An Empirical Investigation into Software Effort Estimation by Analogy

Christopher Schofield

A thesis submitted as partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

June 1998

Bournemouth University

Abstract

Most practitioners recognise the important part accurate estimates of development effort play in the successful management of major software projects. However, it is widely recognised that current estimation techniques are often very inaccurate, while studies (Heemstra 1992; Lederer and Prasad 1993) have shown that effort estimation research is not being effectively transferred from the research domain into practical application. Traditionally, research has been almost exclusively focused on the advancement of algorithmic models (e.g. COCOMO (Boehm 1981) and SLIM (Putnam 1978)), where effort is commonly expressed as a function of system size. However, in recent years there has been a discernible movement away from algorithmic models with non-algorithmic systems (often encompassing machine learning facets) being actively researched. This is potentially a very exciting and important time in this field, with new approaches regularly being proposed. One such technique, estimation by analogy, is the focus of this thesis.

The principle behind estimation by analogy is that past experience can often provide insights and solutions to present problems. Software projects are characterised in terms of collectable features (such as the number of screens or the size of the functional requirements) and stored in a historical case base as they are completed. Once a case base of sufficient size has been cultivated, new projects can be estimated by finding similar historical projects and re-using the recorded effort.

To make estimation by analogy feasible it became necessary to construct a software tool, dubbed ANGEL, which allowed the collection of historical project data and the generation of estimates for new software projects. A substantial empirical validation of the approach was made encompassing approximately 250 real historical software projects across eight industrial data sets, using stepwise regression as a benchmark. Significance tests on the results accepted the hypothesis (at the 1% confidence level) that estimation by analogy is a superior prediction system to stepwise regression in terms of accuracy. A study was also made of the sensitivity of the analogy approach. By growing project data sets in a pseudo time-series fashion it was possible to answer pertinent questions about the approach, such as, what are the effects of outlying projects and what is the minimum data set size?

The main conclusions of this work are that estimation by analogy is a viable estimation technique that would seem to offer some advantages over algorithmic approaches including, improved accuracy, easier use of categorical features and an ability to operate even where no statistical relationships can be found.

Acknowledgements

I would first and foremost like to thank Professor Martin Shepperd for his dedicated supervision and unwavering support, without which this thesis would never have got off the ground. Thanks Martin, I owe you a lot.

I would also like to record my thanks to several other people who have made contributions to this thesis. In particular, Dr. Frank Milsom, Michelle Cartwright, Colin Kirsopp, Austin Rainer, Tina Lepinioti, Dr. Liguang Chen and Jacqui Holmes.

A great deal of thanks must also go to Justine Tyler and my parents, who have supported me throughout, putting up with my thoroughly unsociable behaviour and late nights.

The research contained within this thesis has been supported by British Telecom and the Enterprise in Higher Education (EHE) initiative.

List of Publications

The publications listed below are based upon work presented in this thesis:

- Schofield, C. and M.J. Shepperd (1995). "Software Support for Cost Estimation by Analogy." ESCOM 95, Rolduc, The Netherlands.

- Shepperd, M.J., C. Schofield, and B. Kitchenham. (1996). "Effort Estimation Using Analogy." ICSE-18, Berlin.

- Schofield, C. and M. J. Shepperd (1996). "Estimation by Analogy: A Case Study." ESCOM 96, Wilmslow, UK.,

- Shepperd, M. J. and C. Schofield (1997). "New Techniques for Estimation from Function Points." IFPUG Fall Conference, Scottsdale, Arizona,

- Shepperd, M. J. and C. Schofield (1997). "Estimating Software Project Effort Using Analogies." IEEE Transactions on Software Engineering **23**(11): 736 - 743.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Most practitioners recognise the important part accurate estimates of development effort play in the successful completion of major software projects. Accurate estimates are not only necessary for tendering bids, where both over and under estimates can be financially disastrous, but also for monitoring progress, scheduling resources and evaluating risk factors.

The estimation of project effort however is far from easy. For one thing software projects are commonly one offs, which renders much of the past estimating experience difficult to use. Couple this with the complex human and political machinations of many software companies and the need for estimates, when little more than sketchy details are known about proposed systems, and you have a very poor basis on which to found estimates.

It is now four decades since the first attempts (Farr and Zargorski 1965; Nelson 1967) were made to capture and model the factors that affect software development effort. Unfortunately, the little evidence that is available suggests that for the most part, the industrial community is very slow at embracing research advances in software estimation technology (Lederer and Prasad 1993; Subramanian and Breslawski 1995). For example Heemstra (1992) reports the results of a survey of 598 Dutch software companies which found that while 50% captured data on completed projects, only 14% made any attempt to generate any formal models. To some extent this can be seen as a failure by researchers to address the real needs of the software community, who are under pressure to make estimates based on ill defined specifications and ever changing technology. Unfortunately, what the research community has to offer is estimation solutions that require clearly specified problems with measurable features (Kitchenham 1996).

Until recently the weight of effort estimation research has largely been focused upon the use of algorithmic models, where typically effort is expressed as a function of product size. A good example of such a model is Boehm's COCOMO (1981) which provides a number of equations that, it is hoped, adequately model the user's development environment. However, it is a major failing of this approach that it is dependent on quantifiable inputs and often not appropriate at the bidding stage of a project, when the most important estimates are often

required. Another serious problem has been the lack of consistent accuracy experienced when using these models (see for example Kemerer (1987) who reports absolute average errors between 85 and 772 percent for four popular cost models). In response to these problems many researchers (Vicinanza and Prietolla 1990; Karunanithi, Whitley et al. 1992; Mukhopadhyay and Kekre 1992; Samson, Ellison et al. 1993; Venkatachalam 1993; Jorgensen 1995; Serluca 1995; Prietula, Vincinanza et al. 1996; Gray and MacDonell 1997) are currently exploring a variety of non-algorithmic techniques (typically incorporating some 'machine learning' element). It is hoped that these will provide solutions more suitable for practitioners, together with a greater degree of accuracy than is currently being experienced.

The focus of this thesis is on one such machine learning technique known as estimation by analogy.

1.0 Motivation for Thesis

The potential benefits of accurately estimating development costs are large, especially when the vast amount of money spent on new and legacy software systems is considered; yet it is widely recognised (e.g. (Heemstra 1992; Lederer and Prasad 1993)) that few companies are proficient at estimating effort. The motivation for this thesis is essentially to provide the estimating community with a fresh approach to the estimation problem, which might complement present practices. The main reasons for this are:

i) *Poor results from algorithmic models.* Numerous empirical studies into the accuracy of algorithmic models have been published in the literature (for example (Golden, Mueller et al. 1981; Kemerer 1987)). Unfortunately, the over-riding trend is inaccuracy and inconsistency with average errors over 100% common. By exploring techniques other than algorithmic models it will be possible to build effort prediction systems that are not necessarily reliant on there being a strong statistical relationship present.

ii) *Too much research effort has been spent on algorithmic models to the detriment of other potential techniques* (Kitchenham 1996). Algorithmic models have absorbed the greater part of four decades of research effort in effort estimation; however, there is little tangible evidence of any improvement in accuracy or indeed usage. The suggestion, therefore, is to apply research effort to more diverse estimation techniques that might better address the problems experienced by practitioners

iii) *Methods more appropriate for early estimation are required.* As has been stated, a major problem with the use of algorithmic models is their dependence on quantifiable inputs. This often renders them ineffective during the early stages of a software project's conception. More appropriate approaches need to be found that can make estimates using the type of data that is present during the early stages of a project.

1.1 Research Objectives

The work described within this thesis is a practical investigation into the accuracy and efficacy of a non-algorithmic approach to the effort estimation problem. The technique, known as estimation by analogy, has received little attention from the software community and this work is an attempt to partially redress this imbalance with the following objectives:

i) *To investigate the viability of analogical reasoning for the purpose of estimating the required effort to complete software projects.*

ii) *To develop an automated tool that supports the functionality required to generate estimates by analogical reasoning.*

iii) *To validate the analogical reasoning technique on data taken from industrial environments.*

1.2 Scope of the Investigation

While this work could conceivably be applied to other software measurement problems such as the prediction of project duration or defect density, the focus of this thesis is exclusively on the prediction of software project effort. In reality, it is project costs rather than effort that we are trying to capture. However, for a number of reasons, such as:

i) work effort is easier to compare across different companies,
ii) production costs are often too sensitive to be made public,
iii) cost is determined largely by effort,

effort (measured in for example work hours) is used as a convenient proxy with the assumption that there is a calculable linear relationship between effort and cost. Although effort is by no means the only driver of project costs, it is usually by far the most significant. The definition of what constitutes effort varies widely between development environments

studied but is expected to include as a minimum, the effort expended during the requirements definition, design, coding and testing phases.

The term software project is not restricted to the development of new software but can also refer to maintenance or enhancement projects, which reflects the large amount of effort spent on such projects. However, hypothetical or educational projects such as those commonly carried out by students are not considered within this thesis.

During the course of the thesis the terms effort and cost will be used interchangeably, as will the terms estimation and prediction. This is commonplace within the literature.

1.3 Outline of Thesis

Chapter 2:

This chapter examines the general principles of effort estimation and in particular looks at the body of research on algorithmic prediction systems. It concludes that research into algorithmic models has reached a natural zenith with the use of simple statistical techniques. It also agrees with Kitchenham (1996) that too much attention has been focused on algorithmic models to the detriment of other potential techniques.

Chapter 3:

In response to the lack of convincing results from algorithmic models, chapter 3 examines a range of non-algorithmic approaches to effort estimation that are coming to the attention of researchers as possible answers to the effort estimation problem. It concludes that these new techniques can potentially offer a number of advantages over algorithmic techniques and that more research work is certainly warranted.

Chapter 4:

This chapter focuses on one particular non-algorithmic technique known as estimation by analogy or case-based reasoning, which has received very little attention in the effort estimation literature. It describes the development of the approach and construction of a software tool that facilitates estimation by analogy.

Chapter 5:

The software tool described in the previous chapters is now applied to eight industrial data sets to empirically validate the analogy approach. The results from the tool are compared to a commonly used algorithmic approach (stepwise linear regression) and conclusions are drawn about their relative accuracy. It is found that the analogy approach is very flexible and can be used in circumstances that prohibit the use of algorithmic models.

Chapter 6:

Chapter 6 describes a study of the sensitivity of the analogy approach when applied to four data sets. By adopting a pseudo time series analysis approach, questions about aspects of dynamic behaviour can be answered such as, is it sensitive to the addition of outlying projects and what is the minimum number of projects required before the technique becomes effective?

Chapter 7:

The final chapter summarises the preceding research work and concludes that estimation by analogy is a suitable alternative or complement to algorithmic modelling techniques. The contributions of the work to empirical software engineering are stated before limitations of the work are acknowledged and avenues for further work are explored.

Chapter 2

A History of Research Progress in the Development of Algorithmic Software Cost Models

2.0 Introduction

A brief history of software effort estimation is presented here to give context to the current state of the art, and as a prelude and justification for the research presented in later chapters. By critical discussion of all the major approaches proposed from the mid 1960s to the present, it is hoped to show the significant themes and developments that have shaped estimation research and practice.

Traditionally estimation practice has been divided between seven separate approaches (Boehm 1981). These are:

- Algorithmic Models: where mathematical models are used to represent effort as a function of one or more variables.
- Expert Judgement: where one or more 'domain experts' are consulted.
- Analogy: where historical project details are recalled for use in estimating a new project.
- Top Down: where effort is estimated for the whole project before being divided between its components.
- Bottom Up: where individual components are estimated and then the results aggregated.
- Parkinson: where the available resources determine the estimate.
- Price to Win : where the estimate is influenced by the need to win a contract or be first in the marketplace.

Of the seven, the last two, Parkinson and Price to Win are not really estimation techniques as such and should not have any involvement in the estimation process. Of the remaining five techniques, four (expert judgement, analogy, Top down, and bottom up) are usually considered as informal non-repeatable approaches, where a domain expert is normally

required. Only algorithmic models are independent of the availability of domain experts and can be seen as a repeatable process. It is for this reason that algorithmic models have been the primary focus of estimation research effort.

Algorithmic models attempt to represent the relationship between effort and one or more project characteristics. The main 'cost driver' used in such a model is usually taken to be some notion of the size of the software, for example the number of lines of source code, so that a very simplistic example model might be of the form:

$$\text{Effort} = \alpha * \text{size}$$

(Eqn. 2.1)

where α is a productivity constant. More sophisticated models introduce economies or diseconomies of scale coefficients. It is this type of model that is the focus of the remainder of this chapter. This chapter will show how 'the state of the art' in effort estimation research has matured over time. From the initial ad hoc use of productivity factors in modelling effort, through the introduction of complexity factors as a way of calibrating a model, on to the development of various function counts as alternatives to lines of code, and finally, after a series of empirical validations failed to demonstrate the accuracy of complex constrained models, to a more pragmatic approach where a simple unconstrained process for estimation is advocated based around simple statistical procedures.

2.1 Evolution of Effort Estimation Models

The earliest software cost models began to appear in the literature from the mid 1960's onwards, perhaps arising from the practice of measuring employee productivity (Mohanty 1981). Good descriptions of these early cost models can be found in (Mohanty 1981) and (Boehm 1981) while Jeffery (1991) provides a more up to date survey coupled with a method of categorising the models into three streams:

i) Economic.

Models developed from the economic studies of historical project data, typically utilising regression analysis

ii) Rayleigh.

Models based upon the Rayleigh-curve

iii) Function Points.

Models that utilise measures of a programs functionality providing some advantage over lines of code

Classifying effort models is useful as it allows the general principles of each to be discussed without the need to study the vagaries of each individual model. Since Jeffery's study, another category has emerged.

iv) Non-algorithmic.

This new class represents a significant movement away from the traditional algorithmic models and incorporates technologies such as neural networks, fuzzy logic systems and case-based reasoning[1]. It will be shown in the main body of this chapter, that research interest in non-algorithmic approaches is partly the result of the fact that mathematical modelling of effort has reached a research zenith in the practices suggested by researchers such as those involved with the MERMAID project (Kok, Kitchenham et al. 1990).

2.2 Early Economic Models - Pre 1976

Although it wasn't until the mid 1960's that people began to develop and disseminate software cost estimation models, people like Herbert Bennington (1983) were estimating the effort required to produce large scale software systems as early as 1956. As a member of the SAGE project that adopted many software management techniques that were subsequently ignored by their peers, he estimated that it would cost $5,500,000 to produce a 100,000 instruction system program.

'In other words, the time and cost required to prepare a system program are comparable with the time and cost of building the computer itself'

The SAGE costs as estimated by Bennington 'chilled' many of his peers as the common goal of the era was to produce instructions that cost less than $1 per line rather than $50.

---

[1] Although grouped under a single category, in truth each non-algorithmic approach could be considered in its own category as each is very distinctive.

2.2.1 The First Algorithmic Models

It was to be another ten years before the first models based upon statistical techniques were proposed, the earliest perhaps being the SDC (System Development Corporation) (Nelson 1967) and the Farr and Zagorski (1965) models. These early models were characterised by their emphasis on covering large numbers of productivity factors, at the expense of the models construction. The SDC collected 104 variables in all for 169 software projects and used a simple linear regression technique to build the best possible model for the data (Eqn. 2.2.)

MM = -33.63

+9.15(Lack of requirements) (0-2)

+10.73(Stability of design) (0-3)

+0.51(% Math instructions)

+0.46(% Storage/retrieval instructions)

+0.40(No. of subprograms)

+7.28(Language) (0-1)

-21.45(Business application) (0-1)

+13.53(Stand-alone program) (0-1)

+12.35(First program on computer) (0-1)

+58.82(Concurrent hardware development) (0-1)

+30.61(Random access device used) (0-1)

+29.55(Different host, target hardware) (0-1)

+0.54(No. of personnel trips)

-25.20(Developed by military organisation) (0-1)

(Eqn. 2.2)

Where MM stands for Man Months of effort and an attribute that is followed by figures in brackets requires the user to supply a value in the range indicated.

Boehm (1981) notes that, even when applied to the data from which it was developed, the model is not a very accurate predictor and further, that the algorithm is counter intuitive in that the constant is below zero. This gives the opportunity for an estimate of effort to be negative for small projects. Kitchenham (1990) also adds that the negative constant value implies that there are relationships amongst the input variables that result in the effort being over-estimated when all the variables are treated as independent. It is interesting to note that this model, while having some factors that can be seen as proxies of size (e.g. no. of subprograms and no. of personnel trips), has no definitive size parameter. The lesson soon

learnt from early attempts, such as these, was that multiple factor cost models tended to be unstable and that there would be little chance of porting these models to different environments due to the attributes selected.

2.2.2 Wolverton

Wolverton's (1974) approach to estimation assumes that software cost (measured in dollars rather than man-months) is linearly proportional to size. The four inputs to this model are:

a) number of object instructions
b) the degree of system difficulty (in the range 0 to 100 or easy to hard)
c) the novelty of the system (new or old)
d) the application area (control, i/o, pre/post-processor, algorithm, data management or time critical).

Wolverton provides 10 equations that model cost per object as a function of (b), (c) and (d) and thus the estimate becomes, the number of object instructions multiplied by the cost per individual object.

This model represents a step forward for a number of reasons. First, in differentiating between different application areas the model adopts a homogenisation strategy that is thus far unique. Second the input variables used are more intuitive than some of those used by his predecessors and third, the individual equations are kept simple with size and difficulty being the main inputs. The major criticism of the Wolverton model is that it adopts object lines of code rather than source lines of code as the input metric and that the output is measured in dollars rather than man-months. It is possibly because of this that the Wolverton model received less recognition than was perhaps deserved.

2.3 Later Economic Models - Post 1976

The economic models proposed from the late 1970's began to capitalise on the experiences, successes and mistakes of their predecessors. The individual algorithms often became simpler, single factor models with lines of code becoming the dominant expression of program size. Another element introduced to many of these models was a system of predictor or cost driver variables which were used to further refine estimates. Many of these variables have been identified as productivity factors and had been incorporated into the earlier cost models.

2.3.1 Walston & Felix

Walston and Felix (1977) developed their effort model from a database of sixty projects collected in IBM's Federal Systems division. They expressed the relationship between effort (E) and program size (S) in the following equation (Eqn. 2.3)

$$E = 5.2 * S^{0.91} \qquad \textbf{(Eqn. 2.3)}$$

It is interesting to note that the equation has an exponent less than 1.0, which means that there are economies of scale. That is, productivity increases as program size increases. This was one of the few studies to find this. Unfortunately the equation didn't adequately estimate actual effort for the projects from which it was developed. This led Walston and Felix to try incorporating more of the information available. The project database held information on a number of project factors and Walston and Felix used these to develop a productivity index. Sixty-eight factors were selected for analysis and refined by correlation analysis to twenty-nine that were found to be significantly correlated to productivity. The productivity index was calculated as follows:

$$I = \sum_{i=1} W_i X_i \qquad \textbf{(Eqn. 2.4)}$$

where:

I = productivity index for a project

W_i = question weighting, calculated as one-half $\log_{10}$ of the ratio of total productivity change (highest to the lowest) indicated for a given question i

X_i = question response (+1, 0 or -1) depending on whether the response indicates increased, nominal or decreased productivity.

They then used I in a regression equation to calculate productivity L which was in turn used to determine effort in the following equation:

$$E = S / L \qquad \textbf{(Eqn. 2.5)}$$

where S is measured in lines of code.

Walston and Felix were aware that some of the productivity variables might be correlated but pragmatically chose not to take this into account. It is likely that many of the variables are

almost certainly correlated, for example Conte et al. (1986) point to variables 15 - 18: "structured programming", "design and code inspections", "top-down development" and "chief programmer team usage" as being highly correlated, since a manager who encourages structured programming is equally likely to encourage all four practices. In the final analysis this model is important, not so much for the effort equations proposed, but rather, for the productivity factors used, many of which appear in later models from this period such as programmer experience and usage of modern practices.

2.3.2 COCOMO

Boehm's (1981) COCOMO (COnstructive COst MOdel) is without doubt the most widely studied of all the cost models presented here. The popularity of this model is due to its ease of accessibility (other contemporary models, such as Rubin's ESTIMACS (Rubin 1983) and Putnam's SLIM (Putnam 1978), remain unpublished) and ease of use. COCOMO is more than just a cost model in that it also incorporates models for development time and schedule, but the focus of this thesis chapter is with COCOMO's effort estimation. COCOMO presents three single factor effort equations that relate size, measured in Thousands of Delivered Source Instructions (KDSI) to effort in Man-Months (MM), see Eqn. 2.6.

$$MM_{nom} = a(KDSI)^b \qquad \textbf{(Eqn. 2.6)}$$

The values for a and b depend on the development mode (Organic, Semi-detached or Embedded) of the project and are summarised in table 2.1. Organic mode projects are small (typically less than 50 KDSI of new software) developed within stable environments with a relaxed requirement specification. A project is classed as Embedded if it is relatively large and is operated under tight constraints. This type of project will usually require a greater degree of innovation. The Semi-detached project lies between these two extremes.

Mode	Basic		Intermediate/Detailed	
	a	*b*	*a*	*b*
Organic	2.4	1.05	3.2	1.05
Semi-detached	3.0	1.12	3.0	1.12
Embedded	3.6	1.20	2.8	1.20

Table 2.1 COCOMO parameter values

It can be seen that all of Boehm's equations demonstrate diseconomies of scale, that is to say, the larger the product, the lower the productivity. Also, as might be expected, the size of the

diseconomy (i.e. coefficient *b*) is increased as the projects become more difficult to control (i.e. from Organic to Embedded). This is basically the extent of Boehm's basic model, which should be regarded as a quick and rough estimate.

The intermediate version of the COCOMO model allows the basic estimate to be adjusted by 15 cost drivers that are intended as refinements to the estimate to take account of local project characteristics. The cost drivers are spread across four categories (product, computer, personnel and project) and are assigned ratings[2] (six ratings from Very Low to Extra High) depending on the extent to which they affect the project in question. So for example, the first driver, *required software reliability*, can be rated 0.75 if reliability is a minor consideration, while if human life is dependent on the reliability of the product the rating would be Very High or 1.40. The basic effort estimate is then multiplied by each of these factors to determine the adjusted estimate value. Finally, the intermediate model recognises that when the different components of the project have been determined, it is likely that individual components will have different cost driver ratings. These components can be estimated individually using the process described above and collated to produce an even further refined estimate.

The philosophy behind the detailed version of COCOMO is that the more detail provided as input to a cost estimate, the more accurate the resulting estimate is likely to be. This is reflected in the introduction of phase sensitive effort multipliers, for example, a driver concerned with computer response time is unlikely to have much effect on the requirements phase. Also, Boehm suggests that the different levels of the project hierarchy (e.g. module, subsystem and system) should be treated differently to achieve more accurate estimates.

2.3.3 Critique of the COCOMO Approach

Considering the wide-spread popularity of the COCOMO model, it is remarkable that, to date, it has received very little critical attention in the literature[3]. Kitchenham and Taylor (1984) discuss some of the problems posed by COCOMO's underlying assumptions. First, they point out that all the input parameters, the model coefficients, the cost drivers and their ratings, and the distribution of effort across phases, were estimated by experts and thus may be subject to human bias. Second, and related, is that the number of variables that would need to be collected to perform a validation of all the model parameters is huge. And third, they draw our attention to the fact that there is a dichotomy between the value of the parameters for basic COCOMO and those of the intermediate and detailed equations. This further

---

[2] The values for which are pre-defined by Boehm.

demonstrates the subjective nature of the whole process and highlights the question of whether the parameters are transportable to other environments.

Kitchenham and Taylor (1984) also assessed the stability of COCOMO for a test project of 14,000 lines of code. They found that poor choice of development mode was more dangerous than a mis-estimate of size and further, that a mis-estimate of just one of the 15 cost drivers was potentially as dangerous as either of the above. This last point is highlighted by Conte et al. (1986) who demonstrate that the maximum estimate (i.e. all cost drivers at the highest value) can be 800 times the minimum estimate for a given lines of code count. Criticism of the cost drivers is also made by Kitchenham (1992) who re-iterates the findings of the MERMAID Esprit projects that: a) there is evidence that some of the cost drivers are not independent, b) that some of the cost drivers may not be relevant in all environments and c) that it is difficult to make sure that estimators evaluate the cost drivers in the way they were intended.

A further common problem associated with COCOMO, and indeed all of the models thus far discussed, is the need for subjective estimates of size and productivity drivers. The risk here is that poor input estimates will lead to misleading effort projections although Boehm does recommend that the inputs be re-evaluated as more information becomes available.

On a more positive note, Miyazaki and Mori (1985) report an attempt to calibrate the COCOMO model by tailoring it to their own environment. By following the prescribed tailoring methodology and pairing down the list of cost drivers they generate a model that, for their data, has a relative error of 20%.

2.3.4 Revisions to the COCOMO Model

Since the seminal 1981 COCOMO work, Boehm has twice developed variants of COCOMO to reflect advances in software technology. Ada-COCOMO (Boehm and Royce 1989) was first proposed in 1987 and was based on data collected from projects using both the Ada language and process model. Similarly, with the original COCOMO model, the input metric is size which is refined by a number of cost drivers. However, the nominal effort equation includes four weights that reflect the Ada design process before any cost drivers are used. The cost drivers remain true to the original model with the addition of four new classifications:

RUSE : The need for reusable code

VMVH : Volatility of virtual host machine

---

[3] The reasons for this are not clear-cut although it is plausible that COCOMO is pitched at the right level of complexity while

VMVT : Volatility of target machine

SECU : Security classification

However, the values for the drivers are revised from the original with the nominal rating no longer exclusively 1.0.

A further revision to the COCOMO model i.e. COCOMO 2.0, (Boehm, Clark et al. 1995; Boehm 1997) has recently been unveiled, that claims to be tailored to modern software engineering practices such as rapid development. The underlying principles of the original COCOMO again remain preserved, but a number of models are now proposed that represent what the authors consider to be the key market sectors of future software development. Unfortunately, the models still appear to be experiential (in terms of the cost drivers and equation co-efficients) and complex (in relation to calibration). One major positive change is that COCOMO 2.0 allows size to be expressed in terms of the available data such as Object and Function Points. This removes some of the guess work involved in the estimation of lines of code.

2.4 Rayleigh-Curve Models

The use of the Rayleigh probability curve for effort and staff level modelling was first suggested by Norden (1963), based upon an investigation he conducted into the build-up and decline of staff levels in engineering and development projects at IBM. The result was a series of manpower curves that he found to be similar in nature to the Rayleigh-curve (Fig. 2.1). The Rayleigh-curve is an example of an exponentially declining curve where a project is described with an initial sharp build-up of staff levels followed by a gradual reduction as the project evolves and people graduate to new projects leaving only maintenance staff.

Figure 2.1 : The Rayleigh-curve

remaining mechanistic and prescriptive enough for it to be appealing to software managers.

2.4.1 The Putnam Model

The Rayleigh-curve was first applied to software development projects by Putnam (1978) when he analysed a very large database of military projects. This led him to promote a Proprietary estimation tool, called SLIM. The specific equation that models the relationship between effort, size and time is given below.

$$S_S = c\,K^{1/3}T^{4/3}$$

(Eqn 2.7)

Where S_S is measured in lines of code (LOC). The constant c is a technology factor that takes into account the affect of numerous productivity related aspects of the project such as complexity. K represents the total life-cycle effort excluding requirements specification and T is the development time measured in years. The constant **c**, described by Putnam as a "funnel through which all system development must pass", can be assigned one of 20 values in the range 610 to 57,314 and as c increases so does productivity. The estimator also has control of the slope of the curve using what is known as the Manpower Build-up Index (MBI). The higher the MBI, the sharper the build up of staff at the start of the project.

Rayleigh-curve models place particular emphasis on the trade-off's between effort and development time. Basically, a reduction in development time leads to a severe increase in effort required, with the opposite being true for time increases. Rewriting equation 2.7 to look at the life-cycle effort gives:

$$K = (S_s/c)^3/T^4$$

(Eqn 2.8)

This has come under attack from a number of researchers (Parr 1980; Basili and Beane 1981; Jeffery 1987; Kitchenham 1992). In fact Putnam himself found from studying 750 projects that the relationship held for only 251.

2.4.2 Critique of The Putnam Model

Putnam's assumptions and model have been subject to a number of criticisms in the literature. Both Jeffery (1987) and Kitchenham (1992) have challenged the assumption that a reduction in the time-scales increases effort and vice versa. Kitchenham and Taylor (1984) found that

prediction of effort from the Putnam model is very sensitive to the mis-estimation of both size and the c value. Estimation of c, just one level either side of its correct level, can have a dramatic effect on the estimate; likewise with mis-estimates of the LOC value S_s. They also found that the amount of sensitivity experienced varied in proportion to the size of the c and S_s values being used (smaller values being more sensitive).

Parr (1980) criticises many of the underlying assumptions of the Putnam model including the fact that it disregards the initial stages (exploratory design /specification) of a project. He argues, that in reality every project starts with a certain staffing level and that these early activities have a major influence on the rest of the project. In an attempt to correct the problems with the Rayleigh-curve Parr offers a new curve known as $sech^2$ (Fig 2.2) which has a non-zero y-intercept.

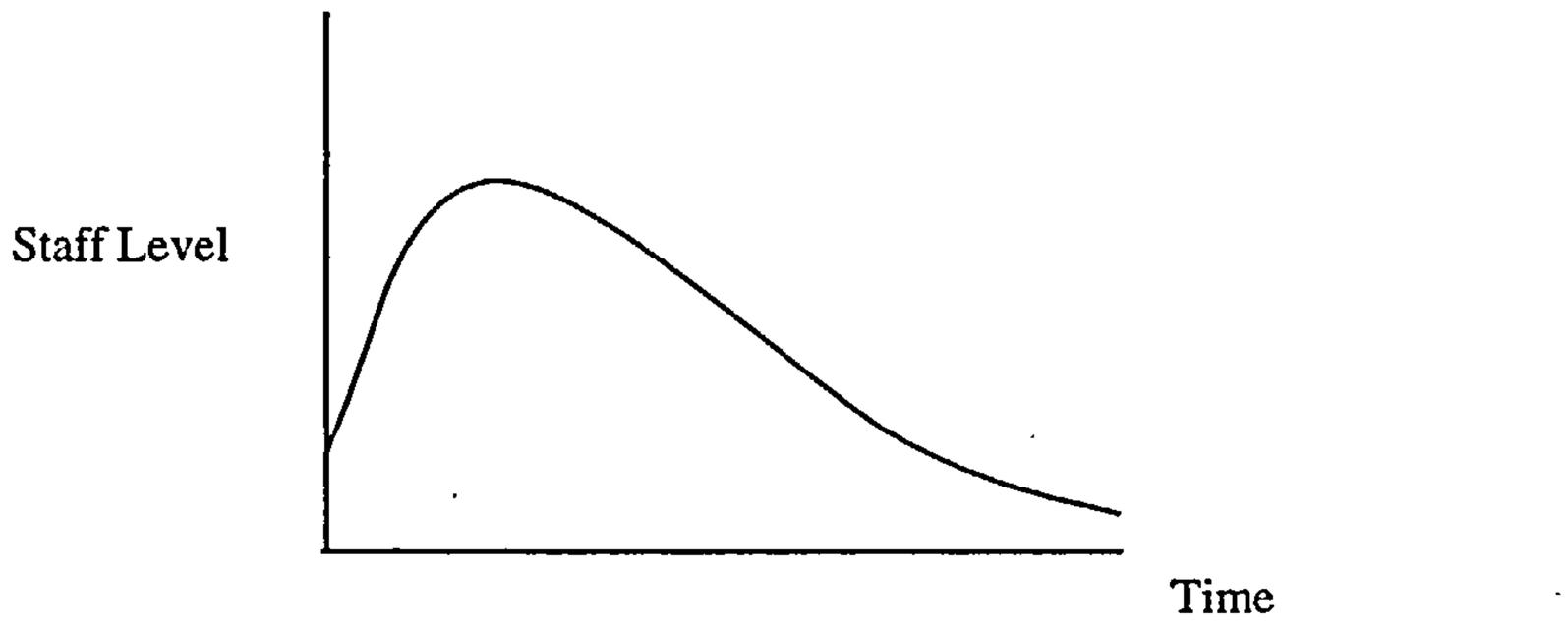


Figure 2.2 : Parr's $sech^2$ curve

Basili and Beane (1981) compared the Putnam and Parr curves along with a parabola and trapezoid on seven projects. They found that of the four, the Parr curve was the most consistent with the data and that Putnam's model fitted least well.

2.5 Function Point Models

The numerous problems associated with the use of lines of code when estimating effort have been well documented in the literature (DeMarco 1982; Jones 1986). In the search for an alternative input metric for project size Albrecht (Albrecht 1979; Albrecht and Gaffney 1983), proposed a measurement of system functionality that could be collected at the requirements documentation stage. This technique rapidly gained popularity throughout the 1980's and 1990's because of the overwhelming advantages it offers over LOC, such as availability earlier in the project life-cycle and language independence.

2.5.1 Albrecht's Function Points

Albrecht, first published the Function Point methodology while working at IBM in 1979. The Function Point is a dimensionless unit of system functionality that can be evaluated by the analysis of a project requirements document, classifying and counting the individual function types. Albrecht originally proposed four function types (Albrecht 1979): files, inputs, outputs and inquiries with one set of associated weights and ten General System Characteristics. By 1983 (Albrecht and Gaffney 1983) this was expanded (table 2.2) with an extra function type, three sets of weighting values and fourteen General System Characteristics.

Function Type	Simple	Average	Complex
External Input	3	4	6
External Output	4	5	7
Logical Internal file	7	10	15
External Interface File	5	7	10
External inquiry	3	4	6

Table 2.2 1983 function types and weights

The individual functions identified from the specification are weighted according to their complexity and the sum of the weighted function types becomes the Unadjusted Function Point count (UFP) for the system. Albrecht originally provided textual guidelines on how to rate the complexity of function types. More recently a more objective approach has been devised that relates complexity to the number of file, record and data element types referenced by the function type (IFPUG 1994).

