Database Complexity Metrics

Coral Calero, Mario Piattini, Marcela Genero

Grupo ALARCOS – Departamento de Informática
Universidad de Castilla-La Mancha– SPAIN
email: {ccalero, mpiattin, mgenero}@inf-cr.uclm.es

Abstract

Metrics are useful mechanisms for improving the quality of software products and also for determining the best ways to help practitioners and researchers. Unfortunately, almost all the metrics put forward focus on program characteristics disregarding databases. However, databases are becoming more complex, and it is necessary to measure schemata complexity in order to understand, monitor, control, predict and improve database development and maintenance projects. In this paper, we will present different measures in order to measure the complexity that affects the maintainability of the relational, object-relational and active database schemas. However it is not enough to propose the metrics, a formal validation is also needed for knowing their mathematical characteristics. We will present the two main tendencies in metrics formal validation, axiomatic approaches and measurement theory. However, research into software measurement is needed from a theoretical but also from a practical point of view ([12]). For this reason, we will also present some of the experiments that we have developed for the different kinds of databases.

1. Introduction

Software engineers have been proposing large quantities of metrics for software products, processes and resources ([10], [23], [37]). Metrics are useful mechanisms for improving the quality of software products and also for determining the best ways to help practitioners and researchers ([26]). Unfortunately, almost all the metrics put forward focus on program characteristics (e.g. McCabe ([21]) cyclomatic number) disregarding databases ([34]). As far as databases are concerned, metrics have been used for comparing data models rather than the schemata itself. Several authors ([2], [16], [17], [18], [32], [33]) have compared the most well known models such as E/R, NIAM and relational using different metrics. Although we think this work is interesting, metrics for comparing schemata are needed mostly for practical purposes, like choosing between different design alternatives or giving designers limit values for certain characteristics (analogously to value 10 for Mc Cabe complexity of programs). Some recent proposals have been published for conceptual schemata ([20], [24], [28]) but for conventional databases, such as relational ones, nothing has been proposed, excepting normalization theory.

Databases are becoming more complex, and it is necessary to measure schemata complexity in order to understand, monitor, control, predict and improve database development and maintenance projects. In modern Information Systems (IS), the database has become a crucial component, so there is a need to propose and study some measures to assess its quality.

Database quality depends on several factors: functionality, reliability, usability, efficiency, maintainability and portability ([15]). Our focus is on maintainability because maintenance accounts for 60 to 90 percent of life cycle costs and it is considered the most important concern for modern IS departments ([11], [22], [29]).

The International Standard, ISO/IEC 9126, distinguishes five subcharacteristics for maintainability: analysability, changeability, stability, testability and compliance. Analysability, changeability and testability are in turn influenced by complexity ([19]). However, a general complexity measure is "*the impossible holy grail*" ([9]), i.e. it is impossible to get one value that captures all the complexity factors of a database . Henderson-Sellers ([14]) distinguishes three types of complexity: computational, psychological and representational, and for psychological complexity he considers three components: problem complexity, human cognitive factors and product complexity. The last one is our focus and for our purposes the product will be databases.

Our goal is to propose internal measures for databases, which can characterise their complexity helping to assess database maintainability (the external quality characteristic). In the next section we will present the framework followed to define and validate database metrics.

Section three summarizes the proposed metrics for relational, object-relational and active databases. In section four the formal validation of some of these metrics is described. Some empirical validations are presented in section five and conclusions and future work will be presented in sections six and seven respectively.

2. A Framework for Developing and Validating Database Metrics.

As we have said previously, our goal is to define metrics for controlling database maintainability. However, metrics definition must be done in a methodological way, so it is necessary to follow a number of steps to ensure the reliability of the proposed metrics. Figure 2 presents the method we apply for the metrics proposal.

