PROCEEDINGS

INTERNATIONAL SYMPOSIUM ON DATABASE TECHNOLOGY & SOFTWARE ENGINEERING, WEB AND COOPERATIVE SYSTEMS

in conjunction with

INTERSYMP 2000

The 12th International Conference on System Research
Informatics and Cybernetics

August 1 to August 4, 2000, Baden-Baden
Germany

Edited by
G.E. Lasker, University of Windsor, Canada
W. Gerhardt, Delft University of Technology, The Netherlands

Published by

THE INTERNATIONAL INSTITUTE FOR ADVANCED STUDIES
IN SYSTEMS RESEARCH AND CYBERNETICS

PROCEEDINGS

INTERNATIONAL SYMPOSIUM ON DATABASE TECHNOLOGY & SOFTWARE ENGINEERING, WEB AND COOPERATIVE SYSTEMS

Edited by
G.E. Lasker, University of Windsor, Canada
W. Gerhardt, Delft University of Technology, The Netherlands

Sponsored by
Bitbybit - Information Systems Delft, The Netherlands

Papers are reproduced here from camera-ready copies prepared by the authors.
Canadian Cataloguing in Publication Data

Main entry under title:
International Symposium on Database Technology & Software Engineering, WEB and Cooperation Systems

Proceedings of the Symposium in conjunction with INTERSYMP'2000,
12th International Conference held in Baden-Baden, Germany, Aug. 1-4, 2000.

1. Databases - Congresses. 2. Software Engineering-Congresses.
2. WEB Applications-Congresses.
I. Lasker, G.E. (George Eric), 1935.
II. Gerhardt, W. (Waltraud), 1949.
III. International Conference on System Research, Informatics and Cybernetics (12th, 2000: Baden-Baden, Germany)

ISBN 0921836-57-0

Publisher: The International Institut for Advanced Studies in System Research and Cybernetics
c/o Dr. George E. Lasker
School of Computer Science
University of Windsor
Windsor, Ontario
Canada N9B3P4

PREFACE

The International Symposium on Database Technology & Software Engineering, WEB and Cooperative Systems takes place in conjunction with INTERSYMP 2000, the 12th International Conference on System Research, Informatics and Cybernetics.

The symposium provides a unique international forum for discussions about possible synergistic effects and altering impacts of Database Technology & Software Engineering with the purpose to build flexible, scalable and well maintainable large, distributed and cooperative information systems, offering information definition and manipulation functionality through the Web. We formulate as a hypothesis for the contributions of the participants, that Database Technology and Software Engineering have to learn from each other, or more strongly said, have to "reuse" each other to approach the aforementioned challenging systems.

We invited researchers and practitioners to exchange opinions and provide ideas on how to make a step forward in designing, realizing and maintaining the aforementioned systems from an overall systems' point of view. We included thinking about directions of Database and Software Engineering research. Our invitation is based on our strong belief that system thinking would help to design and realize information systems with a complex functionality, however with a less complex architecture. One step in the direction of system thinking is to consider an information system not exclusively from a descriptive point of view or from a procedural one.

We like to thank all collegues for submission of papers and for joining us at the symposium.
Thanks go to the organizers of the conference, especially to Mr. Prof. G. Lasker from the International Institute for Advanced Studies in Systems Research and Cybernetics for offering the possibility of the symposium and for his great support in preparation and realization of the symposium. A lot of thanks to Bitbybit-Information Systems for sponsoring the proceedings, and Mrs. C. van Driel and Mr. M. Abolhassani from Delft University of Technology for their kind support in the preparation.

