The Application of Case-Based Reasoning to Early Software Project Cost Estimation and Risk Assessment

Sarah Jane Delany
Department of Computer Science
DIT Kevin Street, Dublin

Pádraig Cunningham
Department of Computer Science
Trinity College Dublin

TCD-CS-2000-10

Abstract

In this paper we assess the applicability of case-based reasoning to the difficult problem of early software project cost estimation. We conclude that a comprehensive case representation is not available early in the project and suggest instead that the objective should be risk assessment rather than cost estimation. In reaching this conclusion the existing techniques for cost estimation are discussed and evaluated. A case representation capturing the available predictive features for early estimation is identified and presented. The lack of features to predict size early in the development life cycle indicates a limitation of the conventional CBR model – and indeed any knowledge-based approach. If a complete problem representation is not available an automated reasoning mechanism will not be able to produce good cost estimates. The alternative we propose is to focus on a measure called the productivity coefficient rather than the expected effort. The productivity coefficient gives a measure of the potential risk revealed by the characteristics of a project compared with previous project experiences. The utility of this approach is described in a sample scenario.

1 Introduction

Case-Based Reasoning (CBR) is a problem solving technique based on the reuse of past experiences. For this reason there is considerable optimism about its use in difficult problem solving areas where human expertise is evidently experience based. It is particularly suitable in *weak theory* domains, that is on types of problems where cause and effect are not well understood.

Software cost estimation is such a weak theory domain and much research has been done on the use of CBR in this area (Mukhopadhyay *et al.* 1992; Prietula *et al.* 1996; Bisio & Malabocchia 1995; Finnie *et al.*1997b). Efforts at software project cost estimation later on in the software development life cycle are most successful because parameters defining the size of the project are available at that stage. However, accurate estimation early in the project has the greatest strategic impact and is dependent on human expertise. Since this expertise is experience based it should be possible to produce the same competence in a CBR system. In fact a difficult problem such as this highlights some limitations of the conventional CBR model. In particular, the dependence on a complete problem representation causes problems. Even factors that the human expert is considering implicitly must be represented as explicit features in the CBR system.

In this paper we present a study of the use of CBR for software cost estimation early in the project life cycle. We show the possible case representations that can be drawn from the wider literature on cost estimation and show how these

representations fail to adequately capture project size. At present it appears impossible to identify features early in the life cycle that define the size of the project. Instead we propose that the best alternative is to pursue a solution that produces an estimate of potential risk for a specific project based on previous experiences. This approach is similar in philosophy to that of Madachy (1995 & 1997) where cost factors and used to estimate risk rather than effort. Madachy's approach is rule-based while the approach presented here is case-based. It is to be expected that a case-based approach will have knowledge engineering advantages over the rule-based alternative. That is, given an adequate problem representation it is easier to populate a case-base with cases that encode the knowledge implicitly than it is to come up with the rules that capture the causal interactions in the problem domain. However it must be acknowledged that the data capture problem of compiling such cases in a software development environment is considerable.

Section 2 of the paper discusses and evaluates the different techniques in use for cost estimation. Section 3 describes the process of case-based reasoning and its applicability to this problem. Section 4 presents the proposed case representation which is evaluated and discussed in Section 5. Section 6 presents the conclusions and future work.

2 Cost Estimation

There are a number of different techniques for estimating software development costs described in the literature. These include the use of

- algorithmic models which predict estimates of effort and duration using parametric equations;
- expert judgement involving predictions based on the skill and experience of one or more experts and
- analogy involving the comparison of one or more completed projects with details of a new project to predict cost and duration.

2.1 Algorithmic Models

Algorithmic models use mathematical formulae to predict effort and duration as a function of a number of variables. They are usually derived from detailed statistical analysis of data collected from completed software development projects. There are a variety of algorithmic techniques in use (Heemstra 1992) but Kitchenham categorises them into two groups – the empirical factor models and the constraint models (Kitchenham 1991). Empirical factor models provide an estimate of a value of a cost parameter. They are derived by applying statistical techniques to data from previous projects. The main empirical factor model in use is the COCOMO model (Boehm 1981; Boehm 1984). Other models falling into this category are the ESTIMACS and PRICE SP proprietory models (Chatzoglou & Macaulay 1996), the TRW Wolverton model (Wolverton 1974), Softcost (Tausworthe 1981) and DOTY (Herd *et al.* 1977). The second categorisation of models, constraint models, demonstrate the relationship over time between the various cost factors such as effort, cost, schedule or staffing levels. These models can depict the effects of varying the schedule against the effort and staffing levels. The main constraint model in use is Putnam's SLIM model (Putnam 1978) Other models in this category include the Jensen model (Jensen 1983; Jensen 1984) and COPMO (Thebaut & Shen 1984).

Almost every model mentioned above uses an estimate of the number of source lines of code or the size of object code as a size driver. In addition to the obvious difficulty of estimating this in advance, is the variation in counting methods which may change the number by a wide margin (Jones 1986). There is another categorisation of algorithmic models that provide an alternative sizing technique to counting lines of code. These models promote the use of a measure of the functionality of the system as a measure of its size. These models use attributes such as number of input/output files, reports, displays, or particular specification or design elements as inputs to the sizing phase. The advantages here are that these size drivers are known earlier in the development life cycle and can be estimated from a design specification with higher certainty than lines of code. However, there are problems also with function points, firstly the effort involved in the collection of the input data and the difficulty in getting consistent estimates from different individuals performing the counting (Kemerer 1989, Symons 1988). The most common models of this type are Albrecht's Function Point Analysis (Albrecht 1979; Albrecht & Gaffney 1983), De Marco's Function Bang (De Marco 1982) and the SPQR-20 model (Jones 1986), commercially available as the Checkmark product.

