Resource Estimation in Software Engineering

Lionel C. Briand
Carleton University
Systems and Computer Engineering Dept.
Ottawa, ON K1S 5B6 Canada
briand@sce.carleton.ca

Isabella Wieczorek
Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6
67661 Kaiserslautern
wieczo@iese.fhg.de

International Software Engineering Research Network, Technical Report

Resource Estimation in Software Engineering

Lionel C. Briand and Isabella Wieczorek

1 Introduction

This paper presents a comprehensive overview of the state of the art in software resource estimation. We describe common estimation methods and also provide an evaluation framework to systematically compare and assess alternative estimation methods. Though we have tried to be as precise and objective as possible, it is inevitable that such a comparison exercise be somewhat subjective. We however, provide as much information as possible, so that the reader can form his or her own opinion on the methods to employ. We also discuss the applications of such estimation methods and provide practical guidelines.

Understanding this article does not require any specific expertise in resource estimation or quantitative modeling. However, certain method descriptions are brief and the level of understanding that can be expected from such a text depends, to a certain extent, upon the reader's knowledge. Our objective is to provide the reader with a comprehension of existing software resource estimation methods as well as with the tools to reason about estimation methods and how they relate to the reader's problems.

Section 2 briefly describes the problems at hand, the history and the current status of resource estimation in software engineering research and practice. Section 3 provides a comprehensive, though certainly not complete, overview of resource estimation methods. Project sizing, an important issue related to resource estimation, is then discussed in Section 4. Section 5 defines an evaluation framework that allows us to make systematic and justified comparisons in Section 6. Section 7 provides guidelines regarding the selection of appropriate estimation methods, in a given context. Section 8 describes typical scenarios for using resource estimation methods, thereby relating them to software management practice. Section 9 attempts to define important research and practice directions, requiring the collaboration of academia and industry. Section 10 then concludes this article by summarizing the main points made throughout the article.

2 Resource Estimation in Software Engineering

This section briefly summarizes current practices regarding resource estimation in software engineering. This will shed light on the importance of the research being conducted and presented in subsequent sections.

2.1 Current Practices and Motivations

The estimation of resource expenditures (e.g., effort, schedule) is an essential software project management activity. Inaccuracies in such estimations have repeatedly shown to lead to disasters (Heemstra, 1992). Delivering software on-time and within budget is a critical concern for organizations as underestimating software projects can have detrimental effects on business reputation, competitiveness, and performance (Putnam and Myers 1992; Lederer and Prasad, 1993). On the other hand, overestimated projects can result in poor resource allocation and missed opportunities to take on other projects. Industry has recognized that by improving project estimation methods, problems with resource allocation and schedule forecasting can be overcome (Heemstra, 1992). Project managers commonly stress the importance of improving estimation accuracy and methods to support better estimates (Lederer et al., 1990, Lederer and Prasad, 1992). Accurate estimates are crucial for better project planning, tracking, and control and pave the way for successful project delivery.

Software resource estimation is considered to be more difficult than resource estimation in other industries. Software organizations typically develop new products as opposed to fabricate the same product over and over again. This leads to many difficulties in estimating the resources of a project, especially in early project phases. Many studies have confirmed that a majority of projects overrun their budget (e.g., Jenkins et al., 1984; Heemstra, 1992; Lederer and Prasad, 1992; Lederer and Prasad, 1995).

Practitioners are often faced with the lack of explicit cost and resource data collected from past projects (Heemstra, 1992). Moreover, estimators rely more on their personal memory and informal procedures than on documented facts, standards, or arithmetic rules (Hihn, 1991; Lederer and Prasad, 1992). Hence, estimates based on expert judgement and based on the available capacity prove to be quite popular (Heemstra, 1992). If actual data was recorded than only little time is spent to determine reasons for any difference between actual and plan data (van Genuchten, 1991).

Many commercial packages are available on the market and one might be tempted to use some of them. However, experience has shown that one should not rely on software estimation tools for an accurate estimate, i.e., no difference was found between organizations using the tools and those not using the tools in terms of accuracy (Lederer and Prasad, 1992).

The difficulties in software estimation are related to a variety of practical, measurement, and modeling issues. To address these issues, it is necessary to follow a systematic estimation process that increases the estimation quality and repeatability. A systematic estimation process may be supported by techniques, models, or/and tools. A number of resource estimation methods (i.e., a combination of techniques, models, tools) are currently available and they show contrasting strengths and weaknesses as well as different underlying assumptions. This makes it difficult to decide which one is best suited in a given context. To appropriately select one or more estimation methods, it is necessary to assess and compare their characteristics and their impact in practical estimation contexts.

2.2 History and Brief Overview

The study of software resource estimation started as early as the late 1950’s and 1960’s (Norden, 1958; Nelson, 1966). (Brooks, 1975) raised concerns over “gutless estimating” whereby managers tended to produce “wish-derived estimates” and emphasized a need for an approach capable of justifying the estimate. The 1970’s proved to be a significant period of estimation methods’ development. Most of the early methods provided ready-to-use models, including pre-defined cost drivers that were then applied to obtain direct point estimates (Wolverton, 1974; Walston-Felix, 1977; Putnam, 1978; Albrecht, 1979). At that point, it became clear that there were difficulties in selecting cost drivers from an ever-increasing list of variables that were believed to influence software development effort. Practical experience has shown that early methods either emphasized project sizing, cost drivers, or expert judgement, but never a combination of the three (Conte et al., 1986). Hence, Conte stressed the need for methods that incorporate a combination of analytic equations, statistical data fitting, and expert judgement.

These methods take into account adjustments to nominal estimates made by experts. The best-known method of this type, proposed by (Boehm, 1981), is the Constructive Cost Model (COCOMO). It provides equations that incorporate system size as the principal effort driver. Predicted development effort is then adjusted to accommodate the influence of 15 additional cost drivers. Other examples of this type of methods are SLIM (Putnam, 1978) and COPMO (Conte et al., 1986).

In the 1980's, widely used parametric methods (Putnam, 1978; Albrecht, 1979; Boehm, 1981; Bailey and Basili, 1981) were compared using data sets of various sizes and environments. Some of the main conclusions were that these models perform poorly when applied uncalibrated to other environments (Kitchenham and Taylor, 1985; Conte et al., 1986; Kemerer, 1987). Moreover, many estimation methods were automated and packaged into commercial tools (Stutzke, 1996).

Software development is a complex dynamic process (Abdel-Hamid and Madnick, 1991). We know little about the complex, ever-changing relationships that explain variations in productivity. Therefore, the 1990's saw the introduction and evaluation of non-parametric modeling techniques based on machine learning algorithms, such as Optimized Set Reduction (Briand et al., 1992), Artificial Neural Networks (Jørgensen, 1995; Finnie at al., 1997), CART regression trees (Srinivasan and Fisher, 1995; Kitchenham, 1998), and Analogy-based estimation (Mukhopadyay et al., 1992; Shepperd and Schofield, 1997; Walkerden and Jeffery, 1999).

These methods typically make weak assumptions about the data and some of them produce models that can easily be interpreted and can accommodate complex relationships and interactions between cost drivers. They are therefore well suited to early exploratory investigations and theory construction.

In addition, expert judgment and methods combining expert opinion with historical data have been investigated and compared. Examples are, subjective effort estimation (Höst and Wohlin, 1998; Stensrud and Myrtveit, 1998), modeling based on expert knowledge elicitation (Briand et al. 1998a), and techniques combining expert opinion and project data (Chulani et al., 1999). As discussed further below, such approaches are likely to be key in the future developments of the software resource estimation field.

2.3 Status and Main Obstacles

Cost estimation techniques have drawn upon a variety of fields, statistics, machine learning, and knowledge acquisition. Given the diversity of estimation techniques one is faced with the difficult exercise of determining which technique would be the best in given circumstances. In order to assess a technique's appropriateness, the underlying assumptions, strengths, and weaknesses have to be known and its performances must be assessed.

Homogeneous, company-specific data are believed to form a good basis for accurate resource estimates. Data collection, however, is an expensive, time-consuming process for individual organizations. There have been recent developments in standard data collection. The collaboration of organizations to form multi-organizational data sets provides the possibility for reduced data collection costs, faster data accumulation and shared information benefits. Their administrators offer a standard channel of data collection. Therefore, the pertinent question remains whether multi-organizational data are valuable to estimation methods.

3 Overview of Estimation Methods

This section gives a general overview of the resource estimation literature classifies existing cost estimation methods into categories according to their underlying assumptions and modeling characteristics, and describes a selection of methods in more detail with a focus on effort estimation.

There is a wealth of research addressing the software resource estimation problem. Research activities can be classed as:

1. Evaluation of effort estimation methods in different contexts. Investigations are aimed at (a) determining which method has the greatest effort prediction accuracy (e.g., Jørgensen 1995; Finnie et al., 1997; Walkerden and Jeffery, 1999; Briand et al., 1999b; Briand et al., 2000) (b) proposing new or combined methods that could provide better estimates (e.g., Conte et al. 1986; Briand et al., 1992; Schepperd and Schofield, 1997; Stensrud and Myrtveit, 1998).

2. Identification of significant cost drivers and productivity factors across different contexts (e.g., Subramanian and Breslawski, 1994; Maxwell et al., 1996; Briand et al., 1999a).

3. Assessment of current industry software development practices (e.g., Heemstra, 1992; Lederer and Prasad, 1992; Lederer and Prasad, 1998)

4. Calibration of effort estimation methods to tailor them to individual organizations (e.g., Miyazaki and Mori, 1985; Cuelenaere, 1987).

General overviews and surveys of software cost estimation can be found in several papers. Stutzke (Stutzke, 1996) gives a chronological overview of generic cost estimation methods and tools, such as COCOMO (Constructive Cost Model), SLIM (Software Life Cycle Management), PRICE-S (Parametric Review of Information for Cost Evaluation), or Function Points (FP)[1]. Kitchenham and Boehm give overviews and subjective evaluations of well-known cost models (Kitchenham, 1990; Boehm, 1981). Wakerden and Jeffery (Walkerden and Jeffery, 1997) give a comprehensive overview of the cost estimation process and its relation to the Quality Improvement Paradigm (Basili and Rombach, 1988), cost estimation models and practices. Lederer and Prasad (Lederer and Prasad, 1992) derive practical management guidelines based on a survey of 114 IT managers in the US. Heemstra (Heemstra, 1992) reports on findings from studying cost estimation practices of 400 Dutch companies. His work mainly comprises the usage of different methods by different organizations. Wrigley and Dexter (Wrigley and Dexter, 1987) provide a review of cost estimation methods and stress several issues in cost estimation, like the software sizing problem, or the independence of factors impacting development effort.

3.1 Classification Schema

Researchers have made a number of attempts to classify software cost models. This is useful, because it permits the evaluation and comparison of model types. Boehm (Boehm, 1981; Boehm, 1984) introduced seven classes: Algorithmic Models, Expert Judgment, Analogy, Parkinson, Price to Win, Top-Down, Bottom-Up. Some of these classes, like Price-to-Win, cannot really be considered to be an estimation technique. Moreover, some classes are not orthogonal, e.g., expert

[1] The latter one is not really comparable as this is mostly a sizing technique as discussed in Section 4.

judgment can be used following a bottom-up estimation process. Similarly, it is difficult to distinguish, for example, the Algorithmic and the Top-Down method. Walkerden and Jeffery (Walkerden and Jeffery, 1997) defined a framework consisting of four classes of prediction methods: Empirical, Analogical, Theoretical, and Heuristic. Unfortunately, they state that expert judgment cannot be included in their framework. In addition, the classes Analogy and Heuristic can overlap, as heuristics can be included in the analogy estimation process (see adaptation rules Section 3.5.2). Moreover, it is not evident why methods using Analogy are not empirical as well since certain components of the analogy method can be derived empirically (see similarity functions 3.5.2). Kitchenham (Fenton and Pfleeger, 1996) classifies current approaches to cost estimation into four classes: expert opinion, analogy, decomposition, and models. Here, decomposition can be seen as estimating the effort in a bottom-up manner. Thus, this category overlaps with the other three classes as it not orthogonal to them.

Unfortunately, classification schemes are subjective and there is no agreement about the best one (Kitchenham and de Neumann, 1990). Our classification is not fully satisfactory but is designed to follow our argumentation and evaluation regarding the various types of estimation methods. The classification schema is hierarchical, starting from two main categories (Model Based Methods, Non-Model Based Method) that are further refined into sub-categories. The hierarchy should cover all possible types of resource estimation methods, without being overly complicated for our purpose. Such a classification will help us talking in general terms of a certain type of method.

Figure 1 summarizes the classification schema we propose for cost estimation methods. The letters in brackets are explained and used in Section 5. Each class is described in the following sub-sections.

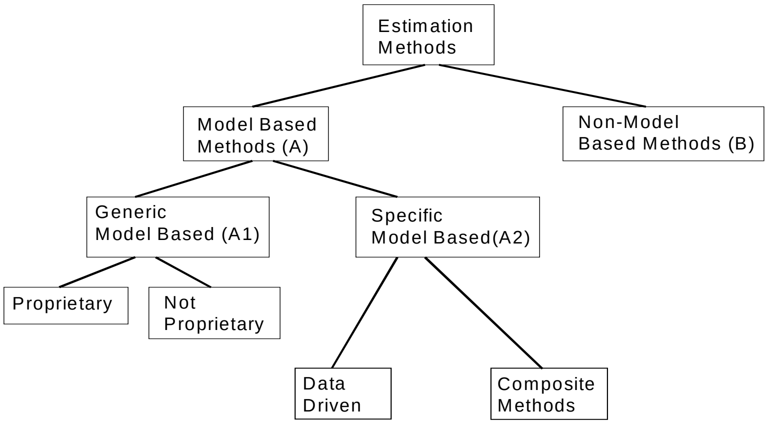

Figure 1: Classification of Resource Estimation Methods

3.1.1 Model Based Methods

Model-based estimation methods, in general, involve at least one modeling method, one model, and one model application method (see Section 5.2). An effort estimation model usually takes a number of inputs (productivity factors and an estimate of system size) and produces an effort point estimate or distribution.

- **Generic Model Based**
  These estimation methods generate models that are assumed to be generally applicable across different contexts.

  - *Proprietary*:
    Modeling methods and models are not fully documented or public domain.

  - *Not Proprietary*:
    Modeling methods and models are documented and public domain.

- **Specific Model Based**:
  Specific Model Based estimation methods include local models whose validity is only ensured in the context where they have been developed.
  - *Data Driven*:
    Modeling methods are based on data analysis, i.e., the models are derived from data. We may distinguish here further between parametric and non-parametric modeling methods. Parametric methods require the a priori specification of a functional relationship between project attributes and cost. The modeling method then tries to fit the underlying data in the best way possible using the assumed functional form. Non-Parametric methods derive models that do not make specific assumptions about the functional relationship between project attributes and cost (although, to a certain extent, there are always assumptions being made).

  - *Composite Methods*:
    Models are built based on combining expert opinion and data-driven modeling techniques. The modeling method describes how to apply and combine them in order to build a final estimation model.

3.1.2 Non-Model Based Methods

Non-model based estimation methods consist of one or more estimation techniques together with a specification of how to apply them in a certain context. These methods are not involving any model building but just direct estimation.

Usually Non-Model based methods involve consulting one or more experts to derive a subjective effort estimate. The effort for a project can be determined in a bottom-up or top down manner. A top-down approach involves the estimation of the effort for the total project and a splitting it among the various system components and/or activities. Estimating effort in a bottom-up manner involves effort estimates for each activity and/or component separately and the total effort is an aggregation of the individual estimates, possibly involving an additional overhead.

3.2 Description of Selected Methods

There exists a large number of software cost estimation methods. In order to scope down the list of methods we will discuss here, we focus on methods that fulfill the following high-level criteria:

1. **Recency:** We will exclude methods that have been developed more than 15 years ago and were not updated since then. Therefore, we will not discuss in detail models like the Walston-Felix Model (Walston and Felix, 1977), the Bailey-Basili Model (Bailey and Basili, 1981), the SEER (System Evaluation and Estimation Resources) model (Jensen, 1984) or the COPMO (Comparative Programming MOdel) model (Conte et al., 1986).

2. **Level of Description:** Moreover, the focus is on methods that are not proprietary. We can only describe methods for which the information is publicly available, unambiguous, and somewhat unbiased. Therefore, we will not discuss in detail the generic proprietary methods ("black-box" methods). The level of detail of the description for this kind of methods depends on the level of available, public information. Examples of proprietary estimation methods (and tools) are PRICE-S (Cuelenaere et al., 1987; Freiman and Park, 1979; Price Systems), Knowledge Plan (Software Productivity Research; Jones, 1998), ESTIMACS (Rubin 1985; Kemerer, 1987; Kusters et al., 1990).

3. **Level of Experience:** We only consider methods for which experience has been gained and reported in software engineering resource estimation. This means an initial utility should already be demonstrated.

4. **Interpretability:** We will focus on methods for which results are interpretable, i.e., we know which productivity factors have a significant impact and how they relate to resource expenditures. Non-interpretable results are not very likely to be used in practice, as software practitioners usually want to have clear justifications for an estimate they will use for project planning. We will, therefore, not provide a detailed discussion of Artificial Neural Networks. The reader is referred to (Zaruda, 1992; Cheng and Titterinton, 1994).

3.3 Examples of Non-Proprietary Methods

3.3.1 COCOMO – COnstructive COst MOdel

COCOMO I is one of the best-known and best-documented software effort estimation methods (Boehm, 1981). It is a set of three modeling levels: Basic, Intermediate, and Detailed. They all include a relationship between system size (in terms of KDSI delivered source instructions) and development effort (in terms of person month). The intermediate and detailed COCOMO estimates are refined by a number of adjustments to the basic equation. COCOMO provides equations for effort and duration, where the effort estimates excludes feasibility and requirements analysis, installation, and maintenance effort. The basic COCOMO takes the following relationship between effort and size

$$PersonMonth = a(KDSI)^{b}$$

The coefficients *a* and *b* depend on COCOMO's modeling level (basic, intermediate, detailed) and the mode of the project to be estimated (organic, semi-detached, embedded). In all the cases, the value of *b* is greater than 1, thus suggesting that COCOMO assumes diseconomies of scale. This means that for larger projects the productivity is relatively lower than for smaller projects. The coefficient values were determined by expert opinion. The COCOMO database (63 projects) was used to refine the values provided by the experts, though no systematic, documented process was followed.

The mode of a project is one of three possibilities. Organic is used when relatively small software teams are developing within a highly familiar in-house environment. Embedded is used when tight constraints are prevalent in a project. Semi-detached is the mid-point between these two extremes.

Intermediate and Detailed COCOMO adjust the basic equation by multiplicative factors. These adjustments should account for the specific project features that make it deviate from the productivity of the average (nominal) project. The adjustments are based on ranking 15 cost-drivers. Each cost-driver's influence is modeled by multipliers that either increase or decrease the nominal effort. The equations for intermediate and detailed COCOMO take the following general form

$$Effort = a\, Size^{b} \prod_{i=1}^{15} EM_{i}$$

where EM_i is a multiplier for cost-driver *i.*

Intermediate COCOMO is to be used when the major components of the product are identified. This permits effort estimates to be made on a component basis. Detailed COCOMO even uses cost driver multipliers that differ for each development phase.

Some adaptations to the original version of COCOMO exist which can cope with adapted code, assess maintenance effort, and account for other development processes than for the traditional waterfall process. In the late 1980's, the Ada COCOMO model was developed to address the specific needs of Ada projects.

The **COCOMO II** research started in 1994 and is initially described (Boehm et al., 1995). COCOMO II has a tailorable mix of three models, Applications Composition, Early Design, and Post Architecture. The Application Composition stage involves prototyping efforts. The Early Design stage involves a small number of cost drivers, because not enough is known at this stage to support fine-grained cost estimation. The Post Architecture model is typically used after the software architecture is well defined and estimates for the entire development life cycle. It is a detailed extension of the early design model.

COCOMO II (Post Architecture) uses 17 effort multipliers and 5 exponential scale factors to adjust for project (replacing the COCOMO I development modes), platform, personnel, and product characteristics. The scale factors determine the dis/economies of scale of the software under development and replace the development modes in COCOMO I. The post architecture model takes the following form.

$$Effort = a\, Size^{b} \prod_{i=1}^{17} EM_{i}$$

$$where\, b = 1.01 + 0.01 \overset{5}{\underset{j}{}} ScaleFactor_{j}$$

Major new capabilities of COCOMO II are (1) size measurement is tailorable involving KLOC, Function Points, or Object Points, (2) COCOMO II accounts for reuse and reengineering, (3) exponent-driver approach to model diseconomies of scale, (4) several additions, deletions, and updates to previous cost drivers (Boehm et al., 1996).

