era del 7%. Esto nos indica que este modelo da buenos resultados para el conjunto de programas estudiados [ITAK82].

3.1.3 Puntos de Función de Albrecht

Albrecht propuso una medida llamada "Puntos de Función", [ALBR83], como base para predecir el tamaño del software. El número de "puntos de función" es la suma total del numero de "entradas", "salidas", "ficheros maestros", y "formatos de pantallas" que se han previsto en el software. Albrecht utiliza los siguientes pesos para determinar los "puntos de función":

F = (número de entradas) x 4
+ (número de salidas) x 5
+ (número de formatos de pantallas) x 4
+ (número de ficheros maestros) X 10

donde F es el número de "puntos de función". El número de puntos de función puede estimarse en una fase inicial del ciclo de desarrollo, a partir de las especificaciones sobre los requerimientos básicos, o a partir de los documentos de diseño detallado.

En un análisis realizado sobre 18 programas en Cobol y 4 en PL/1, pertenecientes a un sistema de gestión, se encontró que los "puntos de función" métrica estaban relacionados con las líneas de código fuente: los coeficientes de correlación eran 0.854 para Cobol y 0,997 para PL/1 [ALBR83]. Este análisis nos permite decir que la estimación del número de "puntos de función" en una fase inicial del proceso de desarrollo, es una aproximación prometedora [ALBR83].

3.1.4 La Métrica Ciclomática de McCabe

A mediados de los setenta, Thomas J. McCabe adaptó un concepto matemático procedente de la teoría de grafos para obtener una nueva métrica de la complejidad del software llamada el "Número Ciclomático de McCabe" o la Médida McCabe [McCA76].

Para cada módulo de un programa estructurado es posible obtener su gráfico correspondiente, denominado grafo de control de programa, con un nodo de entrada único y un nodo de salida único. Los nodos del grafo excepto el de entrada y salida, se corresponden a segmentos de código secuenciales o decisiones en el módulo, y los arcos representan los posibles controles de flujo en el módulo. Para aplicar la teoría de grafos, a cada nodo se debe poder llegar desde los otros nodos. Además, se añade un arco desde el nodo de salida al de entrada. En Las Figuras 3.1 y 3.2 mostrábamos dos programas realizados en PASCAL, en las figuras 3.3 y 3.4 mostramos el grafo de control para dichos programas.

El "número ciclomático", V, de una gráfica conectada es el número de caminos linealmente independientes en la gráfica.

$$V(G) = e - n + 2p$$

donde, "e" es el número de arcos, "n" el el número de nodos, y "p" el el número de componentes conectadas.

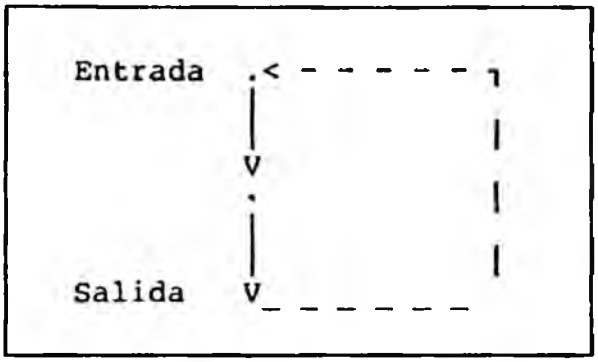


Figura 3.3

Grafo de Control del Programa 1

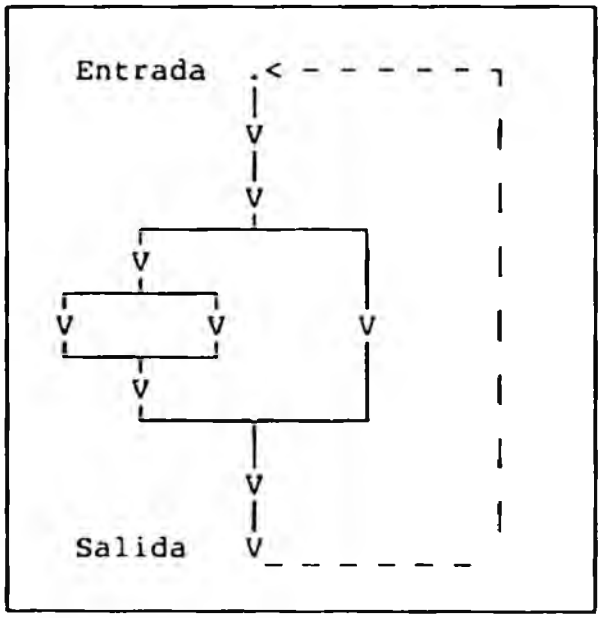


Figura 3.4

Grafo de Control del Programa 2

Aplicando esta fórmula, el número ciclomático de cada uno de los programas 1 y 2 (figura 3.1 y figura 3.2) es:

Programa 1: V(G) = 3 - 3 + 2 = 2

Programa 2: V(G) = 13 - 11 + 2 = 4

Resumiendo, el proceso de predicir el tamaño del programa basado en el vocabulario del programa (operadores y operandos únicos) o en datos elementales de entrada/salida ha tenido un éxito moderado (Halsdtead, Itakura).

Albrecht ha propuesto una aproximación para predecir indirectamente el tamaño de un programa partiendo de cualquier tipo de especificación requerida o documento diseñado. De cualquier modo, apenas hay evidencias que muestren que esta aproximación sea factible.

La Métrica Ciclomática de McCabe es buena para medir la complejidad del flujo, pero su debilidad es la medición del tamaño. Todo programa compuesto por instrucciones secuenciales tendrá el mismo "número ciclomático" que el Programa 1 (Figura 3.1) independientemente del tamaño.

3.2 ELECCION DE UNA APROXIMACION PARA LA ESTIMACIÓN DEL TAMAÑO

Se ha demostrado que la ecuación de longitud de Halstead es válida para programas escritos en diversos lenguajes y con organizaciones diferentes (téngase en cuenta, que las aproximaciones de Itakura y Albrecht han sido validadas usando

predominantemente programas construidos en Cobol, dentro de una organización de datos única). Además, contamos con programas analizadores que calculan los valores de las métricas de "la ciencia del software". Es por ésto, que escogemos como punto de partida para nuestro estudio de la estimación del tamaño la ecuación de longitud de Halstead.

Partiendo de la premisa de que el tamaño del programa medido en N puede ser aproximado mediante una función de operandos y operadores únicos, el paso siguiente es investigar si podemos pronosticar esos dos componentes de N' (la estimación de N) en una fase inicial. Recordemos que cuando hablamos de fase inicial nos estamos refiriendo al final de la etapa de diseño, dentro del ciclo de vida de un proyecto de software.

3.2.1 Estimación Inicial de Operadores Unicos

En los programas grandes y complejos, es muy probable que casi todos los operadores o palabras-clave que forman el lenguaje en el que se realizan dichos programas, estén incluidos en los mismos. Por lo tanto, en este caso, el valor de n_1 debe ser una constante a la que añadimos el número de llamadas a procedimientos, las referencias a funciones, y las instrucciones de bifurcación incondicional (es decir, GOTO). En lenguajes tales como Pascal y PL/S, donde queda descartado el uso de bifurcaciones incondicionales, n_1 debe ser casi constante para los programas grandes y complejos.

En un estudio realizado sobre 490 módulos de PL/S, el valor medio de n_1 fue de 46, con una desviación estandar de 18 [FITS80c]. Para la mayoría de los programas realizados en Pascal analizados en nuestro estudio, el valor medio de n_1 es de 67 con una desviación estandar de 9. En consecuencia observamos, que para programas codificados en lenguajes tales como el Pascal o el PL/S, donde el uso de las bifurcaciones incondicionales esta descartado, n_1 puede aproximarse por medio de una constante predeterminada.

En otras palabras, el tamaño de un programa grande puede ser predicho basándonos únicamente en n_2 y si ésto es cierto, la estimación inicial del tamaño del programa puede ser circunscrita a una estimación inicial de n_2.

3.2.2 Estimación Inicial de Operandos Unicos

La métrica n_2, tal y como se define en "la ciencia del software", es el número de operandos únicos contenidos en un programa. En un lenguaje de alto nivel, la mayoría de los operandos son las variables reales usadas en el programa.

Queremos construir un programa de tal manera que n_2 pueda estimarse en una primera fase del proceso de desarrollo. Para lograrlo utilizamos la estrategia de desarrollo denominada "Primera Estructura de Datos Top-Down". Utilizando esta estrategia, el proceso de diseño de programas incluye las actividades siguientes:

1.- Identificar la estructura subyacente y los componentes más importantes del programa

2.- Especificar las conexiones de datos y de procedimientos entre los componentes

3.- Establecer un esquema detallado de las representaciones de datos y del desarrollo de los algoritmos básicos para su manipulación.

Siguiendo estas actividades, un gran porcentaje de las estructuras de datos y variables se pueden identificar y desarrollar al final de la etapa de diseño. Después de dicha etapa, el número de nuevas estructuras de datos y variables introducidas es, en proporción, muy pequeño.

Las estrategias de diseño Top-down ponen atención inicialmente en los aspectos globales de todo el sistema, conforme el diseño progresa, el sistema se descompone en subsistemas, poniéndosele mayor consideración a los detalles específicos. El backtracking (retroceso) resulta fundamental en este tipo de diseño. Conforme las decisiones se descomponen en niveles más elementales puede resultar aparente que una decisión de alto nivel genere una ineficiencia o desorganización; así, una decisión puede tener que ser reconsiderada y el sistema reestructurado convenientemente.

Suponemos que utilizando la estrategia de desarrollo de programas "Primera Estructura de Datos Top-Down", el valor de n_2 al final de la etapa de diseño, podría ser aproximadamente el 80% del valor final. Este número está basado em nuestra propia especulación. Sin embargo, los resultados obtenidos en el Capítulo 5, mostrarán que es una buena decisión para el conjunto de n_2 que denominamos VAR.

Si la premisa es cierta, el valor final de n_2 puede estimarse en 1,25 veces el valor de n_2 obtenido al final de la fase de diseño (ya que 100/80=1,25). Por ejemplo, supongamos que el valor inicial de n_2 es 164. En este caso el valor final de n_2 se podría estimar en 205 (=164 x 1,25).

Podemos decir entonces que los operandos en un lenguaje de alto nivel representan las variables reales usadas en el programa. Las variables incluirían entradas, salidas, bloques de control, áreas de comunicación, áreas de trabajo, tablas y similares. Además, la mayoría de las labores que se realizan en el diseño de programas, crean una representación detallada de los datos elementales que se deben procesar. Las representaciones de los datos se pueden usar para estimar el tamaño del programa.

En las secciones restantes de este capítulo se muestran los procedimientos experimentales y los resultados obtenidos de un experimento realizado para confirmar lo expuesto.

3.3 TRABAJO BASICO PARA EL ESTUDIO EMPIRICO

3.3.1 Participantes

En este experimento, al que denominamos EXPER-1, participaron 21 estudiantes de Informática que realizaban un curso monográfico sobre metrícas del software durante el Verano de 1988 en la Universidad de Deusto. Todos las materias incluidas en el experimento habían sido tratadas en los diferentes cursos relativos a las áreas de Sistemas Informáticos y Programación.

3.3.2 Tarea

Se seleccionaron cuatro programas que se iban a realizar en PASCAL. Los cuatro programas tenían una dificultad parecida; se seleccionaron del temario de prácticas de los cursos avanzados de programación. Eran lo bastante cortos para que cada uno de ellos se pudiera realizar dentro del límite de tiempo planteado (dos o tres semanas). Por otra parte, eran lo suficientemente largos para que los resultados del experimento tuvieran algún significado. Se esperaba que el tamaño de cada programa fuera aproximadamente de quinientas líneas de código, excluyendo comentarios.

A cada participante se le asignó un programa y se le concedieron un máximo de tres semanas de tiempo para completarlo. Las características de estos programas se describen brevemente a continuación:

A.- Código de Huffman:

Códificar y decodificar un fichero que contiene prosa usando el código Huffman y los algoritmos de Huffman tal y como se describen en [HORO77]. Este programa se asignó a seis sujetos seleccionados al azar.

B.- Calculador:

Toma expresiones aritméticas convencionales, las traduce a lenguaje postfijo, y evalua su valor. Este programa fue asignado a cinco sujetos seleccionados al azar.

C.- Intérprete de LISP:

Un intérprete de un subconjunto del lenguaje LISP, escrito en Pascal. Este programa se asignó a cinco sujetos seleccionados al azar.

D.- Traductor:

Traduce un programa realizado en un lenguaje simple de bases de datos (DBL) a un programa Pascal. Este programa se asignó a 5 sujetos seleccionados al azar.

3.3.3 Proceso de Desarrollo

Era importante que los participantes en este experimento además de construir programas, siguieran también los procedimientos experimentales prescritos en el desarrollo de sus programas. Se les pidió que hicieran un esfuerzo para seguir dichos procedimientos, incluso en el caso de que no fuera su "estilo favorito" de programación. El proceso de desarrollo de programas en este experimento fue considerado como un procedimiento de tres etapas:

1.- Etapa de Especificación.

A lo largo de esta etapa, los participantes tenían que leer las especificaciones del programa cuidadosamente hasta obtener una buena idea de los requerimientos del programa.

2.- Etapa de Diseño.

A lo largo de esta etapa, los participantes tenían que diseñar sus programas usando la estrategia de

desarrollo "primera estructura de datos Top-Down" tal y como se ha descrito en la sección previa. Esto quiere decir, que en esta etapa, deberían identificar la estructura básica y la mayoría de los componentes de sus programas, determinar las interconexiones de datos y procesos entre los componentes, identificar tanto las variables globales como las locales y las estructuras de datos, y hacer un esquema de los algoritmos básicos para poder manipularlos. Al final de esta etapa, se pidió a los participantes que nos entregarán una versión del programa sin errores de sintaxis que incluyera el diseño completo del programa utilizando la estrategia de desarrollo ya mencionada.

3.- Etapa de Implementación.

A lo largo de esta etapa, los participantes tenían que modificar sus programas y realizar los cambios necesarios de manera que sus programas funcionaran usando los datos de prueba oficiales.

3.3.4 Captura de Datos

Puesto que estabamos interesados en el tiempo que cada individuo pasaba construyendo su programa, les pedimos que llevaran la cuenta del tiempo total utilizado en cada etapa, comenzando desde el final de la etapa de especificación.

Para poder recopilar los valores iniciales de n_2, los participantes entregaron al final de la etapa de diseño las versiones sintácticamente correctas. Debido a que teníamos dichas versiones, y que el ordenador era capaz de leerlas, los valores iniciales de n_2 podian derivarse de ellas. Además, para investigar con qué exactitud podían los participantes estimar el tamaño de sus programas durante el desarrollo del programa, se les pidió que escribieran sus propias estimaciones del tamaño final del programa (en LOC) al final de la etapa de diseño. Estas estimaciones serán denominadas más adelante como "estimaciones de tamaño directas" ya que proceden "directamente" de los programadores.

3.3.5 Estadísticas

Se analizaron todas las versiones de los programas entregadas. Las estadísticas que resumen las cuatro medidas importantes usadas en nuestro estudio para los 21 programas realizados en EXPER-1 se ofrecen en la tabla 3.1. El apéndice E contiene los datos detallados. Las cuatro medidas importantes incluyen el esfuerzo de desarrollo en horas (E), el tamaño del programa en líneas de código (S), el contador de variable única (VAR), y el contador de operando único (n_2).

Tabla 3.1

Estadísticas del Estudio EXPER-1

Programa	Individuos	Medida	Mín	Máx	Media	Desv.Est.
A	6	E(horas)	19	54	29	13
		S(líneas C.)	243	401	337	57
		VAR(variab.)	49	105	61	22
		n (operando)	76	143	94	25
		2				
B	5	E(horas)	18	31	22	6
		S(líneas C.)	332	523	407	74
		VAR(variab.)	62	119	84	21
		n (operando)	110	159	127	20
		2				
C	5	E(horas)	29	67	52	16
		S(líneas C.)	435	662	566	91
		VAR(variab.)	48	118	78	33
		n (operando)	71	150	110	36
		2				
D	5	E(horas)	10	34	20	9
		S(líneas C.)	215	466	342	91
		VAR(variab.)	50	59	62	16
		n (operando)	121	153	147	27
		2				

3.4 EVALUACION DE LA APROXIMACION

En esta sección y en capítulos posteriores de esta tesis, el éxito de una estimación (es decir, el resultado de un estimador) se evaluará en términos de su capacidad para predecir los valores reales. Para medir el éxito utilizaremos tanto análisis gráfico, como varias medidas diseñadas para reflejar la relación existente entre los valores actuales y los predichos.

Previamente, llegamos a la conclusión de que estas medidas descritas a continuación son las más apropiadas para nuestro estudio.

Debe hacerse notar que no hemos usado en nuestro estudio el coeficiente de correlación. Hay dos razones por las que no lo hacemos: la primera es, que nosotros estabamos interesados en comparar los valores predichos con los reales, y no una mera asociación lineal entre ellos. La segunda es que la mayoría de nuestros datos tienen una distribución cercana a la normal con una parte vacía cercana al final inferior. El coeficiente de correlación entre dos grupos de datos con esta distribución, es normalmente demasiado pequeño para tener un significado a la hora de hacer la comparación.

Como no hay una sola medida que pueda caracterizar completamente el resultado de una estimación, la combinación de estas medidas proporcionan un resumen conciso de los resultados de una estimación.

3.4.1 Medidas de Evaluación

Análisis Gráfico. Puesto que el éxito de una estimación se juzgará de acuerdo con su calidad de ajuste con los datos disponibles, la manera más directa de lograrlo es a través de una gráfica en la cual se puedan representar tanto los datos predichos, como lo datos reales. Por ejemplo, la Figura 3.5

muestra el tamaño predicho en LOC de 21 programas reflejados contra sus tamaños reales. Puesto que los valores máximos de los dos ejes son iguales, si cualquiera de los valores predichos es una buena aproximación del valor real correspondiente, entonces, los puntos correspondientes en el gráfico deben estar próximos a la diagonal que va de izquierda a derecha y de abajo hacia arriba, diagonal llamada la "Diagonal 45". Los puntos situados por encima de esta línea indican una sobrestimación; los puntos bajo la línea indican una subestimación. Por lo tanto, se puede usar una gráfica como la mostrada en dicha Figura 3.5, para visualizar rápidamente la calidad de ajuste entre los valores predichos y los reales. Sin embargo, la interpretación de una gráfica es a menudo demasiado subjetiva para ser de gran valor estimativo.

Promedio del Error Relativo (RE'). El promedio del error relativo se define como:

$$RE' = \frac{1}{n} \sum_{i=1}^{n} \frac{(Y_i - Y'_i)}{Y_i} \qquad (3.3)$$

donde, Y_i e Y'_i son los valores predichos y reales i-ésimos, de una colección de n parejas de valores. El error indivual se representa:

$$\frac{Y_i - Y'_i}{Y_i}$$

para i = 1, 2, 3, . . . , n.

Los errores individuales si son de signo opuesto tienden a cancelarse unos a otros, por lo tanto, si se producen grandes sobrestimaciones y grandes subestimaciones, unas y otras pueden equilibrarse, resultando que el valor de RE' puede ser muy pequeño. Sin embargo, en otras circunstancias, puede indicar anomalías de una estimación. Por ejemplo, un valor grande positivo o negativo de RE' implica que los valores reales son fuertemente sobre/subestimados por el método de estimación dado. Nosotros sugerimos que el valor de RE' de una buena estimación debe ser pequeño (p.ej., $-0,15 \leq RE' \leq +0,15$).

Promedio de la Magnitud del Error Relativo (MRE'). Esta medida se define por:

$$MRE_i = \frac{|Y_i - Y'_i|}{Y_i} \qquad (3.4)$$

y

$$MRE' = \frac{1}{n} \sum MRE_i \qquad (3.5)$$

donde Y_i e Y'_i son los valores ya indicados. Así, para n puntos de datos, MRE' es el promedio del error relativo absoluto entre

los valores predichos y los reales. Por lo tanto, cuanto menor sea el valor de MRE', mejor será la estimación. Puesto que los errores de signo opuesto no se cancelan unos a otros, este criterio debe representar adecuadamente el promedio de éxito de un método de estimación. Consideramos que un valor bajo de MRE' (p.ej., MRE'≤0,25) se corresponde con un resultado de estimación satisfactorio.

Predicción a Nivel L (PRED(L)). La tercera medida es el porcentaje de puntos de información para los cuales MRE ≤ L para un cierto nivel de error L (MRE es lo mismo que MRE_i definido en la ecuación (3.4) para cualquier valor de i). El valor de esta medida puede oscilar entre 0 y el 100%. Por ejemplo, si asumimos que en un conjunto de 60 puntos de información hay 50 de ellos para los cuales MRE ≤ 0,20. Entonces, esta medida puede ser PRED(0,20) = 83% (50 / 60 ≈ 83). Para un nivel dado L, cuanto mayor sea el porcentaje, mejor será la estimación. Consideramos entonces, que un resultado de estimación de PRED(0,25) mayor o igual al 75% debe ser razonablemente exacto. Esto es debido a que para tal resultado, por lo menos un 75% de los valores predichos caen dentro de un rango de error pequeño (25%) de los valores reales.

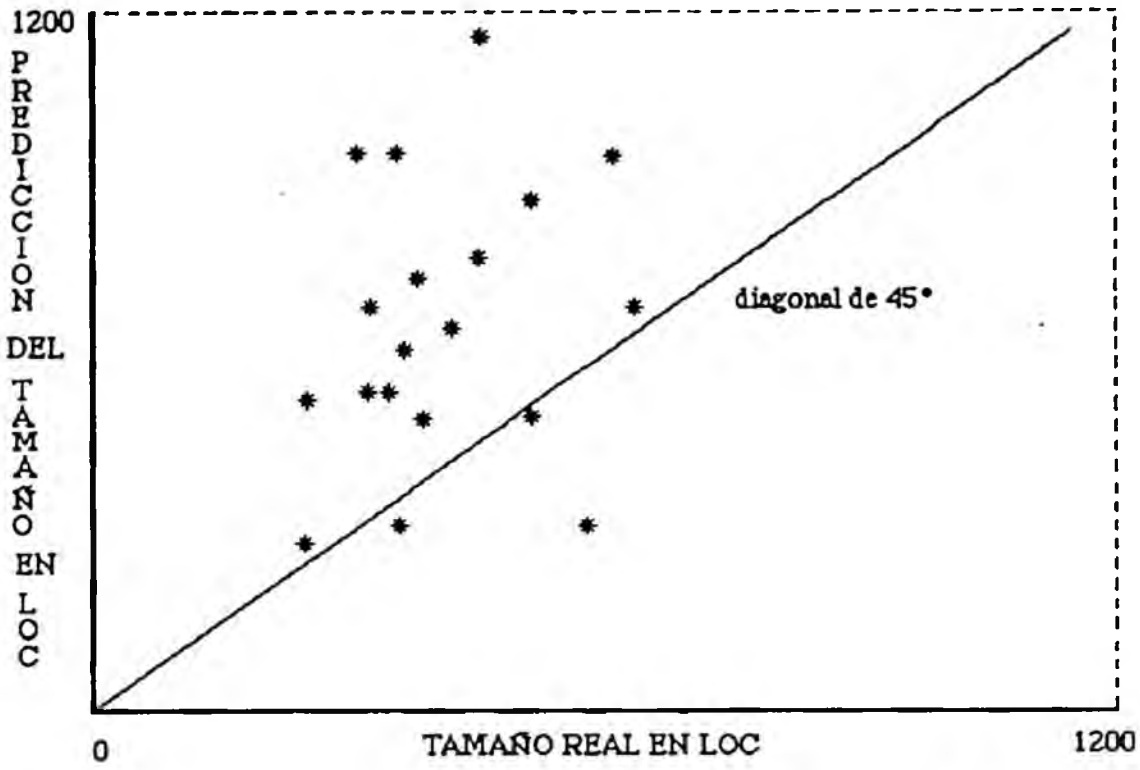
Perfil MRE. PRED(L) refleja la bondad del resultado de una estimación en un nivel específico de precisión, a veces queremos evaluar la bondad de una estimación a varios niveles de exactitud, para tener una descripción más completa de los resultados de la estimación. Con este propósito se diseñó el

perfil MRE [THEB83]. Este perfil considera a PRED(L), como el porcentaje de puntos para los cuales MRE ≤ L, para varios valores de L (por ejemplo, L = 0,10, 0,20, 0,30...0,90).

3.4.2 Criterio de Comparación

Adicionalmente, compararemos los resultados de un modelo con otro para ver cual de ellos produce estimaciones más precisas. Al comparar los resultados de dos modelos, no es suficiente comparar un único resumen numérico, tal como MRE' o PRED(0,25), ya que cada uno de ellos solamente caracteriza un aspecto del resultado de la estimación. Por lo tanto, necesitamos alguna medida que tenga en cuenta la naturaleza global de los datos.

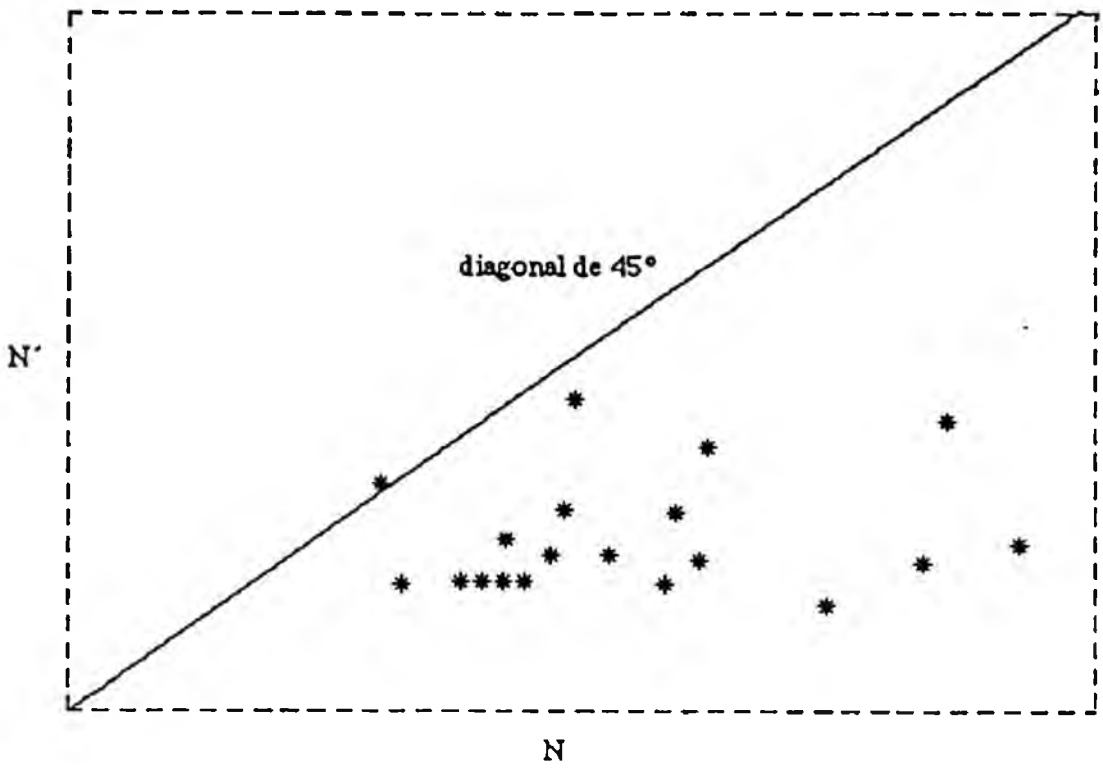
Comparación de Gráficos. La manera más directa para comparar el resultado de dos estimaciones es considerar sus gráficos. Gráficos drásticamente diferentes, pueden darnos una idea rápida de como los resultados de dos estimaciones difieren una de otra. Por ejemplo, los gráficos mostrados en la Figura 3.5 y en la Figura 3.6 parecen ciertamente diferentes. La mayor parte de las estimaciones, muestran en el primer gráfico que se han sobrepredicho los valores reales mientras que la subpredicción aparece la mayoría de las veces en la segunda gráfica. De cualquier modo, esto es solamente un análisis informal, y la interpretación a menudo suele ser demasiado subjetiva para ser útil.



RE' = - 0.70 MRE' = 0.77 PRED (0.25) = 19 %

Figura 3.5

Estimación Inicial Directa del Tamaño (EXPER-1)



RE' = 0.56 MRE' = 0.56 PRED (0.25) = 10 %

Figura 3.6

Estimación Inicial Indirecta del Tamaño (EXPER-1)

Comparación de Perfiles MRE. Una manera más formal de comparar los resultados de dos estimaciones es comparar sus perfiles MRE. Si el PRED(L) de una estimación es siempre (o en la mayoría de los casos) mayor que el de la otra estimación a cualquier nivel de L, es razonable concluir que esta estimación es superior. Por ejemplo, la Figura 3.8 contiene el perfil MRE para dos conjuntos de estimaciones. El eje vertical representa el porcentaje de estimaciones para las cuales MRE $\leq$ L. El eje horizontal representa el límite de error L. Consideremos, por ejemplo, los puntos encuadrados; mientras que las estimaciones de la "línea continua" están dentro del 20% de los valores reales más del 40% del tiempo de desarrollo, las estimaciones de la "línea discontinua" están dentro del 20% de los valores reales menos del 10% del tiempo. Esto debería dejar claro que cuanto más se acerque el perfil MRE a la esquina superior izquierda, mejor es el resultado de la estimación. Así en la Figura 3.8, observamos que las estimaciones de la "línea continua" son mucho mejores que las estimaciones de la "línea discontinua".

Comparación de Valores MRE: Test sobre MRE. Otra manera de comparar dos resultados de estimaciones es comparar sus valores MRE. Tal y como ya se ha argumentado, no podemos comparar solamente los valores medios de MRE en dos estimaciones. Por ejemplo, si el valor medio de MRE de un método es solamente ligeramente menor que el del segundo, no podemos concluir que el primer método es mejor que el segundo puesto que necesitamos considerar la distribución general de los valores MRE. Una manera formal de determinar si la diferencia entre los valores MRE

medios de dos estimaciones son significativos, estadísticamente hablando, es el uso del test que a continuación describimos y al que denominamos Prueba o Test T, propuesto por [NIE75].

Supongamos que Y_i representa el i-ésimo valor real de n números. Sean Y'_{i1} e Y'_{i2} dos estimaciones diferentes para el mismo valor real Y_i usando el método 1 y el método 2, respectivamente. Posteriormente, permitamos que U_{i1} sea el MRE de Y'_{i1}, y que U_{i2} sea el MRE de Y'_{i2}. Esto es,

$$U_{i1} = \frac{|Y_i - Y'_{i1}|}{Y_i} \qquad (3.6)$$

y,

$$U_{i2} = \frac{|Y_i - Y'_{i2}|}{Y_i} \qquad (3.7)$$

Usando esta prueba, podemos decidir si las medias de U_{i1} y U_{i2} (designados como U_1 y U_2 respectivamente) son realmente diferentes para un nivel de significación dado. En otras palabras, nuestra hipótesis nula será:

$$H_0 : U_1 = U_2 \qquad (3.8)$$

y la hipótesis alternativa será:

$$H_1 : U_1 \neq U_2 \qquad (3.9)$$

Por lo tanto, si el resultado de la prueba muestra que la hipótesis nula H_0 puede ser rechazada en un nivel específico de significación α, esto implica que U_1 es diferente significativamente de U_2 a ese nivel α. Si la hipótesis nula no puede ser rechazada al nivel α, entonces no podemos concluir que el primer método de estimación es diferente significativamente del segundo.

En la siguiente sección de este capítulo y en capítulos posteriores de esta tesis, la comparación entre los diferentes métodos de estimación se realizará basándonos en los perfiles MRE así como en la prueba de los valores medios de MRE.

3.4.3 Evaluación de la Estimación Inicial Directa del Tamaño

Tal y como se ha explicado en la Sección 3.3, se recopilaron las estimaciones de tamaño directas realizadas por los participantes en el experimento EXPER-1, en cada momento crucial que se producía durante el proceso de desarrollo. Las estimaciones de tamaño directo evaluadas fueron las recopiladas al final de la etapa de diseño.

Como ya hemos mencionado, la manera más directa de examinar los resultados de una estimación es mirar su representación gráfica. La gráfica para la estimación del tamaño directo se muestra en la Figura 3.5. El eje horizontal representa el tamaño real (en LOC), y el eje vertical representa el tamaño estimado.

La figura 3.5 muestra una gráfica en la que la mayor parte de los puntos se encuentran ubicados en la parte central izquierda del gráfico. Esto significa que el valor predicho típico era mayor que el valor real correspondiente. Adicionalmente, la dispersión vertical de todos los puntos es mayor que la dispersión horizontal, lo que implica que el rango de los valores predichos era más grande que el de los valores reales. Ya que la mayoría de los puntos de la figura están ubicados por encima de la diagonal de 45°, la sobrestimación apareció más amenudo que la subestimación y la cantidad de sobrestimación era grande. Esto se confirma por el alto valor negativo de RE' (-0,70).

Así, en la Figura 3.5, vemos que las estimaciones de tamaño directas realizadas por los participantes fueron normalmente sobrestimaciones con un porcentaje del 70% del tamaño real. Este resultado sorprendente fue debido parcialmente al hecho de que se controló muy poco el cuidado con que los participantes habían realizado sus estimaciones. Puesto que la exactitud de las estimaciones no era crítica para ellos y el proceso de estimación no estaba bien controlado, no es sorprendente descubrir que sus estimaciones fueron muy pobres.

3.4.4 Evaluación de la Estimación Inicial Indirecta del Tamaño

Tal y como se ha descrito anteriormente, la aproximación propuesta para una estimación inicial indirecta del tamaño requiere la estimación inicial de n_1 y n_2. El valor medio de n_1 para los 21 programas realizados para este experimento era de 64;

usamos esta constante para aproximar n_1 en nuestro estudio. Adicionalmente, tal y como ha sido propuesto en la Sección 3.2 bajo la estrategia de desarrollo "primera estructura de datos Top-dowm", la estimación inicial de n_2 se podía obtener multiplicando el valor inicial de n_2 al final de la fase de diseño por 1,25. Por lo tanto, estimabamos el tamaño del programa, N, usando la ecuación de longitud de Halstead. Se tomaba como valor de n_1 la constante 64, y se estimaba n_2 tal y como se ha descrito.

Representando gráficamente el tamaño estimado (N') contra el tamaño real (N) obtenemos la gráfica mostrada en la Figura 3.6. La mayoría de los puntos están localizados debajo de la diagonal de 45°, lo que significa que la mayoría de las estimaciones subpredijeron los valores reales por un margen bastante grande. Esto queda reflejado por el gran valor positivo de RE' (+0,58). Es decir, la aproximación propuesta para la estimación indirecta del tamaño no generó resultados satisfactorios para este conjunto de datos.

Análisis del Error. Debido a que nuestra aproximación de la estimación indirecta del tamaño implica tres fases, se realizó un análisis para ver que fase ocasionaba los mayores errores. En la Tabla 3.2 resumimos los resultados de este análisis.

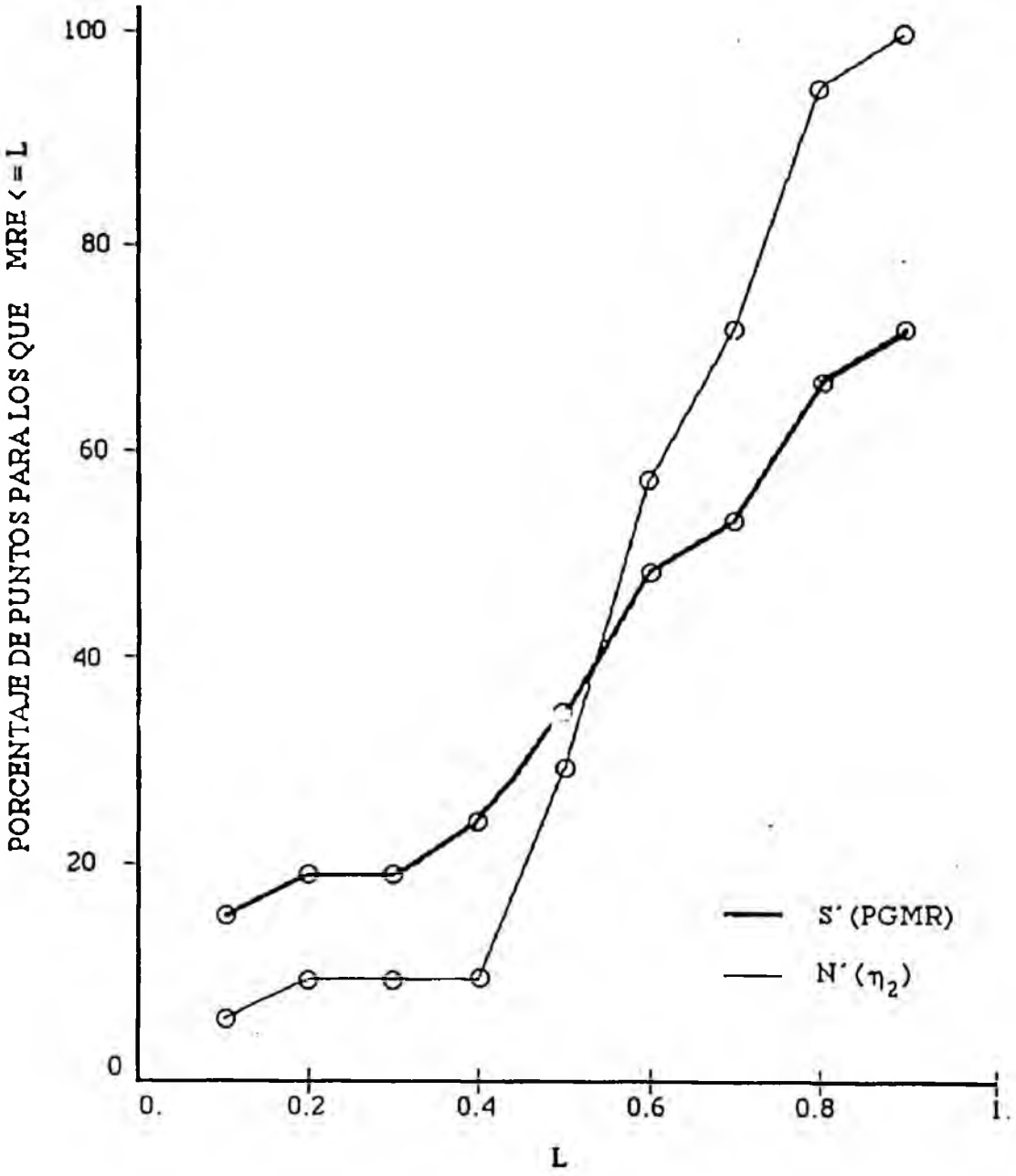


Figura 3.7

Estimación del Tamaño Directa frente a la Estimación Indirecta

(EXPER-1)

Evaluación de la Estimación Inicial de n_1. La primera fila de la Tabla 3.2 muestra los resultados de la estimación inicial de n_1 usando el valor constante 64 (indicada como $n'_1(64)$). Ya que el valor de MRE' es pequeño (0,08) y el porcentaje de PRED(0,25) es alto (95%), podemos decir que esta estimación es precisa.

