Estimating the development cost of custom software

Ioannis Stamelos\textsuperscript{a,}\textsuperscript{*}, Lefteris Angelis\textsuperscript{a}, Maurizio Morisio\textsuperscript{b}, Evaggelos Sakellaris\textsuperscript{c}, George L. Bleris\textsuperscript{a}

\textsuperscript{a}Department of Informatics, Aristotle University, Thessaloniki 54124, Greece
\textsuperscript{b}Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy
\textsuperscript{c}Singular International, 10A Dodekanisou Street, Thessaloniki 54626, Greece

Accepted 3 October 2002

Abstract

In this paper an approach for the estimation of software development costs is presented. The method is based on the characterization of the software to be developed in terms of project and environment attributes and comparison with some similar completed project(s) recovered from a historical database. A case study is also presented, focusing on the calibration and application of the method on 59 information systems implementing supply chain functions in industry. Various strategies are explored, the best of which predicted effort quite effectively, with a mean estimation error of 24\% with respect to the actual effort.

© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Software development effort; Cost estimation; Information system; Supply chain

1. Introduction

The use of computers in enterprises has provided powerful production and management tools and has resulted in enhanced functionality and productivity. There is a continuously growing need for special information technology (IT) solutions for enterprises regarding generic core processes, such as order processing, production planning and scheduling, manufacturing, logistics [8,21,26], or more specific tasks, such as supplier relationship management [12]. The utilization of IT for enterprise systems requires continuously increasing investments in machinery and software applications. Very often such investments are realized through the specification and implementation of custom information systems, i.e. information systems that are developed to meet the needs of a specific enterprise. These systems can be distinguished in “data intensive” systems, devoted to the support and management of the enterprise business processes and “control intensive” systems aiming to the control of production lines. In this paper, only the former type of systems will be considered.

Over the past 10 years there is a strong tendency to acquire off-the-shelf commercial information systems, known as Enterprise Resource Planning (ERP) systems. The implementation of such systems involves mainly the adaptation of the system to the business needs of the enterprise through system parameterization and, vice
versa, the re-engineering of business processes to match system specifications [21]. According to a recent study by Price Waterhouse [10], it was estimated that in the year 2000 two-thirds of all the business software would have been bought off-the-shelf. Consequently, ERP systems seem to concern two out of three information system implementations. One out of three cases is still a “traditional” one, with new software development or enhancement of existing software being the focus of the project. Moreover, there will be always need for custom applications satisfying specific needs of an enterprise. The above considerations imply that custom software development will continue to be a core process for building IT infrastructure.

An information system is developed according to a system life cycle, composed of development phases such as requirement specification and analysis, system design, coding and testing. Various parallel activities, such as project management, planning and software quality assurance, are equally important. Project planning requires the determination of the necessary resources to complete the project and the development of a schedule to be followed. Therefore, the generation of estimates of effort and time is required, a task known as Software Cost Estimation (SCE). The most important cost component in information systems development is the personnel costs related to the development effort of custom software components [25]. Consequently, SCE is focusing on the prediction of the human effort to specify, analyze, design, code and test software. Other activities such as project management itself, project support and software quality assurance are also taken into account. Effort estimations are helpful both for the IT client and the IT developer. In particular, based on these estimations, the acquiring organization may assess and monitor implementation costs, evaluate bids and develop realistic budgets and schedules. In the rest of the paper we will always use the term “cost” to denote human effort for software development.

Unfortunately, it has been observed that this task is one of the hardest in IT development: the majority of the projects involving software development suffer from budget and schedule overruns, caused, among other reasons, by insufficient initial estimations [6,25]. The same problem is being repeated in ERP implementation projects: although ERP successes are largely publicized, it appears that too many ERP projects are also failing [10,11,30]. It has been estimated that approximately 90% of ERP implementations are late or over budget [23]. The reasons behind this major problem seem to be poor cost and schedule estimations or changes in project scope rather than project management failure [11].