Baden-Baden, 2nd of August 2000

Waltraud Gerhardt., Jan van Katwijk, Mike Jenks

Delft University of Technology, The Netherlands and Oxford Brookes University, UK

Table of contents

APPLYING SOFTWARE METRICS TO DATABASES[1]

Coral Calero, Mario Piattini and Marcela Genero
Escuela Superior de Informática. Universidad de Castilla-La Mancha
Ronda Calatrava, 5 13071 Ciudad Real. Spain
Phone :+34 926 295300 (ext. 3715). Fax. + 34 926 29 53 54
E-mail: {ccalero, mpiattin, megenero}@inf-cr.uclm.es

Software measurement is widely recognised as an effective means to understand, monitor, control, predict and improve software development and maintenance projects; and also for determining the best ways to help practitioners and researchers. Software engineers have been putting forward huge quantities of metrics for software products, processes and resources. Unfortunately, almost all the metrics proposed until now, have focused on programme characteristics disregarding databases. As far as databases are concerned, metrics have been used for comparing data models (E/R, NIAM, ...) rather than the schemata itself. Although we think this work is interesting, metrics for comparing schemata are needed most for practical purposes, like choosing between different design alternatives or giving designers limit values for certain characteristics.

This neglect could be explained as databases have developed until recently just a secondary role with minor contribution to the complexity of the overall system. Nowadays, databases are introduced in most of the important information systems, becoming their essential core. Because of this, we think it is very important to "re-use" software engineering metrics expertise in order to develop metrics for databases. We have proposed metrics for conceptual schemata, and for relational and object-relational databases.

1. Introduction

Nowadays, in a global and increasingly competitive market, quality is a critical success factor for all economical and organisational aspects, and especially in Information Systems (IS). Developing and selecting high quality software applications is fundamental. It is important that the software applications can be evaluated for every relevant quality characteristic using validated metrics.

Software engineers have been putting forward huge quantities of metrics for software products, processes and resources (Melton, 1996; Fenton and Pfleeger, 1997). Unfortunately, almost all the metrics proposed are focused on programme, disregarding data-related quality (Sneed and Foshag, 1998). This neglect could be explained as databases have developed until recently just a secondary role with minor contribution to the quality of the overall system. Nowadays, databases are introduced in most of the important IS, becoming their essential core.

In this paper we propose different metrics to analyse the quality of different kinds of databases. Following the ISO/IEC 9126 (ISO, 1994) quality model, several characteristics can be identified in software quality: functionality, reliability, usability, efficiency, maintainability and portability. Taking into account that maintenance arrange between 60 and 90 percent of life cycle costs (Card and Glass, 1990; Pigoski, 1997), we focus our work on maintainability. ISO/IEC 9126 distinguishes five subcharacteristics for maintainability: analysability, changeability, stability, testability and compliance (see figure 1). Analysability, changeability and testability are in turn influenced by complexity (Li and Chen, 1987). However, a general complexity measure is "*the impossible holy grail*" (Fenton, 1994). Henderson-Sellers (1996) 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 (Henderson-Sellers, 1996). The last one is our focus.

---

[1] 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)

