A Data Complexity Formula for Deriving Time-to-Build Estimates from Non-relational to Relational Databases

Prof. Philip J. Sallis[1]
Department of Information Science
University of Otago

November 1993

Abstract

Despite the many qualitative elements of software time-to-build estimating, some observable features can be quantified; even if the resulting set of variables observed is arbitrary. Such is the case when estimating the expected duration for database re-engineering. If we assume that for any extant database, an entity-relationship model(ERM) can be produced from which a new normalised schema is generated, then our estimating task needs to quantify both the complexity of the ensuing ERM and also the data modelling knowledge of the 're-engineer'. Whilst there may be additional variables to be considered, a set of primary elements required for estimating the duration of the task have been identified. The formula proposed in this paper is arbitrary but it is intended as an instrument for measuring ER model complexity, such that time-to-build estimates can be made for the task of re-engineering extant non-relational databases into relational form.

[1] Address correspondence to: Prof. P.J. Sallis, Chairman, Department of Information Science, University of Otago, P.O. Box 56, Dunedin, New Zealand. Fax: +64 3 479 8311 Email: psallis@commerce.otago.ac.nz

1 Introduction

When an existing set of data structures no longer best serves the needs for system efficiency or effectiveness, a decision must be taken either to begin again the process of data definition or to re-engineer the existing database such that it fits the new requirements; probably within a relational structure.

This paper describes a formula based on the assumption that it is possible to identify the variables that exist in the process of re-engineering an existing database, such that duration or 'time-to-build' estimates can be made. It is primarily intended that this paper should describe the formula and the concepts and assumptions which are at its foundation, but a description of some on-going experimentation with the formula is given as an indication of the way forward for the research.

The work arises from a belief that in many cases there exists a desire to re-engineer existing databases but from a management perspective, knowledge of the complexity of the task and thereby its duration, is difficult to ascertain. In the many studies of why systems are late, poor estimation is most often seen as a primary cause. In a recent survey by van Genuchten(1991) he states that of the 80% of respondents who said that their projects were sometimes or usually late, 51% pointed to over-optimistic estimation as the most often used reason and some 9% said these were always a reason. In short, managers continue to depend upon poorly founded estimates from computing 'professionals'.

Following the earlier celebrated work by Brooks(1975), sufficient evidence has since been accumulated [see for example Boehm(1981) and more recently Bollinger and McGowan(1991)] to state generally that system complexity is directly related to system size. Intuitively at least, it seems reasonable to assume that duration is thereby also related to complexity because size must be a primary factor in any building (or re-building) task. Futhermore, although the task of creating a reliable duration estimation formula may be elusive, it seems intuitively possible to quantify sufficient aspects of the process in order to bring it greater precision. Particularly, this assumption seems sound when the entity-relationship approach to modelling data is used. This approach [see Benwell, Firns and Sallis (1990)] is now well entrenched in both theory and practice, with ample evidence of its rigour and suitability for deriving logical data models from which relational database schemas can be generated.

In order to construct an entity-relationship model (ERM), data entities, their attributes and relationships between one another need to be identified (therefore, 'quantified'), and the time it

takes to accomplish this task depends upon the knowledge of the data modeller. Further, the inherent complexity of the model (and therefore, of the system), impacts on the data modeller's knowledge and ability so that it will ultimately influence the re-engineering duration.

So much for the complexity of the ERM and its impact on the duration of the re-engineering task, but the existing database itself has inherent complexity and the level to which it is analysed will also further influence the duration estimate. Davey (1992) refers to three levels of complexity for the re-engineering of databases. *Level one* is to develop a conceptual model from the record or table definitions of an extant database. *Level two* is to develop a conceptual model by analysing attribute definitions, thus defining functional dependency between attributes. *Level three* is to conduct an analysis of the database contents. For an ERM to be constructed, at least *Level two* needs to be entered upon and obviously *Level three* analysis will be required.