In 1997, a new COCOMO II version included a 10% weighted average approach to adjust the a-priori expert-determined model parameters. The underlying database consisted of 83 projects and included 166 data points in 1998. A new version COCOMO II, version 1998, involved Bayesian Statistics to adjust the expert-determined model parameters. Major steps involve to (1) determine a-priori multipliers for cost-drivers using expert judgment (prior information), (2) estimate data-based multipliers based on a sample of project data, (3) combine non-sample prior information with data information using Bayesian inference statistics, (4) estimate multipliers for combined information (posterior information). The Bayesian Theorem combines prior information (expert knowledge) with sample information (data model) and derives the posterior information (final estimates) (Chulani et al., 1999). Usually the multiplier information is obtained through distributions. If the variance of an a-priori (expert) probability distribution for a certain multiplier is smaller than the corresponding sample data variance, then the posterior distribution will be closer to the a-priori distribution. We are in the presence of noisy data and more trust should be given then to the prior (expert) information. It is worth noting that COCOMO II is also what we called in Figure 1 a composite method and that we could have described in Section 3.5 too. We decided to leave it in this section as this is also a generic model and this is where the first version of COCOMO is described.

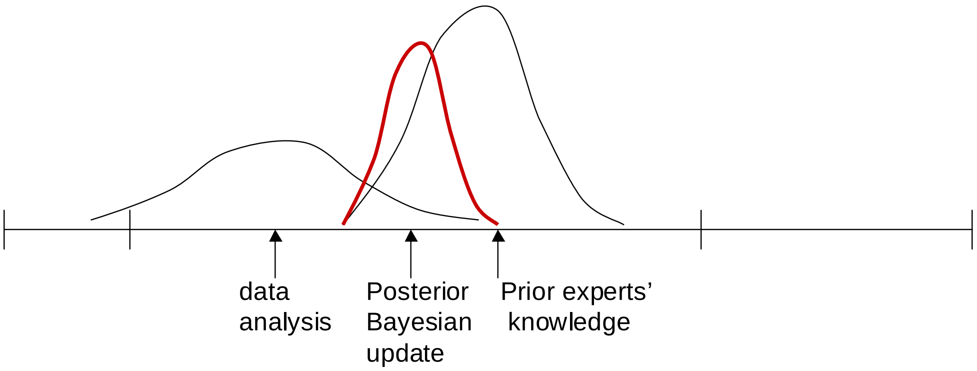

Figure 2: Example of a prior, posterior, and sample Distribution

Figure 2 illustrates the situation described above and could be the multiplier information for any of the cost-drivers in the COCOMO model. The arrows indicate the mean values of the distributions. In this case, the posterior distribution is closer to the experts' distribution.

3.3.2 SLIM – Software Life Cycle Management

The Putnam method is based on an equation of staffing profiles for research and development projects (Putnam, 1978), (Londeix, 1987), (Putnam and Myers, 1992). Its major assumption is that the Rayleigh curve can be used to model staff levels on large (>70,000 KDSI) software projects. Plotting the number of people working on a project is a function of time and a project starts with relatively few people. The manpower reaches a peak and falls off and the decrease in manpower during testing is slower than the earlier build up. Putnam assumes that the point in time when the staff level is at its peak should correspond to the project development time. Development effort is assumed to be 40% of the total life cycle cost. Putnam explicitly excludes requirements analysis and feasibility studies from the life cycle.

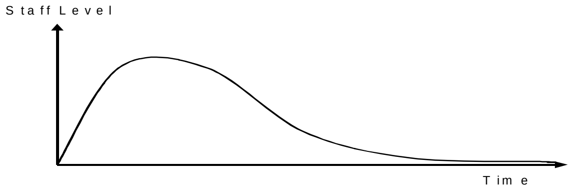

Figure 3: Rayleigh Curve Example

The basic Rayleigh form is characterized through a differential equation

$$y' = 2Kat\,exp(-at^2)$$

y' is the staff build-up rate, *t* is the elapsed time from start of design to product replacement, *K* is the total area under the curve presenting the total life cycle including maintenance, *a* is a constant that determines the shape of the curve.

In order to estimate project effort (*K*) or development time (t_d) two equations have been introduced and can be derived after several steps.

$$t_d = \left[(S)^3 / \left(D_0 C^3\right)\right]^{1/7}$$
$$K = (S/C)^{9/7}(D_0)^{4/7}$$

S is system size measured in KDSI, D_0 is the manpower acceleration (can take six different discrete values depending on the type of project), *C* is called the technology factor (different values are represented by varying factors such as hardware constraints, personnel experience, programming experience). To apply the Putnam model it necessary to determine the *C*, *S* and D_0 parameters up-front.

3.4 Examples of Data Driven Estimation Methods

This section describes a selection of existing data-driven estimation methods. Key characteristics and short examples are provided for each estimation method.

3.4.1 CART - Classification and Regression Trees

Two types of decision trees are distinguished, called classification and regression trees (Breiman et al., 1984; Salford Systems). The intention of classification trees is to generate a prediction for a categorical (nominal, ordinal) variable (Briand et al., 1999c; Koshgoftaar et al., 1999), whereas regression trees generate a prediction along a continuous interval or ratio scale. In the context of software resource estimation, it is therefore natural to use regression trees.

Regression trees classify instances (in our case software projects) with respect to a certain variable (in our case productivity). A regression tree is a collection of rules of the form: *if (condition 1 and …and condition N) then Z* and basically form a stepwise partition of the data set being used. The dependent variable (Z) for a tree may be, for example, effort (Srinivasan and Fisher, 1995) or productivity (Briand et al., 1998; Briand et al., 1999b; Kitchenham, 1998).

Each node of a tree specifies a condition based on one of the project variables selected. Each branch corresponds to possible values of this variable. Regression trees can deal with variables measured on different scale types.

Building a regression tree involves recursively splitting the data set until (binary recursive partitioning) a stopping criterion is satisfied. The key elements to build a regression tree are: (1) recursively splitting each node in a tree in an effective way, (2) deciding when a tree is complete, (3) computing relevant statistics (e.g., quartiles) for each terminal node. There exist several definitions for split criteria. For example, the selection of the variable that maximally reduces the mean squared error of the dependent variable (Salford Systems). Usually splitting generates two partitions but there are other possible approaches (Porter and Selby, 1990).

The example in Figure 4 is a regression tree derived from analyzing European Space Agency projects (Briand et al., 1998). Each terminal node represents the average productivity of projects that are characterized by the path from the root. Each node has a condition. If a condition is true then the path on the left is taken. On the first level of the tree, projects are first split according to their team size. If the team size is lower or equal to 7 persons, these projects are split further according to their category. For 29 projects the team size is lower or equal to 7 persons; 27 projects have a team size greater than 7 persons. Following the left branch: Projects falling in the categories “on board systems”, or “simulators” have a mean productivity of 0.35 KLOC/PM (thousands of lines of source code per person-month). Projects with more than 7 team members and where tool usage is between low and nominal have a predicted productivity of 0.09 KLOC/PM. This is the case for 16 projects. Tool ≤ nominal would mean no tool or basic lower CASE tools, and higher than nominal would mean upper CASE, project management and documentation tools used extensively. Projects with a team larger than 7 persons and high-very high usage of tools have a predicted productivity of 0.217 KLOC/PM. This holds for 11 projects of the analyzed projects.

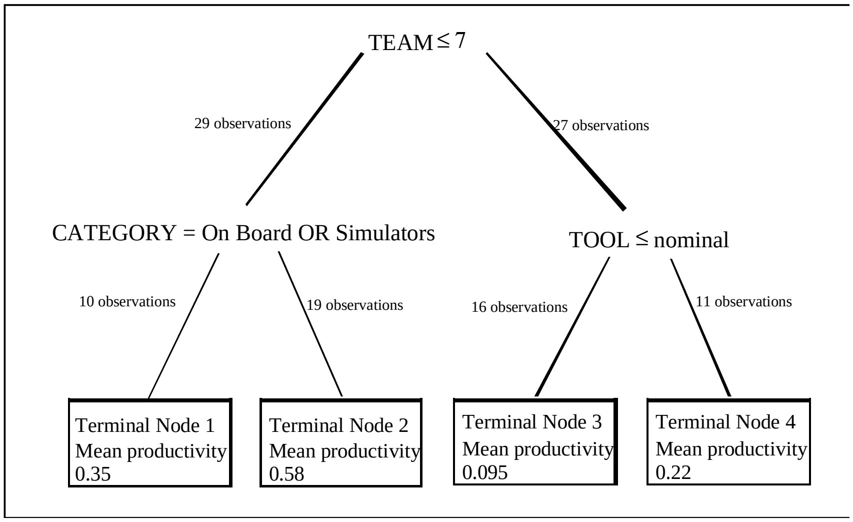

Figure 4: Example of a Regression Tree

A new project can be classified by starting at the root node of the tree and selecting a branch based on the project’s specific variable value. One moves down the tree until a terminal node is reached. For each terminal node and based on the projects it contains, the mean, median, and quartile values are computed for productivity. These statistics can be used for benchmarking purposes. One can, for example, determine whether a project's productivity is significantly below the node mean value. More precisely, one may determine in which quartile of the node distribution the new project lies. If the project lies in the first quartile, then it is significantly lower than expected and the reason why this happened should be investigated. Regression trees may also be used for prediction purposes. Productivity for a node can be predicted by using the mean value of the observations in the node (Briand et al., 1998). This requires building the tree only with information that can be estimated at the beginning of a project.

3.4.2 OSR – Optimized Set Reduction

Optimized Set Reduction (OSR) (Briand et al., 1992; Briand et al., 1993) determines subsets in a historical data set (i.e., training data set) that are "optimal" for the prediction of a given new project to estimate. It combines machine learning principles with robust statistics to derive subsets that provide the best characterizations of the project to be estimated. The generated model consists of a collection of logical expressions (also called rules, patterns) that represent trends in a training data set that are relevant to the estimation at hand. The main motivations behind OSR was to identify robust machine learning algorithm which would better exploit project data sets and still generate easy-to-interpret rule-based models.

As mentioned, OSR dynamically builds a different model for each project to be estimated. For each test project, subsets of "similar" projects are extracted in the underlying training set. Like regression trees, this is a stepwise process and, at each step, an independent variable is selected to reduce the subset further and the projects that have the same value (or belong to the same class of values) as the test project are retained. One major difference with regression trees is that the selected variable only has to be a good predictor in the value range relevant to the project to estimate. The (sub)set reduction stops when a termination criterion is met. For example, if the subset consists of less than a number of projects or the difference in dependent variable distribution with the parent (sub)set is not significant.

A characteristic of the "optimal" subsets is that they are characterized by a set of conditions that are true for all objects in that subset and they have optimal probability distributions on the range of the dependent variable. This means that they concentrate a large number of projects in a small number of dependent variable categories (if nominal or ordinal) or on a small part of the range (if interval or ratio). A prediction is made based on a terminal subset that optimally characterizes the project to be assessed. It is also conceivable to use several subsets and generate several predictions whose range reflects the uncertainty of the model. An example application of OSR for the COCOMO data set (Boehm, 1981) can be found in (Briand et al., 1992). The example rule below consist of a conjunction of three of the COCOMO cost-drivers (required reliability, virtual machine experience, data base size):

RELY=High AND VEXP=High AND DATA=Low => 299 LOC/Person-Month

Such logical expression is generated for a given test project and is optimal for that particular project. This project's productivity can be predicted based on the probability distribution of the optimal subset, e.g., by taking the mean or median. In addition to a point estimate, a standard deviation can easily be estimated based on the subset of past projects characterized by the rule. Because OSR dynamically generates patterns that are specific and optimal for the project to be estimated, it uses the data set at hand in a more efficient way than, say, regression trees.

3.4.3 Stepwise ANOVA – Stepwise Analysis of Variance

This procedure combines ANOVA with OLS regression and aims to alleviate problems associated with analyzing unbalanced data sets (Kitchenham, 1998). The values that ordinal and nominal-scale variables can take are called levels. A specific combination of variables and levels is called cells. An unbalanced data set has either unequal numbers of observations in different possible cells or a number of observations in the cells that are disproportional to the numbers in the different levels of each variable. Most software engineering data sets are unbalanced and therefore impacts of variables may be concealed and spurious impacts of variables can be observed (Searl, 1987).

The Analysis of Variance (ANOVA) usually decides whether the different levels of an independent variable (e.g., a cost factor) affect the dependent variable (e.g., effort). If they do, the whole variable has a significant impact. ANOVA is used for ordinal or nominal variables. The stepwise procedure applies ANOVA using each independent variable in turn. It identifies the most significant variable and removes its effect by computing the residuals (difference between actual and predicted values). ANOVA is then applied again using the remaining variables on the residuals. This is repeated until all significant variables are identified. Ratio and interval variables can be included in this procedure. Their impact on the independent variable is obtained using OLS regression (for a description of regression see Section 3). The final model is an equation with the most significant factors. For example: if RELY (Reliability) with three levels, and RVOL (Requirements Volatility) with two levels were found to be the most significant, the final model was:

Predicted Productivity = $\mu_{1,1\mathrm{RELY}} + \mu_{1,2\mathrm{RELY}} + \mu_{1,3\mathrm{RELY}} + \mu_{2,1\mathrm{RVOL}} + \mu_{2,2\mathrm{RVOL}}$

Where, $\mu_{\text{<iteration>,<level><independent variable>}}$ is the mean productivity value in case the cost-driver <independent variable> has the value <level>. In this example, the cost-driver RVOL is identified as significant in the second iteration after the effect of RELY was removed.

3.4.4 OLS – Ordinary Least Squares Regression

Ordinary least-square regression (OLS) assumes a functional form relating one dependent variable (e.g., effort), to one or more independent variables (i.e., cost drivers) (Berry and Feldman, 1985).

With least-squares regression, one has first to specify a model (form of relationship between dependent and independent variables). The least squares regression method then fits the data to the specified model trying to minimise the overall sum of squared errors. This is different, for example, from machine learning techniques where no model needs to be specified beforehand.

There are several commonly used measures/concepts when applying regression analysis. We will shortly explain their meaning. For further details, refer to (Schroeder et al., 1986).

In general, a linear regression equation has the following form[2]:

$$DepVar = a + (b_1 \times IndepVar_1) + \ldots + (b_n \times IndepVar_n)$$

[2] If the relationship is exponential, the natural logarithm has to be applied on the variables involved and a linear regression equation can be used

Where *DepVar* stands for dependent variable and the *IndepVar*'s are the independent variables. The dependent variable is the one that is to be estimated. The independent variables are the ones that have an influence on the dependent variable. We wish to estimate the coefficients that minimise the distance between the actual and estimated values of the *DepVar*. The coefficients $b_1 \dots b_n$ are called regression coefficients and *a* is referred to as the intercept. It is the point where the regression plane crosses the y-axis.

Four functional forms have been investigated in the literature for modeling the relationship between system size and effort. These are summarized in Table 1 (Briand et al., 1999a).

Model Specification	**Model Name**
$Effort = a + (b \times Size)$	Linear Model
$Effort = a + (b \times Size) + (c \times Size^2)$	Quadratic Model
$Effort = e^a \times Size^b$	Log-linear Model
$Effort = e^a \times Size^b \times Size^{c \times \ln Size}$	Translog Model

Table 1: Functional Forms of Regression-based Effort Estimation Models

The statistical significance is a measure of the likelihood for a result to be "true", i.e., representative of the population. The p-value is the probability of error that is involved in accepting the observed result as valid. The higher the p-value, the more likely the observed relation between a dependent and an independent variable may be due to chance. More precisely, the p-value for one of the coefficients indicates the probability of finding a value that is different from zero, if the value is zero in the population. If the p-value is high then we have weak evidence that the coefficient value is different from zero. In such a case, the corresponding *IndepVar* may have no impact on the *DepVar*. For example, a p-value of 0.05 indicates that there is a 5% probability that the relation between the variables is due to chance. In many cases, a p-value of 0.05 or 0.01 is chosen as an acceptable error level threshold.

Model Specification	**Parameter Estimates**		**p-value**	**Std Error**	**R^2**	**MMRE**
$\ln(Effort) = a + (b \times \ln(KLOC)) + (c \times \ln(TEAM))$	a	6.23	<0.0001	0.22	0.72	0.48
	b	0.39	<0.0001	0.08		
	c	0.99	<0.0001	0.13		

Table 2: Example of OLS regression model

Table 2 shows an example of a regression model (Briand et al., 1999a) derived from the analysis of Space and Military projects. The first column gives the model's specification. The second column provides the estimates for the coefficients and the intercept. The p-values are smaller than the selected threshold (0.05) and thus the variables used in the model are significant. A regression analysis determines estimates of the true coefficients based on the sample of data points available. The Standard Error of the estimated coefficients is provided in the fourth column. To assess the percentage of variance explained by the regression model the coefficient of determination is used (R^2). This value ranges from 0 to 1 and denotes the amount of variability in the data that is explained by the regression line. To evaluate the model's predictions, a standard measure used is the Mean Magnitude of Relative Error (MMRE). This is usually computed following standard evaluation processes such as cross-validation (Briand et al., 1999a).

OLS regression makes some assumptions. One of the important assumptions is that the variation in error (or residual) is on average constant on the dependent variable range. This means that the difference between the actual value and the predicted value does not change for projects. This is called the homoscedasticity assumption. Regression can only deal with interval or ratio variables, though established techniques exist to include nominal or ordinal variables. Regression models are sensitive to outlying observations in the training data set. This may cause misleading prediction equations not faithfully reflecting the trends in the data. To alleviate this problem, it is useful to identify and possibly remove outliers from the data before building an OLS regression model. Standard techniques also exist for outlier analysis. However, despite the fact that many of the difficulties discussed above are addressed in the literature on OLS regression, these techniques remain difficult to use for most engineers and managers and require extensive training and experience.

3.4.5 Robust Regression

The main concepts behind OLS regression also hold for robust regression (RR). But in order to remedy the sensitivity to outlying observations alternative minimizing methods are proposed. Instead of minimizing the sum of squares of absolute error, like in OLS regression, other methods are applied in robust regression. LMS (least median of squares), for example, minimizes the median of squares of absolute error (Rousseuw and Leroy, 1987). Other examples are the LBRS (least-squares of balanced relative errors) method minimizing the sum of squares of balanced relative error, or LIRS (least-squares of inverted balanced relative errors) that minimizes the sum of squares of inverted balanced relative error (Miyazaki et al., 1994).

Robust regression aims to be robust against outlying observations thus addressing an important issue in software engineering data. In contrast to OLS regression there is no closed-form formula that can be applied for some of the alternative algorithms (Rousseuw and Leroy, 1987). Therefore, tool support is needed and some of the methods are computationally intensive.

3.5 Examples of Composite Estimation Methods

3.5.1 COBRA – Cost Estimation Benchmarking and Risk Analysis

COBRA (COst Estimation Benchmarking and Risk Assessment) is a hybrid cost estimation method combining algorithmic and experiential approaches (Briand et. al, 1998). The core of COBRA is to develop a productivity estimation model that consists of two components. The first component is a causal model that produces a cost overhead estimate. The causal model consists of factors affecting the cost of projects within a local environment (cost-drivers). The causal model is obtained through expert knowledge acquisition (e.g., involving experienced project managers).

An example is presented in Figure 5. The arrows indicate direct relationships. The '+' indicates a positive relationship, and a '–' indicates a negative relationship. Where one arrow points to another this indicates an interaction effect. For example, an interaction exists between Project Team Capabilities and Reliability Requirements. In this case, decreased project team capability magnifies the positive relationship between reliability requirements and cost.

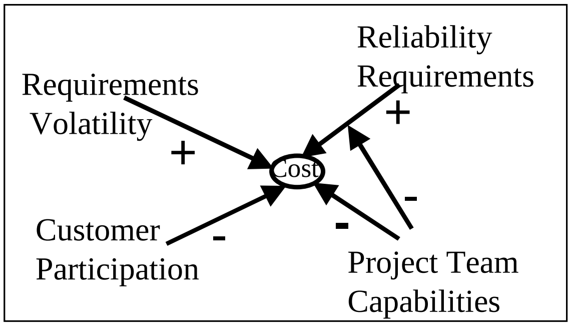

Figure 5: Example of Causal Model

The qualitative information is quantified through expert opinion elicitation and its inherent uncertainty is explicitly captured for all relationships in the causal model. The quantification is the percentage of cost overhead above that of a nominal project and is referred to as *cost overhead multiplier*. For example, if an expert gave a value of 20% overhead for a certain value of a cost-driver (e.g., "high" if cost-driver was measured on an ordinal scale, similarly to the COCOMO model), that would mean that the cost of the project would be 120% that of the nominal project.

The multipliers for the cost-drivers are modeled as distributions to capture the uncertainty inherent to expert opinion. Triangular distributions can be used (minimum, most likely, and maximum value) and reflect the experts' opinion about each cost-driver's impact on cost.

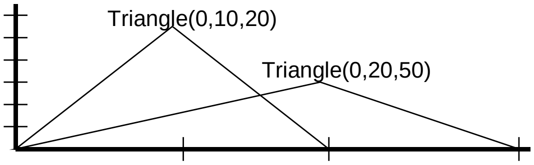

Figure 6: Triangular Distributions

Figure 6 provides examples of triangular distributions. The variance in distribution reflects the uncertainty in expert’s opinion. Specific techniques have been designed to quantify interactions but this is out of the scope of this short summary.