Tabla 3.2

Análisis del Error de la Estimación Indirecta del Tamaño

(EXPER-1)

Estimación	MRE'	PRED(0,25)	RE'
$n'_1(64)$	0,08	95%	-0,01
n'_2	0,48	19%	+0,48
$N'(n_1, n_2)$	0,33	48%	+0,32

Evaluación de la Estimación Inicial de n_2. El resultado de la estimación inicial de n_2 usando su valor medido inicialmente multiplicado por 1,25 se resume en la segunda fila de la tabla. La notación n'_2 representa el valor estimado de n_2. Con un valor de MRE de 0,48 y un porcentaje de PRED(0,25) del 19%, el valor estimado de n_2 no se puede considerar un buen estimador para el valor real n_2.

La aproximación propuesta para la estimación inicial de n_2 se basa en la hipótesis de que bajo la estrategia "primera estructura de datos top-down", debemos obtener un alto porcentaje (80%) de n_2 al final de la etapa de diseño. Para validar esta hipótesis, se compararon los valores de n_2 obtenidos al final de la etapa de diseño en este experimento, con los valores finales de n_2. Se descubrió que los valores iniciales de n_2 eran solamente un 42% del valor final, en lugar del esperado 80%. En otras palabras, menos de la mitad del valor final de n_2 estaba a nuestra disposición al final de la etapa de diseño para cada programa realizado en este experimento.

Intentamos descubrir por qué la hipótesis del desarrollo inicial de n_2 no era válida. Comenzamos analizando los componentes de n_2 que son principalmente: variables, constantes y series de caracteres. Vamos a referirnos al primer componente "variables" como VAR (contador de variable única). Para un programa dado, VAR se calcula tomando el valor de n_2 y restándole el número de constantes numéricas y series de caracteres que aparecen en dicho programa. Suponemos que utilizando la estrategia de desarrollo "primera estructura de datos Top-Down", la mayoría de los componentes de VAR se crearán durante la etapa de diseño y la porción que queda de n_2 (constantes y series de caracteres) se añadidará cuando se desarolle el código real. Pero, los datos reales no soportan totalmente este supuesto. Al comparar los valores iniciales y finales de VAR para cada programa, descubrimos que el valor inicial erá alrededor del 52% del valor final. Por lo tanto, tal y como habíamos supuesto,

durante la fase de diseño se creaba un alto porcentaje de VAR (52% para VAR, contra un 42% para n_2). De cualquier modo este porcentaje era todavía mucho más bajo de lo esperado.

Un examen posterior de los ratios entre los valores iniciales y finales de VAR, revelan que estos ratios varían considerablemente. Cinco de los 21 ratios estaban por encima del 0,70, y 3 por debajo del 0,20. Además, las versiones iniciales de estos 3 programas no pueden considerarse realmente versiones terminadas de diseñar. Las instrucciones declarativas contenidas en estas versiones iniciales eran solamente un pequeño subconjunto de las intrucciones declarativas finales. Por lo tanto, especulamos que los individuos que realizaron estos tres programas, no dedicaron el tiempo suficiente a la etapa de diseño. Esta especulación nos lleva de nuevo a la cuestión de la falta de control sobre los participantes en este experimento.

Es necesario señalar que al principio de este experimento, se explicó a los participantes las reglas y procedimientos experimentales, y parecía que las habían comprendido bien. Ellos trabajaron en sus programas a su propio ritmo, tomando nota del tiempo total de programación usando sus propios métodos, y estimando el tamaño del programa de acuerdo con sus propias estrategias de estimación. No tenian que informar al administrador del experimento acerca de los problemas encontrados. No se realizó ningún intento para "controlar" esas actividades. Esto es, no hubo reglas experimentales que

aseguraran que los participantes seguían la estrategia de desarrollo requerida, dedicando el tiempo suficiente al diseño, y realizando estimaciones cuidadosas.

La forma ideal de controlar estas actividades sería reunir a todos los participantes en una habitación durante varios días y dirigir y observar su comportamiento de programación cuidadosamente. Puesto que el promedio del tiempo total del proceso de desarrollo involucrado en este experimento fue aproximadamente de 30 horas, nuestros limitados recursos, obviamente, no nos permitían hacer semejante cosa.

Concluimos que, aunque no habíamos tenido la situación ideal para controlar este tipo de experimentos, se podían hacer algunas mejoras. El segundo experimento, EXPER-2, fue diseñado con este fín (además del propósito de confirmar algunos de los hallazgos descubiertos en EXPER-1).

Evaluación de la Ecuación de Longitud de Halstead. La última fila de la Tabla 3.2 resume los resultados de la estimación de la ecuación de longitud de Halstead para el conjunto de programas de este experimento. La notación $N'(n_1, n_2)$, representa el tamaño estimado de N obtenido sustituyendo los valores finales de n_1 y n_2 en la ecuación de longitud de Halstead. Se observa que el tamaño estimado de N, incluso usando los valores reales de n_1 y n_2, no puede considerarse un estimador exacto del tamaño real, ya que el valor MRE' no era demasiado pequeño (0,33) y el porcentaje PRED(0,25) no era demasiado alto (48%).

Los resultados parecen confirmar los hallazgos de un estudio de Johnston y Lister [JOHN81] en donde N' es una subestimación de N con un promedio del 37% para 9 programas en Pascal. Debería tenerse en cuenta que las evidencias empíricas que apoyan la ecuación de longitud de Halstead han sido centradas en Fortran [HALS77], PL/1 [ELSH75] , y Cobol [SHEN81].

3.4.5 Enfrentamiento de las Estimaciones Iniciales del Tamaño

Basándonos en los resultados presentados en las dos secciones precedentes, está claro que ni las estimaciones del tamaño directas ni las indirectas pueden considerarse aceptables de acuerdo con el criterio de evaluación establecido inicialmente. Sin embargo, para tener una idea de la diferencia entre ambos métodos, los resultados de la estimación usando estos dos métodos se compararon utilizando el criterio descrito anteriormente. La Tabla 3.3 resume los resultados.

La primera fila de la Tabla 3.3 muestra el resultado de la estimación indirecta del tamaño. La notación $N'(64,n'_2)$ representa el tamaño estimado de N basada en la ecuación de longitud de Halstead con la constante 64 en lugar de n_1 y con el valor estimado de n_2 en lugar de n_2. Por otra parte, el resultado de la estimación directa del tamaño se resume en la segunda fila de la tabla con la notación S'(PGMR) representando el tamaño estimado en LOC realizado por los programadores. Las entradas de

la segunda, tercera y cuarta columnas de la Tabla 3.3 (bajo las etiquetas "MRE'", "PRED(0,25)", y "RE'"), resumen el resultado de la estimación en términos del criterio de evaluación descrito en la Sección 3.4.1. Las entradas en la última columna de la tabla, dan los rangos de los resultados para los dos métodos de estimación de acuerdo con los resultados de test respecto a los valores MRE.

Tabla 3.3

Estimación Inicial del Tamaño:

Directa contra Indirecta

(EXPER-1)

Estimación	MRE'	PRED(0,25)	RE'	rango
N'(G4,n') 2	0,56	10%	+0,56	igual
S'(PGMR)	0,77	19%	-0,70	igual

Como ya se mencionó en la sección 3.4.2, consideramos que la diferencia entre dos métodos de estimación es significativa estadísticamente, a un cierto nivel de significación si los resultados de la prueba T indican que, a ese nivel, el valor medio de MRE del primer método difiere de manera significativa del valor del segundo. El valor de este nivel de significación debe escogerse basándonos en los errores de Tipo I (rechazar H_0 cuando es cierta) en oposición al error de Tipo II (aceptar H_0 cuando es falsa) [NIE75]. No hay un criterio absoluto para

determinar el nivel apropiado de significación, que una prueba debe tener. El valor de corte, 0,10 se situó relativamente alto. Entonces, basándonos en los resultados de la prueba T, vimos que la estimación del tamaño indirecta no era mejor que la estimación del tamaño directa para $\alpha < 0,10$, y es por ésto que los rangos alcanzados son indicados como "igual". En el resto de esta tesis, si dos estimaciones alcanzan rangos diferentes, esto implica que sus resultados son diferentes unos de otros, para $\alpha < 0,10$, de acuerdo con los resultados de la prueba T.

Tal y como hemos señalado en la Sección 3.4.2, una manera simple y directa de comparar los resultados de dos o más estimaciones es comparar sus perfiles MRE. Los perfiles MRE de las dos estimaciones del tamaño se muestran en la Figura 3.7. Estos perfiles MRE parecen apoyar lo que han mostrado los resultados de la prueba T, es decir, la estimación de tamaño indirecta no es significativamente diferente de la estimación de tamaño directa. Como se puede ver en la Figura 3.7, los resultados de la estimación de tamaño indirecta parecen ser peores que los resultados de la estimación del tamaño directa, en términos del porcentaje de PRED(L) para $L \leq 0,5$. Pero la situación cambia para $L \geq 0,6$. Por lo tanto, estos resultados sugieren que ambos métodos, la estimación inicial del tamaño directa, y la estimación inicial del tamaño indirecta (usando la ecuación de longitud de Halstead), obtenidos al final de la etapa de diseño en el experimento EXPER-1, fueron muy pobres.

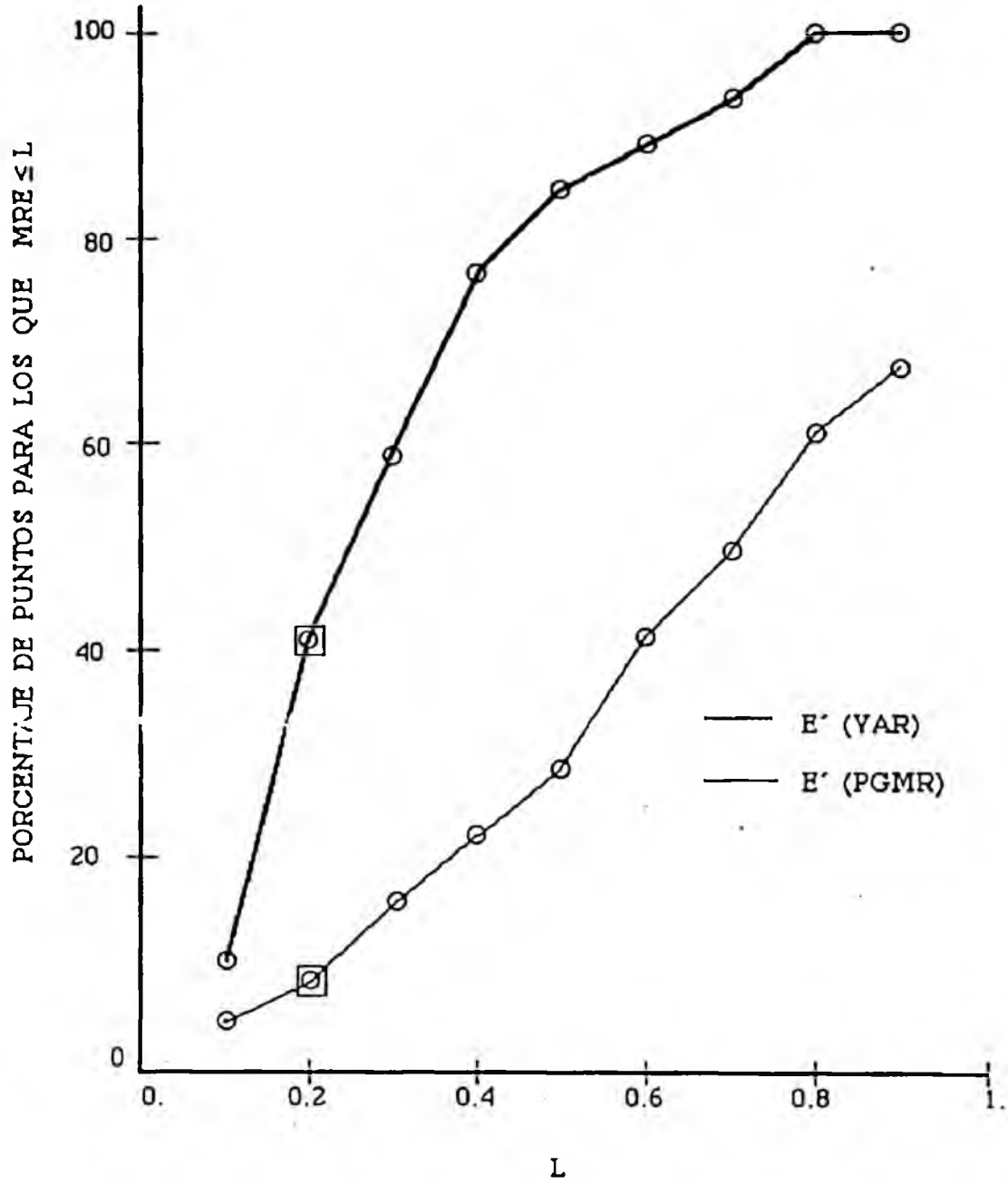


Figura 3.8

Perfil MRE de dos Estimaciones

3.5 ESTUDIO PRELIMINAR

Hemos estudiado una aproximación a la estimación del tamaño indirecta basada en la ecuación de longitud de Halstead. La posibilidad de realizar esta aproximación con éxito depende de tres factores:

1.- La estimación inicial exacta de los operadores únicos (n_1).

2.- La estimación inicial exacta de los operandos únicos (n_2).

3.- La validez de la ecuación de longitud.

Para evaluar la posibilidad de efectuar esta aproximación se realizó un experimento de construcción de programas denominado EXPER-1. Las siguientes notas resumen nuestros resultados.

1.- Este estudio demuestra que se puede aproximar la estimación inicial del valor de n_1 usando una constante apropiada.

2.- Este estudio no justifica la hipótesis que indica que n_2 se puede estimar bien en una etapa inicial. Atribuimos este hecho a la falta de control sobre las condiciones en las que los participantes han seguido las directrices del experimento. Sin embargo, hemos identificado a un

subconjunto de n_2 llamado VAR, y se ha sugerido la posibilidad de efectuar una estimación inicial de VAR.

3.- No se ha justificado la validez de la ecuación de longitud de Halstead con los programas en Pascal realizados en este experimento. Ha aparecido una subestimación significativa en más ocasiones de las esperadas.

Resumiendo, en este estudio preliminar no se ha demostrado el éxito de la aproximación propuesta a la estimación del tamaño indirecta. En el siguiente estudio empírico, EXPER-2, se consideran otros modelos de estimación a parte de la ecuación de longitud de Hasltead y se imponen controles experimentales más rigurosos.

CAPITULO 4

LA EVOLUCION DE LA METRICA

En este capítulo, comenzamos discutiendo las ideas básicas de nuestro estudio de la evolución de la métrica. A continuación, presentaremos un método de estimación del esfuerzo y del tamaño basados en la evolución de la métrica.

4.1 MOTIVACION DEL ESTUDIO

Nuestro estudio de la evolución de la métrica se ha visto motivado por las ideas ilustradas en la Figura 4.1.

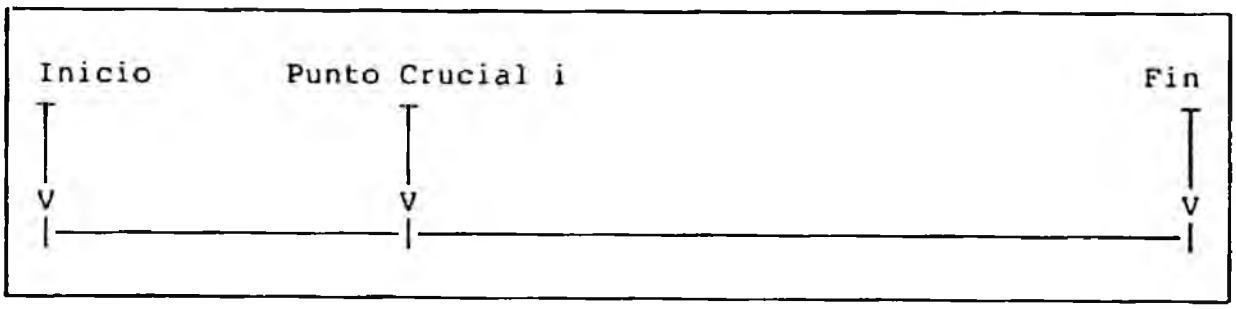


Figura 4.1

Motivación para el Estudio de la Evolución de la Métrica

Tal y como se describe en la figura 4.1, queríamos averiguar si podíamos reunir información tangible en el punto crucial i, que se pueda utilizar para hacer algunas inferencias acerca del futuro proceso de desarrollo. Algunos elementos de interés son: el esfuerzo de desarrollo, el tamaño del proyecto y la calidad del producto. Algunos ejemplos son:

1.- ¿Podemos localizar áreas problemáticas en un diseño, o predecir la calidad del producto final mediante un cuidadoso repaso del trabajo de diseño?

2.- ¿Es posible predecir el tamaño final del programa a partir de la información disponible en una etapa inicial?

3.- ¿Es posible predecir el esfuerzo que queda por efectuar para realizar el proyecto a partir de la información disponible en una etapa inicial?

Los trabajos de Troy [TROY81] abordaron la primera cuestión. Las métricas de diseño consideradas en su estudio tenían en cuenta, aproximadamente, entre el 50% y 60% del número de modificaciones de programa realizadas durante la integración y prueba del sistema. Esto sugiere que las mediciones realizadas en una etapa del trabajo de diseño inicial pueden ser útiles para identificar deficiencias de diseño antes de la fase de codificación, lo que reduciría posiblemente el costo del desarrollo del software.

Nuestra investigación intenta ayudar a encontrar la respuesta a las dos últimas preguntas. En primer lugar, consideramos qué clase de información tangible esta disponible en una etapa inicial y cómo se la puede reunir de una manera objetiva y algorítmica. Se podría sugerir que si se aplica al diseño del programa la técnica de "Diseño Estructurado" [YOUR79] los documentos de diseño, como por ejemplo los "Diagramas Estructurados", estarán disponibles al final del diseño. Por otra parte, si se utiliza en el diseño del programa la metodología de diseño orientada a las estructuras de datos, propuesta por Warnier y Orr [WARN74], los documentos de diseño denominadis "Diagramas Warnier-Orr" estarán disponibles al final del diseño. Además, si se utiliza un lenguaje de diseño de programas [SAMM82] las versiones iniciales del LDP estarán disponibles al final del diseño. Pero, las herramientas automáticas para estas tres metodologías no estan disponibles en nuestro entorno de investigación. También es posible utilizar una herramienta C.A.S.E., que permite automatizar muchos aspectos del proceso de desarrollo del software [MYNA90]. Sin embargo, nuestra aproximación para reunir información tangible se centra en la evolución de las métricas del software.

Como se ha mencionado en el Capítulo 2, la "evolución de la métrica del software" trata sobre la variación de los valores de la métrica con el tiempo de desarrollo. Muchas métricas del software están determinadas por la versión final del programa fuente. Para que una métrica sea útil en una estimación del esfuerzo inicial, necesita estar disponible pronto, mucho antes

del final del proceso de desarrrollo. Por lo tanto, para reunir la información de la evolución de los datos de una métrica, es necesario disponer de instrumentación apropiada para medir los valores sucesivos de la métrica a lo largo del proceso de desarrollo.

4.2 FACTORES QUE AFECTAN A LA EVOLUCION DE LA METRICA

La Figura 2.3 muestra un ejemplo de la evolución del tamaño en LOC, otro ejemplo se muestra en la Figura 4.2. La curva representada en la Figura 4.2, es creciente con derivada decreciente, para abreviar, en este trabajo denominamos a las curvas que evolucionan de esa manera, curvas convexas. Por lo tanto, podemos llamar al patrón de la evolución, tal como se muestra en la Figura 4.2, patrón convexo. Como se puede ver en estos dos ejemplos, el comportamiento de la evolución para la misma métrica puede diferir de manera drástica. Esto nos lleva a considerar que factores afectan significativamente la evolución de la métrica.

La evolución de la métrica no es un proceso aleatorio. Puesto que cada métrica representa una característica de programa, el comportamiento de la evolución de una métrica dependerá, naturalmente, de como varía la característica correspondiente durante el proceso de desarrollo. Suponemos que el modo en que varía una característica de programa depende al menos de dos factores:

- Factor 1: El Programador

Es probable que los estilos de programación individuales provoquen las diferencias más significativas con respecto a como se desarrolla un rasgo del programa. Así, el factor programador es uno de los factores importantes a considerar.

- Factor 2: El Programa

Una característica concreta para un programa determinado puede ser más fácil de desarrollar en una etapa inicial que para otro programa.

Por estas razones, el comportamiento de la evolución de una métrica dada puede variar considerablemente para diferentes programas escritos por personas diferentes.

De todos modos, si podemos controlar, hasta cierto punto, la manera en que se desarrolla un rasgo de programa, reduciendo los factores de programador y programa tanto como sea posible, surgirá un comportamiento de la evolución más consistente para la métrica correspondiente. Denominamos a las reglas y procedimientos seguidos por los programadores en la construcción del software, "estrategias de desarrollo". Por ejemplo, si la estrategia de desarrollo requiere que la mayor parte de las estructuras de datos se desarrollen tan pronto como sea posible, entonces, podemos esperar que los valores de la métrica de las

estructuras de datos se incrementarán pronto muy rapidamente en el proceso de desarrollo. Por otra parte, si la estrategia de desarrollo requiere que un rasgo de programa se desarrolle tarde en el proceso, la métrica correspondiente probablemente no se aproximará a su valor final hasta una etapa posterior.

Basándonos en los argumentos presentados, establecemos la hipótesis de que si una estrategia de desarrollo está debidamente reforzada con respecto a un rasgo de programa, se observará un patrón de evolución consistente para la métrica correspondiente. Es decir, se reducirá la variación en el comportamiento de la evolución de la métrica correspondiente.

4.3 PATRONES DE EVOLUCION

De acuerdo con nuestra suposición en lo que respecta a la evolución de la métrica, una métrica puede tener cualquier patrón de evolución deseado siempre y cuando sea debidamente reforzada la estrategia de desarrollo requerida. Esto implica que si la estrategia de desarrollo requiere que un rasgo de programa dado (por ejemplo, estructuras de datos, estructuras de control, etc) se desarrolle tan pronto como sea posible, la métrica para ese rasgo debe tener una evolución convexa. En la práctica, sin embargo, no toda estrategia es intuitivamente atractiva. Por ejemplo, una estrategia que requiera que el código del programa se desarrolle tan pronto como sea posible, es menos atractiva que una estrategia que requiera el desarrollo inicial de las

estructuras de datos o de las estructuras de control. Por lo tanto, no todas las estrategias de desarrollo son realizables en la práctica.

Además, no todo patrón de evolución es atractivo en lo que se refiere a la estimación del tamaño y del esfuerzo en este estudio. Consideramos que un patrón de evolución es atractivo si su comportamiento futuro puede ser facilmente determinado en una etapa inicial. Veamos dos patrones de evolución que pueden ser muy útiles para las estimaciones iniciales del esfuerzo y tamaño del programa.

4.3.1 Evolución Convexa

Si, en una etapa inicial, el valor de una métrica crece y alcanza rapidamente la mayor parte de su valor final, la clasificamos como una evolución convexa. Es decir, consideramos que una evolución es convexa si en un punto inicial, la mayor parte de los valores finales de la métrica se han producido. Por ejemplo, tal y como se muestra en la Figura 4.3, el valor de esta métrica aumenta con el paso del tiempo tan rápidamente que alcanza el 80% del valor final de la métrica en el punto del 30% del tiempo de desarrollo.

Una curva de ecuación:

$$y(t) = t^{\beta} \qquad (4.1)$$

es definida como una curva convexa de orden β, para β ≤ 0,5. La curva convexa de orden 0,1 es la curva más alta de la Figura 4.4. De acuerdo con esta curva, el valor de la métrica en el punto del 20% del tiempo de desarrollo es igual, más o menos, al 85% del valor final de la métrica. La curva convexa de orden 0,5 es la curva más baja de la Figura 4.4. De acuerdo con esta curva, el valor de la métrica en el punto del 20% del tiempo de desarrollo es igual, más o menos, al 45% del valor final de la métrica. Las curvas convexas de ordenes 0,2, 0,3, y 0,4, también se muestran en la Figura 4.4. Por lo tanto, una evolución se define como una evolución convexa de orden β si sigue aproximadamente una curva convexa de orden β.

Supongamos que la evolución de una métrica es una evolución convexa de orden igual o menor a 0,2. En una etapa inicial cuando se producen la mayoría de los valores de la métrica, debemos ser capaces de determinar el comportamiento futuro de la evolución de esta métrica de manera razonable, siendo esta una de las razones por las que consideramos a la evolución convexa tan atractiva.

4.3.2 Evolución Lineal

Una evolución métrica es lineal si sigue aproximadamente una línea recta que pasa a través del origen del gráfico de evolución. La razón para considerar atractiva la evolución lineal, es que se puede determinar el comportamiento futuro de la evolución tan pronto como se pueda determinar el ratio de crecimiento. Esto se ilustra en la Figura 4.5. En esta figura, la

evolución del tamaño de un programa en LOC se representa mediante dos "+", y cinco "*". Esta figura muestra que el tamaño de este programa en LOC evoluciona aproximadamente de modo lineal con el tiempo de desarrollo.

4.4 ESTRATEGIAS Y METRICAS PARA LOS PATRONES PRESENTADOS

Vamos a considerar las métricas y estrategias que pueden conducirnos a los dos patrones de evolución presentados.

4.4.1 Métrica y Estrategia para la Evolución Convexa

Como se ha mencionado anteriormente, en principio, cualquier estrategia de desarrollo que requiere un desarrollo inicial de un rasgo de programa dado, nos llevará a una evolución convexa para la métrica correspondiente. Ya que nuestros analizadores de programas, facilitan la medición inicial de las métricas de estructuras de datos, la estrategia de desarrollo que investigamos en este estudio es la estrategia "primera estructura de datos top-down" descrita en el Capítulo 3. Utilizando esta estrategia, se deben definir un alto porcentaje de estructuras de datos y variables durante el diseño del programa. Después de la etapa de diseño el porcentaje de nuevas estructuras de datos y variables adicionadas al programa, decrecerá rapidamente. Por lo tanto, suponemos que bajo esta estrategia una métrica que cuente las estructuras de datos y variables debe tener una evolución convexa.

4.4.2 Métrica y Estrategia para la Evolución Lineal

Otra estrategia de desarrollo considerada en este estudio es la estrategia de desarrollo "incremental" [YOUR79]. Usando esta estrategia, el proceso de desarrollo del programa procederá de la siguiente manera:

1. Diseñar, codificar y probar un módulo por si mismo.
2. Añadir otro módulo.
3. Probar y depurar la combinación tanto como sea posible.
4. Repetir los pasos 2 y 3 hasta completar el programa.

Esta estrategia requiere que en un momento determinado se añada un módulo al programa. Cuando un nuevo módulo se ha codificado y añadido al programa, se prueba y depura el programa completo. Ya que el proceso de depuración es más sistemático y organizado, suponemos que el tiempo de codificación y depuración, cuando se incluye un nuevo módulo, será probablemente proporcional al tamaño del módulo. En otras palabras, usando la estrategia incremental, el tamaño del programa crecerá un poco en cada período de tiempo. Por lo tanto bajo esta estrategia, el tamaño del programa crecerá linealmente con el tiempo de desarrollo.

Figura 4.2

Evolución del Tamaño

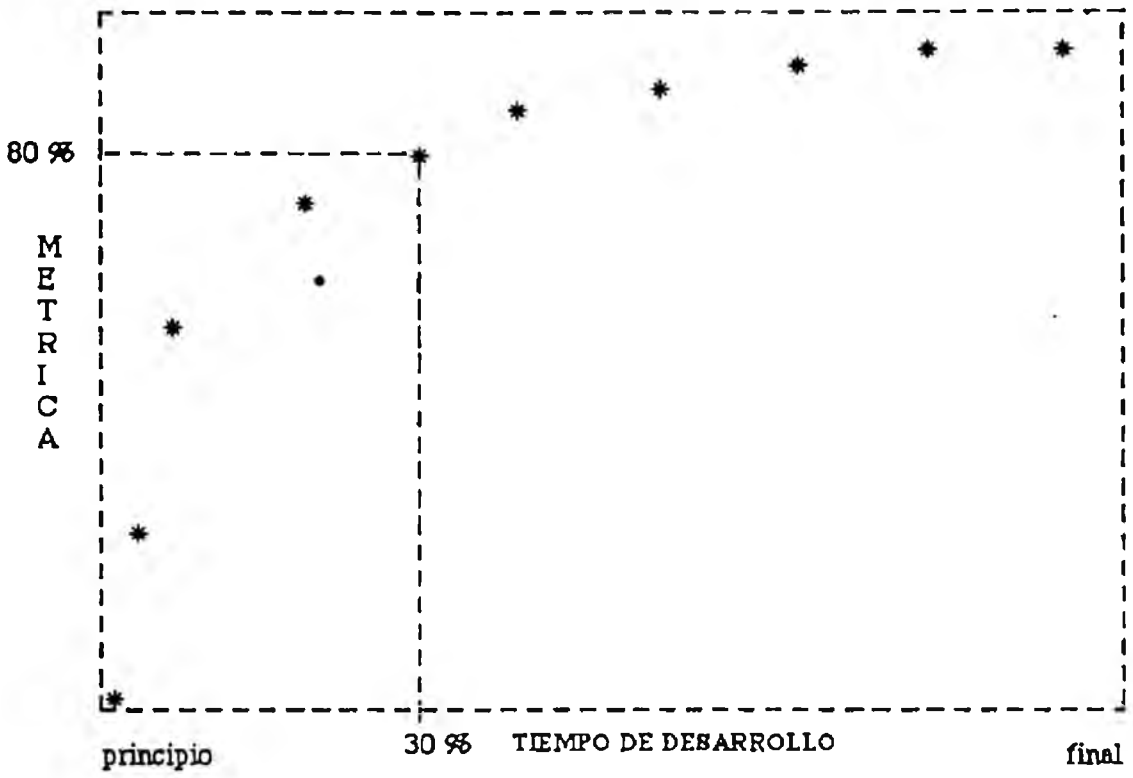


Figura 4.3

Evolución Convexa

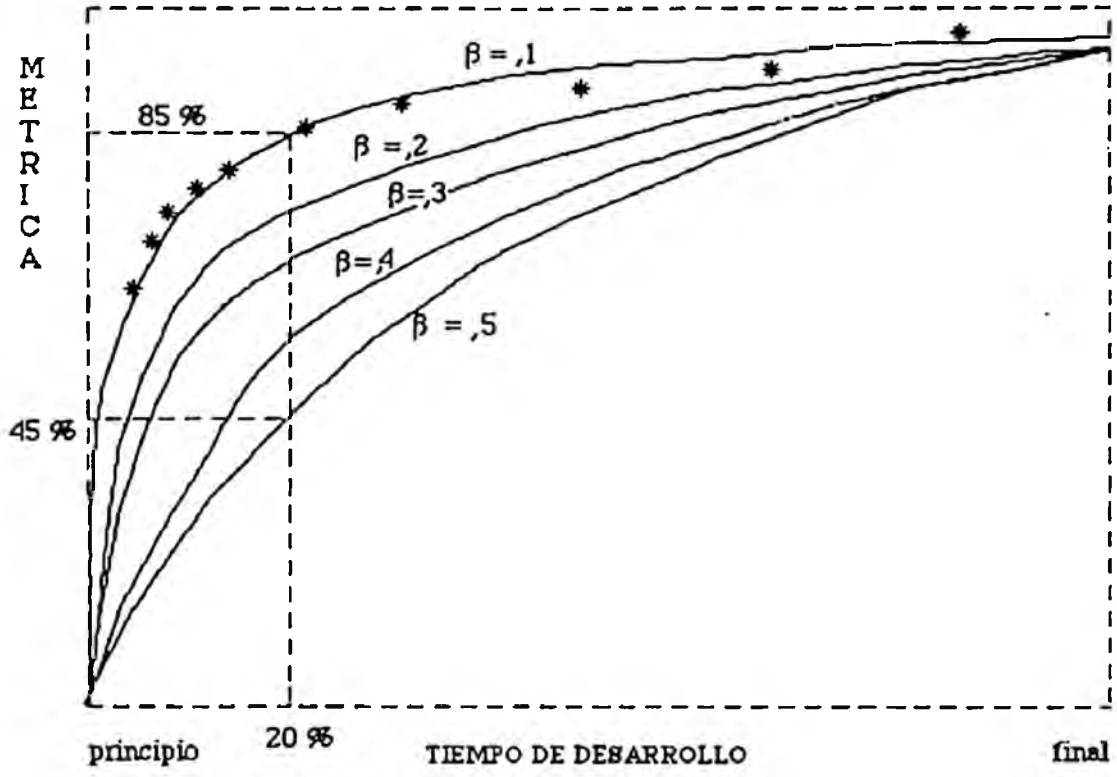


Figura 4.4

Evolución Convexa: Rango 0,1-0,5

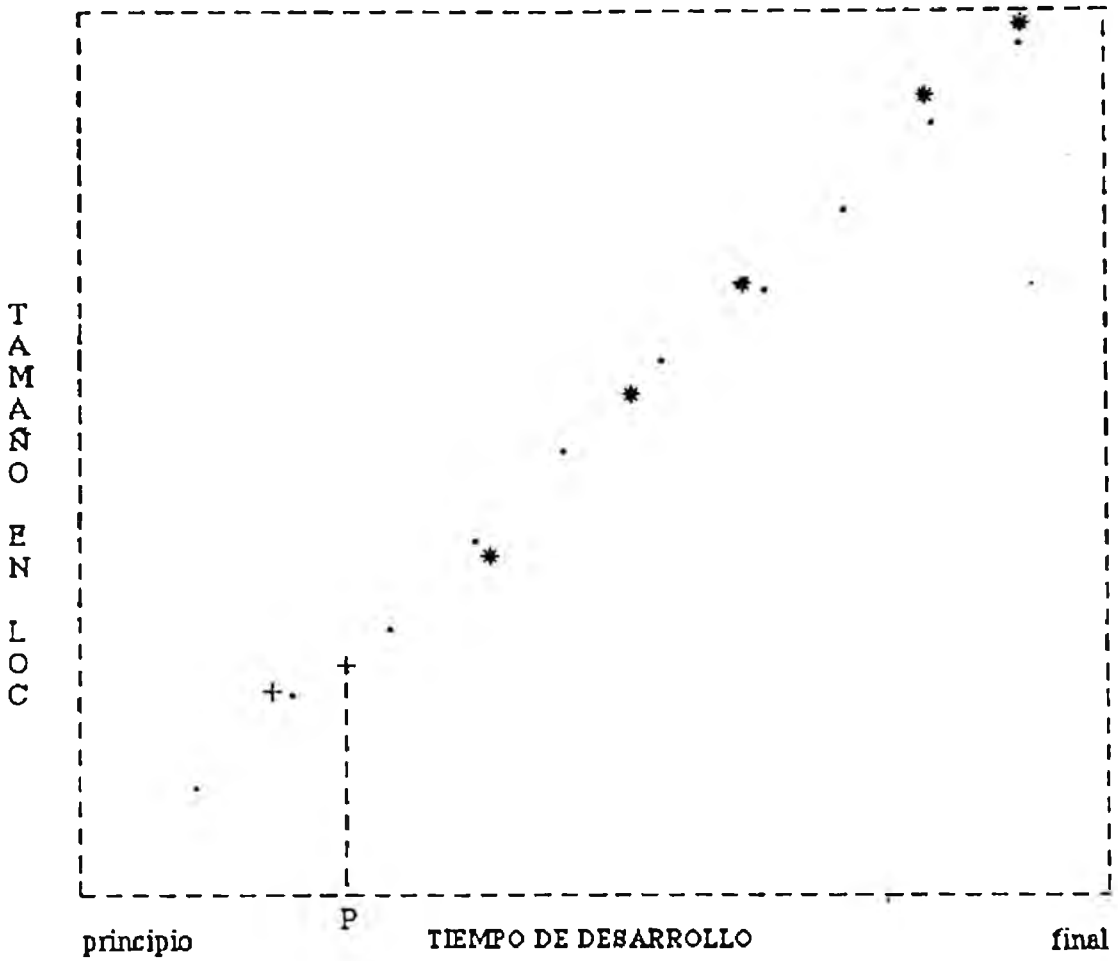


Figura 4.5

Predicción Inicial del Comportamiento de la Evolución Lineal

4.5 MODELO DE ESTIMACION BASADO EN LA EVOLUCION DE LA METRICA

Es sabido que la evolución de la métrica tiene que ver con los valores de la métrica en varios puntos del tiempo de desarrollo, por lo tanto podemos sacar alguna consecuencia acerca del tiempo total de programación basándonos en el comportamiento de evolución de la métrica. Si la métrica en cuestión está relacionada muy de cerca con el tamaño del programa, podemos también predecir el tamaño final del programa basándonos en el comportamiento de la evolución de esta métrica. Esta idea se ilustra en la Figura 4.6.

Supongamos que hay dos métricas del tamaño que miden esencialmente el mismo parámetro (por ejemplo, el tamaño) en la versión final del programa. Asumimos que los valores de estas dos métricas evolucionarán de modo bastante diferente durante el proceso de desarrollo. Supongamos que bajo una cierta estrategia de desarrollo una métrica sigue una evolución convexa y la otra sigue una evolución lineal. Puesto que estas dos métricas tendrán el mismo valor al final del proceso de desarrollo, de acuerdo con nuestra premisa, la intersección de estas dos curvas (el punto Q en la Figura 4.6) nos dará una buena estimación del tamaño final del programa, así como del tiempo de desarrollo final.

Partiendo de la discusión precedente, parece que hay dos premisas que estan subyacentes bajo la aproximación de la estimación:

1.- Existe una métrica relacionada con el tamaño que puede convertirse en tamaño con una precisión razonable.

2.- Esta métrica relacionada con el tamaño y la métrica del tamaño real evolucionan de modo bastante diferente durante el proceso de desarrollo. Una evoluciona de manera convexa y la otra evoluciona de modo lineal.

En el Capítulo 3, se han introducido dos métricas:

- n_2, el contador de operando único
- VAR, el contador de variable única

En este estudio, las consideramos métricas relacionadas con el tamaño, puesto que se mostrará en el siguiente capítulo, que ambas métricas pueden convertirse en la métrica del tamaño S con una precisión razonable.