1	Data Communications
2	Distributed Functions
3	Performance
4	Heavily Used Configuration
5	Transaction Rate
6	Online Data Entry
7	End User Efficiency
8	Online Update
9	Complex Processing
10	Reusability
11	Installation Ease
12	Operational Ease
13	Multiple Sites
14	Facilitate Change

Table 2.3 General System Characteristics (GSC)

The next step is to assess the extent to which the fourteen General System Characteristics (Listed in table 2.3) impact on the projects development environment. Each characteristic can be rated from 0 - *no influence* to 5 - *strong influence.* The sum of all the characteristics is then modified (Eqn. 2.9) to become the Value Adjustment Factor (VAF) in the range 0.65 - 1.35.

$$VAF = 0.65 + 0.01\sum_{i=1}^{14} GSC_i \qquad \textbf{(Eqn. 2.9)}$$

And that in turn is multiplied by the UFP to create the Adjusted Function Point (AFP) count. Thus the AFP value will be within ±35% of the original UFP figure.

2.5.2 Critique of the Function Point Approach

As already mentioned, the use of Albrecht's Function Points is considered to have a number of advantages over LOC as an input to a cost model. Not least of these is the fact that the function size can be counted, rather than estimated, at an earlier stage in a project's life-cycle. However, many researchers have expressed concern over the underlying philosophy of Function Points, while others have empirically validated the ability of Function Points to predict effort on different data sets with varying degrees of success.

Problems have been reported by researchers investigating inter-rater reliability, that is, the potential problem of different Function Point counters generating dissimilar results for the same system. This problem is accentuated by the difficulties of automating the collection process. A study by Kemerer and Porter (1992) reported variations in the range of counts, over 3 case studies, to be around 12%. Low and Jeffery (1990) found within- organisation variation to be within 30%. These both confirmed Rudolph's (1983) findings that Function Points were estimated within 30% of the mean.

Function Points represent an experiential approach to effort modelling. That is, the model components are derived by expert opinion and by trial and error. This leaves Shepperd (1994) expressing surprise at the "non-linearity, indeed arbitrariness" of the weightings for the function types (e.g. 3 - 4 - 6) and the breakpoints when determining function type complexity (e.g. 1-4, 5-15, ≥16). He also notes the possible conflict of interest between this "debate and trial" approach and Albrecht's justification for the weights as "numbers reflecting the relative value of the function to the user/customer". The value of a function from a user's point of view doesn't have a direct influence on project costs. Symons (1988) is another who cannot rationalise some of the differences between weighting values. He suggests that a "more objective assessment of the weights seems advisable". Another problem reported by Kitchenham and Kansala (1993) and by Jefferey and Stathis (1993) is the lack of independence between the counts for some of the function types. The use of inter-dependent inputs to an estimation model can lead to the repeated capture of the same underlying phenomenon, artificially enhancing its overall effect.

Once the AFP count has been calculated it is used either as an input to a cost model, perhaps using regression techniques, or is converted to LOC for input to a model such as COCOMO. This conversion can either be through one of the published conversion rates, see for example (Behrens 1983), or preferably by the collection of historical LOC and Function Point data so that the conversion rate can reflect local conditions. A number of studies have focused on verifying the assumption that Function Points are strongly correlated with effort (Albrecht and Gaffney 1983; Kemerer 1987; Desharnais 1988). As might be expected Albrecht reports a strong positive correlation ($R^2 = 0.87$) using the data set from which Function Points were developed. However this is placed in perspective by Knaff and Sacks (1986) who point out that without three particularly large projects (R^2 is known to be sensitive to outlying points) the R^2 is only 0.42. Kemerer and Desharnais's findings are more typical of the general pattern. The results they got from correlating AFP to effort on two independent data sets were an R^2 of 0.55 and 0.54 respectively. It is interesting to note that in both these experiments, the R^2 values for UFP counts were not significantly different from the AFP values, calling into question the benefit of the VAF.

Many researchers see the advantages to be gained from calibrating Function Points to the environment in which they are being collected. This enables companies to produce cost estimates initially based on informal analogy, and later, when the historical database has grown sufficiently, by use of statistical approaches. However, the calibration of Function Points counts poses a considerable problem when the five Function types and their three weights, the fourteen General Systems Characteristics and the coefficients (Productivity function α and perhaps exponential β) of the cost model are considered. For this reason Shepperd (1994) suggests that calibration be kept simplistic with adjustments to the coefficients of the model.

2.5.3 Adaptations to Albrecht's Function Points

Mark II Function Points (Symons 1988; Symons 1991) were proposed by Symons as an alternative to Albrecht's approach. Driven by some of the shortcomings of Function Points outlined above, Symons adopted the view that a system is made up of a collection of logical transactions, as opposed to delivered customer functions, each having an input, process and output component. The Unadjusted Function count for Mark II (based upon data collected for twelve systems) is defined as :

$$UFP = 0.44N_i + 1.67N_e + 0.38N_o \qquad \textbf{(Eqn. 2.10)}$$

where N_i is the number of input data element-types

N_e is the number of entity-types referenced

N_o is the number of output data element-types

Next the Adjusted Function Point count is determined using a similar set of system characteristics to Albrecht's, with the addition of six new GSC's (table 2.4), the twentieth representing any characteristics the user feels should be included.

15	Requirements of Other Applications
16	Security, Privacy, Auditability
17	User Training Needs
17	Direct Use by Third Parties
19	Extraordinary Documentation Requirements
20	*Client Defined Characteristics*

Table 2.4 Symon's new General System Characteristics

When Symons attempted calibration of the 20 GSC's he found that for some of the factors, a coefficient of 0.005, rather than Albrecht's mandatory 0.01, was more accurate, which contradicts the notion that Function Points are language independent.

2.6 Empirical Validation of Cost Models

While researchers continue to expose the strengths and weaknesses of the underlying theory behind cost models, to the pragmatic software manager one of the major justifications for the use of cost models is the level of accuracy (s)he can expect. The adoption and appropriate use of any one of the models described above requires a great deal of investment in collection and training time. Thus, some indication of the relative accuracy of different techniques when applied to data other than that which they were cultivated from is an essential research area. Unfortunately the number of empirical validations of cost models is very inadequate due to the lack[4] of available historical data.

It should be noted that there are a number of factors other than accuracy that should be considered in the validation of a cost model, such as the availability and objectivity of the input parameters, robustness of the model etc. However for the purposes of research presented in future chapters, attention will be directed solely on studies of model accuracy.

2.6.1 Validation Criteria

A number of validation criteria are proposed in the literature (Boehm 1981; Conte, Dunsmore et al. 1986; Miyazaki 1993; Miyazaki, Terakado et al. 1994) and the most commonly used are

[4] There are various reasons for the lack of data, such as, many companies being reluctant to publish data for political reasons. However, the major barrier is the amount of effort required to collect the appropriate data.

briefly discussed below. Note, Eact is actual effort, Epred is predicted effort and n is the number of projects.

(i) Total Error - $\sum_{i=1}^{i=n}(Eact - Epred)$

Also commonly known as the sum of the residuals, this measures the total difference between actual and predicted values. An obvious drawback with this measure is that it provides no indication of the relative size of individual errors. However, when assessed as part of a global strategy, where estimates are managed across the range of a company's projects, a total error of zero would be desirable (Kitchenham and Linkman 1997).

(ii) Mean Percentage Error - $100/n \sum_{i=1}^{i=n}\left(\frac{Eact - Epred}{Eact}\right)_i$

The lack of attention to relative error size using total error discussed above is overcome with mean percentage error, a measure that takes account of the average size of estimate errors relative to the size of the project actuals. The major drawback with this measure is that it includes the direction of the errors, thus under-estimates and over-estimate can cancel each other out.

(iii) Mean Magnitude of Relative Error (MMRE) - $100/n \sum_{i=1}^{i=n}\left(\frac{|Eact - Epred|}{Eact}\right)_i$

MMRE is an indicator of the average error given by a prediction system. It differs from mean percentage error in that it takes the mean absolute relative error value and thus ignores the direction of the error. As a result information on the overall degree of over or under-estimation is lost. On the plus side, it can give a better indication of errors associated with an individual estimate.

(iv) Balanced Mean Magnitude of Relative Error (BMMRE) - $100/n \sum_{i=1}^{i=n}\left(\frac{|Eact - Epred|}{\min(Eact, Epred)}\right)_i$

A major problem with MMRE is that it is not symmetrical (Miyazaki 1993), i.e. while under-estimates must be between zero and one, over-estimates are unbounded, thus the measure is biased towards prediction systems that under-estimate. BMMRE overcomes this problem by dividing the absolute error by whichever of the actual and estimate value is the smallest.

(v) Prediction at level n or Pred (n)

Pred(n) measures the percentage of estimates that are within n% of the actual values. Conte et al. (1986) suggest that n should be set at 25% and that a good prediction system should

achieve this accuracy level 75% of the time. The drawback with Pred(n) is that it provides no indication of the accuracy of estimates that don't fall within n%.

(vi) Adjusted Coefficient of Determination - Adj R^2

Adj R^2 is an indication of the amount to which the independent variables, the inputs to a prediction system, explain the variation seen in the dependent variable, i.e. effort. A value of Adj R^2 tending towards one demonstrates a prediction system where a change in effort is effectively explained by a change in the independent variable(s). An Adj R^2 value tending towards zero demonstrates a system where little of the variance is accounted for. The Adj R^2 is generally used in preference to the raw R^2 value because it compensates for the introduction of extra variables in multiple regression and allows comparison between models with differing numbers of variables.

Various objective criteria for judging model performance have been proposed and used by validation researchers. Unfortunately, the problem is that no one individual measure provides an overall picture of performance and further, some techniques have model bias. Thus combinations of validation techniques would seem more useful. The value of having a number of independent techniques can be seen in the variety of ways model output can be used by different organisations. For example, many companies leave the problem of estimating to the individual project manager who is only interested in the difference between estimate and actual effort. Other companies use the manager's estimates at an organisational level. Kitchenham (1997) discusses a portfolio management strategy where uncertainties and risk are managed across the range of a company's projects. She advocates the use of the sum of the residuals (or total error) to monitor how well an estimation model used in this way behaves. If the sum of the residuals is close to zero, the model is considered well behaved, even if individual projects have large over or under-estimates. If the sum of the residuals is different to zero, new estimates with the model can be corrected by adding the mean of the residuals. Further evidence of the disparity between performance indicators is provided by Schofield (1997) who tested four performance indicators (total error, MMRE, BMMRE, Pred(n)) on three estimation techniques (least squares regression (LSR) and two analogy based approaches - see Chapter 4) (see table 2.5). He found that all three techniques were considered 'most suitable' under at least one of the performance indicators and thus selection of performance indicators should reflect the goals of the estimator.

Indicator	LSR	Analogy 1	Analogy 2
Total Error	(1)[5] 8.56	(3) 1494.92	(2) 925.79
MMRE	(3) 0.86	(1) 0.39	(2) 0.51
BMMRE	(2) 95.65	(3) 97.99	(1) 84.86
Pred(25)	(=2) 44%	(=2) 44%	(1) 55%

Table 2.5 Comparing performance indicators

2.6.2 Published Validation Research

Empirical studies of cost models (whether validating individual models or comparing the performance of different models) rely on the collection of historical project information so that the estimates of project effort generated can be compared to the actual effort figures. These estimate errors are then usually collated and, in the more recent surveys, represented using one or more of the performance indicators described above.

Perhaps the earliest independent validation study was that of Golden et al. (1981), who looked at Putnam's SLIM and more specifically the duration and effort estimates produced for four projects undertaken at Xerox. While the overall results were promising, the variance of individual estimates coupled with the size of the data set, raises doubts over whether this level of performance could be sustained for a larger sample of projects. However, a study of staff loading over one of the projects lent some credence to the use of a Rayleigh-curve. A second evaluation of the Rayleigh-curve was carried out by Wiener-Ehrlich et al. (1984) who used four data processing projects to again test duration and effort estimates. Generally they found that the Rayleigh-curve estimated the phases reasonably well, but tended to underestimate the maintenance phase until they defined maintenance as solely 'corrective' (Lientz and Swanson 1980). A further study of the Rayleigh-curve, through the SLIM model, was conducted by Jeffery (1987) on a database of 47 data processing projects. He concluded that their was no support for the contention that productivity declines as time increases.

Kitchenham and Taylor studied both the SLIM and the COCOMO models initially on 20 projects (Kitchenham and Taylor 1984) and then 33 projects (Kitchenham and Taylor 1985) taken from ICL and British Telecom. They found large discrepancies between the actual and estimated effort values when using the basic COCOMO model and that the collected data did not conform to the Rayleigh-curve model. They concluded that both models required

---

[5] The values in parenthesis indicate a rating of performance indicator 1-best, 3-worst

calibration before they could be used sensibly. Further they suggest that estimates could be improved by using additional information as it becomes available.

Another evaluation of COCOMO was carried out by Miyazaki and Mori (1985) who calibrated Boehm's intermediate model to a set of 33 application software projects. Without calibration they found that COCOMO overestimated effort; with calibration they found they obtained a better fit in terms of Pred(20) than Boehm did on his data set. Their revised nominal effort equation became:

$$MM_{nom} = 2.15.(KDSI)^{0.94} \qquad \textbf{(Eqn. 2.11)}$$

Note that the Miyazaki and Mori equation is opposed to the idea that software projects exhibit diseconomies of scale. They also discarded three of Boehm's cost drivers (Virtual Machine Volatility, Analyst Capability and Main Storage Constraint) to improve the model.

An influential comparison of four popular cost models was carried out by Kemerer (1987) who collected information on 15 data processing projects. The four models under the spotlight were SLIM, COCOMO, Function Points and Estimacs and a summary of the results obtained in terms of MMRE and Pred(25) is shown in table 2.6.

Model	MMRE	Pred(25)	R^2-After Calibration
SLIM	771.87 %	7 %	87.8 %
COCOMO - Inter	538.82 %	0 %	59.9 %
Function Points	102.74 %	33 %	55.3 %
Estimacs	85.48 %	22 %	13.4 %

Table 2.6 Results from Kemerer data set

Both COCOMO and SLIM consistently overestimated effort and performed considerably less well than the other two models, Function Points and Estimacs which, it should be noted, were developed in similar environments to the data in this study. However, when Kemerer calibrated each model by using Albrecht's technique of using the estimates as the independent variable in a regression equation, he found that the SLOC models SLIM and COCOMO produced the best 'fit' in terms of R^2. Another important result of Kemerer's work was that the productivity factors used in the COCOMO and FPA models added virtually nothing to the results.

2.7 Current State of the Art?

The lessons learnt over the past thirty years research into effort estimation models (and more particularly algorithmic models of the type that have been discussed throughout this chapter) has recently culminated in a more mature and appropriate approach to algorithmic cost modelling, namely the MERMAID[6] approach.

The MERMAID project was a four year ESPRIT II initiative that had the goal of 'improving support for estimators in the area of software sizing, effort estimation, risk analysis, resource monitoring and progress monitoring' (Kok, Kitchenham et al. 1990). By analysing the problems associated with former models, particularly the lack of accuracy and the fact that environmental relationships are rarely constant, the MERMAID consortium were able to specify an approach which they believed brought together the best practices in cost estimation.

In brief, the MERMAID approach advocates the calibration of models to the environment in which they are to be used. This, they believe, is best achieved by using statistical methods such as stepwise regression that can create models that evolve with the environment and provide stable and unbiased estimates. A further initiative is the use of phase based input variables in a attempt to eliminate estimated inputs, the use of which represents a major criticism levelled at many of the models previously proposed. The model's input parameters are used as and when they become available, and the user is not restricted to any single input parameter as with other models. Another important feature of the approach, and associated tool, is the ability to statistically analyse the model in terms of a number of performance indicators such as MMRE, Pred(25) and R^2.

The question is, why are the approaches outlined above better and more likely to be accurate than other algorithmic models? The answer lies in the simplicity and flexibility of the approach and in the way MERMAID encourages the building of locally based models. First, it has been shown that accuracy increases when data is separated into more homogenous clumps (Gulezian 1991). Second, the phase-based approach allows an organisation to create estimates using different models at different stages in the life-cycle. These models are continually improved as more data becomes available. Third, the method does not force a change to operating procedures or data collection practices.

---

[6] P2046 MERMAID (MEtrication and Resource Modelling AID)

As the MERMAID consortium point out, their approach while incorporating many of the best practices available is not without flaws. For example a major obstacle to the use of MERMAID is that it demands that companies collect a substantial amount of historical project data. This is obviously going to take a great deal of time and effort for many companies. MERMAID's solution is to include facilities for a form of estimation by analogy that requires substantially less data[7] and is not as susceptible to outliers.

The MERMAID approach has so far been subject to few independent examinations. One study by Campobasso et al. (1995) looked at the relative accuracy of following the MERMAID approach, when compared to Intermediate COCOMO and to a calibrated Proprietary cost estimation tool. Two data sets were collected of 26 and 46 projects respectively, and the results are reported below (table 2.7.)

Data set	Model	MMRE	Pred(25)
Data set - 1	COCOMO	135 %	0 %
	Proprietary Tool	71 %	20 %
	MERMAID	18 %	82 %
Data set - 2	COCOMO	41 %	24 %
	MERMAID	36 %	45 %

Table 2.7 Results from an analysis of MERMAID, COCOMO and a proprietary prediction system

The MERMAID approach helped to build two models that were both accurate and superior in performance to the other techniques studied. The authors concluded that simple local models, derived from local data, make better predictions than general complex models. The results, at least for the data sets under study, are a vindication of the MERMAID approach.

2.8 Summary

The most important themes and experiences of thirty years of effort estimation research have been discussed in this chapter. This chapter has attempted to show how the present level of research maturity has been obtained through a great deal of trial and error, and how present estimation practice and research goals have been shaped by the past.

---

[7] Optimally between 4 and 9 projects.

Research into repeatable methods of estimating effort began in the mid 60's in an ad hoc fashion where great numbers of productivity factors were collected and analysed together to create large complex algorithms. From this early work researchers were able identify the most important factors and began to create models based on the LOC measure of program size using economies or dis-economies of scale as appropriate. Some models such as COCOMO allowed the estimator to 'calibrate' original estimates by applying a number of productivity drivers. A significant break away from the use of estimated LOC was started by Albrecht who proposed a measure of function size that could be counted earlier in the project life-cycle. However, the Function Point metric, like LOC, remains relatively controversial. The 80's saw a series of empirical studies of the accuracy and validity of some of the more important models. However, the results obtained were less than encouraging. The net result of all this is that most researchers now agree that simple, unconstrained statistical approaches such as stepwise regression are the best way to create models that are dynamic, environment independent and testable.

The next chapter will go on to describe how estimation researchers have begun to examine non-algorithmic approaches in a search for more suitable prediction systems and increased levels of accuracy.

Chapter 3

Recent Research Directions: Non-Algorithmic Estimation Techniques

3.0 Introduction

As was stated in the previous chapter, the main thrust of research work in the field of software effort estimation has been in the development of algorithmic models. Unfortunately, the algorithmic approaches have been unable to demonstrate consistently adequate results, with errors of 100% or greater typical even after model calibration (see for example (Conte, Dunsmore et al. 1986; Kemerer 1987)). One possible reason why these models have not proven fruitful is that they are often unable to adequately model the complex set of relationships that are evident in many software development environments. It can be the case that a model is successful within a well-constrained environment, however, few are flexible enough to perform well outside their domain. Consequently, researchers are beginning to turn their attention to the search for alternative non-algorithmic solutions to the estimation problem, and in particular to a set of approaches that could be regarded as 'machine learning' in nature.

Machine learning techniques have been used successfully in solving many difficult problems such as speech recognition from text (Sejnowski and Rosenberg 1987) and adaptive control (Narendra and Parthasarathy 1987). It is only relatively recently that the use of machine learning techniques has been proposed as an alternative way of predicting effort.

Machine learning is a term used to loosely group together a set of techniques that can be seen to embody some of the facets of the human mind, that allow us to solve hugely complex problems (such as recognising a face in a crowd) at an incredible speed which dwarfs even the fastest computers[8] (Schank 1982; Gentner 1983). However, much of the fine detail of how the human mind works is still poorly understood and it is for this reason, that much of the work presented in this chapter is still in its infancy and remains largely untested.

---

[8] While this remains generally true, an IBM computer 'Deep Blue' did recently beat the world chess champion. Chess has long been regarded as the acid test of whether 'intelligent' machines will one day supersede human mental abilities.

This chapter will consider a variety of techniques that might be utilised in effort estimation that attempt to 'learn' the underlying relationships present in the software environment, such as neural networks and case-based reasoning.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANN's), inspired by the architecture of biological neural networks, are massively parallel systems comprising simple interconnected processors (or neurons - see fig. 3.1). The neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold. This output then becomes an excitatory or inhibitory input to other neurons in the network and the process continues until an output is generated. Artificial neurons are intended to be roughly analogous to biological neurons, for example, the weighted inputs represent biological synapses, the interconnections between neurons represent the dendrites and axons, while the threshold function represents the activity in the biological soma. However, it must be understood that much of our knowledge of how the human brain works is based upon conjecture.

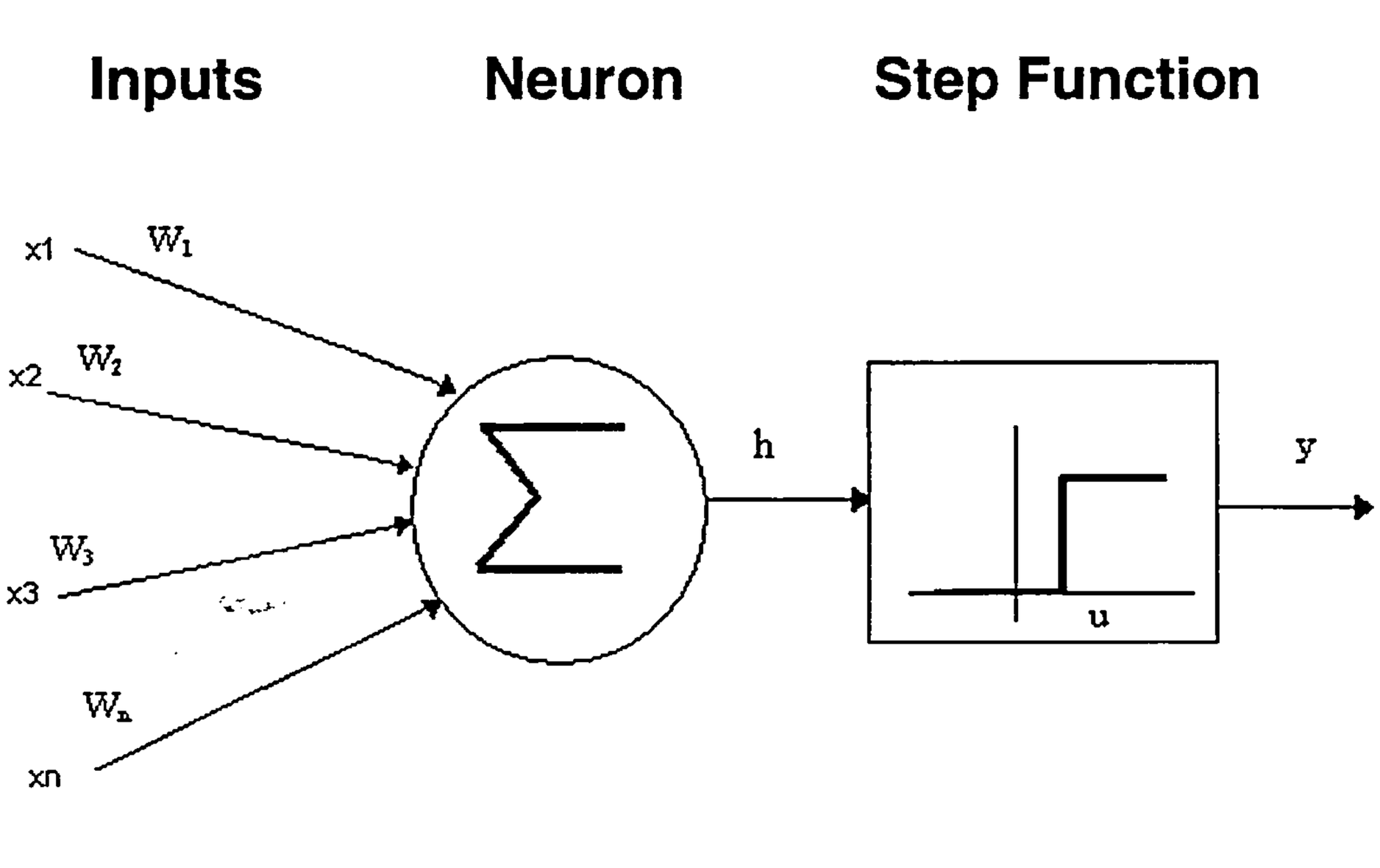


Figure 3.1 : A McCulloch and Pitts neuron

As has been stated, an artificial neuron computes the weighted sum of its n inputs, x_j, where j = 1,2,.....n, and generates an output of 1 if this sum is above a certain threshold u. Otherwise, an output of 0 results. Mathematically speaking:

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right)$$

(Eqn. 3.1)

where $\theta(.)$ is a unit step function at 0 and w_j is the synapse weight associated with the jth input. *u* is considered as another weight i.e. $w_0 = -u$ attached to the neuron with a constant input of $x_0 = 1$ (i.e. the threshold). Positive weights model excitatory synapses, while negative weights model inhibitory ones. The activation function in figure 3.1 is known as a Step function, however, there are a number of functions that can be utilised such as Gaussian, Linear, Sigmoid and Tanh. It is the Sigmoid function that is the most frequently used in ANNs.

Neural network architectures are divided into two groups:

- **feed-forward** networks where no loops in the network path occur and
- **feedback** networks that have recursive loops

Of the different architectures, the feed-forward multi-layer perceptron is the most commonly used. As the name suggests, the network has no loops between nodes and is static, which means that only one set of outputs result from a given input. As a result feed-forward networks, unlike their counterparts, have no 'memory' of any previous network state. Figure 3.2 illustrates a possible network architecture[9] configured for the estimation of software project effort. The inputs to this network are unadjusted Function Points, the number of screens, development language and the project type. At first, the system is initialised with random weights. The network then 'learns' the relationships implicit in a set of data by adjusting the weightings when presented with a combination of inputs and outputs that are known as the training set. There are a number of training algorithms that can be used to train the network, each having particular areas of speciality. The most common learning algorithm used by software metrics researchers is Back-Propagation[10], which is used in prediction and classification problems. However as Gray and MacDonell (1997) point out, the emphasis on just one learning algorithm reflects the lack of understanding of the ever-advancing state of ANN research. After training, the network is ready to make estimates for new inputs.

---

[9] In this case a multi-layer perceptron. This type of network has a number of neurons which are organised into interconnected layers including input and output layers and one or more hidden layers.

[10] The Back-Propogation algorithm has three stages: feed-forward of the input error, Back-Propogation of the output error, and adjustment of the network weights.

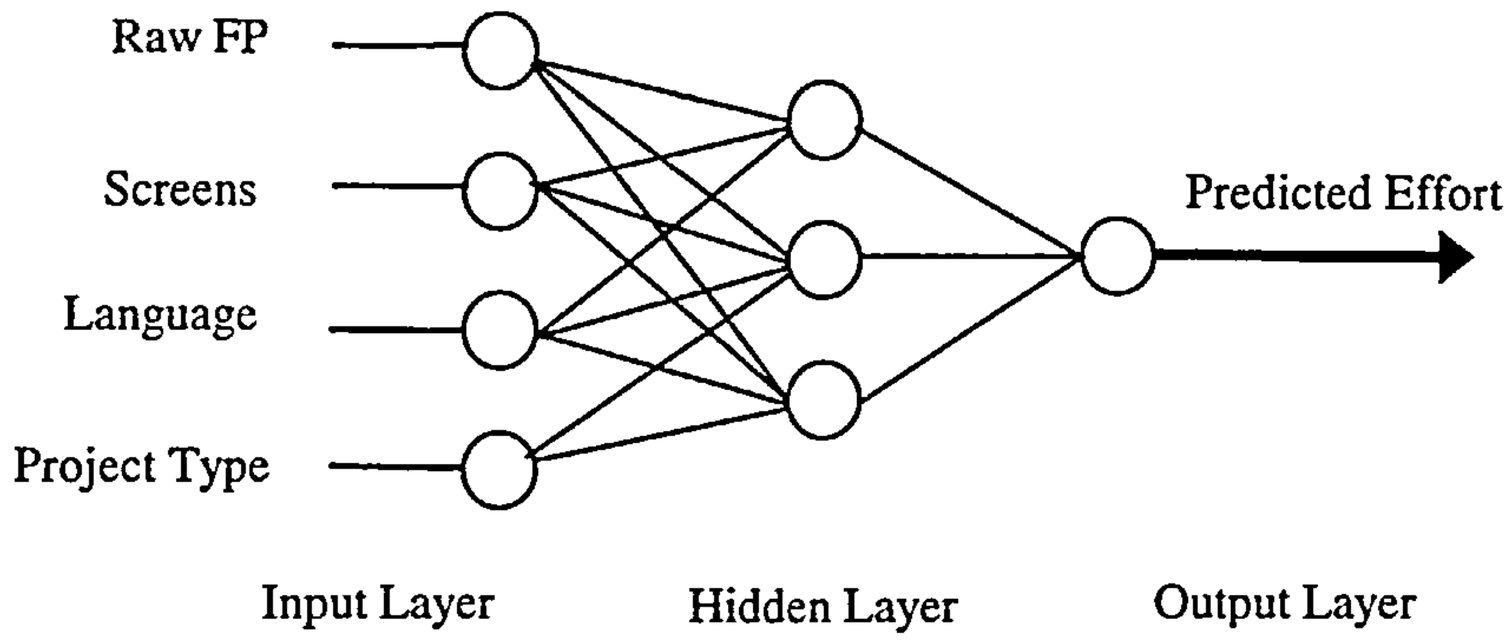


Figure 3.2 : A multi-layer perceptron

A number of studies looking at the use of neural nets to predict software development effort can be found in the literature. On the whole these studies have focused most attention on the accuracy of the approach when compared to algorithmic models and little on the suitability[11] of the approach for building effort prediction systems. In general the strategy for testing neural networks is to divide the historical data collected into sets, one used to train the data (usually the larger set) and one to test the trained network.

Venkatachalam (1993) chose a Back-Propagation learning algorithm used on a multi-layer perceptron to predict software effort and development time. The preliminary results obtained were seen as promising by Venkatachalam, when he applied the approach to data (22 inputs) taken from the COCOMO database.

Wittig and Finnie (1997) also employed a back-propagation multi-layer perceptron when they predicted development effort on the Desharnias (1988) and Australian Metrics Association data sets. Using test sets of just 10 randomly selected projects and keeping the remaining 71 and 105 projects respectively as the training sets, they produced very encouraging results. The neural network was able to predicting effort within 25% of the actual values, more than 75% of the time, which compares well with other techniques.

Jorgenson (1995), again, reports the use of a multi-layer perceptron with a back-propagation algorithm on a data set comprising 109 maintenance projects. His study compared four different approaches to estimation of maintenance effort: (i) regression models, (ii) a neural net and (iii) a form of pattern recognition against (iv) a simple baseline rule of thumb model: *effort is equal to size divided by the mean productivity*. The neural network was found to perform

---

[11] Possible reasons for deeming them unsuitable could include their interpretability, suitability to the type of data sets common to software environments or their life-cycle availability (see section 3.6).

less well than the best regression model, in terms of MMRE, but very favourably in terms of Pred(25). On the negative side, he found the neural network to be one of the least robust approaches. He measured robustness by studying the accuracy of each technique when applied to the same five test data sets. Of the techniques, the neural network demonstrated the largest differences in prediction accuracy.

Serluca (1995), reports the use of a back-propagation network on the MERMAID-2 data set. The result obtained using the full data set was far superior to regression and marginally better than an estimation by analogy method. However when the data set was separated into two more homogenous and therefore smaller clumps, the neural net performed very poorly, while the two other methods improved considerably. This led Serluca to conclude that neural nets require large training sets before they give accurate predictions.

Karunanithi et al. (1992) use neural nets for the purpose of predicting software reliability. They appear to be more imaginative in their approach to the adoption of network architecture and learning algorithm, opting to try both a feed-forward and a feed-back (Jordan) network with a learning algorithm known as cascade-correlation that combines both incremental development of the network and learning by back-propagation in one. The authors conclude that neural network models produce more accurate prediction systems then analytical models.

An Albus multi-layer perceptron is utilised by Samson et al. (1993) to predict software effort. The neural net is compared to linear regression on the COCOMO data set with results of 428% and 521% in terms of MMRE. Although out-performing regression the neural nets result is still very poor and unconvincing.

A multi-layer architecture coupled with a back-propagation learning algorithm is again used by Srinivasan and Fisher (1995) to analyse the Kemerer data set. As a consequence of the small size of the Kemerer data set (15 projects), it is used entirely as the test set and the network is trained on the COCOMO data set (63 projects). An MMRE of 70% is obtained using the network which is compared with algorithmic techniques: a Function Points based regression model (MMRE = 103%), basic COCOMO (MMRE = 610%) and SLIM (MMRE = 772%). The results of this experiment are very supportive of neural nets, however, it should be noted that the neural net is trained on data from a different data set to that which it is tested on. This is by no means a bad result in that it provides some evidence that neural network approaches may be utilised across environments. Unfortunately, they also found that the results were sensitive to the number of hidden units and layers. To help find the best network

configuration, they suggest dividing the training data further so that different configurations can be tested before learning begins.

Hughes (1996), like Jorgenson compares a range of approaches to effort estimation including, analogy, regression and a neural network. The data set comprised 33 telecoms projects and was initially divided into two homogenous groups, which led to mixed results from the neural network (MMRE = 163% and 41%). When the data set was combined, the MMRE improved to 55%, which reinforces the fact that neural networks can flourish when presented with a larger data set. At the same time, results from other techniques including analogy and regression deteriorated.

MacDonell and Gray (1996), while exploring alternatives to regression analysis made an accuracy comparison of a number of different estimation techniques on the Desharnais data set. The techniques studied included Function Point productivity, least squares regression, least median squares regression[12] and a neural network. The best statistical model found (predicting 30% of the validation set within 10% of their actual values) was a least squares regression model with all outliers removed. However, the best approach, overall, was the neural network which was superior to all other methods in terms of MMRE (44% - half of the nearest statistical approach) and Pred(25) (63%).