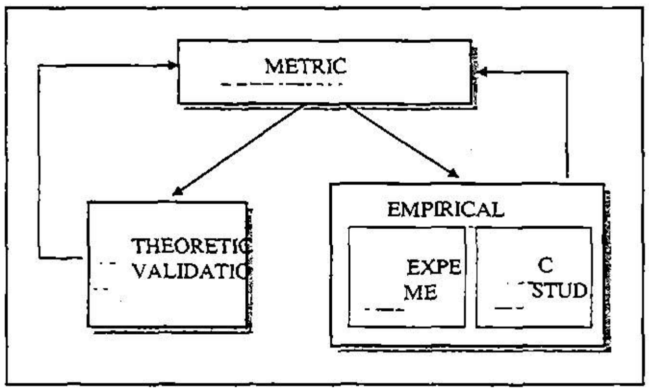

Figure 2. Steps followed in the definition and validation of the database metrics

In this figure we have three main activities:

- **Metrics definition.** The first step is the proposal of metrics. Although it looks simple, it is an important one in ensuring metrics are correctly defined. This definition is made taking into account the specific characteristics of the database we want to measure and the experience of database designers and administrators of these databases.
- **Theoretical validation.** The second step is the formal validation of the metrics. The formal validation helps us to know when and how to apply the metrics. There are two main tendencies in metrics validation: the frameworks based on axiomatic approaches and the ones based on the measurement theory. The goal of the first ones is merely definitional. The most well-known frameworks of this type are those proposed by [35], [3] and [25]. The measurement theory-based frameworks (such as [37] or [36]) specify a general framework in which measures should be defined. Measurement theory gives clear definitions of terminology, a sound basis of software measures, criteria for experimentation, conditions for validation of software measures, foundations of prediction models, empirical properties of software measures, and criteria for measurement scales.
- **Empirical validation.** The goal of this step is to prove the practical utility of the proposed metrics. Although there are various ways of performing this step, basically we can divide the empirical validation into experimentation and case studies. Experimentation is usually made using controlled experiments and the case studies usually work with real data. Both of them are necessary, the controlled experiments as a first approach and the case studies for backing up the results .

3. Proposed Metrics

In this section, we present the different metrics that we have proposed for relational, object-relational and active databases. For each kind of database, a brief summary of its main characteristics is given and an example using ANSI/ISO SQL:1999 code is used to illustrate the calculation of the proposed metrics.

Metrics for Relational Databases

Traditionally, the only indicator used to measure the "quality" of relational databases has been the normalization theory, with which [13] propose to obtain a normalization ratio. However, we think that normalization is not enough to measure complexity in relational databases, so we propose the following four metrics in addition to normalization ([5]):

Number of attributes (NA)

NA is the number of attributes in all the tables of the schema.

Depth Referential Tree (DRT)

DRT is defined as the length of the longest referential path in the database schema. Cycles are only considered once.

Number of Foreign Keys (NFK)

The NFK metric is defined as the number of foreign keys in the schema.

Cohesion of the schema (COS)

COS is defined as the sum of the square of the number of tables in each unrelated subgraph of the database schemata, that is:

$$COS = \sum_{i=1}^{|US|} NTUSi^2$$

|US| number of unrelated subgraphs
NTUS$_i$ number of tables in the related subgraph "i"

We apply the previous metrics to the following example (suppliers-and-parts database) taken from [6]:

```
CREATE TABLE S
( S#        S#,
 SNAME      NAME,
 STATUS     STATUS,
 CITY       CITY,
 PRIMARY KEY (S#));
CREATE TABLE P
( P#          P#
 PNAME        NAME,
 COLOR        COLOR,
 WEIGHT       WEIGHT,
 CITY         CITY,
 PRIMARY KEY (P#));
CREATE TABLE SP
( S#          S#,
 P#           P#,
 QTY          QTY,
 PRIMARY KEY (S#, P#),
 FOREIGN        KEY        (S#)
REFERENCES S,
 FOREIGN        KEY        (P#)
REFERENCES P);
```

This relational database schema can be represented as
a relational graph (see figure 3).

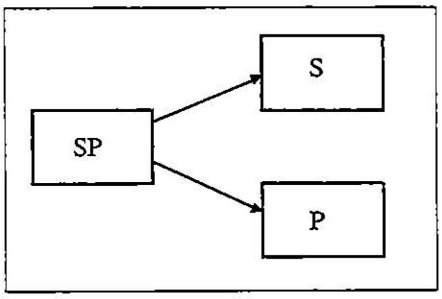

Figure 3. Relational graph for the example

In this schema the value of the metrics are: NA = 12, DRT = 1, NFK = 2, COS = 9.