Ya que que estas dos métricas, n_2 y VAR, estan relacionadas con las estructuras de datos de un programa, deberán exhibir una evolución convexa bajo la estrategia de desarrollo "primera estructura de datos top-down". También hay que tener en cuenta que en la Sección 4.3.2, supusimos que bajo una estrategia de desarrollo "incremental", el tamaño del programa evolucionaría linealmente con el tiempo de desarrollo. Por lo tanto, la aproximación a la estimación del esfuerzo y del tamaño entendida

en este estudio, está basada en la evolución convexa de las métricas relacionadas del tamaño n_2 y VAR, y la evolución lineal de las métricas del tamaño S y N.

4.6 GENERACION DE MODELOS DE LA EVOLUCION METRICA

Cuando observamos que existe un patrón consistente para una métrica, es necesario realizar un proceso cuantitativo para poder aplicar este conocimiento más sistemáticamente. Es decir, necesitamos un modelo matemático. Esperamos que este modelo matemático:

1.- Exhiba los aspectos cualitativos de la evolución de una métrica

2.- Sea tan simple como se pueda, ya que un modelo simple es más fácil de construir, entender, y usar.

Debe señalarse que un modelo matemático típico solamente se puede usar para describir el comportamiento "general" de un fenómeno observado. No tiene en cuenta necesariamente las variaciones individuales. Por tanto, debe esperarse un margen de error razonable cuando el comportamiento real se compara con la tendencia fijada. En las secciones siguientes, proponemos el uso de unas funciones del tiempo de desarrollo relativamente simples para establecer un modelo matemático para los dos patrones de evolución de la sección previa: convexa y lineal.

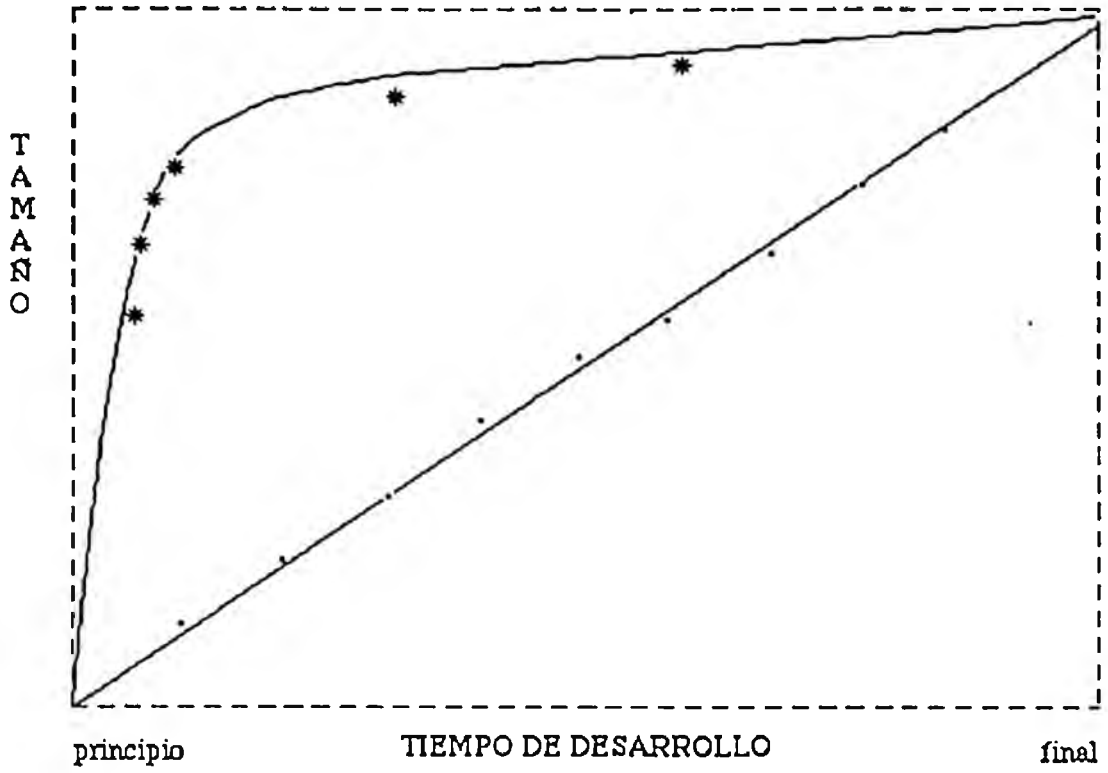


Figura 4.6

Un Modelo de Estimación Basado en la Evolución de la Métrica

4.6.1 Obtención del Modelo de la Evolución Convexa

La función que consideramos para realizar el modelo de la evolución convexa es un modelo que depende de dos parámetros

$$y(t) = \alpha \times t^{\beta} \qquad (4.2)$$

El parámetro α en la ecuación (4.2) se denomina parámetro de escala. El parametro β se denomina parámetro de forma, debido a que la forma de la curva esta completamente determinada por su valor. En la Figura 4.4, se han mostrado varias curvas convexas definidas por la ecuación (4.2) para β = 0,1, 0,2, 0,3, 0,4, y 0,5. Elegimos este modelo para la evolución convexa por dos motivos, en primer lugar por que representa una curva convexa para $0 < \beta \leq 0,5$, y en segundo lugar por la sencillez de su uso.

Téngase en cuenta que la función indicada representa una línea recta cuando $\beta = 1$ y una curva creciente con derivada creciente, a la que denominaremos cóncava, cuando $\beta > 1$. Por lo tanto, esta clase de función es una forma general para modelos de evolución convexas, lineales, y cóncavos.

4.6.2 Obtención del Modelo de la Evolución Lineal

Para generar un patrón de evolución lineal, el camino más sencillo es usar una función lineal que pase a través del origen

$$y(t) = \alpha \times t \qquad (4.3)$$

donde α es el parámetro de escala definido anteriormente. Un modelo lineal más general es:

$$y(t) = b + \alpha \times t, \text{ donde } \alpha > 0 \text{ y } b\ 0$$

no considerándose la posibilidad de t igual a cero, ya que en ese caso el valor de la métrica será 0. Téngase en cuenta, que la ecuación (4.3) es un caso especial de la ecuación (4.2), es decir, para $\beta = 1$.

4.7 PREDICCION INICIAL DEL COMPORTAMIENTO DE LA EVOLUCION DE LA METRICA

Vamos a considerar la manera en la que se pueden aplicar los modelos de evolución (4.2) y (4.3) al desarrollo de programas reales . Empezaremos con el modelo de un parámetro descrito en la ecuación (4.3).

4.7.1 Ajuste de Curva para el Modelo de un Parámetro

Ya que solamente se necesita determinar un parámetro en el modelo de ecuación (4.3), se puede dibujar una línea recta tan pronto como el primer punto de información esté disponible. Esta línea irá a través del origen y el punto obtenido. Esto se ilustra en la Figura 4.7, donde con el signo "+" se designa al primer punto de información, y todos los "." representan la línea dibujada. Cuando aparece un segundo punto, obtenemos una línea

menos angulosa, basándonos en estos dos puntos. Esto se muestra en la Figura 4.8 donde con los dos signos "+" se designan los dos primeros puntos.

De manera similar, para cada intervalo de tiempo se realiza una nueva observación, en consecuencia, podemos obtener una curva de mejor fijación hasta dicho punto (hay que tener en cuenta que el término "curva" se usa en un sentido general que incluye "línea recta").

Usando este procedimiento de fijación de curvas, la curva puede continuar ajustándose tanto más, cuantos más puntos de información se obtengan a través del proceso de desarrollo. Como consecuencia, este proceso es progresivo en el sentido de que producirá líneas que se aproximan progresivamente a la línea final mejor ajustada. Adicionalmente, de modo sucesivo se fijarán curvas basadas en todas las medidas previas del tamaño del programa. Así, no se descarta ninguna información previa. La Figura 4.9 muestra la curva final de mejor ajuste para un conjunto de siete puntos utilizados como ejemplo.

4.7.2 Ajuste de Curva para el Modelo de dos Parámetros

Consideremos ahora el procedimiento de ajuste de curva para el modelo de dos parámetros indicado en la ecuación (4.2). Hay dos aproximaciones para el ajuste de curvas de dos parámetros, dependiendo de si el valor del parámetro de forma se determina

dinámicamente durante el proceso de desarrollo real, o si viene predeterminado antes del proceso de desarrollo basándonos en información histórica.

Determinación Dinámica del Valor del Parámetro de Forma. Ya que en la ecuación (4.2) "y(t)" contiene dos parámetros, α y β, se requieren al menos dos puntos de información para calcular estimadores para ellos. Como consecuencia, no se puede determinar una curva inicial hasta que no se disponga de dos puntos de información. Esta curva inicial irá a través del origen y de los dos primeros puntos. La forma de la curva inicial (puede ser cóncava o convexa) obtenida usando los dos primeros puntos será bastante sensible a su posición relativa con respecto al origen. Esto se ilustra en la Figura 4.10 (cóncava) y en la Figura 4.11 (convexa) donde los dos "+" designan los puntos medidos y todos los "." representan la curva ajustada.

Valor Predeterminado del Parámetro de Forma. Basándonos en las observaciones anteriores, parece que necesitamos una β predeterminada para ajustar "y(t)" al conjunto de puntos medidos, de tal manera que la extrapolación que se haga pueda ser de algún modo más controlada. Una manera natural de determinar el valor propio de β es a través del análisis empírico. Es decir, el proceso de desarrollo de un conjunto de programas generados usando la misma estrategia puede observarse mediante la medición de los valores de la métrica en contraposición con el tiempo de desarrollo. Con el ajuste de "y(t)" a todos los puntos medidos a través de una curva ajustada de menor ángulo, podemos obtener

valores apropiados para α y β de manera que "y(t)" defina la curva de evolución final mejor ajustada para la métrica. El mismo procedimiento se puede aplicar al proceso de desarrollo para todos los programas observados. Esto, entonces, generará parejas de valores para α y β, correspondientes a la evolución de la métrica en un desarrollo de programa particular. El valor medio de todas estas β, representa, por lo tanto, el comportamiento medio de la evolución de la métrica para el grupo de programas observados. Si todos los programas observados se han desarrollado utilizando la misma estrategia de desarrollo, es razonable asumir que podemos usar el valor medio de β para predecir el comportamiento de evolución general de la métrica para futuros programas construidos utilizando una estrategia similar.

4.7.3 Ajuste de Curva para el Modelo Transformado de un Parámetro

Habiendo determinado un valor propio para β (designado por β') para una estrategia de diseño dada, el modelo de dos parámetros indicado en la ecuación (4.2) puede ahora transformarse en un modelo que depende de un parámetro:

$$y(t) = \alpha \times t^{\beta'} \qquad (4.5)$$

donde $\alpha > 0$, y β' es una constante empíricamente derivada que depende de la estrategia de desarrollo. Este modelo modificado puede entonces usarse para aproximar la evolución de la métrica en programas escritos bajo la misma estrategia de desarrollo, como aquel del que se deriva β'.

Para un valor dado de β' en el modelo de un párametro de la ecuación (4.5), se puede aplicar el mismo procedimiento de ajuste de curva de un parámetro descrito en la sección 4.7.1, para determinar el comportamiento de la evolución de la métrica en una etapa inicial.

4.8 EVOLUCION DE LA METRICA

Las ideas y aseveraciones de la aproximación a la estimación del tamaño y del esfuerzo en esta investigación, pueden resumirse de la siguiente manera:

1.- Los valores de ciertas métricas del software, tales como VAR, n_2, y S, pueden reunirse en varios puntos a través del proceso de desarrollo.

2.- Bajo la estrategia de desarrollo "primera estructura de datos top-down", la evolución de las métricas de estructura de datos VAR y n_2 será convexa y seremos capaces de determinar el comportamiento de su evolución futura en una etapa inicial.

3.- Bajo la estrategia "incremental", el valor de la métrica del tamaño S evolucionará linealmente con el tiempo de desarrollo y seremos capaces de determinar su comportamiento de evolución futura en una etapa inicial.

4.- Las métricas VAR y n_2 son métricas del tamaño relacionadas (es decir, se pueden convertir en métricas del tamaño, con precisión razonable).

Para investigar la posibilidad de realización de nuestra aproximación, hemos de abordar los dos puntos que a continuación mostramos. También indicamos el capítulo de este trabajo en el que se abordan.

1.- Metodología para validar estas aseveraciones (este capítulo).

2.- Evaluación de la validez de estas premisas a través de los estudios empíricos llevados a cabo en esta investigación (Capítulos 5, 6, y 7).

4.9 TRABAJO BASICO PARA EL ESTUDIO EMPIRICO

4.9.1 Participantes

Participaron en este experimento 44 estudiantes que realizaban un curso sobre las métricas del software durante el verano de 1989 en le Universidad de Deusto. Denominamos a este experimento EXPER-2.

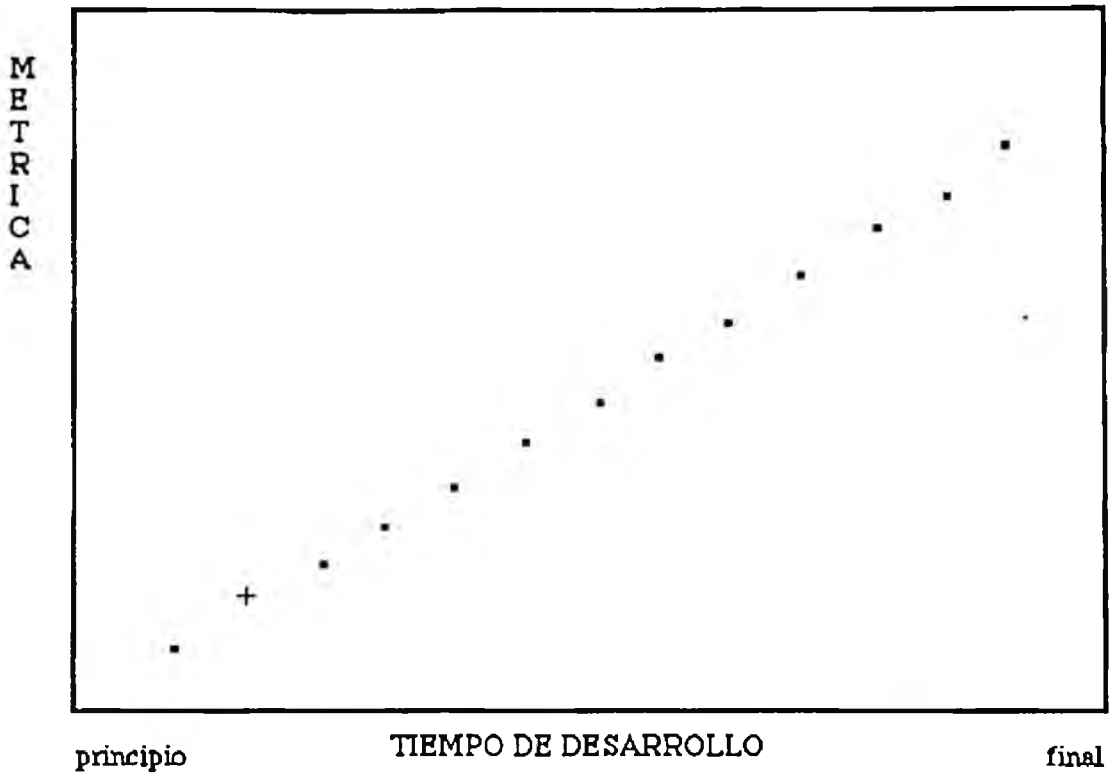


Figura 4.7

Curva Ajustada para el Modelo de un Parámetro: Un Punto

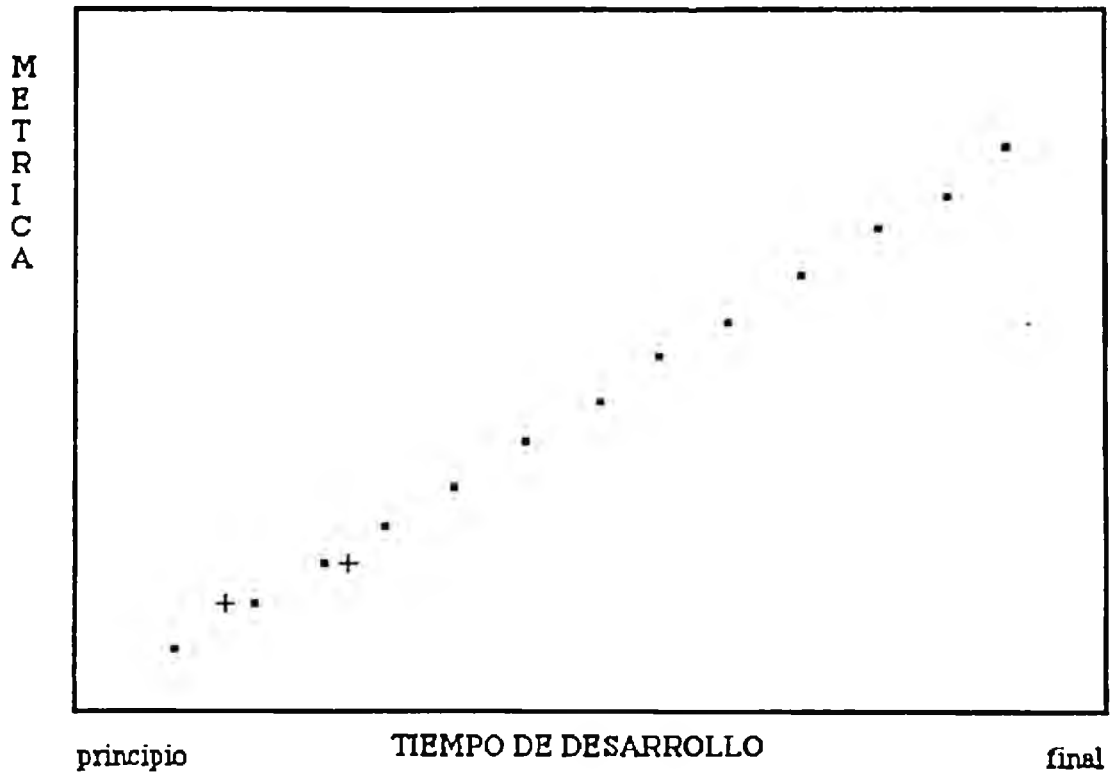


Figura 4.8

Curva Ajustada para el Modelo de un Parámetro: Dos Puntos

4.9.2 Tarea

Se solicitó a los participantes que realizaran dos programas en Pascal dentro de un período de tiempo de cinco semanas. Los dos programas se escogieron de los cuatro del primer experimento EXPER-1: Programa B ("calculador") y Programa D ("traductor").

4.9.3 Diseño Experimental - Control de los Factores de Confusión

Como se ha mencionado anteriormente, en este experimento queríamos tener más control sobre ciertas condiciones empíricas. Pretendíamos asegurarnos de que:

1.- Se seguían las estrategias de desarrollo requeridas.

2.- Se dedicaba el tiempo suficiente a la fase de diseño.

3.- Las estimaciones se realizaran cuidadosamente.

Para ello se incluyeron dos requerimientos de experimentación: "sesiones de trabajo de cuatro horas", y "entrevistas en puntos cruciales".

Como se ha mencionado en la Sección 4.2, suponemos que la evolución de una métrica depende al menos de dos factores: el programa, y el programador. Todos los participantes tenían que hacer los dos programas. Este requerimiento nos permitía observar

si los resultados dependían de un estilo de programación individual cuando el factor programa se mantenía constante. Todos los participantes tenían que realizar ambos programas, para poder controlar el factor programador.

Programar es una actividad mental cuya naturaleza es de un trabajo intensivo. Incluso si a todos los participantes se les pide usar una estrategia de desarrollo específica que se les ha explicado claramente, es todavía una pregunta sin respuesta hasta que punto seguirán estas estrategias. Además, el hecho de requerir a todos los participantes la construcción de los mismos programas, no asegura que ellos ejecuten las mismas tareas experimentales. Este es un problema típico en cualquier experimento de construcción de programas. Esto es debido a que el mismo programa puede tener varias aproximaciones al diseño que pueden llevarnos a cantidades sustancialmente diferentes del esfuerzo de desarrollo [DIJK72]. A pesar de esto, sin embargo, creemos que controlando esto tan cuidadosamente como sea posible, podemos, al menos, reducir la variabilidad causada por este factor.

4.9.4 Estrategia de Desarrollo

Tal y como se realizó en el primer experimento, se solicitó de los participantes el uso de dos estrategias de desarrollo específicas en la construcción de sus programas. Estas dos estrategias son la estrategia "primera estructura de datos top-

down",- y la estrategia "incremental" descritas anteriormente. Veamos, ahora, las reglas experimentales concernientes a estas dos estrategias.

Se consideró que el proceso de desarrollo de programa en este experimento era un procedimiento de cuatro etapas:

1.- Etapa de Especificación.

De forma similar al primer experimento, durante esta etapa se dieron las especificaciones de programa a los participantes que debían leerlas cuidadosamente y entender bien que requería cada programa.

2.- Etapa de Diseño Inicial.

Se pidió a los participantes que diseñaran sus programas usando la estrategia de desarrollo "primera estructura de datos top-down". En esta etapa del diseño del programa, se esperaba que los participantes se concentraran en el diseño global del programa. Es decir:

a) Identificar las estructuras básicas y los componentes más importantes del programa.

b) Determinar los procesos y conexiones de datos entre los componentes.

c) Desarrollar las estructuras de datos más importantes (mayormente de manera global) y diseñar los algoritmos básicos para manipularlos.

3.- Etapa de Diseño Detallado.

En esta etapa, además de revisar el diseño global del programa, los participantes tenían que desarrollar las estructuras de datos locales.

4.- Etapa de Codificación y Depuración.

Manteniendo nuestra premisa del uso de la estrategia "incremental" durante esta etapa, se pidió a los participantes el uso de la misma para codificar sus programas y al mismo tiempo probarlos y depurarlos. Como se ha indicado anteriormente, usando esta estrategia, solamente se codifica y añade un componente al programa en un momento determinado. En este caso un componente es uno o varios procedimientos relacionados en Pascal. Cada vez que un nuevo componente se añade al programa, el programa completo se prueba y se depura tan a fondo como sea posible, antes de que el siguiente componente entre a formar parte de él. Por lo tanto, esta aproximación puede describirse de la siguiente manera:

a) Codificar y probar un único componente.

b) Añadir otro componente.

c) Probar y depurar la combinación tanto como sea posible.

d) Repetir los pasos 2 y 3 hasta completar el programa.

4.9.5 Recopilación de Datos

Sesión de Trabajo de 4 Horas. De la misma manera que en el primer experimento, estabamos interesados en saber exactamente el tiempo que cada participante consumía en cada etapa de construcción de los programas. Para poder recoger con más precisión los tiempos consumidos, les sugerimos con much insistencia, que no trabajasen en un programa si no contaban con un período de tiempo con una posibilidad de interrupción mínima. A estos períodos de tiempo les denominamos "sesión de trabajo". Pedimos que cada sesión no fuese de menos de una hora y de no más de cuatro horas.

Entrevistas en Puntos Cruciales. La finalización de cada una de las cuatro etapas era un punto crucial. Además, dentro de cada etapa, el final de cada sesión de trabajo era también un punto crucial. Al igual que en el primer experimento, se les pidió que en cada punto crucial completaran un informe de los progresos denominado "Informe de Punto Crucial". Una diferencia con respecto al primer experimento fué que se les pidió que

informaran al administrador del experimento en cada punto crucial, y junto con el administrador completaran el informe de punto crucial. A esto lo llamamos "Entrevista de Punto Crucial". El Jefe de Grupo guardó todos los informes de punto crucial a lo largo del proceso de desarrollo. Además, en cada entrevista de punto crucial, eramos capaces de asegurarnos que los participantes seguian las estrategias requeridas y la regla "sesión de trabajo de cuatro horas".

Por lo tanto, para mantener nuestra premisa de recoger los datos de la métrica durante el proceso de desarrollo, se recogieron versiones del programa sin errores de sintaxis en todos los puntos cruciales. Con cuarenta y cuatro participantes escribiendo dos programas, cada uno de los cuales requería más o menos 8 puntos cruciales, se complementaron cerca de setecientas entrevistas de punto crucial durante el experimento.

Estimaciones Subjetivas "Estructuradas". De manera similar al primer experimento, se recogieron durante el proceso de desarrollo las estimaciones subjetivas de cada programa. De cualquier modo, y de forma adicional a la estimación del tiempo total de desarrollo y del tamaño final del programa, los participantes también estimaban VAR y n_2. En el Capítulo 5, se muestra como estas estimaciones subjetivas de VAR y n_2 se usaron como base para estimaciones indirectas del tamaño.

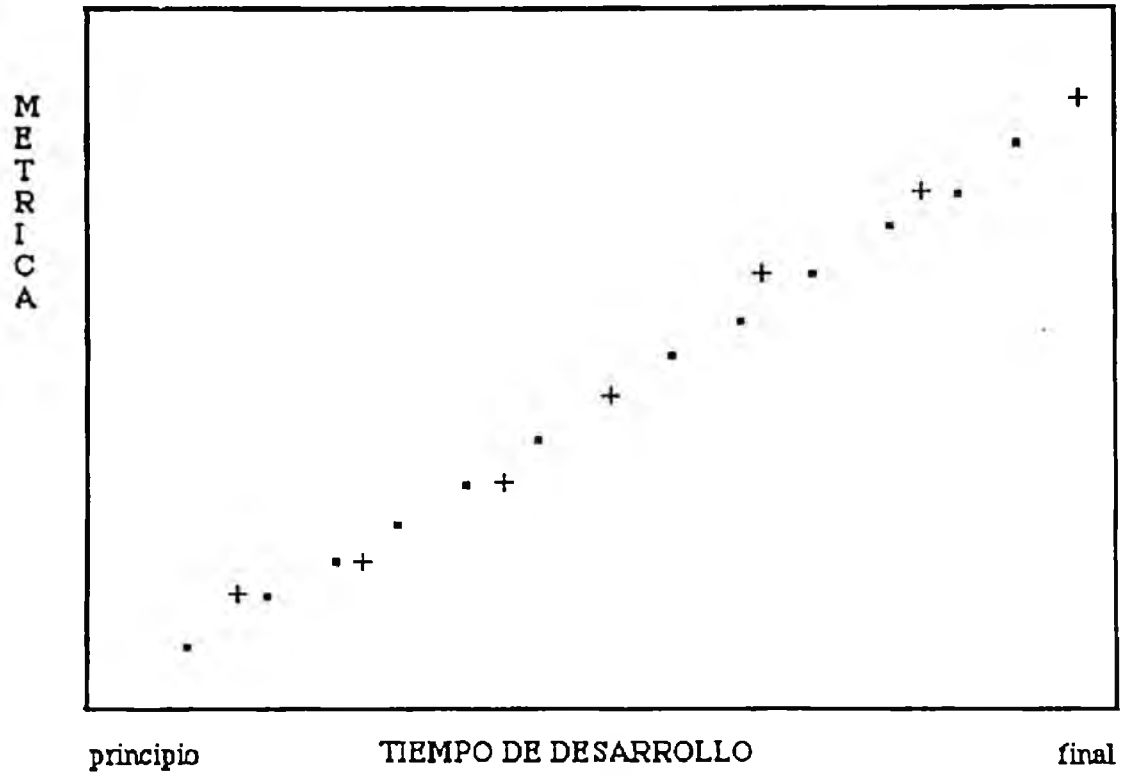


Figura 4.9

Curva Ajustada para el Modelo de un Parámetro

Deberíamos señalar que las estimaciones subjetivas agrupadas en este experimento deben considerarse estimaciones subjetivas estructuradas ya que se han producido de una manera organizada. Cuando una versión del programa sintácticamente correcto se convertía en un punto crucial, el administrador analizaba esta versión intermedia para obtener los valores de la métrica en ese punto. A cada participante se le pidió que actualizara sus propias estimaciones del esfuerzo total, tamaño total, VAR, y n_2 basándose en los valores reales "in situ". Los participantes hacían sus mejores estimaciones con tales datos en su mente, además de su conocimiento a cerca del estado del desarrollo del programa. Por lo tanto, las estimacione subjetivas se reunieron en este experimento y deben ser distinguidas de suposiciones típicamente "ad-hoc", es decir, para un fín determinado.

Cuestionario Post-Experimental. Para obtener de los participantes información relativa a las estrategias de desarrollo y procedimientos experimentales, se les presentó un cuestionario después de que terminaron sus tareas experimentales. Las preguntas del cuestionario cubrían aspectos tales como su familiaridad con el programa, con las estrategias de desarrollo, si era dificil seguir estas estrategias, si estas estrategias se podían utilizar de manera real, si las reglas y requerimientos del experimento habían tenido un impacto significativo en los resultados, etc. El propósito del cuestionario era determinar si había consenso a cerca de que factores habían tenido mayor impacto sobre el resultado y también nos permite conocer los

efectos de los controles del experimento sobre los participantes. Los resultados del cuestionario relativos al uso de las estrategias de desarrollo se resumen en el Capítulo 8.

Estadísticas. Las estadísticas de resumen de las cuatro medidas importantes usadas en nuestro estudio para los 88 programas completados en EXPER-2 se muestran en la Tabla 4.1. Los Apéndices F y G contienen la información detallada.

Tabla 4.1

Estadísticas del Estudio EXPER-2

Programa	Individuos	Medida	Mín	Máx	Media	Desv.Est.
1	6	E(horas)	16	54	27	9
		S(líneas C.)	267	784	463	96
		VAR(variab.)	40	118	64	16
		n (operando) 2	80	157	111	17
2	5	E(horas)	9	41	17	6
		S(líneas C.)	180	645	355	98
		VAR(variab.)	18	98	46	20
		n (operando) 2	84	193	122	31

4.9.6 Comparación entre EXPER-1 y EXPER-2

Las mayores diferencias entre en primer y el segundo experimento se resumen en la Tabla 4.2. Como ya se ha mencionado, la mejora más importante de EXPER-2 sobre EXPER-1 es el control experimental más riguroso.

Se logró un control riguroso a través de unos procedimientos experimentales de diseño cuidadosos y de las estrategias de desarrollo llevadas a cabo debidamente.

Tabla 4.2

Comparación entre EXPER-1 y EXPER-2

Rasgos	EXPER-1	EXPER-2
Participante	21 estudiantes	44 estudiantes
Tarea	Escribir 1 programa en dos semanas	Escribir 2 programas en cinco semanas
Temas de Programación	Cuatro Programas: - Huffman (6) - Calculador (5) - Intérprete (5) - Traductor (5)	Dos Programas: - Calculador (44) - Traductor (44)
Estrategia de Desarrollo	Primera estructura de datos Top Down	- Primera estructura de datos Top-Down - Incremental
Etapas de Desarrollo	- Especificación - Diseño - Codificación - Depuración	- Especificación - Diseño inicial - Diseño detallado - Codificación/depuración
Entrevistas punto crucial	No	Si
Sesión de Trabajo	Cinco horas por sesión, no controladas	Cuatro horas por sesión, controladas
Estimaciones Subjetivas	Sin estructurar y realizadas sin cuidado	Estructuradas, completadas en las estrevistas de punto crucial
Métricas Estimadas	Esfuerzo y Tamaño	Esfuerzo, tamaño, VAR y n_2

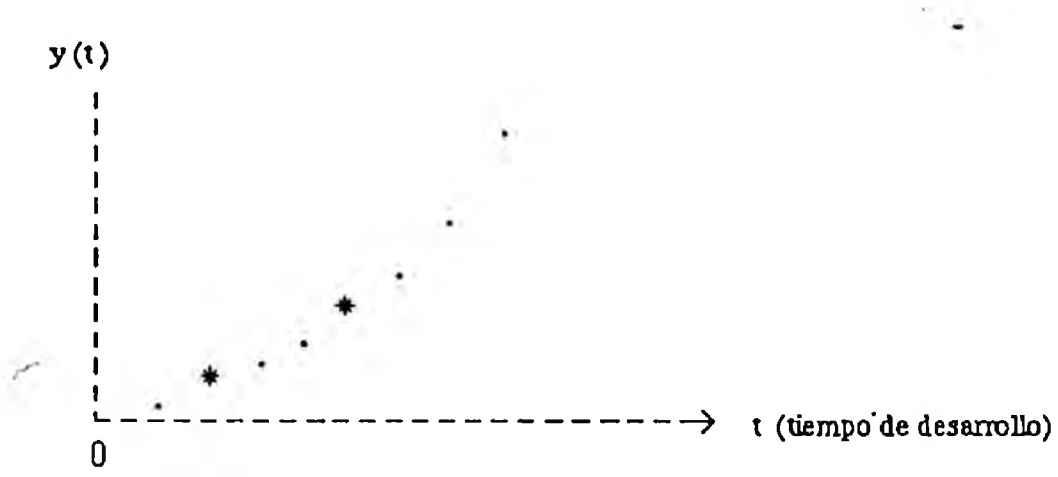


Figura 4.10

Curva Ajustada para el Modelo de dos Parámetros: Cóncava

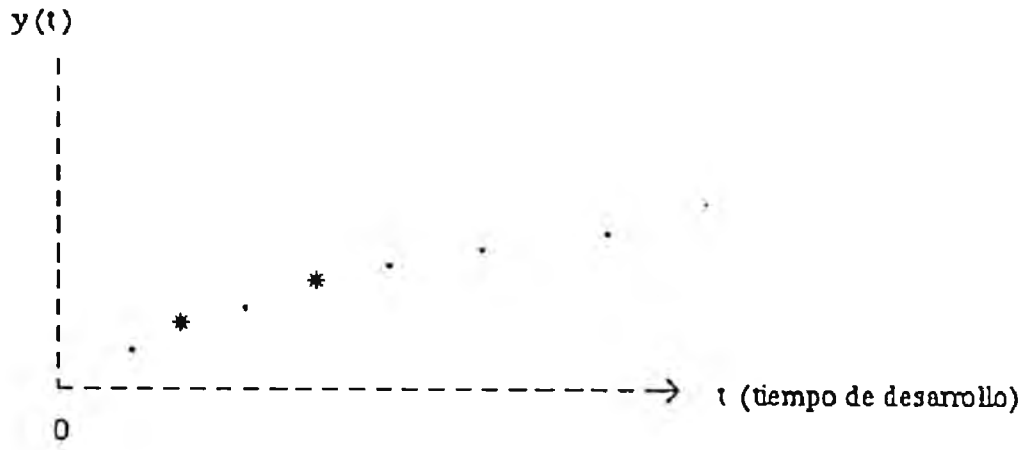


Figura 4.11

Curva Ajustada para el Modelo de dos Parámetros: Convexa

CAPITULO 5

CONFIRMACION DE LA ESTIMACION DEL TAMAÑO

En el Capítulo 3 hemos indicado que nuestra aproximación a la estimación indirecta del tamaño, basada en n_2 y usando la ecuación de longitud de Halstead, no obtuvo resultados satisfactorios en el estudio EXPER-1. Supusimos que un subconjunto de n_2 llamado VAR, podía reemplazar a n_2 para una estimación mejor. En este capítulo se consideran otros modelos de estimación del tamaño además del de Halstead, que relacionan el tamaño del programa con n_2 o con VAR. Evaluamos la aproximación a la estimación del tamaño indirecta basada en n_2 y VAR usando estos modelos, utilizando los datos recogidos en nuestro segundo experimento, EXPER-2.

5.1 MODELOS DE ESTIMACION DEL TAMAÑO BASADOS EN n_2

5.1.1 Nuevo Estudio sobre la Ecuación de Longitud de Halstead

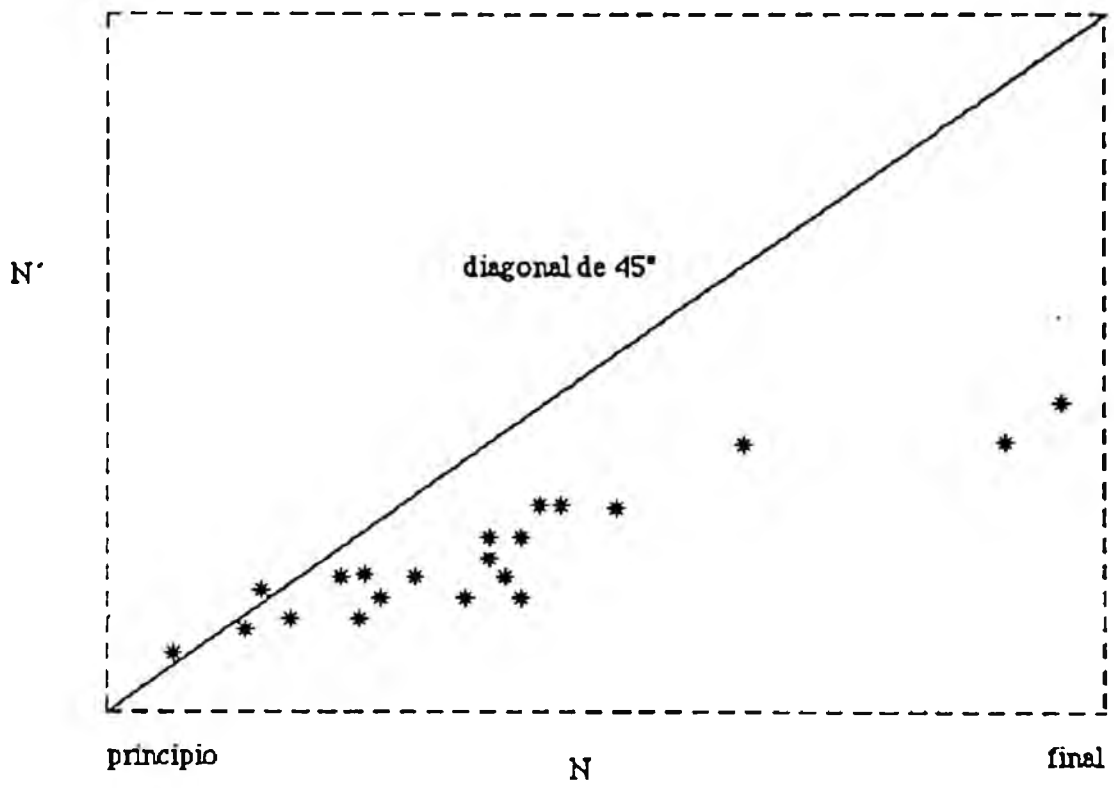
Como se apuntó en el Capítulo 3, surgieron algunas preguntas con respecto a la validez de la ecuación de longitud de Halstead para los programas realizados en Pascal, ya que en los datos del estudio EXPER-1, N' subestimó a la N en un 32%. Para investigar la validez de la ecuación de longitud de Halstead con programas

codificados en Pascal, analizamos un grupo de 78 programas. La Tabla 5.1 muestra el origen de los programas y los rangos de tamaño de los mismos. El primer conjunto de datos denominado GRUPO-1 estaba formado por 24 programas de medio a gran tamaño procedentes de varias aplicaciones tales como sistemas de manejo de base de datos, rutinas de manejo de librerias, rutinas de utilidad, etc. Estos programas provenian de cursos realizados en la Facultad de Informática de la Universidad de Deusto. En el Apendice H se muestran los resultados de la métrica para estos 24 programas. El segundo grupo de programas denominado GRUPO-2, estaba formado por 33 proyectos de programación de un curso de la Facultad relativo al manejo de bases de datos y organizaciones de ficheros. En el Apendice I se muestran los resultados de la métrica para estos 33 programas. El tercer conjunto de programas proviene del estudio EXPER-1. Denominamos a estos tres conjuntos de programas "datos de ensayo".