Algorithmic models are derived from project data in specific software environments and are then used in software environments potentially very different from that in which they were derived. This points to two problems. Firstly, the need to ensure that any values input to the models are consistent with the model requirements and secondly, the need to reflect the basic characteristics of the software environment. This all indicates a need for calibration. This is supported in the cost estimation literature as there is significant evidence that calibration with an organisation's historic project data is crucial (Kemerer 1987; Jack & Mannion 1995; Cuelanaere *et al*. 1987; Kitchenham & Taylor 1985).

Validation of the algorithmic cost models is difficult to do, as it requires large amounts of data from completed projects. There is a lack of data on past projects as data collection is not common within the software development community. In spite of this there have been a number of studies attempting to evaluate the effectiveness of the models (Kemerer 1987; Rubin 1985; Mohanty 1981; Bredero *et al.* 1989; Heemstra 1992; Kusters *et al* 1990). This research into the use of the algorithmic models for cost estimation has shown that the models perform badly.

There are a number of reasons for the failure of the algorithmic cost models. The surveys referenced above agree that poor results from the models are due in part to using the models incorrectly. Much of the time models are used without calibration (Heemstra 1992). In addition, the input parameters to the models used in the calculation of the estimate are subjective. This can mean different results when applied to the same problem, as demonstrated in the surveys.

Aside from these, there are intrinsic reasons for the cost estimation algorithmic models not performing adequately. A majority of models do not support calibration. Associated with this is the lack of availability of data from previous software projects which is necessary for the calibration of those models which do support calibration. However, the overriding reason for failure of these models is that the models have been generalised. They were derived from data available *post hoc* from completed projects. These datasets may incorporate characteristics and peculiarities that are difficult to assess at the start of another project (Kitchenham 1991). The studies of algorithmic models also do their estimating after the fact.

This is unrealistic when it comes to applying a model to a new project as the factors that influence the cost may not emerge until later on in the development process and may not be available at the time of estimation.

2.2 Expert Judgement and Estimation by Analogy

The other techniques used to estimate the costs of software development are 'expert judgement' where the predictions are ostensibly based on the skill, understanding and experience of one or more experts and 'estimation by analogy' involving the comparison of one or more completed projects with details of a new project to predict cost and duration . There appears to be considerable overlap in the literature between the categories of estimation by expert judgement and estimation by analogy. Boehm describes expert judgement as "consulting with one or more experts, who use their experience and understanding of the proposed project to arrive at an estimate of its cost" (Boehm 1981, p333). His definition of estimation by analogy is that it "involves reasoning by analogy with one or more completed projects to relate their actual costs to an estimate of the cost of a similar new project" (ibid, p336). Heemstra states that the foundation of the analogy estimation technique is "an analysed database of similar historical projects or similar project parts or modules" (Heemstra 1992, p630). Vigder and Kark classify estimation techniques as simply model based or analogy based, where analogy based modelling involves estimating costs by comparing the current project with previous projects (Vigder and Kark 1994). As with Heemstra, this requires maintenance of a history of past projects. However, according to Vigder and Kark this history of past projects can be maintained in the memory of an estimator – which they categorise as the informal analogy model. The more formal analogy model is an actual record of the data from past history. Vigder and Kark's informal analogy model could be described as expert judgement, where the expert is depending on his/her experience to predict costs. Hughes definition of expert judgement is "where an estimate is based on the experience of one or more people who are familiar with the development of software applications similar to that currently being sized" (Hughes 1996, p68). This definition in itself implies use of analogical reasoning. The overlap is also evident in work by Vincinanza (Vincinanza *et al.* 1991). For the purposes of this paper let us define estimation by analogy as the prediction of estimates by comparison with previously completed projects where information on those projects is available in some sort of documented format. Therefore Vigder and Kark's informal analogy model, i.e. use of an expert's memory, falls under the categorisation of expert judgement.

There is significant evidence in the literature that expert judgement is the most dominant method of estimation (Wrigley & Dexter 1987; Heemstra & Kusters 1991; Vigder & Kark 1994). However, there has not been much research to examine the use of expert judgement to estimate development effort or duration. While there is little evidence in research as to what forms the basis of an expert's judgement, it is believed to be subject to bias and political pressure (Hughes 1996). It has even been described as guessing (Kitchenham 1991). However, the research into expert judgement is promising for expert judgement as an estimating technique (Vicinanza *et al.* 1991; Atkinson & Shepperd 1994). Although the use of estimation by analogy formally through comparison with documented details from previous projects is rare, that which exists has shown that it performs better than algorithmic models (Cowderoy & Jenkins 1988; Shepperd *et al.* 1996; Shepperd & Scholfield 1996). The main issue with this technique is the lack of

available project data with which to compare the new project (Vigder & Kark 1994; Heemstra 1992). Estimation using similarities to previous projects is much more common on an informal basis, through the use of memory and recall.

The actual usage in industry of the different cost estimation techniques does not reflect the volume of literature and research on the subject. There is considerable literature available on the algorithmic models but these models are not widely used in industry (Heemstra 1992), perhaps as their accuracy is questionable and/or due to a lack of data with which to calibrate. The most dominant way of estimating costs is to depend somehow on previous experience, however subjective or biased this may appear. This use of previous experience may involve using your memory as an expert estimator (expert judgement) or by comparison with documented past projects (estimation by analogy). This reliance on previous experience leads to an expectation that this process can be automated or formalised in a CBR system.