Metrics for Object-Relational Databases

An object-relational database schema is composed of a number of tables related by referential integrity, which have columns that can be defined over simple or complex (user-defined) data types. We define the next metrics for object-relational databases ([4]):

Schema Size (SS)

We define the size of a system as the sum of the size of every table (TS) in the schema:

$$SS = \sum_{i=1}^{NT} TSi$$

The table size (TS) measures the size not only in terms of the simple columns (defined using simple domains) but also in terms of complex columns (defined using user-defined classes). Formally it can be defined as the sum of the total size of the simple columns (TSSC) and the total size of the complex columns (TSCC) in the table. TSSC is simply the number of simple columns in the table (considering that each simple domain has a size equal to one). TSCC is defined as the sum of complex columns size (CCS). The size of a complex column is no more than the size of the class hierarchy above the column and is defined as weighted by the number of complex columns which use the hierarchy. Finally, the size of a class hierarchy is defined as the sum of the size of each class on the hierarchy. For more details about the precise definition of this metric see [4].

Complexity of references between tables (DRT, NFK)

In object-relational databases, other characteristics of relational databases are preserved. Metrics related with the referential integrity, such as NFK and DRT proposed in the previous section, can also be used.

We can apply these metrics to the following example:

```
CREATE TYPE project AS (
name      CHAR(10),
          budget    FLOAT);
```

```
CREATE TYPE employee AS (
emp_num          INTEGER,
level            INTEGER,
salary_base      FLOAT,
proj             project)
method calc_salary()
      RETURNS DECIMAL(7,2));
```

Name data type	SMDT	SADT SAS + CAS	DTS
PROJECT	0	2 + 0	2
EMPLOYEE	1	3 + 2	6

```
CREATE TABLE works (
key              CHAR(10),
emp              employee,
adm_date DATE,
manager          employee);
```

Let us assume that all methods have a cyclomatic complexity equal to 1. In table 1, we present the value of the size of each data type.

Table 1. Size values of the data types

Then, the values for the other metrics are: SHC = 6, CCS = 3, TSCC = 6, TSSC = 2, TS = 8, SS = 8.

Metrics for Active Databases

When measuring active databases, we can make use of the notion of a triggering graph as defined in [1]. A triggering graph is a pair <S, L> where S represents the set of ECA rules, and L is a set of directed arcs where an arc is drawn from Si to Sj, if Si's action causes the happening of an event occurrence that participates in Sj's events. This notion of triggering graph is modified by [7] in two aspects. Firstly, arcs are weighted by the number of potential event occurrences produced by the triggering rule (i.e. Si) that could affect the rule triggered off (i.e. Sj's event). Secondly, the nodes S are extended with the set of transactions T. A transaction is an atomic set of (database) actions where any of these actions could correspond to an event triggering one or more rules. Therefore, T nodes will have outgoing links but never incoming links, as we assume that a transaction can never be triggered from within a rule's action or another transaction.

The active database could be characterised by the following triggering graph measures ([8]):

- NA, the minimum number of anchors required to encompass the whole set of potential causes of Si. An anchor is a transaction node of the triggering graph, which has a link (either directly or transitively) with at least one cause of Si.
- D, the distance. This measure corresponds to the length of the longest path that connects Si with any of its anchors.
- TP, the triggering potential. Given a triggering graph < S, L >, and a node of the graph, rule Si, the number of causes of Si, is the sum of weights of the incoming arcs arriving at Si. The triggering potential for a rule R is the quotient between the number of potential causes of Si, and Si's event cardinality.

```
CREATE TRIGGER ONE
AFTER DELETE ON TABLE3
FOR EACH ROW
WHEN (OLD.NUMBER=3)
BEGIN
  DELETE FROM TABLE4 WHERE TABLE4.S#=:TABLE3.J#:
END ONE;
CREATE TRIGGER TWO
AFTER DELETE ON TABLE4
FOR EACH ROW
WHEN (OLD.NAME='SMITH')
BEGIN
  DELETE FROM TABLE5 WHERE TABLE4.S#=:OLD.S#:
END TWO;
```

For trigger ONE, NA = 1, TP = 1 and D = 1, for trigger TWO, NA = 1, TP = 1, and D = 2.