In the past two decades, researchers have proposed various approaches and techniques to resolve the problems in SCE. A number of criteria have been proposed to assess the efficiency of an SCE method when applied on a project dataset. The most widely accepted criterion is the mean magnitude relative error (MMRE), defined as a percentage of the actual effort spent on the project. An acceptable target value for MMRE is 25% [7], meaning that in average a project may be estimated with a relative error of 25% with respect to its actual effort. Although this value may seem relatively high to managers that are not familiar with SCE methods, it is justified by the fact that software is a complicated human artifact that is also difficult to measure. The result is that SCE is based on noisy data and fuzzy project information involving always some degree of uncertainty. Experience has shown that it is always hard for any generic SCE technique to produce accuracy figures that are better than the target value of 25% when applied to some project dataset. Moreover, various studies have shown that MMRE values higher than 100% are quite common [16,17,18]. Estimation of ERP implementation costs is also in its infancy. A recent study in this field [24] produced estimates that led to a MMRE of over 90%. Fairly good accuracy is obtained in specific code development environments (see [13] for an example).

From the above discussion, it is evident that in order to manage an information system development project it is important to be able to predict the effort needed to produce its software components. Hence, accurate, general purpose, estimation techniques and tools are needed to support managers in this task. In this paper, an estimation approach is presented and the promising results of a case study are reported. Our proposal is based on an already established approach, namely estimation by analogy, but focuses on the calibration of the method before its actual application. The case study was performed on a number of software projects related to supply chain activities for manufacturing companies. We chose this set of projects for their
strategic importance in the field of enterprise information systems. The case study investigates also the effect that the combination of automated and human provided estimates has on prediction accuracy. Section 2 discusses SCE and presents the approach that was followed, namely the calibrated estimation by analogy method, based on function point analysis for software sizing. Section 3 describes the project dataset on which the case study was performed. Section 4 presents the estimation accuracy results and discusses the findings. Finally, a concluding section summarizes the paper and discusses the future research issues.

2. Software cost estimation

As mentioned above, a crucial activity in the initial project phases is the estimation of the necessary project development effort and time. The three main SCE methods are expert judgment, algorithmic cost estimation and estimation by analogy [5]. Expert judgment, the most commonly used method, is based on the accumulated experience of a team of experts. Algorithmic cost estimation involves the application of a cost model, i.e. one or more mathematical formulas which, typically, have been derived through statistical data analysis. Finally, estimation by analogy compares the software project under consideration with a few similar historical projects (i.e. projects with known characteristics, effort and schedule).

Managers of information systems rely mainly on expert judgment because estimates are produced easily and without the need for complicated tools or techniques. This approach results in the generation of estimates in an ad hoc, non-repeatable and non-controllable manner. The amount of badly estimated projects that has been observed reveals the problems of expert judgment.

As a consequence, the need for automated support and modeling techniques has arisen. A typical algorithmic cost prediction model may be of the form

\[ \text{EFFORT} = a \times (\text{SIZE})^b, \]

where EFFORT is typically measured in man months and SIZE is the size of the software measured with some suitable metric (see Section 2.3). The most common modeling technique applied within SCE is ordinary least square regression (OLS). OLS fits a specified model to the historical data trying to minimize the overall sum of squared errors.

On the other hand, estimation by analogy is a technique that has been proposed for a long time as a valid alternative to expert judgment and algorithmic cost estimation. However, only recently [27], it has been presented in the form of a detailed estimation methodology and has been applied uniformly on a number of cost datasets. This technique is the core of our approach and will be further detailed in the following subsection.

There are two other important issues in SCE: the calibration of a cost method and the management of the estimation uncertainty. Calibration is important when algorithmic cost models or estimation by analogy is applied and refers to the improvement of the accuracy of the method by taking into account the peculiarities of the project dataset on which it is applied (i.e. projects of one software organization or projects implementing applications for a specific industry or business process). For example, in the case of model (1), calibration would require the re-calculation of the coefficients \(a\) and \(b\), in order to increase the precision of the estimation.