The second component of COBRA uses data from past similar projects identifying a relationship between cost overhead and productivity. Note that this is a simple bivariate relationship that does not require a large dataset. This is important as it explains why COBRA does not have demanding data requirements, as opposed to data-driven estimation techniques.

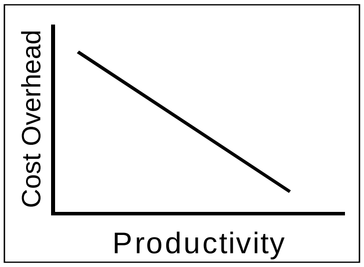

Figure 7: Example of a Cost Overhead - Productivity Relationship

Figure 7 illustrates a linear relationship between productivity and cost overhead that can be obtained from past project data. It indicates that the higher the cost overhead the lower the productivity.

Using COBRA the uncertainty of a prediction can be evaluated (for the purpose of risk assessment), a productivity or effort point estimate may be obtained for an actual project, and benchmarking can be performed of the new project against the underlying past project data. We will describe an example scenario regarding the use of COBRA for risk assessment.

To produce a cost overhead estimate for an actual project, the project is characterized in terms of the cost factors in the causal model. The actual project values are transformed into parameters of triangular distributions. Running a Monte Carlo simulation, sampling is performed from each of the distributions and each sampled value is summed obtaining a cost overhead estimate. This is repeated many times (e.g., 1000) and a distribution of cost overhead values is generated. In order to interpret the cost overhead values from the distribution, the second component of COBRA is needed. Using the cost overhead - productivity relationship, the amount of money, or effort corresponding to a cost overhead value can be determined. In order to evaluate the probability that a project will be over budget, one can generate the descending cumulative cost overhead distribution for the actual project (Figure 8).

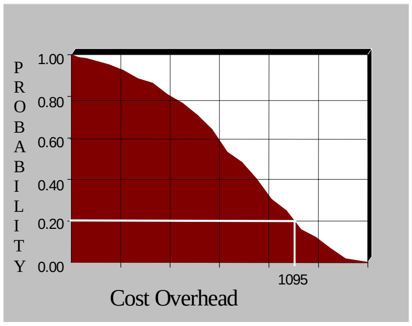

Figure 8: Cumulative Probability of Cost Overhead

Let us assume that the available budget for a project is 1100 Units and that this project’s cost overhead is characterized by the distribution in Figure 8. There is roughly a 80% probability that the project will stay within budget. If this probability represents an acceptable risk in a particular context, the project budget may be approved.

COBRA assumes orthogonal variables, which are defined on “approximately” interval scales. COBRA does not require a specific measure for system size. But to use the model, system size needs to be estimated in some way.

Some advantages of COBRA are (1) its applicability in environment with only little project data is available, (2) the explicit modeling of reusable expert knowledge regarding project estimation, (3) it is more likely to be accepted by an organization’s practitioners as the models are rooted in their own experience. The main disadvantages are that COBRA (1) requires experts to be available for interviews and (2) that knowledge elicitation is still a difficult task requiring extensive training and experience.

3.5.2 Analogy–Based Methods

Analogy is a problem solving technique (Delany et al., 1998). It solves a new problem by adapting solutions that were used to solve an old problem. One or more cases (i.e., projects) similar to current problem are retrieved and attempted to modify these to fit the current problem parameters. In software effort estimation, every case is a previous project while the current problem is to extract a suitable estimate for the current project.

The basic idea is to identify the completed projects that are the most similar (analogues) to a new project. Major decisions to be made by the user are (1) the selection of relevant project attributes (i.e., cost-drivers) (2) the selection of appropriate similarity/distance functions, and (3) the number of analogues to consider for prediction.

Similarity functions may be defined with the help of experts. As software datasets commonly contain variables of different scale types (nominal, ordinal, interval, ratio), measures were suggested to combine mixed types of variables. Kaufman and Rousseeuw proposed a dissimilarity coefficient (Kaufman and Rousseeuw, 1990). An example of a simple distance measure is based on the unweighted Euclidean distance using variables normalized between 0 and 1 (Shepperd and Schofield, 1997). The overall *distance(P_i, P_j)* between two projects P_i and P_j is defined as:

$$distance(P_i,P_j) = \sqrt{\frac{\sum_{k=1}^{n} \delta(P_{ik},P_{jk})}{n}}$$

where *n* is the number of variables. A variable could be, for example, the team size (continuous) or project environment (discrete). The distance regarding a given variable *k* between two projects P_i and P_j is $\delta(P_{ik}, P_{jk})$:

$$\delta(P_{ik},P_{jk}) = \begin{cases} \left(\dfrac{|P_{ik}-P_{jk}|}{max_k - min_k}\right)^2, & if\ k\ is\ continuous \\ 0, & if\ k\ is\ categorical\ AND\ P_{ik} = P_{jk} \\ 1, & if\ k\ is\ categorical\ AND\ P_{ik} \neq P_{jk} \end{cases}$$

where value max_k/min_k is the maximum/minimum possible value of variable *k*. If two projects have the same project environment value, the distance is 0, if their project environment is different, the corresponding distance for this variable is 1. A variation is to compute weighted Euclidean distances where each variable receives a weight according to its perceived importance.

The selection of relevant project attributes may be determined by looking at the optimal combination of variables by implementing a comprehensive search as implemented in the ANGEL tool (AnaloGy SoftwarE tooL, Bournemouth University). This is, however, inadequate for a high number of variables and projects, as reported in (Shepperd and Schofield, 1997). Another strategy is proposed by Finnie et al. (Finnie et al., 1997). They applied to all categorical variables a two-tailed t-test to determine variables that show significant influence on productivity.

Attributes	**New Project**	**Retrieved Project 1**	**Retrieved Project 2**
Project Category	Real Time	Real Time	Simulators
Programming Language	C	C	C
Team Size	10	10	9
System Size	150	200	175
Effort	**?**	1000	950
Similarity		**80%**	**77%**

Table 3: Example of Analogy based retrieval

For prediction, one may use the effort value of the most similar analogue(s). For example, in Table 3, 1000 would be the predicted value for a new project if the best analogue was used (retrieved project 1). When considering more than one analogue, simply the mean value (mean effort from retrieved project 1 and 2), or a weighted mean may be used for prediction. Adaptation rules may be defined with the help of experts to devise the final estimation for the new project. A simple example of an adaptation rule is an adjustment of the predicted effort using the size of the actual project (e.g., Walkerden and Jeffery, 1999). For example, assume the system size of the new project is 150 units, the system size of the most similar project is 200 units and its effort is 1000 Units. With the application of an adaptation rule the predicted effort is 150/200*1000=750 Units, rather than 1000 Units without applying the adaptation rule.

3.6 Evaluation of Non-Model Based Methods

Expert judgment involves the consulting of one or more experts to derive directly one resource expenditure estimate (Hughes, 1996). Experts usually use their experience and understanding of a new project and available information about the new and past projects to derive an estimate. Examples of the information used are design requirements, source code, software tools available, rules of thumb, resources available, size/complexity of the new functions, data from past projects, or feedback from past estimates. In contrast to model-based methods, the procedure to derive the final estimate (i.e., the estimation technique applied, see Section 5, Figure 9) may be more “intuitive” to some practitioners.

One approach is based on estimating as a group. This appears to have advantages such as the enthusiasm of some people may be balanced by the caution of others. However, this approach involves the risk that people with stronger personalities may dominate (Baron and Greenberg, 1990). To reduce these risks and to avoid the causes of bias of individual experts, the adoption of the Delphi technique is suggested (Boehm, 1981; Boehm, 1984; Chulani et al., 1999). First, a group coordinator presents a specification and a form to be filled out. Then, experts anonymously and independently record their estimates for a new project on the form. The group coordinator prepares a summary of the experts’ responses on a new form and requests another iteration and the rational behind the estimate. The experts again anonymously fill out the forms. The process is iterated for as many rounds as appropriate. No discussion among the experts is taking place during the whole process. One variation of the standard Delphi-Method includes group meetings, held after each iteration, and focusing on discussing large variations in the estimates. It combines the advantages of free discussion with the advantages of anonymous estimation.

Another approach is reported by Vicinanza et al. who used tape-recorded verbal protocols to analyze the reasoning process of experts. They report on different approaches used by experts, namely analogical and algorithmic approaches (Vicinanza et al., 1991). The algorithmic strategy was very similar in many respects to other methods such as COCOMO. The procedure starts with a base-productivity rate that serves as an anchor. This is then adjusted to compensate for productivity factors that impact the project effort being estimated. The analogical approach was quite different trying to understand the type of application to be developed. This was done with the help of available sizing information, like Function Points. Once the application type was determined, an analogy between the application to be estimated and a similar application previously managed was formed. The effort that the past project required became the reference for further adjustments.

There are may possible factors that affect bias and uncertainty in individual expert's opinions, such as pessimistic / optimistic opinion (Boehm, 1981), the uncertainty of judgments (DeMarco, 1982; Braun and Yaniv, 1992), the role of memory in complex problem solving (Vicinanza et al., 1991), the involvement of the estimator in the development (Lederer and Prasad, 1993), or the predictability of the environment (Hoch and Schkade, 1996). This suggests that, among other things, it is important to record the uncertainty associated with experts' estimates. Parameters such as the highest possible value, the most likely value may be estimated (Höst and Wohlin, 1998; Briand et al., 1998a). These parameters can follow several distributions. It may also be preferable to derive the estimate using several experts. If more than one expert is consulted, one needs to combine the individual estimates into one single estimate. One possibility is to compute the mean or median of all individual estimates (if point estimates are provided). This is a quick method, but may be biased by one or two extreme estimates. If distributions are provided, all individual distributions may be equally considered. Another possibility is to hold group meetings for as long as it is necessary to converge towards a single estimate. The drawback is that, for various reasons, some experts may be overly influenced by other group members.

The use of decomposition and extended problem structuring improve the tractability of difficult estimation problems and offer help in improving estimation performance (MacGregor and Lichtenstein, 1991). Subjective software cost estimation may follow a top-down approach where an overall estimate for a project is derived first and the total cost is then split among the various product components. A complementary way is the bottom-up estimation. The cost of each component is estimated and then summed to derive an estimate for the overall product. If the software job is organized in a work breakdown structure (WBS), it ensures that the costs of activities such as integration and configuration management are included.

4 Sizing Projects

It is widely accepted that among a variety of many factors that affect the cost of software projects, software size is one of the key cost drivers. The size of a software system can be measured following two main approaches: solution-oriented (source lines of code LOC) and problem-oriented (functional sizing). Expressing software size in terms of LOC creates a lot of problems. Much discussion, for example, concerns the validity of using LOC to measure the size of a system.

1. No accepted standard definition exists for LOC (Low and Jeffery 1990; Jones 1986). Jones suggests that there are eleven major variations that may be broadly split into two groups. These are concerned with program level and project level variations. Variations at the program level are: (1) count only executable lines, (2) counts executable lines plus data definitions, (3) count executable lines, data definitions, and comments, (4) count executable lines, data definitions, comments, and Job Language, (5) count lines and physical lines on an input screen, (6) counts lines are terminated by logical delimiters. Variations on the project level are important when the program is not completely new. The following variations exist: (1) count only new lines, (2) count new lines and changes lines, (3) count new lines, changed lines, and reused lines, (4) count all delivered lines plus temporary scaffold code, (5) count all delivered lines, temporary code, and support code.

2. The number of LOC depends upon the programming language used and the individual programming style (Kemerer 1992). Hence, counting variations can also affect the size of applications written in multiple languages. Thus, is it not possible to directly compare the productivity of projects developed using different languages.

3. LOC are difficult to estimate early on in the project life cycle. However, it is necessary to be able to predict the size of the final product as early and accurately as possible, if cost models based on system size are to be useful (Heemstra and Kusters 1991).

4. Finally, LOC emphasizes coding effort, which is only one part of the implementation phase of a software project (Matson et al., 1994).

To overcome these problems a promising set of measures were defined that capture the size of a software product in terms of its functional characteristics, i.e., based on the size of the problem to solve, rather than on the size of the solution to be developed for the problem. These functional size measures define elements that can be counted from early life cycle documents. The idea originates from Albrecht’s work on Function Points (FP) at IBM (Albrecht 1979; Albrecht, Gaffney 1983). The FP counts are based on system features seen by the end user. In 1982, DeMarco also published a description of a different kind of functional measures that he termed the “bang metrics” (DeMarco, 1982). These metrics are based on counts derived from data flow diagrams, entity-relationship models, and state diagrams. A frequently mentioned functional size measure is IFPUG Function Points (International Function Point Users Group 1994). It defines five function types that can be counted from early life cycle documents and then stipulates a linear weighting scheme for each of the five function types depending on their complexity. The five function types are classified as follows:

1. An external input is any elementary process of an application that processes data or control information that enters from outside the boundary of the application.

2. An external output is an elementary process of an application that generates data or control information that exits the boundary of the application.

3. An internal logical file is a user identifiable group of logically related data or control information maintained through an elementary process of the application.

4. An external interface file is a user identifiable group of logically related data or control information references by the application but maintained within the boundary of a different application.

5. An external inquiry is an elementary process of the application that is made of an input-output combination that results in data retrieval. The input side is the control information that defined the request for data. The output side contains no derived data.

The Function Point (FP) count can be adjusted for technical processing complexities. Although FP overcomes some of the problems associated with LOC, it has some other limitations.

1. The rational for the IFPUG weighting scheme is not clear and it is also of concern that using the complexity weights does not necessarily explain more variation in effort compared to non-weighted function points (Jeffery and Stathis 1996). Therefore, some practitioners do not apply the weighting scheme or devise their own.

2. The structure of the measure is such that the function types are not orthogonal. And, there is evidence that not all function types are required for building an effort estimation model (Jeffery and Stathis, 1996; Kitchenham and Känsälä 1993).

3. FP apply mainly to Management Information Systems. Hence, the five function types can be derived from entity-relationship diagrams and data flow diagrams. However, in the context of real-time systems and scientific software, additional elements are important such as algorithms, or the number of state transitions.

To address some of these shortcomings, additional types of functional size measures were developed. To mention a few: 3D Function Points (Whitmire, 1998) extend function points to measure attributes from three dimensions: data, function, and control and can be applied to object-oriented software. An Object Points analysis was proposed for Integrated CASE Environments (Banker et al. 1992). The weighting for each type of object was determined in Delphi estimation sessions. Feature Points (Jones 1997) were proposed as a superset of the FP metric introducing a new parameter, algorithms, in addition to the five standard function types. This metric was designed to work equally well with MIS applications, and other types of software such as real-time software or embedded software. The Full Function Points (Abran et al. 1997) was proposed to generalize functional size measurement to real-time systems. It essentially provides additional dimensions to the five original FP dimensions. These new dimensions look at the complexity of communicating processes and their synchronization. The fundamental principles on which the FP model is built remain the same though.

In some cases it is useful that LOC information can be “backfired” to FP counts and vice versa (Jones 1997). LOC counts can be obtained by multiplying the FP count by a source-language expansion factor. This factor is obtained from empirical observations of the relationships between various source languages and FP counts. However, the language expansion factors may vary. Main causes of variation are explained by variations in programming skills and the fact that FP involve subjective assessments so that different counts may be obtained by different analysts.

Despite the well known functional sizing approaches, practitioners tend to produce ad-hoc size estimates based on expert knowledge. Basic types of sizing where experts are involved are (Kitchenham and de Neumann 1990):

1. Estimation-refinement methods. These methods involve the combination of several independent estimates from individual experts (Boehm 1981). Alternatively, estimation refinement may consist of relative size ranking, which involves ranking software components into a size order relative to some reference components (Bozoki). Finally, size can be

estimated in terms of ranges, usually including a most likely estimate and an upper and lower bound (Boehm 1981).

2. Estimation by structural decomposition, which involves decomposing a system into its basic components that are assumed to be of similar size. An estimate is derived by multiplying the number of components by the estimate of the unit size. Usually, this method requires the combination with another method in order to estimate the size of the unit size (Britcher and Gaffney, 1985).

3. Sizing by analogy, which derives a size estimate for a new product from comparing it to previously developed systems with similar functions and requirements. This type of sizing approach depends upon a historical database that consists of descriptions of previously developed software functions. A new product is sized by determining the similarities and differences between the database entries and the new product. Two subclasses of this type of sizing exist: (1) the comparison of application functions is based on identifying software of similar application type and function (Putnam 1987), (2) the comparison of project attributes is based on the comparison of cost-driver like attributes, such as complexity, peak staffing level, or requirements volatility. The basic concept of the project attribute comparison approach was developed by Saaty 1996 and was applied to software sizing by Lambert (Lambert, 1986).

From a general perspective, it is clear that sizing is one of the key issues to address to devise accurate, cost-effective ways of predicting resource expenditures. Though may pertinent ideas have been reported in the literature, no existing solution can claim to provide a universal answer. FP-based methods may be too expensive and complex to apply in some environments. Their weighting and underlying assumptions may turn out not to fit reality well enough in the context of application to provide accurate answers. Approaches based on inter-project comparisons are promising but need to be further investigated in order to empirically determine their potential accuracy. Especially for projects that are enhancements of existing projects, we need better ways of assessing the amount of change. In fact, from a practical standpoint, an organization needs to determine, based on its lifecycle documentation (requirements, specifications), what information can be extracted, possibly automatically, in order to facilitate project sizing. We believe that an empirical but rigorous approach, based on analyzing local project data and identifying key size dimensions, is likely to yield more adequate and accurate results (Mukhopadhyay and Kekre, 1992).

5 Framework for Comparison and Evaluation

In order to make the comparison and discussion of resource estimation methods systematic and structured, we need a comparison framework. That is what we define here in this section. The first subsection provides more details on our motivations and objectives. Section 5.2 attempts to generalize all estimation methods into a common set of concepts and structure. Section 5.3 defines a set of evaluation criteria aimed at helping practitioners assess alternative resource estimation methods. Guidelines for the evaluation of these criteria are then provided in Section 5.4. Though a detailed evaluation of the estimation methods presented in Section 6 will be provided in Section 6, Section 5.5 discusses the advantages and drawbacks of different categories of methods, from an analytical viewpoint. Because estimation methods cannot be entirely evaluated by analytical means, empirical studies are required if we want to gain a sufficient understanding of their respective predictive capabilities. But empirical studies need to be designed and their resulting data analyzed. This is covered in Section 5.6.

5.1 Rationale and Motivations

One common problem is the resource estimation literature is the lack of standard terminology for discussing estimation methods. In the next section, we define important concepts and terms that will be used throughout this paper. Furthermore, by defining the terminology precisely, we will lay down a structure that will help us compare estimation methods and their categories (see Figure 1, Section 3). This is important as all estimation methods have differences but also numerous commonalities.

We will then identify a number of evaluation criteria that will then be used to compare categories of estimation methods using our terminology and comparison structure. Last, because estimation techniques have also to be evaluated through empirical means (e.g., their predictive accuracy), we will discuss a number of important points regarding the design of empirical studies on resource estimation. This will also be referred to in Section 6.2 when reporting and discussing existing empirical results.

5.2 Elements of Resource Estimation Methods

We distinguish among the following terms and concepts: modeling technique, modeling method, model, application method, estimation techniques, estimate, and estimation method. These terms are used inconsistently across the literature and are usually a source of confusion. Though the result of subjective choices, the definitions we provide below will help bring clarity and structure into our discussions.

- **Modeling technique:** A procedure to construct a model from available data and/or expert knowledge. More than one modeling techniques may be applied to derive one particular estimation model. For example, COCOMO II (Chulani et al., 1999) derives a model using linear regression analysis and Bayesian inference, two distinct modeling techniques.

- **Modeling method:** One or more modeling techniques are used together with guidelines on how to appropriately apply them together to build a model for a given purpose, i.e., estimating software project cost. For example, consider COCOMO II, using the two techniques mentioned above, to produce a complete effort predictive model. Bayesian inference is applied to refine COCOMO cost multipliers, originally based on expert opinion, by using project data (Chulani et al., 1999).

- **Model:** This is an unambiguous, reusable representation of the relationship between resource expenditures (e.g., development effort, cost, productivity) and its most important driving factors (commonly referred to as cost-drivers). The COCOMO II model (Boehm et al., 1995) is a good example of a development effort estimation model.

- **Model application method:** A procedure specifying how to obtain an estimate by applying one or several models in a specific context. For example, one may use several modeling techniques, yielding distinct models. All models may then be used to produce distinct estimates and, if these estimates are not consistent, a specified procedure is then used to determine the most plausible estimate or a (weighted) average that will be retained as the final estimate. Models can be applied for different purposes, such as effort prediction, risk analysis, or productivity benchmarking. Application methods define how models and estimates are used to achieve these objectives. Such procedures are referred to as *model application methods*.