Tabla 5.1

Fuentes y Rangos de Tamaño de los Datos de Ensayo

Conjunto de Datos	Programas	Fuente	Rango de Tamaño en LOC
GRUPO-1	24	ejemplos al azar	502 a 4572
GRUPO-2	33	asignaciones de clase	270 a 927
EXPER-1	21	experimento	215 a 662



Coeficiente de correlación = 0.97

RE' = 0,42 MRE' = 0.42 PRED (0.25) = 0,18

Figura 5.1

Evolución de la Ecuación de Longitud (Datos de Ensayo)

La Figura 5.1 muestra la gráfica en la que se enfrentan N' y N para los 78 programas analizados. Está claro que N' es una subestimación de N por un valor considerable. El RE' de la estimación de N' contra N era +0,42; MRE' era 0,42 (RE' y MRE' son iguales porque no hay sobrestimaciones), y PRED(0,25) era el 18%. Estos resultados nos permiten considerar otros formas de estimar el tamaño a partir de n_2. Una primera alternativa es considerar un modelo de regresión.

5.1.2 Modelo de Estimación del Tamaño usando la Ecuación de Regresión

Ya que se ha demostrado que N está muy relacionada con S, y que S es la medida más convencional del tamaño del programa, se analizó la relación entre n_2 y S. Como se mencionó en el Capítulo 2, S nos da el tamaño real del programa en LOC. Descubrimos que n_2 y S, para el conjunto de los 78 programas estudiados, estaban muy relacionados. Por ello, en primer lugar, consideramos el modelo de regresión lineal:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \qquad (5.1)$$

donde:

Y_i es el valor de la variable dependiente S en la observación i-ésima.

β_0 y β_1 son parametros

X_i es el valor de la variable independiente de n_2 en la i-ésima observación

ϵ_i es independiente y distribuida normalmente con valor medio de 0 y varianza constante

i asume valores de 1 a n.

Los valores desconocidos de los parametros β_0 y β_1 se estiman por el método del angulo menor. Es decir, los valores estimados se escogen de tal manera que se minimize la suma de los angulos residuales. El atractivo de esta aproximación se basa en el hecho de que b_0 y b_1, los estimadores de β_0 y β_1, respectivamente, son independientes y tienen la varianza minima entre todos los estimadores lineales imparciales [NETE74].

5.1.3 Aplicación de la Ecuación de Regresión

La ecuación de regresión obtenida usando el modelo (5.1) queda para nuestros datos de ensayo:

$$S'(n_2) = 52,5 + 3,52n_2 \qquad (5.2)$$

Debido a que la constante "52,5" es relativamente pequeña en comparación con el valor medio de S "918", vamos a considerar a continuación el modelo de regresión lineal que pasa por el origen:

$$Y_i = \beta_1 X_i + \epsilon_i \qquad (5.3)$$

La ecuación de regresión obtenida usando este modelo para los datos de ensayo es:

$$S'(n_2) = 3,64n_2 \qquad (5.4)$$

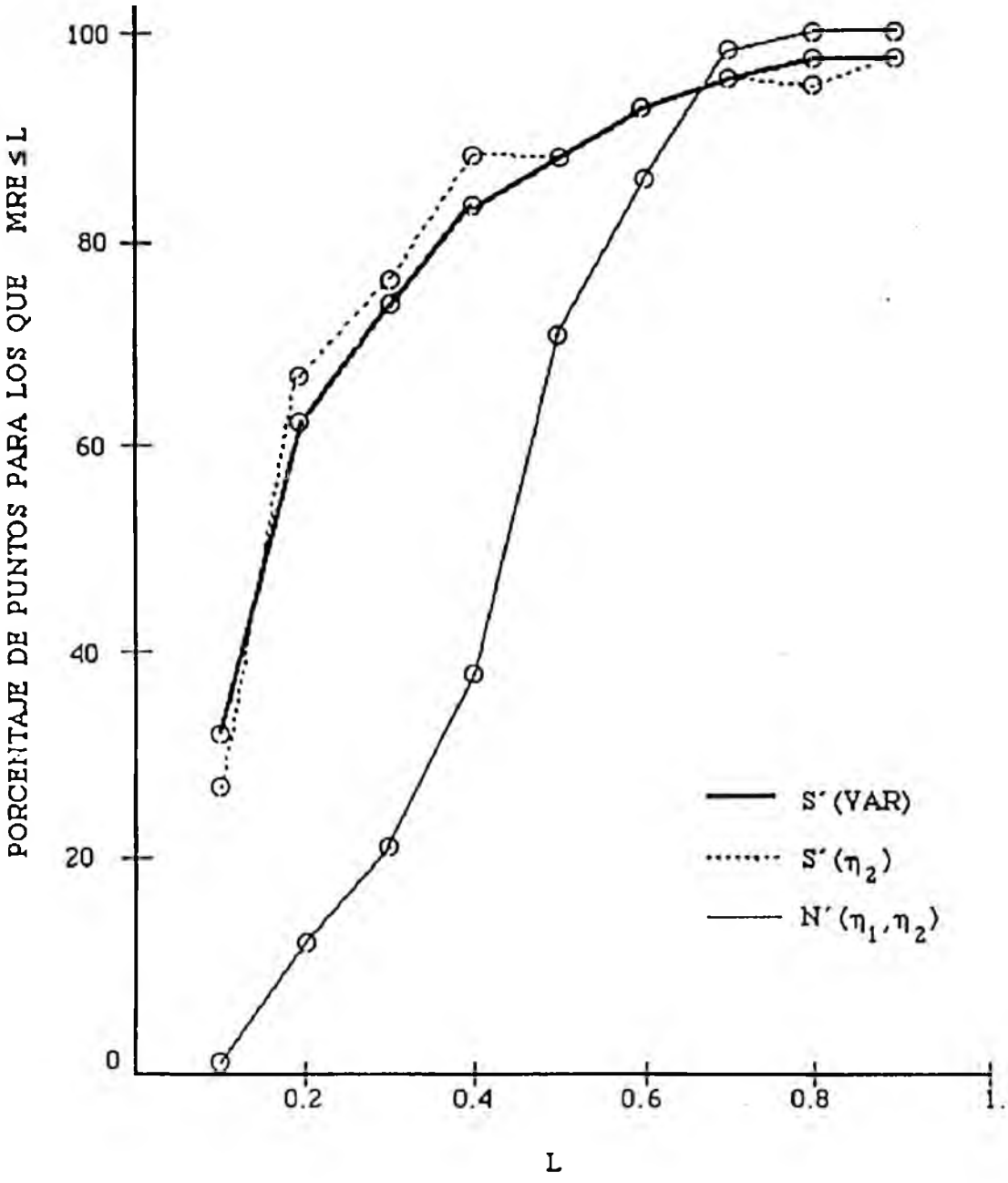


Figura 5.2

La Ecuación de Longitud frente a la Ecuación de Regresión

(Datos de Ensayo)

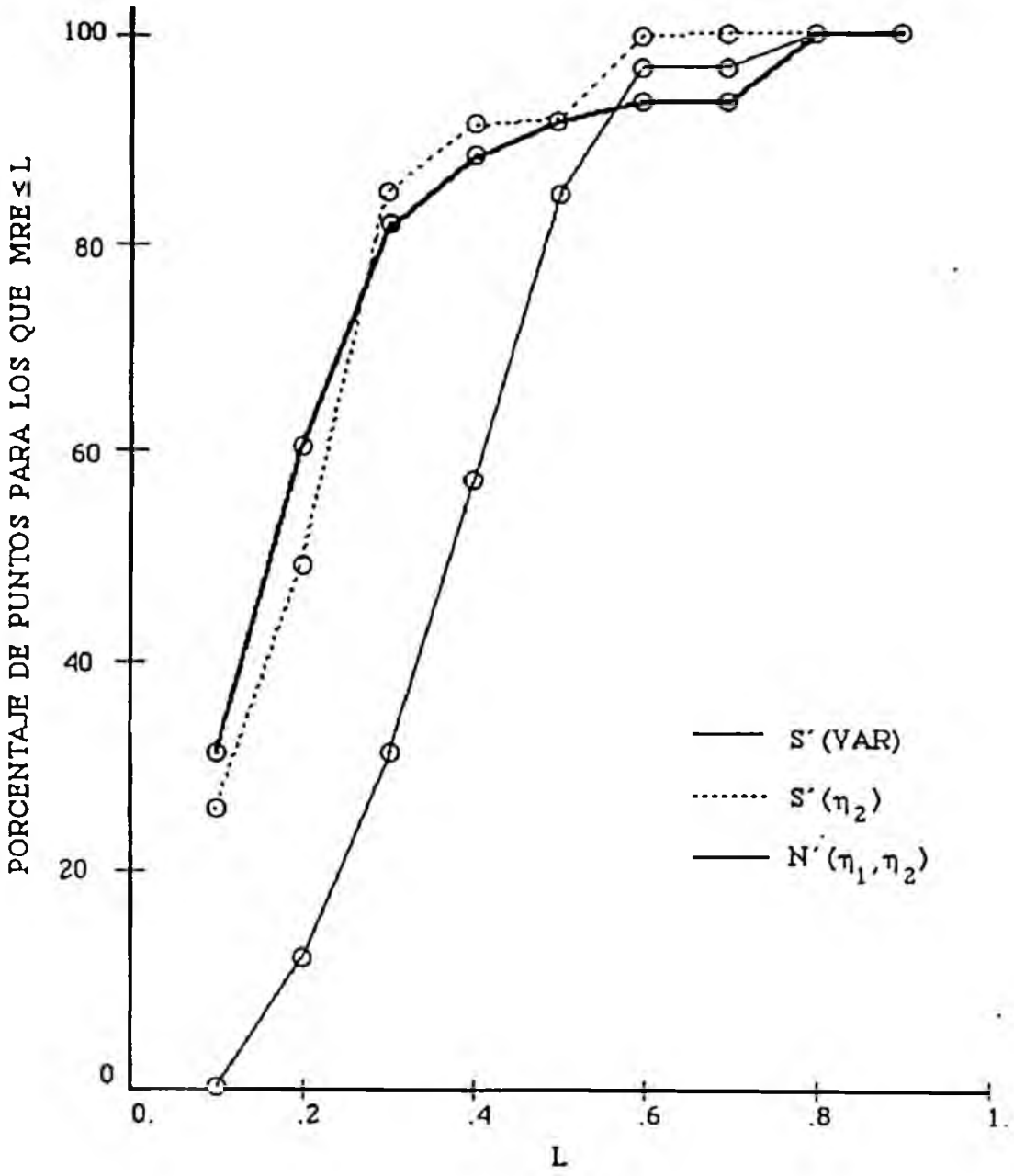


Figura 5.3

La Ecuación de Longitud frente a la Ecuación de Regresión

(EXPER-2a)

5.2 MODELOS DE ESTIMACION DEL TAMAÑO BASADOS EN VAR

De manera similar al modelo del tamaño que relaciona n_2 con S, vamos a considerar en primer lugar el modelo de regresión lineal (5.1). La ecuación de regresión entre S y VAR obtenida para los datos de ensayo es:

$$S'(VAR) = 102 + 5,31VAR \qquad (5.5)$$

Debido a que la constante "102" no es muy pequeña en comparación con el valor medio de S, se ha mantenido la constante en esta ecuación de regresión.

5.3 COMPARACION ENTRE LOS MODELOS DE ESTIMACION DEL TAMAÑO

5.3.1 Datos de Ensayo

Vamos a comprobar la validez de las ecuaciones de regresión (5.4) y (5.5) usando, en primer lugar, los datos de ensayo. En la siguiente subsección, estas mismas ecuaciones se aplicarán a los datos del estudio EXPER-2. Para cada uno de los 78 programas de los datos de ensayo, se convertía VAR en S usando la ecuación de regresión (5.5), y el valor convertido se usaba para estimar la S real.

En la Tabla 5.2 resumimos los datos de la comparación entre los modelos de estimación del tamaño. En la primera fila de los datos de estudio se resumen los resultados de esta estimación. La notación S'(VAR) representa el tamaño estimado en LOC usando el valor de VAR real y la ecuación (5.5). De la misma manera, $S'(n_2)$, se muestra en la segunda fila de la tabla, y representa el tamaño estimado en LOC usando el valor real de n_2 y la ecuación (5.4). Adicionalmente, la N estimada usando los valores reales de n_1 y n_2 y la ecuación de longitud de Halstead, se representa como $N'(n_1,n_2)$ en la tercera fila de la tabla 5.2.

Como se muestra en la Tabla 5.2, los resultados de la estimación basados en VAR y n_2 usando la ecuación de regresión, parecen ser mucho más satisfactorios que aquellos obtenidos con el uso de la ecuación de Longitud de Halstead. Los valores MRE son más pequeños; los valores PRED(0,25) son más grandes. La cuarta columna de la tabla indica que, de acuerdo con los resultados de la prueba de la T para $\alpha < 0,10$, la estimación que usa la ecuación de regresión es mejor de manera significativa que la que usa la ecuación de longitud de Halstead. Los perfiles MRE de estas tres estimaciones se muestran en la Figura 5.2.

Estos resultados sugieren que partiendo sólo de un elemento, VAR o n_2, podemos calcular con una precisión razonable la estimación del tamaño usando la ecuación (5.4) o la ecuación (5.5). Parece que esta estimación del tamaño es mucho más precisa que la estimación del tamaño usando la ecuación de longitud de Halstead, la cual requiere dos elementos, n_1 y n_2.

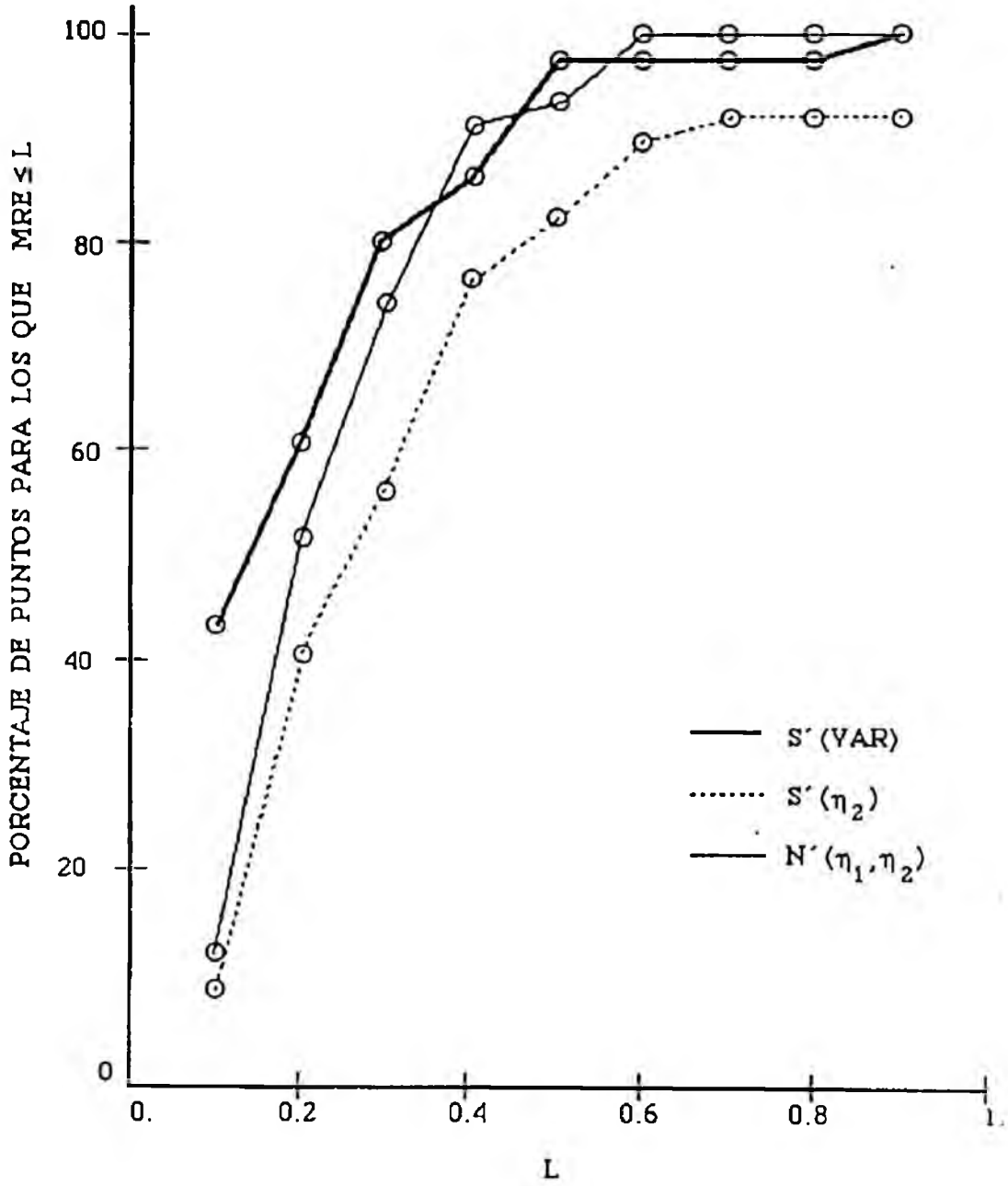


Figura 5.4

La Ecuación de Longitud frente a la Ecuación de Regresión

(EXPER-2b)

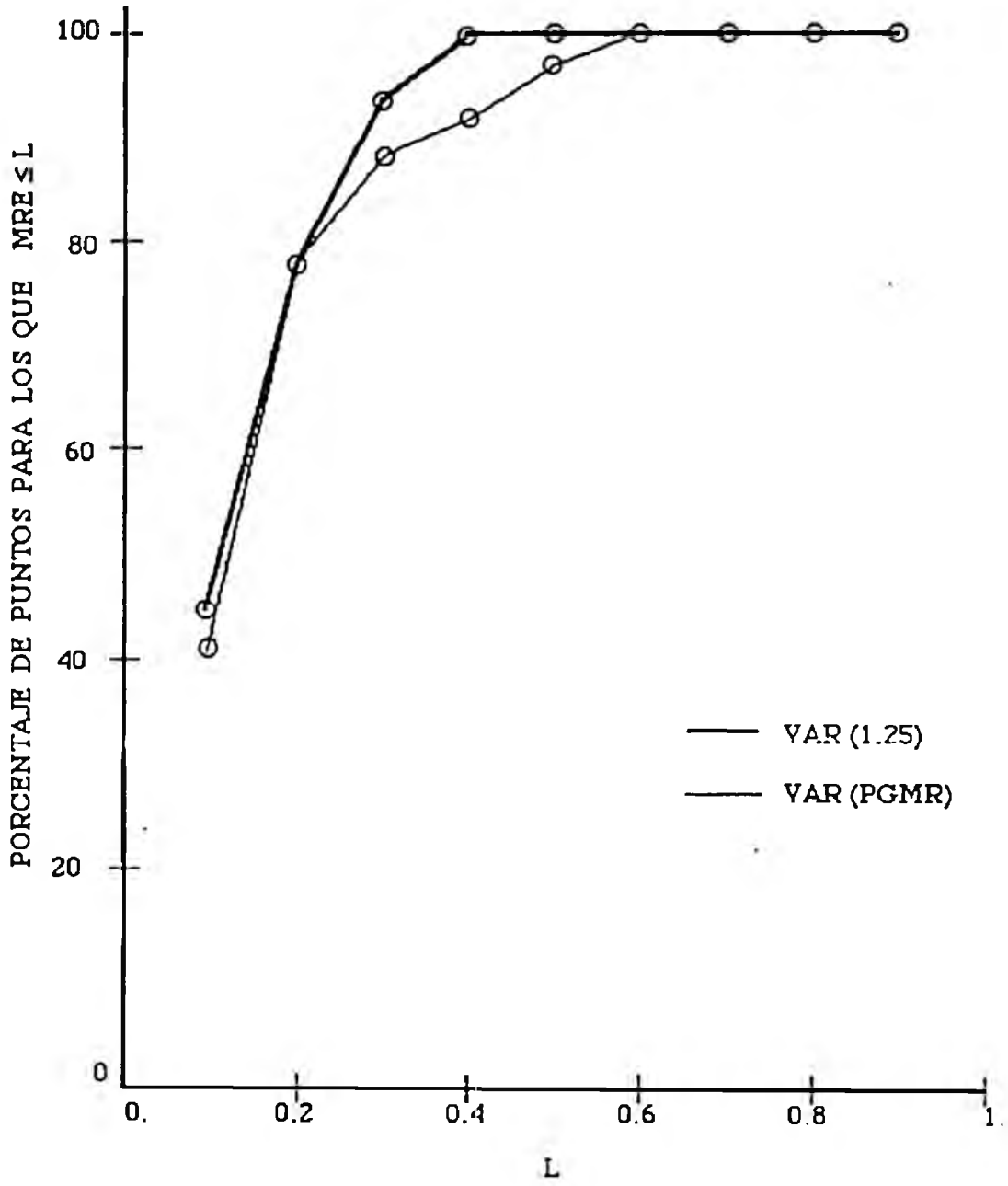


Figura 5.5

VAR (1,25) frente a VAR (PGMR) (EXPER-2a)

5.3.2 Datos de EXPER-2

Debido a que las dos ecuaciones de regresión, (5.4) y (5.5), se evaluaron usando el mismo conjunto de datos del cual se habían derivado, los resultados alcanzados no son suficientes.

Para confirmar la validez de estas dos ecuaciones usando datos independientes de los datos de ensayo, se aplicaron a los dos conjuntos de datos recogidos en nuestro segundo experimento, EXPER-2. En la segunda parte de la Tabla 5.2 se resumen estos resultados.

Tabla 5.2

Comparación de Modelos de Estimación del Tamaño

Datos	Estimación	MRE'	PRED(0,25)	RE'	rango
De Ensayo	S'(VAR)	0,21	64%	-0,08	1
	$S'(n_2)$	0,22	73%	+0,01	1
	$N'(n_1,n_2)$	0,42	18%	+0,42	3
EXPER-2a	S'(VAR)	0,19	80%	+0,01	1
	$S'(n_2)$	0,19	73%	+0,10	1
	$N'(n_1,n_2)$	0,38	18%	+0,38	3
EXPER-2b	S'(VAR)	0,19	73%	+0,01	1
	$S'(n_2)$	0,32	55%	-0,29	3
	$N'(n_1,n_2)$	0,22	61%	+0,22	1

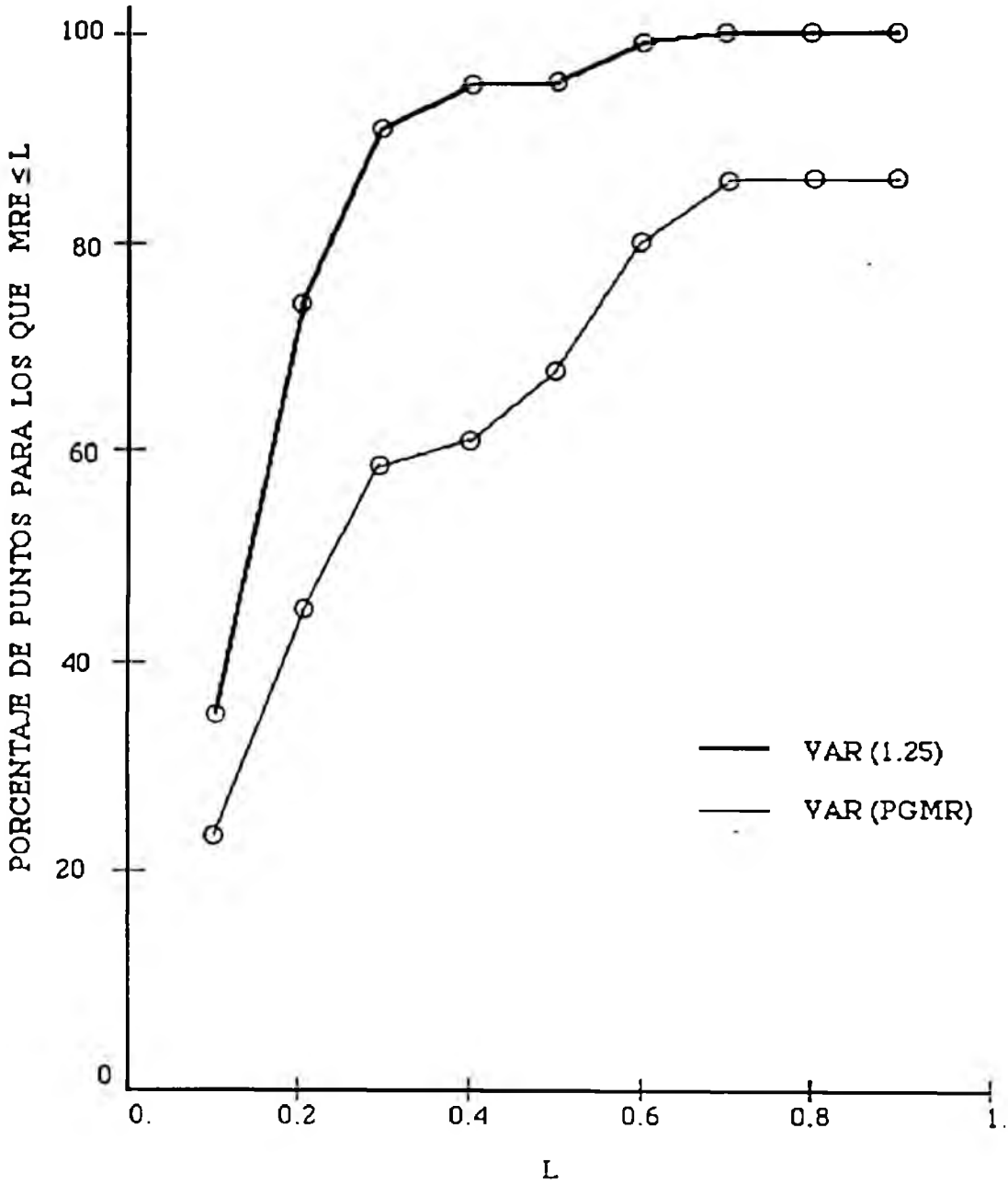


Figura 5.6

VAR (1,25) frente a VAR (PGMR) (EXPER-2b)

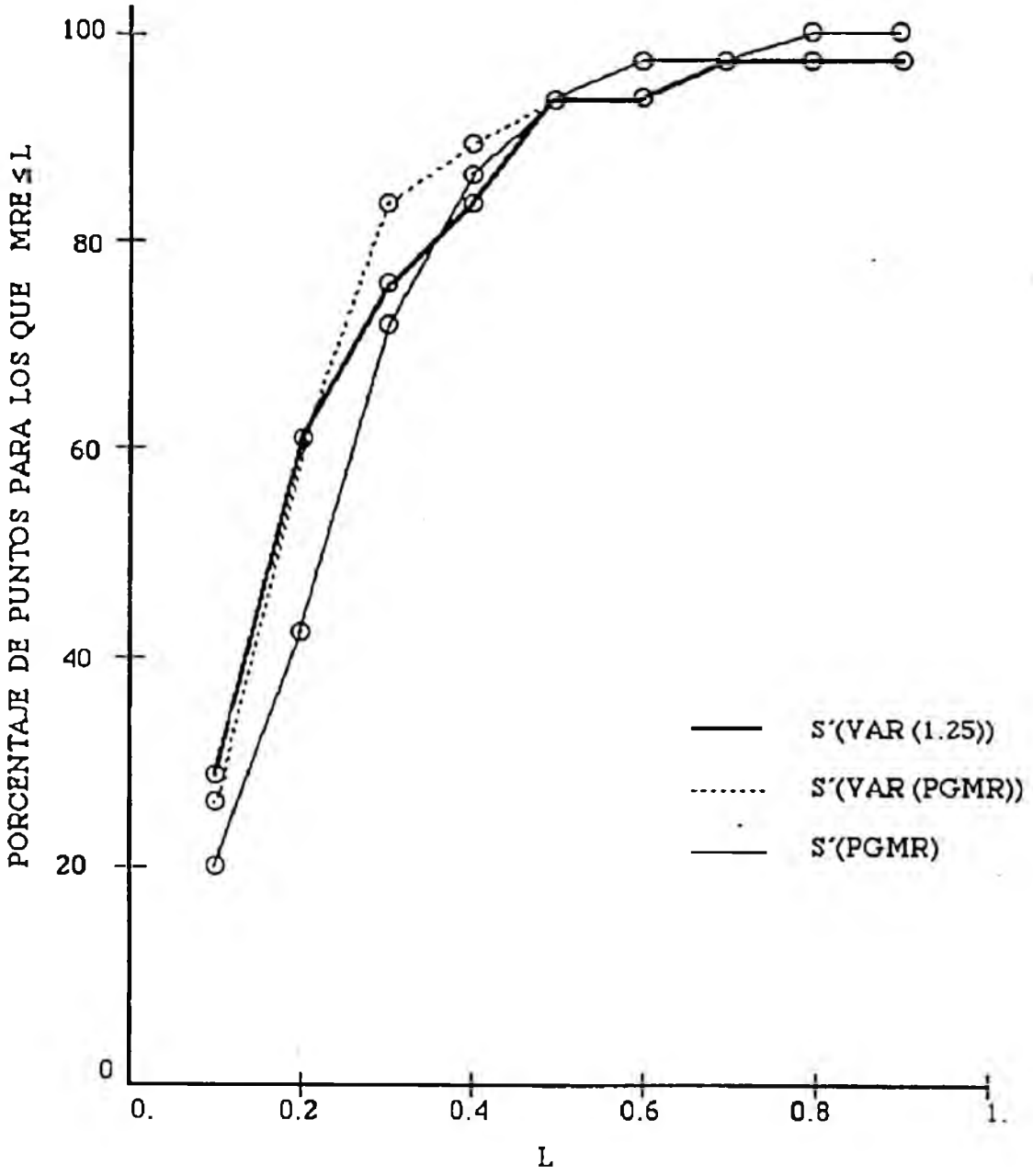


Figura 5.7

Estimación Inicial del Tamaño (EXPER-2a)

Como se muestra en la tabla 5.2, para el conjunto de programas EXPER-2a, las estimaciones del tamaño usando las ecuaciones de regresión fueron mejores que aquellas que usaban la ecuación de longitud de Halstead. Sin embargo, este resultado no se repite para el EXPER-2b, ya que las estimaciones del tamaño basadas en n_2 usando la ecuación de regresión no fueron tan buenas como las de los otros dos conjuntos de estimaciones. Adicionalmente, la estimación N usando la ecuación de longitud de Halstead para EXPER-2b fue substancialmente mejor que las de EXPER-2a, a pesar de que todavía subestimaban el valor real de N en un 22%. El motivo de este resultado está basado en las especificaciones del programa de EXPER-2b, el "traductor", ya que dicho programa contenía mayormente procesos recursivos y se utilizó para su diseño la técnica de análisis sintáctico de descenso recursivo, por lo que los programas fueron de menor tamaño de lo habitual. Los perfiles MRE para las estimaciones de EXPER-2a y EXPER-2b, se muestran en las Figuras 5.3 y 5.4.

Al comparar los resultados obtenidos en EXPER-2a y EXPER-2b, tal y como se muestran en la Tabla 5.2, y en las Figuras 5.3 y 5.4, vemos que el modelo de estimación del tamaño basado en VAR usando la ecuación de regresión (5.5) parece ser más consistente y fiable que los otros dos modelos. Este resultado es esperanzador, ya que implica que tan pronto como podamos estimar VAR, podremos estimar el tamaño. Posteriormente, tal y como hemos argumentado en el Capítulo 3, la estimación inicial de VAR debe ser más precisa que la estimación inicial de n_2 porque una

porción mayor de VAR se crea en una etapa inicial. Así, el siguiente paso en nuestra discusión es mostrar la precisión con que fué estimada VAR en una etapa inicial.

5.4 ESTIMACION INICIAL DE VAR

Como ya hemos mencionado, una estimación inicial es la estimación obtenida al final de la fase de diseño detallado. Había dos maneras de obtener tal estimación de VAR en el segundo experimento. La primera estimación inicial de VAR se calculaba multiplicando 1,25 por el valor inicial de VAR medido al final de la etapa de diseño. Este método se basaba en nuestra especulación de que bajo la estrategia de desarrollo "primera estructura de datos Top-down", aproximadamente un 80% de VAR se crearía al final de la etapa de diseño. Las estimaciones de VAR obtenidas de esta manera se designan como VAR'(1,25). La segunda estimación inicial de VAR procedía directamente de las propias estimaciones de VAR de cada programador al final de la etapa de diseño. Este tipo de estimación del VAR se designa como VAR'(PGMR).

La Tabla 5.3 resume los resultados de la estimación inicial de VAR para el EXPER-2a y el EXPER-2b usando los métodos arriba indicados. En EXPER-2a, ambos métodos producian resultados bastante satisfactorios; los valores de MRE' son pequeños y los porcentajes de PRED(0.25) son altos. Cuando se comparan los resultados generados por estos dos métodos, se observa que no son diferentes de modo significativo. Los

resultados para EXPER-2b son otra historia. El primer método todavía mantiene resultados satisfactorios. Pero, las estimaciones subjetivas de VAR parecian sobrestimar el valor de VAR real. Los resultados de la prueba T mostraban que el primer método generaba estimaciones más precisas que el segundo método. Los perfiles MRE para estas estimaciones del EXPER-2a estan contenidas en las Figuras 5.5 y 5.6.

Tabla 5.3

Estimación Inicial de VAR (EXPER-2)

Programa	Estimación	MRE'	PRED(0,25)	RE'	rango
1	VAR'(1,25)	0,13	89%	+0,01	igual
	VAR'(PGMR)	0,14	84%	-0,01	igual
2	VAR'(1,25)	0,16	77%	-0,02	1
	VAR'(PGMR)	0,39	50%	-0,33	2

La razón por la que el primer método de estimación de VAR fué tan bueno era porque el promedio de los ratios entre los valores iniciales de VAR y los valores finales de VAR fueron de 0,79 y 0,81 en EXPER-2a y EXPER-2b, respectivamente. Una razón por la que el segundo método funcionaba bien al menos en EXPER-2a, es porque las estimaciones subjetivas recogidas en EXPER-2 son estimaciones subjetivas "estructuradas" como se mencionó en la Sección 3.9. Es decir, los participantes recibierón los valores corrientes de VAR al final de la etapa de diseño detallado antes de que fueran realizadas las estimaciones.

Ya que estos participantes podían decir si se había creado la mayor parte de VAR en el punto de estimación, sus estimaciones subjetivas de VAR debían estar muy cercanas a las estimaciones generadas por el primer método.

De cualquier modo, las estimaciones subjetivas de VAR no fueron tan buenas para el EXPER-2b. En las entrevistas post-experimentales, la mayoría de los participantes indicaron que cuando leyeron por primera vez las especificaciones del programa 2 (un traductor de un lenguaje de base de datos), les pareció un programa muy difícil. No se percataron de que el programa no era tan difícil como habian pensado hasta encontrarse en la mitad de camino del proceso de construcción. Este cálculo mal realizado, se reflejaba en su sobrestimación de VAR. De hecho, esta es una buena demostración de que incluso las estimaciones subjetivas "estructuradas" pueden también no ser fiables.

5.5 ESTIMACION INICIAL DEL TAMAÑO

En la sección 5.3, hemos mostrado que las estimaciones del tamaño basadas en los valores reales de VAR usando la ecuación de regresión (5.5) fueron bastante precisas para los programas de EXPER-2. En la sección precedente, se han discutido dos métodos para la estimación de VAR en una etapa inicial. Se ha demostrado que uno de ellos trabaja bastante bien. En esta sección evaluaremos nuestra aproximación indirecta al tamaño inicial combinando los resultados mostrados en las dos secciones previas.

Es decir, la estimación inicial de VAR obtenida usando los dos métodos descritos antes, se usará en la ecuación de regresión (5.5) para calcular la estimación del tamaño inicial en LOC. Estos dos tipos de estimaciones del tamaño se representan como S'(VAR'(1,25)) y S'(VAR'(PGMR)), respectivamente. Sus resultados se comparan con los resultados de las estimaciones del tamaño subjetivas en LOC, designadas como S'(PGMR). La Tabla 5.4 resume los resultados de esta comparación.

Tabla 5.4

Estimación Inicial del Tamaño (EXPER-2)

Programa	Estimación	MRE'	PRED(0,25)	RE'	rango
1	S'(VAR'(1,25))	0,21	68%	+0,02	igual
	S'(VAR'(PGMR))	0,20	75%	+0,01	igual
	S'(PGMR)	0,23	57%	+0,02	igual
2	S'(VAR'(1,25))	0,23	59%	-0,00	1
	S'(VAR'(PGMR))	0,30	57%	-0,18	1
	S'(PGMR)	0,42	52%	-0,39	3

Como se muestra en la Tabla 5.4, para EXPER-2a, los tres métodos para la estimación del tamaño al final de la etapa de diseño detallado no fueron diferentes de modo significativo unos de otros. Por una parte, en EXPER-2b, los dos primeros métodos parecían ser significativamente mejores que el tercero. En cualquier caso, debe tenerse en cuenta que entre los tres métodos que consideramos, el primero parece ser el más consistente

teniendo presente los dos programas. Suponemos que esto es debido al hecho de que el primer método no implica ningun tipo de estimación subjetiva. Los perfiles MRE para estas estimaciones del EXPER-2a, y del EXPER-2b, se muestran en las Figuras 5.7 y 5.8.

5.6 CONFIRMACION DE LA ESTIMACION

La viabilidad de nuestra aproximación indirecta a la estimación del tamaño ha sido demostrada en nuestro segundo experimento. Se han elaborado los siguientes comentarios basándonos en la conlusión antes mencionada:

1.- Se ha demostrado que un modelo de estimación del tamaño basado en la ecuación de regresión, actúa significativamente mejor que la ecuación de longitud de Halstead para los programas Pascal. En este modelo del tamaño, aproximábamos el tamaño del programa en LOC como una función de VAR. Obteníamos los valores de parámetro del modelo usando datos independientes de los datos recogidos en el experimento EXPER-2.

2.- Nuestra suposición de que podíamos estimar VAR de manera adecuada en una etapa inicial, se ha confirmado por los resultados del experimento EXPER-2. Es decir, bajo la estrategia "primera

estructura de datos top-down" controlada con propiedad, la mayor parte de VAR (es decir, el 80%) se creará en una etapa inicial y entonces, el valor final de VAR se estimará de manera fácil.