On the other hand, as already mentioned above, estimates involve uncertainty for various reasons [20]. For example, there is an inherent inaccuracy in the estimation process because an estimate is a probabilistic assessment of a future condition, and therefore involves uncertainty. Other reasons are the unclear requirements and project implications in the early phases. Taking into account the above discussion, it is both safer and realistic to produce interval estimates, i.e. a lower and an upper bound along with some probability that the real effort belongs to that interval. Relying blindly on a point (i.e. single value) estimate, given the error magnitude associated to the cost estimation techniques, may lead easily in wrong managerial decisions and project failure. An interval estimate may provide a point estimate for practical purposes (the mean of the interval), but still gives invaluable information about the reliability of the estimation process and provides the basis for risk and what-if project analysis.

The generation of confidence intervals in OLS is a typical procedure. In [3], a simulation approach for generating confidence intervals when estimating by analogy was proposed. This is accomplished by the
generation of an empirical distribution of the historical projects, obtained through the bootstrap statistical technique [9]. Bootstrap generates a large number of samples from the original project dataset. Applying analogy, an estimate is obtained from each sample and is used to produce an empirical estimate distribution, which in turn provides an estimate confidence interval.

This paper is based on the application of this method with the assistance of a tool, namely bootstrap based analogy cost estimation (BRACE), which helps in calibrating the method on a target project dataset. In order to render the paper self-contained, we will briefly describe in the following subsections, the method, the tool and the sizing method used in the case study.

2.1. Software estimation by analogy

The main aspect of the method is the utilization of historical information from completed projects with known effort. Estimation by analogy is essentially a case-based reasoning approach [27]. Initially, it is necessary to characterize the new active project, with attributes identical to the ones of the completed projects registered in the database. Examples of project attributes are source code length, programming language and personnel experience. Attributes are distinguished in quantitative (such as source code length, measured in number of programming language instructions) or qualitative (such as personnel experience, measured on an ordinal scale with values "low", "medium" and "high"). This situation is depicted in Table 1.

A basic notion is the distance between two projects. Distance measures the degree of dissimilarity between the two projects in terms of their attributes. A distance metric is used, based on the values of the k variables for these projects. The most known such distance metric is the Euclidean or straight-line distance which has a straightforward geometrical meaning as the distance of two points in the k-dimensional Euclidean space:

\[ d_{new,i} = \left( \sum_{j=1}^{k} (Y_j - X_{ij})^2 \right)^{1/2}, \quad i = 1, 2, \ldots, n. \]  

Other known distance metrics are the Minkowski distance, the Manhattan distance, the Canberra distance, the Kaufman–Rousseeuw distance, the Czekanowski coefficient and the Chebychev or “Maximum” distance (see [3] for full definition). The Minkowski distance is provided as another example:

\[ d_{new,i} = \left( \sum_{j=1}^{k} |Y_j - X_{ij}|^\lambda \right)^{1/\lambda}, \quad \lambda \text{ is an integer, } i = 1, 2, \ldots, n. \]  

When qualitative variables are encountered, the term \( Y_j - X_{ij} \) in (2) and (3) may be calculated according to the formula:

\[ Y_i - X_{ij} = \begin{cases} 1 & \text{if } X_{ij} \neq Y_j \\ 0 & \text{if } X_{ij} = Y_j \end{cases} \]  

when \( j \)th variable is binary or nominal.