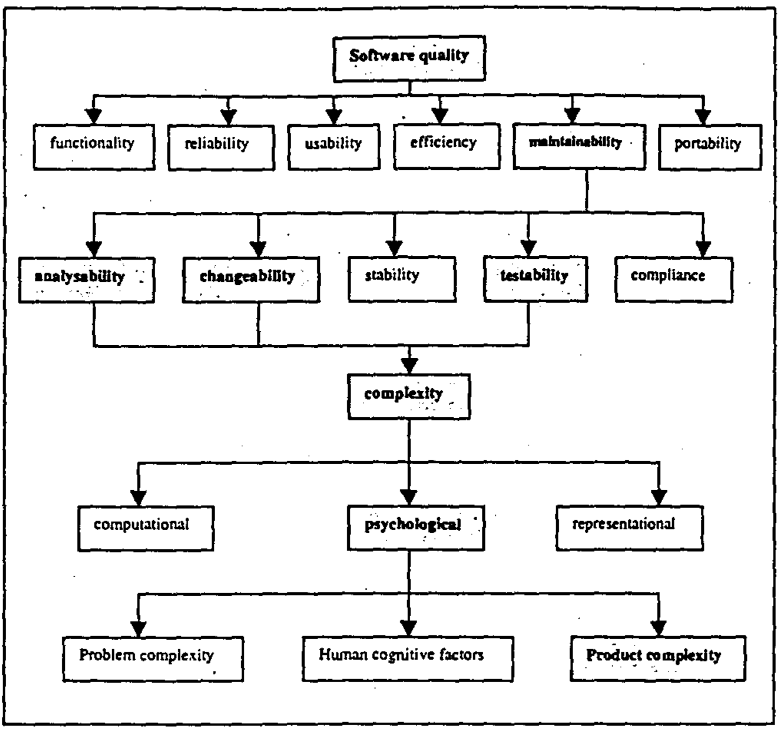


Figure 1. Relation between products complexity metrics and software quality

So, the metrics we define are for measure internal attributes of different kinds of databases, and characterise their complexity, which can help to assess the database maintainability (the external attribute). In section 2, we present the metrics for conceptual modelling, third section shows the metrics for relational databases and metrics for object-relational come on section four. Finally, section 5 presents the conclusions and future work.

2. Metrics for conceptual modelling

We propose five metrics (ERSS, DA, RR, EDIT, SD) for conceptual data models, in particular for the most widely known Entity Relationship (E/R) schemas. Since the aim is that of simplifying the E/R model, the objective will be to minimise the value of these metrics.

ER Schema Size metric (ERSS). We define ER Schema Size metric as the sum of the size of each entity (ES) plus the sum of the size of each relationship (RS) within the E/R schema. The Entity Size (ES) is the sum of the size of its simple attributes (SA) which are considered to have a size equal to 1 (so that SA will be the same as the number of simple attributes N^{SA}) plus the size of its composite attributes (CAS). A composite attribute is an attribute composed of a set of simple attributes, so CAS is the number of attributes in this set. The Relationship Size (RS) is the sum of the size of the relationships without attributes (RWAS) plus the size of the relationships with attributes (RAS). The Relationship Without Attributes Size (RWAS) is equal to the number of relationships without attributes (N^{RWA}). The Relationship With Attributes Size (RAS) is the sum of the size of its simple attributes (RSA), which are considered to have a size equal to 1 (so that RSA will be the same as the number of simple attributes N^{RSA}) plus the sum of the size of its composite attributes (RCAS). As we mentioned above, a composite attribute is an attribute composed of a set of simple attributes, so that RCAS is the number of attributes in this set. When we calculate the size of the relationships without attributes (RWAS), we must consider the IS_A relationships. In this case, we consider one relationship for each pair child-parent within the IS_A relationship. Then ERSS metric is calculated thus:

$$ERSS = N^{SA} + \sum_{i=1}^{N^{CA}} CAS + N^{RWA} + N^{RSA} + \sum_{i=1}^{N^{RCA}} RCAS$$

Derived Attribute metric (DA). An E/R conceptual schema is minimal when every aspect of the requirements appears once in the schema, i.e. an E/R conceptual schema is minimal if it does not have any redundancies. One of the sources of redundancies in the E/R schemas is the existence of derived attributes.

An attribute is derived when its value can be calculated or deduced from the values of other attributes. We define Derived Attributes metric (DA) as the number of derived attributes existing in the E/R schema.

Redundant Relationship metric (RR). Another source of redundancy in an E/R schema is the existence of redundant relationships. We define Redundant Relationship metric (RR) as the number of relationships that are redundant in the E/R schema.

Entity Depth Inheritance Tree metric (EDIT). Entity Depth Inheritance Tree (EDIT) metric is based on DIT metric proposed by Chidamber and Kemerer (1994). When an E/R schema contains IS_A relationships, that represent the concept of simple or multiple generalisation/specialisation, we may have an inheritance hierarchy of several levels. For this characteristic we consider the following metric: *EDIT* = Maximum depth in the inheritance tree.