Adicionalmente, los hallazgos siguientes ilustran la mejora de los resultados obtenidos en EXPER-2 sobre EXPER-1, debido a los controles experimentales mucho más rigurosos:

1.- Se mejoró la estimación inicial de VAR. El promedio de los ratios entre los valores iniciales y finales de VAR fueron 0,52 para EXPER-1, 0,79 para EXPER-2a, y 0,81 para EXPER-2b. Atribuimos esta mejora a un mejor control sobre los participantes durante el desarrollo del experimento.

2.- Se mejoraron las estimaciones subjetivas del tamaño. Esto puede atribuirse a nuestro mejor control del proceso de estimación.

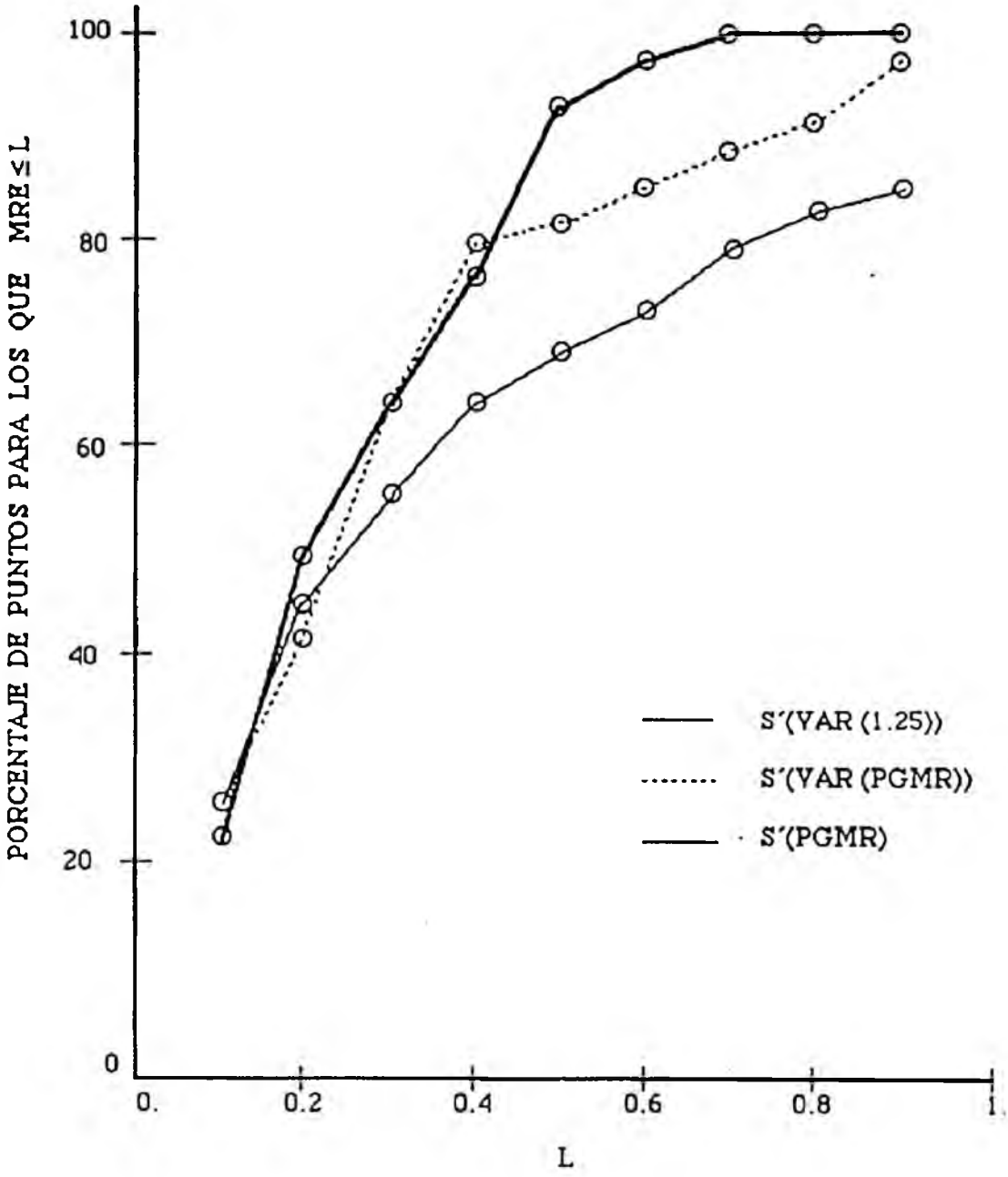


Figura 5.8

Estimación Inicial del Tamaño (EXPER-2b)

CAPITULO 6

MODELOS DE EVOLUCION DE LA METRICA

Tal y como se ha discutido en el Capítulo 4, usando las estrategias de desarrollo en el discutidas y aplicándolas con propiedad, cualquier métrica puede tener un patrón de evolución predecible. Por lo tanto el comportamiento de la evolución de VAR, n_2 y S debe tener patrones predecibles. Para validar esta premisa, examinamos el comportamiento de la evolución de las tres métricas con programas similares construidos por personas diferentes y usando identicas estrategias de desarrollo. En este capítulo se presentan varios patrones de evolución representativos de estas tres métricas. También mostramos los resultados de tipificar el comportamiento de la evolución de dichas métricas usando el modelo de curva discutido en el Capítulo 4.

6.1 EVOLUCION DEL TAMAÑO EN LOC

6.1.1 Hipótesis Relativa a la Evolución del Tamaño

Como se ha visto en la Sección 3.4.2, hemos supuesto que bajo la estrategia de desarrollo incremental, el tamaño del programa se incrementa linealmente con el tiempo de desarrollo.

Para comprobarlo usamos los datos del estudio EXPER-2, y examinanos la evolución del tamaño individual en LOC y los datos normalizados de todos los programas.

6.1.2 Evolución Individual del Tamaño

Los gráficos de evolución del tamaño de los programas que vamos a tratar en este capítulo se encuentran en los Apéndices A y B. Los gráficos de evolución del tamaño de EXPER-2 se encuentran clasificados en cuatro categorias de acuerdo con el criterio que ilustramos a continuación.

Consideremos el gráfico de evolución del tamaño mostrado en al figura 6.1. Primero, dibujamos una línea recta desde el origen P_0 al punto final P_n (donde n es es el número de puntos medidos). Cada punto observado P_i (i=1,2,...,n-1), tiene la forma:

$$P_i = (x_i, y_i)$$

donde x_i es el tiempo, e y_i es igual a S. Por cada punto observado analizamos su correspondiente punto "ajustado" P'_i en la línea recta $P_0 P_n$. En la Figura 6.1 los puntos observados se representan por "*" y los puntos fijados se representan por ".". Entonces, calculamos el error normalizado por mínimos cuadrados entre los puntos observados y los puntos ajustados:

$$rmse = \frac{\left[\dfrac{\sum_{i=1}^{n-1} (Y'_i - Y_i)^2}{n-1}\right]^{\frac{1}{2}}}{S_f} \qquad (6.1)$$

donde rmse es el error calculado para la S final (S_f, es decir, Y_n). Y_i e Y'_i son las coordenadas y, de P_i y P'_i, respectivamente. Debido a que la linea obtenida pasa siempre a través del punto final P_n, no se incluye Y_n en el cálculo del rmse. Un valor pequeño de rmse calculado utilizando este esquema, implica que todos los puntos medidos pueden aproximarse mediante la línea recta $P_0 P_n$ con una pequeña variación. Por lo tanto, podemos clasificar cada patrón de acuerdo con la estructura siguiente:

- Completamente lineal - rmse $\leq$ 0,05

  Todos los puntos de datos pueden representarse por la línea recta $P_0 P_n$ con un promedio de variación del 5% o inferior.

- Mayormente lineal - 0,05 < rmse $\leq$ 0,10

  Todos los puntos de datos pueden representarse por la línea recta $P_0 P_n$ con un promedio de variación del 10% o inferior.

- Lejanamente lineal - $10 < rmse \leq 0{,}15$

Todos los puntos de datos pueden representarse por la línea recta $P_0 P_n$ con un promedio de variación del 15% o inferior.

- No lineal - $15 < rmse$

Todos los puntos de datos muestran un patrón errático o un patrón que tiene más del 15% de variación sobre la línea recta $P_0 P_n$.

Este criterio de clasificación por categorías es muy conservador y puede que un análisis casual de los patrones llamados no lineales termine denominándoles lineales en mayor o menor medida.

Usando este criterio, clasificamos como patrones completamente lineales seis patrones de evolución del tamaño del experimento EXPER-2a y cinco del EXPER-2b. Dos ejemplos de estos patrones se muestran en la Figura A.1 para el EXPER-2a y en la Figura B.1 para EXPER-2b. Merece la pena hacer notar, que estos dos patrones los realizó un participante que aseguraba haber seguido estrictamente las estrategias de desarrollo y las reglas experimentales. La información relativa al modo en que los participantes siguieron las reglas experimentales se obtuvo a través de las "entrevistas de punto crucial" realizadas en cada punto crucial durante el experimento.

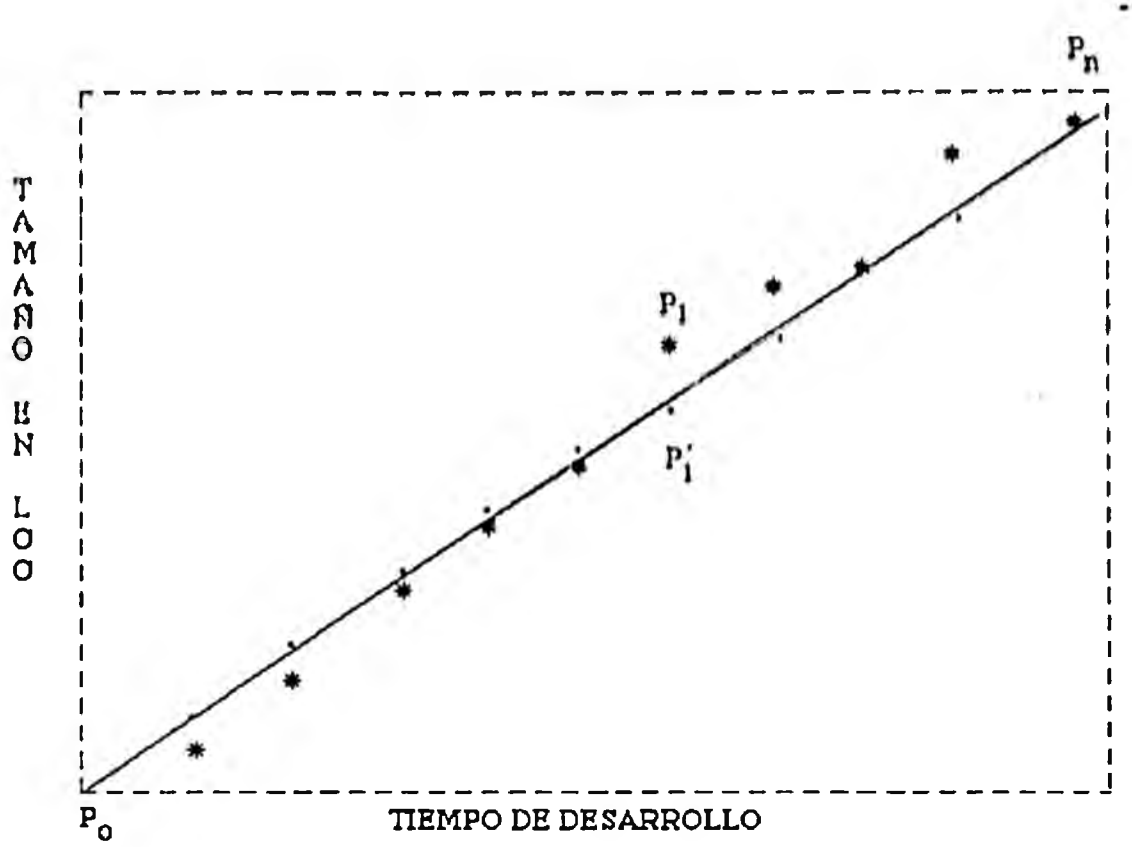


Figura 6.1

Clasificación de los Patrones de la Evolución Lineal

La Tabla 6.1 resume la distribución de los patrones de evolución del tamaño para los experimentos EXPER-2a y EXPER-2b usando nuestro criterio de clasificación. En la figura 4.2 se muestra un ejemplo de patrones de evolución del tamaño clasificados como mayormente lineales.

Tabla 6.1

Distribución de los Patrones de Evolución del Tamaño

Clasificación	EXPER-2a	EXPER-2b
Completamente lineal	6 (14%)	5 (11%)
Mayormente lineal	25 (56%)	16 (37%)
Lejanamente lineal	7 (16%)	15 (34%)
No lineal	6 (14%)	5 (18%)
Total	44 (100%)	44 (100%)

Ya hemos mencionado que el proceso de programación es una actividad intelectual compleja que envuelve muchos factores todavía no controlados. Está claro, que el procedimiento de recopilación de información instrumentado en este experimento está sujeto a errores de medición. En consecuencia, no se esperaba que bajo la estrategia incremental, la mayoría de los patrones de evolución del tamaño pudieran caer dentro de la categoria completamente lineal. En su lugar, se esperaba que la mayoría de los patrones pertenecieran a una de las tres primeras categorias: completamente lineal, mayormente lineal, lejanamente lineal, siendo este el caso, tal y como se puede ver en la Tabla

6.1. Aproximadamente el 86% (es decir, 38 de 44) de los patrones de evolución del tamaño de EXPER-2a pueden clasificarse en estas trés categorias. En EXPER-2b, este número es el 82% (es decir, 36 de 44). Un análisis de los datos de la Tabla 6.1 revela que la distribución de los patrones es diferente de manera significativa ($\alpha < 0,05$) de una distribución aleatoria. Estos resultados sugieren que bajo la estrategia de desarrollo incremental el tamaño del programa crece linealmente con el tiempo de desarrollo.

6.1.3 Evolución de la Métrica como Representación del Proceso de Desarrollo

Como se ha mencionado en el Capítulo 2, la evolución de la métrica puede ser una herramienta útil para representar el proceso de desarrollo. Ya que el gráfico de evolución de una métrica se puede ver como una representación gráfica del proceso de desarrollo, éste constituye una huella característica de dicho proceso. De hecho, podemos considerar la evolución de la métrica como un vehiculo sencillo que resume el proceso de desarrollo del programa de modo que los programadores, analistas y directores, puedan utilizarla.

Las Figuras A.3, A.4, A.5, y A.6, contienen cuatro casos excepcionales de la evolución del tamaño en este experimento. Las Figuras A.3 y A.4 ilustran la situación en donde ha aparecido la variación mayor durante el proceso. La Figura A.5 ilustra la

situación en la que el participante no siguió la estrategia de desarrollo "incremental" porque no la entendió completamente. Un participante que no tenía mucha experiencia en programación, originó la Figura A.6. En los cuatro casos un monitor del experimento podía haber utilizado las representaciones para determinar una situación inusual que requería atención. Además descubrimos que un par de participantes no habia seguido la estrategia incremental, observando los patrones de evolución del tamaño de sus programas. Esta evidencia sugiere que el grafico de evolución del tamaño del programa puede usarse como una herramienta de control para observar que está ocurriendo (o que pasó) durante el proceso de desarrollo.

6.1.4 Evolución Normalizada del Tamaño

Al hablar de normalizar, estamos refiriendonos a la forma de representar gráficamente los datos obtenidos a partir de nuestros experimentos. En las representaciones gráficas que realizamos, los ejes de coordenadas van desde el 0% al 100%, al proceso de conversión de las magnitudes medidas en un tanto por ciento, entre los limites indicados, lo denominamos normalizar.

Mezclamos y normalizamos los puntos de información obtenidos a partir de los programas que forman EXPER-2a, y hacemos lo mismo con los que forman EXPER-2b. El eje vertical representa el tamaño normalizado del programa y el eje horizontal representa el tiempo de desarrollo normalizado. La normalización del tiempo y del tamaño hace que los programas se comparen en diferentes rangos de

tamaño y tiempo de desarrollo. Aunque los datos contienen una dispersión considerable debido a las diferencias individuales, la tendencia general de la evolución del tamaño aparece de modo bastante cercano a la lineal (especialmente para EXPER-2a).

6.1.5 Obtención de un Modelo de la Evolución del Tamaño

Una vez observada la evolución del tamaño individual, así como la normalizada para EXPER-2, quisimos hacer un modelo de este comportamiento de manera que pudieramos aplicar este conocimiento de manera más sistemática para el uso práctico. Como se discutió en el Capítulo 4, para realizar un modelo de la evolución lineal, se puede usar una forma funcional simple del tiempo de desarrollo:

$$y(t) = \alpha \times t^{\Gamma}, \text{ donde } \Gamma = \beta_k \qquad (6.2)$$

La pregunta crítica es como elegir un valor apropiado para el parametro de forma, β_k.

Como ya hemos apuntado, la forma de una curva del tipo de la reflejada en la ecuación (6.2), queda completamente determinada por el valor de β_k. Esto es, mientras que el valor de β_k no cambie, la forma de la curva permanecerá constante, independientemente de la escala vertical u horizontal. Esto implica que si el modelo de ecuación (6.2) es la base para crear

un modelo del comportamiento de la evolución, podemos usar el diagrama de evolución normalizada (en lugar del diagrama de evolución individual) para encontrar un valor optimo de β_k. Nuestro método para encontrar un valor optimo de β_k se basa en los errores rmse. Para un diagrama de evolución normalizado, podemos ajustar varias curvas usando diferentes valores de β_k. Puesto que cada métrica considerada en este estudio evolucionará desde cero hasta su valor final, forzamos a todas las curvas en el diagrama normalizado, a pasar por el origen y el punto final. Estos dos puntos se corresponden con la esquina inferior izquierda y con la esquina superior derecha del diagrama. Para cada curva realizada de esta manera, calculamos el error rmse como se ha definido en la sección 6.1.2. Consideramos que el valor de β_k que origina el menor valor de rmse, es el valor optimo de β_k, siendo este el valor que mejor caracteriza la evolución de una métrica dada, en terminos del criterio del error rmse.

Sabemos que la curva mejor debe ser cercana a la lineal, por lo tanto el valor óptimo de β_k debe estar cercano a 1. Entonces, utilizando el método arriba descrito, los valores de la curva con β_k variando desde 0,7 a 1,3 con un incremento del 0,01 se ajustan al diagrama de la evolución del tamaño normalizado, y se calculan los valores rmse para β_k =0,70, 0,71, 0,72, ... 1,30. Se obtienen los valores optimos:

	EXPER-2a	EXPER-2b
β_k	0,94	0,96
rmse	0,11	0,13

Este resultado nos indica para los programas de EXPER-2a la curva que mejor modela la evolución del tamaño es:

$$S(t) = \alpha \times t^{0,94} \qquad (6.3)$$

y para los programas de EXPER-2b las curva que mejor modela la evolución mencionada es:

$$S(t) = \alpha \times t^{0,96} \qquad (6.4)$$

Ya que 0,94 y 0,96 estan muy cercanos a 1,00, estos hallazgos sugieren que bajo la estrategia de desarrollo incremental el tamaño del programa evoluciona de manera lineal con respecto al tiempo de desarrollo.

6.2 EVOLUCION DE VAR

6.2.1 Hipótesis Relativa a la Evolución de VAR

Tal y como se ha discutido en el Capítulo 4, hemos supuesto que bajo la estrategia de desarrollo "primera estructura de datos top-down", la métrica VAR tiene una evolución convexa. Es decir, en una etapa inicial, debe alcanzar rápidamente un valor cercano a su valor final. Como en las secciones previas, vamos a examinar la evolución de los datos normalizada e individual de VAR para validar nuestra hipótesis.

6.2.2 Evolución Individual de VAR

Los diagramas de evolución de VAR mencionados en este capítulo se encuentran en el Apendice C. La mayoría de estos patrones de evolución parecen ser convexos. Dos ejemplos se muestran en las Figuras C.1 y C.2. Dos casos excepcionales , que merecen la pena mencionarse, se muestran en la Figura C.3 (donde el participante no siguió las reglas experimentales) y en la Figura C.4 (donde el participante realizó un cambio drástico en una etapa tardía que ocasionó una caída repentina en la cuenta de VAR entre el noveno y decimo punto crucial).

6.2.3 Evolución Normalizada de VAR

Los diagramas de evolución de VAR normalizados muestran que la tendencia general de la evolución de VAR es convexa, aunque los datos todavía contienen una dispersión considerable.

6.2.4 Obtención de un Modelo de la Evolución de VAR

Usando el método de obtención de modelos introducido en la Sección 6.1, hemos observado que una curva convexa de la forma:

$$VAR(t) = \alpha \times t^{0,19} \qquad (6.5)$$

modela la evolución de VAR con un rmse igual a 0,11 para el estudio EXPER-2a, o de la forma:

$$VAR(t) = \alpha \times t^{0,20} \qquad (6.6)$$

con un rmse igual a 0,21 para el estudio EXPER-2b.

Por lo tanto, podemos decir que bajo la estrategia de desarrollo "primera estructura de datos top-down", la métrica VAR evoluciona de una manera convexa.

6.3 EVOLUCION DE n_2

6.3.1 Hipótesis Relativa la Evolución de n_2

Como hemos visto en el Capítulo 4, suponemos que la métrica n_2 tiene un patrón convexo bajo la estrategia de desarrollo "primera estructura de datos top-down". Esto es, deberá alcanzar rápidamente un valor muy próximo a su valor final en una etapa inicial.

Los diagramas de evolución de n_2 de los programas del estudio EXPER-2 mencionados en este capítulo se encuentran en el Apéndice D. La mayoría de los patrones de evolución de n_2 parecen ser convexos. Las figuras D.1 y D.2 muestran dos ejemplos.

6.3.2 Obtención de un Modelo de Evolución de n_2

Utilizando el método de creación de modelos introducido en la Sección 6.1.6, nos dimos cuenta que una curva convexa nos permitía modelar mejor la evolución de n_2. Una curva de ecuación:

$$n_2(t) = \alpha \times t^{0,38} \qquad (6.7)$$

con rmse igual a 0,11 para el estudio EXPER-2a, o

$$n_2(t) = \alpha \times t^{0,68} \qquad (6.8)$$

con rmse igual a 0,16 para el estudio EXPER-2b. Los resultados obtenidos nos sugieren que bajo la estrategia de desarrollo "primera estructura de datos top-down" la evolución de n_2 es de algún modo convexo, pero no parece ser tan convexo como habíamos esperado. Adviértase, también, que hay mayor variabilidad entre los valores de EXPER-2a y EXPER-2b (es decir, 0,38 contra 0,68) que la encontrada con S o con VAR.

6.4 EVOLUCION DE LA METRICA

En las secciones precedentes, hemos observado que se puede modelar la evolución de VAR, de n_2, y de S utilizando curvas de la forma:

$$y(t) = \alpha \times t^{\beta} \qquad (6.9)$$

con valores apropiados de β para las diferentes métricas y conjuntos de datos. Los valores de β derivados de la evolución de los datos utilizados en este estudio se resumen en la Tabla 6.2.

Tabla 6.2

Resumen de los Valores β para la Evolución de la Métrica

Conjunto	Participantes	β_{VAR}	β_{S}	β_{n_2}
EXPER-2a	44	0,19	0,94	0,38
EXPER-2b	44	0,20	0,96	0,68

Las siguientes observaciones sumarizan los resultados de este análisis:

(1) Los valores de β_{VAR} para el EXPER-2a y el EXPER-2b son casi iguales, aproximadamente 0,20. Esto sugiere que si se sigue de forma apropiada la estrategia "primera estructura de datos top-down" durante el desarrollo del programa, el comportamiento medio de la evolución será convexo.

(2) Los valores de β_{S} para el EXPER-2a y el EXPER-2b son casi iguales, aproximadamente 0,95. Esto implica que si se sigue de forma adecuada la estrategia incremental durante el desarrollo del programa, el comportamiento medio de la evolución de S será lineal.

(3) Para el mismo conjunto de datos, mencionado en las dos observaciones anteriores, β_{VAR} es menor que β_{n2}. Esto nos sugiere que el comportamiento medio de la evolución de VAR, bajo la estrategia de desarrollo "primera estructura de datos top-down", será más convexo que el de n_2. Este resultado confirma nuestra especulación mencionada en el Capítulo 3, de que la mayor parte de VAR se crea

inicialmente usando la estrategia "primera estructura de datos top-down", y la porción restante de n_2 se añadirá después de la etapa de diseño.

(4) Los valores de β_{n2} para el estudio EXPER-2a y el estudio EXPER-2b son drásticamente diferentes (0,38 contra 0,68). Esto significa que el comportamiento medio de la evolución de n_2 es menos convexo para el estudio EXPER-2b que para el EXPER-2a. Una razón de esto es debida a que VAR constituye una pequeña parte de n_2 en EXPER-2b. Por termino medio, el valor de VAR suponía solamente un 36% de n_2 para el EXPER-2b. Para el EXPER-2a suponía un 57%. Así, aunque la mayor parte de VAR se creó en una etapa inicial para el EXPER-2b sólo una porción relativamente pequeña del n_2 se creó al mismo tiempo. Este resultado sugiere que el comportamiento medio de la evolución de una métrica tal como n_2 dependerá tanto de la estrategia de desarrollo como del tipo de programa.

(5) Uniendo las dos primeras observaciones, vemos que que el comportamiento medio de la evolución de una métrica (tal como VAR o S) dependerá de modo significativo de la estrategia de desarrollo que se siga.

Resumiendo, basándonos en el comportamiento de la evolución de VAR, n_2, y S, que hemos observado en el estudio EXPER-2, concluimos que la evolución de la métrica no es un proceso aleatorio. Los factores que contribuyen a la variación de este proceso incluyen diferencias individuales, estrategias de desarrollo, y tipos de programas. Para las dos métricas S y VAR, el comportamiento medio de la evolución depende de la estrategia de desarrollo que se utiliza. Para la métrica n_2 el comportamiento medio de la evolución depende tanto de la estrategia de desarrollo como del tipo de programa.

CAPITULO 7

ESTIMACION INICIAL DEL TAMAÑO Y DEL ESFUERZO

En el Capitulo 4 hemos presentado un modelo de estimación del tamaño y del esfuerzo basado en la evolución de la métrica. En este capítulo evaluaremos este modelo en términos de su capacidad para predecir el tamaño real del programa y el tiempo de desarrollo real. Comenzamos describiendo el procedimiento de estimación definido.

7.1 PROCEDIMIENTO DE ESTIMACION

Como se ha visto anteriormente, nuestro modelo de estimación está basado en la evolución de una métrica del tamaño, S o N, y una métrica relacionada del tamaño, VAR o n_2. Recordemos que S es el tamaño del programa en LOC, N es el número de apariciones de operadores y operandos, VAR es el número de variables únicas, y n_2 es el número de operandos únicos. En el Capitulo 5 se demostró que VAR es la métrica del tamaño relacionada más consistente y fiable con respecto a la métrica del tamaño S. En consecuencia, en este capítulo solamente presentamos los resultados obtenidos basándonos en S y VAR. Partimos del hecho de que la evolución de S se puede ver a través de una curva de ecuación:

$$S(t) = \alpha \times t^{\Gamma}, \text{ donde } \Gamma = \beta_S \qquad (7.1)$$

y la evolución de VAR se puede ver mediante una curva de ecuación:

$$S(\iota) = \alpha \times t^{\sigma}, \text{ donde } \sigma = \beta_{VAR} \qquad (7.2)$$

en donde β_S y β_{VAR} son 1,00 y 0,20 respectivamente. Además, suponemos que tenemos a nuestra disposición un modelo del tamaño apropiado para convertir VAR en S. Un ejemplo de tal modelo del tamaño es la ecuación de regresión (5.5) que repetimos a continuación:

$$S'(VAR) = 102 + 5{,}31\ VAR \qquad (7.3)$$

Este procedimiento de estimación se ilustra en la Figura 7.1. Para un programa dado, tomamos medidas iniciales de S y VAR en dos puntos iniciales P_1 y P_2. Al enfrentar las dos medidas de S al tiempo de desarrollo obtenemos los puntos S_1 y S_2. Convertimos los valores medidos de VAR en S, usando la ecuación (7.3). Al enfrentar los dos valores convertidos en S al tiempo de desarrollo, obtenemos los puntos V_1 y V_2. Partiendo de la definición de estas notaciones, podemos describir el procedimiento de estimación en tres pasos:

1.- Fijaremos el modelo de evolución S (Ecuación (7.1)) para los puntos iniciales ("ajuste de curva inicial"). La curva generada (línea continua) indica el comportamiento de evolución de la S predicha.

2.- De manera similar, fijaremos el modelo de evolución de VAR (Ecuación (7.2)) para los puntos iniciales V_1 y V_2. La curva generada (línea discontinua) representa el comportamiento de la evolución predicho de la S obtenida a partir de VAR.

3.- Las dos curvas generadas se interceptan en el punto Q. La coordenada vertical de Q indica la predicción del tamaño, llamada S'(VAR), y la coordenada horizontal corresponde a la predicción del esfuerzo, designada E'(VAR).

El valor de S'(VAR) obtenido en el tercer paso es el modelo estimado del tamaño. El valor de E'(VAR) obtenido en el tercer paso es el modelo estimado del esfuerzo. Posteriormente los compararemos con el tamaño real del programa y su tiempo de desarrollo real.

Partiendo del procedimiento de estimación arriba indicado, puede parecer que al aplicarlo aparecen tres fuentes de error potenciales:

1.- Errores en la obtención de la curva, introducidos en el primer paso al predecir el comportamiento de evolución de S.

2.- Errores en la estimación del tamaño, introducidos en el segundo paso al convertir VAR en S.

3.- Errores en la obtención de la curva, introducidos en el segundo paso, al predecir el comportamiento de la evolución de la S obtenida a partir de VAR.

Debe tenerse en cuenta que los errores que estas tres fuentes originan pueden reforzarse o cancelarse unos a otros con respecto a los errores finales de estimación para el tamaño y el esfuerzo. Como se ha mencionado en la sección 4.5, este método de estimación asume la existencia de un modelo del tamaño preciso para convertir VAR en S. De tal manera, que si el modelo del tamaño usado para convertir VAR en S produce errores, dichos errores confundiran los resultados de la estimación obtenidos usando este procedimiento. Por lo tanto, evaluaremos nuestro modelo de estimación evitando que los errores causados por el modelo del tamaño nos confundan. Es decir, asumimos que tenemos a nuestra disposición un modelo del tamaño "perfecto", de tal manera que se pueda convertir VAR en S de modo preciso.

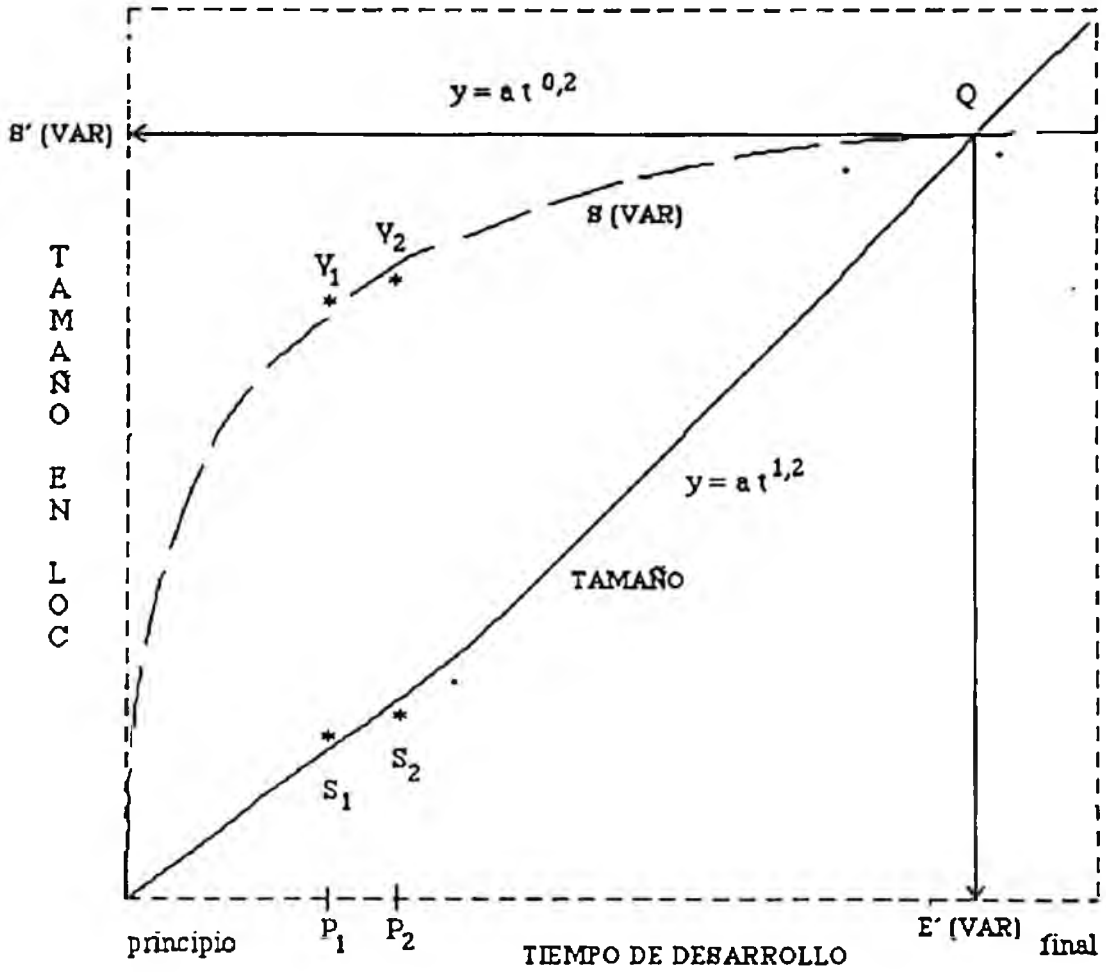


Figura 7.1

Estimación Inicial del Esfuerzo y del Tamaño

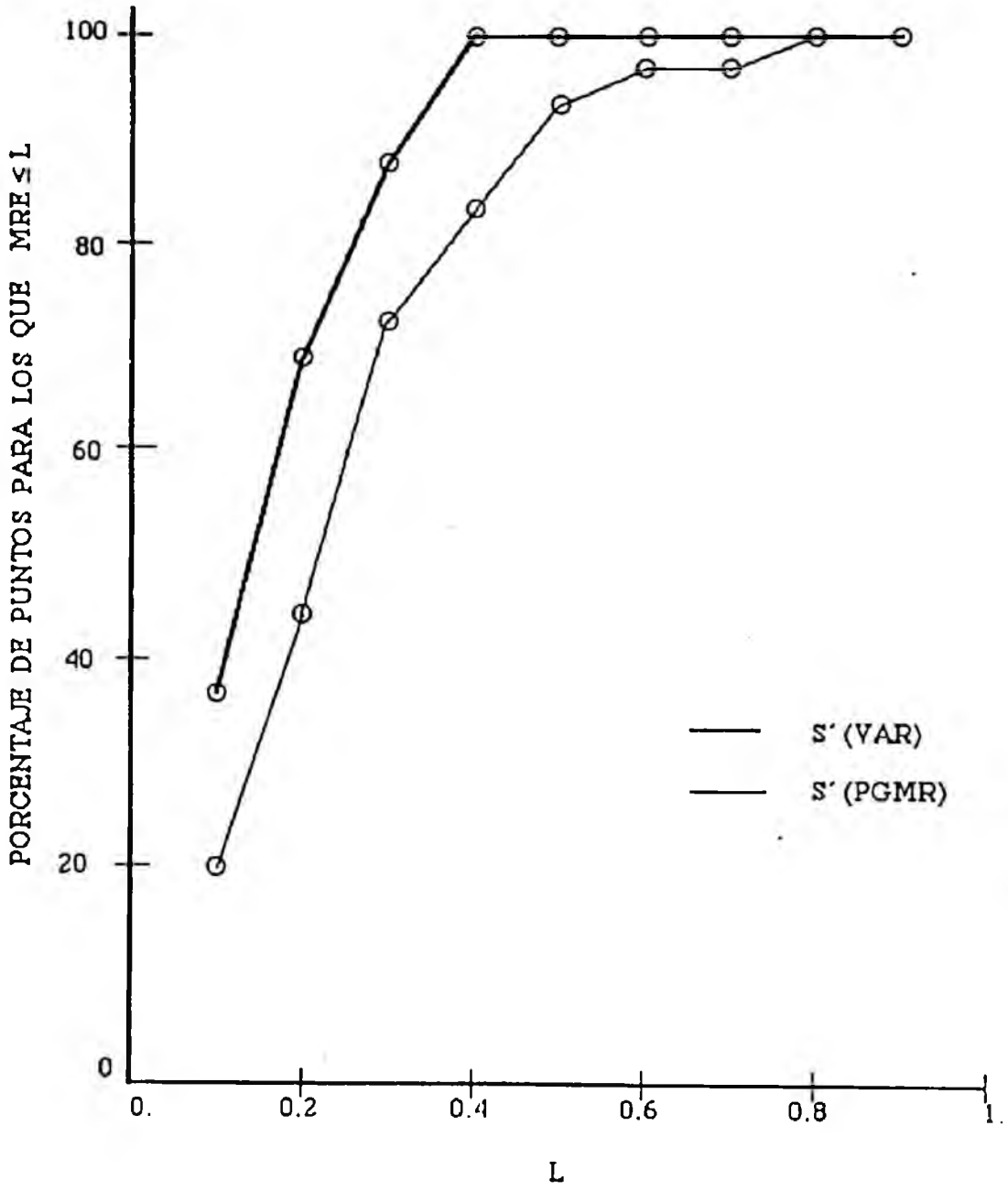


Figura 7.2

Estimación del Tamaño en base al Modelo Perfecto del Tamaño

(EXPER-2a)

En este trabajo, para obtener un modelo del tamaño perfecto, realizamos el siguiente cálculo: multiplicamos los valores de evolución del contador de VAR, por el cociente obtenido de dividir, el valor final de S entre el valor final de VAR. De esta manera, el valor final de S será igual que el valor final de la métrica S obtenida a partir de VAR y por lo tanto, tenemos un modelo del tamaño perfecto.

7.2 ESTIMACION USANDO UN MODELO PERFECTO DEL TAMAÑO

En esta sección, evaluaremos nuestro modelo de estimación bajo la premisa de que se trata de un modelo del tamaño perfecto. Para ello, utilizamos el primer grupo de datos recogido en el estudio EXPER-2. Bajo esta suposición, necesitamos dar valores apropiados a β_S y β_{VAR} para poder aplicar nuestro procedimiento de estimación. Como se ha discutido en la Sección 4.7.2, estos valores del parámetro se pueden derivar de datos históricos en los que se hayan utilizado las mismas estrategias de desarrollo, Puesto que no teníamos tales datos para el estudio EXPER-2, utilizamos como punto de partida los valores de β que caracterizan el comportamiento medio de la evolución de S y VAR en dicho estudio.