The estimation of the effort using analogies is based on the completed projects that are similar to the new one. The user of the method has to calculate the distances of the new project from all database projects and identify few “neighbor” projects, i.e. with a relatively small distance value. Missing values (empty cells in Table 1) require further adjustment by the method of analogies. The estimation of the effort is eventually obtained by some combination of the efforts of the neighbor projects. Typically, the statistic used is the mean or the median of these effort values. Since the relationship between effort and size is well established, the statistic may also be adjusted using the neighbor project size [31]. For example, in the case of one neighbor, linear size adjustment is performed by the following formula:

\[ \text{EFFORT}_{\text{NEW}} = \frac{\text{SIZE}_{\text{NEW}}}{\text{SIZE}_{\text{NEIGHBOR}}} \times \text{EFFORT}_{\text{NEIGHBOR}}. \]
The calibration of the analogy-based method requires the detection of the best configuration of the available method options. The parameters that may be adjusted are: (a) the distance metric by which the projects of the database will be sorted according to their similarity to the one under estimation (e.g. Euclidean distance, Manhattan distance); (b) the number of closest projects (analogies); (c) the set of attributes for judging analogy; and (d) the statistic that will be computed from the efforts of the closest projects and will serve as estimation for the unknown effort. In addition, the statistic may be size-adjusted, as described above.

In [27], a sensitivity analysis was performed, studying the dynamic behavior of estimation by analogy by simulating the growth of a dataset over time. It was found out that MMRE decreases as the size of the dataset grows, and that it tends to stabilize beyond some number of projects.

2.2. BRACE

The authors have designed and implemented a software tool, called bootstrap-based analogy cost estimation, that supports the practical application of the analogy-based method using the bootstrap approach mentioned above [28]. Bootstrap is used both for method calibration and calculation of confidence intervals. The tool provides a flexible interface that allows the user to experiment with the different calibration options. The main functions of BRACE are:

1. typical input/output functions and file management facilities;
2. definition of attributes and project characterization;
3. project/attribute management (e.g. exclusion of projects/attributes from calculations);
4. choice of options to be considered for method calibration (with and without bootstrap);
5. determination of the best attribute set (the one providing the better accuracy results according to some criterion);
6. generation of estimations for a single project (with and without bootstrap).

In the calibration phase, the tool works on a dataset of historical projects. The tool estimates each project’s effort in turn (a procedure called jack knifing) using the other projects to find analogies. Normally, accuracy is determined in terms of the MMRE criterion, calculated as the mean of the magnitude relative errors (MRE) of all projects, according to the formula:

$$\text{MRE} = \frac{\text{EFFORT}_{\text{ACTUAL}} - \text{EFFORT}_{\text{ESTIMATED}}}{\text{EFFORT}_{\text{ACTUAL}}}. \tag{6}$$

As mentioned in Section 1, an acceptable value for MMRE is 25%. Another accuracy measure that is also considered is PRED(25) [7]. PRED(25) computes the number of projects that have been estimated with an MRE of less than 25%. An SCE method achieving a PRED(25) value of 75% is considered good. After the calibration phase, the tool may estimate new projects using the method configuration that was found to be the best. Fig. 1 provides a screenshot of BRACE.

2.3. Software sizing

One of the major issues in SCE is the estimation of the size of a software component. The reason for that is the inherent dependence of cost and quality on the size. Large software systems cost more and present more quality problems than smaller systems. In the early years of SCE, source code length metrics, such as the number of lines of code or program instructions, used to be quite popular for representing the software size. However, these metrics present many problems when used for SCE purposes. The two most important concerns are that these metrics are not easily predicted a priori and their definition is very sensitive to the programming style used by the software vendor. Moreover, they are not easily validated because source code is not always delivered to the IT client.

A more reliable technique, namely function point analysis [2, 14], measures the amount of functionality that will be embedded in the system under construction. The technique provides rules for counting the occurrences of five system types (Table 2).

The analyst determines all the system types by inspecting the system requirements specification documents. He/she must also assign a complexity value to each system type found. The complexity values are “low”, “medium” and “high” and are determined on the basis of certain rules provided by the International Function Point User Group [14] to
reduce subjectivity in the application of the method. The complexity values are translated to numerical values (weights). The next step is the calculation of the weighted sum of all system types, which is further adjusted by the value adjustment factor, a coefficient that is based on 14 technology complexity parameters. The final result is the function point count, obtained through the use of an empirical mathematical formula. As an example, the Microsoft Word for Windows version 6.0, has approximately 5000 function points.

