EVALUATING A SOFTWARE COSTING METHOD BASED ON SOFTWARE FEATURES AND CASE BASED REASONING

Christopher Irgens,
University of Strathclyde,
Department of Design, Manufacture and Engineering Management,
75 Montrose Street, Glasgow G1 1X, UK.
E-mail:chris.irgens@dmem.strath.ac.uk

Sherif Tawfik,
Arab Academy for Science and Technology
Information and Documentation Centre,
P. O. Box: 1029, Alexandria, EGYPT.
E-mail:Sherif226@hotmail.com

Lenka Landryova,
VSB-Technical University of Ostrava,
Department of Control Systems and Instrumentation,
tr.17. Listopadu 15, 708 33 Ostrava, CZECH REPUBLIC.
E-mail:lenka.landryova@vsb.cz

A common weakness of most software cost estimating models is their limited usefulness to predict the cost accurately and quickly at an early stage of the software development life cycle. This is due to the lack of data about the code quantity and code complexity at that point in the software cycle.. A method for software cost estimation has been developed to address the problem of the early and accurate establishing of cost based on Case Based Reasoning. Using the software type, functions and primitive ***(x)R(y)*** *Z-operations as parameters for characterising and handling the cases.*
The output from the model has been analysed with respect to actual recorded historical test-case cost.

1. INTRODUCTION

The software cost estimation in the early stages of the software development life cycle depends on the requirement specification of the software. The approach is to divide the software system into standard tasks, functions, and operators. The CBR process is made easier since one software system provides accumulated knowledge of a number of standard functions; each in turn providing accumulated knowledge of a number standard operators, all of which can be expressed in Z. Cases can be characterised and constructed according to standard functions and operations/operators expressed in Z. In this way the need for a large *project case-base* is diminished. Thus the main CBR disadvantage is also diminished, (Aamodt et al,1994), (Kolodner, 1993). This provides a 'featuring' mechanism that can be used to project information as has already been shown valid and useful in the product design domain (Irgens,1995).

Although there have been examples of successful CBR tools for software cost estimation (Mair, 1999), (Finnie, 1997), (Schofield, 1998), (Prietula et al, 1996). Most existing methods use the available software features to predict the software size. Then this size is used as input to one of the algorithmic models such as COCOMO or Function Point Analysis to estimate the cost. Other research concludes that in specific application domains, process control application, it is possible to estimate the software size from the user specified application features. The limitation here is that the application is restricted to a very specific domain (Mukhopadhyay et al, 1992). Sarah Jane Delay states that it appears impossible to identify features early in the life cycle that defines the size of the project (Delay, 1998).

2. THE METHOD

The new method is mimicking the successful approach taken in the product design domain (Irgens, 1995) by creating a 'feature-based' approach based upon the decomposed functionality of the software as specified during the design stage of the life-cycle. In order to provide the necessary standard notation and rigour, the use of a formal specification language was used.

Z is a formal specification language, which has been progressively developed and widely applied since its inception at Oxford University in 1980 (Sommerville, 1997). Formality implies precise, unambiguous description in the tradition of mathematics. Z is based on typed set-theory, coupled with a structuring mechanism (i.e., the schema calculus is one of its key features). A schema is a collection of variable declarations and predicates giving the relationships between the variables. This basic construct is used to structure the description of a system. The schema is divided into two main parts: declarations (or signature) and predicate. Declarative information such as object names and types are given in the signature. The predicate section provides the relationships between those objects that must hold. Preconditions and post-conditions typically are not labelled explicitly. Z describes both the state space and transitions on the states and places no restrictions on the style of specifications. The schema calculus further defines the rule of operations on schemas such as inheritance, composition, and information hiding.

1.1 The Model Mechanism

The implemented model is seen as two main parts: The first part is designed for handling the operations of adding and updating the cases in the case library. The second part is designed to enable the software project manager to determine the project effort in the early stage of the software development life cycle.

1.1.1 Part One (Adding and Updating Cases).

This part consists of programs for handling the data which have been gathered from previous historical software projects (cases) into the case library. Each of these programs has standard functions to make it simple for the project manager to handle the data in the case library.

1.1.2 Part Two (Software effort estimation process)

For determining the software project effort estimation, the project manager will go through one or more than one stage in this implemented part. These stages are as follows:

Stage (A)

The first step in this part will ask the project manager to specify the categorisation level of the software project and then to specify the system activity class that the new system belongs to.