Study	Learning Algorithm	Data set	No. of Projects	Predicting	Results
Venkatachalam	Back-Propagation	COCOMO	63	Development Effort & Time	"Promising"
Wittig & Finnie	Back-Propagation	Desharnais/ ASMA	81 136	Development Effort	Pred(25) = 25%
Jorgenson	Back-Propagation	Jorgenson	109	Maintenance Effort	MMRE = 100%
Serluca	Back-Propagation	Mermaid-2	28	Development Effort	MMRE = 76%
Karunanithi et al.	Cascade-Correlation	N/A	N/A	Reliability	"More accurate than algorithmic models"
Samson et al.	Back-Propagation	COCOMO	63	Development Effort	MMRE = 428%
Srinivasan & Fisher	Back-Propagation	Kemerer & COCOMO	78	Development Effort	MMRE = 70%
Hughes	Back-Propagation	Hughes	33	Development Effort	MMRE = 55%

Table 3.1 Summary of neural network effort prediction studies

[12] A form of robust regression which is less sensitive to outliers than least squares regression.

The ability of neural networks to generalise and solve problems of great complexity has been proven in a number of areas. Based on the evidence so far (a summary of the studies are presented in Table 3.1), they have great potential in the area of effort prediction. However, despite the increasing popularity of the approach, a number of flaws remain that can make neural networks difficult to utilise and completely unsuitable under some conditions. First, ANN's can be considered as black boxes[13]. The knowledge stored in the architecture and the synapse weights is not easily explained. It is this ability, to explain the relationship between the inputs and output, that is considered important by Davis et al. (1997), if neural networks are to gain user acceptance. Second, guidelines for the construction of neural network topologies (units and layers), and the way they are trained (learning algorithms and number of learning epochs) are very vague. Further, even with the same architecture, results will not be repeatable due to the arbitrariness of the random weights. A third and final problem, that is particularly pertinent to the software community, is the amount of data required to usefully train a network. Many software organisations would find it difficult to collect an adequate amount of data to make the technique viable.

In spite of these problems, there remains a great deal of interest in this approach and the accuracy of the results suggest that neural network approaches can be considered at least comparable with algorithmic approaches. A great deal of further research needs to be done before neural nets are accepted as common practice. This research would preferably look at some of the large number of algorithms and networks, rather than at just back-propagation on a multi-layer perceptron.

3.2 Rule Induction Systems

Rule based systems have been implemented successfully in a number of different domains including aerospace, manufacturing, business and medicine. However, as yet, there have been few attempts to harness rule based reasoning to solve software development problems and no serious attempts to use rule based systems for predicting effort. Thus the inclusion of this approach serves only to highlight its potential as an alternative prediction technique.

In a rule-based system, known facts, stored in a knowledge base, are matched against a set of rules from a rule base. If the premise of a rule is found to be true then that rule fires which leads to the inference of new facts. This process continues until no more rules can be fired and

---

[13] This is seen as an advantage by Karunanithi et al. (Karunanithi, Whitley et al. 1992) in that the users need not concern themselves with the underlying process being modelled.

some final output can be determined. A demonstration of how a rule based system could be utilised to predict effort is given below:

IF *Function Points* > 500 OR *Function Points* > 350 AND *Military Project*
THEN *Complexity*= 4

IF *Complexity* >3 AND *Team experience* < 4
THEN Effort = 1000-1500 Man-Days

In this example, if the first rule is satisfied, a new fact is established - that *Complexity* is equal to 4 - which in turn becomes part of the premise for the second rule. In practice, rule based systems are often vastly more complex, typically containing hundreds of embedded rules.

To be classed as a machine learning technique, the system must infer the set of rules from actual development data rather than have them supplied by a human expert as with traditional rule based systems. This has the effect of removing 'human bias' and also helps to overcome another common problem, rule base maintenance over time. Luckily, even though the rules are often complex, they are usually visible and thus are easier to understand.

3.3 Fuzzy Systems

It is often the case, in poorly defined areas such as software development, that measurements and relationships are expressed in a "fuzzy" way rather than with precise values. This uncertainty is the price that is paid for attempting to solve complex problems. For example, when asked about an impending software project, a manager is more likely to express the size with terms such as "very large" or "medium-small" rather than in exact figures. This fuzzy view potentially affords a broader and richer field of data and the manipulation of that data than do more traditional methods (Brule 1985)

The idea of three (true, false and indeterminate) or more valued logic has been around for over a millennium. However, the seminal work on fuzzy set theory and fuzzy logic was proposed by Zadeh (1965) in 1965. The basic principle is that set membership values are assigned to observations in the range 0.0 to 1.0 where a value of 0.0 represents absolute falsity and 1.0 represents absolute truth. The values in-between 0.0 and 1.0 represent a degree of partial truth. As an example, take the statement:

"Module X is very small"

If the module contains say 50 lines of code, we might decide to assign a truth value of (0.7) to this statement or in other words we believe the module to be "more or less, very small". Note, this should not be confused with a 70% probability which supposes that the module is or is not very small excluding any middle ground. For a good summary see (Zadeh 1988).

There are a number of ways data fuzzification could potentially be applied to the effort estimation problem. One way would be to construct a rule induction system (as described above) replacing the crisp facts with fuzzy inputs (Fig 3.3). An inference engine would then use the rulebase to map inputs to a fuzzy output which can either be translated back to a crisp value or left as a fuzzy value (which might be seen as desirable as it highlights the speculative nature of the estimate). As MacDonell and Gray (1996) note, expressing targets with fuzzy values allows for a less harsh form of commitment.

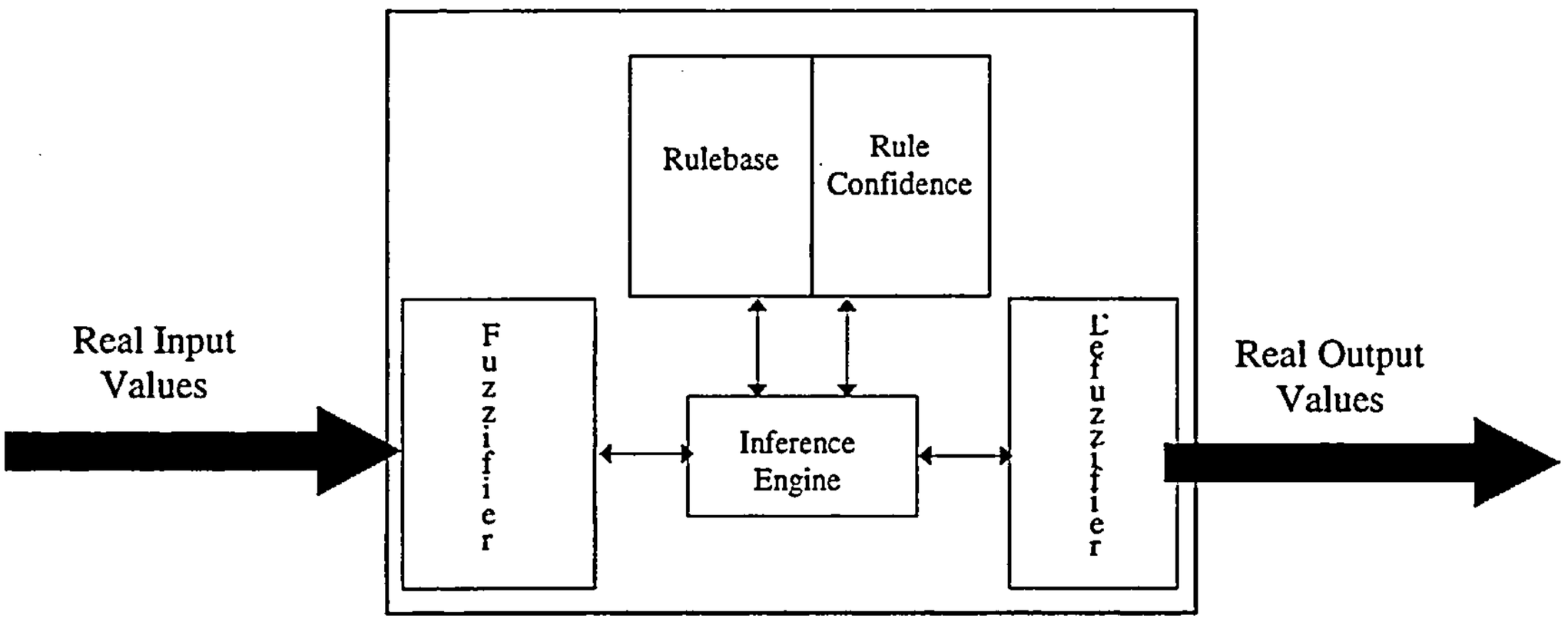


Figure 3.3 : A fuzzy rule based system

Neuro-fuzzy systems are another potential application for fuzzy concepts. A neuro-fuzzy system attempts to combine the strengths of fuzzy theory and neural networks while overcoming many of their associated weaknesses. One of the major criticisms targeted at neural network approaches is that they often over adapt to the data on which they are trained, which inhibits generalisation and is also a problem when noisy or unreliable data is presented. This is overcome in neuro-fuzzy systems by fuzzifying the input data, improving the ability to model unseen data. Neuro-fuzzy systems also have the advantage that they are "grey boxes", that is, their reasoning is not totally opaque as with standard neural networks. Neuro-fuzzy systems have been used successfully in a number of applications including diagnosing potential cases of breast cancer (Bridgett, Brandt et al. 1995).

Again, at present, the use of either technique for estimating software effort remains speculative. However, the amount of research on both subjects would seem to suggest that they are reasonably well evolved techniques and there is no reason why either may not be utilised for effort prediction purposes.

3.4 Regression Trees

Regression trees, unlike their counterpart classification tress, solve quantifiable as opposed classifiable problems. The regression tree approach requires training data from which to learn the rules that appear on each of the leaf nodes. The algorithms work by using the features of the data, that are thought to influence the outcome of the output feature, to create a tree structure that branches based upon the values of the features. The most common tree structure involves the choice of two branches, but multiple branches are possible.

There have been a couple of reports of the use of such trees to assess aspects of software development, see for example (Porter and Selby 1990). Srinivasan and Fisher (1995), describe the use of a regression tree tool, CartX, for effort estimation. CartX was used on the Kemerer data set. They concluded that, although the technique was superior to COCOMO and SLIM, the results were less impressive than the use of a Function Points driven statistical model or a back-propagation driven neural network.

The limited use of regression trees for effort estimation purposes has, so far, produced few results that appear to be sensitive to tree construction decisions such as tree depth and the algorithm that creates the tree. A further substantial weakness of the approach is that it is unable to process features that have values outside of the range of the training data. On the positive side they can provide insights into the nature of the decision process, as opposed to techniques that can be considered 'black boxes' such as neural networks.

3.5 Case-Based Reasoning (CBR)

Case-based reasoning is a problem solving approach that has received a great deal of attention recently. It has its origins in the work of people such as Gentner (1983), looking at analogical reasoning, and Schank (1982) who studied dynamic memory and the role of previous situations (or cases) in learning and problem solving. Development of the first true case-based reasoning system, CYRUS, is attributed to Kolodner (1983) who used the work of Schank in a basic question and answer system that held knowledge of the various meetings of former US

secretary of state Cyrus Vance. More sophisticated systems built more recently have been applied to problems such as dispute resolution, speech recognition, medical diagnosis and Chinese cooking!

Cases are abstractions of events that are limited in time and space. They are viewed by cognitive psychologists as episodic knowledge. In case-based reasoning terms, cases are problems that have been solved (or have failed to be solved) using a particular problem solving mechanism. Each case contains a description of the problem and the solution (assuming it was solved) to the problem that was found for that case. The actual definition of what constitutes a case-based reasoning system is open to interpretation, as the use of historical cases to reason about future cases is a key part of all machine learning approaches at some level. For the purpose of this thesis, case-based reasoning is taken to refer to systems that retain, retrieve, revise and reuse explicit cases.

The case-based reasoning process has been described by Aamodt and Plaza (1994) (Figure 3.4) as cyclic and composed of four stages:

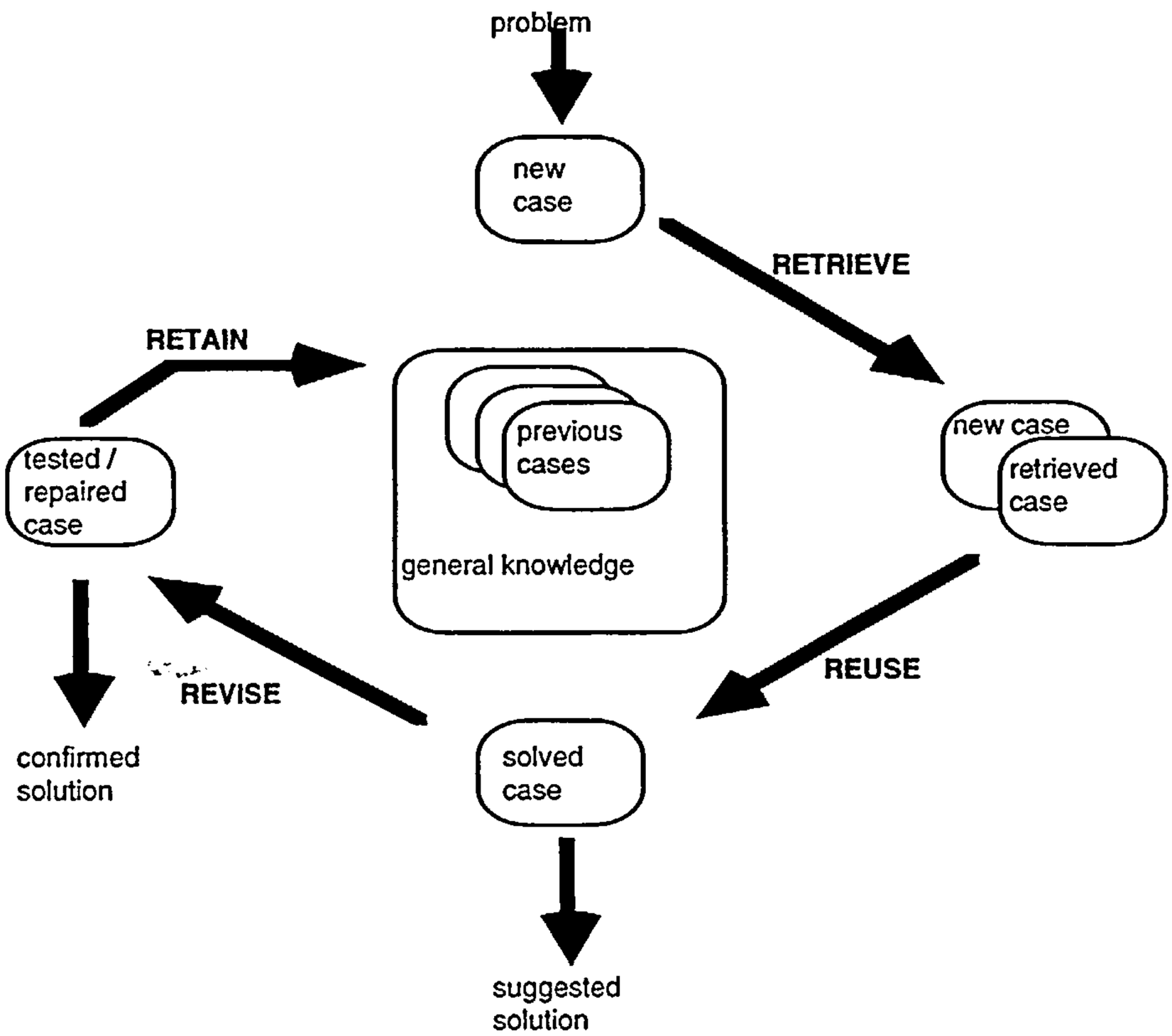


Figure 3.4 : The case-based reasoning cycle

RETRIEVAL of similar cases

REUSE of the retrieved cases to find a solution to the problem
REVISION of the proposed solution if necessary
RETENTION of the solution to form a new case

When a new problem arises, a possible solution can be found by retrieving similar cases from the case repository. The knowledge available with these similar cases can then be reused to form a solution to the new problem. The solution may be revised based upon experience of reusing previous cases and the outcome retained to supplement the case repository. This leaves a number of issues that must be dealt with before a case-based reasoning system can be effectively deployed. First, how cases are characterised, second, how similarity is discerned when retrieving cases and third how solutions can be revised.

Case characterisation poses a difficult problem when developing repositories for new case based reasoning systems. Parameters used in CBR systems can be either quantifiable (measured on the ratio or interval scale) or categorical (measured on the ordinal or nominal scale). Some expert knowledge of the problem space is required so that those features that are judged suitable for the purpose of gauging similarity are collected. Inevitably though, feature selection is a pragmatic task and must rely on the information that is available at the time that the classification or prediction problem is to be solved. Rich and Knight (1995) discuss the problem of choosing insufficiently general features. Again the solution seems to be to use an expert.

The way in which similarity is gauged between cases is one of the major topics in case-based reasoning. There are a number of approaches described in the literature (Aha 1991; Watson and Marir 1994) including a number of preference heuristics proposed by Kolodoner (1993)

Nearest neighbour algorithms - these are either based upon straightforward distance measures or the sum of squares of the differences for each variable. In either case each variable must be first standardised (so that it has an equal influence) and then weighted according to the degree of importance attached to the feature. A common algorithm is given by Aha (1991)

$$SIM(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{i \in P} Feature\_dissimilarity(C_{1i}, C_{2i})}} \qquad \textbf{(Eqn. 3.2)}$$

where P is the set of features, C_1 and C_2 are cases and

$$Feature\_dissimilarity(C_{1i}, C_{2i}) \begin{cases} (C_{1i} - C_{2i})^2 \\ 0 \\ 1 \end{cases} \qquad \textbf{(Eqn. 3.3)}$$

where i) features are numeric, ii) the features are categorical and $C_1=C_2$ or iii) the features are categorical and $C_1 \neq C_2$ respectively.

This approach is the most popular within the case-based reasoning community. In a survey of commercial CBR tools carried out by Watson an Marir (1994), all 10 were reported as using a variant of the nearest neighbour algorithm.

Static similarity measure - Althoff (1996) proposes a variant for features based on the following measure:

$$SIM(C_1, C_2) = \frac{a.\#E}{a.\#E + b.\#D + c.\#U1 + d.\#U2} \qquad \textbf{(Eqn. 3.4)}$$

where E is the set of features with the same values for cases 1 and 2, D is the set of features with different values, U1 is the set of features with known values for case 1 and not case 2 and U2 is the set of features with known values for case 2 but not case 1. In addition a, b, c and d are parameters for which Althoff suggests the values a=1, b=2 and c,d=0.5. This approach suffers from the problem of continuous attributes being almost but not quite the same. In other words, if features do not match no account is taken of how dissimilar they are.

Knowledge guided induction - here an expert manually identifies key features that are thought to affect the goal feature.

Template retrieval - similar to query by example database interfaces, i.e. the user can supply ranges, and all cases that match are retrieved. Often used as a precursor to nearest neighbour algorithms.

Goal directed preference – cases are characterised by their goals.

Specificity preference – cases with features that are an exact match are preferred over those that match in general

Frequency preference – The most recently selected cases are given preference.

Breiman et al. (1984), while examining a simple similarity algorithm, found a number of inadequacies. First, they are computationally intensive. Generally speaking though, the search time involved in retrieving cases using a nearest neighbour algorithm will only increase linearly with the number of cases. However, Aha (1991) proposes a number of algorithms that are only marginally less accurate while being more efficient. Second, the algorithms are intolerant of noise and irrelevant features. However this can be overcome by building in learning so that irrelevant features can be identified and the important ones can be given more weight in the similarity measure. And third, the use of symbolic or categorical parameters is problematic. The way most similarity algorithms approach such variables is Boolean in that they are either a match or a mismatch with no in-between.

While many researchers (Aha 1991; Veloso and Carbonell 1991) have sought more accurate and efficient ways of discerning similarity between cases, some consider revision or adaptation to be the most important challenge in CBR research. Adaptation is not an issue where problems are repetitive, however, in an environment where novel problems are continuously being experienced, adaptation is essential. The problem is that once the most similar cases have been found, how do we go about adapting them so that the best solution for a new case can be found? Leake (1996) describes two distinct approaches to adaptation. The first approach involves the use of rules to facilitate adaptation which is inevitably subject to the problem of knowledge elicitation. The second approach involves finding ways to reduce the need for adaptation such as favouring cases with features that are more likely to be adaptable.

An early attempt to develop a CBR system dedicated to the selection of similar software projects for the purpose of estimating effort is described by Vicinanza et al. (1990). They created Estor, a case-based analogical retrieval system, by studying the way in which experts approached ten separate cost estimation problems. Using domain knowledge supplied by one of the experts, Estor was able to produce its own effort estimates using an analogy searching approach and adapt those estimates using rules inferred from the estimator's own protocols. The performance of the estimates produced were comparable, in terms of R^2, to the expert's own and far superior to those obtained using the regression based techniques, Function Points and COCOMO.

A second, more recent, example of software effort estimation using a case-based reasoning system is described by Bisio and Malabocchia (1995). They also developed a CBR tool, called FACE (Finding Analogies for Cost Estimation), and assessed it using the COCOMO data set. One interesting feature of their approach is that all candidate analogies from the case

repository are given a normalised score θ between 0 and 100 (100 being a perfect match) as to their similarity to the target case. The user can indicate the threshold (typically $\theta = 70$) over and above which cases can be used to form an estimate. If no cases are found (i.e. no cases have scores above the θ threshold score), then no reliable estimation can be performed. Although difficult to compare due to the limited number of projects used in the experiments[14], FACE appears to perform very favourably against algorithmic techniques.

It appears that the case-based reasoning community is thriving and that there is evidence from early research that case-base reasoning could be adapted to the effort estimation problem. The most obvious drawback with case based reasoning systems is that they have little generalisation power when confronted with new cases that have not been previously observed although this problem may be countered by the use of adaptation. They do however have a number of advantages over other methods. First, they are able to function effectively where the number of observations is small. Second, they are able to explain the reasoning process behind the selection of the analogical cases and therefore the output. And third, they seem to thrive where the problem domain is not well understood.

3.6 Comparison of Approaches

The previous sections have discussed a number of machine learning techniques that could potentially be applied to the estimation of effort. However, it is likely that each approach will be optimal under different circumstances. This section will now discuss the different techniques in relation to a set of subjective criteria under which prediction systems are often judged:

- Interpretability

This criterion considers the degree to which the reasoning behind the output of a model is explained by its internal workings. Obviously, the ability to understand the reasoning behind an estimate has a large effect on the amount of confidence one can place on the estimate. Neural networks are an example of an approach that is at essentially totally opaque. The reasoning power of a neural network is concentrated in the synaptic weights on the inputs to each node, unfortunately, these have proven very hard to interpret[15] into any meaningful explanation of their output. Hybrid neuro-fuzzy systems are regarded as easier to interpret

---

[14] The setting of the θ often meant that for many of the projects no reliable estimates could be made.

than normal neural networks because it is possible to examine their 'internal rules'. In contrast to neural networks, the hidden layer units of a neuro-fuzzy system are more akin to rules where excitatory inputs (in the form of fuzzy membership degrees from the initial input) represent positive rules. The outputs then represents the degree to which a rule has fired which determine the relative activation of the output fuzzy membership neurons.

The other techniques described within this chapter can be considered translucent in comparison to neural networks. Rule-based and regression-tree systems often give explicit readable rules as output. The main criticism though is that they are often very complex with numerous nested IF-THEN statements in the case of rule-based systems and branches in the case of regression trees. Finally the case-based reasoning approach affords the clearest interpretation value as it is usually based directly upon the similarity of one or more explicit input cases. However, even here the reasoning can get a little muddied where adaptation is employed.

- Ability to Generalise

This criterion considers the ability of each approach to generalise i.e. its ability to estimate when presented with novel data. Very often when training a model there is a trade off between the level of accuracy achieved and its ability to generalise. By trying to achieve the most accurate model, it is possible to over train the model to the data which will often render it woefully inaccurate when presented with data from outside of the training domain.

For neural networks over fitting to the training data is a serious problem that can lead to the network placing too much emphasis on individual values. However, when trained properly[16] neural networks are exceptional at generalisation. Neuro-fuzzy systems, by contrast, automatically overcome this problem by using fuzzy inputs that remove any chance that the model will focus on individual specific values. The ability of rule-based systems and regression trees to generalise is limited by the extent of their knowledge. If a new case has values outside of the current rules then they are unable to generalise. A case-based reasoning system is also limited by the available cases it can select. However, its ability to adapt estimates based upon case differences means that it retains some power to generalise.

---

[15] There are some examples of attempts to convert the weights into rule form but the author knows of no successful attempts

[16] This often means restricting the amount of training the network undertakes so that it is not over-trained on the data.

- Suitability to Software Project Data Sets

Unfortunately, for this criterion there is very little evidence available. Most of the evidence concerns neural networks, which have been found to be more accurate in terms of accuracy as the size of the data set increases. This might be a problem for software companies who are rarely able to collect large amounts of data (less than 30 projects is common). However, when neural networks are trained properly (i.e. not over trained to the data) their ability to generalise is considerable, and therefore, it is conceivable that data from different company environments might be combined to increase the size of a data set.

- Life-cycle Availability

Life cycle availability handles the coverage the approach affords to the software life-cycle. In other words at which stages does each technique become viable. Again there is little evidence for this point. However, there is no reason to suppose that all of the techniques described in this chapter could not be used at any point along the software life-cycle. The ability of each to handle categorical data is important especially for the bidding stages of a project where there are likely to be few quantifiable inputs.

In truth, it is very difficult at present to say which approach is superior under which conditions. Neural networks have been shown to be accurate, but reservations still remain over the amount of data required to make them effective and their poor level of interpretability. The efficacy of the remaining techniques remains speculative however, a tentative comparison of all of the approaches described above can be found in (MacDonell and Gray 1996).

3.7 Summary

There has been an increasing amount of interest in alternative approaches to effort estimation in the last few years. Of these approaches some, notably neural networks and case-based reasoning, are being actively researched and the likelihood is that they will soon be recognised as viable alternatives, or better still complementary, to regression based approaches. Other approaches such as neuro-fuzzy systems, crisp/fuzzy rule-based systems and regression trees are, as yet, mainly speculative and perhaps warrant more serious attention. It is clear though, that each technique has its own advantages and disadvantages under different circumstances, and that it would be useful to identify under which

circumstances each could be most effectively used. As yet, little evidence exists about the accuracy and efficacy of any one approach and there clearly is a need for more systematic comparisons between approaches.

The following chapters will describe an application of case-based reasoning technology, known as ANGEL, which has been used for predicting software effort. The ANGEL tool was developed with reference to the case-based reasoning theory covered in this chapter.

Chapter 4

The ANGEL approach to Effort Estimation

4.0 Introduction

Analogical reasoning is a fundamental part of human cognition (Oppenheimer 1956; Vosniadou and Ortony 1989). It is necessary for recognition, classification and learning; it also extends its influence into the realms of discovery and creativity. We use analogical reasoning whenever we make new decisions by recalling related past experiences, for example, if we buy food from a shop because previous purchases have been good value for money, we are reasoning by analogy. Analogical reasoning has long been recognised as being related to intelligence. Raven (1938) defines intellectual ability as the "..ability to reason by analogy from awareness of relations between experienced characters". Thus analogy has been a major component of many tests of intellectual ability and has been employed in a number of 'intelligent systems' that solve complex problems (see for example (Evans 1968; Winston 1970; Raven, Court, *et al.* 1986.)). The theories of how humans solve problems using analogies are numerous, see for example (Johnson 1962; Evans 1968; Winston 1970). However, generally speaking, all the theories embody some or all of the following chain of processes:

i) Encode attributes of a new task into an internal representation.
ii) Infer a relationship between a previous task and the new task.
iii) Identify a mapping between the previous task and the new task.
iv) Apply the mapping to the solution for the previous task to give a candidate solution for the new task.
v) Modify the candidate solution

The use of analogy for software effort estimation has been proposed by a number of researchers. Boehm (1981), back in 1981 considered informal human analogy to be one of the seven available estimation techniques. He considered its main advantage to be the fact that estimates were based upon experiences that could be analysed to determine the specific similarities/differences and their possible impacts on the new project. Cowderoy and Jenkins (1988) suggest that analogies can be found at different levels of granularity, from phase level upwards. They adopt a 5-step approach to the recognition of useful analogies:

1. Select analogies from similar domain.
2. Assess the similarities between the current environment and the analogy, reject if differences too great.
3. Assess the quality and reliability of the analogy. Where a number of potential analogies exist, reject any with suspicious backgrounds (e.g. dubious progress reported).
4. Consider known special cases (e.g. differences in methods of working).
5. Review the list of analogies and reject any regarded as still being unsuitable.

These steps, they suggest, could be partially automated in an Estimation Decision Support System (EDSS) or in a less formal system of spreadsheets, instruction manuals and library of knowledge. Despite this early recognition of the potential of analogical reasoning, it is only recently that the approach has received anything more than lip service from software engineering researchers. This lack of attention is even stranger when framed in the context of two reports (Heemstra 1992; Lederer and Prasad 1993) which both looked at the usage of cost estimation techniques across 598 and 112 organisations respectively. Both report that informal analogy based estimation was by far the most predominant technique.

The notion of similarity is implicitly tied in with the process of analogy in that a successful analogy between two cases is dependent on there being some element of similarity between them. Software effort estimation by analogy involves systematically searching for similarities between a target project that is to be undertaken, and historical source projects, then forming estimates based upon the effort recorded for the selected source analogy. As with all estimation techniques (with the exception of expert judgement) analogy requires the collection of historical data. In common with other techniques, the more homogenous (i.e. coming from the same environment) the data the greater the confidence we can place in the estimates produced.

Although an association between human analogical reasoning and automated estimation by analogy is being made, it is clear that there is a great deal of difference in their complexity and implementation. The similarity mechanism utilised in analogy effort estimation is a relatively simple proximity measure, whereas, much of the knowledge we have about human analogical processes, although based upon conjecture, suggests that they are a great deal more intricate.

4.1 Reasoning by Analogy Vs Case-Based Reasoning

Reasoning by analogy is often used as a synonym for case-based reasoning (Section 3.5) and essentially, they are the same method. However, analogical reasoning systems are considered to be distinct in that they are able to solve cross domain problems (i.e. the problems do not have to be from the same area as their potential solutions, although there must be some cross-over of attributes[17]), whereas case-based reasoning is predominantly a single domain problem solver. A further difference between the two methods is the amount of adaptation undertaken. Typically case-based reasoning systems place a great deal of emphasis on the adaptation process, whereas, analogical reasoning systems (in-line with human analogical processes) undertake little or no adaptation (Aarmodt and Plaza 1994). In all other respects the two approaches can be considered identical. As Section 3.5 has already covered most of the fundamental aspects of all case-based reasoning systems, the rest of this chapter will be concerned with the specific features of the ANGEL estimation system.

4.2 Effort Estimation by Analogy : The ANGEL Approach

Software estimation using analogy can be seen as a specific use of analogical reasoning and as has been stated, its basis is the matching of one or more projects from a historical case base, to a new project for which an estimate is required. Projects are characterised by a number of descriptor features that are used to measure between project similarity. Once the most similar completed projects have been found, the known effort values for these projects can be used to form an estimate of the effort for the new project. As a consequence of its relationship to case-based reasoning, it also inherits many of the issues that surround case-based reasoning systems. First, we have to determine how best to describe projects. Second, once we have characterised projects, how do we then discern similarity and how much confidence can we place in the resulting analogies? Third, how do we find and deal with factors that might cause noise? And fourth, how do we use the known effort values from analogous projects?

4.2.1 Characterising Projects

Apart from the goal feature (the variable that we may wish to estimate i.e. effort) we must also characterise a project with one or more descriptor features. Descriptor features can be

---

[17] The classic example is our solar system being used as a source of analogy to help in the understanding of the structure of an atom.

either quantifiable (interval, ratio or absolute) or categorical (nominal or ordinal) variables; they must be available at the point when an estimate of the goal feature is required, so commonly collected measures such as lines of code are usually impractical as they are only available much further down the life-cycle.

There are no specific guidelines for the identification of candidate descriptor features other than to be pragmatic in choosing features that can be easily collected. However, it is recommended that features that have a direct bearing on effort are used, such as functionality or complexity measures; collecting features that have no perceived relationship to project effort is likely to result in the selection of poor analogies. This is not to say that a so-called 'expert' is required for the identification of pertinent features. As will be described later in this chapter, the ANGEL approach incorporates a mechanism for finding the best combination of features presented to it, based upon the historical data it is presented with.

4.2.2 Similarity Measures

The case-based reasoning community have identified a number of different similarity measures (Section 3.5) however, the measures that have found practical use are predominantly the nearest neighbour algorithms. The nearest neighbour algorithm used in estimation by analogy involves measuring Euclidean distance in n dimensional space (Eqn. 4.1) where n is the number of features, and x and y are two project cases.

$$ED(x, y) = \sqrt{\sum_{i=1}^{i=n} (x_i - y_i)^2}$$

(Eqn. 4.1)

Euclidean distance was chosen as the measure of similarity because the straight line distance between two points is the simplest, most commonly used, proximity measure (Suppes, Krantz et al. 1989). Before the similarity measure can be used, it is important that the feature values for the projects are standardised[18] (in this case between 0 and 1) so that all the features contribute equally to the measure of similarity. As has been stated, two different feature types can be incorporated into the similarity measure, namely quantifiable features and categorical features. The use of quantifiable data is not problematic, however, categorical features have

[18] Each feature value is divided by that features range.

no notion of interval and thus can only be described as identical or different[19] to one another. This leaves us with the problem of assigning standardised values for different and identical. Within this project we have chosen to assign a distance of 1 where the feature values are different and 0 where they are identical. This decision was made on the grounds that expressing categorical similarity on the boundaries of the standardised feature range is a common practice within the case-based reasoning community (see for example (Aha 1991)) and also that the values 0 and 1 (within the range 0 to 1) were deemed the most suitable for expressing the notions of identical and different.

Figure 4.1 demonstrates how the similarity mechanism works with 3 projects that are characterised with three descriptor features: experience, Function Points and number of subsystems. It can be seen that the Euclidean distance between the new project and project A is less than the distance between the new project and project B. Thus, in this case, project A would be selected as the closest candidate analogy and its known effort value would contribute to the estimate for the new project.