Table 2
System types used in function point analysis

<table>
<thead>
<tr>
<th>System types</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal logical files (ILF)</td>
<td>Logical groupings of data in a system that are maintained by an end user</td>
<td>Inventory issue records, payroll records, product sales</td>
</tr>
<tr>
<td>External interface files (EIF)</td>
<td>Groupings of data from another system that are used only for reference purposes</td>
<td>Application data extracted and read from other applications, parameter data maintained outside the application</td>
</tr>
<tr>
<td>External inquiries (UI)</td>
<td>Allow users to select and display specific data from ILFs or EIFs</td>
<td>Item description, employee data, payment data, logon screens, display, browse user functions</td>
</tr>
<tr>
<td>External inputs (EI)</td>
<td>Functions that allow users to add, change, delete data in ILFs</td>
<td>Screen input of a sale, a receipt, an appointment, an order; messages from other applications</td>
</tr>
<tr>
<td>External outputs (EO)</td>
<td>Outputs of data that reside in ILFs and EIFs produced by the user</td>
<td>Account statements, weekly sales reports, bar coded labels, payroll checks</td>
</tr>
</tbody>
</table>
in total [1]. An application for spell checking would have 50–100 function points.

There are many advantages related to function points. They provide a mechanism that both software developers and users can utilize to define and measure functional requirements. They measure both user data and user needs to address these data and they are independent from the programming language used. The error associated to function point measurement by different experienced analysts was found to be relatively low: a mean difference of about 10% was reported in a case study by Kemerer [19]. Another important aspect of function point analysis is that it has reached a certain level of maturity, i.e. it has been applied by many software organizations, for quite a long time, to evaluate their systems capability from a user’s point of view. Nowadays a considerable amount of projects measured in terms of function points are available, providing the possibility to perform statistical analysis on them.

3. The case study dataset

The database used in our analysis is the International Software Benchmarking Standards Group (ISBSG) project repository, release 6, which contains currently 789 projects [15]. ISBSG is a non-profit organization, which maintains, develops and exploits a repository of international software project metrics to help software developers with project estimation and benchmarking. The increasing number of publications of studies based on the ISBSG releases suggests that this international cost database is becoming the standard benchmark for SCE research [4,17,22,28].

Projects in the ISBSG database are characterized by six attribute groups:

1. project nature (e.g. organization type, business area type, application type, intended market, development type, application development platform, application target platform, architecture, development methods, project time).
2. project work effort data (in man months);
3. project size data (in function points and optionally in lines of code);
4. project quality data (application defects, user satisfaction);
5. project cost data (monetary cost);
6. project estimation data (aiming to provide an indication of how closely the initial estimate matched what actually occurred).

The collection procedure pays much attention to the quality of the submitted data. A specific field is used to accommodate a rating code (A, B, or C) applied to the project data by the ISBSG quality reviewers to denote the quality of the submitted information (e.g. precision of the measurements).

In order to assess the accuracy of the analogy-based method, a suitable project subset had to be devised. In particular, it was attempted to include all types of projects that concerned various supply chain activities and other supporting activities. We did not distinguish the types of enterprises (e.g. companies from specific industry sectors) relying on the power of the analogy method to find correct analogies between projects. Only projects with data quality rating A or B have been considered, following a recommendation by ISBSG (rating C implies that the project submitted had some fundamental shortcomings in the data).

Another important decision concerned the set of attributes that could be used in an estimation process. The attributes chosen are shown in Table 3, along with a mnemonic code and a short description. All of them can be determined from the project specifications apart from two, namely maximum team size and project elapsed time, that can be determined by the project managers using some expert judgment approach or by simply guessing.