4. Metrics Formal Validation

There are two main tendencies in metrics validation: the frameworks based on axiomatic approaches and the ones based on the measurement theory. The goal of the first ones is merely definitional. On this kind of formal framework, a set of formal properties is defined for a given software attribute and it is possible to use this property set for classifying the proposed measures.

The most well-known frameworks of this type are those proposed by [35], [3], and [25]. The main goal of axiomatisation in software metrics research is the clarification of concepts to ensure that new metrics are in some sense valid. The measurement theory-based frameworks (such as [37] or [36]) specify a general framework in which measures should be defined. Measurement theory gives clear definitions of terminology, a sound basis of software measures, criteria for experimentation, conditions for validation of software measures, foundations of prediction models, empirical properties of software measures, and criteria for measurement scales. The discussion of scale types is important for statistical operations.

In this section, we will present the results of the formal verification of the presented metrics with both validation techniques [3] as an example of axiomatic approach and [37] as an example of using the measurement theory.

In table 4, we present the results obtained for all the presented metrics on both formal frameworks.

		BRIAND ET AL(1996)	ZUSE(1998)
RELVENAL	NFK	COMPLEXITY	ABOVE THE ORDINAL
	DRT	LENGTH	ABOVE THE ORDINAL
	NA	SIZE	ABOVE THE ORDINAL
	COS	SIZE	RATIO
	SS	SIZE	ABOVE THE ORDINAL
	TS	SIZE	ABOVE THE ORDINAL
	NA	COMPLEXITY	ABOVE THE ORDINAL
	TP	NOT CLASSIFIABLE	ABOVE THE ORDINAL
	D	LENGTH	ABOVE THE ORDINAL

Table 4. Summary of metrics formal validation

With the axiomatic approach results we can know, for example for relational databases that we need some metrics for capturing cohesion and coupling and covering all the characteristics defined by the framework. From the measurement theory results we can know what kind of operations it is possible to make with the defined metrics, the statistics that it is possible to apply to them etc.

5. Metrics Empirical Validation

In the past, empirical validation has been an informal process relying on the credibility of the proposer. Often , when a measure was identified theoretically as an effective measure of complexity, practitioners and researchers began to use the metric without questioning its validity. Today, many researchers and practitioners assume that validation of a measure (from a theoretical point of view) is not sufficient for widespread acceptance. They expect the empirical validation to demonstrate that the measure itself can be validated. Useful results of an experimentation depend on careful, rigorous and complete experimental design. A claim that a measure is valid because it is a good predictor of some interesting attribute can be justified only by formulating a hypothesis about the relationship and then testing the hypothesis ([10]).

In the rest of this section, we summarize different experiments that we have done with some of the metrics discussed in this chapter. All of these initial experiments require further experimentation in order to validate the findings. However, these results can be useful as a starting point for future research.

A complete description of the experiments can be found in [5] for relational database metrics, in [27] for object-relational ones and in [8] for active ones.

Relational experiment

Our objective was to demonstrate that the metrics related with referential integrity (DRT and NFK) can be used for measuring the complexity of the relational database schema, which influences in the relational database understandability. The participants of this study were computer science students at the University of Castilla-La Mancha (Spain), who were enrolled in a two-semester databases course. Based on the results of this experiment, we concluded that the number of foreign keys in a relational database schema is a more solid indicator of its understandability than the length of the referential tree. This metric is not relevant by itself, but can modulate the effect of the number of foreign keys in a database system

Object-relational experiment

Five object-relational databases were used in this experiment with an average of 10 relations per database. Five subjects participated in the experiment. All of them were experienced in both relational databases and object-oriented programming. To analyze the usefulness of the metrics, we used two techniques: C4.5 ([30]), a machine learning algorithms, and RoC ([31]), a robust Bayesian classifier. In conclusion, both the techniques discover that the table size is a good indicator for the understandability of a table. The depth of the referential tree is also presented as an indicator by C4.5 but not clearly by RoC. The number of foreign keys does not seem to have a real impact on the understandability of a table.