3 Case-based Reasoning

Case-based reasoning (CBR) (Kolodner 1992; Kolodner 1993; Watson & Marir 1994; Barletta 1991; Slade 1991) is a relatively simple concept - it involves matching the current problem against ones that have already been encountered in the past and reworking the solutions of the past problems in the current context. It can be represented as a cyclical process that is divided into the four following sub processes as depicted in Figure 1 (Aamodt & Plaza 1994):

- retrieve the most similar case or cases from the case base
- reuse the case to solve the problem
- revise the proposed solution, if necessary
- retain the solution for future problem solving

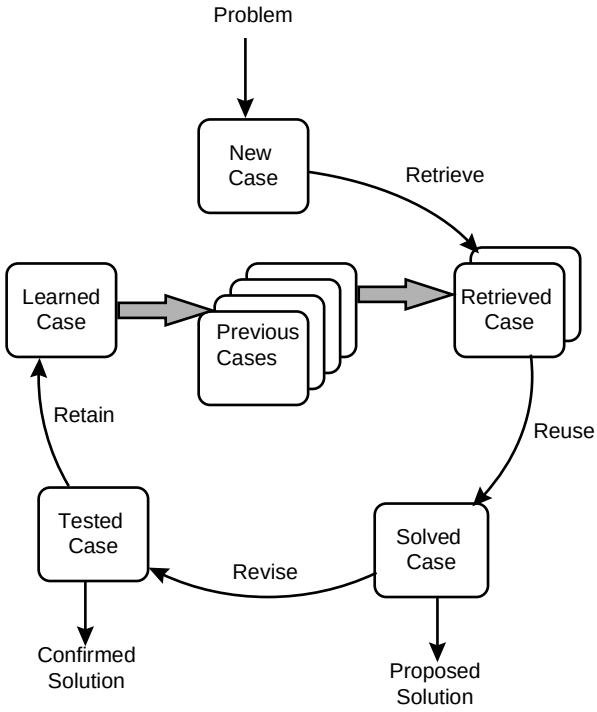

Figure 1 : The CBR cycle (adapted from Aamodt & Plaza 1994)

A new problem, described as a case, is compared to the existing cases in the case base and the most similar case or cases are *retrieved*. These cases are combined

and *reused* (i.e. adapted) to suggest a solution for the new problem. The solution proposed may need to be *revised* (i.e. evaluated and corrected) somewhat if it is not a valid solution. This verified solution is *retained* by adding it as a new case to the case base or as amendments to existing cases in the case base for use in future problem solving.

3.1 The application of CBR to Cost Estimation

CBR offers a number of advantages over the other cost estimation techniques. The developers of the various algorithmic models have attempted to derive models that quantify the causal dependencies within the domain. As these models do not effectively solve the problem, this suggests that the domain is difficult to model without clear rules or a clear understanding of all the different elements that contribute to cost estimation. The main advantage of CBR over the use of algorithmic models is that the use of CBR avoids the need to model the domain.

One of the difficulties with any of the existing techniques is the lack of project history data within organisations. This data is used to calibrate algorithmic models and for comparison purposes in estimation by analogy. A lack of data also means that the problem space is not uniformly covered. There may be a greater intensity of sample projects in certain areas than in others. There may be other areas of the problem space where there is no past project data available. For techniques such as algorithmic techniques and estimation by analogy this makes deriving general models from the data unworkable. CBR, however, will use the data that is relevant and available to make a prediction. Furthermore, a case base will also provide an effective means of storing data on historical projects. Another problem with historic data is that frequently it may be incomplete. However, the performance of a CBR application will degrade gracefully in situations of incomplete data.

CBR also has the advantage of possessing the capability to explain its reasoning. It is possible to view the cases which are retrieved as similar to the target case and to view the adaptation strategies that operate on the retrieved cases which result in the prediction. It also allows manual adaptation so an expert (such as an experienced project manager) can extrapolate from the similar retrieved cases and adjust the recommended solution if they feel it necessary.

Lastly, as CBR is a machine learning technique, a CBR system will augment its case base with new project scenarios over time. This is important as the software development process is a constantly changing process with new technologies, new methods, new techniques continually being introduced and adopted. A good cost estimation technique needs to be able to handle this natural evolution of software development (Bisio & Malabocchia 1995).

In recent years there has been some concrete research in the application of CBR to cost estimation which suggests that CBR can provide a practical aid to software development managers (Mukhopadhyay *et al.* 1992; Prietula *et al.* 1996; Bisio & Malabocchia 1995; Finnie *et al.* 1997a). However, this research focuses on problem that are similar in nature - similar types of applications from similar organisations. The data sets used primarily describe mainframe systems, developed in third generation languages and are restricted, within each CBR system, to the same types of applications. They do not take into account the variety of platforms and new technologies that contribute to software development in the 1990s. In addition, examination of the features or attributes that contribute to the case representation used by these systems imply that a detailed specification

of software was available at the time that the software cost estimation was undertaken. Thus the focus has been on estimation at a late stage in the development life cycle.

The accuracy of software estimates has a direct impact on the quality of an organisation's software investment decisions. Accurate estimation as early as possible in the development life cycle is important as it results in better priced software, realistic development schedules and efficient acquisition and allocation of resources.

Our research focuses on applying CBR to early cost estimation so that some of these benefits may be realised.

4 Proposed Case Representation

The case representation is the template for the cases in the case base. The features of the case representation are those aspects of the domain and the problem that are considered to be most significant in determining the solution and/or outcome. Clearly the choice of the features to be included in the case representation is critical to the success of the CBR process. The definition of a case representation is not the only factor that contributes to the success of the CBR application, there are a number of implementation choices and issues with the actual reasoning mechanism itself. However, the case is the foundation upon which CBR works and as such is a key component of any CBR system.

To identify a case representation for early estimation involves identifying a case representation for a software development project. The case representation must include those aspects or characteristics of a development project that have the most influence on the development effort of a software system. To apply to early estimation these characteristics must be known early in the development life cycle. The proposed case representation must be derived from the factors, or cost drivers, that are believed to influence the cost or effort of software development. More specifically, those cost drivers that are known early in the development stage of a project - at the project specification stage or system engineering stage.