- **Estimation technique:** A procedure to devise a resource expenditure estimate directly from available expert knowledge and/or project data. No model building is involved. The Delphi technique (Boehm, 1981) is an example of estimation technique based on expert opinion.

- **Estimate:** A concrete value or distribution that reflects the most knowledgeable statement that can be made at a point in time about the resources needed for a future or ongoing project.

- **Estimation Method:** It consists of either (1) one or several models, possibly a modeling method and a model together with a method to apply the model(s), or (2) one or more estimation techniques. This distinction is made because some estimation methods do not include any explicit resource modeling. The estimates are solely based on expert judgment.

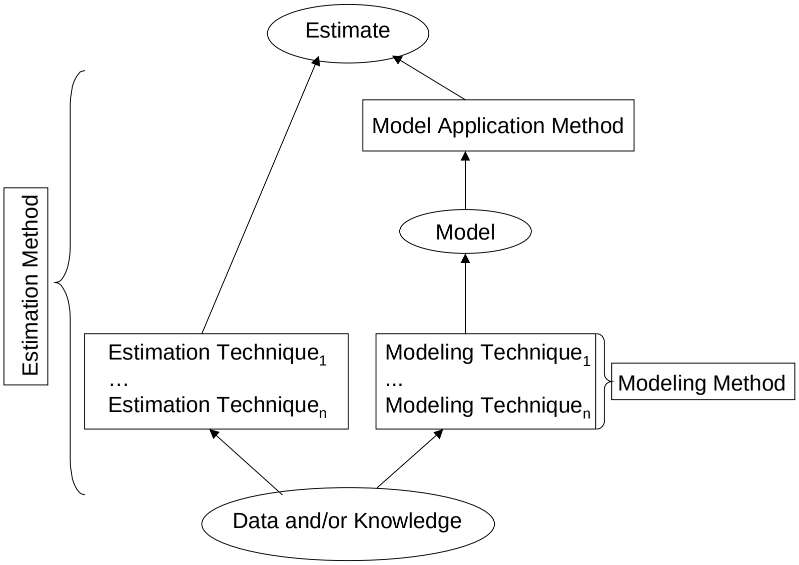

Figure 9: Resource Estimation Terminology and Relationships

Figure 9 illustrates the definitions above and their relationships. It summarizes the various ways a resource expenditure estimate can be obtained. A specific path in Figure 9, coupled with a specific selection of techniques and possibly following an appropriate modeling method, forms an estimation method. Given some resource related data and/or expert knowledge, either an estimation technique or one/several modeling technique/s is/are applied. When modeling is selected as a basis to obtain estimates, the application of one or several modeling techniques (i.e., following a modeling method) leads to an estimation model. To obtain an estimate for a certain purpose , an application method describes how to use the estimation model and derive the estimate. Following the left path in Figure 9, where one or several estimation techniques are applied, leads directly to an estimate. As no explicit model is produced, no model application method is required. In Section 5.5, we will discuss the drawbacks and advantages of selecting a path over the other. It is also worth noting that, for proprietary methods and tools, the modeling techniques, methods, and even the models themselves, are not public domain knowledge. However, application methods are sometimes suggested by the vendors.

5.3 Evaluation Criteria

In most of the software resource estimation literature, evaluation criteria have focused on the predictive capability of estimation models. However, when evaluating estimation methods and their components, many other criteria are relevant to make decisions regarding the adoption or development of an estimation method. Some of these criteria are subjective in nature but are nonetheless important. Practitioners should consider them explicitly and systematically when comparing resource estimation methods.

The evaluation criteria defined in this study are based on existing studies (e.g., Kitchenham, 1990; Andolfi et al. 1991; MacDonell, 1994) and our experience with resource estimation methods. It does not claim to be complete but it has served us well in the past. We do not go as far as to capture crisp criteria that can be assigned a score. But our definitions will help structure our discussion and evaluation of existing resource estimation models and methods. Table 4 gives an overview of three categories of evaluation criteria. The first category is related to criteria for assessing models and their estimates (Model and Estimate Criteria). The second category of criteria deals with the process to build the final estimate (Estimation Method Criteria). The third group considers the applications of an estimation method (Application Criteria) to obtain a final resource expenditure estimate. As we will see (Section 7), the relative weight of some these criteria will be context dependent. For example, across environments, project managers may feel more or less comfortable dealing with the complexity of certain statistical tools.

Some of the criteria are not applicable in all circumstances. For example, the “Inputs required” criterion is only applicable to generic models (e.g., COCOMO) where input variables (cost drivers) are fixed and meant to be generic to accommodate various environments. The letter in brackets specifies the level in the hierarchy of cost estimation methods (Figure 1, Section 3.1) for which a criterion should be considered. For example, “A” indicates that the criterion is only applicable to model-based methods. If no letter is given, the criterion is applicable for all defined types of models.

Model and Estimate Criteria	**Estimation Method Criteria**	**Application Criteria**
Quality of model and estimate Inputs required (A1) Completeness (A1) Type of estimates (A1) Calibration (A1) Interpretability (A)	Assumptions Repeatability Complexity Automation (Modeling) (A) Transparency	Application Coverage (A) Generalizability Comprehensiveness (A1) Availability of estimates (A1) Automation (Method Usage)

Table 4: Evaluation Criteria for Cost Estimation Methods

- **Model and Estimate Criteria:** The category consists of criteria that are important when trying to assess an estimation model or technique, and its estimates (Table 5). Important questions with this respect may be the following. How good is the model’s predictive accuracy? What are the required inputs? What kinds of resource expenditures are estimated? What kinds of estimates are produced? Can the model be calibrated? Is the model interpretable by software practitioners?

Model and Estimate Criteria	**Description**
Quality of model and estimate	An important criterion regarding the quality of an estimation model and estimate , or its estimates, is predictive accuracy. This compares the predicted resource expenditures with actual values, e.g., in terms of the relative error. The higher the quality of an estimate, the less the risk associated with an estimate, the more likely is an estimation method to be accepted by practitioners. Detailed quantitative criteria are proposed in the next section. This criterion is often considered the most important as models or techniques have to be sufficiently accurate to be even considered as an alternative for resource estimation.
Input variables required	This category considers the kind of inputs required to develop and use a model. For generic models, we will consider whether it is possible to tailor inputs to a particular environment and the extent to which they can be objectively assessed in terms of their contribution to estimating resource expenditures. For example, COCOMO II (Boehm et al., 1995) proposes a set of up to 17 cost-drivers (or model input variables) that one has to estimate and use to produce an effort estimate.
Completeness of estimates:	This category evaluates a model's capability to provide estimates for different project resource expenditures like, effort, cost, or duration. During project planning, support is needed for all these types of resources. However, effort estimation has been the focus of most research, as it is believed that cost and duration can then be derived from effort estimates.
Type of estimates:	This category assesses the different possible types of estimates that a model can provide, like point estimates, interval estimates, or probability distributions. In general, the uncertainty of a resource expenditure estimate should be modeled. This is particularly important in software engineering where decisions are made within the context of large uncertainties and based on risk analysis. For example, it can be a range of values that has a given probability (e.g., 0.95) of including the actual cost (Kitchenham and de Neumann, 1990).
Calibration:	This category captures the extent to which a model can be calibrated based on project data and to which extent calibration is clearly supported by a modeling method. Usually, generic models such as COCOMO need to be calibrated to different environments (Kemerer 1987; Jeffery and Low, 1990). But proprietary models and tools do not always provide such a capability in a satisfactory form.
Interpretability:	This category specifies the extent to which a model is easy to interpret by a software engineering practitioner, (e.g., project manager). A model that consists of a multivariate regression equation, for example, may not be very easy to interpret, and thus might not be accepted by practitioners in certain contexts. It is often the case that practitioners want to understand how an estimate was obtained before relying on it. Those human factors play a very important role in the adoption of a resource estimation method.

Table 5: Description of Model and Estimate Criteria

- **Estimation Method Criteria:** This category deals with the estimation method as a whole, producing a final estimate, possibly based on a combination of estimation models and techniques (Table 6). The main questions covered by this category are the following. What are the main underlying assumptions made by the underlying estimation model(s) and how realistic are they? How repeatable is the process to derive an estimate? How complex is the method to apply? To which extent is the model development supported by a tool? , How transparent is the estimation method? Several of these questions are interrelated but still capture different aspects.

Estimation Method Criteria	Description
Assumptions	This criterion assesses how realistic are the underlying assumptions of the estimation model(s) in a given context. The more unrealistic the assumptions, the more risky the application of an estimation method.
Repeatability	The repeatability of an estimation method captures the extent to which the steps to use models and techniques, combine their results, and obtain a final estimate, are clearly defined. The better defined and specified an estimation method, the more independent the estimate from any specific human estimator.
Complexity	This characterizes the cognitive complexity of the steps that are required to generate an estimate. The more complex an estimation method, the higher the effort invested into estimates, the more error-prone, the less likely to be adopted by practitioners.
Automation of Modeling	This criterion captures the extent of tool support that is available to apply a modeling method in order to derive estimation models. The effort to derive models and evaluate them is drastically reduced when effective tool support is available.
Transparency	This assesses the extent to which the algorithms and heuristics of the estimation method are documented and justified by a clear rationale. This is different from repeatability as the estimation process may be well defined but proprietary and invisible to the estimator. This criterion mostly applies to proprietary estimation methods embedded into commercial tools.

Table 6: Description of Estimation Method Criteria

- **Application Criteria:** This category focuses on the applicability of a model with the help of a model application method (Table 7). The main related questions are the following. For which purposes can an estimation model be applied? In which context can the estimation method be applied and was applied to date? For what purposes can the model be applied? How early in the lifecycle can a model be applied?

Application Criteria	Description
Application Coverage	This category evaluates the extent of possible applications of a model. Questions addressed here are the following: Can the provided models be used for prediction, benchmarking, and/or risk-assessment? Can usage scenarios be readily identified for these purposes?
Generalizability	This assesses the extent to which an estimation method is applicable across development environments. This depends on the conceptual assumptions underlying the estimation methods and its underlying models (if any) and may be supported by empirical evidence reported in existing studies.
Comprehensiveness	This tells how fine grained an estimate can be (e.g., effort predicted at the project, phase, or activity level), and what project activities can be included into an estimate (e.g., administrative and management overhead activities).
Availability of Estimates	This category captures the applicability of an estimation method during the various stages of software development. This mainly depends on the availability of the inputs required by the estimation model(s) or techniques to obtain an estimate. For example, COCOMO II provides ways to obtain estimates at different stages of a project, each requiring different sets of input variables are used for subsequent stages (Boehm et al. 1995).
Automation of Method Usage	For a given estimation method, this criterion captures the extent to which the derivation of a final estimate for different purposes such as prediction, risk analysis, and benchmarking are supported by tools.

Table 7: Description of Application Criteria

5.4 Assessment of Evaluation Criteria

This section discusses how the above-mentioned evaluation criteria can be assessed. All the criteria but one are inherently subjective. We provide guidelines by characterizing the extreme situations corresponding to these criteria (Table 8, Table 9, Table 10). If one wishes to assign scores based on these criteria, they can be measured, for example, on a four-point subjective measurement scale (Spector, 1992). Measurement, whether subjective or objective, helps us be more systematic in assessing evaluation criteria and provide a way to summarize results into a concise table. We will use such subjective measurement to perform a subjective scoring of estimation methods in Section 6.

- **Model and Estimate Criteria**

Model and Estimate Criteria	**Description of Assessment**
Quality of model and Estimate	The most commonly used measure is the *mean magnitude of relative error* (MMRE). Conte et al. (Conte et al.1986) consider $MMRE \leq 0.25$ as acceptable for effort prediction models. In many studies, the prediction level (Pred(l)) is also given, looking at the proportion of observations for which the *MRE* is lower of equal to *l*. Conte et al. recommend an acceptable criterion for an effort prediction model to be $Pred(0.25) \geq 0.75$. (But this is inherently subjective.) If *Pred(0.25)* would be equal to 1, for all of the observations the MRE values would be lower or equal to 0.25. Different levels *l* are used across published studies. MRE is defined as follows: $MRE_i = \frac{\lvert ActualValue_i - PredictedValue_i \rvert}{ActualValue_i}$ The *MRE* value is calculated for each observation *i* whose value is predicted. It has also been proposed to use the predicted value as a denominator, instead of the actual value. The argument is that estimators are interested in the error relative to their estimate, since this is what will be used to devise project plans. The *MRE* over multiple observations, say *N*, can be characterized by the *Mean MRE* (MMRE): $MMRE = \frac{1}{N} {}_{i} \frac{\lvert ActualValue_i - PredictedValue_i \rvert}{ActualValue_i}$ Besides the mean, looking at other aspects of the *MRE* distribution (e.g., median, quartiles) may also be of interest. The *prediction level* is defined as: $PRED(l) = \frac{k}{N}$ where *k* is the number of observations for which the MRE is lower or equal to *l*. All the measures mentioned above are equally valid. They just provide different insights into the accuracy of models. Which one may be more appropriate is context dependent. Our evaluation in Section 6 summarizes the results and trends on the quality of model estimates, for a variety of estimation methods reported in the literature.
Inputs required:	*Low:* The estimation model requires only a limited number of well-defined, orthogonal inputs. Thus the model is less expensive and complex to use. Also, the model does not depend on a restrictive definition of system size that makes it inconvenient to use in a significant number of environments (e.g., based on a very specific notation such as Petri-nets) and at different stages of development (e.g., SLOC). *High:* A number of difficult-to-estimate inputs are required to enable estimation. The model is dependent on a restrictive subset of size measures.
Completeness of estimates	*High*: The estimation method provides estimates for all required types of project resources, e.g., effort, cost, duration of project *Low*: The estimation method only provides a restrictive and insufficient set of estimates, e.g., effort estimate only, and no support for other types of resources.

Model and Estimate Criteria	**Description of Assessment**
Type of estimates	*Complete*: A probability distribution of the estimated resource is provided; i.e. the range of the possible values together with their probabilities, thus capturing the uncertainty resulting from estimation. *Poor*: Only a point estimate is provided by the model; i.e., usually the most likely or mean value.
Calibration	*High*: Calibration is supported through, for example, adaptation rules (e.g., analogy) or the computation of empirical factors to adjust model estimates. *Low*: No calibration supported. The model estimates have to be used as is.
Interpretability:	*High*: The model's form is easy to interpret, e.g., simple decision rules in a decision tree. *Low*: The model can only be interpreted by experts in the estimation modeling technique(s), e.g., interpretation of regression coefficients with interacting independent variables.

Table 8: Assessment of Model and Estimate Criteria

- **Estimation Method Criteria**

Estimation Method Criteria	**Description of Assessment**
Assumptions	*Risky*: Unrealistic assumptions must be fulfilled to apply a modeling method. These assumptions may concern the underlying modeling techniques (e.g. assume a functional relationship between variables, distributions), the required knowledge to use the method (e.g., extended expert knowledge), or data requirements (i.e., large amount of data required). *Harmless*: Few, realistic assumptions.
Repeatability	*High*: A precisely defined and documented, well-supported (tools, guidelines) estimation method. In other words, the estimate is independent of the estimator as the method is precisely and operationally described. *Low:* The estimate depends on the estimator's knowledge and experience. Few guidelines are provided on how to apply the estimation method and, if applicable, to build its underlying models. The accuracy of the final estimate is very likely to depend heavily on the estimator's expertise.
Complexity	*High:* Applying the estimation method is complex. More than one modeling or estimation technique may have to be applied. They may involve many time consuming, technically difficult steps. The combined use of modeling techniques may involve complex interfaces and interactions between them. *Low*: The models or techniques are easy to build and apply. No more than a few days' training is necessary.
Automation of Modeling	*High*: The steps involved in the estimation method are fully automated, i.e., model construction and calibration activities. *Low*: No tool support at all or tool support involves complex decisions and procedures (e.g., parameters with no intuitive setting procedures).
Transparency	*High:* The estimation method is well defined and fully justified. The algorithms involved are clearly specified. *Low*: The estimation method is proprietary and not available. In this case, by definition, it is also not justified.

Table 9: Assessment of Estimation Method Criteria

- **Application Criteria**

Application Criteria	**Description of Assessment**
Application Coverage	*High*: The estimation method supports a complete, integrated set of applications: effort and schedule predictions, productivity benchmarking, risk analysis. *Low*: The Estimation method is limited in terms of the types of resource management applications it supports, e.g., point estimates of effort only.
Generalizability	*High*: The estimation method is applicable to a broad range of contexts, application domains, life cycles. *Limited*: The estimation method is specifically aimed at an application domain, environment, or life cycle. It may, for example, rely on a project size measure requiring a specific requirements engineering method.
Comprehensiveness	*High*: The estimation method encompasses all project activities and provides estimates at the needed level of granularity, e.g., phase, activity level. *Low:* Estimates are only performed at a rough level of granularity (e.g., entire project), and does not include all required project activities.
Availability of Estimates	*Early*: The method and model can be applied at the proposal phase, or at least after the definition of the requirements, though it is typically not be very precise at such an early point. *Late*: The method cannot be appropriately applied before the later phases of development, e.g., detailed design phase, thus limiting its applicability.
Automation of Method Usage	*High*: The steps involved in applying the estimation method for the purpose of estimation, risk analysis, and benchmarking are fully automated. *Low*: No tool support to perform resource estimation, risk analysis, or benchmarking.

Table 10: Assessment of Application Criteria

5.5 Main Variations across Methods and their Consequences

The evaluation of the criteria presented above is partly context dependent. For example, whether an estimation method is complex depends on the training and background of the estimators. However, some general trends, that will be refined in Section 6, can still be identified and are presented below.

It is not possible, based on analytical reasoning, to presume of the predictive accuracy (*quality of models and estimates*) of the different estimation methods and techniques presented in Section 3. Only empirical studies, based on simulation (Pickard et al., 1999) or empirical data (Briand et. al, 2000) can help us determine what modeling and estimation techniques are likely to be adequate in the context of software engineering planning.

Due to various human factors, techniques based on expert opinion will tend to be less *repeatable*. Though many techniques have been developed over the years to help expert elicitation (Meyer and Booker 1991), the cognitive process of estimating software resources is complex and still not understood. Also, due to the fact that such techniques are contingent to having experts spend time on the project planning process, these estimates may not be needed (*availability*) when they are required. Also, modeling uncertainty in the context of expert knowledge elicitation is a notably difficult problem (Meyer and Booker 1991). This makes it difficult to perform risk analysis based on estimation uncertainty.

Data driven techniques based on statistics are usually difficult to handle by most practitioners. Worse, their results are difficult to *interpret*, thus forcing the estimator to rely on prediction results as a black box. Stensrud and Myrtveit (Stensrud and Myrtveit, 1999a) report that, despite a clear improvement imputable to the use of regression models, the participants of their experiment were still not convinced, by the end of the exercise, of the usefulness of such models. Various psychological obstacles are usually a significant problem to overcome when introducing statistical techniques. In addition, data driven techniques, by definition, require a large amount of (relevant, recent) data that are rarely available in most organizations. This is one reason for which several instances of multi-organization project databases have been attempted and are discussed across the literature (STTF).

Certain non-statistical, data driven modeling techniques such as analogy and regression trees have triggered the curiosity of researchers and practitioners alike because of their simplicity and the intuitive form of their output. However, several studies have also shown that they could repeatedly perform poorly as the modeling mechanisms on which they rely may be too rough to capture the subtleties of the structures underlying the data (*quality of model and estimate*).

Section 6 will use the evaluation criteria presented above to discuss the estimation methods and techniques presented in Section 3.

5.6 Empirical Studies on the Quality of Estimates

The problem with estimation methods is that their capability to predict accurately depends on many factors that are specific to every estimation context. For data driven techniques, the distributions of the data, the form of the underlying relationship between resource expenditures and cost drivers, and the form itself of the cost drivers (e.g., categorical, continuous) play an important role in determining the capability of a modeling technique to fit data properly. For example, it is well known that regression trees work satisfactorily in a context where data are categorical or ordinal, cost drivers interact, and the relationship with interval scale cost driver is strongly non-linear (Salford Systems).

Empirical studies of data driven techniques are either based on actual project databases (Briand et al., 1999b; Briand et al., 2000) or simulation studies (Pickard et al., 1999). The latter ones are few because of the inherent difficulties they present. To generate representative simulation samples, it is important to make decisions regarding the distributions of data, the form of relationships between cost drivers and resource expenditures (e.g., interactions, linearity), and the size of the data set. However, it is, in our context, difficult to a priori decide about such settings. Making vary such parameters across the range would require very complex simulation studies and would be inevitably incomplete.

Studies based on actual data sets present the problem that they may not be representative. This is especially true if the data come from one or a few organizations and the data set is small. In this case, some modeling techniques may turn out to be accurate out of sheer luck. This result may not be representative of most situations. However, a couple of studies have been based on multi-organization databases and have shown consistent results. If such results were to be confirmed statistical techniques would then show to be rather optimal, or at least not worse than other types of techniques.