Active experiment

Our objective was to assess the influence of D and TP in rule interaction understandability. However, such understanding could be influenced by how the reasoning is conducted. As rules can be seen as cause-and-effect links, two questions can be posed by the user: "what effects can a rule produce (forward reasoning)?" or "how can an effect be produced (backward reasoning)?". The participants of the experiment were final-year computer science students at the University of the Basque Country, who were enrolled in an advance database course . The students were already familiar with relational database, and some laboratories were previously conducted on the definition of triggers. For the forward experiment, we concluded that the triggering potential in a database schema is a solid indicator of its understandability, and that the distance is not relevant by itself and cannot modulate the effect of the triggering potential. For the backward experiment, we concluded that both metrics are solid indicators of its understandability.

All the experiments described need to be replicated in order to obtain more consistent results. However, controlled experiments made in a laboratory are useful as a starting point but present some problems such a the large number of variables that can cause differences. Therefore, it is convenient to also run case studies working with real data.

6. Conclusions and Future Work

Databases are becoming more complex, and it is necessary to measure schemata complexity in order to understand, monitor, control, predict and improve database development and maintenance projects. Database metrics could help designers, choosing between alternative semantically equivalent schemata, to select the most maintainable one and understand their contribution to the overall IS maintainability.

We have put forward different measures (for internal attributes) in order to measure the complexity that affects the maintainability (an external attribute) of the relational, object-relational and active database schemas.

However it is not enough to propose the metrics, a formal validation is also needed for knowing their mathematical characteristics. We have presented the two main tendencies in metrics formal validation, axiomatic approaches and measurement theory. Although the information obtained from both techniques is different, the final objective is the same, to obtain objective mathematical information of the metrics we are working on.

However, as we have indicated previously, research into software measurement is needed also from a practical point of view. We have presented some of the experiments that we have developed for the different kinds of databases. Nevertheless controlled experiments have problems (like the large number of variables that causes differences) and limits (they do not scale up, are done in a class in training situations, are made in vitro and face a variety of threats of validity). Therefore it is convenient to run multiple studies, mixing controlled experiments and case studies. For these reasons, a more in depth empirical evaluation is under way in collaboration with industrial and public organizations in "real" situations.

ACKNOWLEDGEMENT

This research is part of the MANTICA project, partially supported by the CICYT and the European Union (CICYT-1FD97-0168) and by the CALIDAT project carried out by Cronos Ibérica (supported by the Consejería de Educación de la Comunidad de Madrid, Nr. 09/0013/1999)

REFERENCES

1. Aiken, A., Hellerstein, J.M. and Widom, J. (1995). Static analysis techniques for predicting the behaviour of active database rules. ACM Transactions on Databases, 20 (1), 3-41.
2. Batra, D., Hoffer, J.A. and Bostrom, R.P. (1990). A comparison of user performance between the relational and the extended entity relationship models in the discovery phase of database design. CACM, 33 (2), 126-139.
3. Briand, L.C., Morasca, S. And Basili, V. (1996). Property-based software engineering measurement. IEEE Transactions on Software Engineering, Vol.22(1), pp. 68-85.
4. Calero, C., Piattini, M., Ruiz, F. and Polo, M. (1999) Validation of metrics for Object-Relational Databases, International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (ECOOP99), Lisbon (Portugal), 14-18 June
5. Calero, C., Piattini, M., Genero, M., Serrano, M. and Caballero, I. (2000). Metrics for Relational Databases Maintainability, UKAIS2000, Cardiff, UK.
6. Date, C.J. (1995). An Introduction to Database Systems. 6th. Addison-Wesley, Reading, Massachusetts.
7. Díaz, O. and Piattini, M. (1999). Metrics for active databases maintainability. CAISE'99. Heidelberg, June 16-18.
8. Díaz, O., Piattini. M. y Calero, C. (2001). Measuring active databases maintainability.