4.1 Cost Drivers

Our analysis of the factors affecting cost identified three main sources of information. These included

- the algorithmic models;
- the information that estimators use and would like to have when they are estimating and
- risk factors which determine the risk involved in an implementation, and therefore also effect the effort involved in development.

A study was made of each of these sources of cost drivers. Many of the cost estimation models have adjustment factors built into them. These factors are determined from a number of variables that are believed to influence the cost of the software development. These variables, such as product complexity, analyst capability, computer turnaround time are usually judged on an ascending scale from low to high. For each value selected there is a numerical adjustment that is applied to the effort estimate. Identification of these cost drivers was made from analysis of reviews of cost drivers (Noth and Kretzschmar 1984; Wrigley & Dexter 1987) and empirical studies to identify which cost drivers are significant in

affecting productivity that have been published over the years. (Brooks 1981; Behrens 1983; Vosburgh 1984; Thadhani 1984; Boehm & Papaccio 1990; Kitchenham 1992)

The experts also use and require certain variables or inputs to develop a prediction of effort or cost. Identification of these inputs was made from a number of surveys and studies to identify those factors considered most important by estimation experts or project managers and other software personnel involved in estimating development work (Hughes 1996; Subramanian & Breslawski 1994; Subramanian & Breslawski 1995; Kendall & Lamb 1977; Lederer & Prasad 1992).

Evaluating the risk involved in a software development project requires assessing those factors that influence the occurrence of undesirable events. Undesirable events threaten successful software development and influence the outcome of a software development project. A successful software development project is one which is delivered on time, within budget and to the agreed user specification. Factors which can influence the success or failure of a project must therefore also have an influence on the effort required to perform the development work. The main sources of such risk factors were studies by Barki and Saarinen & Versalainen (Barki *et al.* 1993; Saarinen & Vepsalainen 1993).

A study by Noth & Kretzschmar found that more than 1200 cost drivers were mentioned in the cost estimation literature (Noth and Kretzschmar 1984). However, in our study over 200 potential cost drivers were considered. The features that we identified as predictive and having the potential to be known early in the development life cycle are presented in Table 1.

All current estimation techniques use some measure of the size of the system as an input to the estimation process. Consistently, with all algorithmic models a measure of system size, usually in lines of code (LOC) or function points, is one of the inputs to the prediction of effort. With estimation by analogy it is also important to choose at least one variable or feature to act as a size driver (Shepperd *et al.* 1996). Thus, a size driver appears important to the estimation process.

There are two main difficulties with size drivers. Firstly, any measure of system size identified early in the development life cycle will itself be an estimate. Secondly and more generally, the size drivers of LOC or function points are inappropriate development ‘targets’ and inappropriate inputs to the estimation process (Wrigley & Dexter 1991; Wrigley & Dexter 1987). Consider LOC firstly, LOC is the net result of the development effort. LOC themselves do not cause effort, effort results in LOC. With function points, counting function points gives the impression of measuring the system requirements. The target in system development is to satisfy the requirements. The requirements describe the required functionality of the system to be developed. However, an extra report or screen in the system can increase the function point count but may not contribute towards satisfying the required functionality.

Table 1: Case Features

Feature
Management Top management support/commitment Project manager's experience (e.g. number of projects) Project manager's success rating (some indication of performance on previous projects) Number of similar projects manager has managed Manager's familiarity with team Project priority
Project Team Team IT experience Team understanding/experience of application
Users User understanding of requirements Extent of user support/participation User IT competence and experience User/Management agreement
System/Application Critical business system Architecture Type (expanded to different types appropriate to an organisation, e.g. standalone, distributed data, distributed processing…) Operating mode (batch, online, real time) Need for new hardware Hardware concurrently developed Required integration with other systems Required reliability
Development Method/Process Development Process used (expanded to different types appropriate to an organisation e.g. classic life cycle, prototyping, object oriented…) Project novelty (of method and tools used) Programming type (structured, rule-based, functional) Need for new system software Use of tools Use of standards Use of design and code inspections

This poses a problem for early estimation. Without an accurate size input it is not reasonable to expect an accurate estimate of effort (e.g. in man months).

4.2 Proposed Productivity Measure

The effort involved in software development can be described simply as depending on the size of the software to be developed (e.g. LOC) and on the productivity of the developers (e.g. amount of time to develop one LOC) as detailed in the following equation.

$$\text{Effort} \propto \frac{\text{Size}}{\text{Productivity}}$$

As a measure of system size is unavailable and inappropriate for early estimation, it is more appropriate to use the available predictive features to come up with a measure which will indicate the effect that the case features will have on the

expected productivity of the project. Let us call this measure 'the productivity coefficient'. The productivity coefficient for a case in the case-base can be calculated as the ratio of the actual productivity of that case (available after the project is completed) to the average development productivity of the organisation calibrated across all the cases in the case-base.

$$\text{Productivity Coefficient} = \frac{\text{Actual Case Productivity}}{\text{Average Productivity}}$$

A productivity coefficient of value close to one indicates that the project is one which will produce a productivity level very close to the organisation's average productivity. This suggests that there may be little risk associated with this project. A productivity coefficient value less than one indicates that the expected productivity of this project is well below the average productivity for the organisation which may effect the actual cost of development. This suggests that there are characteristics about this project that suggest some level of risk is involved. A productivity coefficient value of greater than one indicates an expected higher level of productivity than average for the organisation suggesting confidence and familiarity with the work being undertaken.

The productivity coefficient will provide senior management with information to assist them in making decisions on the viability of a project or on the cost of a project. It will indicate whether the characteristics of the project indicate that the project is at risk of lower productivity than normal indicating a higher cost of development.