There are two possible directions in this stage. If there are historical cases found, these projects will be displayed. Otherwise, the user will be directed to stage (B).

The user can choose one or more from the historical projects, and choose those most suitable to the new task. If the new software systems tasks are covered completely in this step, then the project manager will go directly to the next step. Otherwise, he will proceed to stage (B).

The project manager will select the appropriate project features, which affect the computing of the total estimated effort for the new project.

In this step, the model will use all the input data for estimating the new project effort and gives the result to the project manager.

In this step, the project manager could adapt the result or accept it as it is and add it to the case library.

Stage (B)

If there is no previous project for both the input categorisation and the input system activity class, or there is a new task that is not matched by any previous projects for the same categorisation and system activity class, the project manager has to divide every task into it's primary functions and specify every function by using the Z function specification to count the number of occurrences for every operator in each function in the task.

The project manager has to input the type of every function in the new task and complete by entering the number of occurrence of every operator type. Historic cases are retrieved using the nearest neighbour matching technique. The project manager can accept the effort suggested or he can modify the result by himself and store the result. The project manager will stay in step 2 until finished entering all the required new tasks of the system.

3. THE EVALUATION OF THE METHOD

The objective of the evaluation process is to measure the extent to which the model meets its predicted performance, it is necessary that the evaluation includes:

- o Evaluation of the retrieval algorithm in the CBR function;
- o Evaluation of the method used in the CBR function to calculate the cost;
- o Evaluate the output result with respect to the actual cost.

3.1 Test Data Collection

The historical software function and cost data was needed in order to build the case library used to evaluate the prototype. The data collection was done with the help of the staff in the Information and Documentation Centre in the collaborating establishment The Arab Academy for Science and Technology and Maritime Transport (AASTMT) and was based upon historical cost information from software systems developed for internal use.

Four large completed historical software projects provide the functional and historical cost data used to evaluate the implemented model. These projects were:

- o Pharmacy Inventory system;
- o Clinic accounting system;
- o Food Inventory system;
- o Cafeteria Sales system.

3.2 Data documents

A number of documents are used in AASTMT to manage and record its software projects. The test data was collected from historical project information regarding task, functions, operations and associated spent project hours. Furthermore, the estimation method requires design information so that the software may be characterised by function type and operations profile. The number and types of Z operators are used for this purpose. Therefore a *Function Specification in Z-Notation* is also required. Each project is simplified to its basic functions. All functions are described in Z-notation and each predicate is further simplified into simple predicate in the form of ***(x) R (y)***. The number and types of the function's simple predicates composes the function's characteristic feature and can be used for the case-based search and reasoning mechanism. Every function is therefore summarised and characterised using the number of occurrences of every Z operator. This operation/operator profile forms the key feature for every basic function.

Table 1 shows a sample of Z function specification for the simple function **'add-account'**, while table 2 shows the number of occurrences of Z operators of the form ***(x)R(y)*** in **'add-account'**. These operator sets are the key *features* for every basic function forming the functions characteristics. In this manner the software project may be characterised by function type and operator density by type.

Table 1, A Sample of Function Specification in Z-Notation

add-account
HB1-mainacc : HB1mainacc
HB1-mainacc' : HB1mainacc
HB1-mainacc-record : account-key ⇸ account-data
S-mainacc-no? : mainacc-no
S-mainacc-name? : mainacc-name
S-acc-kind? :: = Madin | Dain
a-data : **P** account-data
Mess! : report
(S-mainacc-no? ∉Dom HB1-mainacc-record
a-data = (S-mainacc-name?,S-acc-kind?)
HB1-mainacc ' = HB1-mainacc ∪{S-mainacc-no? ↦ a-data}
Mess! = "Main Account was added")
∨
(S-mainacc-no? ∈ Dom account-record
Mess! = "Main Account already exist")

Table 2, The number of occurrences of Z-operators in **'add-account'**

Operator	Number of occurrences
:	7
?	8
!	3
⇸	1
∉	1
↦	1
=	4
∈	1
∪	1
If-else	1
:: =	1
P	1
|	1
,	1

4. EVALUATION RESULTS

Using the Food Inventory system tasks as test case for the model produced the results in Table 3.