The time-to-build (duration) issue raised by this paper is two-fold. First, in relation to the database re-engineering task, we need to measure the extent to which both data structure complexity and the engineer's knowledge (and therefore, 'productivity potential') combine as factors in the duration formula. Second, the influence of the engineer's knowledge factor on the duration formula needs to be assessed.

A well-known formula for estimating duration of software development that has been used with some success is **D = C(G+J)**. Rakos(1990) provides a comprehensive discussion with examples of this formula based on the original 1967 paper by IBM. In this formula **D** is duration, **C** is complexity, **G** is general knowledge and **J** is number of years experience. The complexity factor is a weighted value from a table of system function points and programmer productivity is a function of knowledge and experience. The factor tables for General Experience(**G**) and Job Knowledge(**J**) are reproduced here in *Appendix A*.

There are serious questions of validity of the factors in this formula. For example, the global applicability of the function points and their weights and the variability of programmer skill over time, the availability of productivity aids in the form of software tools, working conditions, and mood changes to name a few. Nonetheless, in this paper, the formula is considered extendable for use when estimating database reverse engineering time.

2 A New Formula

A first version of a new formula is proposed, **D** = **C(K)** where **D** is re-engineering duration, **C** is ERM complexity and **K** is an individual's knowledge of data modelling techniques. The formula relates to the creation time for ERMs, not programs, so values for the complexity factor **C** need to be calculated using an entirely different set of characteristics from those weighting factors given in *Appendix A*. The knowledge factor **K** will also be derived differently from the tables in *Appendix A*.

Extending this 'first version' a little, but not using a set of weighting factors, the complexity factor **C** is redefined by the equation $\mathbf{C} = \Sigma R_{a+p}$ where R is a relationship between two data entities, a is the number of attributes in each of the two related entities (entity attributes used in more than one relationship are counted again for each entity pair), and p is a participation value associated with each relationship. Participation p is the summation of $c(o+d)$ where c is cardinality, o is optionality and d is dependence for each relationship. Cardinality is given a multiplicitive operation because it is considered to be of an order more complex in nature than either optionality or dependence.

The values of p are calculated from an index where cardinality (not including many-to-many relationships) is either [**1** for a 1:1 relationship] or [**2** for a 1:N or a N:1 relationship], optionality is either [**0**] or [**1**] and dependence is either [**0**] or [**1**]. The formula multiplies c by the summed values of o and d to produce a participation value p.

The complexity summation $\mathbf{C} = \Sigma R_{a+p}$ always assumes two data entities for each relationship (**e=2**) as a constant, where R=**1** for each summation of a,p. Therefore, if five relationships each return a value of 24 (being the result of adding a total of 20 attributes (**a**) for each entity-pair to a participation value(**p**) of 4 for each relationship **R**), the resulting complexity 'score' will be 120. This would be true for the ERM shown below in Figure 1.