It is important to note that although this coefficient can give an indication of where risk may be involved in a project, it is not an actual measure of risk. It provides an indication of an organisation's potential productivity based on its past experiences with 'similar' development projects. The additional features required in the case representation are presented in Table 2.

Table 2: Additional Case Features

Features
See Table 1 for list of other case features
Productivity Coefficient
Actual Effort Actual Size

5 Evaluation & discussion

Within case-based reasoning the case representation is critical. When relevant features are not identified in advance the case representation is not complete. If these hidden features, not identified for the case representation, are important in certain situations then the CBR technique will not work in these circumstances. The case representation presented in Table and Table may not be complete for two reasons. Firstly, the case representation includes those features that are indicative of effort or productivity and are known at an early stage in the development life cycle. The features have been selected from a variety of cost drivers. Not all of the cost drivers identified are included in the case representation as only those known at a system engineering and planning stage can be included

for early estimation. For example, the level of top management support will be known but the level of changes to requirements cannot be known. As certain features cannot be known in advance and are therefore not included, the proposed case representation may not be complete.

In addition, the problem of cost estimation has not been solved. This suggests that there could be factors that influence cost that have not been identified or agreed upon in the existing literature. There may therefore be hidden features that are important to the prediction. If this is the position, CBR as a technique will not work well. A full evaluation of the problem using sample cases is necessary to see whether this case representation can predict the productivity coefficient accurately. If not, there may be hidden features that are meaningful to the prediction and outcome.

The proposed CBR solution has two main limitations to its application. A number of the features included in the case representation are susceptible to subjective quantification, for example *project manager experience*, *IT team experience*, *project novelty* or *user understanding of requirements*. One organisation or individual may measure experience in number of years, while another may measure experience in number of projects. The average productivity across organisations can also be very different. Due to these facts this CBR solution will only be applicable locally within organisations. Use across organisations may result in inconsistent definitions of variables or quantification and subjectivity in the rating of the features.

In addition, the proposed CBR solution is most applicable to large organisations for two main reasons. Firstly and more importantly, larger organisations are more likely to have enough 'similar' projects to provide an adequate case-base. The case representation includes many human factors which have been identified as important in estimating software development effort. Larger organisations may have formalised procedures for personnel and project evaluation. Outputs from these procedures could provide quantified inputs to the case representation.

Constant features are features that have the same value over all cases in the case base. These features do not have an effect on the CBR process and are not needed in the derivation of the solution. There are a number of features within the proposed case representation that may be constant, particularly since this approach is most likely to be successful within a single large organisation. A standard design methodology or company approach to development, common within large software development units, would mean that features such as use of tools, use of standards or use of design and code inspections may be constant. Furthermore, within a single application of the CBR technique in a large organisation there is a possibility that any hidden features may also be constant. Constant features are unnecessary features. This suggests that there may be a possibility that even if the case representation were incomplete, the hidden features may not have an effect for cases taken from the same context. A full evaluation of the case representation using sample cases is required to evaluate this possibility.

5.1 Sample Scenario

To demonstrate how the case representation would work consider the following project example. The project data is actual data from a project that was completed in 1994 by an Irish software development company for one of their clients. The details of the software development and client organisations are confidential. To

facilitate ease of reading let the software development company be called Company A. The details of the project in question were discussed with the project manager from Company A responsible for the delivery of the project.

The project involved the detailed analysis, design and development of an asset register. The client supplied a high level user requirements document and provided input to the project by answering queries, reviewing the design and documentation and acceptance testing the software. The client did not provide any personnel to work on the actual design or development.

The application had a client/server architecture, with PC clients and a DEC Alpha server. It was developed using Visual Basic and C on the client side and with a VAX Rdb relational database on the server side.

The project was estimated at the time of accepting the contract as 247 mandays of work. The original estimation was performed by software development project managers within Company A. The method employed was expert judgement. Estimation was based on the managers' prior experience developing similar types of projects. The actual effort involved in project was 406 mandays which was considerable higher (64%) than the original estimate.

To demonstrate how such a project would contribute to assisting with future estimations for Company A, a case needs to be derived for this project. To derive a case for this project values have to be assigned to each of the features in the case representation. Before assigning the values, a metric must be assigned to each feature. For certain features this is a simple quantitative measure, e.g. *the number of similar projects managed*. For some features this is a qualitative measure, e.g. *top management support/commitment*. For other features a measure is not needed, a value is simply chosen from a list of possible values, e.g. *operating mode*, or is simply a Yes/No answer, e.g. *hardware concurrently developed*. Table 3 lists proposed measures for the features in the case representation.

For any organisation using this case representation, specific definitions of qualitative measures, such as Low, Medium and High need to be defined and documented to minimise subjectivity. However, for the purposes of this sample case, explicit definitions are not required.

Table 4 describes the project, as a case, with values for each of the features.

In order for this case to be added to the case base and used for future new projects the outcome must be included to allow a productivity coefficient to be derived for the case. The features required for the outcome, as specified in the case representation, are the actual effort expended on the development and the actual size of the application. The actual effort was 406 mandays. Company A does not generally measure the size of their projects. However in this situation, a function point analysis was performed on the project and the size of the project was calculated to be approximately 250 function points.

To derive the productivity coefficient for this sample case the average productivity for Company A is required. A case base, itself, will provide an average productivity over all the projects in the case base. For the purposes of this sample project a case base is not available. As Company A does not measure projects habitually, there is no data that can be used to identify their average productivity. To facilitate the aims of this evaluation the Industry Average productivity figures, published by Symons will be used (Symons 1991). For a project of approximate size 200 function points the industry average productivity figure is 0.11 function

points per work hour or 0.825 function points per manday (Company A work on a 7.5 hour day).