Schema Density metric (SD). We *define SD metric thus:*

$SD = \sum_{i=1}^{N^E} a_i^2$

a_i is the number of entities which can be directly reached from the entity "i" through relationships.

N^E is the number of entities in the E/R schema.

SD metric is important: from the point of view of the person who must make a reengineering work and must understand the information contained in it; from the program developer's point of view of, since a great number of relationships will imply more programming effort for controlling the information integrity; also for the database and programs maintainer, as far as a little change in one of the future entities may have some influence on some others and, surely, on a set of programs which will be likely to be changed. Consequently, this is one of the important characteristics we must evaluate in order to know the complexity of an E/R schema.

Example. Let us apply the proposed metrics to the E/R schema shown in figure 2:

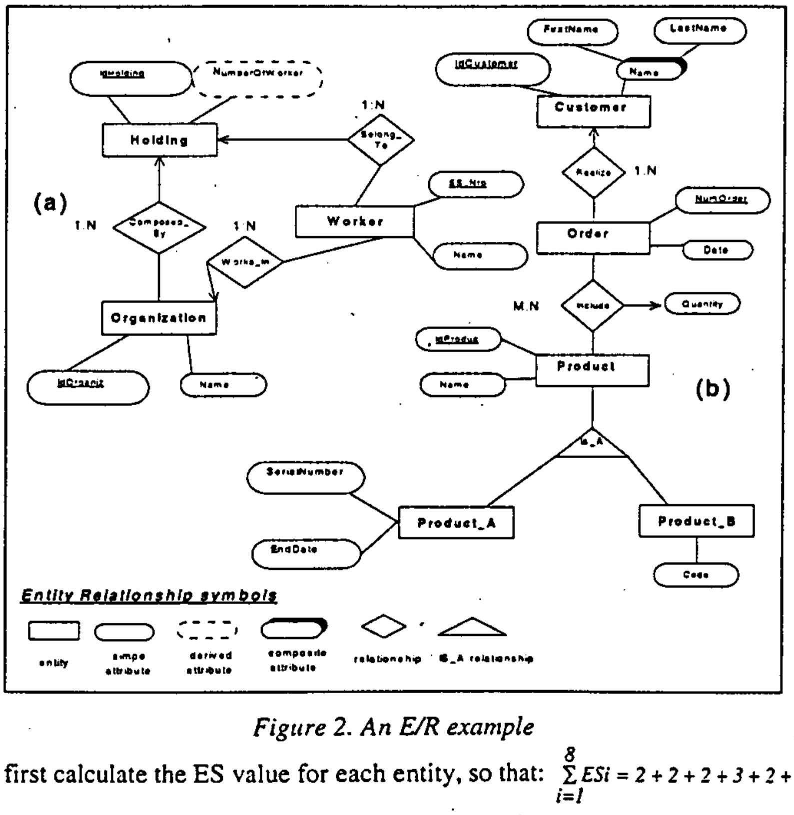


Figure 2. An E/R example

ERSS metric. We first calculate the ES value for each entity, so that: $\sum_{i=1}^{8} ESi = 2+2+2+3+2+2+2+1 = 16$

Now, we calculate the RS value for each relationship. We have to bear in mind the relationships without attributes and the relationships with attributes. There are six relationships without attributes *(Composed_Of, Works_In, Belong_To, Realize,* and two *IS_A relationships*), and there is one relationship with attributes

(*Include*), this relationship has one simple attribute so that: $\sum_{i=1}^{7} RSi = 6 + 1 = 7$, this number is equal to the number of relationship because none of the relationships have composite attributes.

So that $ERSS = \sum_{i=1}^{8} ESi + \sum_{i=1}^{7} RSi = 16 + 7 = 23$

DA metric. There is only one derived attribute, the *NumberOfWorker* attribute of the *Holding* entity, so we calculate DA = 1 .