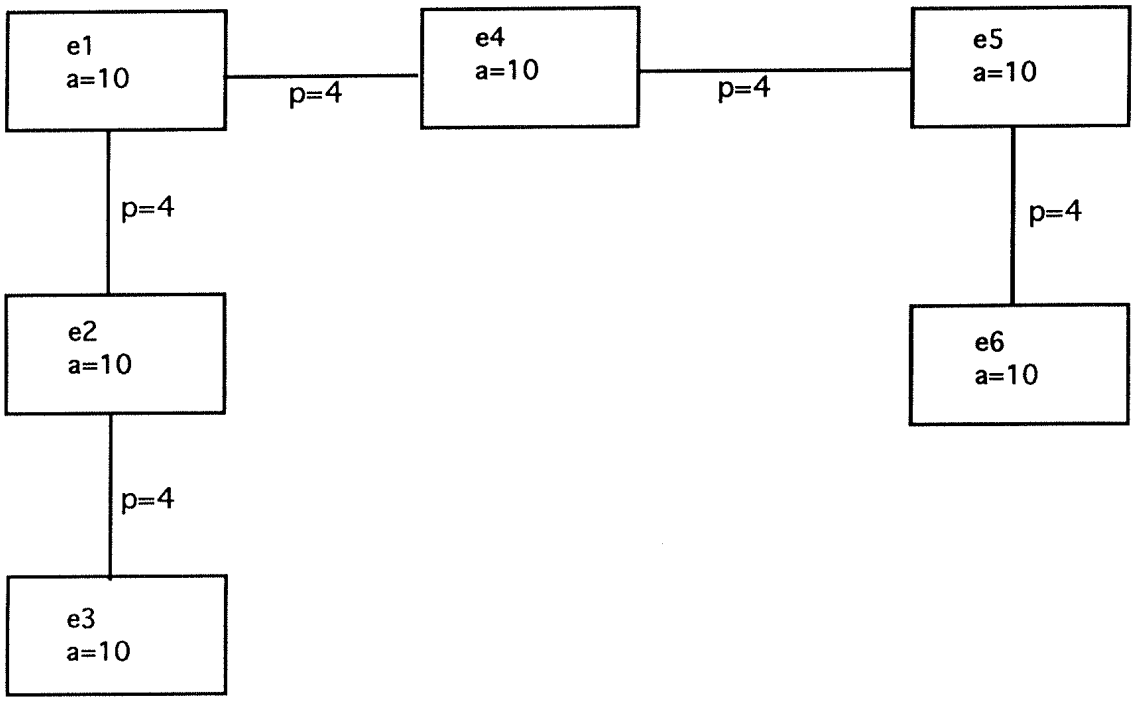

Figure 1 Attribute and participation values for six entity-relationships

Assuming the number of attributes a will be a considerably larger number than the value for p, it is felt appropriate to modify the value of a for each occurrence of R such that the participation factor (p) contributes significantly to the complexity calculation. Obviously, the components of p are vital to the degree of complexity for any given data model.

In this 'first version' the number of attributes **a** dominates the formula so that some means of reducing this abnormality is needed. To acheive this effect, the $\log_2$ of the number of attributes **a** identified in each entity-pair is added to the participation value **p** to give a value for **R**. The values of **R** are then summed for the number of relationships that exist between entity-pairs **e** to give a complexity value for **C**. The duration **D** is then the result of **C(K)**.

As just stated, in order to cater for the magnitude sensitivity in the formula, it was decided to calculate the $\log_2$ (written $\lg$ below) for each occurrence of a. This gives a new 'second version' formula, $C = \Sigma R_{\lg a+p}$ for computing values for **C**. In the example relating to Figure 1 above, this would have the effect of returning a logarithmically modified value for **C** of 34.975, rather than of 120 as would otherwise have been the case. When thought of in terms of person hours for the data modelling task, the modified value appears more intuitively appropriate than the unmodified value. This result is incomplete for the duration calculation at this stage because a value for **K** has not yet been computed.

Values for **K** are individually computed on a more objective basis than merely using a weighted

table such as appears in the *Appendix A*. Discussion of how values for **K** are computed for the new formula is entered into later in the paper.

As mentioned, values for ***R*** are summed for the total number of entity relationships identified in the model. There is an argument for attempting to capture more precisely the non-linearity of the ensuing ERM by introducing a weighting factor each time an entity once counted is re-counted. In this way, an entity that is a member of two or more relationships in the model would be given a factored value representing the additional complexity its duplicate membership provides. Such a factoring would mean that the values for ***R*** would not simply be summed for the number of entities in the model; rather, as each duplicate entity is identified, a weighting would be added $\mathbf{e=n_{i,w}}$ and accumulated into the total. It is intended that experiments with the formula will compare results using both the unweighted and weighted methods.

3 Determining pre-ERM complexity

Obviously the complexity variables mentioned here all relate to ERM characteristics, which are not present in the pre-ERM data structure state. A method for identifying sufficient of these characteristics without conducting a complete data modelling exercise is needed. Once the ERM has been built, a major part of the re-engineering task has been undertaken thereby removing the need for a duration estimating formula.