Table 3: Proposed Feature Measures

Management	Measure
Top management support/commitment	Low, Medium, High
Project manager's experience	Number of Years
Project manager's success rating	Low, Medium, High
Number of similar projects manager has managed	Number
Manager's familiarity with team	Low, Medium, High
Project priority	Low, Medium, High
Project Team	
Team IT experience	Low, Medium, High
Team understanding/experience of application	Low, Medium, High
Users	
User understanding of requirements	Low, Medium, High
Extent of user support/participation	Low, Medium, High
User IT competence and experience	Low, Medium, High
User/Management agreement	Low, Medium, High
System/Application	
Critical business system	Low, Medium, High
Architecture Type	Distinct Value
Operating mode (batch, online, real time)	Distinct Value
Need for new hardware	No/Low, Medium, High
Hardware concurrently developed	Yes/No
Required integration with other systems	Low, Medium, High
Required reliability	Low, Medium, High
Development Method/Process	
Development Process used	Distinct Value
Project novelty (of method and tools used)	Low, Medium, High
Programming type (structured, rule-based, functional)	Distinct Value
Need for new system software	Low, Medium, High
Use of tools	Low, Medium, High
Use of standards	Low, Medium, High
Use of design and code inspections	Low, Medium, High
Productivity Coefficient	
Actual Effort Actual Size	

The productivity coefficient for the project is calculated as the actual productivity of the project divided by the average productivity. The actual productivity of the project (calculated as 250 function points/406 mandays) is 0.616 function points per manday. Therefore the productivity coefficient for the sample project would work out as 0.746.

A productivity coefficient value of less than one indicates that the expected productivity of the project would be well below the average productivity for the organisation which will effect the actual cost of development. Considering the sample project, the actual effort involved in this project was significantly above what had originally been estimated, which may indicate that there were characteristics of the project which were overlooked in the original estimation and resulted in an estimate that was not representative of the effort involved.

Table 4: The Sample Case

Management	Measure
Top management support/commitment	High
Project manager's experience	3
Project manager's success rating	Medium
Number of similar projects manager has managed	0
Manager's familiarity with team	Low
Project priority	Medium
Project Team	
Team IT experience	Low
Team understanding/experience of application	Low
Users	
User understanding of requirements	Medium
Extent of user support/participation	Low
User IT competence and experience	Medium
User/Management agreement	High
System/Application	
Critical business system	Medium
Architecture Type	Client/Server
Operating mode (batch, online, real time)	Online/Batch
Need for new hardware	No
Hardware concurrently developed	N/A
Required integration with other systems	Medium
Required reliability	Medium
Development Method/Process	
Development Process used	Company A Methodology
Project novelty (of method and tools used)	High
Programming type (structured, rule-based, functional)	Structured
Need for new system software	High
Use of tools	Low
Use of standards	High
Use of design and code inspections	Medium
Productivity Coefficient	
Actual Effort	
Actual Size	

It is worth noting that the fact that Industry Averages were used in the calculation instead of actual company productivity averages may contribute somewhat to the productivity coefficient as Company A's productivity rate may not be exactly the same as the Industry Average. However, use of Industry Averages allows the illustration of this example.

With hindsight, Company A can identify some difficulties which were encountered during the project. The project was using new company procedures that were not well established and well known by the staff on the project. The technology used involved new system software which took a significant amount of time to understand and get working adequately. In addition the project team were relatively inexperienced and had never worked as a team previously. These characteristics of the project are identifiable within the features of the case representing the project. Features such as *project novelty*, *need for new system software*, *manager's experience with the team* and *team IT experience* contribute to the case and in this situation may have contributed to the significant overrun in the project.

If the case documented in Table 4 were retrieved as the most likely match for a new project to be undertaken by Company A, the productivity coefficient of 0.74 would be meaningful. It should indicate to Company A that based on past experiences a project similar to the new one they are considering took significantly more effort to develop than their average expectation. This should result in an increase in Company A's estimate of development cost for this new project. Over time and based on experience, this increase in the effort/cost could be related to the value of the productivity coefficient.

6 Conclusions and Future Work

In this paper we identify the difficult problem of project development cost estimation where human competence is evidently experience-based. More specifically, we consider estimation at an early stage in the development life cycle to assist with strategic decision making. If this process is to be automated in a CBR system a case representation capturing the predictive features in the process must be identified; such a case representation is presented here. Certain difficulties presented themselves. With the inability to include features predictive of size the case representation is not complete. Our proposal is that the outcome of the case representation is a measure, called the productivity coefficient, of the effect the case features will have on the expected productivity of the project. It will show where the characteristics of a project, based on previous project experiences, reveal that a project is at risk of lower productivity than normal indicating a higher cost of development. This approach is most likely to be successful within a single large organisation to minimise subjectivity and who can provide an adequate case-base.

The obvious way to extend this research is to test the case representation by evaluating it using a sample case base. This requires data about software development projects to use as sample cases. There are difficulties with getting project data. Firstly there is a lack of project data available (Vigder & Kark 1992; Heemstra 1992; Subramanian & Breslawski 1994). Secondly there is a reluctance among companies to divulge project related data due to competitive reasons even when confidentiality is promised (Cusumano & Kemerer 1990). Collaboration with a software development company would be ideal.

A further extension to this research is to apply the reasoning mechanism to the proposed case representation. This would involve further research and analysis of a number of areas, for example, the most appropriate way to index the cases, identifying the most appropriate retrieval mechanism to employ and identifying adaptation strategies that could be applied.