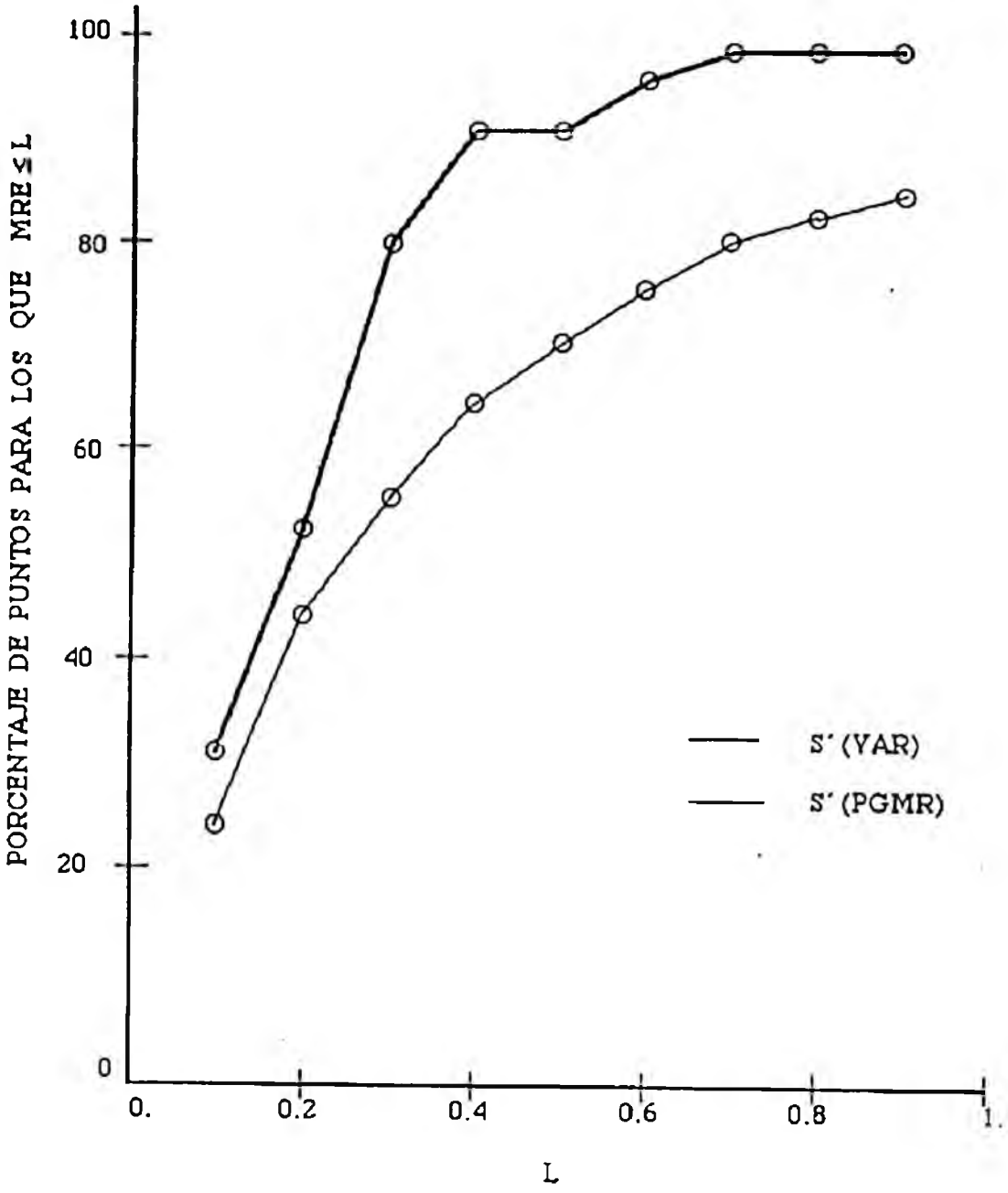


Figura 7.3

Estimación del Tamaño en base al Modelo Perfecto del Tamaño

(EXPER-2b)

7.2.1 Estimación Usando los Valores Iniciales de β_S y β_{VAR}

En los análisis previos realizados sobre los datos de la evolución de la métrica se ha observado que la ecuación:

$$VAR(t) = \alpha \times t^{\Gamma}, \text{ donde } \Gamma = \beta_{VAR} \qquad (7.4)$$

describe de manera apropiada el comportamiento medio de la evolución de VAR, para β_{VAR} igual a 0,19, en EXPER-2a y β_{VAR} igual a 0,20, para EXPER-2b.

También hemos observado que la ecuación:

$$S(t) = \alpha \times t^{\Gamma}, \text{ donde } \Gamma = \beta_S \qquad (7.5)$$

describe perfectamente el comportamiento medio de la Evolución de S, con β_S igual a 0,94 para EXPER-2a, y β_S igual a 0,96 para EXPER-2b. Así, usando las ecuaciones (7.4) y (7.5), se aplicó el procedimiento de estimación descrito en la sección precedente a los datos de la evolución de la métrica inicial recogidos en EXPER-2a y EXPER-2b. De la misma manera se generaron las estimaciones del modelo del tamaño y del esfuerzo.

La Tabla 7.1 resume los resultados de la estimación del tamaño y del esfuerzo. Como ya hemos mencionado, las estimaciones del modelo del tamaño y del esfuerzo se denotan como S'(VAR), y E'(VAR). Los resultados de la estimación del tamaño y del

esfuerzo realizada por los participantes al final de la etapa de diseño y designadas como S'(PGMR) y E'(PGMR), se incluyen también en las tablas por motivos de comparación.

Como se muestra en la Tabla 7.1, el resultado del tamaño estimado mediante el modelo S'(VAR) para ambos grupos de datos parece ser bastante satisfactorio: el 77% de las estimaciones del EXPER-2a y el 66% de las estimaciones del EXPER-2b están dentro del 25% de los valores reales. Ambas son mejores que las estimaciones del tamaño subjetivas (las realizadas por los participantes al final de la etapa de diseño).

Tabla 7.1

Estimación del Tamaño y del Esfuerzo

Usando el Modelo Perfecto del Tamaño

Programa	β VAR	β S	Estimación	MRE'	PRED(0,25)	RE'
1	0,19	0,94	S'(PGMR)	0,23	57%	+0,02
			S'(VAR)	0,16	77%	-0,02
			E'(PGMR)	0,42	57%	-0,35
			E'(VAR)	0,51	41%	-0,42
2	0,20	0,96	E'(PGMR)	0,42	52%	-0,39
			S'(VAR)	0,27	66%	-0,08
			E'(PGMR)	0,86	9%	-0,85
			E'(VAR)	0,68	30%	-0,52

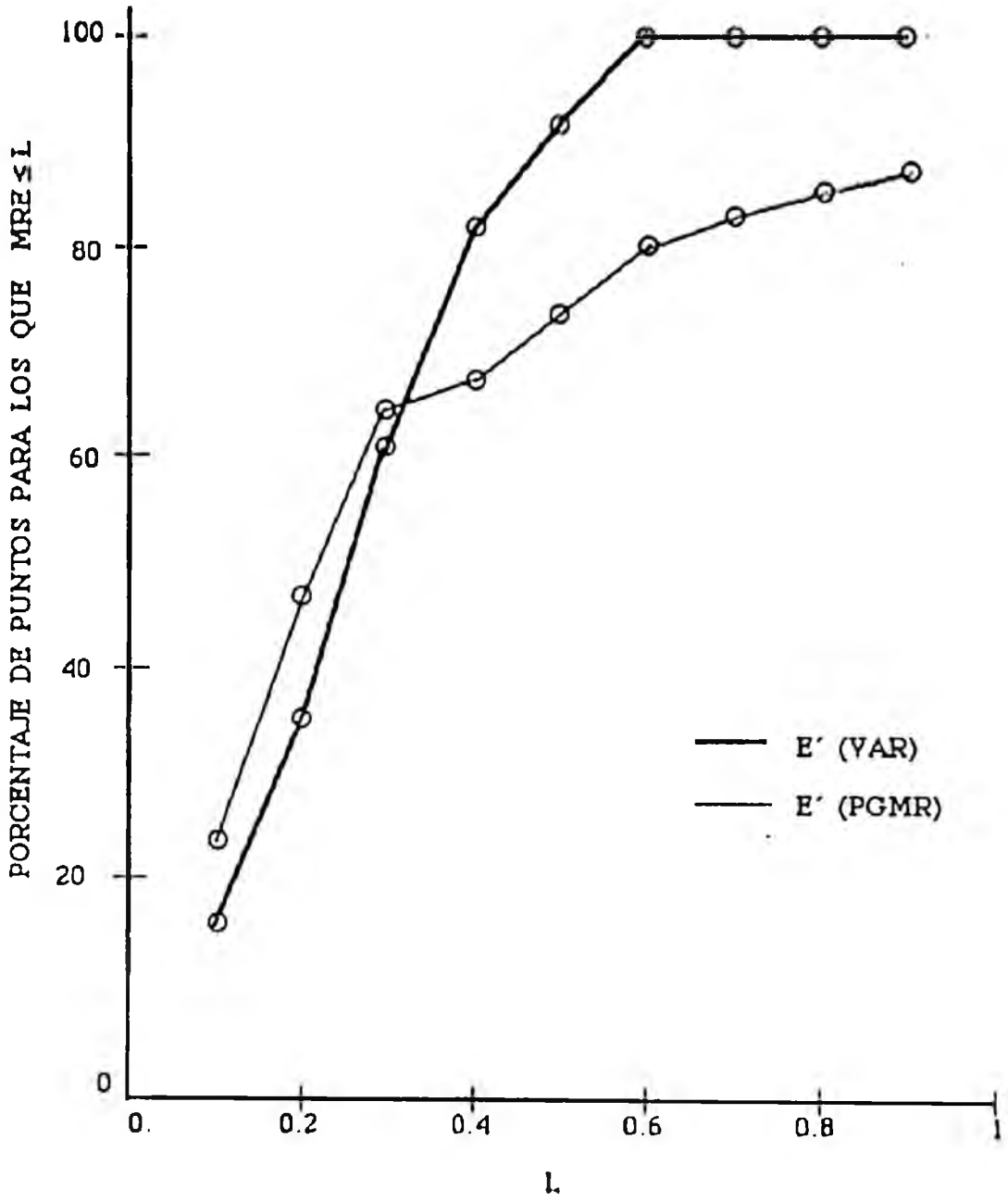


Figura 7.4

Estimación del Esfuerzo en base al Modelo Perfecto del Tamaño

(EXPER-2a)

La Tabla 7.1 también nos muestra que las estimaciónes del esfuerzo subjetivas, E'(PGMR), realizadas por los propios participantes en el experimento EXPER-2a sobrestimaron el esfuerzo real en un porcentaje medio del 35%. Analizando las entrevistas post-experimento, nos dimos cuenta de que una razón para este resultado tan sorprendente fue que la mayoría de los participantes en EXPER-2a se sintieron más seguros sobrestimando el esfuerzo que subestimándolo (estrategia de estimación "trabaja sobre seguro"). Ellos pensaron que hubiera sido mucho peor subestimar el esfuerzo. Este fenómeno de la sobrestimación del esfuerzo fue todavía más evidente en EXPER-2b en donde las estimaciones del esfuerzo subjetivas sobrestimaban de manera consistente el esfuerzo real en un 85%, tal y como se indica en la Tabla 7.1. Además del uso de la estrategia de estimación "trabaja sobre seguro", otra razón por la que los participantes sobrestimaron de modo drástico el esfuerzo real fue que cuando leyeron por primera vez las especificaciones del programa traductor para un lenguaje de bases de datos (EXPER-2b), les pareció un programa muy difícil de realizar. No se dieron cuenta de que el programa no era tan difícil como esperaban hasta que no se hallaron en la mitad del proceso de construcción. Este "error de cálculo" se reflejó también en la sobrestimación del tamaño del programa del EXPER-2b, tal y como se muestra en la Tabla 7.1.

Basándonos en estos hechos, podemos decir que las estimaciones del esfuerzo subjetivas en EXPER-2a y EXPER-2b, y las estimaciones del tamaño subjetivas en EXPER-2b no son

realmente destacables. Esto confirma nuestro concepto de que la estimación del tamaño y del esfuerzo es, de hecho, un problema difícil de resolver incluso para programas de pequeño tamaño.

Como mostramos en la Tabla 7.1, los resultados del esfuerzo estimados por modelo, E'(VAR) no parecen satisfactorios. De todos modos, E'(VAR) no era mucho peor que E'(PGMR) en EXPER-2a, y E'(VAR) era un poco mejor que E'(PGMR) en EXPER-2b.

7.2.2 Ajuste de los Valores de β_S y β_{VAR} a una Mejor Predicción

Tal y como se ha descrito con anterioridad, los resultados de la estimación del tamaño y del esfuerzo usando un modelo del tamaño perfecto dependen de dos parámetros: β_S y β_{VAR}. Nuestra observación de que los resultados de la estimación del tamaño eran satisfactorios, implica que los valores de β_{VAR} (0,19 y 0,20) usados en el procedimiento de estimación eran bastante apropiados.

Por otra parte, realizamos un anàlisis de sensibilidad para ver como cambian los resultados de la estimación del esfuerzo cuando se usan valores diferentes de β_S.

El propósito de un análisis de sensibilidad es estudiar el efecto de los cambios realizados sobre los valores de parámetro del modelo en los resultados del mismo. Es decir, investigar cuan sensibles son los resultados del modelo con respecto a las modificaciones en los valores del parámetro.

Debe quedar claro que para un conjunto dado de datos y un valor de β_{VAR} dado, los resultados de la estimación del esfuerzo usando nuestro procedimiento de estimación solamente dependerán del valor de β_S usado para el ajuste de curva inicial. Así, fijamos β_{VAR} en 0,19. Después incrementamos β_S desde 0,50 hasta 1,5 con un incremento de 0,10. Esta gama de valores tiene un valor medio aproximado de 0,94.

Tabla 7.2

Análisis de Sensibilidad de β_S con β_{VAR} = 0,19
(EXPER-2a)

β_S	MRE': S'(VAR)	MRE': E'(VAR)
0,5	0,72	26,22
0,6	0,40	7,7
0,7	0,26	2,7
0,8	0,19	1,3
0,9	0,17	0,65
1,0	0,15	0,38
1,1	0,15	0,28
1,2	0,15	0,26
1,3	0,15	0,28
1,4	0,15	0,32
1,5	0,16	0,37

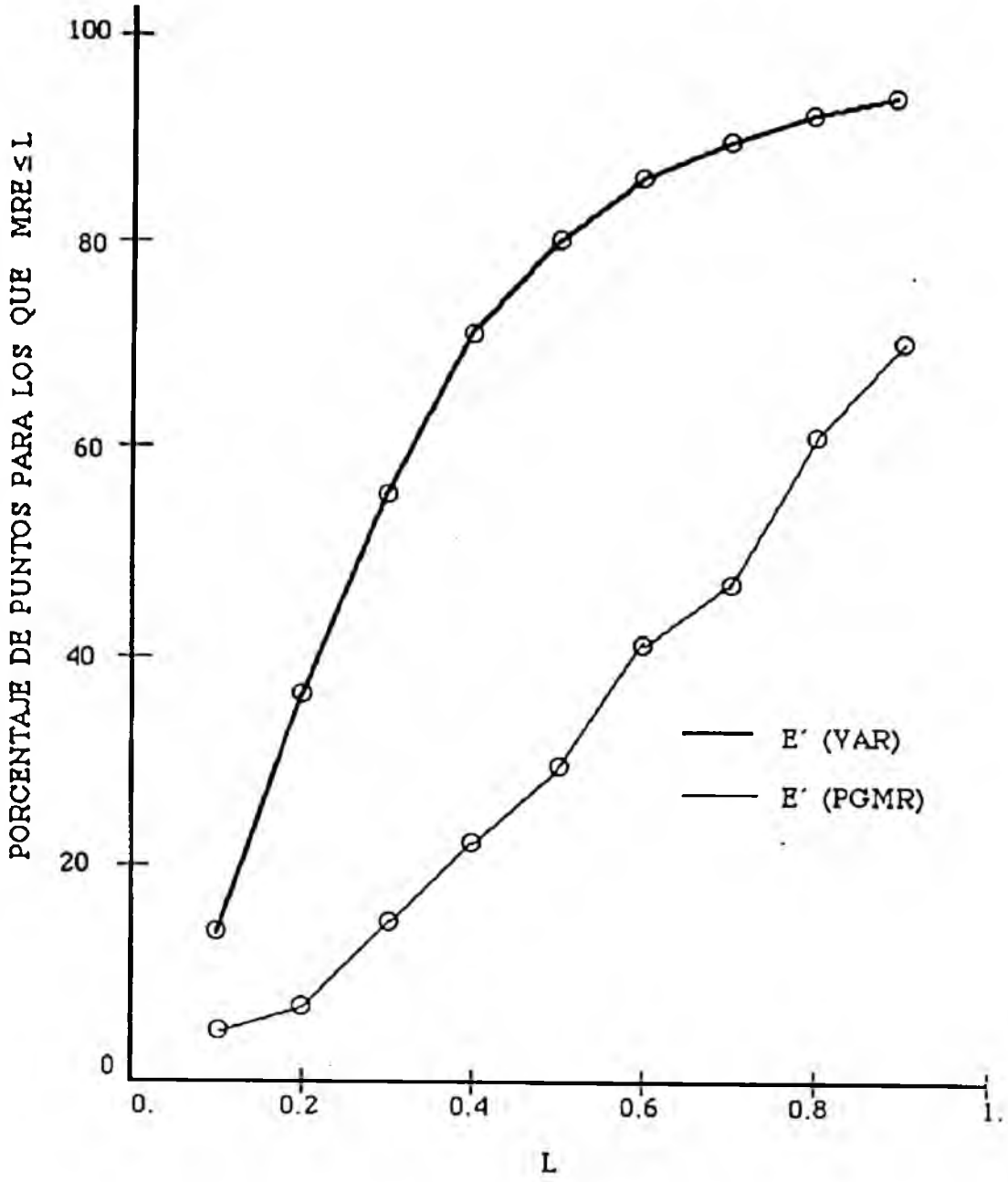


Figura 7.5

Estimación del Esfuerzo en base al Modelo Perfecto del Tamaño

(EXPER-2b)

Los resultados de las estimaciones del tamaño y del esfuerzo para EXPER-2a derivadas de la manera presentada, se resumen en la Tabla 7.2. Solamente representamos los resultados del análisis de sensibilidad de EXPER-2a, ya que son prácticamente idénticos a los de EXPER-2b. Elegimos a MRE' como indicador del éxito, ya que puede representar el éxito de una estimación de manera adecuada.

Como se muestra en la segunda columna de la Tabla 7.2, cuando se incrementa el valor de β_S desde 0,5 hasta 1,5, el MRE' de la estimación del tamaño baja desde 0,72 hasta 0,17 y después permanece dentro del rango comprendido entre 0,15 y 0,17. Por lo tanto los resultados de la estimación del tamaño no fueron sensibles a β_S entre los valores 0,9 y 1,5 cuando β_{VAR} se mantuvo en 0,19.

Por otra parte, como se muestra en la tercera columna de la Tabla 7.2, cuando crece el valor de β_S desde 0,5 a 1,5, el MRE' de la estimación del esfuerzo disminuye desde 26,22 hasta 0,26, y después aumenta hasta 0,37 con el valor mínimo cuando β_S es igual a 1,2. Esto significa que, en el estudio EXPER-2a, la curva que mejor describe el comportamiento medio de la evolución de S, es aquella para la que β_S es igual a 0,94, sin embargo, el valor óptimo de β_S que podemos usar para la estimación inicial del esfuerzo es más grande que 0,94 (es decir, igual a 1,2).

Las razones que nos llevan a estos resultados son:

1.- La curva con β_S igual a 0,94, solamente representa el comportamiento medio de la evolución de S en EXPER-2a. Para algunos programas, valores grandes de β_S eran más apropiados.

2.- Aunque había solamente unos pocos programas en EXPER-2a para los cuales los valores mejor fijados de β_S estaban cercanos a 0,94, el esfuerzo de estimación por modelo para estos programas sobrestimaba de manera considerable el esfuerzo real. Esto es debido a que S para estos programas evolucionaba más lentamente en el proceso de desarrollo.

Basándonos en nuestros análisis, concluimos subrayando el hecho de que necesitamos un valor de β_S mayor que 0,94 para fijar la curva inicial de la evolución de S. Partiendo de nuestro análisis de sensibilidad, el mejor valor parece ser 1,2.

Para ver como los resultados de la estimación del tamaño y del esfuerzo varían tanto con los valores de β_S como con los de β_{VAR}, realizamos también un análisis de sensibilidad de ambos. Las siguientes observaciones resumen los resultados de este análisis.

1.- Los resultados de la estimación del tamaño no eran sensibles a cambios en el valor de β_S. Ya que β_{VAR} se mantuvo entre 0,15 y 0,20, mientras que β_S varió desde 0,9 hasta 1,5 (MRE' $\leq$ 0,18)

2.- Para la estimación del esfuerzo, cuando β_{VAR} estaba entre 0,15 y 0,20, y β_S estaba entre 1,1 y 1,2, el valor de MRE'era menor o igual que 0,28.

3.- La unión de los puntos (1) y (2) indica que cuando β_{VAR} estaba entre 0,15 y 0,20, y β_S estaba entre 1,1 y 1,2, el valor de MRE' es menor o igual a 0,16 para las estimaciones del tamaño, y es menor o igual a 0,28 para las estimaciones del esfuerzo. En base a lo expuesto, seleccionamos los valores 0,20 y 1,2 para β_{VAR} y β_S respectivamente para nuestros análisis posteriores.

7.2.3 Estimación Usando los Valores Ajustados de β_{VAR} y β_S

Como hemos determinado los valores apropiados para β_{VAR} y β_S, aplicamos de nuevo el procedimiento de estimación a los datos del EXPER-2. En la Tabla 7.3 mostramos los resultados de la estimación del tamaño mediante el modelo, con β_{VAR} =0,20 y β_S =1,20, usando la ecuación del tamaño perfecto. Tanto para EXPER-2a como para EXPER-2b, las estimaciones del tamaño mediante el modelo fueron mejores para $\alpha < 0,10$, que las estimaciones del tamaño subjetivas. Los perfiles MRE para los dos grupos de estimaciones del tamaño se muestran en la figura 7.2, para EXPER-2a, y en la Figura 7.3, para EXPER-2b.

Los resultados de la estimación para el modelo de estimación del esfuerzo también se muestran en la Tabla 7.3. Tanto para EXPER-2a como para EXPER-2b, las estimaciones del modelo del esfuerzo fueron mejores para $\alpha < 0,10$, que las estimaciones subjetivas del esfuerzo. Los perfiles MRE para estos dos grupos de estimaciones del esfuerzo se muestran en las Figuras 7.4 para EXPER-2a y 7.5 para EXPER-2b.

Tabla 7.3

Estimación del Tamaño y del Esfuerzo

Usando el Modelo Perfecto del Tamaño

(β_{VAR} =0,20, β_{S} =1,20)

Programa	Estimación	MRE'	PRED(0,25)	RE'	rango
1	S'(VAR)	0,14	84%	+0,06	1
	S'(PGMR)	0,23	57%	+0,02	2
	E'(VAR)	0,26	52%	+0,12	1
	E'(PGMR)	0,42	57%	-0,35	2
2	S'(VAR)	0,24	61%	+0,01	1
	S'(PGMR)	0,42	52%	-0,39	2
	E'(VAR)	0,34	45%	+0,02	1
	E'(PGMR)	0,86	9%	-0,85	2

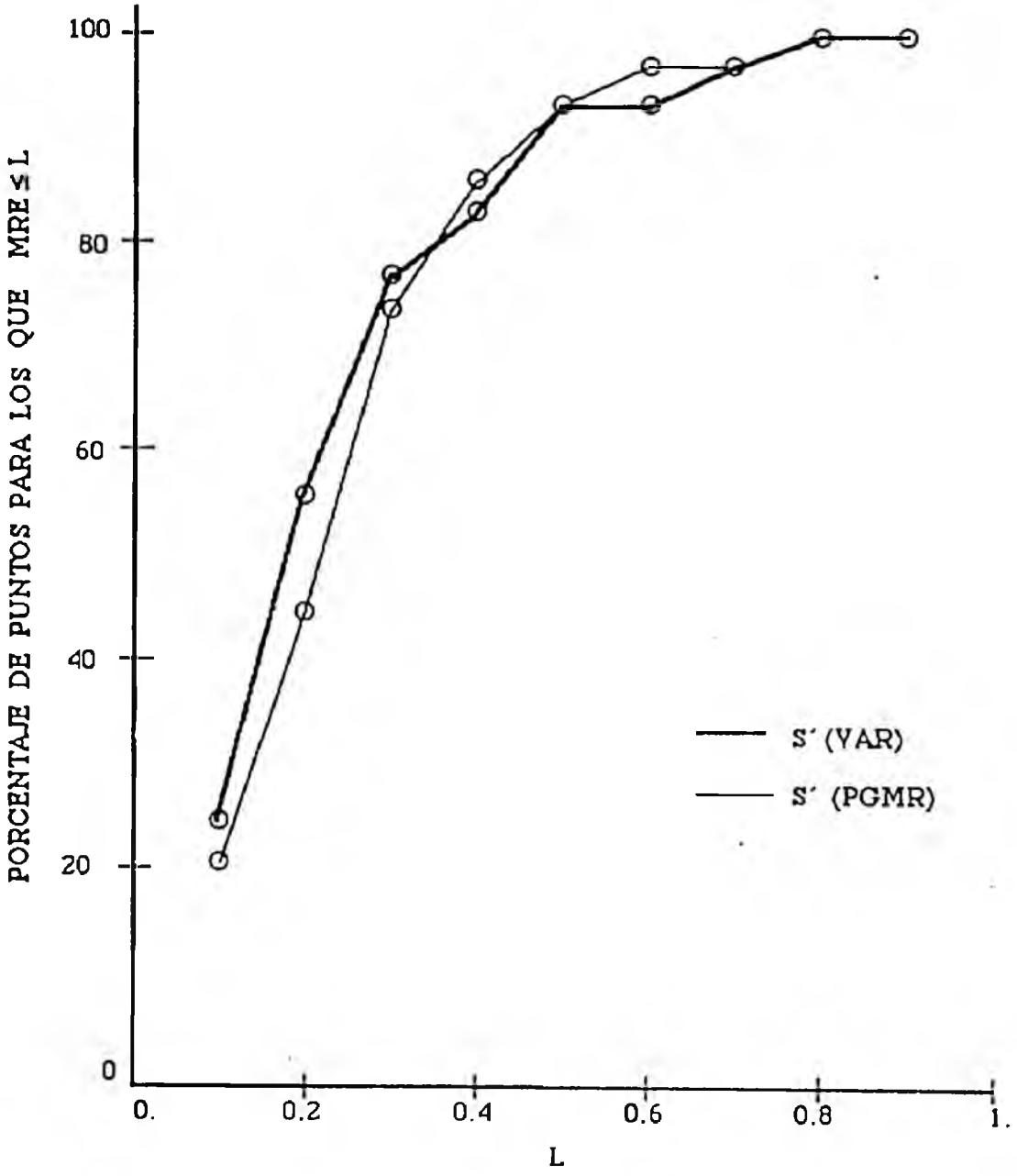


Figura 7.6

Estimación del Tamaño en base al Modelo Imperfecto del Tamaño

(EXPER-2a)

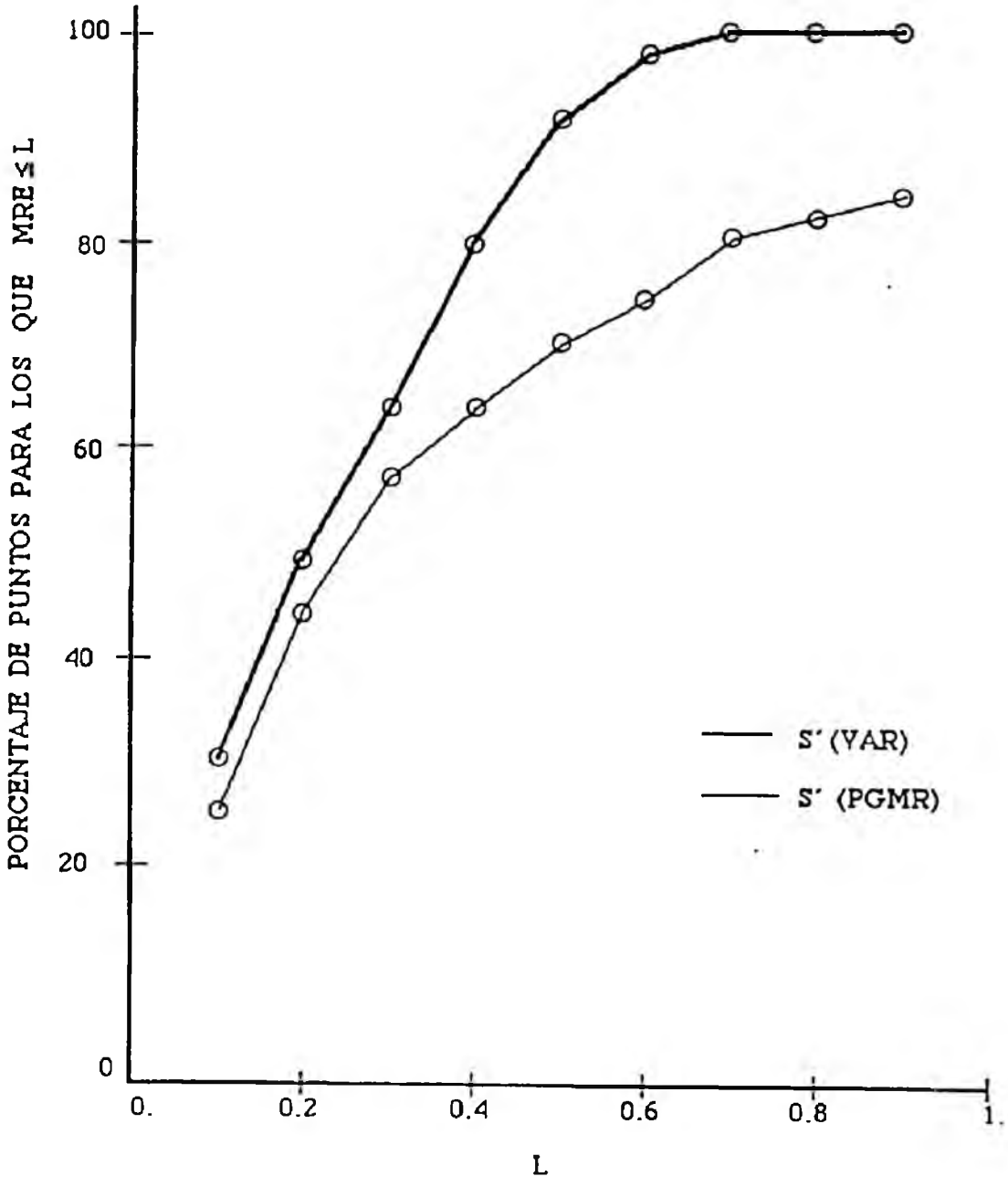


Figura 7.7

Estimación del Tamaño en base al Modelo Imperfecto del Tamaño

(EXPER-2b)

7.3 ESTIMACION UTILIZANDO EL MODELO DEL TAMAÑO IMPERFECTO

En la sección precedente, hemos mostrado que usando un modelo del tamaño perfecto, las estimaciones obtenidas eran bastante mejores que las estimaciones subjetivas. El modelo del tamaño perfecto considerado en nuestro estudio es aquel que para un programa dado puede convertir el valor de VAR en S, sin error. En la práctica, sin embargo, no estará disponible un modelo del tamaño perfecto, solamente existirán algunos modelos del tamaño imperfectos. Cuando VAR se convierte en S utilizando un modelo del tamaño imperfecto, se puede producir una cierta cantidad de error. Por lo tanto, esperamos que cuando apliquemos nuestro modelo de estimación, en base a un modelo del tamaño imperfecto, los resultados sean peores que los ya obtenidos.

Usamos como modelo del tamaño imperfecto la ecuación de regresión (7.3) establecida para convertir VAR en S, que a continuación repetimos esa ecuación:

$$S'(VAR) = 102 + 5,31VAR \qquad (7.6)$$

Debería tenerse en cuenta que si la fórmula de conversión no contiene ningún término constante, la evolución de la métrica S, obtenida a partir de VAR, será exactamente igual que la de VAR original. Sin embargo, si se incluye un término constante (positivo) la evolución "convertida" será más convexa que la evolución original. Por ejemplo, vamos a suponer que tenemos una evolución convexa de VAR, para los valores de VAR igual a 0, 30,

35, 40, 45, 50, y para el tiempo igual a 0, 1, 2, 3, 4, 5. Si partimos del hecho de que VAR se convierte en S usando el modelo del tamaño: S'(VAR) = 100 + 2VAR. Entonces, la evolución de S, obtenida a partir de VAR, para S igual a 0, 160, 170, 180, 190, 200, será más convexa que la evolución de VAR original. La diferencia entre las dos evoluciones depende de la magnitud del término constante. Ya que la fórmula de conversión representada por la ecuación (7.6) contiene un término constante, la evolución de S convertida erá más convexa que la evolución de VAR original. Según los análisis previos, se puede modelar mejor la evolución de VAR en EXPER-2a y EXPER-2b con una curva convexa de la forma:

$$VAR\ (t) = \alpha \times t^{0,20} \qquad (7.7)$$

Cuando el mismo análisis de procedimiento se aplicó a la evolución de S, obtenida a partir de VAR, nos dimos cuenta de que S convertida se podía modelar mejor en EXPER-2a y en EXPER-2b con una curva convexa de la forma:

$$VAR\ (t) = \alpha \times t^{0,15} \qquad (7.8)$$

Es decir, el valor varía ligeramente desde 0,20 a 0,15.

Los resultados de la estimación del tamaño con β_{VAR} igual a 0,15 y β_{S} igual a 1,20 usando el modelo del tamaño imperfecto (7.7) se muestran en la Tabla 7.4. Los resultados previos del tamaño estimado usando un modelo del tamaño perfecto se incluyen

también en la tabla (números entre paréntesis) para su comparación. Para EXPER-2a, el tamaño estimado por modelo no era mejor de modo significativo que el tamaño estimado por los programadores. Sin embargo, para EXPER-2b, el tamaño estimado por modelo es significativamente mejor que el tamaño estimado por los programadores para $\alpha < 0,10$. Los perfiles MRE para los dos grupos de estimaciones del tamaño se muestran en las Figuras 7.6 para EXPER-2a y 7.7 para EXPER-2b.

Estos resultados sugieren que incluso usando un modelo del tamaño imperfecto, las estimaciones del tamaño basadas en el modelo eran todavía mejores o iguales a las estimaciones del tamaño subjetivas. Por lo tanto, nuestro modelo en lo que se refiere al tamaño, es consistente frente a la modificación que supone usar el modelo del tamaño imperfecto, que es más realista, en lugar del modelo del tamaño perfecto.

Los resultados de la estimación del esfuerzo con los valores β_{VAR} igual a 0,15 y β_{S} igual a 1,20 usando la ecuación del tamaño imperfecto se muestran también en la Tabla 7.4. Las figuras entre paréntesis corresponden a los resultados de la estimación del esfuerzo usando un modelo del tamaño perfecto. Para EXPER-2a y EXPER-2b, el esfuerzo estimado por modelo era mejor de modo significativo que el esfuerzo estimado por los programadores para $\alpha < 0,10$. Los perfiles MRE para ambos grupos de estimaciones del esfuerzo se muestran en las Figuras 7.8 y 7.9 para EXPER-2a y EXPER-2b, respectivamente. Estos resultados sugieren que incluso usando un modelo del tamaño imperfecto, las estimaciones del

esfuerzo del modelo eran mejores que las estimaciones del esfuerzo subjetivas. Por lo tanto, en lo que se refiere al esfuerzo, es consistente frente a la modificación que supone usar el modelo del tamaño imperfecto, que es más realista, en lugar del modelo del tamaño perfecto.

Tabla 7.4

Estimación del Tamaño y del Esfuerzo

Usando el Modelo Imperfecto del Tamaño

(β_{VAR} =0,15, β_{S} =1,20)

Programa	Estimación	MRE'	PRED(0,25)	RE'	rango
1	S'(VAR)	0,22 (0,14)	68% (84%)	+0,05 (+0,06)	igual
	S'(PGMR)	0,23	57%	+0,02	igual
	E'(VAR)	0,24 (0,26)	64% (52%)	+0,12 (+0,12)	1
	E'(PGMR)	0,42	57%	-0,35	2
2	S'(VAR)	0,24 (0,24)	61% (61%)	+0,00 (+0,01)	1
	S'(PGMR)	0,42	52%	-0,39	2
	E'(VAR)	0,28 (0,34)	52% (45%)	+0,04 (+0,02)	1
	E'(PGMR)	0,86	9%	-0,85	2

Como ya hemos mencionado, esperábamos que los resultados de la estimación del tamaño y del esfuerzo usando el modelo del tamaño imperfecto fueran peores que los que se obtenían a través del modelo del tamaño perfecto. Sin embargo, nuestros resultados mostraron que éste no era el caso. La razón de ésto se puede explicar usando el gráfico mostrado en la Figura 7.10. Este gráfico es similar al mostrado en la Figura 7.1. Vamos a partir de que:

(1) La esquina inferior derecha (punto A) corresponde al inicio del proceso de desarrollo y el tamaño inicial del programa en LOC (que es 0).

(2) La esquina superior izquierda (punto B) corresponde al tiempo final de desarrollo y al tamaño final del programa.

(3) La curva continua de A a B representa el comportamiento de la evolución predicho del tamaño del programa.

(4) La curva discontinua de A a Q representa el comportamiento de la evolución predicho del tamaño convertido usando el modelo del tamaño perfecto.