There are very few studies based partly or entirely on expert opinion (Vicinanza et al. 1991; Höst and Wohlin, 1998; Myrtveit and Stensrud, 1999a). One advantage though is that other fields have developed a body of knowledge regarding the elicitation and use of expert knowledge (Meyer and Booker 1991). These techniques seem, a priori, largely reusable as they are not application domain specific. The problem in such studies is that the results are likely to depend on the "experts" selected, their level of expertise, the type of estimates they are asked to perform, and the elicitation technique that is followed. How can we characterize expert profiles to be able to explain differences across studies? How to report elicitation procedures so that, once again, variations across results may be more easily explained? These questions currently remain in the realm of research but require to be investigated in industrial context, under realistic conditions.

A last but interesting point made by (Stensrud and Myrtveit, 1999a) is that it may be more realistic to study the impact of data-driven modeling techniques in the context where they are used in combination with expert opinion. The argument is that those are the realistic conditions under which we can expect our modeling techniques to be used. These conditions, it is argued, can have dramatic effects on the results. This highlights the argument that, if we want to assess, in a realistic fashion, alternative estimation methods, we need to think more thoroughly of acceptable case study designs. The external validity of our results depends to a large extent of the conditions under which we experiment with estimation methods.

6 Evaluation and Comparison of Effort Estimation Methods

6.1 Overview of Empirical Studies

For the last twenty years, a large number of studies were published comparing different modeling and estimation techniques. They focused mostly on software *effort* estimation. This section gives an overview of existing empirical studies and attempts to identify general patterns in the results[3]. This overview will be useful in Section 6.4 when we will perform a systematic evaluation of different modeling techniques and estimation methods using our evaluation framework (Section 5). As previously discussed, the *quality of model and estimate* of estimation methods can only be assessed empirically.

Table 11 summarizes the most relevant empirical studies from the 80's and 90's. In cases where authors published several papers about similar work in conferences and journals, Table 11 considers only the most relevant journal publication. Studies are listed in chronological order. For each study, estimation methods are ranked according to their performance. A "1" indicates the best model, "2" the second best, and so on. Tied ranks are used when estimation methods show comparable estimation accuracy. We focus here only on most commonly used estimation methods. Generic models, like COCOMO or SLIM, were often compared in the 80's, whereas people started to include methods based on machine learning algorithms and analogy in the 90's. Regression methods are used in most of the studies. COCOMO is the next most frequently compared method. This suggests we should consider regression analysis and COCOMO as baselines for comparison.

[3] Note that we only consider studies where different types of methods are compared. For example, we did not consider studies that compared different versions of regression.

		Regression[4]	COCOMO	Analogy	SLIM	CART	ANN	Stepwise ANOVA	OSR	Jensen Method	Expert Judgement[5]	Other Methods
1.	Kitchenham, Taylor 1985	1	2		3							
2.	Conte et al. 1986		2		4					3		1[6]
3.	Kemerer 1987	2	3		4							1[7]
4.	Jeffery, Low 1990		2									1[8]
5.	Navlakha 1990		2		1					2		2[9]
6.	Vicinanza et al. 1991	2	3								1	
7.	Briand et al. 1992	2	3						1			
8.	Mukhopadhyay et al. 1992	3	4	2							1	
9.	Mukhopadhyay, Kerke 1992	1-3[10]	2									
10.	Subramanian, Breslawski 1993	1	2									
11.	Bisio, Malabocchia 1995		2	1								
12.	Srinivasan, Fischer 1995	2	4		5	3	1					
13.	Jorgensen 1995	1					2		1			
14.	Chatzuglu, Macaulay 1996	2	3									1[11]
15.	Shepperd, Schofield 1997	2		1								
16.	Finnie et al.1997	2		1			1					
17.	Briand et al.1999b	1		2		1		1				2[12]
18.	Briand et al. 2000	1		2		2		1				2
19.	Dolado, Fernandez 1998	1					2					1[13]
20.	Kitchenham 1998					2		1				
21.	Myrtveit, Stensrud, 1999a	2		3							1-3[14]	
22.	Hughes et al. 1998	2		1			3					3[15]
23.	Walkerden, Jeffery 1999	2		1							1	
24.	Gray and Mac Donell	1					2					1[16]

Table 11: Overview of Empirical Studies

[4] Regression includes OLS-regression using KLOC, FP, Object-Points, etc. to measure system size.

[5] Expert judgement is always supported by some method, data or a tool.

[6] COPMO Cooperative Programming Model

[7] ESTIMACS proprietary FP-based method

[8] CLAIR estimation package, four effort estimation models are derived from regression studies of 146 Australian MIS software projects

[9] Doty model and (static single-variable model; estimated dependent on number of delivered source instructions), IBM model (the same as Doty model but customizes to different environment)

[10] Regression with empirical Feature Point based approach performed best, regression methods with other size measures also tested such as LOC- or FP-based

[11] MARCS Management of the Requirements Capture Stage

[12] Combinations of CART with regression, CART with analogy

[13] GP, genetic programming

[14] Different combinations subject+history+regression, subject+history+analogy better than subject+history only. GP generates a solution by going through a selection of candidate equations and eliminating or re-generating those equations that do not meet certain criteria.

[15] Rule-Induction procedure based on the ID3 algorithm (Quinlan 1986)

[16] Different variants of Robust Regression

6.2 Short Descriptions of Empirical Studies

1. (Kitchenham and Taylor, 1985) used 33 projects that were system utilities written in high-level languages. Size was measured in non-commented lines of code. They compared the SLIM model, the COCOMO model and regression models. Main conclusions were that the SLIM model was not suitable for this environment. The COCOMO model cannot be used indiscriminately in any environment, because the specific form of the cost relationships was inappropriate. Regression models tailored to the environment improved the precision.

2. (Conte et al., 1986) used six data sets from widely differing environments and reported an MMRE variation between 0.70 and 0.90 for their four tested models: SLIM, COCOMO, SOFTCOST, and Jensen's model. As a result of their investigation, they proposed a new model COPMO calibrated separately to the six data sets. This model yielded a MMRE of 0.21.

3. (Kemerer, 1987) used 15 projects from business applications, mainly written in COBOL. He compared SLIM, COCOMO, ESTIMACS, and Function Points (FP). He reported that methods that are not based on KLOC as a size measure (FP and ESTIMACS) performed better than KLOC-based methods (SLIM, COCOMO). But nevertheless the accuracy of all applied methods was far from being satisfactory. The estimation error in terms of the Mean Magnitude of Relative Error (MMRE) varied between 0.87 and 7.72.

4. (Jeffery and Low, 1990) used data from 122 MIS projects from six different organizations covering banking, finance and insurance domains. System size was provided in SLOC and FP. They compared the COCOMO model with SLOC-based and FP-based regression models and found that the regression models performed much better than the uncalibrated COCOMO model.

5. (Navlakha, 1990) performed a case study where he applied the COCOMO model, SLIM, Jensen's Model, the Doty model, and the IBM model (same as Doty model, but calibrated to different environment) to seven projects from different organizations. They found that in some cases the cost model used by organizations is not ideal for their environment.

6. (Vicinanza et al., 1991) compared COCOMO, FP-based regression models, and expert performance and showed that managers made more accurate estimates than an COCOMO model. The underlying data set was a subset of the Kemerer data set mentioned above.

7. Using a combination of the COCOMO and Kemerer data sets, (Briand et al., 1992) compared the COCOMO model, stepwise regression, and optimized set reduction (OSR). OSR outperformed stepwise regression, which in turn outperformed the COCOMO model.

8. (Mukhopadhyay et al., 1992) compared the performance of their CBR-tool Estor, with Expert Judgement, FP, and COCOMO using Kemerer's data set. Estor was developed based on verbal protocols of experts estimating 10 of the 15 Kemerer projects. It was found that experts outperformed Estor, which in turn outperformed the FP model and COCOMO.

9. (Mukhopadhyay and Kerke, 1992) adapted existing cost models using feature points to measure system size early on in the development process. Field data from the manufacturing automation domain was used for validation (58 projects). They adapted COCOMO, and FP-based models and

developed their own application feature-based models. The new developed models outperform significantly the adapted FP and COCOMO models[17].

10. (Subramanian and Breslawski, 1993) compared the COCOMO model with two regression models. They used two different types of factor analysis [ref] to reduce the number of influential factors on cost. They found that the two regression models performed better in terms of MMRE than the COCOMO model. They used the COCOMO data and subdivided a "sizeable" proportion as a training set and a small proportion was randomly selected for testing.

11. (Bisio and Malabocchia, 1995) used their developed tool FACE (Finding Analogies for Cost Estimation) on the COCOMO database. They applied different tests with variations to the retrieval function and applied them together with an adaptation function. Applying these Analogy-based techniques using the FACE tool shows better performance compared to the COCOMO model.

12. (Srinivasan and Fisher, 1995) include five different methods in their comparison: regression trees, artificial neural networks, FP, COCOMO, and SLIM. They used the COCOMO database as a training set (i.e., to develop a model) and tested the results on the Kemerer data (15 projects, mainly business applications). The artificial neural networks outperformed a FP-based model, which in turn, outperformed regression trees. These, in turn, performed better than the COCOMO model that outperformed the SLIM method. The MMRE varied between 0.7 and 7.72.

13. (Jorgensen, 1995) used 109 maintenance projects for testing several variations of regression, artificial neural networks, and combinations of OSR with regression. He found that two multiple regression models and a combination of OSR with regression worked best in terms of accuracy. He recommended the use of OSR together with expert estimates. The maintenance projects were collected in a large Norwegian organization. Most applications were written in Cobol and or Fourth Generation Languages and were connected to a network database or mainframe. The size varied from less than a few thousand LOC to about 500 KLOC. The duration ranged from less than a year to more than 20 years. The applications consisted of service management, personnel administration, billing and invoicing, order entry, inventory control, and payroll.

14. (Chatzuglou and Macaulay, 1996) investigated regression analysis, COCOMO, and their own method MARCS (Management of the Requirements Capture Stage) on a data set that consisted of projects coming from a university environment and different companies (in total 107 projects). MARCS constructs estimates using rule-based analysis. Rules were extracted for 24 variables from the data set. The focus was on the models' ability to predict resources (time, cost, effort) in the requirements capture and analysis stage. It was found that MARCS outperformed regression, which in turn outperformed COCOMO.

15. (Shepperd et al., 1996) used six different data sets (subset of data used in Shepperd and Schofield) and compared analogy (using ANGEL) with different regression models. In all cases, analogy outperformed regression. (Shepperd and Schofield 1997) compared an analogy-based technique (using ANGEL, ANaloGy SoftwarE tool) with stepwise regression. They used nine different data sets (in total 275 projects from 15 different application domains) and report that in all cases analogy outperformed stepwise regression. ANGEL estimates ranged in terms of MMRE between 0.37 and 0.62, the regression MMRE values were between 0.45-1.42.

[17] The adapted models were also benchmarked against results from Kemerer's study. It turned out that the COCOMO models and FP models were similar in both studies.

16. (Finnie et al., 1997) compared an analogy-based method using CBR with different versions of FP-based regression models and artificial neural networks. The data used consisted of 299 projects from 17 different organizations. They report a better performance of Analogy when compared with different regression models. In addition, artificial neural networks outperformed the analogy-based approach.

17. (Briand et al., 1999b) compared commonly used estimation methods: stepwise regression, Stepwise ANOVA, regression trees (CART), and an analogy-based method based on the ANGEL algorithm. In addition, combinations of CART with regression and CART with Analogy were tried. The data set consisted of 206 MIS projects on which they performed a six-fold cross validation. In general, analogy-based methods performed significantly worse than the other methods used. Regression analysis and regression trees, on the other hand, yielded good results. Combinations of methods did not improve the estimates.

18. (Briand et al., 2000) performed a replication of the study in on a dataset coming from contractors of the European Space Agency. Results were consistent, despite the significant differences in application domains and data sources. One notable difference though was that regression trees (CART) did not perform as well.

19. (Dolado and Fernandez , 1998) compared a method based on artificial intelligence, Genetic Programming (GP), regression and neural networks. They used five different data sets, including the COCOMO dataset, Kemerer's data set, and Albrecht and Gaffney's FP data. None of the considered methods could be proved to be clearly superior.

20. (Kitchenham, 1998) compared a stepwise procedure to analyze unbalanced data (stepwise ANOVA) with regression trees (CART). She used the COCOMO data by randomly generating six pairs of training and test sets. The result in terms of MMRE was that stepwise ANOVA (MMRE=0.42) outperformed the generated regression trees (MMRE=0.62).

21. Expert judgment is very rarely reported in comparative studies. It is investigated, in combination with other estimation methods, by (Strensrud and Myrtveit, 1999). They tested several hypotheses on a database consisting of 48 COTS projects from Andersen consulting. They found that subjective estimation in combination with regression or analogy added value compared to subjective estimation when performed with the aid of historical data only. The value of a tool or technique depends on who the tool user is.

22. (Hughes et al., 1998) analyzed six Telecommunication projects containing effort data collected separately for functions. Thus the data set consisted of 51 cases (functions). They compared regression, artificial neural networks, analogy, and regression trees. As a result they recommend the combined use of regression and analogy.

23. (Walkerden and Jeffery, 1998) compared the Analogy-based techniques ACE and ANGEL (which 2 variants of analogy) and compared it with regression analysis and expert judgment. They used a data set from one Australian software organization (19 projects). Expert judgment yielded the best estimates in terms of accuracy followed by Analogy and linear regression.

24. (Grady and Mac Donell, 1999) investigated the influence of different data set characteristics on the performance of three modeling methods, namely, OLS regression, Robust regression, and ANN's. The data sets used were from other studies, however, different partitions of the data into training and test sets were generated (Dolado 1998; Miyazaki et al., 1994; Li and Henry, 1993). Performance was evaluated using different measures, such as MMRE, balanced MMRE

(BMMRE), and the average absolute error (AAR), and different prediction levels regarding MMRE and BMMRE. In all cases robust and least squares regression models performed better than ANN's according to the MMRE measure.

6.3 Critical Analysis of Empirical Studies

A considerable variation can be observed across studies regarding the quality of models and estimates of cost estimation methods. One major reason is the diversity of the underlying project data. Other reasons are the variety of the estimation methods' assumptions and the configuration choices of the methods compared.

Software Engineering Data

Comparisons of software cost estimation methods are based on data sets varying in size, distributions, project environments, projects' age, and so forth. When large data sets were used, the data was not made publicly available. Therefore, most estimation methods were not consistently applied to one large data set and there is no benchmark data set that the community could use for consistent comparisons. Thus, results across different studies are not comparable and general conclusions are difficult to draw. Even when studies use the same data set, because comparison procedures are not standardized, they are usually not comparable (e.g., Briand et al., 1992) and (Srinivasan and Fisher, 1995). Two recent studies that try to overcome this problem and compare most of the commonly used estimation methods on large data sets are (Briand et Al., 1999b, Briand et al., 2000).

Relevance and Definition of Cost-Drivers

The characteristics of the data set have a significant influence on the estimation methods' predictive accuracy. This stems from the fact that some very important project attributes may be missing in the data sets. For example, software development sites may use different tools. Thus, if this factor is not collected explicitly, but is not constant across different sites (Srinivasan and Fisher, 1995), the accuracy of the estimation models will deteriorate. Moreover, the relevance of cost factors may change over time; new factors may become relevant and other previously important factors may become superfluous. Thus, it becomes difficult to use models derived from past data, even if their quality was originally satisfactory.

Also, a lack of precise definitions of the cost factors collected causes large variations in the methods' performance. Much discussion, for example, concerns the validity of using lines-of-code to measure the size of a system. There is no standard definition of how to count LOC, also, no standard definition converting procedural languages to one language-independent counting procedure. One attempt to overcome the second problem are Function Points (FP) (Albrecht and Gaffney 1983, see also Section 4) and its derived methods. Another attempt to overcome the language dependence has been made by Jones (Jones 1998). He provides a table to transform LOC into language independent counts. Beside the LOC issue, factors impacting cost are sometimes defined in an imprecise way, which also contributes to poor estimates (Briand et al., 1999b).

Estimation Method Sensitivity to Configuration Choices

Beside differences the data set characteristics, an estimation method is sensitive to the configuration choices made by the user. For example, Srinivasan and Fisher report a great variation of regression trees (CART) using different error thresholds to build the tree (MMRE ranged from 3.64 to 8.35). Or, there are several ways to tune an analogy-based method, like, using different similarity measures, deciding upon the number of analogs to use for estimation, or selecting appropriate adaptation rules (Shepperd and Schofield, 1997). Moreover, a method's performance is sensitive to the training and test sets used. For example, Kemerer (Kemerer, 1987) found that some of the investigated approaches were sensitive to the test data. Also, (Srinivasan and Fisher 1995) performed 20 randomized trials of training and test set combinations and applying CART. They report a relatively low mean of R^2 values, but many runs yielded a strong linear relationship (i.e., high R^2 values). Another point is that developers of a method may show a positive bias towards their method and therefore try to optimize it for a certain context. Thus, their method turns out to perform best (Lederer and Prasad 1992).

Quality Evaluation

The quality of a model or an estimate (i.e., its accuracy) is mostly evaluated using the MMRE value only. Little is usually provided regarding the reliability of the results and the variation in MRE. A few studies provide the minimum, maximum, and standard deviation of the MRE values (e.g., Mykhopadhyay et al., 1992; Stensrud and Myrtveit 1998). Last, besides the quality of models and estimates, other important aspects of cost estimation methods are rarely evaluated (Kitchenham, 1990) in most studies (e.g., complexity, interpretability, and practical issues).

6.4 Evaluation of Estimation Methods

This subsection assesses existing cost estimation methods using the evaluation framework specified in Section 5. Evaluation criteria are only used when applicable and not all criteria are applicable to all estimation methods (see Table 4). The evaluation is derived in a subjective manner using the results reported in the literature as well as our experience.

For each estimation method category (see Figure 1), we provide a discussion of relevant evaluation criteria. For the sake of clarity, we then summarize our discussions into tables. For the evaluation criterion "Quality of model and estimate", we base our conclusions on the published studies summarized above.

6.4.1 Evaluation of Data Driven Methods

Quality of Models and Estimates: We can conclude that in most studies, Ordinary Least-Square regression (OLS) performed considerably well when compared with other techniques. OSR was not extensively compared to many other methods, but relatively good results were reported so far. Stepwise ANOVA performed as well as or better than the methods it was compared with. Studies evaluating CART regression trees consider this type of models as a good alternative, because of their simplicity, but they seem to provide mixed results. Considering the range in MMRE values across different studies, we observe that CART has the largest variation and OLS has the second largest one. But the latter was also evaluated within many different contexts. Stepwise ANOVA and OSR achieved relatively small variations in MMRE values. Last, the Robust Regression method LBRS (least squares of balanced relative errors) was compared with OLS regression in one study (Miyazaki et al., 1994). The results using the OLS solution were worse than LBRS for all evaluation criteria Miyazaki and others used[18].

Interpretability: CART and OSR models are very easy to interpret. An OSR model consists of logical predicates and a CART model is simply a partition tree. These types of models are more intuitive to practitioners. On the other hand, OLS, stepwise ANOVA, RR are very difficult to interpret for non-statisticians. For a number of reasons beyond the scope of this paper, the influence of a cost factor is often difficult to assess based, for example, on regression coefficients. This is especially true when there are many variables and many interactions amongst them, or when the model includes logarithmic scales or dummy variables (Berry and Feldmann, 1985).

Assumptions: To reliably apply OLS, many assumptions are required and have to be checked (Schroeder et al., 1986). These assumptions concern the constant average variation in error on the dependent variable range (homoscedasticity), outlying observations, interval scaled variables, the size of the data set, correlations among independent variables, and the assumed functional relationship among variables.

A regression model can be distorted by the existence of outlying observations (e.g., Kemerer 1986). This is addressed by RR through alternative minimization criteria rather than the sum of squared errors (Rousseeuw and Leroy, 1987; Miyazaki et al. 1994)

OSR and CART require fewer underlying assumptions than OLS. The flip side is that both require a larger amount of data though. A CART model assumes that a variable is equally relevant as a predictor regardless of its value (homoscedasticity)[19]. Thus, non-useful partitions can be generated (Briand et al., 1992). CART and OSR can easily deal with variables on different measurement levels but require to discretize continuous scales. Interactions between independent variables are taken into account. One characteristic of CART and OSR is that they cannot estimate a value along a dimension that is outside the range of values encountered in the underlying data set (extrapolation). However, this kind of OLS estimates is not recommended either. In contrast to CART, OSR can elegantly deal with missing data. When a value is missing for a variable, OSR still considers the project’s other variables, whereas CART regression trees ignore the whole data point.

Stepwise ANOVA is relatively simple to understand, can deal with variables of different types, copes with data sets consisting of a large number of variables and a small number of cases (i.e., software projects), and identifies confounded independent variables. Furthermore, it assumes a normally distributed dependent variable. But, if a problem, this can sometimes be addressed through logarithmic transformation of the data.