All of the above attributes were considered to be related to productivity in some way, and therefore their contribution to project distance and analogy accuracy was justifiable. For example, function points represent the system size and functionality. It is known from various other studies [5] that projects of similar size exhibit similar productivity. This is due to the software team size and organization characteristics (e.g. large teams have lower productivity because of the higher interaction needed between the team members). Development type is important because is directly related to productivity. Working on an existing system (i.e. maintaining the system) may be harder because of missing documentation, low structural code quality, etc. Finally, implementation date is also important because productivity is expected to improve over
time. Higher productivity rates are obtained by better programming tools and languages, and software processes. Based on the above considerations, ISBSG projects were selected according to their organization type (OT) and business area (BAT) characteristics in order to build the project dataset for the case study. In particular, the projects selected were classified as logistics, manufacturing and inventory projects with respect to either their OT or BA attributes. This process resulted in the formation of a supply chain dataset (SCDS) composed of 59 projects. Table 4 and Fig. 2 present the descriptive statistics and histograms, respectively, that demonstrate that SCDS projects exhibit various sizes, development types, programming languages, etc. The high S.D. values in Table 4 are due to an outlier project, much larger than the other projects in SCDS. The selected projects have been implemented between 1990 and 1997.

4. Method tuning and application results

A number of decisions were necessary before running the experiment. The following estimation strategies were tried.

1. Estimate using as many projects as possible. This strategy required the exclusion of all the attributes for which some project(s) had missing values. The strategy was coded under the name PROJECTS.

2. Estimate using as many attributes as possible. This strategy required the exclusion of all the projects that had some missing value(s) for any attribute(s). The strategy was coded under the name ATTRIBUTES.

3. Find a trade-off between strategies 1 and 2. This required the exclusion of the attributes for which many projects (more than three) had missing values. The strategy was coded under the name TRADEOFF.

4. Take into account, in all of the above cases, the two attributes of Table 3 that require human estimation (maximum team size and elapsed time). This

Table 3
Attributes from the ISBSG dataset used in the supply chain dataset

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Mnemonic code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function points</td>
<td>FP</td>
<td>The number of function points</td>
</tr>
<tr>
<td>Value adjustment factor</td>
<td>VAF</td>
<td>The technology adjustment factor used in function point analysis</td>
</tr>
<tr>
<td>Summary work effort</td>
<td>SWE</td>
<td>The total effort in recorded man hours</td>
</tr>
<tr>
<td>Resource level</td>
<td>RL</td>
<td>Indicates the team composition (e.g. RL = 4 includes end user and client effort)</td>
</tr>
<tr>
<td>Max team size</td>
<td>MTS</td>
<td>The maximum number of people that worked at any time on the project (peak team size) can be estimated by experienced project managers</td>
</tr>
<tr>
<td>Development type</td>
<td>DT</td>
<td>New development, enhancement, re-development</td>
</tr>
<tr>
<td>Development platform</td>
<td>DP</td>
<td>PC, mid range or mainframe</td>
</tr>
<tr>
<td>Language type</td>
<td>LT</td>
<td>2GL, 3GL, 4GL or application generator</td>
</tr>
<tr>
<td>Primary programming language</td>
<td>PPL</td>
<td>PL/1, Cobol, Access, C, etc.</td>
</tr>
<tr>
<td>DBMS used</td>
<td>DU</td>
<td>Whether the project used a DBMS</td>
</tr>
<tr>
<td>Project elapsed time</td>
<td>PET</td>
<td>Total elapsed time for project in months can be estimated by experienced project managers</td>
</tr>
<tr>
<td>Implementation date</td>
<td>ID</td>
<td>Actual date (year) of implementation</td>
</tr>
<tr>
<td>Organization type</td>
<td>OT</td>
<td>The type of organization that submitted the project (e.g. manufacturing, banking)</td>
</tr>
<tr>
<td>Business area type</td>
<td>BAT</td>
<td>Business area being addressed by the project (e.g. personnel, logistics)</td>
</tr>
<tr>
<td>Application type</td>
<td>AT</td>
<td>Type of information system produced by the project (e.g. MIS, process control)</td>
</tr>
</tbody>
</table>