This research focused on providing cost estimation at an early stage in the development life cycle. However, this is one direction for extending the existing research into applying CBR to cost estimation. The other direction is to evaluate CBR as an application to cost estimation across broader domains, i.e. across increased application variability. In order to use CBR in estimation in a broader context there is a need to identify abstract characteristics which are invariant in the transformation to completely different applications. There is also a need to develop adaptation techniques to transform cases between these different application contexts (Delany *et al.* 1998).

One conclusion of this research is that without a size estimate the case representation is not complete and it is not possible to estimate the development

effort accurately. However, Mukhopadhyay & Kekre concluded that in a specific application domain (process control applications) it was possible to estimate the software size from user specified application features (Mukhopadhyay & Kekre 1992). These application features were the functions demanded by the end user and not the result of system design so they were available very early in the development lifecycle. The limitation here is the application is restricted to a very specific application domain but nevertheless, it would be interesting to research the effect of including user specified application features in the case representation.

References

AAMODT A. & E. PLAZA 1994 "Case-based reasoning: foundational issues, methodological variations and system approaches", Artificial Intelligence Communications 7 (1) p.3059.

ALBRECHT A. & GAFFNEY J. 1983 "Software function, source line of code and development effort prediction: a software science validation" IEEE Transactions on Software Engineering 9 (6) p.639-648

ALBRECHT A. 1979 "Measuring application development productivity" Proceedings IBM Applications Development Symposium, GUIDE Int. and SHARE Inc., IBM Corp., Monterey, CA. Oct 14-17.

ATKINSON K. & M. SHEPPERD 1994 "Using function points to find analogies", European Software Cost Modelling Conference, Ivrea Italy, 11-14 May.

BARKI H., S. RIVARD & J. TALBOT 1993 "Towards an assessment of software development risk" Journal of Management Information Systems 10 (2) p.203-225

BARLETTA R. 1991 "An introduction to case-based reasoning" AI Expert, 6 (8) p.42-49.

BEHRENS C. A. 1983 "Measuring the productivity of computer systems development activities with function points" IEEE Transactions on Software Engineering, 9 (6) p.648-652.

BISIO R. & F. MALABOCCHIA 1995 Cost estimation of software projects through case-base reasoning" Case-Based Reasoning Research and Development. First International Conference, ICCBR-95 Proceedings p.11-22.

BOEHM B 1981 Software Engineering Economics, Prentice Hall

BOEHM B 1984 "Software Engineering Economics" IEEE Transactions on Software Engineering 10 (1) p.4-21

BOEHM B. & P. PAPACCIO 1988 "Understanding and controlling software costs" IEEE Transactions on Software Engineering 14 (10) p.1462-1477

BREDERO R., CARACOGLIA G., JAGGERS C., KOK P., TATE G. & VERNER J. 1989 "Comparative evaluation of existing cost estimation tools" Mermaid report D7.1Y

BROOKS W. D. 1981 "Software technology payoff – some statistical evidence" Journal of Systems and Software 2 p.3-9.

CHATZOGLOU P. & L. MACAULEY 1996 “A review of existing models for project planning and estimation and the need for a new approach” International Journal of Project Management 14(3) p.173-18

COWDEROY A. J. & J. O. JENKINS 1988 “Cost estimation by analogy as a good management practice”, Proceedings of Software Engineering 88, ed. Pyle Liverpool, IEEE/BCS p.80-84

CUELENAERE A., M. VAN GENUCHTEN & F. HEEMSTRA 1987 “Calibrating a software cost estimation model: why and how” Information and Software Technology 29 (10) p.558-567.

CUSUMANO M.A. & C. F. KEMERER 1990 “A qualitative analysis of U.S. and Japanese practice and performance in software development” Management Science 36 (11) p.1384-1404.

DE MARCO T. 1982 Controlling software projects: management, measurement and estimation. Yourdon Press, NY.

DELANY S.J., P. CUNNINGHAM & W. WILKE 1998 “The limits of CBR in project estimation” in Proceedings of GWCBR’98 6th German Workshop on Case-Based Reasoning, eds L. Gierl & M. Lenz, Berlin, p.99-108.

FINNIE G.R., WITTIG G.E. & DESHARNAIS J-M 1997a “Estimating software development effort with case-based reasoning”, Proceedings of International Conference on Case-Based Reasoning, D. Leake, E. Plaza, (Eds) p.13-22

FINNIE G.R., WITTIG G.E. & DESHARNAIS J-M 1997b “A comparison of software effort estimation techniques: Using function points with neural networks, case based reasoning and regression models” Journal of Systems and Software

HEEMSTA F. 1992 “Software cost estimation” Information and Software Technology 34 (10) p.627-639

HEEMSTRA F. & R.J. KUSTERS 1991 “Function point analysis: evaluation of a software cost estimation model” European Journal of Information Systems 1 (4) p.229-237.

HERD J. R., J. N. POSTAK, W. E. RUSSELL & K. R. STEWART 1977 “Software cost estimation – study results” Final technical report, RA-DC-TR-77-220, Vol 1, DOTY Associates, Inc., Rockville, MD.

HUGHES R.T. 1996 “Expert judgement as an estimating method”, Information and Software Technology 38 (2) p.67-75.

JACK R. & M. MANNION 1995 “Improving the software cost estimation process” Software Quality Management III Vol. 1 p.245-256

JENSEN R. W. 1983 “An improved macro level software development resource estimation model” Proceedings 5th ISPA Conference, St Louis MO.

JENSEN R.W. 1984 “A comparison of the Jensen and COCOMO schedule and cost estimation models” Proceeding International Society of Parametric Analysis p.96-106

JONES C. 1986 C Programming Productivity, McGraw-Hill, New York, NY

KEMERER C. 1987 “An empirical validation of software cost models” CACM 30 (5) p.416-429.