Repeatability: All methods in this subsection are highly repeatable, because their algorithms are well documented and can be automated. The steps to build a model (modeling method) are also well defined and assumptions are known. Thus the model development does not depend very much on a person’s expertise in estimation.

Complexity: OSR is judged to be most complex, since an OSR model cannot be developed manually and relies on complex algorithms. In contrast to this, an OLS model can be derived with the help of a simple calculator. RR algorithms are more complex than OLS, thus tool support is

[18] This study is not mentioned in Table 11, because this is the only comparison of robust regression with other techniques in the area of cost estimation.

[19] The splitting according to a variable might provide good predictive results when following the left path, but poor predictions may be obtained when following the right path.

required. To develop a CART model is relatively complex as several parameters need to be set and, usually, this is an iterative trial and error process. Tool support is required and several parameters need to be understood by the model developer (e.g., splitting criteria, tree-pruning, selection from alternative trees).

Automation of Modeling: To build an OSR model manually is too effort consuming. Tool support exists and additional services are provided for its application (Fraunhofer IESE). No commercial tool support for Stepwise ANOVA[20] is available to date. As the procedure is relatively simple, it is possible to generate a model manually, or to automate it using a statistical tool script language. OLS is a standard method available in all statistical tool packages. Some tools support CART (e.g., Salford Systems) and RR (e.g., Stata Corporation).

Transparency: All data-driven methods are based on precisely defined procedures and algorithms. Many good books exist describing OLS (e.g., Berry, Feldman 1985). A standard reference for RR is (Rousseeuw and Leroy, 1987). The CART algorithm is described in detail in (Breiman et al., 1984). ANOVA is also a standard method and a stepwise procedure is described in (Kitchenham, 1998). The OSR algorithm is published in (Briand et al., 1992), (Briand et al., 1993).

Application Coverage: OLS, stepwise ANOVA, and RR are typically used for prediction purposes. Most studies report no other usage regarding resource modeling. (Briand et al., 1998b) report on using OLS for productivity benchmarking. (Maxwell and Froselius, 2000) report on benchmarking results derived from the application of stepwise ANOVA. Other purposes than prediction are potentially possible when confidence intervals or probability distributions of estimates are provided by models and tools. An OSR model can be used for prediction, benchmarking, and risk assessment using the probability distributions that are derived from the data subset that optimally characterizes a new project. Similarly, a CART model can be used for prediction and to benchmark a project against similar projects (i.e., in the same terminal node of a regression tree). A description of how to use a CART model for benchmarking is reported in (Briand et al., 1998b).

Generalizability: OLS has widely been applied and evaluated for software cost estimation in many different environments (e.g., MIS, embedded systems, maintenance). Stepwise ANOVA was applied using on the European Space and Military database (Greves and Schreiber, 1996), a data set of MIS projects from Finland (STTF) and the COCOMO data (Boehm 1981). Applications of OSR in the area of cost estimation are limited: the COCOMO and Kemerer data sets, and a set of 109 Norwegian maintenance projects. CART regression trees were applied for cost estimation to the multi-organization data set from the European Space Agency (space and military applications), to the multi-organization Laturi database from Finland, to some Telecommunication projects, and to the COCOMO data and Kemerer data. Robust regression (RR) was applied to predict the effort of three object-oriented systems coded in C++. More specifically, different OO-metrics were compared with respect to their capability to produce satisfactory effort predictions error (Nesi and Querci, 1998). Another application of robust regression was a data set collected by Fujitsu Large Systems Users Group (Miyazaki et al. 1994). The data consisted of 48 systems in 20 companies.

Automation of Method Usage: The application of OLS, stepwise ANOVA, RR, CART, or OSR models is not explicitly supported by any tool. Once a model is developed, it can be used with the help of a programming language internal to a statistical tool. The models (OLS, ANOVA, RR, or

[20] ANOVA itself (which is part of the stepwise procedure) is provided by any statistical package.

OSR) can then be used in the context of iterative evaluation procedures (e.g., cross-validation) to assess the accuracy of these models. Besides this, no tool support exists to help appropriately use these models for resource estimation, risk analysis, or benchmarking. As OSR and CART models are easier to interpret and use their manual application is not as much of a problem as for the other three methods.

Evaluation Criteria	**OLS**	**CART**	**Stepwise ANOVA**	**OSR**	**RR**
Interpretability (low – – high ++)	–	++	–	++	–
Assumptions (many– – few ++)	– –	+	+	+	–
Repeatability (low – – high ++)	++	++	++	++	++
Complexity (high – – low ++)	++	–	++	–	+
Automation of Modeling (low – – high ++)	++	+	–	– –	+
Transparency (low – – high ++)	++	++	++	++	++
Application Coverage (limited – – high ++)	–	+	–	+	–
Generalizability (limited – – high ++)	++	+	+	–	–
Automation of Method Usage (low – – high ++)	–	– –	– –	– –	– –

Table 12: Evaluation of Data-Driven Methods

6.4.2 Evaluation of Composite Methods

Quality of Models and Estimates: As reported from an initial study (Briand et al., 1998a), COBRA performed much better than commonly reported results. The MMRE obtained through a cross validation was 0.09, which is very encouraging, though just an initial result. Analogy performed best, in 60% of the cases reported in published studies. In 30% of the cases it showed the worst predictive accuracy, hence suggesting some instability. Analogy obtained best results when using adaptation rules and heuristics to identify significant variables to measure similarity. A fully automated version of Analogy is more likely to yield good results when used on small data sets with a small number of variables. In this case, a comprehensive search for optimal similarity functions can be performed (Shepperd and Schofield, 1996).

Interpretability: The COBRA modeling procedure yields a causal model of cost drivers on cost. It includes both a qualitative causal structure and a quantification of causal relationships. Such models are clearly interpretable and specific to a given organization where the model is developed. An Analogy model consists of two main components: similarity measure, adaptation rules. It is only interpretable in terms of which cost drivers are selected in the similarity function and which are the relative weights associated with them. However, there is no clearly defined procedure in the literature to define tailored similarity functions. Most articles on the topic use simple distance functions, which equally weigh all cost drivers.

Assumptions: COBRA and Analogy can deal with variables defined on all measurement levels. They do not assume a particular functional relationship between cost and the cost drivers and no assumptions are made about the distribution of the underlying data set. COBRA is an additive model, which aims at defining cost drivers that are orthogonal, though it allows for the introduction of interactions between variables. COBRA, when applied in the context of internal development, requires only little data (at least 10 completed projects) but necessitates access to a representative sample of experts (i.e., experienced project managers). As opposed to what is sometimes suggested, analogy requires expert knowledge (in cost estimation or/and project management) to derive a model (i.e., to define appropriate similarity functions, or adaptation rules).

Repeatability: Even though the algorithm for Analogy is well documented, the method is not very likely to achieve repeatable results, because there are many possibilities to define a similarity function, adaptation rules, or even use analogs once they are identified. As there are currently no clear recommendations or procedures to address these issues, it is dependent on the model developer. In contrast to this, knowledge acquisition procedures are part of the COBRA modeling method. But the results still depend on the selection of experts. Different results may be obtained if the chosen sample of experts is not representative.

Complexity: To build a COBRA model from scratch is relatively complex as it involves the application of knowledge acquisition procedures involving a number of experts. It requires some expertise in cost estimation, knowledge acquisition, and statistics to build an initial model and tool support to run and maintain the model. Although Analogy is claimed to be simple, if it is to be accurate, an adequate similarity function and adaptation rules need to be devised. This is likely to be a complex procedure as little is known on the topic.

Automation of Modeling: Several tools exist to support the Analogy method, such as ANGEL (Shepperd and Schofield 1997), ACE (University of New South Wales), or CBR-Works (University of Kaiserslautern). CBR-Works is more flexible in the sense that is allows the user to define many different similarity measures and adaptation rules. ANGEL is fully automated but just provides model construction capabilities based on a given similarity measure. Moreover ANGEL implements an exhaustive search to identify the significant variables, what makes it unusable when using a large number of project data points and variables. For COBRA, no commercial tool support exists to date, but a model can easily be constructed with the help of other commercial tools, like spreadsheets and simulation tools (Vose, 1996).

Transparency: The two evaluated composite methods are fully documented. Many publications exist describing Analogy and its applications (e.g., Shepperd and Schofield, 1997, Delany et al., 1998). One advantage of Analogy-based estimation is that practitioners can justify their decisions on the basis of previous projects. All the necessary information to understand the COBRA method is described in (Briand et al., 1998a). But in order to be able to build and use a COBRA model, it would require more examples, details, and guidelines on how to build tool support.

Application Coverage: Analogy may be used for prediction purposes and benchmarking. Predictions can be obtained by taking, for example, the mean value of the analogs. To use Analogy for benchmarking, a sufficient number of analogs (e.g., at least 10) must be retrieved to obtain a large enough basis of comparison. A COBRA model can be used for prediction, benchmarking, and risk assessment. Risk analysis is an integral part of COBRA.

Generalizability: One publication exists about COBRA's application to date. The method was applied within some case studies within the MIS and aerospace engineering domains. Initial results are very encouraging, but more studies are necessary. Analogy was evaluated across many different environments, like MIS projects, Telecommunication, 19 projects from Australian organizations, and standard data sets like COCOMO and Kemerer. Nevertheless it is difficult to derive generalizable conclusions, since the results across studies are inconsistent and are very sensitive to how the analogy algorithm was tailored. No conclusions can be drawn regarding conditions under which Analogy works best.

Automation of Method Usage: No tool support is provided to use an Analogy model for cost estimation. No commercial tool support exists for using a COBRA model but it can be developed based on existing commercial tools, e.g., (Vose 1996).

Evaluation Criteria	**Analogy**	**COBRA**
Interpretability (low – – high ++)	–	+
Assumptions (many– – few ++)	++	+
Repeatability (low – – high ++)	–	+
Complexity (high – – low ++)	+	–
Automation of Modeling (low – – high ++)	++	–
Transparency (low – – high ++)	++	+
Application Coverage (low – – high ++)	+	++
Generalizability (low – – high ++)	+	–
Automation of Method Usage (low – – high ++)	– –	– –

Table 13: Evaluation of Composite Methods

6.4.3 Evaluation of Non-Proprietary Methods

Quality of Models and Estimates: The accuracy of the Putnam method[21] was empirically validated and relatively poor results were consistently found by (Conte et al. 1986), (Kemerer 1987), and (Kitchenham and Taylor 1988). The Putnam method performed worst in 80% of the cases reported in comparative studies. In general, the effort estimates are very sensitive to poor estimates of the technology factor.

Boehm validated his COCOMO model in the 1970's on his TRW data set using the prediction level as an evaluation measure. He obtained very good results for intermediate and detailed COCOMO, and quite poor ones for the basic COCOMO. Independent evaluations performed on other data sets have not always produced such good results. In general, the accuracy in terms of MMRE ranges form very good (i.e., below 0.25) to very poor (MMRE=7.58). It was found that intermediate and detailed COCOMO do not necessarily improve the estimates and that the model systematically over-estimates effort (Kemerer 1987). Other studies found a bias towards underestimating effort when using COCOMO. Intermediate and detailed COCOMO may suffer from over-fitting problems.

No publicly available results exist about the performance of COCOMO II compared to other estimation methods. Affiliate organizations are currently using COCOMO II. The COCOMO versions generated in '97 and '98 were compared with each other. It turned out that the

[21] We use the term Putnam's method and SLIM interchangeably; even though SLIM is the name of the tool supporting the method.

calibration of COCOMO II by incorporating expert knowledge using Bayesian Statistics reached promising results outperforming the COCOMO II version 97 (Chulani et al. 1999).

Inputs Required: To use the SLIM method, it is necessary to estimate system size, to determine the technology factor, and appropriate values of the manpower acceleration. Technology factor and manpower acceleration can be calculated using similar past projects. System size in terms of KDSI is to be subjectively estimated. This is a disadvantage, because of the difficulty of estimating KDSI at the beginning of a project and the dependence of the measure on the programming language.

Basic COCOMO I requires the user to provide KDSI as a size measure and to set the development mode. For intermediate and detailed COCOMO, values for 15 generic cost factors, and several parameters to account for code re-use, are required. For the detailed version, these parameters are even required for each module in the system. The determination of parameters for code re-use is highly subjective. It was also found that some of the cost factors are not statistically independent (Subramanian and Breslawski, 1993), (Briand et Al., 1999a).

COCOMO II can handle different size measures (FP, LOC, Object Points). Different numbers of cost drivers and size measure are required depending on the stage of development (Application Composition, Early Design, Post Architecture). The Application Composition model is based on Object Points. Function Points and 7 cost drivers are used for the Early Design model. The Post Architecture model uses source instruction / Function Points, 17 cost drivers, and a set of 5 factors determining the project's scaling exponent.

Completeness of Estimate: The SLIM model provides estimates for effort, duration, and staffing information for the total life cycle and the development part of the life cycle. COCOMO I provides equations to estimate effort, duration, and handles the effect of re-using code from previously developed software. COCOMO II provides cost, effort, and schedule estimation, depending on the model used (i.e., depending on the degree of product understanding and marketplace of the project). It handles the effect of reuse, re-engineering, and maintenance adjusting the used size measures using parameters such as percentage of code modification, or percentage of design modification.

Type of Estimate: The Putnam method yields point estimates. COCOMO I also provides point estimates. COCOMO II produces a point estimate and an uncertainty range, which is dependent on the completeness of the inputs provided.

Calibration: The calibration of a COCOMO II estimate is given through the three different models that can be subsequently used depending on the development stages. Moreover, the 97 and 98 versions are aiming at calibrating the 1994 version by using weights and Bayesian statistics, respectively.

Interpretability: All models are relatively hard to interpret for practitioners, because of their structure. Functional equations need to be understood and coefficients are to be interpreted. In addition, there is no guarantee that such generic models are representative of reality in a given organization.

Assumptions: SLIM assumes the Rayleigh curve distribution of staff loading. The underlying Rayleigh curve assumption does not hold for small and medium sized projects. Cost estimation is only expected to take place at the start of the design and coding, because requirement and specification engineering is not included in the model.

The COCOMO models assume an exponential relationship between cost and system size. The values used for the exponential coefficient imply the assumption of diseconomies of scale. This assumption is not of general validity as shown in several studies (Banker et al., 1989, Briand et al., 1999a). COCOMO I assumes the cost of reuse to be a linear function of the extent that the reused software needs to be modified. COCOMO II uses a non-linear estimation model. This assumption is based on results obtained from the analysis of 3000 reused modules in the NASA Software Engineering Laboratory.

Repeatability: Though the algorithms are well documented their repeatability depends on subjectively estimation of the required input parameters. This may vary depending on the experience of the model developer, the tool support to obtain the estimates, and the point in time of the estimate. However, COCOMO II accounts for the uncertainty of input information.

Complexity: The SLIM model's complexity is relatively low. For COCOMO the complexity increases with the level of detail of the model. For COCOMO I the increasing levels of detail and complexity are the three model types: basic, intermediate, and detailed. For COCOMO II the level of complexity increases according to the following order: Application Composition, Early Design, Post Architecture.

Automation of Model Development: The Putnam method is supported by a tool called SLIM (Software Life-Cycle Management). The tool incorporates an estimation of the required parameter technology factor from the description of the project. SLIM determines the minimum time to develop a given software system. Several commercial tools exist to use COCOMO models.

Transparency: The methods' transparency is very high. Models and assumptions are publicly available and documented.

Application Coverage: SLIM aims at investigating relationships among staffing levels, schedule, and effort. The SLIM tool provides facilities to investigate trade-offs among cost drivers and the effects of uncertainty in the size estimate.

COCOMO I does not provide any risk management features. COCOMO II yields an estimation range rather than just a point estimate and therefore provides a basis for risk assessment.

Generalizability: The SLIM model is claimed to be generally valid for large systems. COCOMO I was developed within a traditional development process, and was a priori not suitable for incremental development. Different development modes are distinguished (organic, semi-detached, embedded). COCOMO II is adapted to feed the needs of new development practices such as development processes tailored to COTS, or reusable software availability. No empirical results are currently available regarding the investigation these capabilities.

Comprehensiveness: Putnam's method does not consider phase or activity work breakdown. The SLIM tool provides information in terms of the effort per major activity per month throughout development. In addition, the tool provides error estimates and feasibility analyses. As the model does not consider the requirement phase, estimation before design or coding is not possible. Both COCOMO I and II are extremely comprehensive. They provide detailed activity distributions of effort and schedule. They also include estimates for maintenance effort, and an adjustment for code re-use. COCOMO II provides prototyping effort when using the Application Composition model. The Architectural Design model involves estimation of the actual development and maintenance phase. The granularity is about the same as for COCOMO I.

Availability of Estimates: For SLIM and COCOMO I estimates cannot be provided early on in the life cycle, because system size in terms of KDSI is reliably available after coding. COCOMO II gives estimates at different levels of granularity depending on the development stage.

Automation of Method Usage: Across the US and Europe SLIM has been purchased in more than 60 organizations. COCOMO also has several tools that support the usage of the method.

Evaluation Criteria	**SLIM**	**COCOMO I**	**COCOMO II**
Inputs required (many – – few ++)	+	–	–
Completeness of estimate (low – – high ++)	–	+	++
Type of Estimate (poor – – complete ++)	–	–	+
Calibration (no – – high ++)	–	+	++
Interpretability (low – – high ++)	–	–	–
Assumptions (many– – few ++)	– –	–	–
Repeatability (low – – high ++)	–	–	+
Complexity (high – – low ++)	+	+	+
Automation of Modeling (low – – high ++)	++	++	++
Transparency (low – – high ++)	++	++	++
Application Coverage (low – – high ++)	–	–	+
Generalizability (low – – high ++)	+	–	+
Comprehensiveness (low – – high ++)	–	++	++
Availability of estimates (late – – early ++)	– –	–	++
Automation of Method Usage (low – – high ++)	++	++	++

Table 14: Evaluation of Not Proprietary Methods

6.4.4 Evaluation of Proprietary Methods

These types of methods are not described in detail in this report, because their internal details are not publicly available. Nevertheless we try to evaluate them based on publicly available information. We cannot, however, provide detailed information for all of our evaluation criteria.

Quality of Model and Estimate: ESTIMACS was applied on the Kemerer data set in 1987 and performed better than SLIM, COCOMO, or OLS regression (see Section 6.1). To the knowledge of the authors, this is the only independent evaluation about the performance of proprietary methods that is publicly available.

Inputs Required: ESTIMACS uses a FP like measure to provide an estimate. 25 questions are used to generate the model. The inputs may be divided into six groups: size variables, product variables, environment variables, personnel variables, project variables, and user factors.

PRICE-S requires project magnitude/size, project application area, the level of new design and code, experience and skill levels of project members, hardware constraints, customer specification, reliability requirements, development environment. To obtain the size of the system, a separate sizing module is available allowing for FP analysis, SLOC, and object-oriented variants of feature points.

Knowledge Plan has a large number of cost factors (more than 100) that can be estimated by the user to obtain an initial estimate. The more answers provided, the more accurate the estimate. They are grouped according to the following categories: classification of the project, personnel, technology, process, environment, and product. The system size is required. The tool supports FP, SLOC, sizing by components, or sizing by analogy to determine the size of the system.

Completeness of Estimate: ESTIMACS provides outputs including effort, schedule, peak staffing level, work breakdown for staff level, effort and cost, function point counts, maintenance effort, risk assessment, financial information.

PRICE-S gives a detailed schedule summary on a monthly basis and the costs incurred as the development progresses, as well as staffing profiles. It also provides risk projections based on variations in the estimated size, the development schedule or index of the application area. The monthly progress summary shows the distribution of effort and associated costs throughout the development project. Moreover, re-use it taken into account.

Knowledge Plan provides estimates of effort, and schedule, cost, resource requirements, and level of quality. Its project management functionality allows the integration of non-software activities into a project plan.

Type of Estimate: ESTIMACS provides point estimates as well as upper and lower bounds for the estimate. The level of risk with the successful implementation of the proposed system is determined based on responses to a questionnaire that examines project factors like system size, structure, and technology.

Knowledge Plan provides point estimates and a level of accuracy for the estimate. The accuracy level is calculated based on the number of similar projects found in the database, the percentage of answered project attributes, and the size measure used.

Calibration: PRICE-S develops a new family of relationships for each new project to be assessed to fit each specific application. Thus, calibration is an integral part of the model development.

The Knowledge Plan user can adjust initial estimates for a project as cost factor values (project attributes) are changing or additional information is provided about the factors.

Interpretability: This characteristic cannot be assessed as the models enclosed in these methods are proprietary.

Assumptions: PRICE-S is a parametric cost estimation method. But it does not assume one single cost estimation relationship and is not based on one single database. PRICE-S is based on a large and complex set of algebraic equations that model the organizational process used to plan and schedule resources for a software project. It parametrically classifies different types of software projects according to the functions that are to be performed. Eight function types are distinguished ranging from mathematical operations to operating systems and interactive real-time systems. The results are summarized into one “Application”-value.