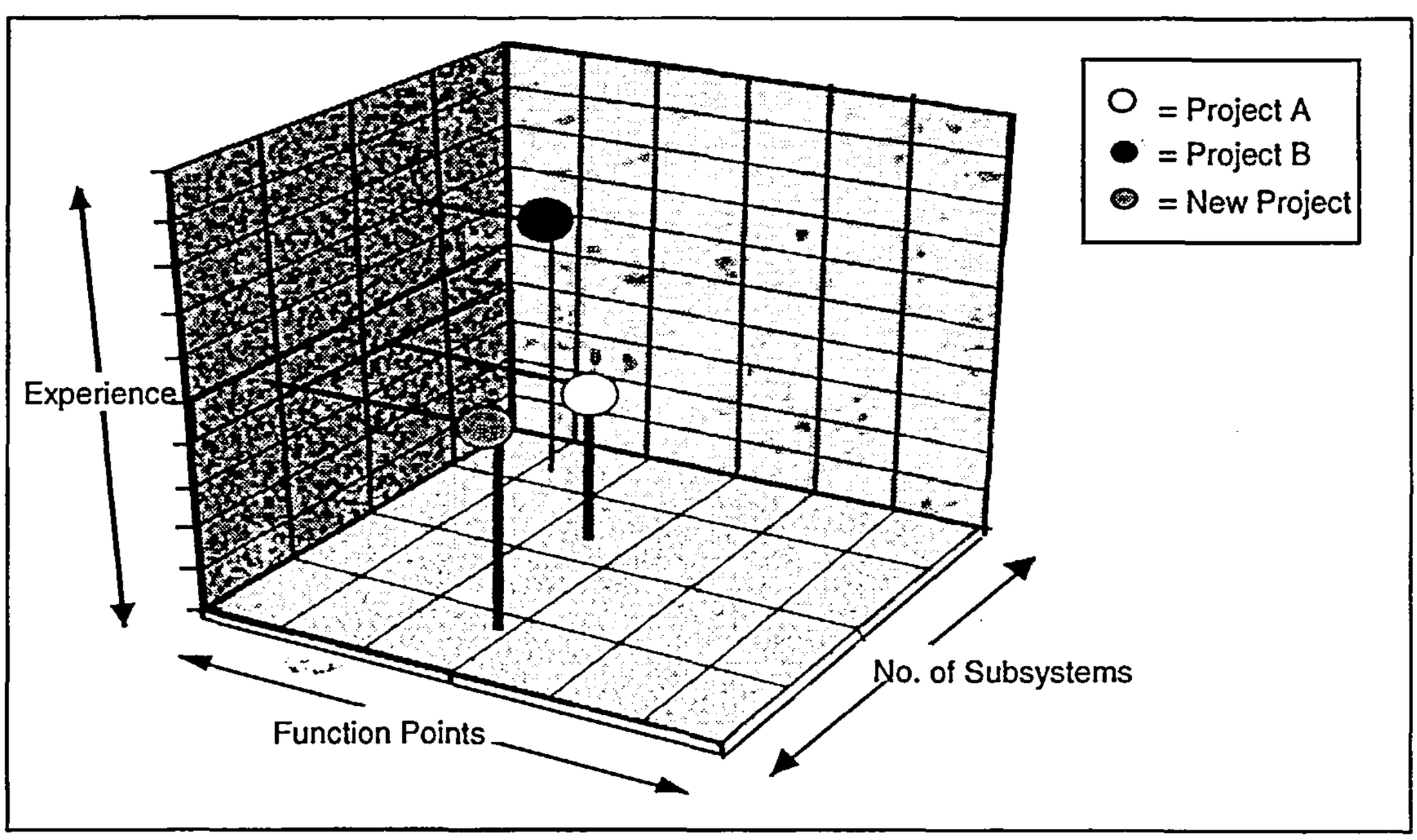


Figure 4.1 : Measuring similarity in three dimensional space

The confidence that can be placed in an estimate can be determined in a number of ways. One possible way is to use the similarity measure between the target and source project. However,

---

[19] Note, ordinal categorical feature values can also be manipulated as being greater or less than each other. However, this information is of no value in the Euclidean distance measure unless a psuedo interval is imposed on the feature values.

this strategy is problematic, in that two projects are similar in terms of the range of all the projects feature values, so that a similarity measure between two projects will change if a new project is added with feature value outside of the present range. Figure 4.2 demonstrates this using projects with, for simplicity, just one descriptor feature (i.e. in one dimensional space). Because all feature values are standardised between 0 and 1, the addition of a new project C has the effect of increasing the relative similarity between projects A and B.

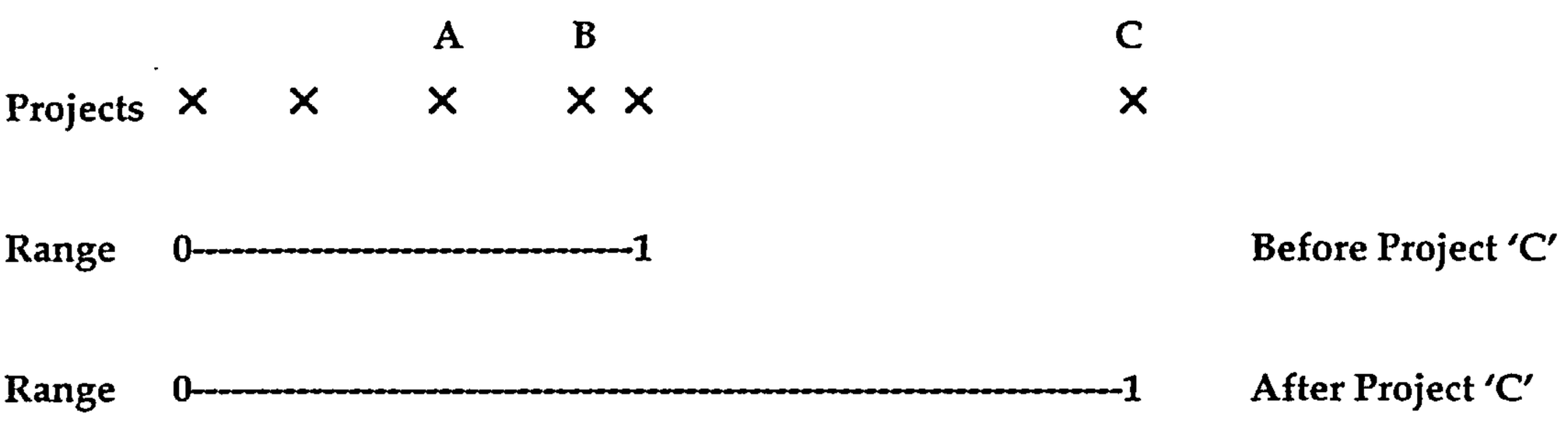
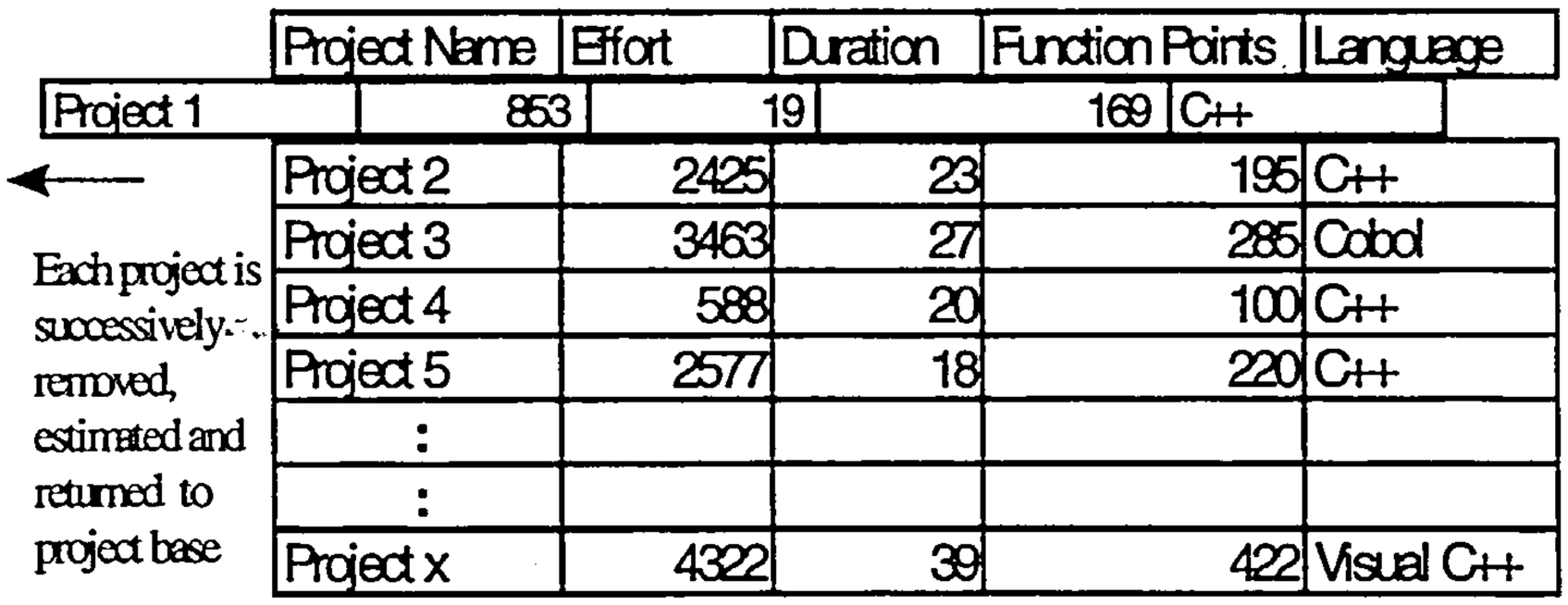


Figure 4.2 : Adding a new project in one dimensional space.

Another possible way of determining confidence is to adopt a technique similar to jack-knifing, that makes individual predictions for all of the historical projects in the Project case base. Basically, each project is successively removed and its effort estimated using the remaining projects as the analogy source (Fig 4.3). The estimated effort for each of the projects is then compared to their actual effort, which yields an indication of how much reliance can be placed upon new estimates from that Project case base.



	Project Name	Effort	Duration	Function Points	Language
Project 1	853	19	169	C++	
Project 2	2425	23	195	C++	
Project 3	3463	27	285	Cobol	
Project 4	588	20	100	C++	
Project 5	2577	18	220	C++	
:					
:					
Project x	4322	39	422	Visual C++	

Figure 4.3 : Jack-knifing a project case base

This is the technique that has been adopted in the ANGEL approach for assessing estimate confidence (expressed in MMRE - see section 2.6). It not only allows the user to assess how accurate an individual estimate may be, but also compare the ANGEL approach with other estimation methods such as algorithmic models.

4.2.3 Dealing with Noisy Features

It is difficult to know in advance which features will be helpful for finding useful analogies and it is likely that where a number of features are being used, some will be adding noise. Obviously some strategy is required to weed out these noisy features. The most comprehensive way this can be accomplished, and the way adopted in the ANGEL approach, is to perform an exhaustive comparison of every possible combination of features, jack-knifing the Project case base each time until the subset of features that return the best confidence figure is found. The major problem here is that an exhaustive search can be computationally expensive. To be more precise, the complexity of an exhaustive jack-knife can be expressed as $m*2^n - 1$ where m is the number of project cases and n is the number of features. Given a constant number of project cases, the time taken to perform such a search will increase exponentially with the number of features. For example, given a case-base of twenty projects on a Pentium 200, ANGEL will take 5 seconds to process 5 features, 2 minutes 40 seconds to process 10 and 45 hours to process 20 features!

4.2.4 Forming a New Estimate

The simplest way to form a new estimate is to copy the effort value from the closest source analogy to the target. However, other strategies might be considered, for example, finding the n closest projects and taking an average of their total effort or alternatively applying a weighting so that the closest projects contribute more to the eventual estimate. It is likely that different strategies will be optimal under different circumstances. For example, taking just the closest analogy leaves the estimate vulnerable to it being a poor or outlying analogy. On the other hand, where numerous analogies are selected, the effect of the closest analogy might be weakened by less important analogies.

The strategy adopted for the ANGEL approach was, first and foremost, pragmatic. It was not within the scope or time-scale of this project to test all the possible ways analogies could be used so four different strategies were chosen as outlined below:

One analogy

Estimate = effort from the closest analogy.

Two analogies

Estimate = average effort from the closest two analogies.

Three analogies

Estimate = average effort from the closest three analogies.

Two analogies (weighted)

Estimate = average effort from the two closest analogies, the first weighted double.

This is as close as the ANGEL approach comes to offering an adaptation mechanism, a feature that is so highly desired in case-based reasoning systems. Other attempts at creating analogical based reasoning systems (Vicinanza and Prietolla 1990) have tried to incorporate such mechanisms, but ultimately expert intervention is required (usually in the form of rules) which makes the system difficult to move between development environments and will make expert intervention (to maintain the rulebase) necessary.

Early experiments using automated estimation by analogy (Atkinson and Shepperd 1994) found that one disadvantage is that it requires a great deal of computation. They overcame this problem by automating the process using SPSS[20] to compute the statistical similarities between projects and then EXCEL to identify the closest analogies, develop an estimate and compute the method performance. Automating the process allowed them to create estimates relatively quickly and accurately. Using this automated technique they found[21] that certain analogy selection techniques out performed regression based techniques (in terms of MMRE) while performing slightly less well than expert judgement. From this work it was evident that to make estimation by analogy viable it would have to be fully automated. This has been done, in the form of a software tool known as ANGEL, and is described below.

4.3 Effort Estimation by Analogy: The ANGEL Tool[22]

From experience, searching for analogies using the approach described in section 4.2 can be both time consuming and error prone, particularly if there are many projects or many variables. For this reason it was decided to automate the process and provide an environment where data can be stored, analogies found and estimates generated. A prototype was developed using Visual Basic 3.0 to run under Windows (3.1 or above) on a PC, and was christened ANaloGy Estimation tooL (ANGEL). Visual Basic was chosen because it allowed for rapid and incremental prototyping of the embryonic ideas while providing a user-friendly interface. Although essentially a prototype system, built with the intention of supporting research into the efficacy of estimation by analogy, a number of commercial companies have

---

[20] A social science statistical package

[21] Using data from 21 real-time projects.

[22] A version of this tool is available at http://dec.bournemouth.ac.uk/dec_ind/decind22/web/Angel.html

shown interest in using the tool in a variety of ways from the simple identification of similar projects as an aid to expert estimators, to full use of the estimate generation facilities.

It was decided early on that ANGEL must not constrain the user by prescribing the collection of any particular features. Thus a shell architecture was developed, that enabled the user to define the features that best characterise their environment. As a result of adopting a shell architecture, ANGEL can conceivably estimate any goal feature (and is not constrained to software development problems) such as development duration, lines of code or testing effort, and is not constrained over which features must be collected. From the start, design decisions have been made with three major guiding influences: expediency, simplicity and openness.

The ANGEL tool separates the process of estimation by analogy into three key areas:

i) Data Templates

Template are simply forms used to describe the environment in which projects are to be undertaken (i.e. meta data).

ii) Project Case Base

A project case base, built from information captured in a template, is the repository for project data.

iii) Estimate Generation

Estimate generation allows the user to generate estimates for target projects based upon source projects in a project case base.

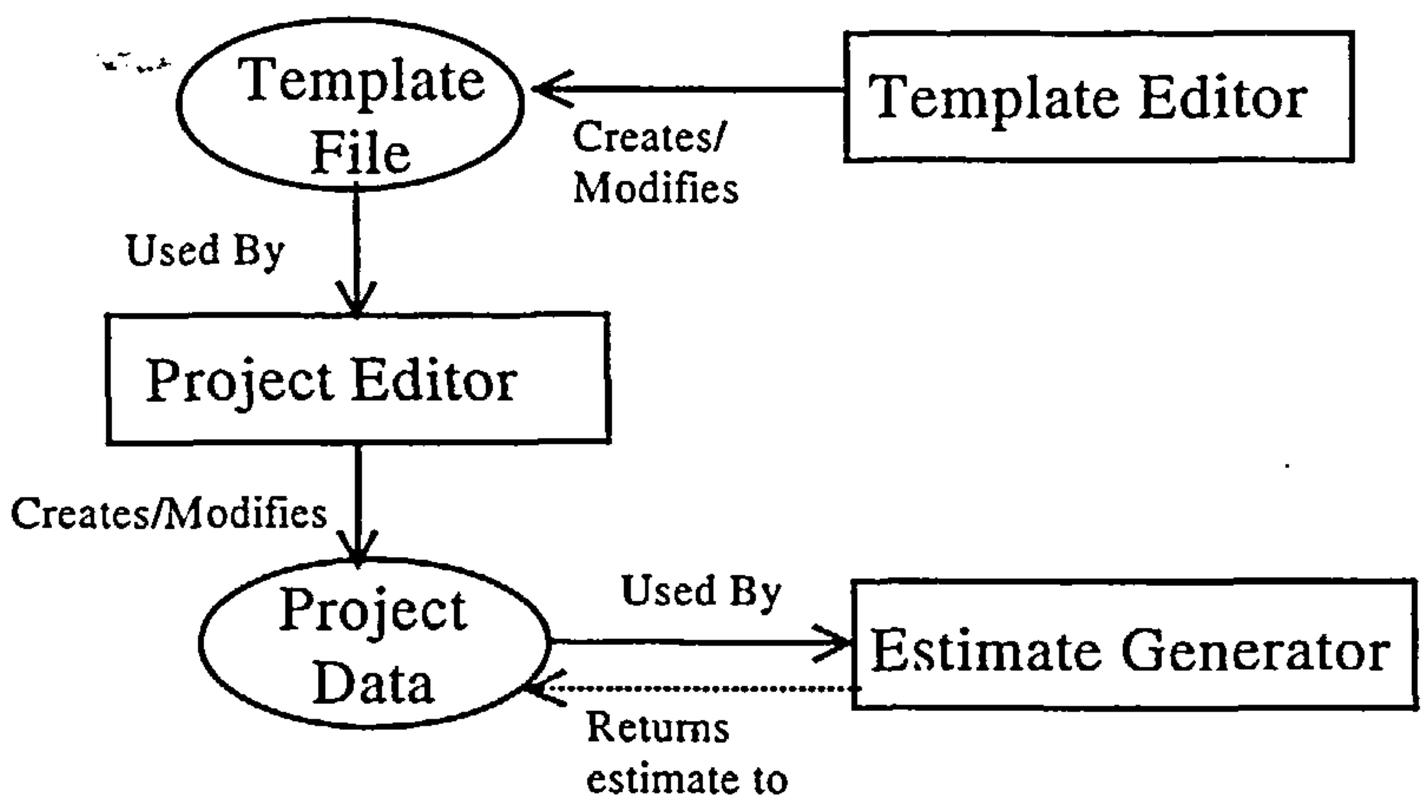


Figure 4.4 : ANGEL schematic

The interaction between the main components of the ANGEL tool is demonstrated in figure 4.4 Before project case data can be stored, the template editor is used to profile the environment from which the data will be collected. The resultant template is stored as a text file. This template file is then used by the project case editor to create a project case base. Once enough projects have been collected in the case base, the estimation generator can be used to generate estimates for any new projects. Ideally, an estimate should then be recorded in the project case base so that it can be compared to the final outcome, although this is not mandatory in ANGEL. Figures 4.5 to 4.9 illustrate ANGEL in operation.

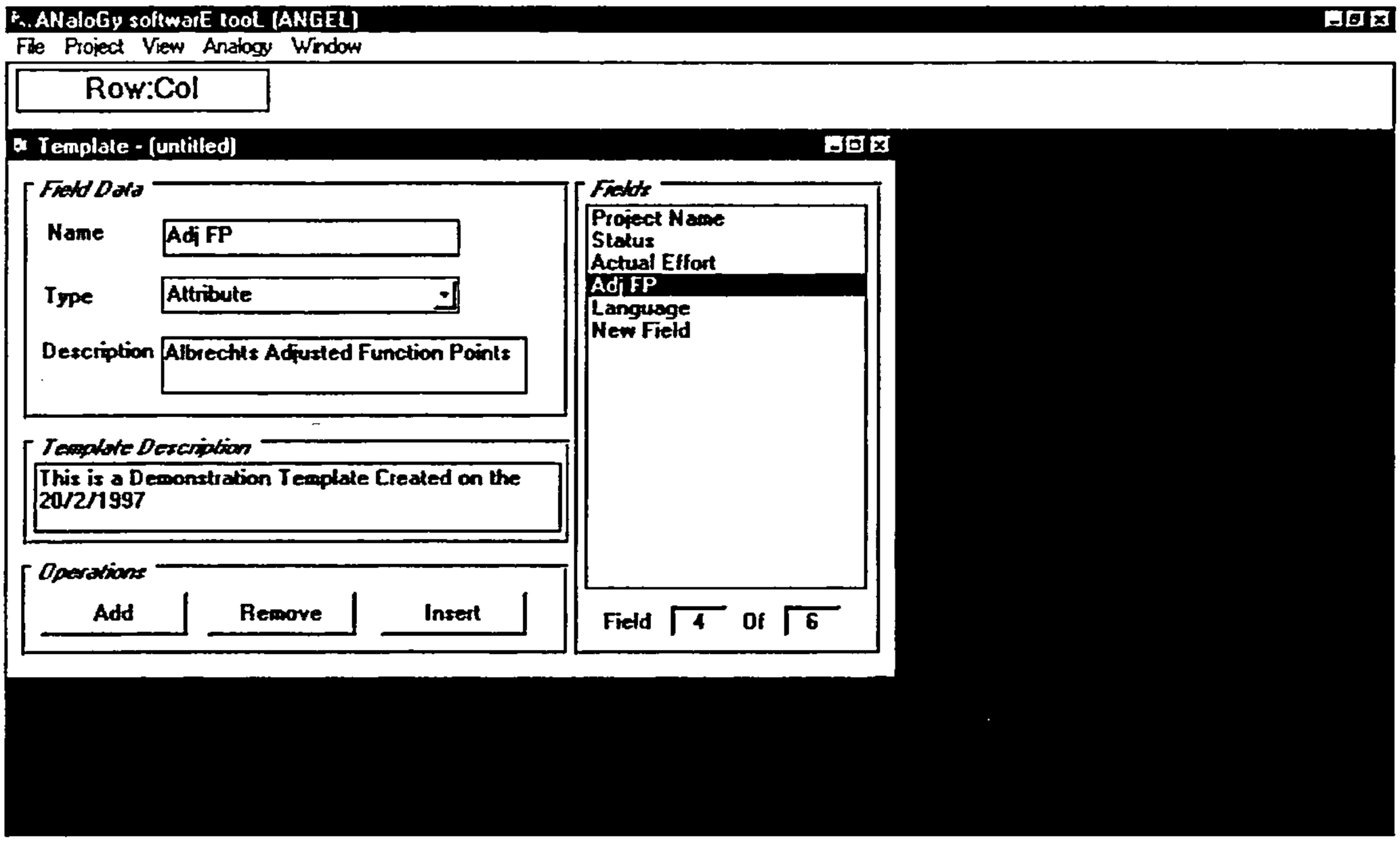


Figure 4.5 : A data templates in ANGEL

Figure 4.5 shows a template for recording environment data. Templates are an important part of this approach because they can be configured to suit the individual data collection environment of an organisation. There is no particular set of features prescribed by the template, in order that the approach can take optimum advantage of the data available at each data collection site. All feature types and names are user determined, except for *Project Name* and *Status* which are mandatory. *Project Name* merely provides a mechanism for uniquely identifying each project or case. *Status* indicates whether a project has been completed, or not and therefore whether it can be used as a source of analogy. Selection of feature type is important at this stage as it determines how ANGEL will treat data when performing estimates. The data types available are *Estimate* (a goal feature type), *Attribute* (a quantifiable

descriptor feature type), *Categorical* (a categorical descriptor feature type) and *Text* (a memo field). Before a template can be transformed into a project data store, it must have at least one *Estimate* feature and one *Attribute* or *Categorical* feature (in addition to the mandatory features).

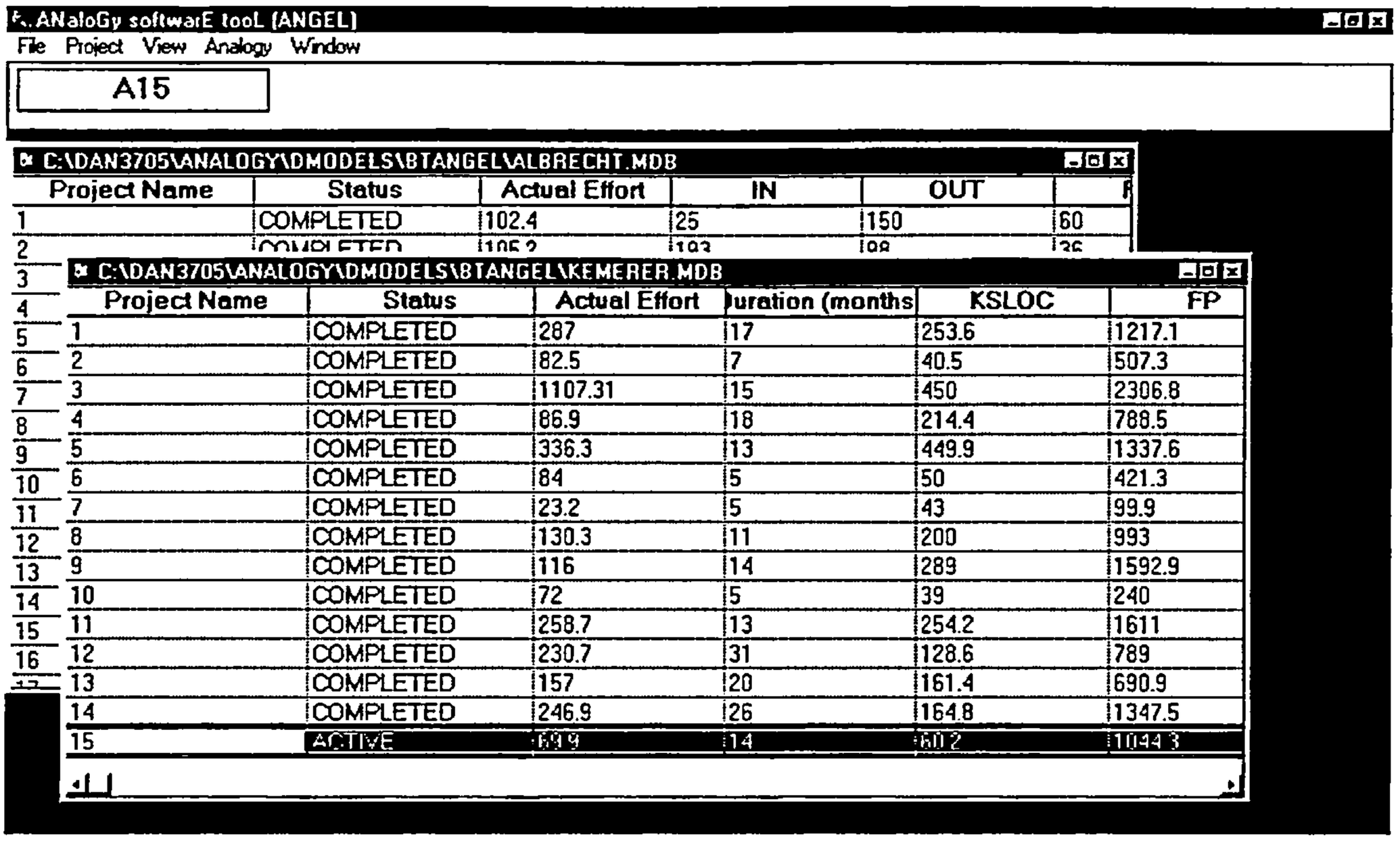


Figure 4.6 : A project database in ANGEL

Project case bases are formed from template files and stored as Microsoft Access readable database files. Figure 4.6 shows two example project case bases holding projects taken from the Albrecht (1979) and Kemerer (1987) data sets respectively. It was decided to display the project case bases as spreadsheets because of the visibility advantages this format affords. Each column in the spreadsheet represents a feature, while each row holds a single project. Figure 4.6 shows all but one project as having a *Status* of 'COMPLETED', a label that marks that project as belonging to the pool of potential source analogies, while one project has an 'ACTIVE' *Status* marking it as a project for which we might want produce an estimate.

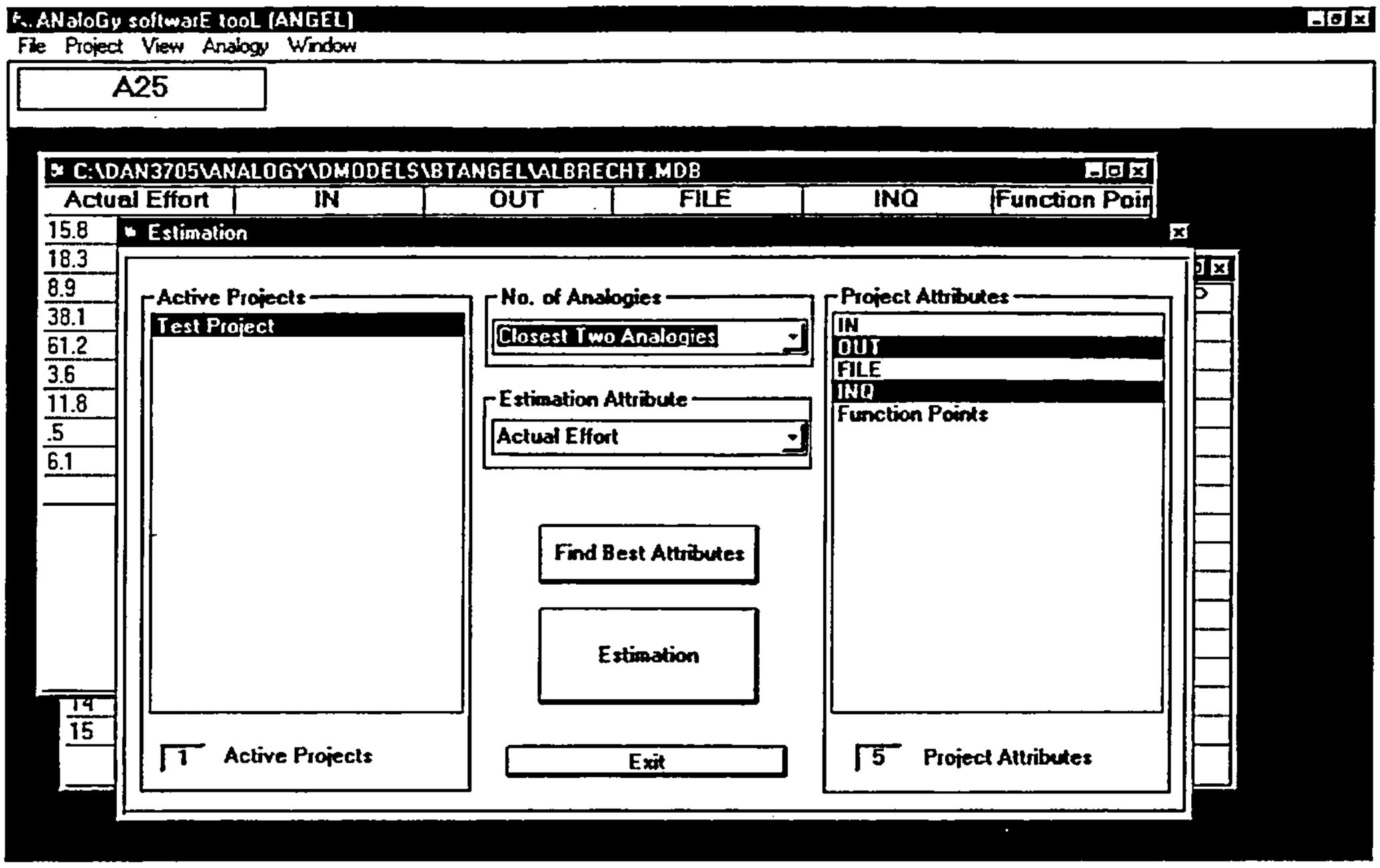


Figure 4.7 : Configuring an estimate in ANGEL

Figure 4.7 shows the estimation generator screen. Before ANGEL can generate an estimate, the user must select an active project and the number of analogies ANGEL will search for (recall that ANGEL currently allows up to three analogies to be used). Having selected the number of analogies, the features that will be used in measuring the Euclidean distance between projects must be selected. As mentioned earlier, the reason for this is that not all collected features will be helpful in finding good analogies; some features may create noise. The chosen features can be all, or just a subset, of the features stored in the project case base. Because the problem of determining these features by hand is very hard[23], ANGEL can also automatically determine the best combination of features to be used for finding analogies for a particular case base. This relies upon a brute force, exhaustive search of all possible feature combinations.

[23] The most obvious features that generally have a strong statistical influence on effort will not necessarily be the most influential features in ANGEL.