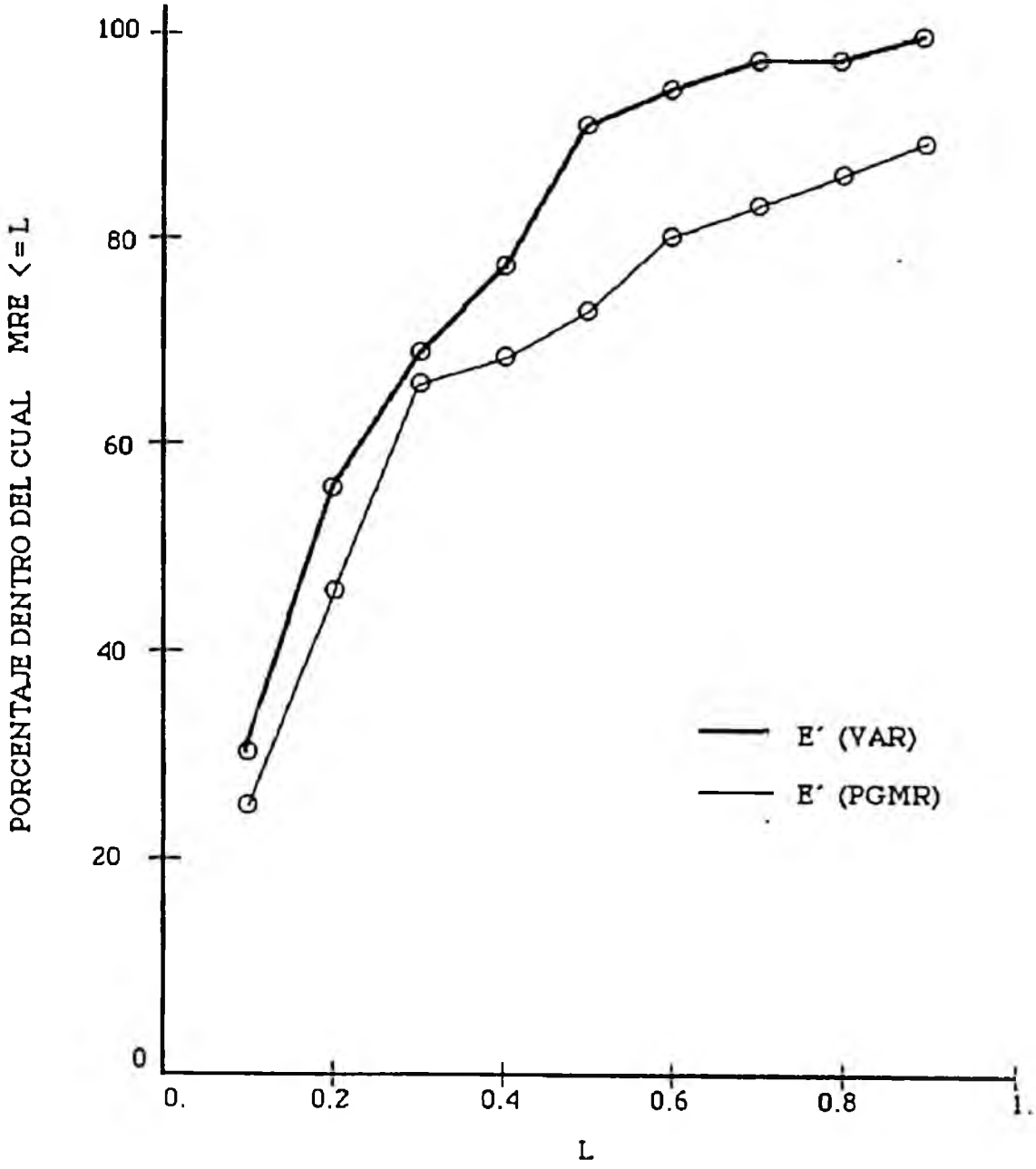


Figura 7.8

Estimación del Esfuerzo en base al Modelo Imperfecto del Tamaño

(EXPER-2a)

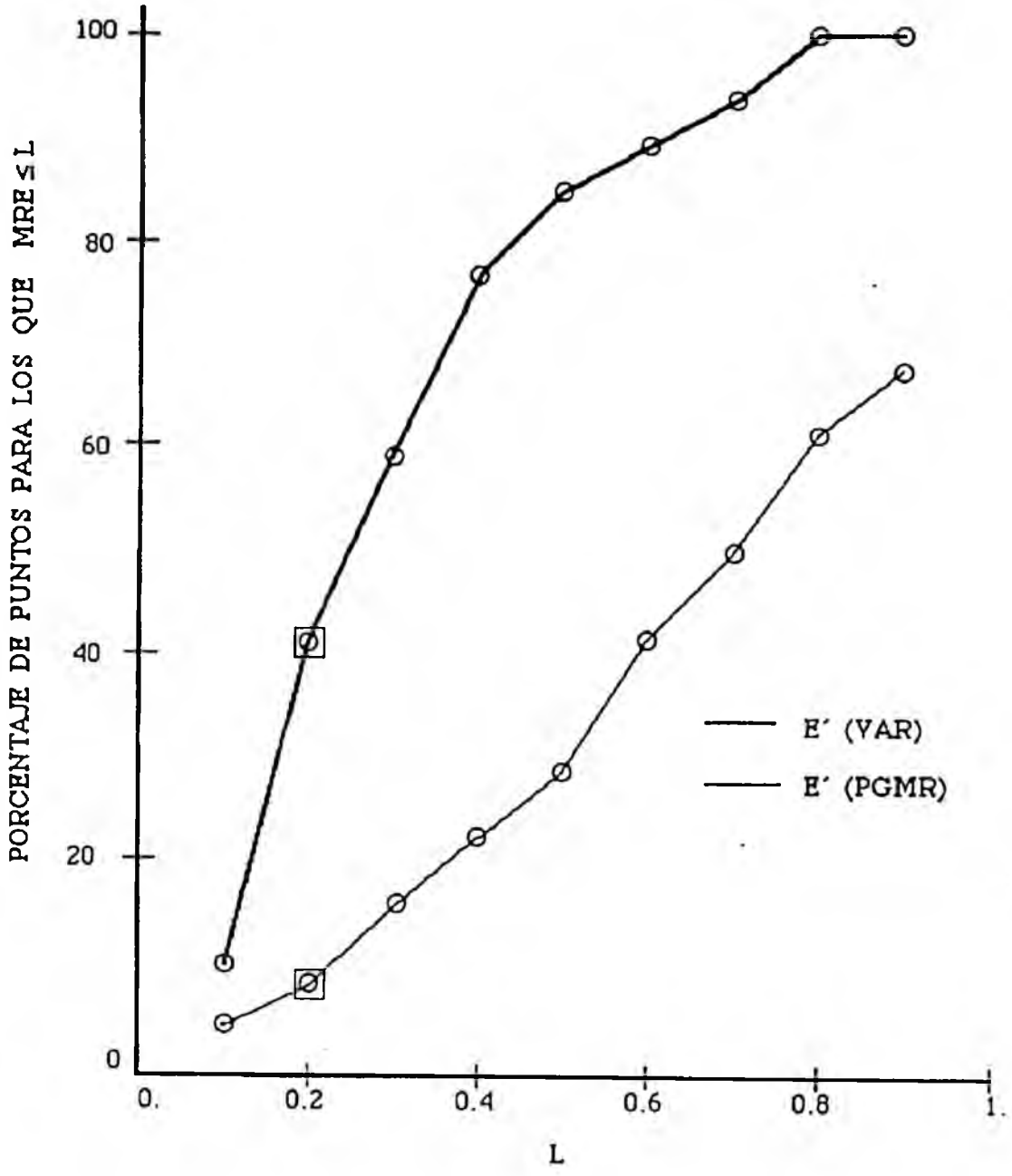


Figura 7.9

Estimación del Esfuerzo en base al Modelo Imperfecto del Tamaño

(EXPER-2b)

Ya que las dos curvas tienen como intersección el punto Q en lugar del punto B, el tamaño (S') y el esfuerzo (E') estimados no son exactamente iguales al tamaño y esfuerzo reales.

Supongamos que usamos el modelo del tamaño imperfecto y que el tamaño convertido sobrestima el tamaño real. Entonces la curva de evolución predicha del tamaño convertido será "más alta" que la original tal y como muestra la curva discontinua que va de A a R. Ya que R está más cercano a B que Q, las estimaciones del tamaño y del esfuerzo usando el modelo del tamaño imperfecto, en este caso son incluso mejores que aquellas dadas usando el modelo del tamaño perfecto.

Por otra parte, si el tamaño convertido subestima el tamaño real, la curva de la evolución predicha del tamaño convertido será "más baja" que la original tal y como muestra la curva discontinua que va de A a P. Ya que P está más distante de B que de Q, la estimación del tamaño y del esfuerzo usando el modelo del tamaño imperfecto es peor que aquella en la que se usa el modelo del tamaño perfecto. El análisis anterior ilustra que el error provocado por el ajuste de curva inicial puede cancelarse o reforzarse por el error causado por el modelo del tamaño imperfecto, tal y como se indica en la Sección 7.1. Esto explica, por otra parte, por qué nuestro modelo de estimación es bastante fuerte con respecto al cambio desde el modelo del tamaño perfecto al modelo del tamaño imperfecto.

7.4 MODELOS DE EVOLUCION

Mediante nuestro experimento EXPER-2 hemos demostrado la capacidad de realización de nuestra aproximación a la estimación del esfuerzo y del tamaño basada en la evolución de la métrica.

En este experimento, se crearon 44 programas para cada una de las dos aplicaciones, usando las estrategias de desarrollo "primera estructura de datos top-down" e "incremental". Para estos programas, el comportamiento medio de la evolución de VAR fue convexa, y el comportamiento medio de la evolución de S fue lineal. Al final de la etapa de diseño, se midieron los valores iniciales de S y VAR a lo largo de los tiempos de desarrollo correspondientes. Adicionalmente, VAR se convirtió en S usando un modelo del tamaño derivado de datos históricos. Basándonos en esta información y tomando como punto de partida el comportamiento de la evolución predicho para S y VAR, cuanto más propiamente se escogieron los valores de los parámetros β_{VAR} y β_{S}, tanto mejor se generaron las estimaciones del tamaño y las estimaciones del esfuerzo.

Nuestros descubrimientos sugieren que las estimaciones del tamaño generadas usando nuestro modelo eran tan buenas o mejores que las estimaciones del tamaño subjetivas realizadas por los programadores. También encontramos que las estimaciones del esfuerzo generadas usando nuestro modelo eran mejores de modo significativo que las estimaciones del esfuerzo subjetivas.

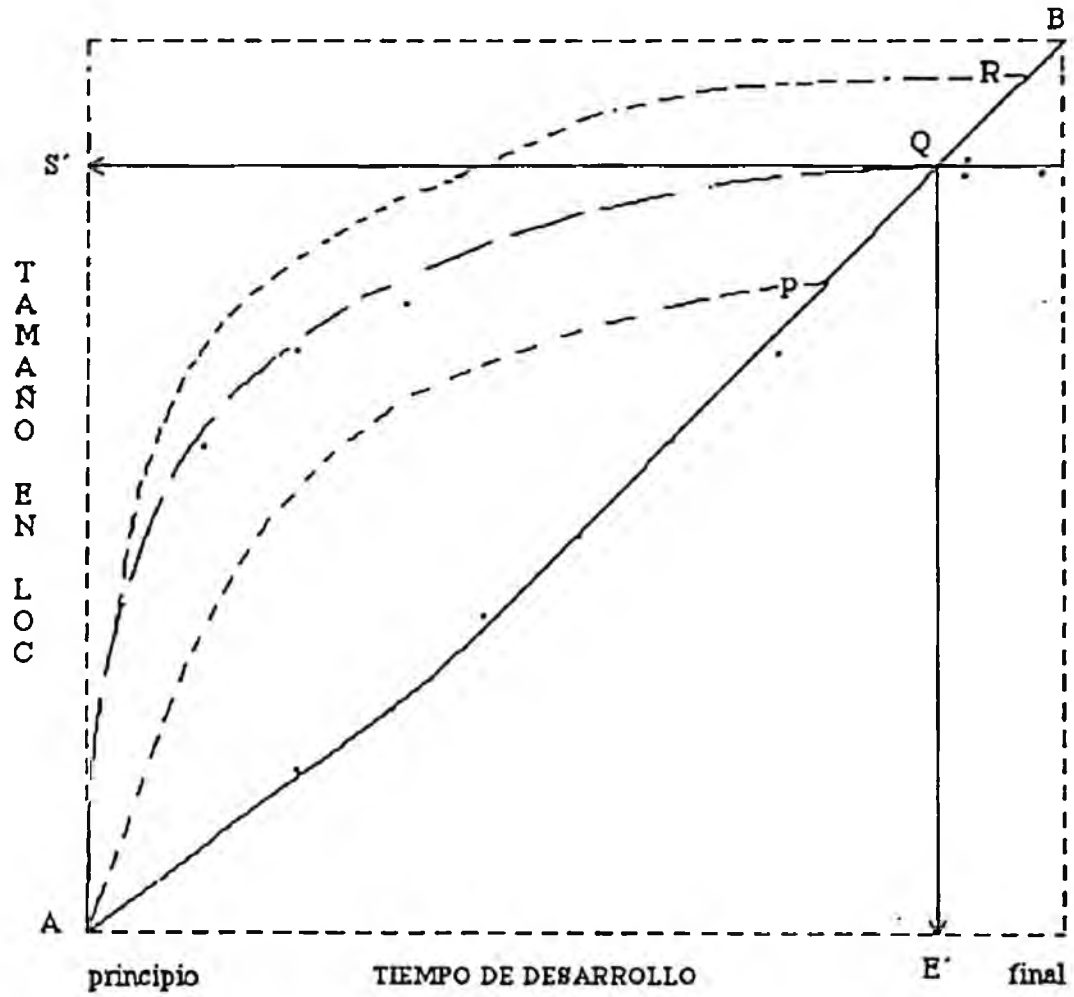


Figura 7.10

Análisis del Error para la Estimación del Tamaño y del Esfuerzo

7.5 LIMITACIONES E IMPLICACIONES

Se puede argumentar que la necesidad de predecir los valores de parámetro para β_{VAR} y β_S es una debilidad de nuestro modelo. Debe tenerse en cuenta que las estrategias de desarrollo usadas pueden determinar realmente escalas iniciales para los valores de esos dos parámetros. De cualquier modo, el minucioso ajuste de estos dos valores puede requerir algún valor histórico. Recordemos que los valores de parámetro 0,20 y 1,20 para β_{VAR} y β_S se determinaron usando los datos del EXPER-2a. Cuando nuestro procedimiento de estimación se aplicó a los datos del EXPER-2b, se usaron exactamente los mismos valores. Adicionalmente, los programas del EXPER-2a y los programas del EXPER-2b eran diferentes tipos de programas construidos usando las mismas estrategias de desarrollo. Consecuentemente, esto demuestra que un buen ajuste de los valores de β_{VAR} y β_S usando datos históricos pueden funcionar para un nuevo proyecto (suponemos que EXPER-2a son los datos históricos, y EXPER-2b son los datos nuevos) en el que se utilicen las mismas estrategias de desarrollo.

Por otra parte, cuando no hay datos históricos disponibles para determinar los valores de β_{VAR} y β_S, sugerimos que todavía se puede aplicar este modelo de alguna manera. Esto es, se puede usar como punto de partida un valor teórico razonable para β_{VAR} y β_S. Por ejemplo, si se siguen de cerca las estrategias de desarrollo "primera estructura de datos top-down" e "incremental", se puede usar el valor 1,2 como valor inicial de

β_S y el valor 0,2 como valor inicial de β_{VAR}. Por lo tanto, a partir de los resultados obtenemos la conclusión de que para programas construidos bajo las mismas condiciones que las de nuestro experimento EXPER-2, nuestra aproximación es útil potencialmente para la estimación inicial del tamaño y del esfuerzo.

CAPITULO 8

RESUMEN Y CONCLUSIONES

Para obtener un modelo de predicción del esfuerzo basado en el comportamiento de la evolución de la métrica, nos habíamos marcado dos puntos de atención. El primero de ellos era desarrollar técnicas para la estimación del tamaño del programa en una etapa inicial del proceso de desarrollo. En los Capítulos 3 y 5 se discutierón los resultados más importantes de este estudio de estimación del tamaño, que a continuación se resumen.

En el Capítulo 3, presentamos una aproximación indirecta a la estimación del tamaño, basada en la ecuación de longitud de Halstead y en una estimación inicial de n_2 (número de operandos únicos). Esta aproximación se evaluó usando los datos recogidos de nuestro primer experimento de construcción de programas, EXPER-1. La estimación del tamaño utilizando la ecuación de longitud de Halstead, para los 21 programas codificados en Pascal que formaban EXPER-1, subestimó el tamaño real por un valor medio del 32%. Además, observamos que los datos de este estudio no soportaban nuestra suposición de partida de que n_2 podía estimarse bien, al final de la etapa de diseño. Los valores iniciales de n_2 eran aproximadamente del 42% de los valores finales. Estos resultados nos sugieren que:

a) Para programas realizados en Pascal, la ecuación de longitud de Halstead, no es un buen modelo para la estimación tamaño.

b) Para la estimación inicial la métrica n_2, no es la más apropiada.

En el Capítulo 4, se demostró la capacidad de realización de nuestra aproximación indirecta a la estimación del tamaño. Para ello nos basamos en los datos recogidos en nuestro segundo experimento, EXPER-2. El modelo del tamaño usado erá un modelo de una ecuación de regresión simple, que aproximaba el tamaño del programa en LOC como una función de la métrica VAR (el número de variables únicas). Los coeficientes de regresión se derivaron usando datos independientes de los datos recogidos en EXPER-2. Se encontró que este modelo es más consistente y fiable que la ecuación de longitud de Halstead, para los 88 programas construidos en el estudio EXPER-2. Además, este estudio nos permitió confirmar nuestra suposición de que VAR puede estimarse correctamente al final de la etapa de diseño. Los valores iniciales de VAR eran aproximadamente el 80% de los valores finales. Como consecuencia, las estimaciones del tamaño indirectas basadas en este nuevo modelo del tamaño y la estimación inicial de VAR fueron tan buenas o significativamente mejores que las estimaciones subjetivas hechas por los programadores. El promedio de error relativo de esta estimación indirecta del tamaño, fue aproximadamente del 22%. Estos resultados sugieren que:

a) Hemos encontrado un modelo del tamaño para programas Pascal mejor que la ecuación de longitud de Halstead.

b) Es más fácil estimar inicialmente la métrica VAR que n_2.

Basándonos en nuestros propios resultados, creemos que se puede mejorar la estimación inicial del tamaño del programa, al final de la etapa de diseño. El éxito en la aplicación de nuestra aproximación depende de dos factores:

a) Disponer de un buen modelo del tamaño, para convertir VAR en el tamaño del programa.

b) Diseñar los programas utilizando de manera correcta y adecuada la estrategia de desarrollo "primera estructura de datos top-down".

En cualquier ambito de organización de datos, puede obtenerse un buen modelo del tamaño basado en la ecuación de regresión usando datos históricos de programas escritos en el mismo lenguaje dentro de la organización dada. El diseño top-down se ha aceptado ampliamente como una estrategia adecuada para muchas situaciones [ver por ejemplo, ZELK79]. Las estrategias de diseño de estructuras orientadas a los datos [PRES82] tales como la metodología Jackson [JACK75] y la metodología Warnier [WARN74], se han utilizado ampliamente en muchas áreas de

aplicación. Por lo tanto, en el mundo empresarial, se puede utilizar la estrategia de desarrollo "primera estructura de datos top-down" empleada en nuestro estudio, de manera amplia y con resultados satisfactorios.

Mientras que no demos validez a nuestra aproximación en proyectos de gran escala, sugerimos la siguiente manera de aplicar nuestra aproximación a estos proyectos. Normalmente un gran proyecto está compuesto de muchos mini-proyectos. El tamaño del proyecto es simplemente la suma de los tamaños de los anteriores. Así, para un gran proyecto, si cada uno de los mini-proyectos se desarrolla usando la misma estrategia descrita en nuestro estudio, el tamaño total del proyecto puede estimarse por la suma individualizada de los mismos. Además, se tendrá en cuenta el tiempo de integración de los mini-proyectos.

A pesar de algunos resultados, que nos animaban a continuar en nuestro estudio de estimación del tamaño, hay que obrar con precaución en la aplicación de estos resultados. Por ejemplo, nuestra aproximación a la estimación del tamaño, se aplicó al final en lugar de al principio de la fase de diseño. Por lo tanto, no podemos aplicar directamente nuestros resultados a estimaciones "iniciales" antes de la etapa de diseño. De acuerdo con Boehm [BOEH84], para seguir un camino válido para el cálculo del tamaño del software, es necesario conocer y entender de manera amplia la naturaleza del producto de software que se desea desarrollar. Nuestra aproximación requiere una comprensión

inicial de la estructura total y de las estructuras de datos del programa al final de la etapa de diseño. Además, puede existir una amplia gama de aproximaciones de diseño para una especificación dada, lo que nos lleva a una amplio gama de tamaños [BOEH84]. Este fue realmente el caso de nuestro estudio EXPER-2. Hay que recordar, que en el estudio EXPER-2 se realizaron 88 programas, 44 con un enunciado, y 44 con otro. Como se ha mostrado en la Tabla 4.1, observamos una variación de tamaños de 1 a 3,5 para programas con el mismo enunciado.

De cualquier modo, las estimaciones iniciales en nuestro estudio se han obtenido al final de la etapa de diseño y las decisiones específicas de diseño son generalmente conocidas en este punto. Esto explica probablemente por qué nuestra aproximación, funcionaba para los programas escritos con el mismo enunciado: obtuvimos nuestras estimaciones en un punto lo suficientemente tardío, cuando la mayor parte de las decisiones críticas de diseño ya habían sido tomadas.

Como se ha discutido en el Capítulo 3, la aproximación de Albrecht a la estimación inicial del tamaño de un programa está basada en los "puntos de función" derivados de las especificaciones del mismo. Vamos a partir de la base de que la aproximación de Albrecht ha sido aplicada al final de la etapa de especificación en el estudio EXPER-2; ya que los 44 programas tanto del EXPER-2a como del EXPER-2b se escribieron a partir del mismo enunciado, la aproximación de Albrecht habría producido una sola estimación del tamaño para todos los programas del EXPER-2a

y del EXPER-2b. Debido a la variación de tamaño de 1 a 3,5 observada para todos los programas de EXPER-2, el error relativo para esta estimación del tamaño habría sido del 250% o más. Por otra parte, cuando nuestra aproximación a la estimación del tamaño, se aplicó al final de la etapa de diseño, el MRE' (la magnitud media del error relativo) producido para esa estimación fue menor del 25%, tal y como hemos mostrado en las Tablas 5.4 y 7.4. Por lo tanto, ya que el desarrollo del software progresa, desde la etapa de especificación a la etapa de diseño, podemos mejorar de modo significativo, la estimación del tamaño, basándonos en la medida inicial del contador de variable única, VAR, durante la etapa de diseño.

Nuestro segundo punto de atención era el estudio de la evolución de la métrica. En el Capítulo 4 se presentaron las ideas básicas de dicho estudio. Los resultados más reveladores se discutieron en los Capítulos 6 y 7.

En el Capítulo 6, observamos que el comportamiento medio de la evolución del tamaño del programa parecía ser lineal para los 88 programas construidos usando la estrategia incremental. El comportamiento medio de la evolución de VAR y n_2 era convexo bajo la estrategia de desarrollo "primera estructura de datos top-down", con una evolución de VAR más convexa que la evolución de n_2. Estos resultados nos indican que la evolución de la métrica no tiene por qué ser un proceso aleatorio. Existen algunos

factores que afectan de manera significativa a este proceso. Hemos identificado que uno de los factores que más influye en esta evolución es la estrategia de desarrollo.

En el Capítulo 7, se evaluó el modelo de estimación del esfuerzo y del tamaño presentado en el Capítulo 4, basándonos en su capacidad para predecir el tamaño real del programa y el tiempo real de desarrollo. Este modelo se basó en la evolución lineal del tamaño del programa y en la evolución convexa de las métricas del tamaño relacionadas bajo las estrategias de desarrollo "incremental" y "primera estructura de datos top-down". Las estimaciones del tamaño y del esfuerzo producidas por este método fueron más precisas que las estimaciones subjetivas del tamaño y del esfuerzo. Esto era debido a una elección apropiada de los valores del parámetro de la ecuación del modelo. Sugerimos que los valores del parámetro de dicha ecuación se pueden determinar en base a las estrategias de desarrollo empleadas.

Aunque demostramos la capacidad de realización de nuestra aproximación basada en la evolución de la métrica para predecir el esfuerzo y el tamaño, la aplicación de los resultados de la estimación a proyectos grandes puede ser bastante limitada. Como se ha mencionado anteriormente, el esfuerzo del proyecto incluye el esfuerzo de los componentes individuales del proyecto, además del esfuerzo de las actividades de interacción entre los componentes. El esfuerzo de los componentes individuales no se tiene en cuenta en el esfuerzo total. De cualquier modo, la

capacidad de predecir el esfuerzo que implica la producción de cada componente debería ser una contribución importante para la predicción del esfuerzo total. Basándonos en los argumentos descritos, consideramos que los resultados de nuestra estimación del tamaño son probablemente más generalizables que los resultados de nuestra aproximación del esfuerzo a un entorno de programación profesional.

Otro resultado interesante en nuestro estudio de evolución de la métrica tiene que ver con las reacciones de los participantes ante las dos estrategias de desarrollo usadas en el estudio EXPER-2. A partir de los resultados del cuestionario post-experimento, encontramos que un porcentaje substancial (85%) de los participantes estaban a favor de usar estas dos estrategias. Vamos a indicar algunos de los comentarios realizados por los participantes sobre estas dos estrategias.

1. Estrategia de diseño Top-down:

   - Fuerza al programador a pensar previamente en todo lo que puede acontecer.

   - Ayuda a evitar grandes rediseños del programa mientras se está desarrollando, ya que el programador ha tenido que concebirlo todo previamente.

2. Estrategia Incremental:

- Cuando se encuentra un error, es más fácil de buscar y de fijar, ya que hay pocos lugares en que localizarlo (porque el resto del código ya ha sido probado).

- Me parece que este es el único modo de escribir un programa grande.

- Para un programa grande y complicado, la efectividad de utilizar la estrategia incremental será más apreciada.

Ya que descubrimos que las estimaciones del esfuerzo y del tamaño se sirvieron de estas dos estrategias de desarrollo, y puesto que es más fácil obtener programadores que usen estrategias que les gusten, fue muy positivo conocer que los participantes en el experimento favorecieron, en general, estas dos estrategias de desarrollo.

Mientras se han realizado algunos progresos en esta investigación, en el estudio de la estimación del tamaño y evolución de la métrica, todavia queda mucho por hacer. A continuación damos algunas sugerencias.

Refiriéndonos al estudio de estimación del tamaño, necesitamos definitivamente experimentos más controlados que impliquen programas más grandes (aproximadamente de 1000 a 5000 LOC) y una gama más amplia de aplicaciones para confirmar nuestros hallazgos. Adicionalmente, deben investigarse otras métricas del tamaño relacionadas además de VAR y n_2. Candidatos posibles incluyen métricas de estructuras de control y métricas de flujo de datos. También deberia experimentarse con las estrategias de desarrollo asociadas que facilitan el desarrollo inicial de estas métricas del tamaño. En lo que se refiere al estudio de evolución de la métrica, debe examinarse el comportamiento de evolución de otrás métricas además de VAR, n_2, S y N bajo otras estrategias de desarrollo.

El fin, a largo plazo, de nuestra investigación es desarrollar técnicas que mejoren la estimación del esfuerzo del software. Entre los factores que influencian el esfuerzo del software, el tamaño no es sólo el dominante, sino que también es el más difícil de estimar en una etapa inicial. Así, el establecimiento de un método que ayude a una estimación del tamaño más precisa es esencial de modo definitivo para el desarrollo de una técnica para una mejor estimación del esfuerzo. Esta tesis ha presentado un método de estimación del tamaño que proporciona una manera objetiva y algoritmica de predecir el tamaño final de un programa basandonos en la medición inicial de ciertas métricas al final de la etapa de diseño.

Si llevamos los resultados de este trabajo al ámbito comercial, hay que tener en cuenta, hoy en día, el gran número de empresas dedicadas a la construcción de software para distintos usuarios, denominadas empresas de servicios informáticos. Para este tipo de empresas la estimación del tamaño de un proyecto de software, es el primer paso para la estimación del esfuerzo de realización del mismo. Si conocemos la cantidad de esfuerzo necesario para realizar un trabajo, conocemos su coste, parte fundamental para estas empresas, ya que en principio, presupuestan el costo de los proyectos que desarrollan.

Esta investigación ha mostrado con éxito que ciertas métricas pueden medirse al final de la etapa de diseño y que pueden proporcionar información valiosa, para una estimación precisa del tamaño, que justifique la planificación de proyectos subsiguientes. Nuestro propósito es continuar la investigación y esperar que nuestras técnicas y resultados estimulen a otros para adentrarse en estos estos difíciles problemas mientras que seguimos progresando y nos acercamos a la finalidad de tener una estimación mejor del esfuerzo del software.

BIBLIOGRAFIA

ALBR83 A.J. Albrecht and J.E. Gaffney,Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering. November 1983

ARME89 Pasquale Armenise, "A Structured Approach to Program Optimization", IEEE Transactions on Software Engineering. February 1989

BAIL81 J. W. Bailey and V. R. Basili, "A Meta-Model for Sofware Development Resource Expenditure, "Proceedings, Figth International Conference on Software Engineerring, IEEE/ACM/NBS, March 1981, pp. 107-116.

BELA79 L. Belady and M. Lehman, "The Characteristics of Large Systems", in Research Directions in Software Technology, P. Wegner, Ed.,Cambridge, MA: M.I.T. Press, 1979, pp. 106-138.

BOEH81 B. M. Boehm, Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, Nj, 1981.

BOEH84 B. M. Boehm, "Software Engineering Economics", IEEE Transactions on Sooftware Engineering, January 1984, pp. 4-21.

BROO74 F. P. Brooks, Jr., The Mythical Man-Month, Addison-Wesley, Inc., Reading, MA, 1975.

BURC82 C. D. Burch, "Purdue Pascal Software Metrics Analyzer User's Manual, "Dept. of Computer Science, Purdue University, May 1982.

CHRI81 K. Christensen, G. P. Fitsos, and C. P. Smith, "A Perspective on Software Science", IBM Systems Journal, Nº 4, 1981, pp.372-387.

COME81 D. Comer, "Principles of Programs Desing Induced from Experience with Small Public Programs", IEEE Transactions on Software Engineering, March 1981, pp. 169-174.

DAVI88 Alan M. Davis, Edward H. Bersoff and Edward R. Comer, "A Strategi for Comparing Alternative Software Lyfe Cycle Models", IEEE Transactions on Software Engineering. October 1988

DEMA81 T. De Marco, "Yourdon Proyect Survey: Final Report", Yourdon, Inc. New York, NY, September 1981.

DEMA82 T. De Marco, Controlling Software Projects: Management, Measurement, and estimation Yourdon Press, Inc., New York, 1982.

DIJK72 E. W. Dijkstra, "The Humble Programer", Communications of the ACM October, 1972, pp. 859-866.

DOTY77 Doty Associates, Inc., Software Cost Estimation Study, RADC-TR-77-220, Rome Air Development Center, Rome, N.Y., 1977.

DUNS84a H. E. Dunsmore, "Software Metrics: an Overview of an Envolving Methodology", Information Procesing and Management, Volume 20, Nº 1-2, 1984, pp. 183-192.

DUNS84b H. E. Dunsmore, V. Y. Shen, and S. D. Conte, "A Comparison of a Few Estimation Models", Journal of Parametrics, March 1984, pp. 5-14.

ELSH78 J. L. Elshoff, "An Investigation into the Effect of the Counting Method Used on Software Science Measurements", ACM SIGPLAN Notices, Vol. 13, February 1978, pp. 30-45.

FITS80a G. P. Fitsos, "Vocavulary Effects in Software Science", IBM Santa Teresa Lab., Tech. Rep. 03.082, January 1980.

FITS80b G. P. Fitsos and C. P. Smith, "Software Science Programming Guidelines", Ibm Santa Teresa Lab., Tech. Rep. 03.098, July 1980.

FITS80c G. P. Fitsos, "Vocabulary Effects in Software Science", Proceedings of COMPSAC 80, October 1980, pp. 751-756.

FREI79 F. R. Freiman and R. E. Park, "Price Software model-version3: an overveiw", Proceedings Workshop on Quantitative Software Models, 1979, pp. 32-41.

HALS77 M. H. Halstead, Elements of Software Science, Elsevier North Holland, Inc. New York, N.Y.,1977.

HOAR84 C. A. R. Hoare, "Programming: Sorcery or Science?" IEEE Software April 1984, pp. 5-18.

HORO77 E. Horowitz and S. Sahni, Fundamentals of Data Structures, Computer Science Press, Inc. Potomac, MD, 1976.

ITAK82 M. Itakura and A. Takagayanagi " A Model for Estimating Program Size and its Evaluation", Proceedings, Sixth International Conference on Software Engineering, September 1982, pp. 104-109.

JACK75 M. Jackson, Principles of Program Desing, Academic Press, 1975.

JACK83 M. Jackson, "System Development", Prentice-Hall, 1983.

JONE78 T. C. Jones, "Measuring Programing Quality and Productivity", IBM Systems Journal, Nº 1, 1978, pp. 39-63.

JOHN81 D. B. Johnston and A. M. Lister, "A Note on the Software Science Length Equation", Software - Practice and experience, Vol. 11, 1981, pp. 875-879.

LICH87 H.F. Li, and W.K. Cheung, "An Empirical Study of Software Metrics", IEEE Transactions on Software Engineering. June 1987

LOND87 Bernard Londeix, Cost Estimation for Software Development, Addison-Wesley Publishing Company, 1987

MCCA76 T. J. McCabe , "A Complexity Measure", IEEE Transactions on Software Engineering, December 1976, pp. 308-320.

MOHA81 S. N. Mohanty, "Software Cost Estimation: Present and Future", Software - Practice and Experience, Vol. 11, 1981, pp. 103-121.

MYNA90 Barbee T. Mynalt and Laura M. Leventhal, "An Evaluation of a CASE-based Approach to teaching Undergraduate Software Engineering", ACM SIGCSE Bolletin, February 1990

NETE74 J. Neter and W. Wasserman, Applied Linear Statistical Models, Irwin, Inc., Homewood, IL, 1974.

NETE78 J. Neter and W. Wasserman and G. A. Whitmore Applied Statistics, Allyn and Bacon, Inc., 1978.

NIE75 N. H. Nie, D. H. Bend, and C. H. Hull, SPSS: Statistical Package for the Social Sciences, 2nd ed., McGraw Hill, Inc., New York, N.Y., 1975.

PAGE86 Alain Pages, and Michel Gondran, System Reliability: Evaluation and Prediction in Engineering, North Oxford Academic, 1986

PRES82 R. S. Pressman, Software Engineering: A Practioner's Approach McGraw Hill, Inc., New York, N.Y., 1982, pp. 205-241.

PUNT78 L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem", IEEE Tranactions on Software Engineering, July 1978, pp. 245-361.

RAMA88 Bina Ramamurthy and Austin Melton, "A Synthesis of Software Science Mesures and the Cyclomatic Number", IEEE Transactions on Software Engineering. August 1988

SAMM82 L. H. Sammet, D. W. Wangh, and R. W. Reiter, "PDL/Ada -A Desing Languaje Based on Ada", ACM Ada Letters, 2, 3 (December 1982), pp. 19-31.

SHAW89 Wade H. Shaw, James Howatt, Robert S. Maness and Dennis Milder, "A Software Science Model of Compile Time", IEEE Transactions on Software Engineering. May 1989.

SHEN83 V. Y. Shen and H. E. Dunsmore, "Analyzing Cobol Programs via Software Science", Dept. Computer Science, Purdue University, Rep. CSD TR-348, August 1980; revised September 1981.

SHEN83 V. Y. Shen and S. D. Conte and H. D. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and its Empirical Support", IEEE Transactions of Software Engineering, March 1983, pp. 155-165.

SHEP80 S. B. Sheppard, P. Milliman, and B. Curtis, "Experimental Evaluation of On-line Program Construction", COMPSAC, 1980, pp. 505-510.

SMIT80 C. P. Smith, " A Software Science Analysis of Programming Size", Proceedings of the ACM Annual Conference, October 1980, pp. 179-185.

SOMM85 I. Sommerville, Software Emngineering, Addison-Wesley Publishing Company, 1985

THAY81 R. H. Thayer, A. B. Pyster and R. C. Wood, "Mayor Issues in Software Engineering Project Management", IEEE Transactions of Software Engineering, July 1981, pp. 333-342.

THAY84 R. H. Thayer, A. B. Pyster, "Guest Editorial: Software Engineering Proyect Management", IEEE Transactions of Software Engineering, January 1984, pp. 2-3.

THEB83 S. M. Thebaut, "The Saturation Effect in Large-Scale Software Development: its Impact and Control", Ph.D. Th., Purdue Univ., May 1983.

TROY81 D. A. Troy and S. H. Zweben, "Measuring the Quality of Structures Desings", Journal of Systems and Software 2,2(1981), pp. 113-120.

WALS77 C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation", IBM Systems Journal, Vol. 16, Nº 1, 1977, pp. 54-73.

WARN74 J. D. Warnier, Logical Construction of Programs,. Van Nostrand 1974.

WEIN70 G. F. Weinwurm, On the Measurement of Computer Programming Auervach, Princeton, N.J., 1970.

WEYU88 Elaine J. Weyuker, "Evaluation Software Complexity Measures", IEEE Transactions on Software Engineering. September 1988.

WINE71 B. J. Winer, Statistical Principles in Experimentals Design, McGraw Hill, 1971.

WOLV74 R. W. Wolverton, "The Cost of Developing Large-scale Software", IEEE Transcriptions on Computers, Vol.23, Nº 6, 1974.

WOOD80 S. N. Woodfield, "Enhanced Effort Estimation by Extending Basic Programming Models to Include Modularity Factors", Ph.D. Thesis, Purdue University, West Lafayette, IN, December 1980.

YAU88 Stephen S. Yau, Robin A. Nicholl, Jeffrey J.P. Tsai, and Syubg-Syang Liu, "An Integrated Lyfe-Cycle Model for Software Maintenance", IEEE Transactions on Software Engineering. August 1988

YOUR79 E. Yourdon and L. Constantine, Structured Design, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.

YUSH88 Tze-Jie Yu, Vincent Y. Shen and Hubert E. Dunsmore, "An Analysis of Several Software Defect Models", IEEE Transactions on Software Engineering. September 1988.