The *Name* attribute (composite attribute) of the *Customer* entity is composed of two attributes, *FirstName* and *LastName,* so when we consider this attribute we add two units. We also include the *Quantity* attribute of the *Include* relationship.

RR metric. The *Belong_To* relationship is redundant, because if one *Organisation* belongs to only one *Holding*, it is easy to get "In which holding works one worker" following the relationships *Works_In* and *Composed_By*, so we calculate RR = 1.

For this metric we consider one relationship for each pair child-parent within the IS_A relationship (one for the relationship between the *Product* and *Product_A* entities, and another for the relationship between the *Product* and *Product_B* entities).

EDIT metric. There is one IS_A relationship, with only one level, between the *Product*, *Product_A* and *Product_B* entities, so that EDIT = 1.

SD metric. The E/R schema has two unrelated subgraphs, so we calculate SD metric thus : SD = SD (a) + SD (b), being SD (a) the Schema Density of the subgraph (a), and SD (b) the Schema Density of the subgraph (b).
SD (a) = $2^2 + 2^2 + 2^2 = 4 + 4 + 4 = 12$. SD (b) = $1^2 + 2^2 + 3^2 + 1^2 + 1^2 = 1 + 4 + 9 + 1 + 1 = 16$.
SD = 12 + 16 = 28

3. Metrics for relational databases

The relational model proposed by Dr. Codd in the late sixties (Codd, 1970), currently dominates the database market. In spite of their diffusion, the only indicator used to measure the quality of relational database has been the normalisation theory, upon which Gray et al. (1991) propose to obtain a normalisation ratio. But normalisation is not enough to characterise database quality. Date (1995) defines a relational database management system as "a system in which, at minimum :

- The data is perceived by the user as tables (and nothing but tables); and
- The operators at the user's disposal are operators that generate new tables from old, and those operators include at least SELECT, PROJECT, JOIN" .

Related with the foreign key concept, the relational model includes the referential integrity rule: the database must not contain any unmatched foreign key values (Date, 1995).

Taking into account the characteristics of the relational databases (tables and referential integrity) we propose the following four metrics: NA, DRT, RD and COS.

NA metric. Number of attributes (NA) is the number of attributes in all the tables of the schema.

DRT metric. Depth Referential Tree (DRT) is a referential integrity related metric and is defined as the length of the longest referential path in the database schema. Every arc of the path is a referential integrity relation between two tables. The value of the metric is given by the number of arcs on the longest path. Cycles are only considered once

RD metric. Referential Degree (RD) metric is also a referential integrity related metric and is defined as the number of foreign keys in the schema.

COS metric. Cohesion of the schema (COS) is defined as the sum of the square of the number of tables per unrelated subgraph in the database:

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

|US| number of unrelated subgraphs
$NTUS_i$ number of tables in the unrelated subgraph "i"

Example. Applying the previous metrics to the following example taken from Elmasri and Navathe (1999):

```
CREATE TABLE EMPLOYEE
            (
            NAMEP                   VARCHAR(15)          NOT NULL.
            INIC                    CHAR.
            SURNAME   VARCHAR(15)     NOT NULL.
            NSS                     CHAR(9)    NOT NULL.
            DATEN                   DATE.
            ADDRESS   VARCHAR(30).
            SEX                     CHAR.
            SALARY    DECIMAL(10.2).
            NSSSUPER  CHAR(9).
            ND                      INT        NOT NULL.
  CONSTRAINT CLPEMP
            PRIMARY KEY (NSS).
  CONSTRAINT CLESUPEREMP
            FOREIGN KEY (NSSSUPER) REFERENCES EMPLOYEE(NSS)
                      ON DELETE SET NULL ON UPDATE CASCADE.
  CONSTRAINT CLEDEPTOEMP
            FOREIGN KEY (ND) REFERENCES DEPARTAMENT (NUMBERD)
                      ON DELETE SET DEFAULT ON UPDATE
CASCADE);


CREATE TABLE DEPARTAMENT
            (
            NAMED     VARCHAR(15)     NOT NULL.
            NUMBERD   INT             NOT NULL.
            NSSGTE    CHAR(9)    NOT NULL.
            DATEINICGTE DATE.
  CONSTRAINT CLPDEPTO
            PRIMARY KEY (NUMBERD).
  CONSTRAINT CLSDEPTO
            UNIQUE(NOMBRED).
  CONSTRAINT CLEGTEDEPTO
            FOREIGN KEY (NSSGTE) REFERENCES EMPLOYEE (NSS)
                      ON DELETE SET DEFAULT ON UPDATE
CASCADE);
```