Knowledge Plan is a knowledge-based estimation method. The project database consists of 6700 projects from many different environments. An estimate for a new project is based on a subset of projects that are “similar” to the new one. To obtain an estimate from the schedule information provided by the user a critical path schedule is calculated.

Repeatability: In general, the results obtained using one of the tools highly depend on the “right” input information being provided. PRICE-S, for example, has a number of functions that are tailored to the specific project characteristics. Different project information will lead to using a different set of functions and thus different results. Knowledge Plan estimates depend in part on the retrieved historical project data. Thus, non-relevant subsets may be retrieved when project characteristics are poorly assessed. In general, the experience with the usage of the tool has also a large impact on the repeatability of the results.

Complexity: Based on the number of inputs required, Knowledge Plan seems to be the most complex model. PRICE-S and ESTIMACS come in second and third position, respectively. But, as the model details and hence what is done with the inputs is not visible, this assessment is very subjective.

Automation of Model Development: All methods are fully supported by tools.

Transparency: These methods’ modeling and implementation details are not public domain.

Application Coverage: PRICE-S can translate uncertainty in project characteristics into assessments of cost and schedule risk.

Knowledge Plan supports project management, re-estimation, shows the impact of progress on schedule, effort, cost or quality, it embodies what-if analysis to explore alternative strategies for personnel, process, and technology.

Generalizability: According to its vendor, PRICE-S is supposed to cover all types of systems, including business systems, communications, command and control, avionics, and space systems. To date, no independent empirical results are available to confirm this.

According to its vendor, Knowledge Plan covers all major software environments. Six different project types are categorized in underlying the data base: 42% systems, 28% MIS, 8% commercial, 2% outsource, 2% end-user, 4% military, and 14% others. Projects are classified by new (30%), enhancement (54%), and maintenance (16%). The percentage of projects in the database by size is as follows: small 21% (~5 PM), medium 46% (~70 PM), and large 33% (~750 PM). The vendors state that many organizations developed estimates within an accuracy of 5%. Unfortunately, this is not confirmed by any publicly available, independent empirical study.

Comprehensiveness: PRICE-S derives schedule and effort estimates for the core development phases, such as design, cost/unit test, and system test, and integration. In addition, estimates are provided for requirement analysis, sw/hw integration. Cost of maintenance is estimated in three categories: software repair, performance improvement, and specification change.

Knowledge Plan derives project plan information for predefined and user-defined sub-tasks.

Availability of Estimates: PRICE-S and Knowledge Plan provide initial estimates early on in the life cycle. They can and should be adjusted when more information becomes available in the project.

Automation of Method Usage: In all cases, complete tool support is available for using the models.

Evaluation Criteria	**PRICE-S**	**Knowledge Plan**	**ESTIMACS**
Inputs required (many – – few ++)	–	– –	–
Completeness of estimate (low – – high ++)	++	++	+
Type of Estimate (poor – – complete ++)	?	+	+
Calibration (no – – high ++)	+	+	?
Interpretability (low – – high ++)	?	?	?
Assumptions (many– – few ++)	–	–	?
Repeatability (low – – high ++)	–	–	–
Complexity (high – – low ++)	–	– –	+
Automation of Modeling (low – – high ++)	++	++	++
Transparency (low – – high ++)	– –	– –	– –
Application Coverage (low – – high ++)	+	++	?
Generalizability (low – – high ++)	?	?	?
Comprehensiveness (low – – high ++)	+	++	?
Availability of estimates (late – – early ++)	+	+	?
Automation of Method Usage (low – – high ++)	++	++	++

Table 15: Evaluation of Proprietary Methods

6.4.5 Evaluation of Non-Model Based Methods

Quality of Estimates: Non-model based methods may achieve highly accurate estimates, when being supported by a tool or combined with other cost estimation methods (e.g., analogy, regression). Using solely non-model based methods may lead to very inaccurate predictions, as we do not know what constitute a good expert in software engineering management. People also tend to underestimate the time required to do a task when they themselves are to do the work (DeMarco, 1982). This is confirmed on small tasks by other researchers, while for larger tasks an overestimation has been observed. However, it has also been reported that bottom-up estimation helped predict projects within 6% of their scheduled dates and costs (Tausworthe, 1980).

Assumptions: Expert Judgment makes no assumptions on data from previous projects. However, it assumes that experts are able to provide unbiased, accurate estimates. This can only be ensured, to a certain extent, by the rigorous use of elicitation techniques (Meyer and Booker 1991).

Repeatability: Approaches are more likely to be repeatable if the estimation process is documented through predefined steps, forms to be filled in by different experts, or requests explanations for estimates. The Delphi-Method estimation process is more formal and thus more repeatable, explanations for the provided estimates are to be provided and thus results are more traceable and justified. Expert Judgment is highly dependent on the individual, often not very well documented, and thus less repeatable. However, this can be addressed by using more formal elicitation methods (Meyer and Booker 1991).

Complexity: The complexity may vary dependent on the number of experts involved, the number and organization of meetings, or the predefined steps to be performed.

Transparency: The transparency depends on the definitions of the estimation techniques to apply. If experts individually derive an estimate, it is usually not evident to determine how they reached their conclusions. But, again, more rigorous elicitation techniques could be used in the future.

Generalizability: Expert Judgment has been extensively used in practice, but scarcely studied, from a scientific perspective, in software engineering. A survey of Dutch organizations, for example, reports a 25.5% usage of Expert Judgment for estimation. 62% of the organizations that produced estimates based them on an ad-hoc process while only 16% used more formal, better defined methods.

Evaluation Criteria	**Expert Judgement**
Assumptions (many– – few ++)	++
Repeatability (low – – high ++)	– –
Complexity (high – – low ++)	+
Transparency (low – – high ++)	– –
Generalizability (low – – high ++)	++

Table 16: Evaluation of Non-Model based Methods

6.5 Evaluation of Different Method Types

This section compares the estimation methods on a higher level of granularity according to the defined classification in Section 3. Therefore, the evaluation is more general and provides an overview.

6.5.1 Model based vs. Non-Model Based Methods

Estimation effort: Model based methods, in contrast to non-model based methods, can be considered as reusable corporate knowledge. The resource estimation process is more independent from individuals' capabilities. In addition, relying on non-model based methods requires heavy involvement of experts to generate an estimate for every new project. This makes it impractical as most organizations' experts have very tight schedules and little availability. Moreover, the cost to derive an estimate cannot be reduced and its accuracy improved from one project to another, since there is no organizational learning process.

6.5.2 Generic Model based vs. Specific Model Based Methods

Validity of assumptions: Generic models assume a universal resource / system size relationship. In practice, it would seem reasonable to assume that different relationships exist across environments. We believe that this relationship is to be individually established in the context of application. When applying specific model-based methods, relationships have to be investigated based on data collection.

Accuracy: Generic algorithms may produce highly inaccurate results, when used without calibration in other contexts. The reason is that they use predefined cost drivers, which are not necessarily valid in every context. The definition of the cost drivers is also generic, and the relationship between different attributes is predefined. (Lederer and Prasad, 1992) conducted a study of cost estimating practices reported by 115 computing managers and professionals. They found that the 63% of users of generic methods (in this case software packages) overrun their estimates while non-users reported that nearly the same percentage: 62%, overrun their estimates. This study indicated that using tool supported generic methods is not associated with a reduction in effort overruns.

6.5.3 Proprietary vs. Not Proprietary Methods

Acceptance: The proprietary models are implemented as a black box and, therefore, the actual underlying algorithms are not available. Therefore, it is difficult to justify an estimate generated with a proprietary method, as its credibility is solely based on the credibility of the vendor. Thus, methods based on proprietary models are less likely to be accepted among practitioners than non-proprietary methods. It is natural to expect that project managers and practitioners want to understand their estimates.

6.5.4 Data Driven vs. Composite Methods

Integrating expert knowledge: The problem with software engineering data is that relatively few data points are usually available in a given organization, and the data is unbalanced and tends to have outliers and missing values (Bisio and Malabicchia, 1995). Therefore, conclusions that just rely on software project data might be misleading or difficult to obtain. Consulting expert knowledge is therefore of practical importance. Composite methods try to incorporate expert knowledge into the model building process. The expert's knowledge is made explicit in the model, which also improves the acceptance of this kind of models.

7 Considerations Influencing the Choice of an Estimation Method

Depending on the application context of resource estimation methods, different weight should be put on the evaluation criteria described in Section 5. In addition, generic factors not related to the estimation methods themselves will affect their applicability. This section briefly discusses the main organizational factors that impact the application of estimation methods and their evaluation.

7.1 Practical Considerations

Level of acceptance of generic models: Depending on the environment, the use of generic models (either commercial or public domain) may not gain acceptance and the estimation method may never make it into practice. This is a common situation for several reasons. First, competent engineers and managers usually like an estimation method to account for their own, specific experience. Second, they usually wish to understand the justifications for an estimate, so that they can trust it but also understand what can be changed about the project to obtain a different estimate. Third, it is common to see generic models to use cost drivers whose definitions are not clearly adequate or comprehensible in a given environment. Therefore, practitioners have to force their way of thinking into a different vocabulary and set of definitions. This is not always easy or practical. To summarize, the level of acceptance of generic models, though not a technical consideration, will determine whether generic, commercial or public domain estimation methods are likely to transfer into practice.

Access to Experts: Certain estimation methods are based fully or partially on some expert elicitation process. This elicitation aims either at building models or at making direct estimates. In both cases, this is time consuming. Since experts are, by definition, extremely busy people, it may be difficult to find the time required for knowledge elicitation. In addition, the required expertise may not exist, if the organization is new or the projects to estimate belong to a new line of product, based on entirely new technology. Then experts with enough expertise to extrapolate their expertise to new situations are required.

Capability to handle Complexity: Certain modeling techniques require a substantial amount of training if one wishes to interpret the models and understand the justifications for an estimate. Unfortunately, such skills may be far beyond what people are willing or capable of learning. This is one motivation for the use of modeling techniques such as regression trees or analogy. Estimation methods based on complex modeling techniques will also require more and better tools support to have a chance to get accepted and applied across the board.

7.2 Technical Considerations

Intended Applications: Some commercial tools are inherently limited in terms of the applications they support. For example, if estimates are point estimates or only provide simple estimation intervals, it is then difficult if not impossible to perform some risk analysis related to a given budget, effort, or schedule. On another note, if a model is requires size measurement that can be obtained only at later stages of development, then no early estimation can be performed.

Availability of Project data: To various degrees, data driven estimation methods as well as composite methods require project data (i.e., cost drivers, effort, project size). Such data may not exist in sufficient amount or may not be exploitable due to inconsistencies and incompleteness in the data collection.

Validity of Data: Though data may be available and its quality may be acceptable, organizations, their processes, and their products may change over time. Usually, only a subset of the existing project data set can be used with a reasonable level of confidence. Such limitations may prevent the use of data-driven estimation methods.

Characteristics of Data: Because of differences in data distributions, the underlying relationships we try to model, and the type of data collected, the ability of model-based estimation methods to produce accurate estimates may vary considerably. It is often difficult to determine beforehand, i.e., before we have actually tried them on the data at hand. For example, regression trees as well as OSR are good at dealing with categorical cost drivers, in a context where their impact on resource expenditure interact, i.e., their impact depends on the other cost drivers' values.

8 Typical Applications

This section describes three application contexts where software resource modeling and estimation is necessary. The first subsection illustrates the outsourcing of software projects. The second subsection summarizes the cost estimation issues from the software supplier's viewpoint. The third subsection describes cost estimation issues when managing software projects.

8.1 Outsourcing Software Projects

Companies can outsource a new system to be developed, maintained and operated by a supplier. Reasons for outsourcing are very different. An organization may decide to outsource because of a temporary shortage in its software development personnel. Another organization may assume that a specialized supplier will provide better services than internal development. Last, developing software may not be considered as a part of an organization's core business and is therefore outsourced.

The outsourcing of development personnel decreases the staffing level of an organization and this is assumed to reduce costs. Usually, suppliers predict large cost reductions and quality improvements. Therefore, it is crucial to appropriately assess incoming, competing sub-contractor bids.

To appropriately assess incoming bids, it is necessary to know about the important project characteristics and expected effects that may drive the cost for projects. When accumulating data from a number of subcontractors, it is possible to build prediction models and create a baseline of comparison for benchmarking purposes. Resource modeling is therefore expected to help negotiate and decide about suppliers in future outsourced projects. Examples regarding benchmarking and prediction models based on the European Space Agency subcontractor database can be found in (Briand et al., 1998b) and (Briand et al., 2000), respectively.

8.2 Supplying Software Products or Services

Supplying organizations have to develop marketing strategies and strategies for the acquisition and tendering process to match the needs of a client organization.

Market pressure increases as the customer becomes more critical towards investments, and put higher demands on their suppliers. A subcontractor's offer for a product or a service has to be competitive with those of other suppliers. To be able to provide a realistic bid, a supplier needs a good estimate of the cost for the whole project. Usually these preliminary estimates are very difficult to obtain, because only very little is know about the project and the product at such an early stage. Thus, the level of uncertainty is relatively high and some analysis needs to be performed to identify the level of risk associated with a bid. This is important as suppliers that deliver insufficient quality or charge too much will disappear from the market or will be taken over by other suppliers.

8.3 Project Management

Most software organizations are working on a fixed price contract basis and within tight schedules. At the start of a project a project manager needs to estimate the resources needed to perform the project and negotiate adequate budget and staffing plans. Once a software project has started, detailed effort and schedule estimates are needed to plan, monitor, and control the project. Throughout the project, the actual effort and schedule is to be checked against the planned values. Risks need to be assessed and preventive actions should be planned. After project completion cost estimation models are used to perform a validated comparison of the project's performance (benchmarking) with similar past projects. In order to perform all of these tasks, a resource estimation model needs to be built that is tailored to the project-specific development process characteristics. The model is usually based on the project managers' expertise and data collected during the project run.

To monitor and control the project properly, one needs (1) a description of the actual development process, (2) data about cost factors considered as most important, and (3) effort actually spent on major activities. This helps to better understand relationships among project attributes and to capture them through resource models. The data collection needs to be defined based on a process model that describes the relevant activities, artifacts, and resources on an appropriate level of granularity.

Project monitoring aims at identifying possible departures from planned schedule, effort, and development progress. If, for example, more effort on requirements definition than expected is spent in the coding phase, the problem should be investigated (e.g., the customers changed many requirements late in the project) and corrective actions taken (e.g., re-negotiation based on hard data). Project control aims at taking corrective actions after deviations from the planned values and based on objective information regarding the causes of such departures.

The characterization of effort expenditures aims at finding out where, when, and how resources are spent on a project. The most common usage of this data is improvement and planning. If enough data from various projects is available, staff and effort planning of new projects is facilitated. It is then possible to estimate which proportion of the overall estimated effort will be spent in each phase, for which activity, and how much effort has to be spent for completing a product.

When estimating the resource expenditure of a project, the most relevant impacting factors on have to be taken into account and their impact needs to be quantified either in a model or during expert estimation. Usually important cost factors differ across companies and/or application domains and they need first to be identified in a given resource estimation context.

To use a resource estimation model at the beginning of a running project, the project manager assigns values to each relevant factor in the model. A project manager is, however, often uncertain about the exact value that a factor should take. This is especially true at early development stages. This uncertainty can be represented as a distribution of values and accounted for in the estimates. Though the output of resource models is often a single value, this is misleading as it falsely conveys an impression of high accuracy. It is important to look at the estimate distribution as well. Not only the distribution resulting from the uncertainty of the model but also the variance resulting from the uncertainty in the values of the model inputs. One can then perform some realistic risk assessment and, for example, determine the probability of overrunning a given budget.

8.4 Productivity Benchmarking

Another application of resource estimation models is productivity benchmarking. For example, when a project is completed, the organization may want to assess how well this project performed with respect to productivity compared to other similar projects. Though other factors than productivity are of interest, this is important as that particular project may have used a different set of techniques or processes and it would be of interest to know whether any visible, practically significant effect can be observed.

The way estimation models are used is to predict the expected productivity distribution from similar past projects. This can then be used as a baseline of comparison. The advantage of using a model as opposed to just looking at the overall project productivity distribution is that we remove the productivity variation due to project factors that are not of interest and that make the comparison between projects meaningless, e.g., application domain. If projects show a strong departure from the expected productivity distribution, then this should be investigated in order to determine whether anything can be learnt to improve development practices.

Other techniques such as Data Envelopment Analysis (DEA) have been used for benchmarking (Myrtveit and Stensrud, 1999b). The difference with using estimation models is that DEA attempts to characterize the *optimal* productivity that can be expected in a given environment, for a given project size. In this case, project productivity is compared to the best productivity that can be expected, not the expected productivity distribution. The advantage is that one may determine a realistic percentage of potential productivity improvement by adopting best practices. The drawback is that the method is sensitive to outliers and does not remove the effects of factors that cannot be controlled or managed.

9 Future Directions

This section discusses crucial issues in software resource estimation, from both research and practical perspectives.

9.1 Research Directions

From a scientific perspective, several improvement issues require to be investigated. First, methods that combine expert knowledge with data-driven techniques are key. They may allow individual organizations to develop specific resource estimation models while alleviating the stringent requirements for project data. Such methods make use of all information available in organizations and therefore make practical and economic sense. So far, composite methods achieved promising results. These are methods based on subjective effort estimation (Höst and Wohlin, 1998), modeling based on expert knowledge elicitation (Briand et al., 1998a), and methods combining expert opinion and project data (Chulani et al., 1999). Thus, to understand how experts reason about schedule, resources, or cost plays an important role in the software cost estimation research. Therefore, collaborations between software resource estimation researchers and researchers in cognitive sciences are needed.

Comprehensive and consistent comparisons of cost estimation methods are still rare (Briand et al. 2000). Thus, conclusions of general validity are still difficult to draw. Researchers need to consistently compare estimation methods across many data sets in order to be able to define guidelines about what kind of models to use for a given environment. This together with some guidance how to use the models would be a valuable contribution to improve current cost estimation practices. Simulation (Pickard et al., 1999) can help address this issue but it is limited because it requires the researchers to make somewhat arbitrary choices in the distributions and structures present in the datasets.

In addition, as discussed by (Stensrud and Myrveit 1999a), resource estimation methods are typically not used in isolation but as a support tool for human estimators. It is therefore important that research in resource estimation takes into account this human factor component.

Most of the research has addressed the estimation of development effort. However, maintenance, COTS- and reuse-based development are common practices and require investigation. In other words, software estimation research needs to be widened to cover new grounds.

Also, as noted above, project sizing is key to accurate software estimates. Though this has been the focus of significant research, there is still no clear picture as to how should organizations tackle this issue. Some researchers and practitioners are proponents of standardized sizing methods, such as Function Points, which have the advantage to be comparable across organizations. However, other software engineering professionals share a more pragmatic view, which consists in organizations to develop their own sizing procedures based on available development artifacts. This requires careful data collection and analysis and is more expensive to develop. But it is likely to lead to more adequate and less expensive sizing procedures. Further research is required to shed light on this crucial issue of project sizing.

9.2 Better Industry Practices

From a practical perspective, regardless of the specific estimation methods to be used, many software organizations need to make a first step and define and integrate an explicit, repeatable estimation process into their development practices. This may require the definition of procedures for data collection and analysis, the selection of appropriate estimation methods, and the definition of appropriate system-sizing mechanisms. A defined estimation process can be analyzed and improved, and this is why it is a key step in the right direction, though the development or selection of an appropriate estimation method may not be an easy task.

A data collection process defines what data to collect, and what procedures to follow in order to ensure efficient data collection, analysis, and exploitation. For organizations' internal development, local project databases need to be developed. Outsourcing organizations should aim at establishing multi-organization databases for their contractors' projects.

The selection of appropriate estimation methods and their application is also a part of a defined estimation process. The aim should be to apply different methods and models at different stages in the development process, depending on the information available. Feedback should be provided comparing the estimates with the actual values and reasons for deviations should be analyzed. This would lead, in a specific application context, to the stepwise refinement of estimation methods.

Software sizing, especially in the early phases of development, requires organizations to look at the information available at various project stages, both in form and content, and determine efficient and repeatable sizing procedures. This may lead to the tailoring of available sizing techniques or to new, possibly more specialized, techniques.

10 Conclusions

This overview article has shown that, although a number of alternatives exist to support the applications of software resource estimation, estimation methods all tend to have drawbacks and advantages. The appropriate selection of a method is therefore context dependent and this is why we have focused in this article on providing the readers with tools necessary to make an appropriate selection.

Though many estimation methods have been developed over the years, we have also seen that appropriately using such methods is not easy, and may not always lead to accurate resource expenditure estimates. But no matter how imperfect the state of the art, one needs a method as ad-hoc estimation processes are the worst alternative one may imagine. At least, a defined estimation method, as unsatisfactory as it may be, has the advantage to be repeatable and to allow for stepwise improvement.

Nevertheless, a wealth of methods and tools are available for anybody in charge of estimating resource expenditures on software projects. Though the development of appropriate, accurate estimation methods never comes free, one or more existing approaches should suit most software development environments.