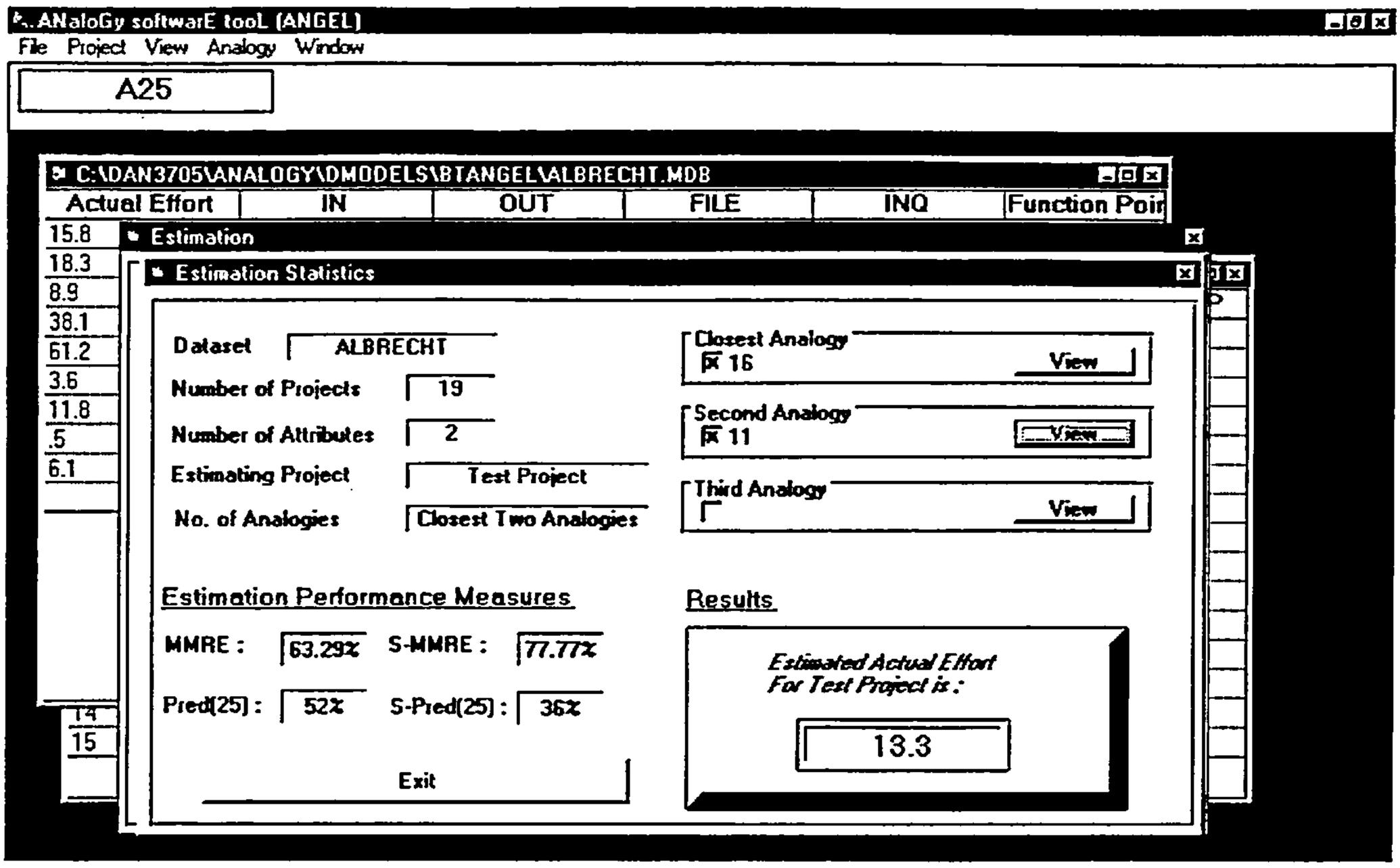


Figure 4.8 : Estimation results using ANGEL

The final step (Figure 4.8) involves predicting effort for a selected project, in this case a test project, using the completed projects from the case base. Here we see a predicted value of 13300 work hours (shown in Figure 4.8 as thousands of work hours). The confidence that we can have in the estimate is automatically provided in the form of the MMRE, Pred(25), SMMRE[24] and S-Pred(25) values[25] as described in section 2.6. A further facility is the ability to examine the source analogies used to create the prediction, in this case projects 16 and 11. Note that the estimate figure represents the average of the effort for the closest two analogous projects. Obviously, on small projects this level of precision is not necessary and perhaps falsely gives an impression that the technique can be so accurate. However, where projects are measured in larger units such as thousand work hours or person years the values after the decimal point obviously become important. At present ANGEL has no provision for recognising the two different situations.

---

[24] Also known as BMMRE - Balanced MMRE [13]

[25] Each performance indicator has its own merits and the ANGEL approach does not advocate the use of any one. However, the user should be aware that ANGEL optimises on MMRE and that the use of other indicators helps to provide a more balanced view.

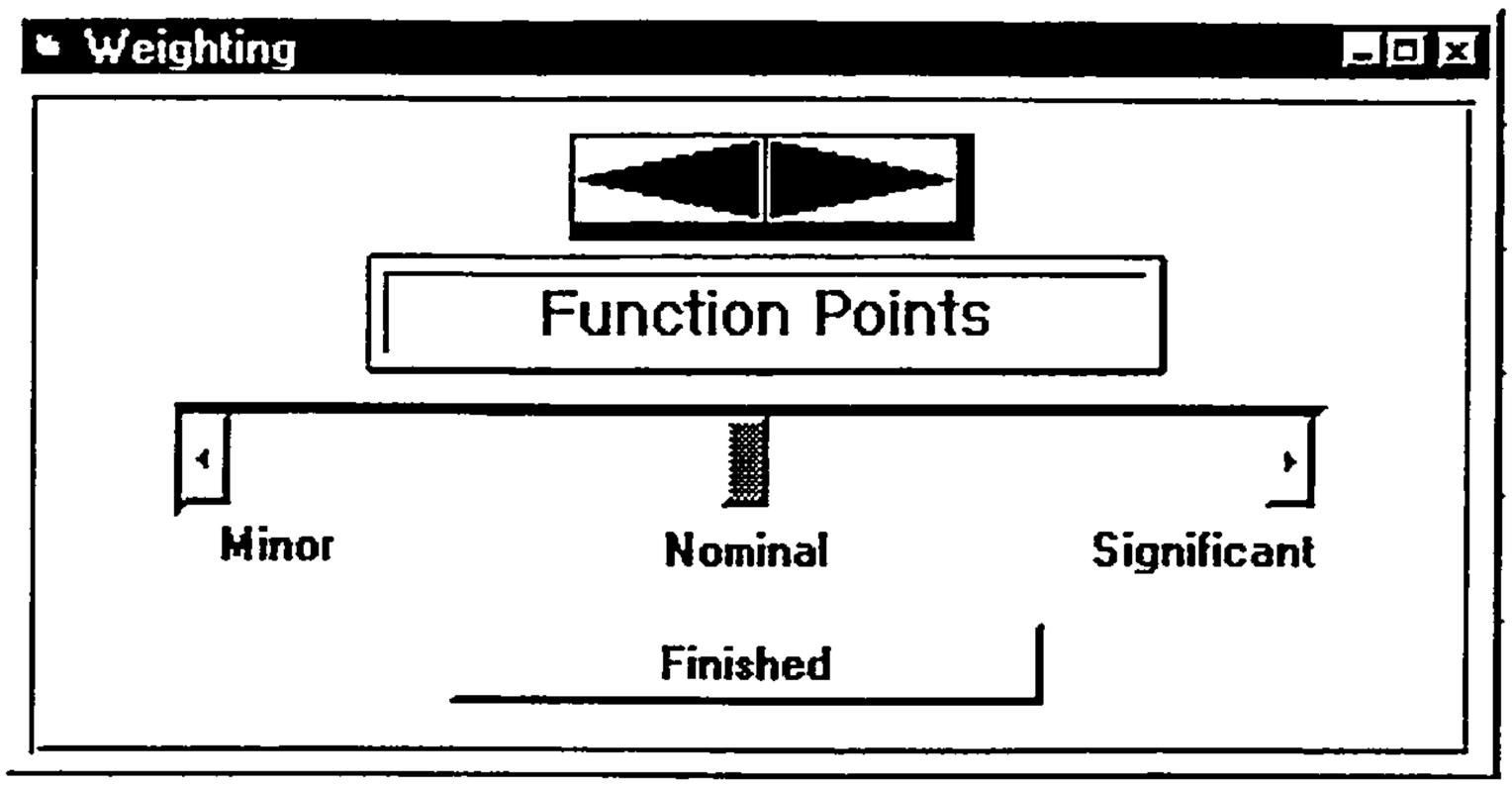


Figure 4.9 : Applying weightings to features

An auxiliary facility available in ANGEL provides the means of applying weightings to project features. Features can be classified as minor, nominal or significant, altering the amount of influence that is exerted by that feature on the similarity measure by *0.5, *1.0 and *1.5 respectively. Initially, all features are set to nominal so that all features contribute equally. The decision to adopt a three point scale for the weighting facility was made on the grounds that to have more points on what is essentially an arbitrary scale goes against the major strengths of the ANGEL tool (i.e. its simplicity and transparency). It was also considered important that whatever the values chosen, the minor and significant values should exert the same effect in both directions from the nominal value. Another function that was considered and later abandoned was a facility that would allow the user to find the best weighting for each feature by carrying out an exhaustive search similar to that used to determine the best set of features. The idea was discarded, however, because it was deemed too computationally expensive in a tool that was already sensitive to time constraints, particularly when trying to determine the best subset of features. It was also considered to go against the simplicity of the analogy technique and blur ANGEL's ability to explain its output.

4.4 Summary

There has been a great deal of anecdotal evidence in the literature indicating that humans solve complex problems by reasoning using analogies. This has led a number of software engineering researchers to consider the use of analogies in order to mimic the reasoning processes of expert human estimators.

This chapter has described a novel approach to the estimation of software development effort by searching for similar or analogous examples from sets of historical software projects. This

approach is very similar to case-based reasoning and has learnt a great deal from the experiences reported in that community. Unfortunately, the approach is computationally expensive and thus requires automated support. This support is provided in a tool, dubbed ANGEL, which allows the collection of project data and the identification of similar projects in order that the effort for new projects might be estimated.

The next chapter will describe how the ANGEL tool was used as part of an empirical analysis comparing the accuracy of the estimation by analogy and a more traditional algorithmic approach.

Chapter 5

An Empirical Investigation of the Accuracy of Estimation by Analogy

5.0 Introduction

Although a variety of different factors must be considered when assessing prediction systems, such as robustness and ease of use (see for example (Boehm 1981; Conte, Dunsmore et al. 1986; Kitchenham 1990)), arguably the most important and certainly the most visible feature, is the relative accuracy of predictions made. To be taken seriously by researchers and more importantly practitioners, any new estimation technique must justify itself first and foremost by its results. This chapter will present the results obtained when using the ANGEL tool to predict project effort for approximately 250 real software development projects across 8 different industrial data sets, summarised in table 5.1. The 8 data sets represent a wide range of development environments including a defence contractor, 2 telecoms companies and a DP services organisation, and were collected between the late 1970's and early 1990's.

Name	n	Description	Source
Albrecht	24	IBM DP Services projects	(Albrecht and Gaffney 1983)
Desharnais	77	Canadian software house - commercial projects	(Desharnais 1988)
Finnish	38	Data collected by the TIEKE organisation from IS projects from 9 different Finnish companies.	Finnish Data set: data set made available to the ESPRIT Mermaid Project by the TIEKE organisation
Hughes	33	Telecoms project builds	See Appendix B
Kemerer	15	Large business applications	(Kemerer 1987)
Mermaid	28	New and enhancements projects	MM2 Data set: Data set made available to the ESPRIT Mermaid Project anonymously
Real-Time1	21	Real-time defence projects	See Appendix B
Telecoms1	18	Enhancement projects from a large telecoms company	See Appendix B

Table 5.1 Data sets used to compare estimation by analogy and regression

Effort is also predicted using a regression based technique to allow the results from ANGEL to be put into perspective, and to provide a comparison between estimation by analogy and a technique regarded by some (Kok, Kitchenham et al. 1990) as 'state of the art' in effort estimation. The results from both techniques are then analysed together using statistical significance testing and conclusions about the relative performance of estimation by analogy are drawn from the results.

5.1 Experimental Procedure

Throughout this investigation a Pentium 200 PC was used to run SPSS and the ANGEL tool. For each data set, the four different analogy selection approaches[26] in ANGEL were compared to stepwise regression run in SPSS. Stepwise regression builds prediction models based upon one or more independent variables where variables are successively entered into the model until no further significant contribution can be made. All the available features (with the exception of nominal and ordinal scale features[27]) are entered into the stepwise regression, except where there are *a priori* reasons for leaving a feature out, for example, where there are missing values. Appendix A contains a full list of the features used for each technique. The four analogy selection approaches (as described in section 4.3) were:

- **One analogy** - the effort from the closest analogy
- **Two analogies** - the mean effort from the two closest analogies
- **Two analogies (weighted)** - the mean effort from the two closest analogies with the first weighted double
- **Three analogies** - the mean effort from the three closest analogies

The prediction accuracy of the different approaches is assessed using MMRE and Pred(25). As has been stated in section 2.6, there are a variety of other performance indicators, however, these two were chosen because they give a balanced view (MMRE tends to be conservative while Pred(25) tends to be more optimistic focusing attention only on the best cases), because they can be appropriately applied to both regression and analogy (unlike, for example, R-

---

[26] At this stage it was difficult to predict which analogy selection approach would work best under different circumstances so the results from all four are presented.

[27] Although it is possible to incorporate such features into a regression model (creating dummy variables for each feature value), their worth in anything other than a exploratory analysis is questionable.

Squared[28]) and because they are the most widely used performance indicators in the literature.

Standard significance testing was also used to test the validity of claims made about the performance of estimation by analogy when compared to stepwise regression in terms of (a) MMRE and (b) Pred(25).

5.2 Notes on the Investigation

It is important to note at this stage the two minor differences between the estimation by analogy and stepwise regression approaches and a third difference in the way they will use the data presented to them.

1) *The performance indicators for estimation by analogy and regression are generated slightly differently*

Estimation by analogy adopts a strategy very similar to jack-knifing where individual projects are removed for estimation and are then returned to the data set. The regression modelling is different in that a regression model is generated using all of the data (hereafter this process is known as 'goodness of fit'). This will, in practice, give an advantage to the regression-based technique because each project will contribute to the generation of the model from which its effort will be estimated. A possible solution to this problem would be to jack-knife the regression technique so that a new model would be built as each project is removed in turn. However, this represents a great deal of manual effort, more than was possible within the time-scales of this project.

2) *Estimation by analogy optimises on MMRE.*

Estimation by analogy decides upon the best set of features by minimising the mean absolute percentage error (MMRE). Regression, on the other hand, creates a line of best fit by minimising the sum of the squared errors. As a result it is likely that comparisons of the two techniques using the MMRE performance indicator will give an advantage to estimation by analogy.

---

[28] After initial exploratory analysis, R^2 appears to be inappropriate for analogy and with certain calculations it is possible to obtain a negative value for R^2.

For the remainder of this thesis, the assumption is made that neither of these have a serious impact on the results, and that the advantage to estimation by analogy from optimising on MMRE is no greater than the advantage to stepwise regression by not jack-knifing.

3) *The handling of ordinal scale data*

The features collected across the eight data sets in Table 5.1 range from ratio scale measures, such as *Years of experience*, to nominal scale measures, such as *Development environment*. When one is considering the use of data for estimation, it is important to take note of the operations that are permissible on each scale type. Measurement theory states (Finkelstein and Leaning 1984) that the only operations applicable on ordinal data are equality and greater/less than comparisons. However, as a number of the data sets are richly characterised with ordinal data, strategies were sought to enable ANGEL to use such data. A decision was made to treat all ordinal variables as interval on the grounds that doing so improves accuracy. This essentially means that ANGEL assumes a linear interval between all ordinal values. This approach is defended by Briand et al. (1996), amongst others, who consider that the rigid application of measurement theory can be "rather sterile in terms of results"; and also by Stevens (1946), the 'father' of measurement theory, who provides a pragmatic sanction for the use of "illegal statistics" on ordinal measures: "In numerous instances it leads to fruitful results. While outlawing of this procedure would probably serve no good purpose,...". However, no researcher would disagree that this approach must be treated cautiously and should not be used if not yielding useful results.

The next section will go on to compare estimation by analogy with regression for the eight data sets in table 5.1. The first data set, the Albrecht data set, will be examined in more detail to demonstrate the process by which all data sets were analysed.

5.3 Data Analysis

5.3.1 The Albrecht data set

A statistical summary of the features used in the analysis of the Albrecht data set is displayed in table 5.2. Note that *KSLOC* has been left out, because it is not generally available before estimation is required. The feature to be predicted (goal feature or dependent variable) is *effort*, measured in work-hours, while the potential independent variables (descriptor features) are *adjusted function points*, the *number of master files*, the *number of inputs*, the *number of inquiries* and the *number of outputs*.

Feature	Count	Min	Max	Mean	Median
Effort	24	.50	105.20	21.88	11.45
FP	24	199.00	1902.00	647.62	506.00
Files	24	3.00	60.00	17.38	11.50
Inputs	24	7.00	193.00	40.25	33.50
Inquiries	24	0.00	75.00	16.88	19.3
Outputs	24	12.00	150.00	47.25	39.00

Table 5.2 Summary statistics for Albrecht data set.

As an example of regression based model building, figure 5.1 shows a linear regression model using *function points* as the independent variable to predict *effort*. Note the negative intercept on the y-intercept, suggesting that projects from this sample have negative fixed costs. For the remainder of this analysis stepwise linear regression will be used to generate models.

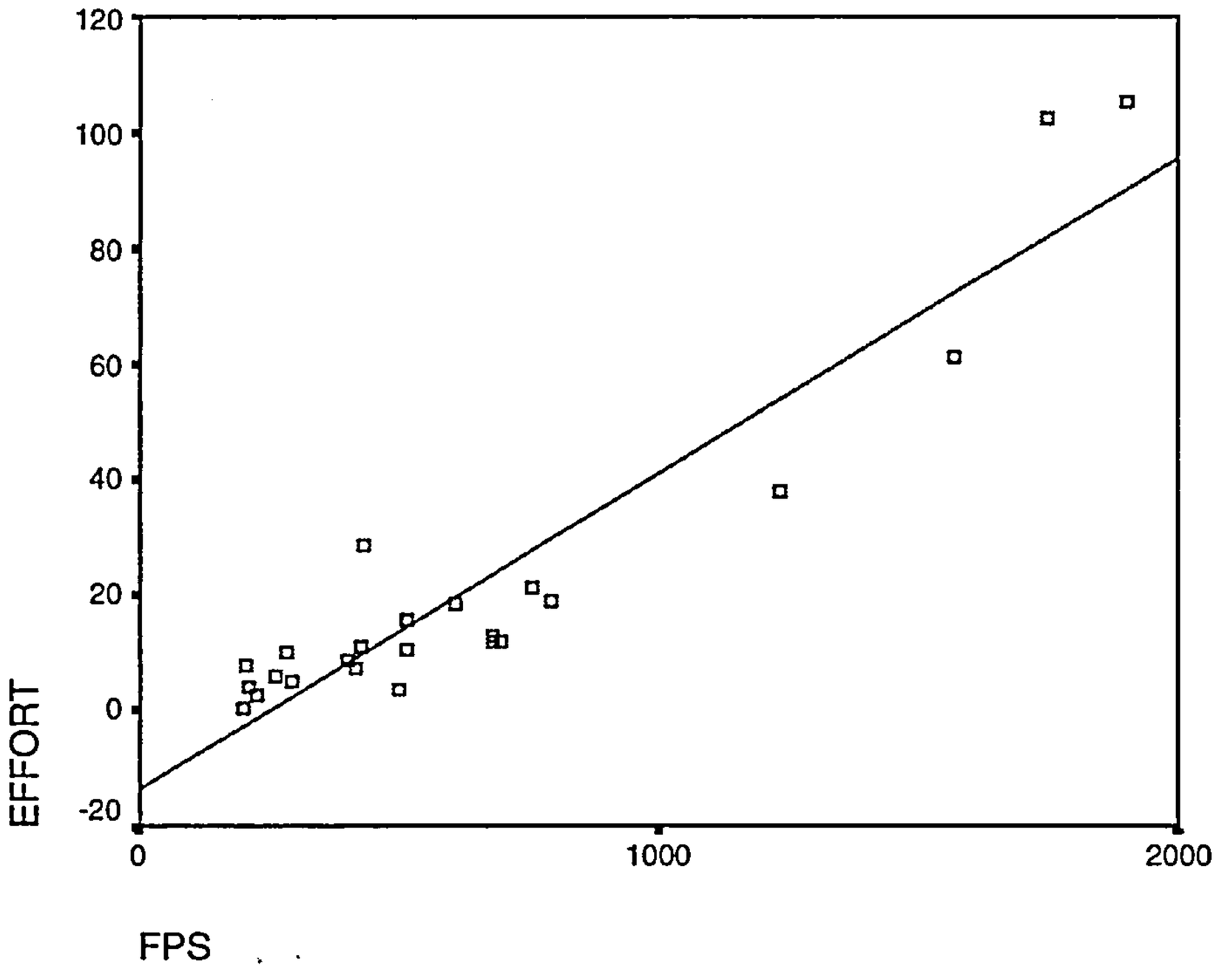


Figure 5.1 : Scatterplot of effort vs Function Points

The prediction system built by stepwise regression rejects the majority of the features as not contributing significantly to a model based on *function points* and *number of inquiries* and thus the regression equation becomes:

$$Effort = -12.08 + (0.04 * function\ points) + (0.42 * number\ of\ inquiries). \quad \textbf{(Eqn. 5.1)}$$

The adjusted R^2 of 0.90 suggests that the model is good with 90% of the variation in effort being explained by variation in *function points* and *number of inquiries*, however the negative

intercept value is not intuitively very appealing as it suggests negative fixed costs. The next step is to predict each project in turn using this model and using the difference between the estimates generated and the known effort to derive the MMRE and Pred(25) statistics.

The analogy results were obtained by using the facility that allows the best subset of features to be found (see Section 4.2). Note the best subset of features is often different, dependent on the number of analogies to be found. The associated MMRE and Pred(25) statistics represent the predictive performance of each of the four analogy selection techniques.

Table 5.3 shows the results from stepwise regression and estimation by analogy in terms of MMRE and Pred (25).

Prediction Method	MMRE	Pred(25)
Stepwise Regression	74%	25%
One Analogy	67%	33%
Two Analogies	66%	37%
Two Analogies (weighted)	61%	41%
Three Analogies	62%	33%

Table 5.3 Regression vs Analogy for the Albrecht data set

As can be seen from table 5.3, estimation by analogy out performs stepwise regression in all cases for both MMRE and Pred(25). For this data set at least, the optimum number of analogies appears to be two, when the first analogy is weighted double, which returns an MMRE of 61% and a Pred(25) of 41%. Note how similar the four estimation by analogy results are (a 6% difference in MMRE and an 8% difference in Pred(25), between the best and worst results), a pattern that will be repeated throughout this analysis.

The same procedure is followed for the remaining 7 data sets, with the overall results presented in Tables 5.15 and 5.16.

5.3.2 The Desharnais data set

It is uncommon in the field of software project estimation to come across a data set as large as that collected by Desharnais (1988) from a Canadian software house. At 77 projects, collected over 3 different development environments, the data set is twice the size of the next largest data set in this investigation. The data set is also relatively rich in features that can potentially be used to estimate effort (for a full summary of the features see appendix A.2).

The Stepwise regression equation again rejected most of the features as not contributing significantly to its initial model:

Effort = 150.816 + (16.454*adjFP) **(Eqn. 5.2)**

with an adjusted R^2 of 0.53.

Prediction Method	MMRE	Pred(25)
Stepwise Regression	66%	42%
One Analogy	37%	45%
Two Analogies	38%	37%
Two Analogies (weighted)	36%	37%
Three Analogies	34%	49%

Table 5.4 Regression vs Analogy for the Desharnais data set

The strategy of searching for three analogies was found to be the most accurate for this data set, predicting 49% of the projects within 25% of their actual effort and achieving an MMRE nearly twice that of the stepwise regression model (Table 5.4). The three other analogy approaches were also notably superior to stepwise regression in terms of MMRE but similar in terms of Pred(25)

Recall from chapter two that algorithmic techniques such as stepwise regression prefer more homogenous data and that provided the number of projects remains high enough for statistical relationships to be found, they will generally be more accurate when data is partitioned into related groups. In the case of the Desharnais data set, it is possible to partition the data based upon the three different development environments of 44, 23 and 10 projects respectively, and doing so improves the accuracy of stepwise regression from the overall figure of MMRE = 66% to 41%, 29% and 49% respectively. At the same time the best analogy approach improves in two of the sets, and worsens in one, while still remaining superior to stepwise regression.

Data set	Stepwise Regression	Analogy
Desharnais - Dev Env 1	41%	37%
Desharnais - Dev Env 2	29%	29%
Desharnais - Dev Env 3	49%	26%

Table 5.5 MMRE results for the partitioned Desharnais data sets

Data set	Stepwise Regression	Analogy
Desharnais - Dev Env 1	45%	47%
Desharnais - Dev Env 2	48%	47%
Desharnais - Dev Env 3	50%	70%

Table 5.6 Pred(25) results for the partitioned Desharnais data sets

Partitioning also appears to have a positive effect on the Pred(25) values for stepwise regression improving the figure from Pred(25) = 42% to 45%, 48% and 50%. The result for Dev Env 2 is particularly interesting where the stepwise regression model performs marginally better than estimation by analogy. Another interesting result is obtained by using estimation by analogy on Dev Env 3. Although only 10 project cases are available, accuracy figures of MMRE = 26% and Pred(25) = 70% are returned (the best within this investigation and soberingly, the only results close to the values Conte et al. (1986) considered should be returned by a good effort prediction system) providing some insight into the question of the minimum number of project cases needed to make estimation by analogy viable.

5.3.3 The Finnish data set

The Finnish data set, comprising 38 projects from a variety of different organisations, was very rich in predictor features, having 29 in all. This provided a problem for the analogy estimation approach as finding the best subset of 29 features on a Pentium 200 would take approximately 20 years! However, performance figures for analogy were generated by analysing smaller subsets of features. This meant that the optimal subset may not have been found, however, by adopting informal search heuristics, it was possible to find subsets of features that were considered to be close to the optimal subset and that produced accurate results.

Using stepwise regression on the available continuous features (see appendix A3) resulted in the following model, with an adjusted R^2 of 0.39:

Effort = 899.709+(121.975*ON) + (148.863*FN) **(Eqn. 5.3)**

Prediction Method	MMRE	Pred(25)
Stepwise Regression	114%	26%
One Analogy	48%	23%
Two Analogies	42%	44%
Two Analogies (weighted)	41%	39%
Three Analogies	52%	26%

Table 5.7 Regression vs Analogy for the Finnish data set

Although not necessarily able to find the best subset of all the features, table 5.7 shows that analogy is still able to estimate effort almost 3 times more accurately than stepwise regression when searching for two analogies, weighted and unweighted. This time, the most accurate technique in terms of the Pred(25) statistic is the two analogies approach while choosing two analogies weighted returns a marginally superior MMRE result.

5.3.4 The Hughes data set

The Hughes data set collected from a large size telecommunications company contains information on 33 projects with 14 features (summarised in appendix A.4) including development effort.

The stepwise regression equation has a relatively high adjusted R^2 of 0.80, and produces the following model:

Effort = 626.87 + (205.40*C5) + (559.36*C3) + (313.54*C8) + (-854.28*C4) + (55.66*C9)
+ (-26.91*C10) **(Eqn. 5.4)**

An interesting quirk of this model is that the amount of effort needed to complete a project is increased by the experience of the block designer (C9). On the surface, this appears to be another example of a counter intuitive model, however further investigation might reveal a plausible reason for this such as, the more experience designers being given the more difficult tasks.

Prediction Method	MMRE	Pred(25)
Stepwise Regression	72%	42%
One Analogy	37%	51%
Two Analogies	40%	39%
Two Analogies (weighted)	40%	45%
Three Analogies	37%	39%

Table 5.8 Regression vs Analogy for the Hughes data set

However, as Table 5.8 shows, the regression model shows a poor level of accuracy, in terms of MMRE when compared with the best analogy technique which predicts effort on average to within 37% of the actual figure. The picture is less clear cut for the Pred(25) measure although the best analogy (One Analogy) method is still superior.

5.3.5 The Kemerer data set

The Kemerer data set contains two features (summarised in appendix A.5) that can be exploited by regression and analogy for the purposes of estimating effort, these are adjusted Function Points and unadjusted Function Points. Even though there is a high level of correlation ($r_s = 0.98$) between the two variables it was decided to use both features with the expectation that stepwise regression would eliminate the least useful one.

The regression equation generated was Effort = -121.57 + (0.34* FP) which has an adjusted R^2 of 0.55. Table 5.9 summaries the MMRE and Pred(25) values for regression and the best analogy method

Prediction Method	MMRE	Pred(25)
Stepwise Regression	106%	13%
One Analogy	68%	26%
Two Analogies	62%	40%
Two Analogies (weighted)	62%	26%
Three Analogies	64%	40%

Table 5.9 Regression vs Analogy for the Kemerer data set

The results in table 5.9 seem to follow the same pattern as those before it (i.e. analogy significantly out-performing regression) with the strategy of taking the mean of the closest two project's effort being the most fruitful approach this time. Note that this is the data set

with, at 15, the smallest number of projects. This does not, however, seem to pose a great problem to the analogy technique which is able to predict effort within 25% of its actual figure for 40% of projects.

5.3.6 The MERMAID data set

The MERMAID data set comprises 28 new and enhancement projects. The predictor features available include unadjusted and adjusted Function Points counts and the fourteen General System Characteristics that convert the former Function Point count into the latter. Using the Function Point Counts with stepwise regression generates the following model:

$$Effort = 3060.183 + (15.626*RawFP) \qquad \textbf{(Eqn. 5.6)}$$

$Adj R^2 = 0.20.$

Prediction Method	MMRE	Pred(25)
Stepwise Regression	251%	14%
One Analogy	78%	21%
Two Analogies	95%	3%
Two Analogies (weighted)	92%	3%
Three Analogies	117%	28%

Table 5.10 Regression vs Analogy for the MERMAID data set

The mermaid data throws up a number of interesting results. An interesting quirk of using analogy on this data set was that ANGEL chose the *Development Environment*[29] feature as the sole best feature for all of the four analogy techniques. This of course meant that the first E and N type projects were being selected as the source of analogies for each project estimate[30] in the jack-knife procedure. This situation would not normally be expected to yield a good MMRE, however, a combination of the fact that the MMRE measure favours under-estimates and that the first N and E projects in the data set are the smallest in terms of effort, means that the use of the *Development Environment* feature predicts with an MMRE of 69% but, perhaps more revealingly, with a balanced MMRE[31] of 658%. As a consequence, *Development*

---

[29] A categorical indicator of project type , with three values: N – New project; E – Enhancement project; N/A – not applicable.

[30] The way ANGEL handles ties is primitive. If the Euclidean distance of two or more projects is equal, the first project encountered will always be chosen as the closest analogy.

[31] Recall that balanced MMRE unlike MMRE treats over and under-estimates equally.

Environment is removed from the analysis by ANGEL and the results present a more realistic picture.

From Table 5.10, the regression model, with an adjusted R^2 of 0.20, a Pred(25%) of 14% and an MMRE of 251%, does not seem to be able to model the relationships implicit in the data and consequently is a very ineffectual prediction system. The analogy technique, while over three times more accurate than stepwise regression on average, is also not very convincing with a best MMRE of 78%, the poorest result of all the data sets analysed.

MERMAID provides the second opportunity for partitioning of a data set into more homogenous groups, based upon the two development environments containing 18 and 8[32] projects respectively.

Data set	Stepwise Regression	Analogy
MERMAID E	62%	53%
MERMAID N	N/A	60%

Table 5.11 MMRE results for the partitioned MERMAID data sets

Data set	Stepwise Regression	Analogy
MERMAID E	27%	39%
MERMAID N	N/A	25%

Table 5.12 Pred(25) results for the partitioned MERMAID data sets

In the case of the Mermaid E (enhancements) data set, both techniques improve with analogy still the superior technique. However, in part due to the small size of the MERMAID N (new) data set it was not possible to find statistically significant relationships between effort and any of the predictor features which meant that no regression model could be built to compare with the analogy results.

5.3.7 The Real-Time 1 data set

The Real-Time1 data set is interesting due to nature of the features (summarised in appendix A.7) available at the time estimation is required. Not only is there no size related feature, but also all three features in this data set (host machine, life-cycle, and document standard) are

[32] Note that two of the projects cannot be placed in either group.

nominal and thus the building of a regression model is ruled out, since it would be comprised only of dummy variables.

However, the use of analogy was not without its problems. When using ANGEL to analyse the data, a similar situation to that encountered when using a nominal feature for the MERMAID data set occurred. ANGEL in all four cases selected just one of the nominal features, which inevitably led to the repeated selection of the same projects as the source of analogy.

Late on in the analysis a solution to this problem was found with the data reanalysed with each of the 3 features dis-aggregated so that each category value became a binary feature, with a value of 1, indicating the presence of the feature and 0 absence. As an example, the host machine feature was expanded into VAX, SUN and IBM-PC features and a project that was to use a VAX as the host machine would record a 1 for that feature and 0 for the SUN and IBM_PC features. In all other ways the analysis method remained unchanged.

Prediction Method	MMRE	Pred(25)
Stepwise Regression	N/A	N/A
One Analogy	65%	28%
Two Analogies	59%	19%
Two Analogies (weighted)	62%	23%
Three Analogies	60%	14%

Table 5.13 Regression vs Analogy for the Real-Time1 data set

The results in table 5.13 suggest that estimation by analogy is capable of producing acceptable estimates in situations where only categorical data is available. The importance of this cannot be overlooked, as it is often the case that the only type of data available to an estimator (especially at the bidding stage of a contract) is categorical in nature. Also this type of data is often easier to obtain and more likely to be free from errors. On the other hand any results obtained from data sets comprised solely of categorical data should be treated with caution.

5.3.8 The Telecoms 1 data set

The Telecoms1 data set is characterised by only one potential descriptor feature, the number of files, which can be determined with reasonable accurately early on in the project life-cycle.

The regression line that describes the linear relationship between effort and the number of files is:

effort = 95.18 + (1.89*files) **(Eqn. 5.7)**

which has an adjusted R^2 figure of 0.39 suggesting a considerable amount of scatter from the regression line, not a good basis for prediction.

Prediction Method	MMRE	Pred(25)
Stepwise Regression	86%	44%
One Analogy	39%	44%
Two Analogies	51%	55%
Two Analogies (weighted)	46%	50%
Three Analogies	73%	44%

Table 5.14 Regression vs Analogy for the Telecoms1 data set

Table 5.14 shows that analogy is again the superior approach, with the selection of just one analogy the optimum technique in terms of MMRE and two analogies the optimum approach in terms of Pred(25).

5.4 Summary of Results

Data set	Analogy	Stepwise Regression
Albrecht	61%	74%
Desharnais	34%	66%
Desharnais - Dev Env 1	37%	41%
Desharnais - Dev Env 2	29%	29%
Desharnais - Dev Env 3	26%	49%
Finnish	41%	114%
Hughes	37%	72%
Kemerer	62%	106%
Mermaid	78%	251%
Mermaid E projects	53%	62%
Mermaid N projects	60%	N/A
Real-Time1	59%	N/A
Telecoms1	39%	86%

Table 5.15 Summary of comparison between analogy and stepwise regression using MMRE

Tables 5.15 and 5.16 summarise the results of predicting effort for 8 industrial data sets (plus 5 subsets) using estimation by analogy and stepwise regression. Note, The MMRE and Pred(25)

result are in each case taken from the analogy approach which obtained the best MMRE figure. Although on the surface this appears to penalise the stepwise regression model, in reality the user will wish to find the most accurate number of analogies before making an estimate.

The most striking feature of both these tables is that the analogy technique equals or, more commonly, out-performs the regression technique in all but 1 of the 11 data sets for which both techniques could be applied. The one exception is the Desharnais-2 data set which shows fractionally superior performance for regression based prediction when using the Pred(25) indicator. This seemingly gives overwhelming evidence that the analogy technique is superior to stepwise regression based algorithmic methods, at least for the data sets under examination.