KEMERER C. F. 1989 “An agenda for research in the managerial evaluation of computer-aided software engineering (CASE) tool impacts” Proceedings of the 22nd annual Hawaii International Conference on Systems Sciences, p.219-228.

KENDALL R.C. & E.C. LAMB 1977 “Management perspectives on programs, programming and productivity” presented at Guide 45, Atlanta, GA, Nov 1977 p.201-211.

KITCHENHAM 1991 “Making process predictions” in Fenton N. Software Metrics – A Rigorous Approach, Chapman and Hall.

KITCHENHAM B. & N. TAYLOR 1985 “Software project development cost estimation” The Journal of Systems and Software, 5 p.267-278

KITCHENHAM B. 1992 “Empirical studies of assumptions that underlie software cost-estimation models” Information and Software Technology 34 (4) p.211-218

KOLODNER J. 1992 “An introduction to case-based reasoning”, Artificial Intelligence Review 6(1) p.3-34.

KOLODNER J. 1993 Case-based reasoning, Morgan Kaufmann, California

KUSTERS R., M. VAN GENUCHTEM & F. HEEMSTRA 1990 “Are software cost-estimation models accurate?” Information and Software Technology 32 (3) p.187-190.

LEDERER A. & J PRASAD 1992 “Nine management guidelines for better cost estimating” CACM 35 (2) p.51-59

MOHANTY S. 1981 “Software cost estimation, present and future”, Software Practical Experience, 11 p.103-121.

MADACHY R.J. 1995 “Knowledge-Based Risk Assessment and Cost Estimation”, Automated Software Engineering, 2 (3) p.219-230.

MADACHY R.J. 1997 “Heuristic Risk assessment Using Cost Factors”, IEEE Software, 14 (3) p.51-59.

MUKHOPADHYAY T. & S. KEKRE 1992 “Software effort models for early estimation of process control applications” IEEE Transactions on Software Engineering 18 (10) p.915-923.

MUKHOPADHYAY T., S. VICINANZA & M. PRIETULA 1992 “Examining the feasability of a case-based reasoning model for software effort estimation” MIS Quarterly 16 (2) p.155-171

NOTH T. & M KRETZSCHMAR 1984 Estimation of software development projects, Springer Verlag (in German)

PRIETULA M., S. VICINANZA & T. MUKHOPADHYAY 1996 “Software effort estimation with a case-based reasoner” Journal of Experimental and Theoretical Artificial Intelligence 8(3-4) p.341-63

PUTNAM L. 1978 “A general empirical solution to the macro software sizing and estimation problem” IEEE Transaction of Software Engineering 4 (4) p.345-361.

RUBIN H. 1985 “A comparison of cost estimation tools” Proceedings 8th International Conference on Software Engineering, IEEE New York, p.174-180

SAARINEN T. & A. VEPSALAINEN 1993 “Managing the risks of information systems implementation” European Journal of Information Systems 2 (4) p.283-295.

SHEPPERD M. & C. SCHOFIELD 1996 “Effort estimation by analogy: a case study” Presented at the European Control and Metrics Conference, Wilmslow, UK, May 1996. [Online] Available: http://xanadu.bournemouth.ac.uk/ComputingResearch/ChrisSchofield/angel/papers/ESCOM96.html

SHEPPERD M., C. SCHOFIELD & B. KITCHENHAM 1996 “Effort estimation using analogy” Proceedings of the 18th International Conference on Software Engineering p.170-178

SLADE S 1991 “Case-based reasoning: a research paradigm”, Artificial Intelligence Magazine, 12(1) p.42-55.

SUBRAMANIAN G. & S. BRESLAWSKI 1994 “The importance of cost drivers used in cost estimation models: Perceptions of project managers” Proceedings of 1994 Information Resources Management Association International Conference, San Antonio TX, USA p503-507

SUBRAMANIAN G. & S. BRESLAWSKI 1995 “An empirical analysis of software effort estimate alterations” Journal of Systems and Software 31 (2) p.135-141

SYMONS C. 1991 Software sizing and estimating – MkII FPA (Function Point Analysis), Wiley, UK

SYMONS C.R. 1988 “Function point analysis: difficulties and improvements” IEEE Transactions on Software Engineering 14 (1) p.2-11.

TAUSWORTHE R. 1981 “Deep space network software cost estimation model” Publication81-7, Jet Propulsion Laboratory, Pasadena CA.

THADHANI A.J. 1984 “Factors affecting programmer productivity during application development” IBM Systems Journal 23 (1) p.19-35.

THEBAUT S. M. & V. Y. SHEN 1984 “An analytic resource model for large-scale software development” Inf. Proc. Management 20 (1-2)

VICINANZA S., T. MUKHOPADHYAY & M. PRIETULA 1991 “Software effort estimation: an exploratory study of expert performance” Information Systems Research 2 (4) p.243-262

VIGDER M.R & A.W. KARK 1994 “Software cost estimation and control” National Research Council of Canada, Institute for Information Technology, NRC No. 37116. Available online http://wwwsel.iit.nrc.co/abstracts/NRC37116.abs

VOSBURGH J. 1984 “Productivity factors and programming environments” Proceedings of the 7th International Conference on Software Engineering, p.143-152.

WATSON I. & F. MARIR 1994 “Case-based reasoning: a review”, The Knowledge Engineering Review 9 (4) p.327-354.

WOLVERTON R. 1974 “The cost of developing large-scale software” IEEE Transactions on Computers 23 (6) p.615-636

WRIGLEY C.D. & A.S. DEXTER 1987 “Software development estimation models: A review and critique” Proceedings of the ASAC Conference, University of Toronto, Ontario, p.125-138

WRIGLEY C.D. & A.S. DEXTER 1991 “A model for measuring information system size” MIS Quarterly 15 (2) p.245-257.