Table 4
Descriptive statistics for the supply chain dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FP (man hours)</th>
<th>SWE (man hours)</th>
<th>PET (months)</th>
<th>MTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1,067</td>
<td>5,611</td>
<td>11.3</td>
<td>6</td>
</tr>
<tr>
<td>Median</td>
<td>324</td>
<td>2,430</td>
<td>10.0</td>
<td>5</td>
</tr>
<tr>
<td>S.D.</td>
<td>2,670</td>
<td>10,867</td>
<td>9.0</td>
<td>3.6</td>
</tr>
<tr>
<td>Maximum</td>
<td>17,518</td>
<td>55,960</td>
<td>52.0</td>
<td>20</td>
</tr>
<tr>
<td>Minimum</td>
<td>11</td>
<td>97</td>
<td>1.0</td>
<td>1</td>
</tr>
</tbody>
</table>
approach resulted in three additional strategies coded under the names PROJECTS-H, ATTRIBUTES-H and TRADEOFF-H, respectively.

Other calibration strategies are also possible but trying all of them is an unfeasible goal for datasets of even moderate size. In applying the method for all six strategies, we followed a procedure that, according to previous experience with the method and the BRACE tool, produces satisfactory estimation results. The approach consists in finding first the best parameter configuration followed by the choice for the best attribute set. Also, while running the case study the same weight was assigned to all attributes in order to calculate distances. The results are shown in Table 5.

4.1. Discussion of the results

All the strategies produced interesting results and none of them was very far from the aforementioned MMRE and PRED(25) target values, with the exception of PROJECTS and PROJECTS-H. The strategy that aimed at the utilization of the entire dataset (PROJECTS) produced the worst precision results. The inclusion of the two attributes that may be provided through human estimations (MTS and PET in PROJECTS-H) improved the results slightly, achieving an MMRE value of 39.64% but not a satisfactory PRED(25) value. Poor behavior may be explained because of many attributes being excluded in order to have all the projects available for estimation. It can be conjectured that this fact reduced the power of the analogy method.

The strategy that aimed at the opposite direction, i.e. the availability of as many attributes as possible, reduced the number of projects to only 16. Results were much better here (MMRE = 31.71% and PRED(25) = 75.00%) but the inclusion of MTS and PET did not produce any improvement. In addition, only 4 and 2 attributes respectively have been
found to give the best results, although attribute availability for finding analogies should be the advantage of this approach. The behavior of this type of strategy can be considered somewhat counterintuitive. The good accuracy results may be biased because of the small number of projects that remained in the final dataset.

The TRADEOFF strategy produced the best accuracy results, achieving an MMRE value of 23.84% and PRED(25) of 70.37% when MTS and PET were employed. The size of the dataset was 30 and 26 projects for TRADEOFF and TRADEOFF-H, and 8 and 7 attributes respectively were found to provide the best results. It is reasonable to claim that this strategy, based both on the analogy method and human estimation, not only provides the highest accuracy but also is more sensible than the other strategies.

An attempt to generate an estimation model using OLS, taking into account only size as independent variable, produced an MMRE value of 68%. It must be noticed that the generation of regression models using also the qualitative variables of categorical nature such as development type may produce better results [4], but it is still a more cumbersome and less mature technique than OLS and analogy.

Overall, the results may be considered satisfactory, especially if we consider the issue of data uniformity. The projects in SCDS come from a variety of different organizations and have been developed in different countries. Better accuracy might be obtained if projects from more uniform sources are used for deriving analogies. For example, an IT buyer may collect data from a single vendor that has already implemented a number of projects. Cost data may also be collected by national organizations (such as technical chambers or national statistical societies) providing a more uniform database than the ISBSG subset used in our study.