Data set	Analogy	Stepwise Regression
Albrecht	41%	25%
Desharnais	49%	42%
Desharnais - Dev Env 1	47%	45%
Desharnais - Dev Env 2	47%	48%
Desharnais - Dev Env 3	70%	50%
Finnish	39%	26%
Hughes	51%	42%
Kemerer	40%	40%
Mermaid	21%	14%
Mermaid E projects	39%	27%
Mermaid N projects	25%	N/A
Real-Time1	19%	N/A
Telecoms1	44%	44%

Table 5.16 Summary of comparison between analogy and stepwise regression using Pred(25)

To test the validity of this claim, a one-tailed Wilcoxon signed pair test on the a) MMRE and b) Pred(25) results was calculated.

The stated null hypothesis for (a) was:

> H_0: Estimation by analogy is not more accurate at predicting software development effort than stepwise regression using the MMRE indicator

to be rejected in favour of the alternative hypothesis:

H_1: Estimation by analogy is more accurate at predicting software development effort than stepwise regression using the MMRE indicator.

While the stated null hypothesis for (b) was:

H_0: Estimation by analogy is not more accurate at predicting software development effort than stepwise regression using the Pred(25) indicator

to be rejected in favour of the alternative hypothesis:

H_1: Estimation by analogy is more accurate at predicting software development effort than stepwise regression using the Pred(25) indicator.

Note that a direction has been specified in both the alternative hypotheses, which represents a belief that estimation by analogy is superior in terms of accuracy to regression analysis.

Wilcoxon's signed pair test considers information about both the sign of the differences and the magnitude of the differences between pairs, in this case, the results obtained for analogy and stepwise regression. The test based on the MMRE figures for the 11 pairs produced a significant result where p= 0.001 thus the null hypothesis can be rejected at the 0.01 level of confidence and the alternative hypothesis that 'estimation by analogy is a superior to stepwise regression for the MMRE indicator' is accepted. Similarly, the test based on the Pred(25) figures for the 11 pairs produced a significant result where p= 0.0054 thus the second null hypothesis can also be rejected at the 0.01 level of confidence and the alternative hypothesis that 'estimation by analogy is a superior to stepwise regression for the Pred(25) indicator' is also accepted.

Since the original submission of this thesis a more appropriate test[33] of the significance of results from ANGEL has been devised by researchers (Stensrud and Myrtveit 1998) looking at the added value analogy can provide to expert judgement. Managers at Anderson Consulting were asked to estimates a series of software projects three time with first of all no supplementary data, second, access to historical data and third, access to historical data and ANGEL. The overall MMRE figures demonstrated that ANGEL did indeed improve on the estimates based upon historical data alone. However, to test the significance of this result

---

[33] The Wilcoxon signed rank test is sometimes seen as being too optimistic in that very few positive results are required for the null hypothesis to be rejected.

Stensrud and Myrtveit used a t-test on the mean difference between MRE pairs. The test confirmed at the 10% confidence level that ANGEL did indeed improve estimates over those based upon historical data alone.

5.5 Discussion

From the empirical analysis it can be seen that, at least for the sample of data sets analysed, estimation by analogy is superior in terms of results, to estimation based on stepwise regression. Although it must be remembered that the analogy technique optimises on MMRE, the results seen in terms of Pred(25) also show analogy to be superior, if to a lesser extent. Further, from the results, estimation by analogy would seem to have a number of previously unstated advantages over algorithmic prediction systems. First, estimation by analogy succeeds in creating estimates on data where no statistical relationships have been found (e.g. the MERMAID N data set). Second, estimation by analogy remains viable for data based solely on categorical features (e.g. the Real-Time1 data set). Third, estimation by analogy remains accurate for small data sets (e.g. the Kemerer (15 projects), MERMAID N (8 projects) and Telecoms1 (18 projects) data sets). And fourth, estimation by analogy remains accurate where the number of features is limited (e.g. the telecoms1 data set).

Another interesting point is that, for all of the data sets under consideration, the function to find the best subset of features improved, in every case, upon the result of using all the features together. This demonstrates the usefulness of the process of removing noisy features.

This analysis has also highlighted a couple of flaws with the analogy technique. First, the use of categorical attributes can cause a problem under certain conditions (e.g. with the MERMAID and Telecoms1 data sets) where the flawed nature of the MMRE measure is highlighted. As a result, a new strategy for dealing with wholly categorical data sets was developed. Even so, the fact remains that estimates from data sets relying wholly on categorical features should be treated with caution. Second, the use of an exhaustive search to find the best subset of features is computationally expensive and impossible for one of the data sets under study.

This analysis also permits an empirical evaluation of a number of questions relating to the most effective use of estimation by analogy:

- *What is the optimum number of analogies for ANGEL to search for?*

  Table 5.17 shows the optimum number of analogies to use in ANGEL for each data set. 'One Analogy' is the most commonly accurate estimation method, being selected for 5

out of the 13 data sets. 'Two Analogies' is the most accurate 4 times, and both 'Three Analogies' and 'Two Analogies Weighted' are most accurate twice respectively. One of the assumptions made early on in the project was that the selection of just one analogy would be more suitable on small data sets while a larger data set would favour the selection of more analogies[34]. This however, has not been borne out in the results, with for example, the two smallest data sets, MERMAID-N and Desharnais-3 finding respectively two and three analogies to be the optimum number to search for. Even though the selection of 'One Analogy' or Two Analogies' seem to be superior to both 'Three Analogies' and 'Two Analogies weighted', it must be remembered that for many of the data sets the four different method returned remarkably similar accuracy levels.

Data Set	No. of Project Cases	Optimum No. of Analogies
Albrecht	24	Two (weighted)
Desharnais	77	Three
Desharnais - Dev Env 1	42	One
Desharnais – Dev Env 2	23	Two
Desharnais - Dev Env 3	10	Three
Finnish	38	Two (weighted)
Hughes	33	One
Kemerer	15	Two
Mermaid	28	One
Mermaid E projects	18	One
Mermaid N projects	8	Two
Real-Time1	21	Two
Telecoms1	18	One

Table 5.17 Optimum no. of analogies for each data set

- *Does accuracy improve with more homogenous data?*

  The answer to this question appears to be yes, based upon the limited evidence provided by the Desharnais and MERMAID data sets where, in both cases, partitioning of the data set led to more accurate estimates. However, the scale of improvement was relatively poor when compared to that seen in the regression models. This is possibly

[34] The theory behind this assumption was that, where there are a great number of projects, the chances are that for a given new project, there will be more projects similar (clustered close to it) that can be used as analogies than for a data set with few projects.

because analogy bases an estimate for a project on the cluster of projects that are most similar i.e. to a certain extent it automatically partitions the data.

- *What is the minimum number of cases that can be used as source analogies?*

  The smallest number of cases successfully analysed by ANGEL is 8 from the MERMAID-N data set. However, the criteria by which the cut off point in the minimum number of projects to be used is judged, remains unclear. Chapter 6 will analyse this question in more detail.

- *Does accuracy improve on larger data sets?*

  Figures 5.2 and 5.3 show the accuracy of each data set (in terms of MMRE and Pred(25) respectively) plotted against data set size. For both accuracy indicators it is very difficult to discern any consistent pattern which can, in part, be put down to the small sample of data. In terms of MMRE, the best figure results from the second smallest data set (10 projects), however on the whole, the smaller data sets (less than 30 projects) tend to return the poorest results (6 out of 9 over 50% MMRE). The MMRE values for the 4 larger data sets (greater than 30 projects) are more consistent around the 40% mark.

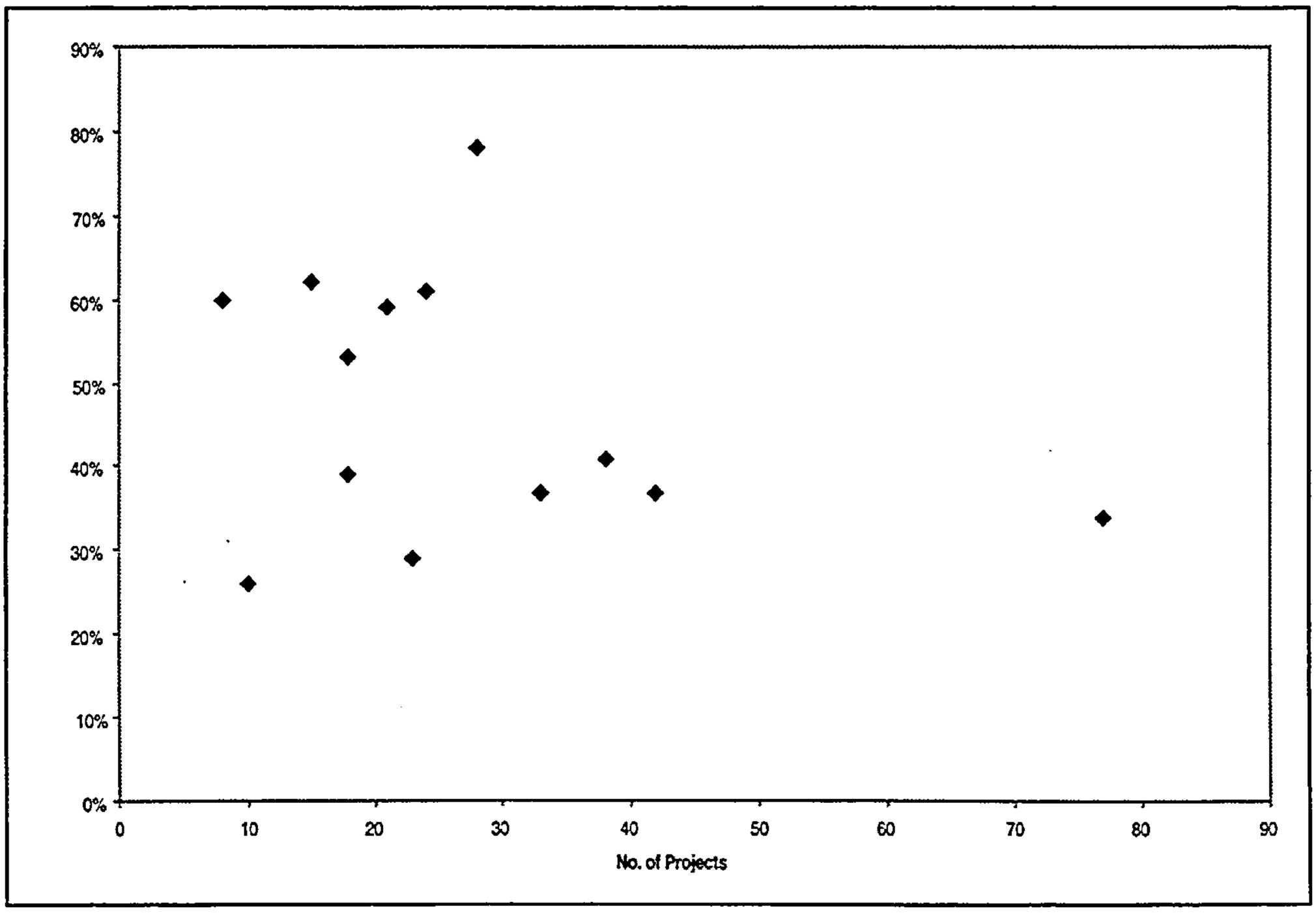


Figure 5.2 : MMRE by no. of cases

The results in terms of Pred(25) are very similar to those seen for MMRE. Again the best result is returned by the second smallest data set however, it is the smaller data sets that are again responsible for the worst results.

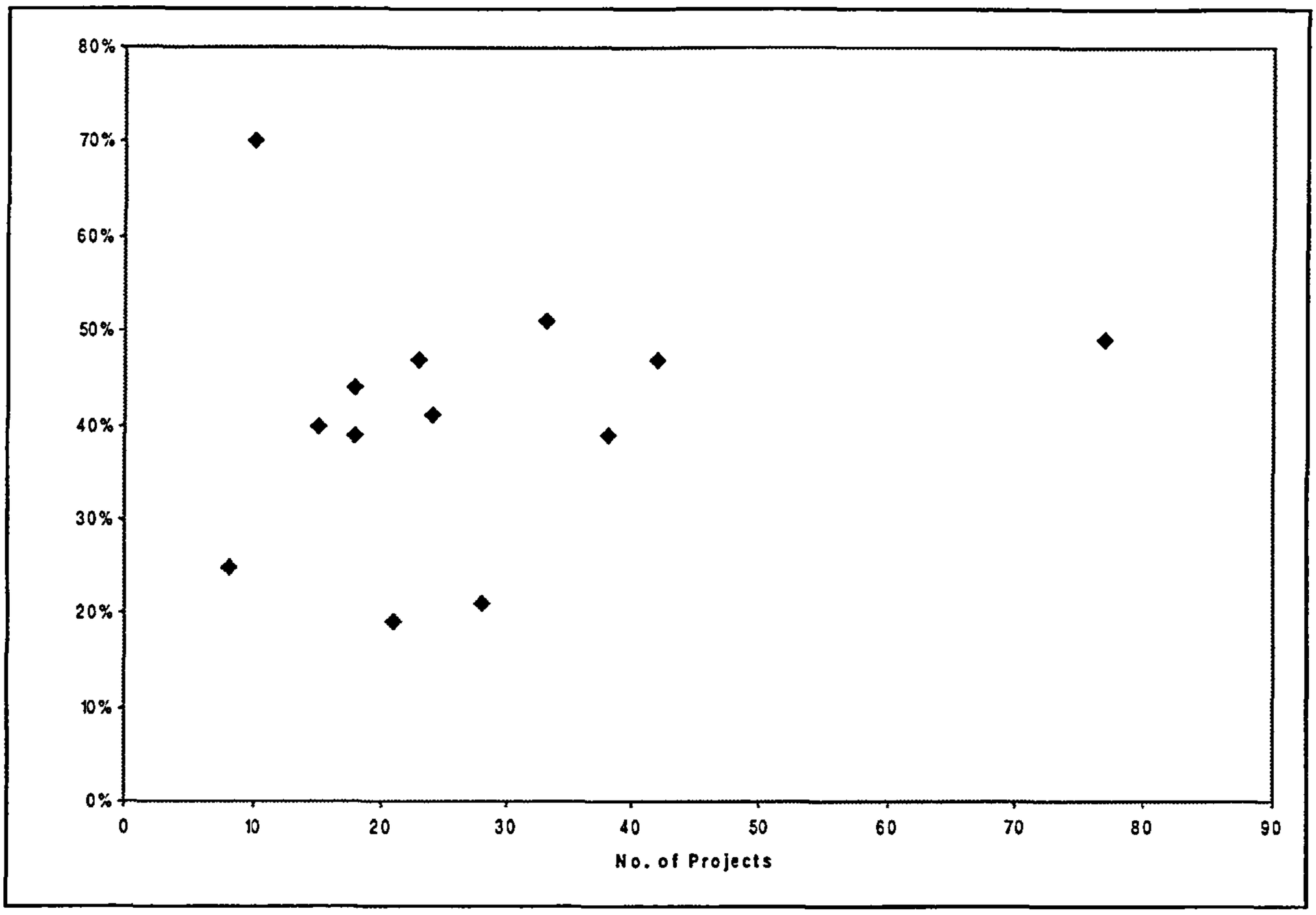


Figure 5.3 : Pred(25) by no. of cases

Perhaps the only conclusion that can be drawn on this question is that, the larger the data set, the more consistent the results are likely to be. This question will be looked at again from a different view point in the next chapter, where individual data sets will be examined to see if accuracy improves as data points are added.

5.6 Summary

In summary, after analysing over 250 software projects, the hypothesis that estimation by analogy is a superior technique for predicting software development effort than a regression based analysis approach has been accepted. This analysis has allowed us to answer (within the limitation mentioned previously) questions about the analogy approach, such as, *what is the optimum number of analogies for ANGEL to search for?* and *does accuracy improve on larger data sets?*. The chapter has also highlighted circumstances under which analogy is a more

appropriate technique than the use of algorithmic models, i.e. where the data set is too small for any statistical relationships to be found; where nominal data is prevalent and where the data is heterogeneous.

Chapter 6

An Investigation into the Sensitivity of Estimation by Analogy

6.0 Introduction

The previous chapter analysed the overall accuracy of the estimation by analogy prediction system on static sets of historical project data. However, whilst extremely useful as a general indicator of predictive accuracy, this kind of analysis only provides a snapshot of the accuracy on a data set at a specific point in time. In reality, a prediction system will evolve over time. New projects will be estimated and when completed, their data will be used to enhance the prediction system. To judge a prediction system based on its accuracy for data sets that can contain many projects, (e.g. the Desharnais data set), is to neglect the fact that data sets are continually growing over time and that the size of the data set is likely to have a strong effect on accuracy and stability of estimates.

The use of sensitivity tests allows a more focused examination of the performance and behaviour of a prediction system. By devising a test that simulates the dynamic growth of a data set over time, questions, such as how many data points are needed for estimation by analogy to be viable and how vulnerable is its accuracy to the addition of a single outlying project, can be answered.

This chapter will describe an analysis that was devised to investigate the dynamic behaviour of the analogy technique. The results from 3 runs of the test on 4 data sets are presented in section 6.3 and discussed in section 6.4 and 6.5. Note that the full complement of 8 data sets was not used, due to the time consuming nature of this style of analysis.

6.1 Questions to be Answered

The sensitivity analysis was designed to answer the following questions:

- Does accuracy improve as the number of project cases increases?
- What is the least number of project cases needed before estimation by analogy becomes stable?
- Is estimation by analogy vulnerable to the addition of outlying[35] data points?
- Is there a recognisable point at which estimation by analogy becomes stable?

The answers to these questions will inevitably be found in the way accuracy changes as data points are fed into the prediction system.

6.2 Design of the Sensitivity Analysis

The analysis procedure involved randomly numbering the projects from 1 to n (where n is the number of projects in the data set). Projects were then added to an empty data set one at a time in their random number order. Thus each data set grew until all of the projects had been added. For each partial data set (starting from two projects) the 'best set of features' function was employed searching for two analogies (unweighted) and the mean absolute prediction error (MMRE) associated with that subset was used as the measure of accuracy thus n-2 accuracy measures were taken. This procedure was repeated three times for each data set under study so as to guard against freak results arising from the randomising procedure.

As was mentioned earlier, constraints on time meant that only four of the data sets could be analysed. The Albrecht and Kemerer data sets were selected as examples for which a comparatively low level of accuracy was achieved. In contrast the Hughes and Telecoms1 data sets showed the highest levels of accuracy.

---

[35] Outlying data points for ANGEL are similar but not identical to outlying data points for algorithmic models. An outlying data point in an analogy system is a point for which there are no suitable analogies identifiable in the case-base. Unlike for algorithmic prediction systems, it only takes the inclusion of one similar outlying project (assuming only one analogy is being searched for) to the case-base to remedy the problem. However, unlike algorithmic systems, analogy is unable to interpolate or extrapolate to form estimates and thus any point sufficiently outside of its current knowledge will be an outlying data point.

6.3 Sensitivity Analysis Results

Figures 6.1 to 6.4 show the accuracy of each of the four data sets over time

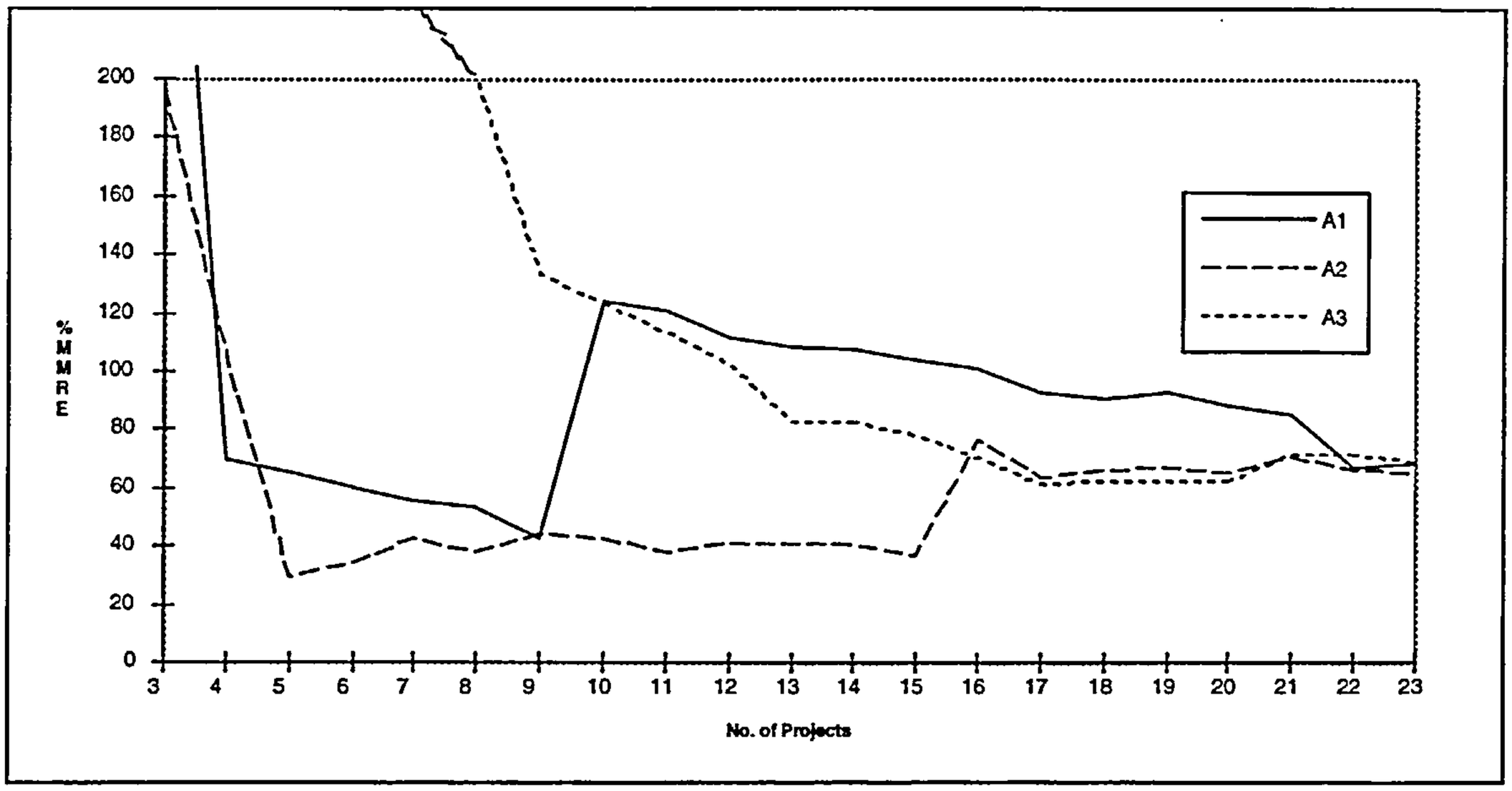


Figure 6.1 : Estimation accuracy over time (Albrecht data set)

Figure 6.1. Shows the behaviour over time of the Albrecht data set. The most striking feature of this analysis is the dissimilarity of the three lines. A1, after a period of increasing accuracy up to the addition of the ninth project, experiences a sudden 100% reduction in accuracy when the tenth project is added. However, after the tenth project, the trend is again a steady increase in accuracy. A2 reaches a high level of accuracy early on (after the fifth project) and maintains an MMRE around 40% up until the sixteenth project is added. The addition of the sixteenth project causes the accuracy level to suddenly decrease to approximately 80% in a way similar, to that seen in A1. From the seventeenth project onwards, accuracy is again consistent between 60% and 70% MMRE. The path of A3 is less turbulent than its predecessors. Starting off with a very poor level of accuracy (the MMRE being well over 200%) until the addition of the ninth project, A3 slowly improves and reaches a consistent level, at approximately 65% MMRE, after 17 project have been added.

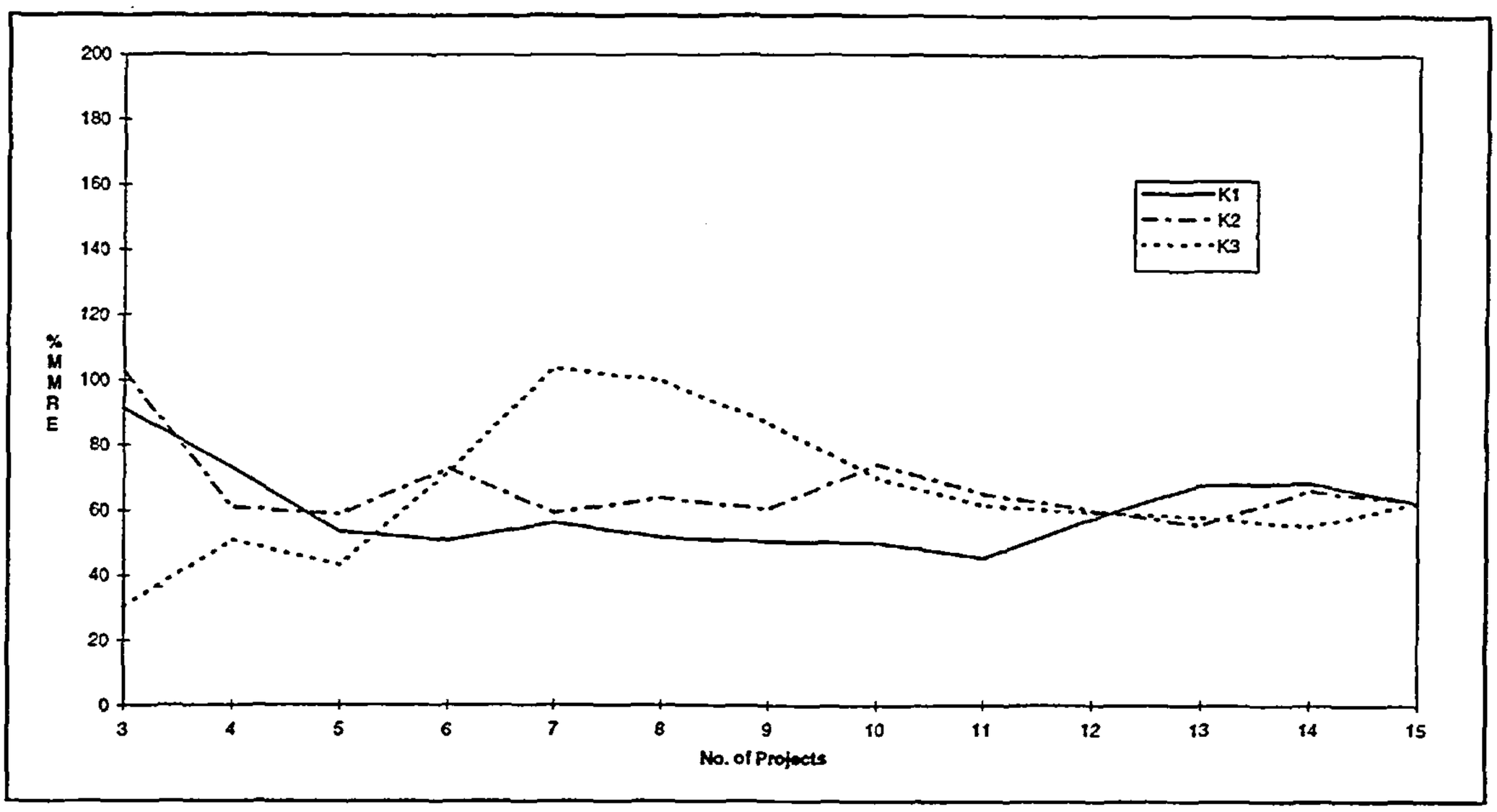


Figure 6.2 : Estimation accuracy over time (Kemerer data set)

The dynamic behaviour of the Kemerer data set is shown in Figure 6.2. Two of the random series (K1 and K2) show very consistent behaviour from the addition of the fifth project onwards with both contained within the boundaries of 50% and 70% MMRE. Random series K3 however, is far less consistent, moving from an initial accuracy peak of 30% to a trough of over 100% MMRE after the seventh project. After this point however, accuracy increases until all the projects have been added.

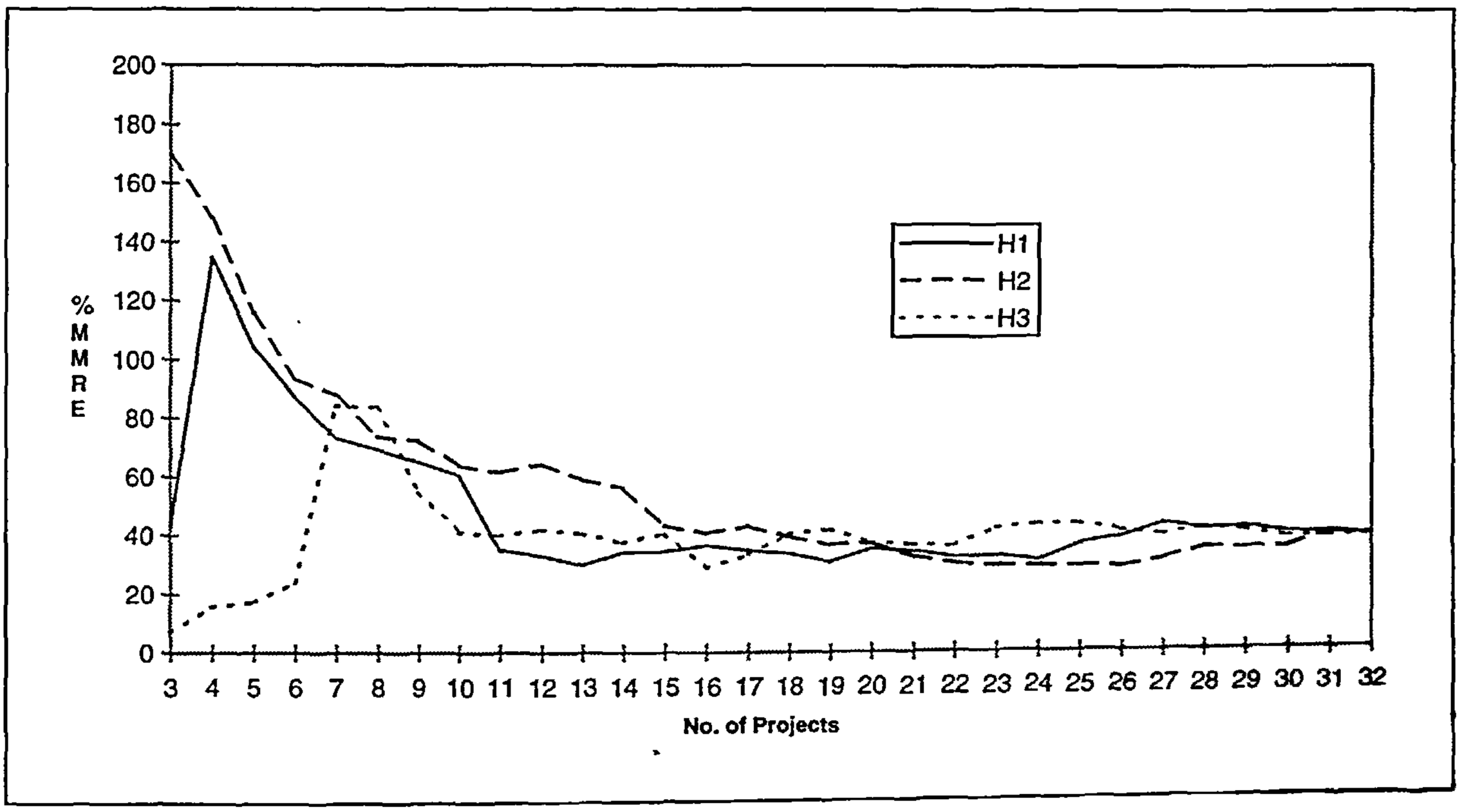


Figure 6.3 : Estimation accuracy over time (Hughes data set)

Figure 6.3. Shows the behaviour over time of the Hughes data set. In contrast to Figure 6.1, the most striking feature of the three random series is their similarity and their consistent level of accuracy after fifteen projects have been added. The three lines begin to converge after the seventh project, with two (H1 and H3) of the three experiencing large reductions in accuracy (similar to those seen in A1 and A3) previous to that point.

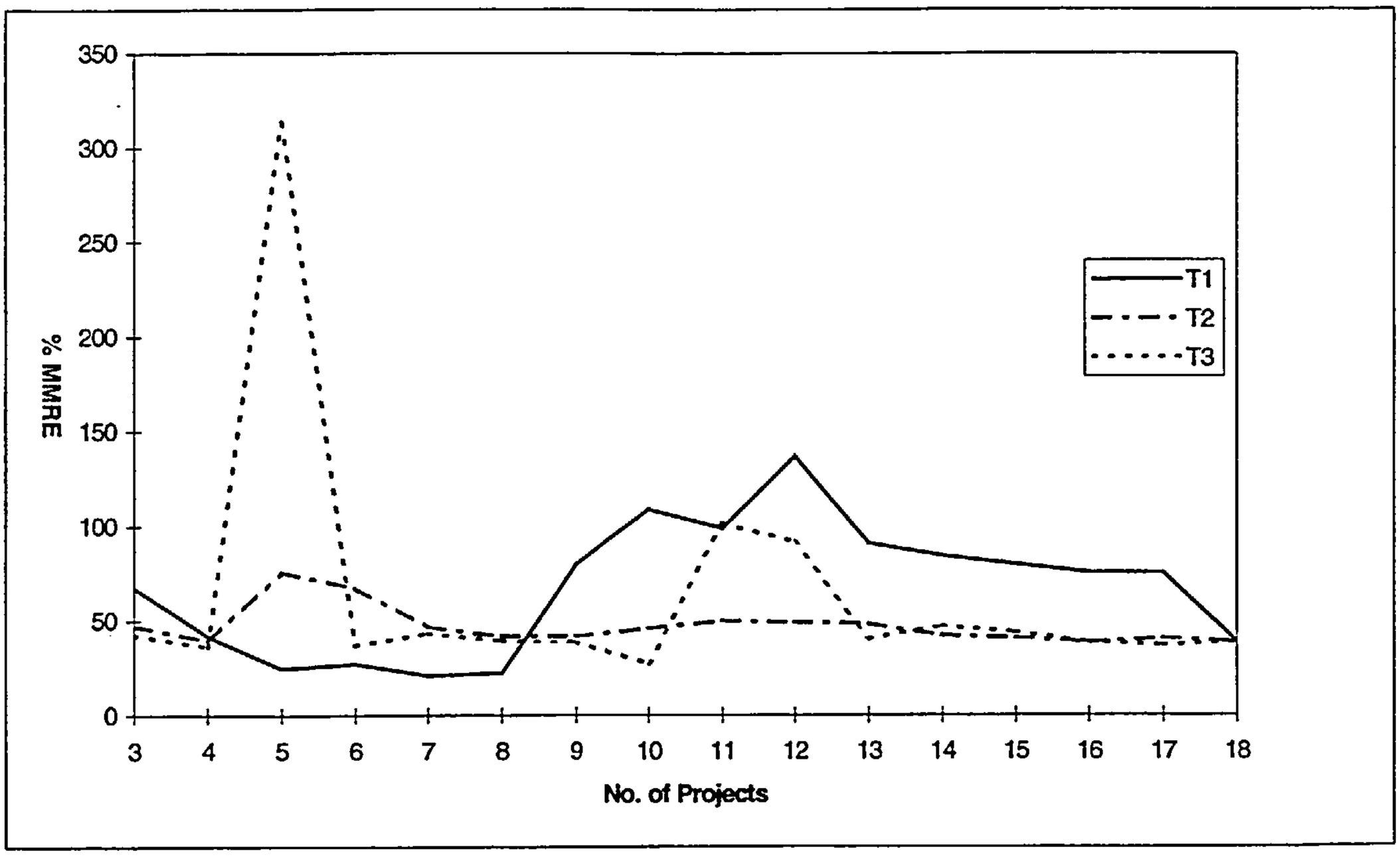


Figure 6.4 : Estimation accuracy over time (Telecoms1 data set)

The three random series in Figure 6.4 exhibit very contrasting behaviour. T1 shows the most volatile pattern with a peak of approximately 30% MMRE, and a trough of approximately 140% MMRE. T2 is the most stable of the three series, after a little variability before 7 projects, the level of accuracy remains markedly constant around 50%. T3 is generally stable at just below 50% MMRE but experiences two major 'blips' at 5 and 11-12 projects.