The approach taken here is crude, but considered sufficient for duration estimation purposes without distorting the reality of the complex decision-making required when building ERMs. A conscious decision has been made <u>not</u> to develop an idiosyncratic category definition or extension to existing ERM terminology such as can be seen as successfully argued in Elmasri, Weeldreyer and Hevner(1985) and by others elsewhere. This may reflect a lack of rigour in the method but it is considered unnecessary for this particular exercise. This decision is supported in part by a recent observation by Cherniavsky and Smith(1991) on the inadequacy of axiomatic properties for software measures. Although their work relates to programming measures, the point is worth making that too much definition of data and process properties can lead to misunderstanding and perhaps imprecision.

Each data set, or file, is counted as an entity $\mathbf{e_i}$. Every data name with each data set is counted as an attribute **aj** for every **ei**. Where indexes exist linking two or more data sets, a relationship **Rn** is said to exist. Where primary and secondary keys exist, dependence **d** is said to exist (*value 1*), and cardinality **c** is said to exist (*value 2*) because the existence of secondary keys denotes a one-to-many relationship between participating entities. If

optionality can be ascertained, a *value 1* for **o** can also be assumed.

An example set of six file descriptions extracted from a total of seventy comprising an operational retail point-of-sale system is shown as Figure 2a below. This system and its files are being used for the experiments relating to the research described here. A model entity-relationship diagram for these data structures appears here as Appendix D.

File/Entity	Attributes
Customer Master	18
Product Master	15
Discounts	15
Barcodes	3
Sales Order Header	15
Sales Order Detail	15

Figure 2a Sample of a pre-ERM file description summary

In Figure 2b below, participation values are given for relationships assumed to exist between these files. These values are computed from observations made concerning the *cardinality*, *optionality* and *dependence* that is assumed to exist between the files, especially when they are re-engineered as an ERM for a future relational database.

Entity-to-entity	Participation Value
Customer,Order Header	4
Customer,Discount	1
Order Header,Order Detail	4
Order Detail,Product	4
Discount,Product	4
Product,Barcode	1

Figure 2b Participation values for related entities

To compute the values of **R**, we sum **a**,**p** for every entity pair, giving the results in Figure 2c below.

Entity-to-entity	p Value	log2	R Value
Customer,Order Header	3	3.4	6.4
Customer,Discount	3	3.4	6.4
Order Header,Order Detail	4	3.4	7.4
Order Detail,Product	4	3.4	7.4
Discount,Product	4	3.4	7.4
Product,Barcode	2	2.8	4.8

Figure 2c Values for R computed from attribute(a) and participation(p) values

If we now sum these results, we have a value for complexity **C** of 39.8 to be input to the formula along with an individual value for **K**. Eventually the formula will produce a duration result in person hours. For example, if the value for **K** was 1.75, with the value for **C** being 39.8, then the duration **D** would be 69.65 person hours. Of course this computation does not take into account the use of computer aided software engineering(CASE) tools or other productivity technologies that could be expected to reduce the time-to-build estimate.

4 Experiments With the Formula

As with any method that is intended for general use within a particular application domain, extensive experimentation is required to validate what is essentially an intuitive formula. Data collection and model building is currently in progress to produce some preliminary results but in describing the formula here, it is appropriate to enter into some discussion of the experimental design.

First, the method of obtaining values for **K** from individual data modellers, (or 're-engineers'), has been undertaken. In broad terms, the experiment takes the form of presenting candidates with an objective assessment instrument to determine their data modelling knowledge. Candidates are also asked to provide information relating to their work experience. It is considered vital that specific application related knowledge and experience are incorporated in the computation of **K**. When this data has been collected for an individual, their experience in years can be given a factor value from both the general experience (**G**) table and job knowledge (**J**) table in *Appendix A*.

The set ERM knowledge-related questions are given in *Appendix B*. Not shown here are the set of file descriptions or the instructions requesting participants to construct an ERM from them. The translation of answers from participants to numeric values representing their knowledge factor is given in *Appendix C*. This factor, whilst arbitrary, is an attempt to obtain a value for **K** that will be meaningful when used in the duration formula. The number of years experience is simply translated to the **G** table in *Appendix A*. It is proposed that an individual's score computed from the questionnaire will provide input to the translation table (**J**) in *Appendix A*, then together with the factor obtained from **G** a value for **K** can be derived.