The error was measured using Mean Magnitude Relative Error (MMRE):

MMRE= $\sum_{i=1}^{n}(|(estimate_i - actual_i)/actual_i|)/n$, where estimate$_i$ is the estimated effort in hours, from the model, actual$_i$ is the actual effort, and n is the number of 'projects'. To establish whether model is biased, the Mean Relative Error (MRE) was used:

MRE$_i$ = $(estimate_i - actual_i)/actual_i)$

Table 3 shows the estimated task-hours against the actual historical records, with the corresponding MRE and MMRE values.

Table 3, Estimated hours against actual recorded hours

Task number	Estimated hours based on CBR	Actual hours	MRE for CBR estimates
AS.01	6.00	6.00	0
AS.02	8.59	8.00	0.073
AS.03	7.58	7.50	0.010
AS.04	13.54	13.50	0.003
AS.05	6.00	6.00	0
AS.06	5.00	3.00	0.667
AS.07	3.00	3.00	0
AS.08	3.00	2.50	0.200
AS.09	3.00	2.00	0.500
AS.10	3.00	2.50	0.200
AS.11	2.00	2.00	0
AS.12	3.75	2.50	0.500
AS.13	2.50	2.50	0
AS.14	3.75	3.00	0.250
AS.15	8.40	8.00	0.050

Giving an MMRE=0.164.

5. CONCLUSION

The prototype was implemented in Microsoft Windows environment using Access DBMS for the working data. The results obtained were satisfactory showing acceptable variation from actual historical values. The collaborating partner has consequently advised its information centre to continue the work in order to develop the prototype for practical purposes.

The limitations can be summarised as follows:

1. The software engineers needed some time to become familiar with the model, specially the CBR function part, which depends on the good familiarity with Z notation, and the process of decomposing each complex predicate into a group of a simple predicates.
2. The implemented prototype is a first version prototype, thus it was difficult to collect and analyse more than the four historical software cases used. However, due to the nature of the method, the resultant decomposition of each software project into its constituent functions and (x)R(y) simple predicates allowed the evaluation of the method as reported above.

It is also quite clear to the authors that:

1. The strengths of both the expert judgment method and the analogy method are combined within the implemented prototype.
2. The new approach uses the software specification, which is closer to the user requirement. Moreover, instead of specifying the software project as one entity, it divides the software system into standard functions, each function is represented in standard Z notation, thus mimicking the successful feature based methods used in product design.
3. The CBR process is made effective since one software system provides knowledge of a number of standard functions; each in turn provides knowledge of a number of standard operators. In this way the need for large number of historical projects is diminished. Thus, by using a small number of historical projects the case library can be adequately populated.

6. REFERENCES.

1. Aamodt, A., Plaza, E. "Case-Based Reasoning: Foundational Issues Methodologies, Variations, and System Approaches", AI Communications. Vol. 7, pp (39-59), 1994.

2. Delay, Sarah Jane et. al. "The Limits Of CBR In Software Project Estimation". 6th German workshop on case based reasoning (GWCBR'98), eds. L.Gierl , M. Lenz, Berlin, pp. (99-108), 1998.

3. Finnie, G. R "A comparison of software effort estimation techniques: using function with neural networks, case-based reasoning and regression models". Journal of systems and software, Vol 39, pp (281-289), 1997.

4. Irgens, C. "Design Support based upon the projection of information across the Product Development life-cycle by means of Case Based Reasoning", Journal: IEE Proceedings on Science, Measurement and Technology Special issue on Manufacturing, Sept.1995, pp:345-349,Volume: 142, Issue:5, ISSN: 1350-2344.

5. Kolodner, Janet. "Case-Based Reasoning". Morgan Kaufman Publishers, California, 1993

6. Mair, Carolyn et. al. "An_Investigation of Machine Learning Based Prediction System". 9 July, 1999. http://dec.buth.ac.uk/ESERG

7. Mukhopadhyay, Tridas ; Kekre, Sunder. "Software Effort Models For Early Estimation Of Process Control Applications". IEEE Transaction on software engineering. Vol. 18, No. 10, PP (915,924), August 1992.

8. Prietula M. et. al. "Software Effort Estimation With A Case Based Reasoning". Journal of Experimental and Theoretical Artificial Intelligence. 8(3-4) PP (341-363), 1996.

9. Schofield, Chris. "Non-Algorithmic Effort Estimation Techniques". Department of Computing, Bournemouth University. ESERG: TR 98-01. 1998.

10. Sommerville, Ian; Sawyer, Pete. "Requirements Enginerring A good Practice Guide". John Willy & Sons Ltd. 1997.