6.4 Discussion of Results

Overall, Figures 6.1, 6.2, 6.3 and 6.4 show that there is a tendency for the MMRE level to improve as the size of the data set grows. Exceptions to this can be seen in both the Albrecht and Telecoms1 analysis and appear to be caused, in the main, by the addition of outlying data points, which will be discussed later. For many of the random series, there is a point at which

the level of accuracy begins to stabilise, which indicates that estimation by analogy can be a high risk technique at below this number of projects.

After early fluctuation, the Hughes data set exhibits little improvement in accuracy levels beyond 15 projects. This is a trend also evident in the other three data sets at approximately this point, although, with the exception of the Albrecht data set, they really contain too few data points to provide compelling evidence. This suggests that data set size is not the most important factor determining accuracy, and indeed that the approximate accuracy of the analogy prediction system is determined early (say after 20 projects have been added) and is then relatively robust (to the introduction of outlying data points) from that point onwards.

An interesting feature of Figure 6.1 is the sharp rise in the MMRE values that occur after 10 projects have been added for random sequence A1 and 16 have been added for random sequence A2. Further investigation reveals that both of these anomalies are linked to the introduction of the same project. The project is third in sequence A3, when predictions are still very poor and thus doesn't show so strongly. A similar pattern occurs in Figure 6.4. Sharp rises are seen in T1 and T3 at the stage where 9 and 5 projects have been added respectively. These rises correspond to the addition of the same project: number 16. Within the Telecoms1 data set, projects 16, 17 and 18 are different in nature to the remaining 15 projects and required relatively little effort to complete. As a result when the first one of these three projects is added there are no analogous projects in the case-base and the estimate made for this project is wildly inaccurate. The reason for the dramatic recovery seen in the MMRE figure for T3 is the addition of project 17, which is very similar to project 16. Note, the same pattern is not seen in T2 because projects 16 and 18 are introduced from the start. All this suggests that the results from estimating by analogy, like regression, can be influenced by outlying projects. However, A1 and A2 demonstrate that the affect of an outlying project is ameliorated as the size of the data set increases.

6.5 Questions Revisited

In this section the questions asked in section 6.2 will be revisited in light of the tests carried out on the four data sets.

Does accuracy improve as the number of project cases increases?

The Hughes data set gives an almost perfect example of a curve of diminishing accuracy improvement over time. The Albrecht data set also appears to be characterised by a general increase in accuracy with size, which is only spoiled by the introduction of an outlying

project. Unfortunately, the other two data sets give no support this theory. When the individual series are combined into an average MMRE over time (Figure 6.5) the trend shown is for the Albrecht and Hughes data sets to improve, the Kemerer data set to remain remarkably constant throughout and the Telecoms1 data set to improve gradually in between large jumps of inaccuracy.

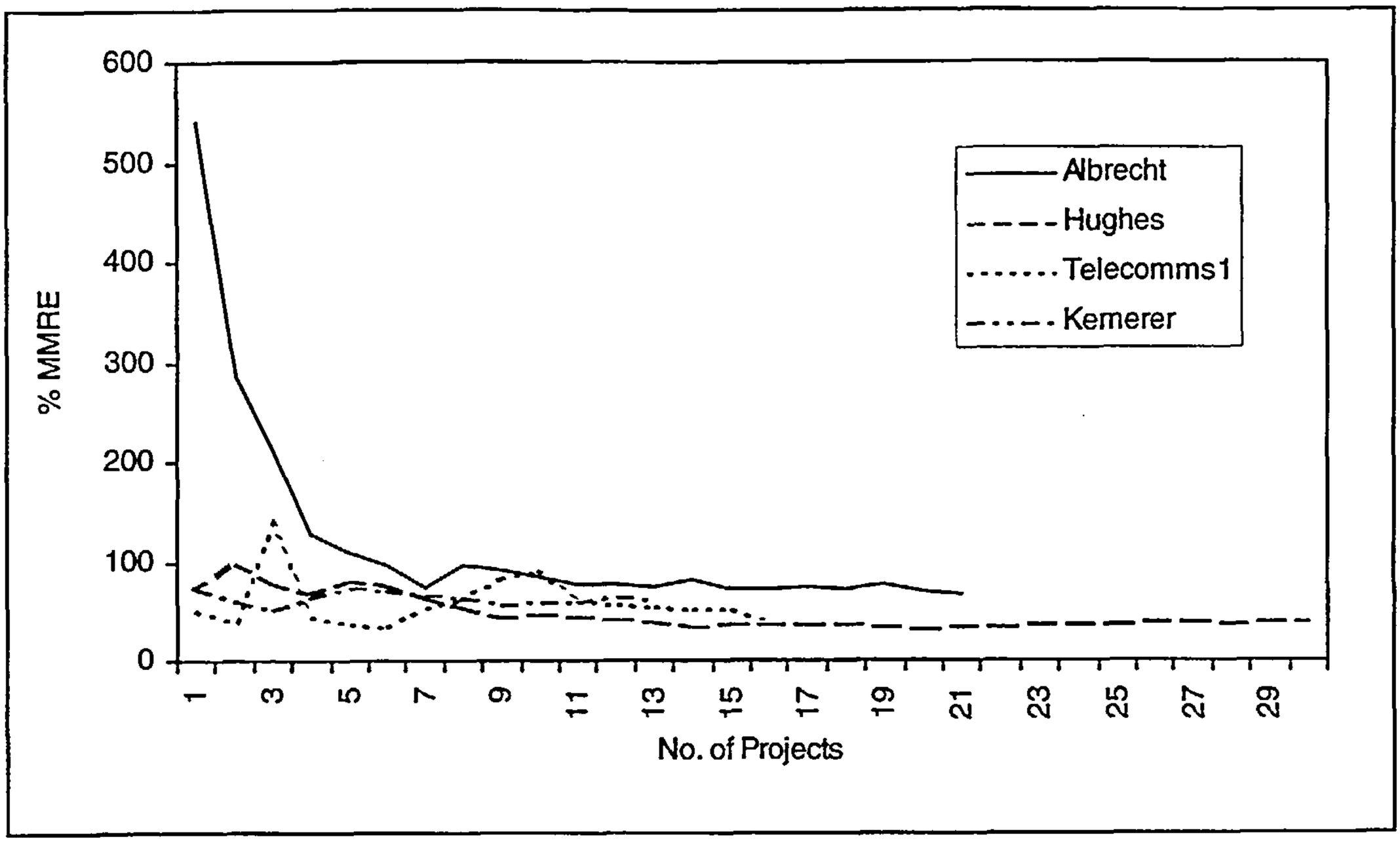


Figure 6.5 : Average Estimation Accuracy Over Time (All Data Sets)

What is the least number of project cases needed before estimation by analogy becomes stable?

The smallest data set analysed in chapter 5, MERMAID-N, contained 8 projects. However, the results of the sensitivity analysis highlight the fact that results from any data set containing less than 10 projects are likely to be very volatile. Thus caution must be exercised when interpreting the results from data sets containing less than this amount of projects.

Is estimation by analogy vulnerable to the addition of outlying data points?

The tests on the Albrecht data set highlight this point clearly and indeed the removal of the offending project, number 23, has a dramatic effect on the accuracy of estimation by analogy, reducing the original MMRE of 62% down to 39% and increasing the Pred(25) figure from 40% up to 47%. One good point to be drawn from Figure 6.1 is that the effect of an outlying

data point seems to be ameliorated as the size of the case base grows. The Telecoms 1 data set also appears to provide strong evidence on the effects of outlying data points. In this case though, the addition of just one project, similar to the outlier, removes the problem. This gives support to the idea that the analogy approach is more adaptable to changes in the estimation environment.

Is there a recognisable point at which estimation by analogy becomes stable?

This has implications for the amount of confidence that can be placed on estimates. Initial observations suggest (e.g. H1-3, T2, T3, K2, K3, and A1-3) that it is possible to discern a point at which the MMRE level stabilises and is not subject to wild fluctuations. This suggests that it might be possible identify a point at which the use of estimation by analogy on a given set of data enters a 'stable mode' in which new estimates can be treated with less suspicion. The Hughes data set certainly points to this being a possibility with the MMRE level stabilising after 15 projects and then remaining remarkably constant over the introduction of the remaining 17 projects. Unfortunately, the other three data sets have too few data points for any compelling conclusions to be drawn on this question.

6.6 Summary

The use of sensitivity tests on four of the project data sets has enabled the dynamic study of estimation by analogy under the more realistic circumstances of data points being added over time. The tests have thrown up a number of interesting characteristics of the approach, such as the fact that accuracy does not always increase with the number of projects and that it can be affected greatly by the introduction of outlying projects. One of the data sets showed evidence of a heightened sensitivity to the introduction of a single outlying project that had a dramatic effect on accuracy throughout the remaining test. The introduction of outlying projects is potentially the most dangerous pitfall of this technique, where the selection of one project as a basis for estimation is common. On the positive side however, the same data set showed that the effect of a outlier is diminished as the number of projects increases. Another important finding is the confirmation of the fact that, due to wild fluctuations in accuracy, estimation by analogy should be considered unsuitable where less than approximately ten projects are available.

In summary, estimation by analogy is a stable and robust estimation approach given an appropriate (typically 10 plus projects) amount of data. The sensitivity test described above

has proved its value in revealing previously unseen characteristics of each data set when used by the analogy approach.

Chapter 7

Conclusions

7.0 Introduction

This chapter will bring together all of the research findings of the previous six chapters and analyse both the importance of the work and its limitations. First of all the research will be briefly summarised, before the objectives stated at the start of the thesis are revisited in order that an assessment can be made of the degree to which they have been achieved. The next section will then focus on the contribution this thesis makes to empirical software engineering discussing the important research findings that have been identified. Thereafter, limitations of the research will also be discussed. The final section will look at where work in this area might be targeted in the future.

7.1 Summary of Work Carried Out

A study of the literature on software effort estimation was made covering many of the main research themes that have shaped present day practises in effort estimation. From the study it was apparent that, by far the greatest amount of research effort has been expended on the development and validation of algorithmic models such as Boehm's COCOMO (Boehm 1981), Albrecht's Function Points (Albrecht 1979; Albrecht and Gaffney 1983) and Putnam's SLIM (Putnam 1978), and that, in the opinion of some (e.g. (Kitchenham 1996)), this focus has been to the detriment of other potential estimation techniques. More recently, alternative estimation techniques have been proposed and explored by software effort researchers such as neural networks, case-based reasoning systems and rule induction systems, however, very little evidence (mainly anecdotal) exists to the efficacy of any of these approaches.

Having identified a need for research in the area of non-algorithmic approaches to effort estimation, this research project has focused upon a novel approach to effort estimation (called estimation by analogy) that involves the direct reuse of solutions to closely matching previous projects as a basis for estimates of new projects.

Before any practical use of the approach could be made, it was necessary to build a software tool that could store projects allowing for varying numbers of their features and calculate the relative similarity of projects so that the closest projects (to a new project) in terms of Euclidean distance could be used in the generation of an estimate for that new project. The use of software project data from eight different software engineering environments enabled an empirical validation of the approach in comparison to the use of an algorithmic method - stepwise regression - which showed (table 5.15 & 5.16) that, in almost all cases under study (comprising over 250 projects), the new approach was superior to the algorithmic model in terms of the MMRE and Pred(25) performance indicators.

The analogy estimation technique was also examined using a novel pseudo time-series study that allowed the dynamic behaviour of the approach to be observed as data sets were grown over time. This allowed the approach to be assessed in a more realistic environment where projects are constantly being added to a data set.

7.2 Research Objectives Revisited

The three objectives stated at the start of this thesis will now be reiterated so that they can be assessed in light of the work presented in the previous chapters.

i) To investigate the viability of analogical reasoning for the purpose of estimating the required effort to complete software projects.

An objective of "investigating the viability" of a new approach is perhaps a little fuzzy and needs to be brought into clearer focus. However, at the outset of the project the approach was little more than an idea thus, for it to be realised, many (and at that stage some unknown) factors would have to be investigated before estimation by analogy could be recognised as a viable technique. These factors include: its theoretical basis, ease of automation, ease of use and widespread applicability.

Theoretical basis

Many of the ideas used in the development of the ANGEL approach to effort estimation, particularly the similarity measure, were drawn from mature well grounded theory that has been examined and applied within the case-based reasoning community for the last 15 years.

Ease of Automation

One of the first and most important questions was whether estimation by analogy could be automated. The importance of this is discussed below, but suffice to say that the time scales involved for the multiple proximity calculations make it impractical for anything other than a computer. Although time consuming, the algorithms that drive estimation by analogy are repetitive in nature and thus were easy to implement.

Ease of use

The process of estimation by analogy can draw many parallels with the way humans informally use analogies from the past to solve problems in the present. Thus the concepts behind estimation by analogy are easy to understand. The major difficulty with this or any estimation approach comes from the peripheral problems, such as data collection. However, data collection is facilitated by the flexibility of ANGEL's templates which do not dictate the mandatory collection of any single feature and which include the ability to use categorical features, which are relatively easy to collect.

Widespread applicability

By permitting the user to process categorical values, the estimation by analogy approach becomes available earlier in the software project life-cycle than algorithmic techniques. This is important to companies that are forced to make early estimates based upon sketchy requirements with a lack of quantitative data. Again, because ANGEL does not predetermine what features must be collected, unlike for example, COCOMO, the approach is less restrictive.

ii) To develop an automated tool that supports the functionality required to generate estimates by analogical reasoning.

In all walks of life a great number of new ideas get stifled before they are fully realised because there are no practical means to implement them. This also holds true for the software engineering community and therefore to fully investigate the potential of the analogy estimation technique it was important to develop a tool that could automate the complex proximity calculations needed to make analogy estimation effective. On top of this, the needs of the software manager, the most likely user of such a tool, are equally important in the acceptance of a new technique. For the majority of managers making software estimates involves spending valuable time away from the practical implementation issues of a project. Therefore to have any chance of acceptance the analogy approach must have a short learning curve and be able to produce rapid results.

This has been achieved in the development of the ANGEL software tool. ANGEL allows a user not only to define project environments and store project data, but also to make predictions for new projects based upon that data. The tool is not constrained to the collection of any project characteristic or data type, which makes it practical within widely different development environments. This also provides the possibility that it can be used for predicting other characteristics of software (such as project duration or fault density) or indeed potentially, within many other areas where estimation is a problem. The tool was developed with a conventional 'windows' style interface to make it easy to adopt and use.

The latest version of ANGEL (2.0). has been demonstrated informally a number of times and its simplicity and ease of use have been widely commended. It is also available in a scaled-down version on the internet and is being used in a number of software companies such as British Telecom. The concepts and algorithms behind ANGEL have also been incorporated into a software quality toolset known as SQUID (Kitchenham, Linkman et al. 1997), which employs analogical reasoning to evaluate the feasibility of and predict values for software quality requirements.

iii) To validate the analogical reasoning technique on data taken from industrial environments.

Validation of a new technique is essential before any legitimate claims can be made about its ability to estimate effort. Validation can take many forms, such as its ability to resist outliers or its ability to generalise when presented with new situations, but this project concentrated first, and foremost, on the accuracy of the approach. A widely recognised algorithmic approach - stepwise regression – was used as the benchmark against which it was judged.

The accuracy experiments were carried out on eight data sets comprising 254 projects, all from the software industry. The relative accuracy of each approach was measured in terms of MMRE and Pred(25) and the outcome was that in all but one case, the analogy approach matched or outperformed the stepwise regression approach. As a result, the alternative hypothesis, that estimation by analogy is a more accurate estimator of software project effort than stepwise regression, was accepted, using wilcoxon signed pairs at $p = 0.001$ for the MMRE performance indicator and $p = 0.0054$ for the Pred(25) indicator.

Many other aspects of the analogy approach were also studied in an analysis of the approach's sensitivity to the addition of new data points. This study provided valuable insights into the dynamic behaviour of estimation by analogy such as its ability to resist outliers, the number of data points needed before it becomes a viable technique and the effect of data set size on the accuracy of the approach. The study showed that the analogy approach

is sensitive to outlying projects, but that some projects that might be classed as outliers for a regression model may not be regarded as such by the analogy approach and visa versa. In terms of the number of data points needed before analogy can be used sensibly used, 8 was the lowest number of projects presented to ANGEL which returned an MMRE of 53% and a Pred(25) of 39%. However, the evidence from the sensitivity analysis indicates that the use of any less than 10 projects is risky. The extent to which data set size affects accuracy remains hazy, although the sensitivity tests did give some indication that accuracy becomes more stable as the data set grows.

7.3 Synopsis of Research Findings

The major research findings of this project have been that software effort estimation by analogy can be used as a viable alternative or complement to present estimation practices and that, at least for the eight data set studied, the approach is superior in accuracy to regression based models. Other important research findings include:

- *The observed accuracy of a prediction system is very dependent on the performance indicator used.* While the analogy approach was judged to be greatly superior to the regression approach in terms of MMRE, the Pred(25) results, while still supporting the superiority of analogy, were less convincing. This confirms the findings of a study by Schofield (1997) who used three different estimation techniques to estimate effort for the same data set and measured the results using four different performance indicators. He found that each of the three techniques were reported as most accurate with at least one of the indicators.

- *The application of weightings to increase the effect of chosen features within the proximity calculations does not appear to have any significant effect on results.* One explanation of this could be that the use of the best attribute subset function allows ANGEL to find those features that have the most influence on effort and therefore apply a natural bias or weighting to them. For example, if a data set contained a number of complexity related measures amongst other features, ANGEL might decide that complexity has an important influence on effort and focus upon those features discarding other less important features.

- *Searching for the best attribute subset has a significant impact on the accuracy of the analogy approach.* For each of the data sets under study, with the exception of Telecoms1, accuracy in terms of MMRE and Pred(25), can be dramatically improved by looking for the subset of variables that best predicts effort. This is an important finding in that it not only allows

us to calibrate the analogy model, but it also means that expert judgement is not necessarily required to pare down the feature list.

- *Analogy can succeed even where no statistical relationships are present.* Recall that the MERMAID-N data set displayed no statistical relationships between the independent features and effort. This however, was not a problem for analogy as it does not look for statistical relationships and as a result was able to predict effect, in less than propitious circumstances, with an MMRE of 60% and Pred(25) of 25%.

- *Analogy can be used throughout the project life-cycle.* The value of a effort estimate is inversely proportionate to point, in the project lifecycle, that it is generated. Unfortunately, the same holds true for the difficulty in developing an estimate. This is a major problem for software estimators because the lack of collectable quantitative data before the requirement specification stage means that traditional algorithmic estimates must be made using estimated input parameters. The analogy approach on the other hand is able to generate estimates by using qualitative (or categorical) data and is therefore available throughout the project life-cycle.

- *Analogy, like algorithmic models, is sensitive to the introduction of outliers.* An analysis of the sensitivity of the analogy approach has revealed that, in common with algorithmic models, it is sensitive to the introduction of outliers. Similarly with algorithmic models, the more data points available, the less effect the outlier will have. Conversely, unlike algorithmic models, assuming only one analogy is being searched for, the effect of an outlier can be nullified with the addition of just one similar data point. Where more than one analogy is being sought, obviously it takes a like amount of similar projects to totally nullify the outlier.

- *Accuracy using analogy improves with data set homogeneity.* Tests on both the Desharnais and the MERMAID data set confirm the assumption that creating more homogeneous data sets can increase accuracy. However, the observed accuracy increases were relatively minor and inferior to those observed for the stepwise regression models.

- *Analogy should be considered as complementary rather than alternative approach.* It was never the intention of this work to find a new estimation technique that would replace existing techniques. Instead it is believed that benefit can be gained by the application of more than one estimation technique in a complementary fashion. In this way estimates can be triangulated, with conflicting estimates pointing to a possible risk and the need for more data, and converging estimates providing a level of confidence.

7.4 Contribution of this Thesis

This work has made the following contributions to the field of software effort estimation:

i) A new approach to effort estimation to the extent that it is ready to be deployed within a software organisation.

ii) A better understanding of the relative accuracy of the estimation by analogy technique in relation to an algorithmic model developed by linear regression.

iii) A tool that facilitates estimation by analogy. The tool is freely available on the internet and is being used by researchers examining software effort data sets (e.g. (Niessink and Van Vliet 1997; Stensrud and Myrtveit 1998)).

iv) A new way of studying the dynamic behaviour of project data sets by artificially introducing data points one at a time and estimating each project from each partial data set.

7.5 Limitations of Work

Some elements of the work presented in this thesis were theoretical in nature and many of the techniques have not been previously applied to software project data. As a result it is important to recognise the limitations of the work reported. The identified limitations are divided between the ANGEL approach, the software tool and the analysis procedure.

7.5.1 Limitations of Approach

- 3 of the 8 data sets analysed in chapter 5 contained features measured on the ordinal scale (such as *Project Manager's Experience* – Appendix A2). The ANGEL approach permits the user to define such variables as being on the interval scale which is clearly in breach of measurement theory. In defence of this approach however, it does improve the accuracy of estimation by analogy (in some cases considerably) for the data sets that contain such features. This less-rigorous approach is defended by Briand et al. (1996) and Stevens (1946), who advocate a more pragmatic to the application of measurement theory.

- MMRE is possibly the most widely used and reported performance indicator and for that reason was chosen as the main measure of accuracy in the ANGEL approach. Unfortunately, the measure is flawed being non-symmetrical and tending to favour systems that underestimate. Due to the fact that the ANGEL approach optimises on MMRE it therefore follows that predictions are more likely to be under estimates.

- There is a need for a strategy to handle the situation where ties are encountered. It is inevitable, especially when there is a significant use of categorical features, that there will occasionally be a situation when two or more projects are identical in terms of their similarity to the target project. This is not a problem where the number of ties is equal too, or less than, the number of analogies sought[36]. If the opposite is true however, then the procedure at present is that the first tied analogies encountered will be used as the sources analogies. This is clearly unsatisfactory, as it is quite possible that the selection of those analogies that have been overlooked would give conflicting figures. In truth, the situation where a number of ties are found, probably points to the fact that the combination of features being used is unsuitable for finding distinct analogies.

7.5.2 Tool Limitations

- The function in ANGEL to search for the best subset of features has proved to be a very effective mechanism for improving estimate accuracy. Unfortunately, the need to evaluate every combination of features against each other, to facilitate this improvement in accuracy, means that there is a limit to the number of features that can realistically be processed[37].

- Although the ANGEL tool is being used or evaluated by a number of major software developers, it must be remembered that it is only a prototype system that was developed with purely research objectives in mind. As a consequence, the tool has not been systematically tested. Following on from this point, the tool was also not developed with the principles of sound ergonomic design high on the priority list.

---

[36] Assuming that no weightings are being placed on the analogies. If two analogies(weighted) are sought then the first tie would still be weighted double.

[37] Every single increment in the number of features used effectively doubles the search time. Thus if 20 feature takes two days to process, 24 feature will take approximately a month.

7.5.3 Analysis Limitations

- The level of accuracy that ANGEL was consistently able to maintain over the eight data sets was encouraging, especially in light of the relative performance of the stepwise regression. However, it must be understood that some of the data sets analysed were quite old and their relevance to today's software development environments must be considered. For example, the Albrecht data set was first published 17 years ago and the Kemerer data set 10 years ago. Moreover, the fact that ANGEL is capable of reliably predicting effort over a range of data sets spanning a number of years (in which time many practices have changed), suggests that it may not be unreasonable to conclude that the analogy approach is widely applicable.

- Another limitation of the analysis procedure was that two separate techniques, jack-knifing and goodness of fit were used to generate the performance figures for analogy and stepwise regression respectively. This was unfortunate and in hindsight it would have been better to adopt the same technique for both approaches. This should have been the jack-knifing procedure for two reasons i) the project to be estimated in ANGEL cannot, for obvious reasons (the main one being that it would always to be found as the closest analogy) be used in the set of source analogies and ii) using goodness of fit means that each project contributes[38] to the model that will be eventually used to evaluate it. In terms of the impact this has on accuracy, as discussed in section 5.2, the use of the goodness of fit gives the stepwise regression technique an advantage over analogy, however, this advantage is thought to be balanced by the advantage gained by analogy from optimising with MMRE.

7.6 Further Work

Suggestions for future work are now made which are divided between three areas: work that would enhance the functionality and effectiveness of the ANGEL tool; work to extend the analogy approach and finally possible future avenues for wider research into effort estimation.

---

[38] A luxury new projects estimated by the model would not be afforded.

7.6.1 Improvements to the ANGEL Tool

There are specific ways in which the ANGEL tool might be improved. An important addition would be a facility to deal with the situation, discussed above, where there are ties between source analogies. Another useful addition to ANGEL would be an ability to conduct the search for the best attribute subset using heuristics to reduce the number of combinations searched for, without overly affecting the potential accuracy.

7.6.2 Research on the Analogy Approach

A possible future research avenue would be to investigate further into the sensitivity of the analogy approach. The tests in chapter 5 looked at overall accuracy, in terms of MMRE, over time by adding projects in random order. Another possible test might look at how well ANGEL was able to predict the next project to be added to the case base revealing its true accuracy more realistically. While a further test might be to track the features that are being selected by the 'best subset' function which would give some indication of the most useful features to be collected in the future. All of these investigations could be automated within the ANGEL environment without much effort.

Within the case-based reasoning community one of the hottest research topics (Leake 1996) is the application of adaptation to estimates, based upon the differences between the observed feature values in the source and target analogies. Although ANGEL does adjust its estimates based upon the number of analogies searched for, it does not at present have any real adaptation abilities. For an adaptation system to be useful in ANGEL, it is important either that it is automatable or is a simple process that a non-domain expert might follow. But perhaps most important of all is that it not be hard coded so that ANGEL remains a shell transferable between environments.

7.6.3 Future Research Avenues

Researchers into the development of effort prediction systems have recently recognised the fact that there are a great many alternatives to algorithmic models and that these techniques such as neural networks and case-based reasoning offer a number of potential advantages (not least improvement in accuracy) over their algorithmic counterparts. However for the potential of any of these techniques to be realised it is important that they are brought closer to the estimation practitioner. For example, in the case of neural networks there is a certain level of competence required before they can deployed effectively by a practitioner. Until tools and

methods are developed that make the use of such technologies easier, without the loss of any of their estimating power, any amount of research proving their accuracy will not encourage practitioners to embrace them.

Repeatable prediction systems such as regression and analogy are at present dependent on the availability of a certain amount of historical data with which to build models or search for pertinent analogies. However, the indications are that the collection of completed project data is not so wide spread amongst software companies (Heemstra 1992). Further evidence of this is provided by the lack of data sets that are seen in the public domain, even after taking the confidentiality of such data into consideration. This points to the need for researchers to look for ways of developing data-less prediction systems. One possible way that this might be achieved using ANGEL would be for an expert to seed a case base with artificial cases that cover a range of potential outcomes. Another approach would be to use a technique such as the analytic hierarchy process, discussed by Saaty (1994), which can use a single known case to help reconstruct unrecorded historical cases.

Strategies for assessing the accuracy of predictions systems are varied. They range from the optimistic goodness of fit and jack-knifing techniques, where most or all of the data points are used to create the model on which they are then tested, to the more pessimistic random sampling of data into training, validation and testing sets on the other end of the scale. The latter technique being preferable as long as there is enough data available to sufficiently populate each of the three set with a representative sample. The decision to use a particular strategy is not always easy, especially with software project data sets, where the number of projects collected is typically less than 50. Another problem widely ignored in the validation literature is that of choosing performance indicators. Typically researchers (the author included) have given a primary reason for choosing a performance indicator as 'it is the most popular in the literature'. The indicator will clearly have a track record and it allows benchmarking, however, each performance indicator tells a different story and more thought must be given to the goal of the validation exercise. As studies into the accuracy of effort prediction systems are becoming more common place, the need for guidelines to help refine validation strategies becomes more essential.

References

Aarmodt, A. and E. Plaza (1994). "Case-Based Reasoning: Foundational Issues, Methodical Variations and System Approaches." AI communications **7**(1):

Aha, W. D. (1991). "Case-Based Learning Algorithms." 1991 DARPA Case-Based Reasoning Workshop, Morgan Kaufmann.

Albrecht, A. J. (1979). "Measuring Application Development Productivity." Proceedings of the IBM Applications Developments Symposium,

Albrecht, A. J. and J. E. Gaffney (1983). "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation." IEEE Transactions on Software Engineering **9**(6): 639-648.

Althoff, K. D. (1996). " Evaluating case-based reasoning systems." Workshop on Case-Based Reasoning: A New Force In Advanced Systems Development,

Atkinson, K. and M. Shepperd (1994). "Using Function Points to Find Cost Analogies." ESCOM 95, Ivrea, Italy,

Basili, R. and J. Beane (1981). "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problem." Journal of System and Software **2**: 59 - 69.

Behrens, C. A. (1983). "Measuring the Productivity of Computer Systems Development Activities With Function Points." IEEE Transactions on Software Engineering **9**(6): 649 - 658.

Bennington, H. D. (1983). "Production of Large Computer Programs." Annals of the History of Computing **5**(4): 350 - 361.

Bisio, R. and F. Malabocchia (1995). "Cost Estimation of Software Projects Through Case Based Reasoning." International Conference on Case Based Reasoning, Sesimbra, Portugal,

Boehm, B., B. Clark, et al. (1995). "The COCOMO 2.0 Software Cost Estimation Model." International Society of Parametric Analysts - 17th Annual Conference.,

Boehm, B. and W. Royce (1989). "Ada COCOMO and the Ada Process Model." Fifth COCOMO Users' Group Meeting, Pittsburgh,

Boehm, B. W. (1981). Software Engineering Economics. New York, Prentice-Hall.

Boehm, B. W. (1997). "COCOMO II Experience and Plans." ESCOM97, Berlin,

Briand, L., K. Emam, et al. (1996). "On The Application of Measurement Theory in Software Engineering." Empirical Software Engineering **1**(1): 61 - 88.

Bridgett, N. A., J. Brandt, et al. (1995). "A Neuro-fuzzy Route to Breast Cancer Diagnosis and Treatment." FUZZ-IEEE/IFES'95, Yokahama, Japan,

Brieman, L., J. H. Friedman, et al. (1984), Classification and Regression Trees. Wadsworth International Group. California.

Brule, J. F. (1985). Fuzzy Systems - A Tutorial. http://www.quadralay.com/www/Fuzzy/Tutorial.html.

Campobasso, A., G. Caracoglia, et al. (1995). "Using Cost Models in Software Industry." ESCOM, Netherlands,

Conte, S., H. E. Dunsmore, et al. (1986). Software Engineering Metrics and Models. Benjamin/Cummings.

Cowderoy, A. J. C. and J. O. Jenkins (1988). "Cost Estimation by Analogy as a Good Management Practice." Proc. Software Engineering 88. Liverpool: IEE/BCS,

Davies, R., B. G. Buchanan, et al. (1997). "Production Rules as a Representation for a Knowledge-Based Consultation Program." Artificial Intelligence **8**: 15 - 45.

DeMarco, T. (1982). Controlling Software Projects. Yourdon Press.

Desharnais, J. M. (1988). Analyse Statistique de la Productivite des Projects de Development en Informatique a Partir de la Technique de Points de Fonction. MSc. Thesis, Montreal, University of Quebec.

Evans, T. G. (1968). A Program for the Geometric Analogy Intelligence Test Questions. Semantic Information Processing. Cambridge, Mass, MIT Press.

Farr, L. and J. Zargorski (1965). "Quantitive Analysis of Programming Cost Factors: A Progress Report." ICC Symposium Proc. Economics of Automatic Data Processing, Amsterdam,

Finkelstein, L. and M. S. Leaning (1984). "A Review of the Fundamental Concepts of Measurement Theory." Measurement **2**(1): 25 - 34.

Gentner, D. (1983). "Structure Mapping - A Theoretical Framework for Analogy." Cognitive Science **7**:

Golden, J. R., J. R. Mueller, et al. (1981). "Software Cost Estimating: Craft or Witchcraft." Database **12**: 12 - 14.

Gray, A. R. and S. G. MacDonell (1997). "A Comparison of Techniques for Developing Predictive Models of Software Metrics." Information and Software Technology **39**: 425 - 437.

Gulezian, R. (1991). "Reformulating and Calibrating COCOMO." Journal of Systems and Software **16**: 235 - 242.

Heemstra, F. J. (1992). "Software Cost Estimation." Information and Software Technology **34**(10): 627 - 639.

Hughes, R. T. (1996). An Evaluation of Machine Learning Techniques for Software Effort Estimation. University of Brighton.

IFPUG (1994). Function Point Counting Practices Manual: Release 4.0. Westerville OH, International Function Point User's Group.