The values for "YES/NO" answers are merely binary; each answer receives either 0 ir 1 as a result. The number of years experience reported in Questions 3 and 4 are taken at face value and translated simply in the ratio of 1:1 for each year. Mindful that experience is only quantifiable for a snapshot in time and that the arbitrary assignment of whole years is fraught with difficulties pertaining to individual variation, it must be stated clearly that this or any formula for estimation purposes is more likely to succeed if the factors for each variable have been based on more than intuitive assignm*ent*. In concluding his remarks concerning the original IBM formula, Rakos(1990) states, *"This method will work if you develop accurate factors...but as [with] any other estimating method, [it] depends upon how well you granularize."*

It may be possible to develop a more granularized set of factors following the analysis of a very large set of data, but this something for the future. As previously stated, the formula is presented here as a self-contained concept; the experiments are intended to extend its viability rather than merely vindicate its applicability. In a sense, the experimental work is a 'second phase' of this research, developed out of the conceptual discussion of method in this paper.

The experiments have an international aspect in order to observe regional differences in the values computed for **K**. Candidates are given a set of data structure definitions corresponding to **complexity level 2** as proposed by Davey (op cit). They are asked to list the entities, attributes and entity-relationships they identify in the given data structures. They are also asked to identify the 'nature' of the relationships between entity pairs in terms of *cardinality*, *optionality* and *dependence*. They are told at the outset not to be concerned if they are unfamiliar with these terms, but nonetheless to express the 'nature' of the entity-relationships in their own terms.

Once collected, an individual's result can be input to the duration formula with data from the ERM. This will produce a value for **D** in terms of design engineer hours required to re-engineer a given extant database or set of file structures.

5 Conclusions

This paper assumes that the construction of ERMs is appropriate and desirable as a means for defining extant non-relational databases prior to re-engineering into relational form. A set of ERM complexity characteristics have been described. In addition, a method for anticipating values from pre-ERM data descriptions that can be used to estimate the duration of database re-engineering tasks has been outlined. Both of these are necessary as input to a viable formula for use in duration estimating and both are factors for evaluating task complexity and the knowledge quantification of those undertaking the task. Both have been discussed in this paper with reference to a formula that has been modified from a long-standing method for estimating the time-to-build programs. Both the original and newly proposed formulae are arbitrary but they are intended as operational instruments to assist in the production of better quality time-to-build estimates than other methods of 'best guess' that might be used.

The research described in this paper is on-going. Considerable experimental work is being undertaken to ascertain the knowledge levels of individuals who may be involved in database re-engineering activities. Some of this experimental work is described above. It is hoped that the results from the experiments will suggest new directions for the research. For example, whether the complexity of the problem dominates the experience of the 're-engineer'. In other words, when **C** dominates **K**. It may be possible to calculate the worst case **K** and the best case **K**, then evaluate (incorporating margins of error) when **C** dominates. Some commentators have suggested that when software gets above a certain size, the ability of the programmer ceased to matter. Such may be the case with ERMs and database schemas.

Although this work examines data-related aspects of the re-engineering task, the point should be made that many user requirements for appropriate information derived from relationships between the data are not necessarily obvious from the data descriptions themselves. Either from the outset or over time, data integrity checks or rules for relating data are in fact, embedded in the program code. An extension to the work outlined here that, it is hoped, will provide an even richer insight to the complexity (and thereby, duration) of the data re-engineering task, will take these issues into account and will develop for example, an index of complexity factors such as values for data-set reads, writes and modifies.

Acknowledgments

Thanks go to several people who have listened and made suggestions, especially Steve MacDonell, Martin Purvis and colleagues at Otago University, but also to Derek Andrews of Leicester University for his ideas of extending the formula in future to include program complexity and for proof-reading the paper.