ZELK79 M. V. Zelkowitz, A. C. Shaw, and J. D. Gannon, Principles of Software Enngineering and Desing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1979..pa

APENDICES

Apendice A:

Gráficos de Evolución del Tamaño (EXPER-2a)

Figura A.1

Evolución del Tamaño Completamente Lineal

Figura A.2

Evolución del Tamaño Casi Lineal

Figura A.3

Evolución del Tamaño con un Cambio Tardío

Figura A.4

Evolución del Tamaño con un Cambio Temprano

Figura A.5

Evolución del Tamaño No Lineal (Ejemplo 1)

Apendice B:

Gráficos de Evolución del Tamaño (EXPER-2b)

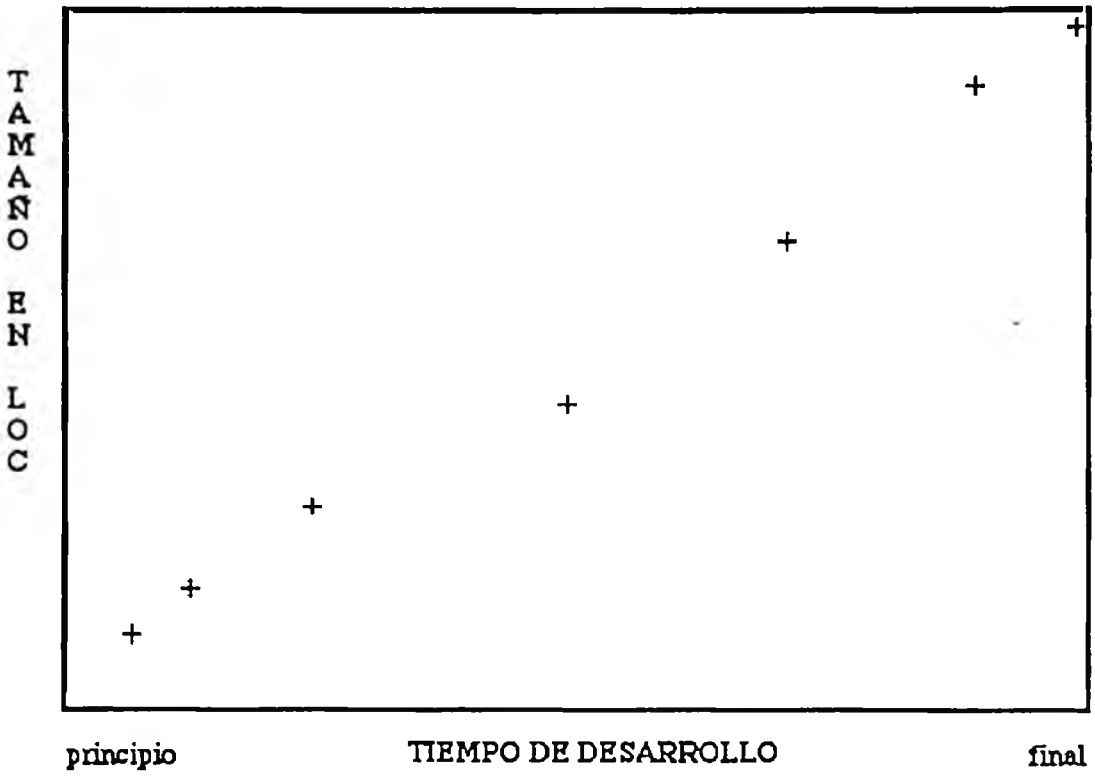


Figura B.1

Evolución del Tamaño Completamente Lineal

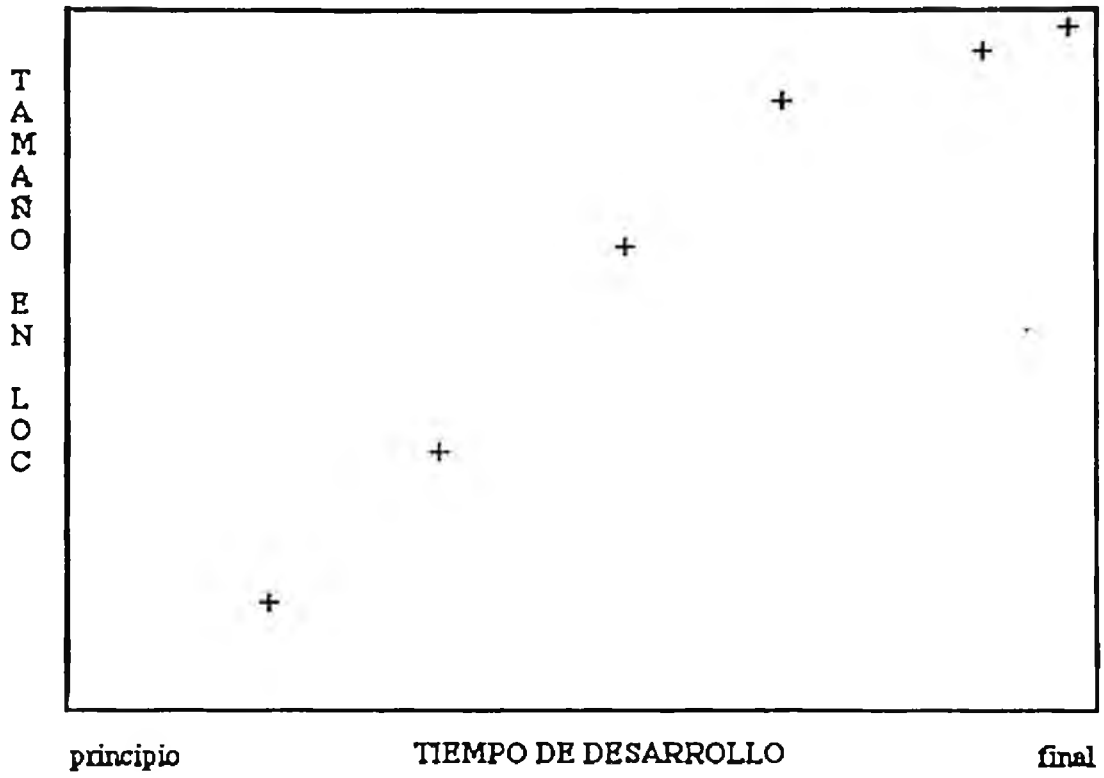


Figura B.2

Evolución del Tamaño Mayormente Lineal

Apendice C:

Gráficos de Evolución de VAR

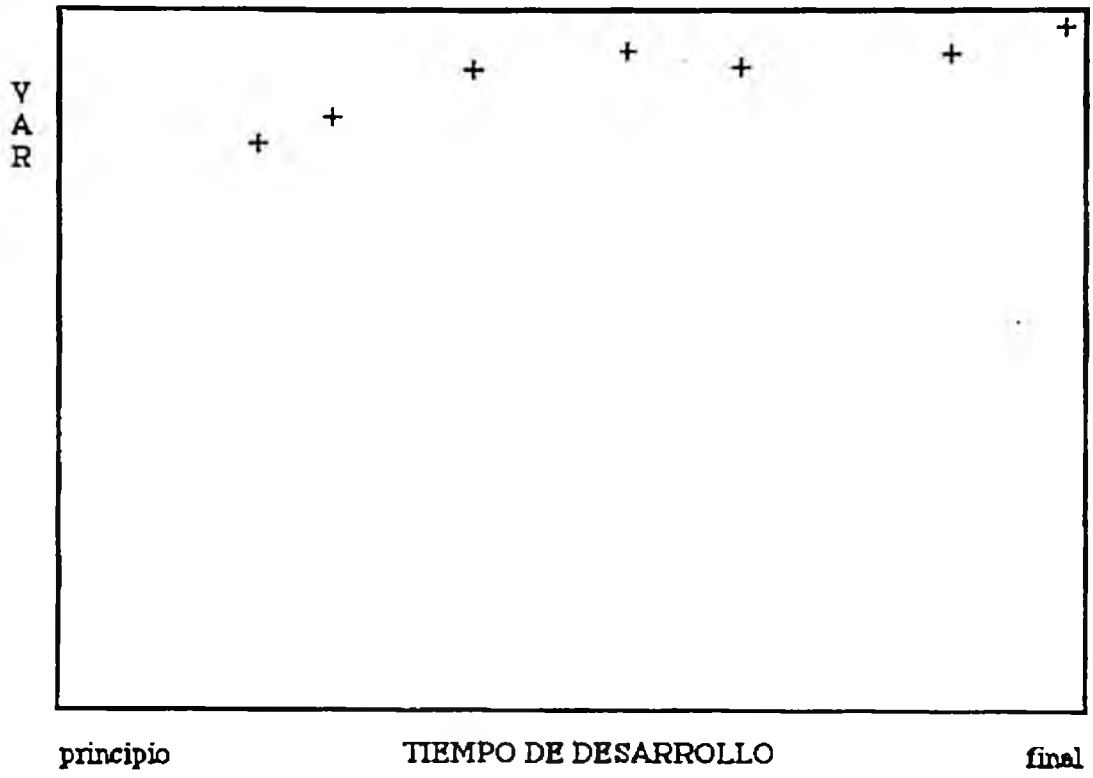


Figura C.1

Evolución de VAR (Ejemplo 1)

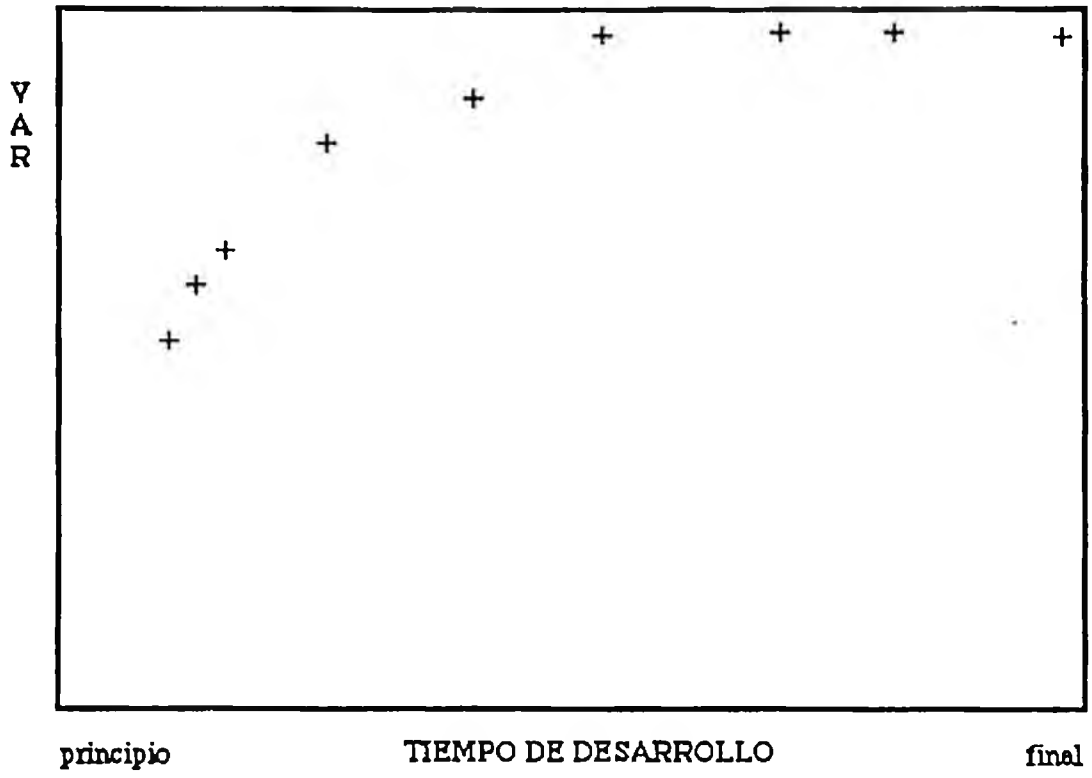


Figura C.2

Evolución de VAR (Ejemplo 2)

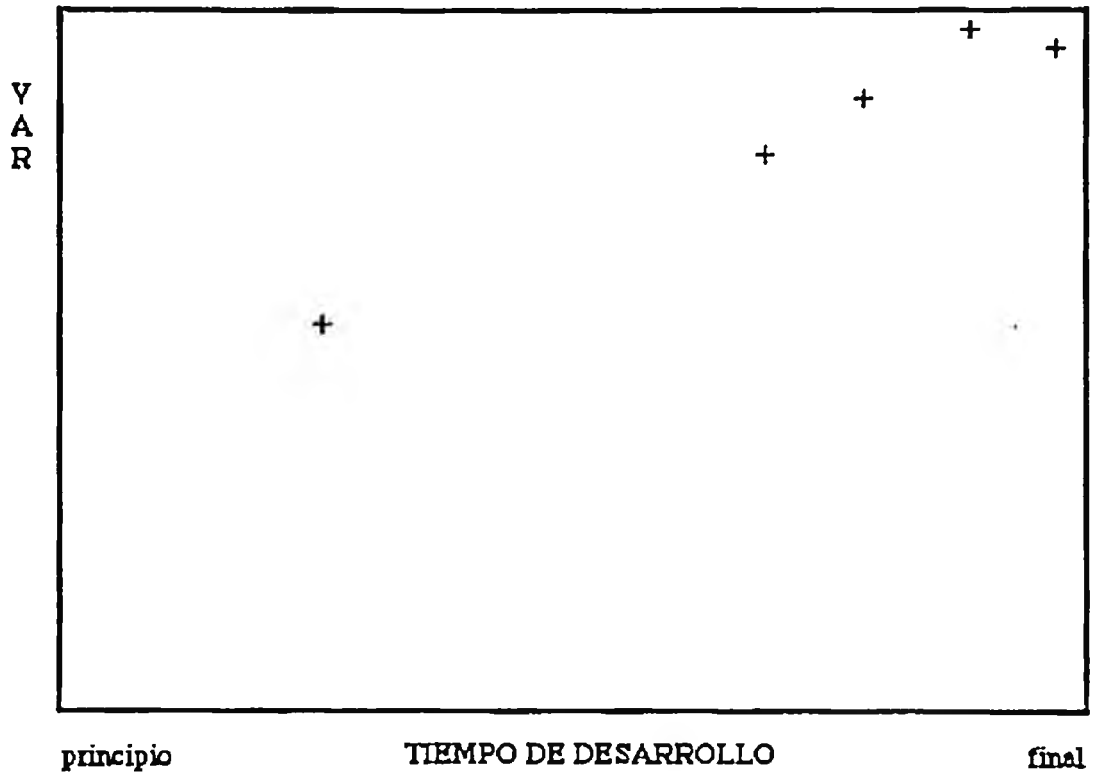


Figura C.3

Evolución de VAR (Caso Excepcional 1)

Figura C.4

Evolución de VAR (Caso Excepcional 2)

Apendice D:

Gráficos de Evolución de n_2

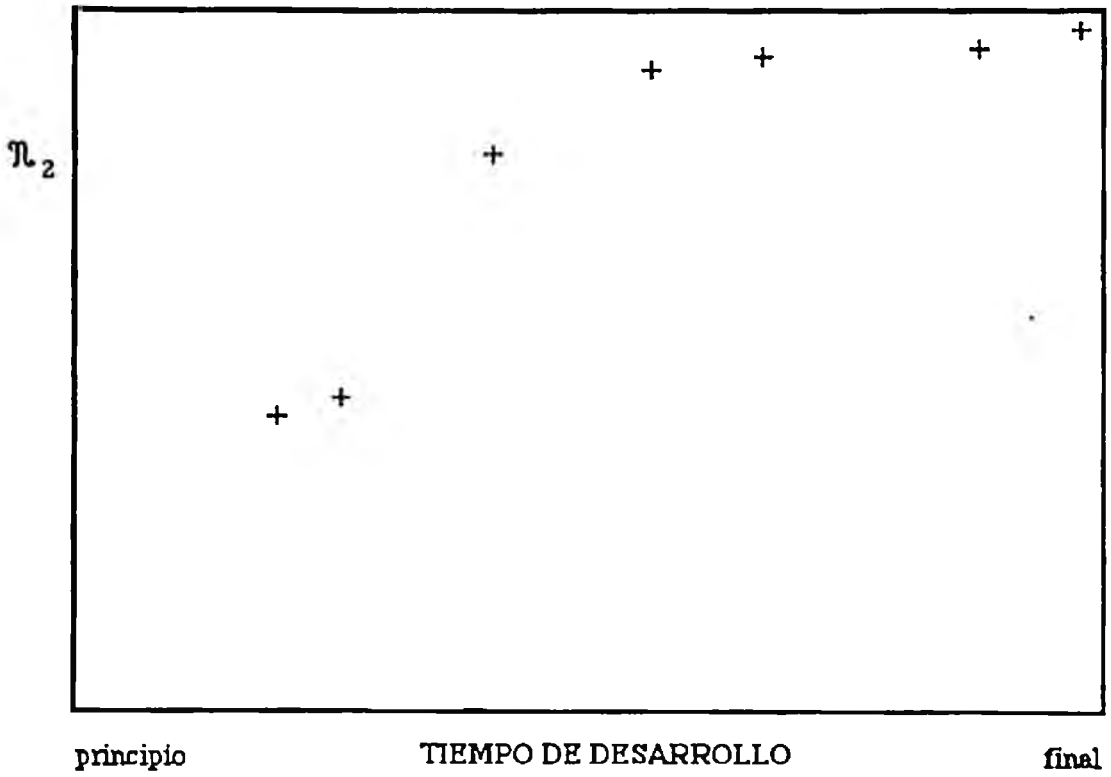


Figura D.1

Evolución de n_2 (Ejemplo 1)

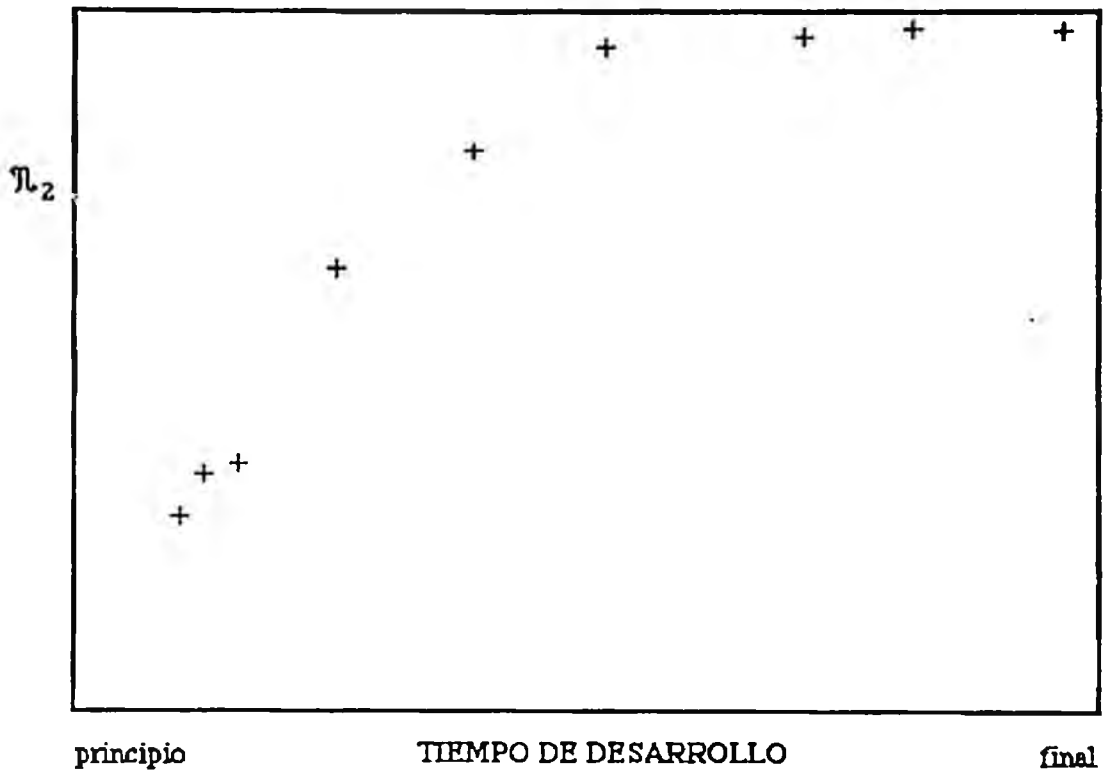


Figura D.2

Evolución de n_2 (Ejemplo 2)

Apendice E:

Datos del Estudio EXPER-1

Indice de Programas	E	S	VAR	n_2
1	53,8	401	54	85
2	17,9	357	84	124
3	18,7	243	49	76
4	66,8	585	48	71
5	22,8	316	52	91
6	23,4	416	62	110
7	27,9	435	63	96
8	33,5	311	59	136
9	30,7	318	51	82
10	31,0	332	77	128
11	50,1	631	109	146
12	18,0	349	50	126
13	19,0	523	119	159
14	65,0	662	118	150
15	21,9	215	62	121
16	27,1	374	105	143
17	18,2	407	76	112
18	48,8	519	54	86
19	14,2	466	52	167
20	18,7	369	57	84
21	10,4	368	89	183

Apendice F:

Datos del Estudio EXPER-2a

TAMAÑO DEL PROGRAMA EN LOC

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	71	103	259	267										
2	73	88	126	182	238	309	344							
3	54	91	220	279	347	398								
4	70	99	222	304	371	412	416	517						
5	32	51	52	153	258	302	375	385	411					
6	66	70	214	227	258	356	419							
7	31	56	100	121	192	263	370	381	381	492	510	513		
8	57	66	212	258	316	389	390	398						
9	58	77	230	236	306	416	456	472	516					
10	79	120	273	341	377									
11	51	51	64	154	231	299	317							
12	63	71	197	284	256	345	456	489	525					
13	54	70	426	223	325	327								
14	54	63	74	127	253	315	342	350						
15	40	187	325	551										
16	22	44	61	115	199	271	295	347	450	454				
17	97	110	85	160	217	349	388	394	433	411	491	492	444	443
18	50	71	162	266	345	360	400	406						
19	81	94	214	331	354	433	514							
20	65	80	204	254	307	460								
21	53	66	89	133	211	321	399							
22	63	77	213	281	343	441	495	493	498					

TAMAÑO DEL PROGRAMA EN LOC

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	55	68	196	297	365									
24	62	102	329	380	438									
25	49	115	141	302	386	379								
26	300	314	421	484										
27	52	88	221	288	387	397	533							
28	45	83	173	266	335	513	567	548						
29	63	86	211	292	314									
30	42	57	114	224	300	380	365	459	524					
31	29	96	177	264	356	448	461							
32	29	43	73	171	239	273	371	487	438	488	618	618		
33	46	65	217	410	482									
34	58	102	124	205	387	344	532	548						
35	36	81	89	127	392	321	388							
36	72	83	188	303	394	487								
37	66	76	244	389	481	513								
38	92	103	209	423	448	551	633	656						
39	51	99	238	250	379	518								
40	43	79	274	185	294	427	523							
41	46	54	177	234	238	266	249	335	346	371	428	444		
42	59	60	287	311	382	567	768	784						
43	74	83	236	370	484	411	414	448						
44	42	63	133	181	216	306	349	451	454					

UAR

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	43	62	65	67										
2	36	38	43	44	43	44	48							
3	33	52	55	57	56	56								
4	41	47	50	52	53	53	53	60						
5	22	28	30	39	44	50	50	50	49					
6	51	57	59	59	55	59	60							
7	23	37	52	52	57	57	57	59	59	62	62	62		
8	36	46	49	52	50	52	53	53						
9	38	54	68	73	88	89	89	92	89					
10	53	86	97	114	105									
11	45	44	49	52	53	56	55							
12	39	41	51	52	53	56	62	62	65					
13	34	55	55	68	61	61								
14	45	48	56	58	69	73	74	76						
15	21	34	33	41										
16	23	31	40	41	46	46	44	44	44	44				
17	48	68	58	71	72	72	72	71	73	71	73	73	74	74
18	37	42	49	56	59	58	60	65						
19	55	64	65	66	67	68	66							
20	44	55	62	68	68	68								
21	38	42	51	55	55	56	61							
22	39	54	60	62	65	65	68	68	68					

UAR

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	39	54	60	62	65	65	68	68	68					
23	24	33	39	40	45									
24	20	23	39	39	40									
25	41	45	51	53	67	70								
26	64	64	66	70										
27	32	54	61	65	67	68	71							
28	41	53	57	58	62	71	72	72						
29	43	51	55	53	53									
30	29	41	42	46	54	61	62	63	63					
31	24	52	56	65	66	68	69							
32	20	34	46	50	50	51	59	62	63	50	53	53		
33	33	43	45	50	63									
34	77	95	100	110	113	113	110	110						
35	27	37	46	55	65	68	70							
36	52	59	61	63	66	66								
37	43	52	56	60	62	62								
38	67	73	76	83	83	90	89	90						
39	38	61	62	63	65	65								
40	19	44	53	46	43	47	57							
41	30	43	46	48	47	47	46	46	47	47	47	46		
42	22	27	28	28	42	42	48	48						
43	44	59	70	74	77	79	83	76						
44	26	44	45	48	50	42	46	47	46					

E

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2,7	9,7	16,2	17,2										
2	4,5	6,3	10,8	13,5	16,5	21,3	23,9							
3	7,8	9,1	14,7	18,8	21,2	27,4								
4	4,3	6,1	13,4	17,7	21,8	25,6	29,8	34,8						
5	2,3	3,1	3,9	6,9	10,9	14,6	19,1	22,3	26,4					
6	4,4	5,9	10,6	13,1	15,8	18,8	21,8							
7	6,1	9,5	15,1	17,3	24,8	28,8	32,8	36,4	41,3	44,9	48,5	49,8		
8	5,3	7,8	8,1	12,6	15,6	19,2	23,2	27,8						
9	3,6	5,3	11,8	16,4	20,8	24,8	28,1	32,3	34,5					
10	7,8	19,3	22,4	25,6	27,7									
11	1,7	1,7	2,6	8,1	11,8	16,3	21,4							
12	3,2	4,3	9,3	12,3	15,6	20,1	23,8	28,1	34,4					
13	1,8	3,8	6,8	11,7	16,8	19,9								
14	4,8	4,9	7,2	12,4	15,8	19,7	24,1	25,8						
15	3,8	5,8	9,3	16,3										
16	3,8	4,6	8,8	10,6	14,1	17,3	20,6	24,5	28,8	30,6				
17	4,1	6,1	8,8	12,8	17,3	21,2	25,7	30,1	34,8	38,9	43,1	47,1	52,5	54,3
18	5,8	5,8	8,3	12,6	15,8	19,5	23,5	26,7						
19	4,6	7,8	11,9	16,9	21,4	25,6	28,9							
20	2,1	4,1	7,6	11,6	16,5	19,9								
21	2,8	2,8	4,1	6,6	9,1	13,2	16,6							
22	4,8	5,7	10,3	19,8	16,7	20,5	23,8	25,5	26,8					

E

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	4,8	5,7	10,3	13,8	16,7	20,5	23,8	25,5	26,8					
23	7,5	12,6	19,0	22,3	26,9									
24	2,5	3,7	8,8	12,8	16,8									
25	2,5	6,2	7,1	11,4	16,7	18,4								
26	10,6	14,6	18,4	24,1										
27	2,8	4,8	10,8	14,8	18,3	22,3	28,3							
28	4,4	8,3	11,3	14,4	19,4	25,6	27,7	31,3						
29	6,1	8,9	12,7	15,6	18,0									
30	4,3	5,8	9,6	13,3	16,4	20,4	23,9	27,2	30,2					
31	2,1	4,8	7,3	9,9	12,5	15,5	15,9							
32	2,3	3,3	6,1	10,0	13,8	17,0	20,8	24,6	28,4	33,8	38,2	40,2		
33	2,6	4,8	8,8	13,0	17,8									
34	2,3	2,8	5,5	8,8	13,0	17,3	21,8	23,3						
35	2,0	4,6	7,6	11,3	15,1	18,7	21,4							
36	4,3	5,2	8,8	12,0	17,3	19,6								
37	2,1	3,7	6,5	9,8	12,9	16,5								
38	4,3	6,1	8,8	14,1	18,2	22,9	27,1	29,8						
39	4,3	6,8	11,7	14,6	19,8	23,9								
40	3,4	6,1	13,7	16,7	22,0	27,7	32,1							
41	4,0	5,5	10,5	13,3	16,3	19,0	21,8	25,6	28,8	34,1	36,6	39,1		
42	2,3	4,3	11,1	16,7	20,8	24,8	30,4	37,2						
43	7,1	9,2	13,9	18,8	23,0	25,8	29,4	33,2						
44	3,5	7,0	10,0	13,0	15,0	18,5	21,5	25,5	30,0					

n_2

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	54	103	110	112										
2	39	41	64	76	77	80	55							
3	37	65	81	88	82	90								
4	45	53	88	94	101	103	103	115						
5	26	33	35	65	75	85	88	90	89					
6	54	60	85	85	94	103	110							
7	26	40	57	57	90	94	107	111	111	121	123	124		
8	43	53	80	83	86	90	95	95						
9	49	73	102	108	119	134	134	139	136					
10	65	103	137	163	157									
11	59	50	62	85	90	115	107							
12	43	40	77	81	86	96	100	105	113					
13	42	66	75	102	171	172								
14	51	53	62	85	109	119	110	122						
15	26	66	88	112										
16	26	34	44	66	75	83	85	87	90	90				
17	52	74	64	96	100	110	111	110	112	110	125	125	126	126
18	41	46	75	97	101	103	110	125						
19	59	60	86	95	96	100	101							
20	62	76	90	96	97	99								
21	44	40	57	71	89	105	110							
22	43	60	92	107	110	125	131	131	134					

n_2

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	28	37	65	71	88									
24	27	43	97	106	118									
25	43	73	79	101	118	102								
26	84	94	102	109										
27	37	68	86	97	110	105	115							
28	44	57	82	98	98	109	109	103						
29	52	59	81	83	84									
30	36	51	70	84	105	112	100	103	107					
31	39	74	95	100	109	112	113							
32	23	37	51	80	87	96	110	122	123	103	113	114		
33	37	48	77	100	443									
34	81	101	114	131	148	148	157	157						
35	32	42	51	70	100	90	161							
36	56	63	85	96	103	112								
37	47	56	92	97	106	100								
38	75	81	103	119	120	120	120	130						
39	42	66	107	99	110	120								
40	24	50	95	74	70	80	100							
41	35	47	71	81	80	90	81	86	84	86	89	92		
42	24	30	48	59	86	94	100	96						
43	55	74	90	106	127	131	135	119						
44	29	56	67	77	82	75	81	90	91					

Apendice G:

Datos del Estudio EXPER-2b

TAMAÑO DEL PROGRAMA EN LOC

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	47	134	140	203	817	228								
2	27	42	78	116	169	215	233							
3	42	51	171	220	271									
4	89	76	172	187	222	267	268							
5	25	56	57	160	197	254	287	269	262	258	300	304		
6	41	79	96	219	300	406	490	445						
7	26	32	74	100	149	269	366	546	628	644	645			
8	27	34	124	238	244									
9	35	43	146	272	424									
10	67	59	469	274	300									
11	54	67	159	227										
12	47	76	171	256	288	304								
13	69	79	226	291	424	422								
14	31	46	88	135	174	180								
15	89	89	197	299	320									
16	41	50	125	218	303	387	438							
17	57	54	61	189	373	392								
18	38	59	66	170	192	277	421	391						
19	63	70	176	366										
20	84	102	288	389										
21	43	60	142	175	204	270								
22	61	79	177	255	344	340								

TAMAÑO DEL PROGRAMA EN LOC

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	48	65	210	313										
24	70	142	169	298	347									
25	95	110	245	390	464									
26	74	90	220	369	392									
27	58	90	145	296	402	512								
28	59	67	144	247	364	365	325							
29	69	75	161	274										
30	55	55	131	272	352	365								
31	19	104	219	319										
32	27	51	95	96	161	329	355							
33	33	51	119	239										
34	70	98	274	376	524									
35	54	79	176	248	291									
36	57	61	154	278										
37	48	87	278	344	342									
38	63	84	234	353	483	543	565	536						
39	44	77	194	427	466	519								
40	72	87	167	287	359	383								
41	51	49	85	123	185	224	256	300						
42	63	63	70	378	387									
43	67	67	219	348	335									
44	26	59	197	254	416	444								

UAR

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	28	33	33	41	42	43								
2	2	13	27	29	28	28	27							
3	15	24	29	29	29									
4	17	30	31	30	31	32	32							
5	2	23	24	27	31	31	32	32	32	36	36	36		
6	14	52	54	64	66	66	66	65						
7	29	32	39	51	57	60	67	73	73	73	73			
8	12	14	22	24	24									
9	20	18	25	26	26									
10	13	28	32	36	36									
11	38	40	41	40										
12	12	25	31	37	37	36								
13	43	53	59	66	70	63								
14	9	17	20	28	43	43								
15	40	40	44	47	46									
16	14	23	30	34	25	39	40							
17	28	31	35	36	39	39								
18	32	35	39	40	45	41	50	53						
19	18	22	32	35										
20	55	71	80	85										
21	24	35	39	39	39	39								
22	44	54	55	58	59	57								

VAR

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	39	44	48	55										
24	24	39	41	41	41									
25	58	56	62	80	83									
26	62	78	79	95	71									
27	52	58	60	61	68	67								
28	51	52	45	49	48	58	48							
29	19	22	27	28										
30	16	16	16	21	22	22								
31	13	26	34	33										
32	6	29	61	62	65	57	48							
33	6	15	15	18										
34	78	90	95	95	98									
35	17	30	31	33	35									
36	21	22	25	25										
37	18	25	28	28	28									
38	43	51	65	75	80	84	84	85						
39	17	39	57	68	70	74								
40	27	36	35	39	47	46								
41	53	17	20	20	20	21	19	19						
42	16	16	17	18	18									
43	64	52	63	66	59									
44	13	24	28	32	34	37								

E

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1,5	3,5	5,3	8,3	10,8	12,3								
2	0,7	1,7	3,7	7,6	10,9	13,9	15,4							
3	4,0	6,0	10,0	14,3	18,3									
4	1,1	4,9	9,3	13,3	17,3	21,4	23,0							
5	0,3	1,8	2,1	4,2	6,2	9,2	11,7	14,2	17,2	20,2	24,2	27,7		
6	0,9	5,4	5,9	10,5	13,0	17,5	21,0	22,5						
7	2,8	6,2	10,6	14,2	17,5	21,2	27,4	31,2	35,3	39,9	41,4			
8	1,9	3,7	7,4	10,8	13,0									
9	4,2	4,2	5,7	8,5	12,7	16,5								
10	1,3	5,2	8,0	11,1	15,3									
11	1,7	4,4	7,2	13,5										
12	4,4	8,4	12,9	16,9	21,4	23,4								
13	1,8	3,3	6,1	8,6	12,6	15,9								
14	1,7	2,8	5,8	10,2	13,8	14,4								
15	3,4	4,9	6,4	9,9	11,6									
16	3,3	5,4	7,5	13,3	17,3	21,0	24,2							
17	2,7	3,8	6,4	11,1	17,3	18,7								
18	2,6	3,3	3,9	7,5	9,1	12,8	16,0	19,9						
19	3,1	4,3	8,7	13,5										
20	2,1	4,4	10,2	13,3										
21	1,9	3,2	5,7	8,2	9,3	10,8								
22	1,1	2,1	5,5	8,8	12,1	14,6								

E

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	4,4	7,1	11,1	15,3										
24	2,1	4,5	4,6	7,2	10,7									
25	2,3	3,9	8,0	11,9	14,2									
26	4,5	7,8	11,4	16,1	19,9									
27	4,5	6,0	9,5	14,5	18,0	22,3								
28	2,4	3,8	8,3	11,6	15,5	19,2	22,8							
29	2,3	3,2	5,7	8,7										
30	1,6	2,3	5,6	9,7	13,3	15,8								
31	1,4	3,5	6,2	10,4										
32	1,8	4,0	6,8	8,2	12,8	16,6	19,3							
33	3,8	5,9	9,2	13,2										
34	1,4	3,5	6,3	9,6	12,3									
35	3,2	4,8	8,7	12,6	15,2									
36	2,6	3,3	8,4	13,9										
37	1,2	2,6	6,3	9,8	10,4									
38	1,7	2,8	6,6	10,5	14,0	18,1	21,8	23,7						
39	1,9	3,5	6,4	10,3	12,9	16,5								
40	4,5	6,2	9,8	13,4	16,9	19,7								
41	4,0	5,5	9,0	11,5	15,5	17,8	20,3	25,3						
42	3,2	5,4	8,0	15,2	17,7									
43	4,1	5,6	9,4	13,6	16,5									
44	1,7	4,7	8,7	12,7	16,7	18,7								

n_2

Indice de Programas	Indice de Puntos Cruciales													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	23	58	104	144	150	151								
2	2	15	31	38	56	74	88							
3	18	29	53	64	84									
4	17	36	77	78	91	103	103							
5	2	25	26	73	87	88	95	93	93	103	105	104		
6	17	55	57	84	97	121	136	136						
7	32	36	46	59	65	95	115	136	154	157	159			
8	14	17	38	102	104									
9	26	25	44	73	98									
10	17	37	89	112	106									
11	52	66	85	103										
12	15	33	55	95	107	110								
13	50	69	95	123	152	144								
14	12	20	31	62	103	103								
15	43	43	64	94	100									
16	17	26	38	55	78	110	119							
17	25	36	48	85	110	110								
18	35	38	43	55	71	88	105	116						
19	23	27	49	88										
20	69	98	118	140										
21	28	41	93	101	108	118								
22	47	58	107	131	178	180								

n_2

Indice de	Indice de Puntos Cruciales													
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
23	41	46	94	133										
24	36	65	75	87	112									
25	52	60	104	145	188									
26	63	72	102	160	144									
27	60	66	102	124	156	167								
28	54	56	65	98	110	111	99							
29	25	31	54	91										
30	21	21	34	70	80	89								
31	10	70	90	100										
32	8	31	91	92	114	144	130							
33	9	17	42	89										
34	82	95	140	146	175									
35	17	34	75	86	106									
36	27	29	41	85										
37	10	20	90	106	106									
38	50	59	83	131	151	156	150	156						
39	10	42	95	153	150	171								
40	31	42	82	106	123	146								
41	55	20	27	30	70	81	04	96						
42	10	10	19	114	91									
43	71	63	100	131	131									
44	17	47	79	93	100	193								

Apendice H:

Datos del Estudio GRUPO-1

Indice de Programas	S	VAR	n_2
1	2418	211	413
2	1451	204	451
3	2049	325	588
4	632	84	144
5	1871	234	427
6	502	73	118
7	3395	646	1020
8	2769	297	729
9	1183	236	380
10	1776	223	440
11	2579	475	690
12	713	96	172
13	2247	397	666
14	817	161	369
15	843	133	182
16	1210	186	308
17	1122	161	295
18	690	195	295
19	2035	564	756
20	1108	327	517
21	4572	703	1073
22	1470	255	439
23	2383	483	761
24	3902	743	952

Apendice I:

Datos del Estudio GRUPO-2

Indice de Programas	S	VAR	n_2
1	481	96	135
2	534	90	135
3	515	81	126
4	401	69	106
5	572	86	123
6	615	98	150
7	613	94	139
8	819	96	144
9	744	104	161
10	773	129	173
11	604	107	155
12	537	111	157
13	647	77	135
14	594	138	181
15	584	92	135
16	589	104	146
17	757	109	142
18	643	87	126
19	454	100	140
20	626	86	129
21	627	95	140
22	507	102	150
23	464	74	104
24	588	91	139
25	585	125	165
26	557	87	137
27	613	76	111
28	533	62	101
29	927	97	124
30	420	86	134
31	676	109	153
32	434	85	130
33	270	56	82