Jeffery, R. (1987). "Time-Sensitive Cost Models in the Commercial MIS Environment." IEEE Transactions on Software Engineering **13**(7): 852-859.

Jeffery, R. (1991). Software Cost Estimation Models. Software Engineer's Reference Book. Oxford, Butterworth-Heinemann. 28/1 - 28 /10.

Jeffery, R. and J. Stathis (1993). "Specification Based Software Sizing : an Empirical Investigation of Function Metrics." NASA Goddard Software Engineering Workshop, Greenbelt, MD, USA,

Johnson, D. M. (1962). "Serial Analysis of Verbal Analogy Problems." Journal of Education Psychology **53**: 86 - 88.

Jones, C. (1986). Programming Productivity. New York, McGraw-Hill.

Jorgensen, M. (1995). "Experience With the Accuracy of Software Maintenance Task Effort Prediction Models." IEEE Transactions on Software Engineering **21**(8): 674-681.

Karunanithi, N., D. Whitley, et al. (1992). "Using Neural Networks in Reliability Prediction." IEEE Software **9**(4): 53 - 59.

Kemerer, C. F. (1987). "An Empirical Validation of Cost Estimation Models." Comms of the ACM **30**(5): 416-429.

Kemerer, C. F. and B. S. Porter (1992). "Improving the Reliability of Function Point Measurement: an Empirical Study." IEEE Transactions on Software Engineering **18**(11): 1011 - 1024.

Kitchenham, B. (1990). Software Development Cost Models. Software Reliability Handbook. Ed. Rook, P. Elsevier.

Kitchenham, B. (1996). "Estimation - A Personal View." ESCOM97, Wilmslow, UK,

Kitchenham, B. and K. Kansala (1993). "Inter-Item Correlations Among Function Points." 15th Intl Conf On Software Engineering, Baltimore, IEEE Computer Society Press.

Kitchenham, B. and S. Linkman (1997). "Estimates, Uncertainty and Risk." IEEE Software **14**(3):

Kitchenham, B., S. Linkman, et al. (1997). "The SQUID Approach to Defining a Quality Model." Software Quality Journal **6**: 211 - 233.

Kitchenham, B. A. (1992). "Empirical Studies of Assumptions That Underlie Software Cost-Estimation Models." Information and Software Technology **34**(4): 211-218.

Kitchenham, B. A. and N. R. Taylor (1984). "Software Cost Models." ICL Technical Journal (May): 73 - 102.

Kitchenham, B. A. and N. R. Taylor (1985). "Software Project Development Cost Estimation." The Journal of System Software **5**: 267-278.

Knaff, F. J. and J. Sacks (1986). "Software Development Effort Prediction Based on Function Points." COMPSAC 86,

Kok, P., B. A. Kitchenham, et al. (1990). "The MERMAID Approach to Software Cost Estimation." Esprit Annual Conference, Brussels,

Kolodner, J. (1983). "Maintaining Organisation in a Dynamic Long Term Memory." Cognitive Science **7**: 234 - 280.

Kolodner, J. L. (1993). Case-Based Reasoning. Morgan-Kaufmann.

Leake, D. (1996). Case-Based Reasoning: Experiences Lessons and Future Directions. Menlo Park, AAAI Press.

Lederer, A. L. and J. Prasad (1993). "Information Systems Software Cost Estimating: A Current Assessment." Journal of Information Technology **8**: 22 - 33.

Lientz, B. P. and E. B. Swanson (1980). Software Maintenance Management. Reading, Addison-Wesley.

Low, G. C. and R. Jeffery (1990). "Function Points in the Estimation and Evaluation of the Software Process." IEEE Transactions on Software Engineering **16**(1): 64-71.

MacDonell, S. G. and A. R. Gray (1996). "Alternatives to Regression Models For Estimating Software Projects." IFPUG Fall Conference, Dallas,

Miyazaki, Y. (1993). "Robust Regression for Developing Software Estimation Models." ESCOM 93, Bristol,

Miyazaki, Y. and K. Mori (1985). "COCOMO Evaluation and Tailoring." Proceedings of 8th International Software Engineering Conference, IEEE Computer Society Press.

Miyazaki, Y., M. Terakado, et al. (1994). "Robust Regression For Developing Software Estimation Models." J. Systems Software 27: 3 - 16.

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

Mukhopadhyay, T. and S. Kekre (1992). "Software Effort Models for Early Estimation of Process Control Applications." IEEE Transactions on Software Engineering 18(10): 915 - 923.

Narendra, K. S. and K. Parthasarathy (1987). "Identification and Control of Dynamical Systems Using Neural Networks." IEEE Transactions on Neural Networks 1: 4-27.

Nelson, E. A. (1967). Management Handbook for Estimation of Computer Programming Costs. Systems Development Corp.

Niessink, F. and H. Van Vliet (1997). "Predicting Maintenance Effort with Function Points." International Conference on Software Maintenance, Bari, Italy,

Norden, P. V. (1963). Useful Tools for Project Management. New York, John Wiley and Sons.

Oppenheimer, J. R. (1956). "Analogy in Science." American Psychological Review 11: 127-135.

Parr, F. N. (1980). "An Alternative to the Rayleigh Curve Model for Software Development Effort." IEEE Transactions on Software Engineering 6(3): 291 - 296.

Porter, A. and R. Selby (1990). "Empirically Guided Software Development Using Metric-Based Classification Trees." IEEE Software 7: 46 - 54.

Prietula, M. J., S. S. Vincinanza, et al. (1996). "Software Effort Estimation With a Case-Based Reasoner." J. Experimental & Theoretical Artificial Intelligence 8: 341 - 363.

Putnam, L. H. (1978). "A General Empirical Solution to the Macro Sizing and Estimating Problem." IEEE Transactions on Software Engineering SE-4(4): 345 - 361.

Raven, J. C. (1938). Progressive Matrices: A Perceptual Test of Intelligence. London: Lewis.

Raven, J. C., J.H. Court, et al. (1986). Coloured Progressive Matrices. London: Lewis.

Rich, E. and K. Knight (1995). Artificial Intelligence. McGraw-Hill.

Rubin, H. A. (1983). "Macroestimation of Software Development Parameters: The Estimacs System." SOFTAIR Conference on Software Development Tools, Techniques and Alternatives, Arlington, IEEE Press, New York.

Rudolph, E. E. (1983). Productivity in Computer Application Development. Dept. Management Studies, Dept. Management Studies, University Auckland.

Saaty, T. L. (1994). "Highlights and Critical Points in the Theory and Application of the Analytic Hierarchy Process." European Journal of Operational Research 74: 426 - 447.

Samson, B., D. Ellison, et al. (1993). "Software Cost Estimation using an Albus Perceptron(CMAC)." Proc. Eight International COCOMO Estimation Meeting, Pittsburgh,

Schank, R. (1982). Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press.

Schofield, C. (1997). "Selecting Criteria for the Evaluation of Effort Prediction Systems." ESCOM 97, Berlin,

Sejnowski, T. J. and C. R. Rosenberg (1987). "Parallel Networks That Learn to Pronounce English Text." Complex Systems **1**: 145-168.

Serluca, C. (1995). An Investigation into Software Effort Estimation Using a Back-Propogation Neural Network. M.Sc. Thesis, Bournemouth University.

Shepperd, M. (1994). "Some Observations on Function Points." CSR Annual Conference, Dublin,

Srinivasan, K. and D. Fisher (1995). "Machine Learning Approaches to Estimating Software Development Effort." IEEE Transactions on Software Engineering **21**(2): 126-136.

Stensrud, E. and I. Myrtveit (1998). "The Added Value of Estimation by Analogy - An Industrial Experiment." The European Software Measurement Conference, Antwerp, Belgium,

Stevens, S. (1946). "On the Theory of Scales of Measurement." Science **103**(2684): 677 - 680.

Subramanian, G. H. and S. Breslawski (1995). "An Empirical Analysis of Software Development Cost Estimation Model Awareness and Usage." Information Resources Management Association: Managing information and communications in a changing global environment, Atlanta; GA, Harrisburg.

Suppes, P., D. H. Krantz, et al. (1989). Foundations of Measurement. London, Academic Press.

Symons, C. R. (1991). Software Sizing and Estimating. MK II FPA. Chichester, John Wiley.

Symons, R. (1988). "Function Point Analysis: Difficulties and Improvements." IEEE Transactions on Software Engineering **14**(1): 2-11.

Veloso, M. M. and J. G. Carbonell (1991). "Variable-Precision Case Retrieval in Analogical Problem Solving." Proceedings of the DARPA Case-Based Reasoning Workshop,

Venkatachalam, A. R. (1993). "Software Cost Estimation Using Artificial Neural Networks." International Joint Conference on Neural Networks, Nagoya, IEEE.

Vicinanza, S. and M. J. Prietolla (1990). "Case Based Reasoning In Software Effort Estimation." Proceedings 11th Int Conf on Information Systems,

Vosniadou, S. and A. Ortony, Ed. (1989). Similarity and Analogical Reasoning. Cambridge, Cambridge University Press.

Walston, C. E. and C. P. Felix (1977). "A Method of Programming Measurement and Estimation." IBM Systems Journal **16**(1): 54-73.

Watson, I. and F. Marir (1994). "Case-Based Reasoning: A Review." The Knowledge Engineering Review **9**(4): 327-354.

Wiener-Ehrlich, W. K., J. R. Hamrick, et al. (1984). "Modeling Software Behaviour in Terms of Formal Lifecycle Curve: Implications for Software Maintenance." IEEE Transactions on Software Engineering **SE-10**: 376 - 282.

Winston, P. H. (1970). Learning Structural Description From Examples. Cambridge, Mass, MIT.

Wittig, G. and G. Finnie (1997). "Estimating Software Development Effort with Connectionist Models." Information and Software Technology **39**: 469 - 476.

Wolverton, R. W. (1974). "The Cost of Developing Large Scale Software." IEEE Transactions on Computers **23**(6):

Zadeh, L. A. (1965). "Fuzzy Sets." Info.& Ctl **8**: 338 - 353.

Zadeh, L. A. (1988). "Fuzzy Logic." IEEE Computer **21**(4): 83 - 93.

Appendix A

Key

✔ - Used in analysis (chapter 5)

✘ - Not used in analysis (chapter 5)

A1. Albrecht Data set

Feature	Description	Analogy	Regression
Effort	Measured in thousands of work hours	✔	✔
FP	Function points count	✔	✔
Files	Number of master files	✔	✔
Inputs	Number of Inputs	✔	✔
Inquires	Number of Inquiries	✔	✔
Outputs	Number of outputs	✔	✔

A2 The Desharnais Data set

Feature	Description	Analogy	Regression
Effort	Measured in hours	✔	✔
ExpEquip	Team experience in years	✔	
ExpProjMan	Project managers experience in years	✔	✘
Trans	Number of transactions	✔	✔
Entities	Number of entities	✔	✔
RawFP	Unadjusted function points	✔	✔
AdjFP	Adjusted function points	✔	✔
DevEnv	Development Environment	✔	✘
YearFin	Year of Completion	✔	✔

A3 The Finnish Data set

Feature	Description	Analogy	Regression
Effort	Measured in hours	✔	✔
FP	Function points count	✔	✔
UA	End user availability	✔	✘
MA	Machine availability	✔	✘
AA	Analyst/designer availability	✔	✘
STD	availability of standards	✔	✘
UM	Use of methods	✔	✘
TA	Tool availability	✔	✘
PC	Procedural complexity	✔	✘
SR	Stability of requirements specification	✔	✘
CQ	Criticality of quality requirements	✔	✘
CP	Criticality of execution time requirements	✔	✘
UT	User training	✔	✘
MET	Methodology	✔	✘
TME	Team application experience	✔	✘
MEX	Team method/tools experience	✔	✘
PME	Project manager experience	✔	✘
IN	Number of inputs	✔	✔
INFP	Input function points	✔	✘
QN	Number of queries	✔	✔
QFP	Query function points	✔	✘
ON	Number of Outputs	✔	✔
OFP	Output function points	✔	✘
SN	Number of interacting systems	✔	✔
SFP	Interacting systems function points	✔	✘
FN	Number of logical master files	✔	✔
FFP	Logical master file function points	✔	✘
HW – Type	Hardware type	✔	✘
AT – Type	Application type	✔	✘

A4 The Hughes Data set

Feature	Description	Analogy	Regression
Effort	Measured in work-hours	✔	✔
C2	Number of parameters in operator commands	✔	✔
C3	Number of parameters used by subscriber input procedures	✔	✔
C4	Outputs which trigger messages	✔	✔
C5	Number of parameters in output/enquiry subscriber procedures	✔	✔
C6	Number of parameters on print-outs	✔	✔
C7	Number of messages passed between blocks	✔	✔
C8	Count of timers used by function	✔	✔
C9	Months of experience of the block designer	✔	✔
C10	Months of experience of the function designer	✔	✔
C11	C3+C5	✔	✔
C12	C9+C10	✔	✔
C13	Number of lines of code patched in the base version of the code that the enhancement has incorporated	✔	✔
C14	Subsystem indicator 0 or 1	✔	✘

A5 The Kemerer Data set

Feature	Description	Analogy	Regression
Effort	Measured in person-months	✔	✔
AdjFP	Adjusted function points	✔	✔
RawFP	Unadjusted function points	✔	✔

A6 The MERMAID Data set

Feature	Description	Analogy	Regression
Effort	Measured in hours	✔	✔
AdjFP	Adjusted function point count	✔	✔
RawFP	Unadjusted function point count	✔	✔
Proj Type	Project Type - New/Enhancement	✔	✘
AF1	Data communications	✔	✘
AF2	Distributed functions	✔	✘
AF3	Performance	✔	✘
AF4	Heavily used configuration	✔	✘
AF5	Transaction rate	✔	✘
AF6	Online data entry	✔	✘
AF7	End user efficiency	✔	✘
AF8	Online update	✔	✘
AF9	Complex processing	✔	✘
AF10	Reusability	✔	✘
AF11	Installation ease	✔	✘
AF12	Operational ease	✔	✘
AF13	Multiple sites	✔	✘
AF14	Facilitates change	✔	✘

A7 The Real-Time1 Data set

Feature	Description	Analogy	Regression
Effort	Measured in person months	✔	✘
Host Machine	Host machine used	✔	✘
Life Cycle	Life cycle used	✔	✘
Documentation Standard	Documentation Standard used	✔	✘

A8 The Telecomms1 Data set.

Feature	Description	Analogy	Regression
Effort	Measured in Person Days	✔	✔
Files	Number of files amended	✔	✔

Appendix B

The Albrecht Data Set

Project Ref	Effort	Inputs	Outputs	Files	Inquiries	FP	SLOC
1	102.4	25	150	60	75	1750	130
2	105.2	193	98	36	70	1902	318
3	11.1	70	27	12	0	428	20
4	21.1	40	60	12	20	759	54
5	28.8	10	69	9	1	431	62
6	10	13	19	23	0	283	28
7	8	34	14	5	0	100	35
8	4.9	17	17	5	15	289	30
9	12.9	45	64	16	14	680	48
10	19	40	60	15	20	794	93
11	10.8	41	27	5	29	512	57
12	2.9	33	17	5	8	224	22
13	7.5	28	41	11	16	417	24
14	12	43	40	35	20	682	42
15	4.1	7	12	8	13	209	40
16	15.8	28	38	9	24	512	96
17	18.3	42	57	5	12	606	40
18	8.9	27	20	6	24	400	52
19	38.1	48	66	50	13	1235	94
20	61.2	69	112	39	21	1572	110
21	3.6	25	28	22	4	500	15
22	11.8	61	68	11	0	694	24
23	0.5	15	15	3	6	199	3
24	6.1	12	15	15	0	260	29

The Desharnais Data set

Project Ref	Effort	Exp Equip	Exp ProjMan	Trans	Raw FP	Adj Factor	Adj FP	Dev Env	Year Fin	Entities
1	5152	1	4	253	305	34	302	1	85	52
2	5635	0	0	197	321	33	315	1	86	124
3	805	4	4	40	100	18	83	1	85	60
4	3829	0	0	200	319	30	303	1	86	119
5	2149	0	0	140	234	24	208	1	86	94
6	2821	0	0	97	186	38	192	1	86	89
7	2569	2	1	119	161	25	145	2	85	42
8	3913	1	2	186	238	25	214	1	83	52
9	7854	3	1	172	260	30	247	1	85	88
10	2422	3	4	78	116	24	103	1	83	38
11	4067	4	1	167	266	24	237	1	84	99
12	9051	2	1	146	258	40	271	1	84	112
13	2282	1	1	33	105	19	88	1	84	72
14	4172	3	4	162	223	32	216	1	85	61
15	4977	4	4	223	344	28	320	1	85	121
16	1617	3	2	119	167	26	152	2	85	48
17	3192	4	3	57	100	43	108	1	85	43
18	3437	4	4	68	384	20	326	2	86	316
19	4494	3	4	9	395	21	340	2	87	386
20	840	4	2	58	92	29	86	1	86	34
21	14973	4	4	318	587	34	581	2	86	269
22	5180	2	4	88	258	34	255	1	85	170
23	5775	2	4	306	438	37	447	1	86	132
24	10577	4	1	304	382	39	397	1	87	78
25	3983	1	4	89	289	33	283	1	86	200
26	3164	4	1	86	316	33	310	1	85	230
27	3542	2	0	71	306	37	312	1	86	235
28	4277	3	1	148	472	39	491	1	85	324
29	7252	4	4	116	286	27	263	1	85	170
30	3948	4	1	175	452	37	461	1	85	277
31	3927	4	3	79	207	27	190	1	86	128
32	710	1	1	145	183	27	168	3	86	38
33	2429	4	4	174	252	41	267	3	87	78
34	6405	1	1	194	285	35	285	1	85	91
35	651	2	2	126	175	38	180	3	88	49
36	9135	1	3	317	436	34	432	2	86	119
37	1435	2	4	289	377	28	351	3	87	88
39	847	1	4	158	217	18	180	3	88	59
40	8050	3	3	302	447	52	523	2	88	145
41	4620	1	1	451	499	28	464	1	87	48
42	2352	2	4	661	793	23	698	3	87	132
43	2174	1	1	64	118	25	106	1	88	54
45	6699	2	1	182	308	35	308	1	86	126
46	14987	2	3	173	505	19	424	1	87	332
47	4004	2	2	252	259	28	241	1	88	7

Project Ref	Effort	Exp Equip	Exp ProjMan	Trans	Raw FP	Adj Factor	Adj FP	Dev Env	Year Fin	Entities
48	12824	4	3	131	311	51	361	1	85	180
49	2331	2	3	106	145	6	103	1	85	39
50	5817	3	3	96	204	29	192	1	85	108
51	2989	2	3	116	188	18	156	1	85	72
52	3136	3	3	86	135	32	131	1	85	49
53	14434	2	3	221	342	35	342	1	85	121
54	2583	1	1	61	157	18	130	1	87	96
55	3647	1	3	132	221	5	155	2	86	89
56	8232	3	7	45	432	16	350	2	86	387
57	3276	1	1	55	167	12	128	2	86	112
58	2723	1	4	124	1176	14	139	2	87	52
59	3472	3	3	120	246	15	196	2	87	126
60	1575	1	2	47	79	14	62	2	87	32
61	2926	1	1	126	233	23	205	2	86	107
62	1876	3	2	101	146	15	117	2	86	45
63	2520	1	1	78	177	14	140	1	86	99
64	1603	4	7	69	143	14	112	1	86	74
65	3626	1	3	194	291	35	290	2	86	97
67	11361	2	4	323	507	35	504	2	87	184
68	1267	1	3	42	73	27	67	2	86	31
69	2548	1	2	74	117	25	105	2	87	43
70	1155	3	4	101	158	9	117	2	87	57
71	546	0	4	97	139	6	99	3	86	42
72	2275	2	3	134	211	13	165	2	84	77
73	9100	4	5	482	709	26	645	2	86	227
74	595	0	2	213	286	6	203	3	84	73
76	13860	2	3	473	655	40	688	2	86	182
77	1400	4	4	229	398	39	414	3	85	169
78	2800	4	3	227	300	34	297	1	83	73
79	9520	4	4	395	588	40	617	1	82	193
80	5880	4	3	469	645	43	697	3	86	176
81	23940	4	4	886	1127	34	1116	1	85	241

The Finnish Data Set

Project Ref	Effort	HW Type	App Type	FP	Company	UA	MA	AA	STD	UM	TA	PC	SR	CQ	CP	UT
20	17778	1	1	1364	10	5	4	4	5	4	3	4	4	5	4	3
21	8800	1	1	648	10	4	4	1	3	3	3	3	3	3	4	3
22	26670	1	1	1282	10	4	2	4	4	3	3	4	3	3	3	4
23	1330	1	1	176	10	3	5	5	5	5	3	4	3	3	4	4
24	14504	1	3	627	10	3	4	3	2	1	2	4	4	2	1	2
25	8498	1	1	1026	10	3	2	3	3	3	2	3	3	3	5	4
26	4830	1	5	561	10	5	3	5	4	3	2	4	4	3	3	5
27	3008	1	1	206	5	2	3	2	3	2	2	4	3	4	2	3
28	2525	1	1	128	5	3	3	3	3	2	3	5	4	4	2	3
29	4500	1	1	1814	3	4	3	3	2	3	3	4	3	3	3	4
30	1455	2	2	609	3	3	4	5	3	3	3	5	2	4	5	3
31	-1	1	2	210	3	3	3	3	3	3	3	4	2	4	3	3
32	1203	3	1	321	3	2	2	1	3	3	3	4	5	3	1	4
33	7537	1	4	355	4	3	4	4	4	4	3	2	3	5	3	3
34	8710	3	1	1058	4	4	4	3	5	4	4	5	3	4	4	3
35	796	3	3	65	4	4	2	2	3	3	2	4	4	4	4	3
36	11023	3	1	374	4	5	5	3	3	2	3	4	5	3	4	4
37	6030	2	1	1584	2	4	4	3	3	3	3	3	3	3	5	3
38	1750	1	1	464	2	4	2	3	3	3	4	3	3	3	3	4
39	2240	1	5	528	2	4	3	4	3	3	3	2	3	2	3	2
40	1105	1	1	233	2	4	4	4	3	3	4	2	3	4	3	4
41	2915	1	5	577	2	4	3	4	3	4	4	4	2	4	3	3
42	2100	1	1	786	8	4	4	3	2	2	3	2	3	2	4	4
43	2100	1	2	232	8	3	4	3	3	3	4	2	3	3	3	2
44	580	1	1	235	8	2	2	3	2	2	3	3	3	3	2	2
45	460	1	5	196	8	2	3	3	3	1	3	3	1	4	3	1
46	6182	1	3	677	6	4	3	4	3	3	3	4	4	4	5	5
47	4713	1	3	1035	6	5	4	4	2	4	4	4	3	3	3	3
48	8700	1	1	1056	6	4	3	3	3	3	3	4	3	4	4	4
49	8095	1	1	1598	6	4	3	4	3	3	3	4	3	3	3	4
50	18690	1	4	1619	9	4	3	4	3	4	3	2	3	4	4	3
51	-1	1	1	1229	9	4	3	3	3	3	3	5	4	4	4	5
52	592	1	3	402	9	3	4	3	3	2	2	3	2	3	1	2
53	23000	1	4	1347	7	3	2	3	3	3	3	4	4	2	4	4
54	17031	1	1	983	7	4	2	4	2	3	2	3	3	4	3	3
55	17200	1	1	1719	7	3	2	3	3	3	3	4	4	2	5	4
56	10850	1	4	1148	7	3	3	2	3	2	3	2	5	2	3	4
57	18900	1	4	1049	7	2	3	3	2	2	2	4	3	3	3	3
58	14568	1	4	755	7	5	4	2	1	2	3	4	3	4	2	3
59	780	1	2	189	7	4	3	2	3	3	4	3	4	4	4	3

Project Ref	MET	TME	MEX	PME	IN	INFP	QN	QFP	ON	OFP	SN	SFP	FN	FFP
20	4	5	3	4	89	485	32	67	43	273	17	119	51	420
21	2	2	4	3	10	33	0	0	19	100	36	246	24	269
22	2	1	1	4	96	514	51	255	23	166	29	176	20	171
23	3	3	2	3	10	40	8	32	4	20	2	14	7	70
24	4	4	5	3	20	80	20	80	28	140	18	126	20	200
25	3	2	2	3	42	162	40	160	40	194	30	210	30	300
26	3	2	4	4	21	81	20	80	29	145	15	105	15	150
27	3	4	5	3	5	22	5	15	3	11	6	34	13	124
28	3	2	4	3	5	33	0	0	2	16	0	0	10	79
29	3	4	4	3	65	252	104	283	61	227	30	291	50	761
30	4	4	4	3	22	79	88	397	2	15	2	10	12	108
31	3	4	4	3	11	24	19	72	14	37	0	0	8	77
32	2	4	4	3	1	4	15	27	2	7	23	263	2	20
33	4	3	2	5	17	47	43	97	6	28	3	9	18	174
34	5	2	4	4	38	138	45	145	69	274	17	170	28	331
35	4	4	5	3	7	15	1	4	6	38	0	0	2	8
36	4	3	4	4	29	142	8	11	19	117	10	77	3	27
37	4	2	4	3	198	404	54	67	46	250	11	74	53	789
38	2	3	2	3	28	86	5	5	12	61	1	3	27	309
39	2	2	5	3	33	100	59	125	7	50	2	12	24	241
40	2	5	1	3	13	25	22	80	1	4	0	0	11	124
41	4	3	5	4	65	177	32	99	0	0	0	0	31	301
42	4	3	4	4	70	226	64	162	4	16	9	59	28	323
43	4	4	4	3	12	37	15	29	24	86	3	13	13	67
44	3	2	2	3	10	26	19	30	9	44	4	20	11	115
45	3	5	4	3	10	32	10	20	9	47	5	25	9	72
46	4	4	5	4	22	70	60	148	43	258	5	29	18	162
47	4	5	1	4	39	207	54	151	15	91	0	0	43	586
48	4	3	4	5	63	227	74	163	30	134	2	12	42	520
49	3	4	5	4	82	409	97	282	21	148	0	0	46	759
50	2	5	4	5	75	354	83	255	44	289	8	61	47	660
51	4	2	3	3	76	407	87	275	53	245	7	36	24	266
52	3	3	4	3	6	48	0	0	0	0	27	187	15	167
53	2	2	1	1	18	96	26	123	36	210	13	83	59	835
54	2	1	1	2	10	60	5	26	26	87	6	60	50	750
55	1	1	1	1	74	290	33	110	36	170	27	249	60	900
56	2	2	2	1	112	270	7	32	18	74	2	6	53	766
57	2	3	3	4	77	258	62	152	132	479	14	82	12	78
58	2	1	2	2	31	142	26	130	33	174	10	85	20	224
59	3	3	3	4	10	24	20	52	5	25	0	0	10	88
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The Hughes Data Set

Project Ref	Effort	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
1	12422	32	10	5	21	11	77	7	44	26	15	70	0	0
2	10839	18	10	3	8	5	2	1	46	8	13	54	0	0
3	10672	18	9	0	3	1	9	1	52	1	9	53	0	0
4	1853	14	2	0	1	6	2	0	50	120	2	170	0	0
5	3890	50	2	1	6	6	9	0	49	120	3	169	0	0
6	1229	26	4	0	0	4	0	0	57	120	4	177	0	0
7	4239	20	2	1	11	6	13	1	49	40	3	89	0	0
8	2344	4	0	0	0	1	7	0	57	40	0	97	0	0
9	3467	25	0	0	0	12	4	1	46	1	0	47	0	0
10	5963	0	0	0	0	0	0	7	90	26	0	116	0	0
11	3779	14	4	0	3	25	5	0	34	8	4	42	0	0
12	4181	22	4	2	7	6	0	0	57	36	6	93	0	0
13	3136	0	0	0	0	0	52	1	57	65	0	122	0	0
14	2948	0	0	0	0	0	0	2	29	40	0	69	0	0
15	4095	1	0	0	0	4	6	0	33	23	0	56	369	0
16	4182	23	0	0	0	11	3	11	47	75	0	122	3072	0
17	3651	0	0	0	0	0	13	0	43	38	0	81	7750	0
18	1499	0	0	0	0	0	12	0	43	75	0	118	7730	0
19	2432	0	1	0	0	0	3	0	34	23	1	57	3830	0
20	321	0	0	0	0	0	12	0	36	13	0	49	63	1
21	358	0	0	0	0	0	6	0	13	13	0	26	54	1
22	693	0	0	0	0	0	11	0	13	13	0	26	111	1
23	723	0	0	0	0	0	12	0	22	22	0	44	561	1
24	408	0	0	0	0	0	7.5	0	5	36	0	41	249	1
25	300	0	0	0	0	0	24.5	0	6	16	0	22	240	1
26	618	0	7	3	2	0	11.5	0	29	32	10	61	534	1
27	1639	4	7	3	2	4	11.5	0	28	32	10	60	720	1
28	1506	4	7	3	0	0	11.5	0	9	17	10	26	291	1
29	5747	4	7	0	10	2	203	1	6	6	7	12	0	1
30	370	0	0	0	0	0	5	0	1	1	0	2	321	1
31	11936	6	8	5	46	3	245	6	62	89	13	151	12	1
32	4680	0	0	0	5	0	229	1	17	17	0	34	0	1
33	3900	13	0	0	0	9	23	4	106	131	0	237	172	1

The Kemerer Data Set

Project Name	Effort	Duration	KSLOC	AdjFP	RawFP
1	287	17	253.6	1217.1	1010
2	82.5	7	40.5	507.3	457
3	1107.31	15	450	2306.8	2284
4	86.9	18	214.4	788.5	881
5	336.3	13	449.9	1337.6	1583
6	84	5	50	421.3	411
7	23.2	5	43	99.9	97
8	130.3	11	200	993	998
9	116	14	289	1592.9	1554
10	72	5	39	240	250
11	258.7	13	254.2	1611	1603
12	230.7	31	128.6	789	724
13	157	20	161.4	690.9	705
14	246.9	26	164.8	1347.5	1375
15	69.9	14	60.2	1044.3	976

The MERMAID Data Set

Project Ref	Effort	Adj FP	Raw FP	Proj Type	AF 1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	238	23	23	E	5	0	4	0	1	5	4	4	4	0	2	4	0	2
2	490	38	42	E	3	2	3	2	2	4	2	2	2	1	0	2	1	0
3	616	36	44	E	3	3	0	0	0	5	0	2	1	0	0	0	3	0
4	910	57	51	E	5	4	4	4	4	5	4	3	5	0	3	2	3	0
5	1540	36	47	E	0	0	0	2	0	0	0	0	4	0	3	2	0	0
6	1680	29	38	E	0	3	0	0	0	1	4	0	2	0	0	1	0	0
7	1750	23	34	E	0	1	2	0	0	0	0	0	0	0	0	0	0	0
8	3234	99	115	N	3	4	0	0	0	1	0	2	0	5	0	2	0	4
9	3360	605	550	N	5	1	4	2	2	5	4	4	2	3	1	4	3	5
10	3850	34	42	E	3	1	0	0	3	1	2	1	4	0	1	1	0	0
11	5460	338	371	N	1	0	0	5	4	5	2	0	0	0	5	4	0	0
12	5110	133	157	E	3	3	3	2	1	1	0	2	2	0	3	0	0	0
13	6440	118	107	E	5	4	4	4	4	5	5	4	4	0	1	5	0	0
14	17920	653	634	N	3	3	3	2	4	4	2	3	1	3	3	5	0	2
15	18620	502	528	E	3	1	0	4	4	0	4	0	4	4	0	0	2	4
16	21280	306	268	N/A	5	5	4	3	4	5	4	4	4	1	2	2	3	3
17	24850	170	179	N	0	0	4	4	5	0	4	0	1	0	5	3	2	2
18	48230	911	884	N	3	3	3	2	4	4	2	3	1	3	3	5	0	2
19	3415	221	235	E	5	0	0	0	0	5	2	3	2	3	0	5	0	4
20	11551	613	626	N/A	3	2	4	2	3	5	0	5	2	1	0	4	0	2
21	4860	1507	1408	N	5	0	3	1	1	5	1	4	4	1	5	4	3	5
22	14224	559	N/A	E														
23	9080	218	291	E	0	0	0	0	2	0	0	0	4	1	3	0	0	0
24	1635	479	499	N	5	0	1	0	0	5	3	4	4	1	0	5	0	3
25	296	26	33	E	4	0	0	0	0	5	0	2	0	0	1	1	1	0
26	3720	125	137	E	5	1	0	1	1	5	0	4	5	0	0	0	3	1
27	4672	205	N/A	E														
28	2065	105	109	E	4	3	1	0	3	2	2	2	4	0	3	1	4	2
29	1690	114	107	E	5	4	4	5	5	5	0	5	3	0	0	1	0	2
30	504	36	38	E	5	0	0	0	0	5	2	3	2	3	0	5	0	4

The Real-Time 1 Data set

Project Ref	Effort	Host Machine	Life Cycle	Documentation Standard
1	573	SUN	RTSAOOD	DOD-2167A
2	446	SUN	RTSAOOD	DOD-2167A
3	127	SUN	RTSAOOD	DOD-2167A
4	400	SUN	RTSAOOD	DOD-2167A
5	189	SUN	RTSAOOD	DOD-2167A
6	200	IBM PC	RTSASD	INTERNAL
7	140	IBM PC	RTSASD	DOD-2167A
8	203	VAX	RTSASD	DOD-2167A
9	260	VAX	MASCOT	INTERNAL
10	2100	VAX	WATERFA	DOD-2167A
11	174	VAX	WATERFA	INTERNAL
12	1000	VAX	RTSAOOD	JSP 188
13	1679	VAX	RTSASD	INTERNAL
14	1225	VAX	RTSASD	INTERNAL
15	394	VAX	RTSASD	INTERNAL
16	1600	SUN	MASCOT	JSP 188
17	107	IBM PC	RTSASD	DOD-2167A
18	134	SUN	RTSAOOD	JSP 188
19	455	VAX	MASCOT	INTERNAL
20	270	VAX	MASCOT	INTERNAL
21	1042	SUN	RTSASD	INTERNAL

The Telecoms 1 Data Set

Project Ref	Actual Effort	Files
1	305.22	105
2	330.29	237
3	333.96	98
4	150.4	24
5	544.61	197
6	117.87	39
7	1115.54	284
8	158.56	37
9	573.71	53
10	276.95	116
11	97.45	38
12	374.34	180
13	167.12	43
14	358.37	84
15	123.1	257
16	23.54	6
17	34.25	5
18	31.8	3