Over the years, one difficulty has been the lack of publicly available and carefully collected project data and the rare collaborations between industry and researchers. Without one or the other, the software resource estimation field is bound to progress at a slow pace. Another issue has been the excessive focus of research on building data-driven models. We think that, for many practical and technical reasons discussed above, this cannot be the answer for most organizations. We believe that a shift in research priorities, as discussed in Section 9, is therefore necessary.

11 Acknowledgements

We would like to acknowledge the DaimlerChrysler research center in Ulm, Germany, for partially funding this work. Lionel Briand's work has also been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

12 Bibliography

T.K. Abdel-Hamid, S.E. Madnick. Software Project Dynamics: An Integrated Approach. Prentice Hall, Englewood Cliffs, March 1993, NJ, USA, 1991

A. Abran, M. Maya, J-M Desharnais, D. St-Pierre. Adapting Function Points to Real-Time Software. American Programmer. 32-42 (November 1997)

A. J. Albrecht. Measuring Application Development Productivity. Proc. Of the Joint SHARE, GUIDE, and IBM Application Developments Symposium .83-92, (1979).

A. J. Albrecht, J. E. Gaffney. Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering. Vol. 9, no. 6, 639-648, (November, 1983)

M. Andolfi, M. Costamanga, E. Paschetta, G. Rosenga. A multicriteria-based methodology for the evaluation of software cost estimation models and tools. IFPUG 96, (1996)

R. D. Banker, S. M. Datar, C. F. Kemerer. A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects. Management Science, Vol. 37, no. 1, 1-18, (1991)

R.D. Banker, R.J. Kaufman, R. Kumar. An empirical test of object-oriented output measurement metrics in a computer aided software engineering (CASE) environment. Journal of Management Information Systems. Vol. 8, no. 3 (Winter 1991-92).

R. Banker, C. Kemerer. Scale Economies in New Software Development. IEEE Transactions on Software Engineering, vol 15, no. 10, 1199-1205, (October, 1989)

J. Bailey, V.R. Basili. A Meta-Model for Software Development Resource Expenditures. Proc 5th International Conference on Software Engineering, 107-116, (July, 1981)

R.A. Baron, J. Greenberg. Behavior in Organizations. Allyn & Bacon (1990)

V. Basili, D. Rombach. The TAME project: Towards improvement-oriented software environments. IEEE Transactions on Software Engineering, vol. 14, no. 6, 758-773, (1988)

W.D. Berry, S. Feldman. Multiple Regression in Practice. Sage Publications, Quantitative Applications in the Social Sciences 50,1985

R. Bisio, F. Malabocchia. Cost Estimation of Software Projects through Case Based Reasoning. Case Based Reasoning Research and Development, Proc. International Conference on Case-Based Reasoning, 11-22, (1995)

B. W. Boehm, R. W. Wolverton. Software Cost Modeling: Some Lessons Learned. Journal of Systems and Software, vol. 1, (1980)

B. Boehm. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ, 1981

B.W. Boehm. Software Engineering Economics. IEEE Transactions on Software Engineering, vol. 10, no. 1, (January, 1984)

B. W. Boehm, B. Clark, E. Horowitz, C. Westland. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. Annals of Software Engineering, vol. 1, 57-94, (1995)

B. Boehm, R. Madachy, R. Selby. The COCOMO 2.0 Software Cost Estimation Model: A Status Report. American Programmer, vol. 9, no. 7, (July, 1996)

Bournemouth University: Automated Project Cost Estimation Using Analogies: The ANGEL project <http://dec.bournemouth.ac.uk/dec_ind/decind22/web/Angel.html>

G. Bozoki. An Expert Judgement Based Software Sizing Model. Lockheed Missiles & Space Company and Target Software, Sunnyvale, California. <http://www.targetsoft-ware.com>

P.A. Braun, I. Yaniv. A Case Study of Expert Judgement: Economists' Probability versus Base-rate Model Forecasts. Journal of Decision Making, vol. 5, 217-231, (1992)

L. Breiman, J. Friedman, R. Ohlsen, C. Stone. Classification and Regression Trees, Wadsworth & Brooks/Cole Advanced Books & Software, 1984

L. C. Briand, V. R. Basili, W. M. Thomas. A Pattern Recognition Approach for Software Engineering Data Analysis, IEEE Transactions on Software Engineering, vol. 18, no. 11, 931-942, (November 1992)

L.C. Briand, V.R. Basili, C.L. Hetmanski. Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development. International Symposium on Software Reliability Engineering, ISSRE, (October, 1992b)

L. C. Briand, V. R. Basili, C. J. Hetmanski. Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components. IEEE Transactions on Software Engineering, vol. 19, no. 11, 1028-1044, (November 1993)

L. C. Briand, K. El Emam, F. Bomarius. COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment, Proc. of the 20th International Conference on Software Engineering, 390-399, (April, 1998a)

L.C. Briand, K. El Emam, I. Wieczorek. A Case Study in Productivity Benchmarking: Methods and Lessons Learned, Proc. ESCOM-ENCRESS 98 Project Control For 2000 and Beyond, Rome, 4-14, (1998b)

L.C. Briand, K. El Emam, I. Wieczorek. Explaining the Cost of European Space and Military Projects. Proc. 21st International Conference on Software Engineering. ICSE 99, Los Angeles, (1999a)

L.C. Briand, K. El Emam, K. Maxwell, D. Surmann, I. Wieczorek. An Assessment and Comparison of Common Cost Software Project Estimation Methods, Proc. International Conference on Software Engineering, ICSE 99, 313-322, (1999b)

L.C. Briand, R. Kempkens, M. Ochs, M. Verlage, K. Lünenburger. Modeling the Factors Driving the Quality of Meetings in the Software Development Process. Proc. ESCOM-SCOPE 99, 17-26, (1999c)

L.C: Briand, T. Langley, I. Wieczorek. A replicated Assessment and Comparison of Common Software Cost Modeling Techniques. Accepted for Publication in Proc. International Conference on Software Engineering, ICSE 2000, June, (2000)

R.N. Britcher, J.E. Gaffney. Reliable size estimates for software systems decomposed as state machines. Proc. COMPSAC, IEEE Computer Society Press, New York (1985)

F.P. Brooks. The Mythical Man Month. Addison-Wesley, Reading MA, USA, 1975

B. Cheng, D.M. Titterington. Neural Networks: a Review from a Statistical Perspective. Statistical Science, Vol. 9, No. 1, 2-54, (1994) Chulani S., Boehm B., Steece B. Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transactions on Software Engineering, 25,4 (1999)

S.D. Conte, H.E. Dunsmore, V.Y.Shen. Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company, Inc. 1986

A.M.E. Cuelenaere, M.J.I.M. van Genuchten, F.J. Heemstra. Calibrating a Software Cost Estimation Model: Why and How. Information and Software Technology, vol. 29, no. 10, 558-567, (1987)

S. J. Delany, P. Cunningham, W. Wilke. The Limits of CBR in Software Project Estimation. Proc. the 6th German Workshop on Case-Based-Reasoning. L. Gierl, M. Lenz (eds.), Berlin, (March, 1998)

T. De Marco. Controlling Software Projects. Yourdan Press, 1982

W.R. Dillon. Multivariate Analysis: Methods and Application. New York: Wiley, 1984

J. Dolado, L. Fernandez. Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation. Proc. the INSPIRE III, Process Improvement Through Training and Education, (1998)

N. E. Fenton, S. L. Pfleeger. Software Metrics – A Rigorous and Practical Approach. 2nd. Edition, Thomson Computer Press, 1996

G.R. Finnie, G.E. Wittig. AI Tools for Software Development Effort Estimation. Proc. of the Conference on Software Engineering: Education and Practice, 113-120, (1996)

G.R. Finnie and G.E. Wittig and J-M. Desharnais. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. Journal of Systems and Software, vol. 39, no. 3, 281-289, (December, 1997)

Fraunhofer IESE, Institute for Experimental Software Engineering <http://www.iese.fhg.de>

F. R. Freiman, R. E. Park. PRICE Software Model-Version 3: An Overview. Proceedings of the IEEE-Pinx Workshop on Quantitative Software Methods, 32-41, (October, 1979)

M. van Genuchten. Why is Software Late? An Empirical Study of Reasons For Delay in Software Development. IEEE Transactions in Software Engineering, vol. 17, no. 6, (June 1991)

A. Gray, S. MacDonell. Software Metrics Data Analysis – Exploring the Relative Performance of Some Commonly Used Modeling Techniques. Empirical Software Engineering, vol. 4, 297-316, (1999)

D. Greves, B. Schreiber.The ESA Initiative for Software Prodcutivity Benchmarking and Effort Estimation. ESA Bulletin, no. 87, (August 1996) <http://esapub.esrin.esa.it>

F.J. Heemstra, R.J. Kusters. Function Point Analysis: Evaluation of a Software Cost Estimation Model. European Journal of Information Systems. Vol. 1, no. 4, 229-237 (October1991)

F.J. Heemstra. Software Cost Estimation, Information and Software Technology, vol. 34, no. 10, (October, 1992)

J. Hihn, H. Habib-agahi. Cost Estimation of Software Intensive Projects: A Survey of Current Practices, Jet Propulsion Laboratory/California Institute of Technoloy, 1991

S.J. Hoch, D.A. Schkade. A psychological approach to decision support systems. Managemenr and Science, vol. 42, 51-64 (1996)

M. Höst, C. Wohlin. An Experimental Study of Individual Subjective Effort Estimation and Combinations of the Estimates. Proc.20st International Conference on Software Engineering, ICSE 98, 332-339, (1998)

R. T. Hughes . Expert judgement as an estimating method. Information and Software Technology, vol. 38, no. 2, 67-75, (1996)

R.T. Hughes, A. Cunliffe, F. Young-Marto. Evaluating software development effort model building techniques for application in a real-time telecommunication environment. IEE Proceedings Software, vol. 145, no. 1, 29-33, (February, 1998)

International Function Point User Group (IFPUG): Function Point Counting Practices Manual, Release 4.0, Westerville: IFPUG Inc. <http://www.ifpug.org/> 1994

R. Jeffery, G. Low. Calibrating Estimaton Tools for Software Development. Software Engineering Journal, 215-221, (July, 1990)

R. Jeffery, J. Stathis. Function Point Sizing: Strucure, Validity and Applicability. Empirical Software Engineering 1, 11-30 (1996).

A.M. Jenkins, J.D. Naumann, J.C. Wetherbe. Empirical Investigation of Systems Development Practices and Results. Information and Management, vol. 7, 73-82, (1984)

R.W. Jensen. A Comparison of the Jensen and COCOMO schedule and cost estimation models. Proceedings of International Society of Parametric Analysis, 96-106, (1984)

T.C. Jones. Programming Productivity. New York: McGraw-Hill, 1986

T.C. Jones. Applied Software Measurement. McGraw-Hill, 1997

T.C. Jones. Estimating Software Costs. McGraw-Hill, 1998

M. Jørgensen. Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. IEEE Transactions on Software Engineering, 21, 8, 674-681, (August, 1995)

L. Kaufman, P. Rousseeuw. Finding Groups in Data. An Introduction to Cluster Analysis. John Wiley & Sons, Wiley Series in Probability and Mathematical Statistics, 1990.

C. F. Kemerer. An Empirical Validation of Software Cost Estimation Models. Communications of the ACM, vol. 30, no. 5, 417-429, (May 1987)

Kitchenham, B.A., Taylor, N. R. Software project development cost estimation. The Journal of Systems and Software vol. 5 267- 278, (1985)

B. Kitchenham, B. de Neumann. Cost Modeling and Estimation. In: Software Reliability Handbook, P. Rook (ed.), Elsevier Applied Science, NY, 1990, pp. 333-376

B. Kitchenham. Software Development Cost Models. In: Software Reliability Handbook, P. Rook (ed.), Elsevier Applied Science, NY, 1990, pp. 487-517

B. Kitchenham, K. Känsälä. Inter-Item Correlations among Function Points. Proc. International Software Metrics Symposium. IEEE Computer Society Press. 11-14, (1993)

B. Kitchenham. A Procedure for Analyzing Unbalanced Datasets. IEEE Transactions on Software Engineering, vol. 24, no. 4, 278-301, (1998)

T.M. Koshgoftaar, E.B. Allen, A. Naik, W. Jones, J. Hudepohl. Using classification trees for software quality models: lessons learned. International Journal of Software Engineering and Knowledge Engieering, vol.9, no. 2, 217-231 (1999)

R.J. Kusters, M.J.I.M. van Genuchten, F.J. Heemstra. Are Software cost-estimation models accurate? Information and Software Technology, vol. 32, no. 3, (April, 1990)

J. M. Lambert. A Software Sizing Model. Journal of Parametrics. Vol. 6 no. 4 (1986)

A. Lederer, J. Prasad. Nine Management Guidelines for Better Cost Estimating, Communications of the ACM, vol. 35, no. 2, (February, 1992)

A. Lederer, J. Prasad. Information Systems Software Cost Estimation. Journal of Information Technology, vol. 8, 22-33, (1993)

A. Lederer, J. Prasad. Causes of Inaccurate Software Development Cost Estimates. Journal of Systems and Software, 125-134, (1995)

A. Lederer, J. Prasad. A Causal Model for Software Cost Estimating Error. IEEE Transactions on Software Engineering. Vol. 24, no. 2, (February, 1998)

B. Londeix, Cost Estimation for Software Development. International Computer Science Series, Addision-Wesley, 1987

G. Low, R. Jeffery. Function Points in the Estimation and Evaluation of the Software Process. IEEE Transactions on Software Engineering. Vol. 16, no. 1, 215-221, (January 1990)

S.G. MacDonell. Comparative Review of Functional Complexity Assessment Methods for Effort Estimation. Software Engineering Journal, 107-116, (1994)

D.C. MacGregor, S. Lichtenstein. Problem Structuring aids for Quantitative Estimation. Journal of Behavioral Decision Making, vol. 4, no. 2, 101-116, (1991)

J. E. Matson, B.E. Barrett, J.M. Mellichamp. Software Development Cost Estimation Using Function Points. IEEE Transactions on Software Engineering. Vol. 20, no. 4, 275-287, (April 1994)

K. Maxwell, L. Van Wassenhove, S. Dutta. Software Development Productivity of European Space, Military, and Industrial Applications. IEEE Transactions on Software Engineering, vol. 22, no 10, 706-718, (October, 1996)

K. Maxwell, P. Froselius. Benchmarking Software Development Productivity. IEEE Software, 80-88, (January-February, 2000)

M. Meyer, J. Booker. Eliciting and Analyzing Expert Judgement – A Practical Guide. Knowledge Based Systems Volume 5, Academic Press, 1991

Y. Miyazaki, K. Mori. COCOMO Evaluation and Tailoring. Proceedings of the 8th International Conference on Software Engineering, pp. 292-299, (1985)

Y. Miyazaki, M. Terakado, K. Ozaki, H. Nozaki. Robust Regression for Developing Software Estimation Models. Journal of Systems and Software. Vol. 27, 3-16, (1994)

T. Mukhopadhyay, Vicinanza, S.S., Prietula, M.J. Examining the feasibility of a case-based reasoning model for software effort estimation. MIS Quarterly, 155-171, (June 1992)

T. Mukhopadhyay, S. Kekre. Software Effort Models for Early Estimation of Process Control Applications, IEEE Transactions on Software Engineering, vol. 18, no. 10, (915-924), (October, 1992)

I. Myrtveit, E. Stensrud. A Controlled Experiment to Assess the Benefits of Estimation with Analogy and Regression Models. IEEE Transactions on Software Engineering, vol. 25, no.4, (August, 1999a)

I. Myrtveit, E. Stensrud. Benchmarking COTS Projects Using Data Envelopment Analysis, Proc. the 6th METRICS 99 Symposium, 269-278, (1999b)

J.K. Navlakha. Choosing a Software Cost Estimation Model for Your Organization: A Case Study. Information & Management, vol. 18, 255-261, (1990)

P.V. Norden. Curve Fitting for a Model of Applied Research and Development Schedunling. IBM Journal Research and Development, vol. 2, no. 3, (July, 1958)

E.A. Nelson. Management Handbook for the Estimation of Computer Programming Costs. SDC, TM-3224, (October, 1966)

P. Nesi, T. Querci. Effort Estimation and Prediction fo Object-Oriented Systems, Journal of Systems and Software, vol. 42, 89-102, (1998)

L. Pickard, B. Kitchenham, S. Linkman. An Investigation of Analysis Techniques for Software Datasets. Proc. The METRICS 99 Symposium, 130-142, (November, 1999)

A. Porter, R. Selby. Evaluating techniques for generating metric-based classification trees. Journal of Systems and Software, vol. 12, 209-218, (July, 1990)

Price Systems: <http://www.pricesystems.com>

L.H. Putnam. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering, vol. 4, no. 4, 345-361, (1978)

L. Putnam. Size Planner, an Automated Sizing Model. Third COCOMO User's Group Meeting, (November 1987)

L.H. Putnam, W. Myers. Measures for Excellence, Reliable Software on Time, Within Budget, Yourdan Press, Englewood Cliffs N.J., 1992

J.R. Quinlan. Induction of Decision Trees. Machine Learning, vol.1 no.1, 81-106, (1986)

P.J. Rousseeuw, A.M. Leroy. Robust Regression and Outlier Detection. Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1987

H.A. Rubin. A Comparison of Cost Estimation Tools. Proc. 8th International Conference of Software Engineering. A Panel Discussion. (1985)

T. Saaty. Multicriteria Decision Making: The Analytic Hierarchy Process, RWS Publications, 1996

Salford Systems <www.salford-systems.com/index.html>

L.D. Schroeder, D.L. Sojoquist, P.E. Stephan. Understanding Regression Analysis - An introductory Guide, Series: Quantitative Applications in the Social Sciences, 57, SAGE University Paper, 1986

M. Shepperd, C. Schofield, B: Kitchenham. Effort Estimation Using Analogy, Proceedings of the 18th international Conference on Software Engineering- ICSE 96, 170-175, (May, 1996)

M. Shepperd, C. Schofield. Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, vol. 23, no 12, 736-743, (1997)

Software Productivity Research: <http://www.spr.com/index.htm>

P. Spector. Summated Rating Scale Construction. Sage Publications, 1992

K. Srinivasan, D. Fisher. Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering, vol. 21, no. 2, (1995)

Stata Corporation <http://www.stata.com/>

STTF (Software Technology Transfer Finland), <http://www.sttf.fi/index.html>

R.D. Stutzke. Software Estimating Technology: A Survey, Crosstalk: The Journal of Defense Software Engineering. Vol. 9, no. 5, 17-22, (May, 1996)

E. Stensrud, I. Myrtveit. Human Performance Estimation with Analogy and Regression Models. Proc. the 5th METRICS 98 Symposium, 205-213, (1998)

G.H. Subramanian, S. Breslawski. Dimensionality Reduction in Software Development Effort Estimation. Journal of Systems and Software, vol. 21, no. 2, 187-196, (1993)

G. Subramanian, S. Breslawski. The Importance of Cost Drivers Used in Cost Estimation Models: Perceptions of Project Managers. Proc. IRMA 503, (1994)

R.C. Tausworthe. The Work Breakdown Structure in Softwrae Project Management, Journal of Systems and Software 1, 181-186, (1980)

S. Vicinanza, T. Mukhopadhyay, M. J. Prietula . Software-Effort Estimation: An Exploratory Study of Expert Performance. Information Systems Research, vol.2, no.4, 243-262, (1991) University of Kaiserslautern, CBR-Works, 4.0 beta. Research Group "Artificial Intelligence – Knowledge-Based Systems", <http:// www.agr.infomratik.uni-kl.de/~lsa/CBRatUKL.html>

University of New South Wales, Center for Advanced Empirical Software Research (CAESAR), ACE tool: <http://www.fce.unsw.edu.au/caesar/tools/ace.html>

D. Vose. Quantitative Risk Analysis - A Guide to Monte Carlo Simulation Modeling. Chichester: John Wiley and Sons. 1996

F. Walkerden, R. Jeffery. Software Cost Estimation: A Review of Models, Process, and Practice. Advances in Computers, vol. 44, 59-125, (1997)

Walkerden F., Jeffery R. An Empirical Study of Analogy-based Software Effort Estimation. Empirical Software Engineering, 4, 2, 135-158, (June 1999)

C.E. Walston, P.C. Felix. A Method of Programming Measurement and Estimation. IBM Systems Journal, 55-73, (1977)

S.A. Whitmire. 3D Function Points: Applications for Object-Oriented Software, Boeing Commercial Airplane Group, 1998

R.W. Wolverton. The cost of Developing Large-Scale Software. IEEE Transactions on Computers, C-23(6), 615-636, (1974)

D. Wrigley, S. Dexter. Software Development Estimation Models: A Review and Critique. Proc. the ASAC Conference, University of Toronto, (1987)

J. Zaruda. Introduction to Artificial Neural Systems. St- Paul, MN: West, 1992