Accepted for publication in Information Systems Journal

9. Fenton, N. (1994). Software Measurement: A Necessary Scientific Basis. IEEE Transactions on Software Engineering, 20(3), 199-206.
10. Fenton, N. and Pfleeger, S. L. (1997). Software Metrics: A Rigorous Approach 2nd. edition. London, Chapman & Hall.
11. Frazer, A. (1992). Reverse engineering-hype, hope or here?. In: P.A.V. Hall, Software Reuse and Reverse Engineering in Practice. Chapman & Hall.
12. Glass, R. (1996). The Relationship Between Theory and Practice in Software Engineering. IEEE Software, November, Vol.39 (11). pp 11-13.
13. Gray R.H.M., Carey B.N., McGlynn N.A., Pengelly A.D., (1991), Design metrics for database systems. BT Technology J, Vol 9, 4 Oct. pp. 69-79
14. Henderson-Sellers, B. (1996). Object-Oriented Metrics - Measures of Complexity. Prentice-Hall, Upper Saddle River, New Jersey.
15. ISO, (1994). Software Product Evaluation-Quality Characteristics and Guidelines for their Use. ISO/IEC Standard 9126, Geneva.
16. Jarvenpaa, S. and Machesky, J. (1986). End user learning behavior in data analysis and data modeling tools. Proc. of the 7th Int. Conf. on Information Systems, San Diego, 152-167.
17. Juhn, S. and Naumann, J. (1985). The effectiveness of data representation characteristics on user validation. Proc. of the 6th Int. Conf. on Information Systems, Indianapolis, 212-226.
18. Kim, Y.-G. and March, S.T. (1995). Comparing Data Modeling Formalisms. CACM 38(6), 103-115.
19. Li, H.F. and Cheng, W.K. (1987). An empirical study of software metrics. IEEE Trans. on Software Engineering, 13 (6), 679-708.
20. MacDonell, S.G., Shepperd, M.J. and Sallis, P.J. (1997). Metrics for Database Systems: An Empirical Study. Proc. Fourth International Software Metrics Symposium – Metrics'97, Albuquerque. IEEE Computer Society, 99-107.
21. McCabe, T.J. (1976). A complexity measure. IEEE Trans. Software Engineering 2(5), 308-320.
22. McClure, C. (1992) The Three R's of Software Automation: Re-engineering, Repository, Reusability. Englewood Cliffs: Prentice-Hall.
23. Melton, A. (ed.) (1996). Software Measurement. London, International Thomson Computer Press.
24. Moody, D. L. (1998). Metrics for Evaluating the Quality of Entity Relationship Models. Proc. of the Seventeenth International Conference on Conceptual Modelling (ER'98), Singapore.
25. Morasca, S. and Briand, L.C. (1997). Towards a Theoretical Framework for measuring software attributes. Proceeding of the Fourth International,Software Metrics Symposium, 119-126.
26. Pfleeger, S. L. (1997). Assessing Software Measurement. IEEE Software. March/April, 25-26.
27. Piattini, M., Calero, C., Sahraoui, H. and Lounis, H. (2000) An empirical study with object-relational database metrics, ECOOP'2000, Cannes, 13 June.
28. Piattini, M., Genero, M., Calero, C., Polo, M. and Ruiz, F. (2001). Metrics for Managing Quality in Information Modelling. In: "Information Modeling in the New Millennium" M. Rossi and K. Siau (eds.), Idea Group Publishing, USA.
29. Pigoski, T.M. (1997). Practical Software Maintenance. Wiley Computer Publishing. New York, USA.
30. Quinlan, J.R., C4.5: Programs for Machine Learning, (1993), Morgan Kaufmann Publishers.
31. Ramoni, M. and Sebastiani, P. Bayesian methods for intelligent data analysis. In M. Berthold and D.J. Hand, editors, An Introduction to Intelligent Data Analysis, (New York, 1999). Springer.
32. Rossi, M. and Brinkkemper, S. (1996). Complexity Metrics for Systems Development Methods and Techniques. Information Systems 21(2), 209-227.
33. Shoval, P. and Even-Chaime, M. (1987). Database schema design: An experimental comparison between normalization and information analysis. Database 18(3), 30-39.
34. Sneed, H.M. and Foshag, O. (1998). Measuring Legacy Database Structures. Proc of The European Software Measurement Conference FESMA 98, Antwerp, May 6-8, Coombes, Van Huysduynen and Peeters (eds.), 199-211.
35. Weyuker, E.J. (1988). Evaluating software complexity measures. IEEE Transactions on Software Engineering Vol.14(9). pp. 1357-1365.
36. Whitmire, S.A. (1997), Object Oriented Design Measurement, Ed. Wiley.
37. Zuse, H. (1998). A Framework of Software Measurement. Berlin, Walter de Gruyter