References

Benwell,G. Firns,P. and Sallis, P. Deriving semantic data models from structured process descriptions of reality. Jnl of Info Tech (1991) 6,15-25.

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

Bollinger, T.B. and McGowan, C. A Critical Look at Software Capability Evaluations. IEEE *Software*, 1991, 25-41.

Brooks, F.B. The Mythical Man-Month: Essays on Software Engineering London, Addison-Wesley, 1975

Cherniavsky,J.C. and Smith,C.H. On Weyuker's Axioms for software complexity measures. IEEE Trans on Soft Eng, Vol 17(6), June 1991.

Davey, James H. Database Re-engineering. CASE Trends, Nov 1992, 24-27

Elmasri, R.,Weeldreyer, J. and Hevner, A. The category concept: an extension to the entity-relationship model. Data & Knowledge Engineering 1(1985), 75-116.

Rakos, John J. Software Project Management for Small to Medium Sized Projects. Prentice Hall, 1990.

van Genuchten,M. Why is Software Late? An empirical study of reasons for delay in software development. IEEE Trans on Soft Eng, Vol 17(6), June 1991.

Appendix A

IBM Productivity Factor Tables

(cited from Rakos,1990)

Table G: General Experience Factors

Programmer type	Years of experience	Factor Range
Senior	5 +	0.5 - 0.75
Average	1.5 - 5	1.0 - 1.5
Junior	0.5 - 1.5	2.0 - 3.0
Trainee	0.0 - 0.5	3.5 - 4.0

Table J: Job Knowledge Factors

Job Knowledge	Knowledge Required		
	Much	Some	None
Detailed knowledge of this job and detailed knowledge of related jobs	0.75	0.25	0.00
Good knowledge of this job and fair knowledge of related jobs	1.25	0.50	0.00
Fair knowledge of this job and no knowledge of related jobs	1.50	0.75	0.00
No knowledge of this job and detailed knowledge of related jobs	1.75	1.00	0.25
No knowledge of this job and no knowledge of related jobs	2.00	1.25	0.25

Appendix B

Questions for participants in the ERM experiments

(This is only a summary of the questionnaire)

Q1. Have you ever heard of ER modelling? Y/N

Q2. Have you ever carried out any ER modelling? Y/N

Q3. Have you ever worked as a full-time employee in business? Y/N

If "Yes", then state the number of years.....

Q4. Have you ever worked in the retail industry? Y/N

If "Yes", then state the number of years.....

Q5. Have you ever developed or been involved in developing business applications software (other than as a student)? Y/N

If "Yes", have you ever designed databases for these systems? Y/N

Q6. Have you ever developed or been involved in developing retail point-f-sale software? Y/N

Q7. What does ERD mean?.....

Q8. What is meant by:

entity.....

attribute.....

relationship.....

conceptual model.....

Q9. What is meant by:

cardinality.....

degree.....

optionality.....

dependency.....

Q10. If the parent entity is deleted in a database model, is it necessary to delete all its children? Y/N

Q11. What does orthogonal mean?.....

Q12. What is meant by 'reverse engineering' in the context of software development and database design?.....

Appendix C

Numeric Value Translation Table for Questionnaire Answers

Question	Answer	Value
Part I		
1	Y/N	1/0
2	Y/N	1/0
3	Y/N	1/0
No of years	x	x(Table G in Appendix A)
4	Y/N	1/0
No of years	y	y(Table J in Appendix A)
5a	Y/N	1/0
5b	Y/N	1/0
6	Y/N	1/0
7	No Answer/ Low/Medium/ High	[0,1,2,3]
8	No Answer/ Low/Medium/ High	[0,1,2,3]
9	No Answer/ Low/Medium/ High	[0,1,2,3]
10	Y/N	1/0
11	No Answer/ Low/Medium/ High	[0,1,2,3]
12	No Answer/ Low/Medium/ High	[0,1,2,3]
Part II		
Quality & Accuracy of ERM		[0,1,2,3]