As an example of the estimation, consider project number 11774 from the ISBSG dataset (Fig. 3). It delivered a relatively small system (FP = 216 and SWE = 2422) and belongs to the aerospace/automotive type of organization and the business area of manufacturing. When estimated with the best strategy (TRADEOFF-H) the project with the closest analogy was found to be project number 31735, a project with almost identical attributes, FP = 89 and SWE = 1106. The mean estimate was 2322 h with a 95% confidence interval ranging from 1232 to 2684. Considering the mean estimate, the estimation error was only 3.6% with respect to the actual effort. From the estimate distribution (derived through the bootstrap procedure) it was determined that the most probable estimate lied in the range 2000–2500 (65.1% probability), with a significant probability (23.1%) assigned to the range 2500–3000 and a much smaller probability (10.8%) assigned to the range 1000–1500. Practically, this means that if the acquiring organization allocated a budget according to an estimate of 3000 h, cost overruns would be avoided with an estimated probability of 99%.
projects and the final measurement of the relative error from the actual and the estimated cost can be considered as a simulation of the real situation where a new project has to be estimated under the assumption that its attributes will remain unchanged throughout the development. However, we must take into account two important sources of uncertainty when estimating the cost of a software project.

(a) There is an inherent uncertainty when we calculate the size of the project or when we assess the qualitative project factors, such as complexity.

(b) The project attributes may change throughout the development due to modifications in user requirements or management decisions.

Regarding the former source of uncertainty, if our assessment is not precise, the error of the cost prediction can be very high. Regarding the latter source of uncertainty, it is clear that under any estimation method, in case of alterations of the initial project attributes the estimation must be repeated, taking into account the new data available.

5. Conclusions and future work

In this paper, the importance of accurate prediction of development effort in information system planning and cost control has been discussed. A method for systematic software cost estimation combining: (a) the analogy-based method; (b) function point sizing; and (c) calibration and evaluation of the accuracy, has been proposed. The method can be applied at the requirements definition phase, i.e. quite early within the system development cycle. Cost prediction is obtained by comparing the target project to completed projects with known characteristics. The overall method accuracy depends on a number of options and it has been shown how calibration may be achieved through the use of a suitable software tool.

The overall approach has been experimented on a project subset derived from a large project repository, composed of typical systems employed by the manufacturing companies. The estimated projects delivered functionality related to the supply chain activities. Different strategies have been tried in order...
to both experiment with the method and circumnavigate the problem of the missing values. The best results have been obtained by a strategy that aimed at a trade-off between the number of attributes and projects, enhanced by the estimations that may be obtained by humans. The results obtained are acceptable compared to the target values set in the software cost estimation literature. The accuracy of the overall approach may be improved if more uniform cost datasets are used. Overall, the case study results showed that software cost estimation is viable for this type of systems, indicating that enterprises may control the costs of developing the software based on similar methods.

The method should be applied to live projects and its accuracy should be assessed carefully. There are also a number of issues where improvements may be obtained. One such issue is the data collection procedure. Care should be taken to avoid the missing values in project characterization, since this results in the exclusion of the project from the cost analysis. Fully characterized projects render also easier the formation of uniform datasets of reasonable sizes. Additionally, data collected should be as precise as possible.

Further enhancements of the method may be also pursued. For example, the combination with least square regression may be of interest, in the sense that the latter may be used when the distance values of the closest analogies are too high, indicating a target project that presents many peculiarities with respect to the completed projects. A replication of the case study is also important, using subsequent ISBSG releases (a new release is already available, containing more than 1000 projects). This will allow the application of the method to larger datasets, with projects potentially exhibiting different productivity rates and technical characteristics. In any case, each new ISBSG release will present a new challenge for the analogy-based method, since the analogy mechanism should detect new relationships among projects and direct estimate generation accordingly. Other interesting aspects should also be explored, such as IT project portfolio estimation [29].

Finally, a major target should be the cost estimation of projects implementing ERP systems in an enterprise. Sizing techniques such as function point analysis may be tried in this area too. For example, the effort for customizing a commercial ERP system may be modeled around the number of system’s function points that should be modified. However, accurate estimation needs historical data, implying that ERP projects should be characterized with appropriate attributes and data should be systematically collected for this project category as well.

Acknowledgements

We would like to thank the anonymous reviewers for helping us improve the paper in various ways.

References