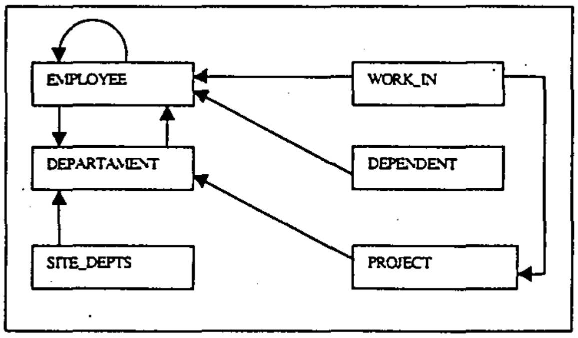
```
CREATE TABLE SITE_DEPTS
            (
            NUMBERD   INT                  NOT NULL.
            SITED                   VARCHAR(15)          NOT NULL.
  PRIMARY KEY (NUMBERD. SITED).
  FOREIGN KEY (NUMBERD) REFERENCES DEPARTAMENT (NUMBERD)
            ON DELETE CASCADE ON UPDATE CASCADE);


CREATE TABLE PROJECT
            (
            NAMEPR    VARCHAR(15)     NOT NULL.
            NUMBERPR INT              NOT NULL.
            SITEPR                  VARCHAR(15).
            NUMD                    INT        NOT NULL.
  PRIMARY KEY (NUMBERP).
  UNIQUE (NAMEP).
  FOREIGN KEY (NUMD) REFERENCES DEPARTAMENT (NUMBERD));


CREATE TABLE WORK_IN
            (
            NSSE                    CHAR(9)    NOT NULL.
            NUMP                    INT        NOT NULL.
            HOURS                   DECIMAL(3.1)         NOT NULL.
  PRIMARY KEY (NSSE. NUMP).
  FOREIGN KEY (NSSE) REFERENCES EMPLOYEE (NSS).
  FOREIGN KEY (NUMP) REFERENCES PROJECT (NUMBERP));


CREATE TABLE DEPENDENT
            (
            NSSE                    CHAR(9)    NOT NULL
            NAME_DEPEND             VARCHAR(15)          NOT NULL.
            SEX                     CHAR.
            DATEAN    DATE.
            RELATION  VARCHAR(8).
  PRIMARY KEY (NSSE. NAME_DEPEND).
  FOREIGN KEY (NSSE) REFERENCES EMPLOYEE (NSS));
```

We can obtain the relational graph and calculate the metrics values:



	NA	DRT	RD	COS
EMPLOYEE	10	3	2	
DEPARTAMENT	4	3	1	
SITE_DEPTS	2	4	1	
PROJECT	4	4	1	
WORK_IN	3	5	2	
DEPENDENT	5	4	1	
SCHEMA	28	5	8	36

4. Metrics for object-relational databases

An object-relational database schema is composed by a number of related tables. Every table have columns that can be defined as a simple data or as a complex data. The type of a simple data may be one of the classic data types as integer, number or character. A complex data is defined above a class (or a UDT, user defined type), which can be related with other classes (types) by generalisation or inheritance associations.

Size of a table. We define the table size (TS) as the sum of the total size of the simple columns (TSSC) and the total size of the complex columns (TSCC) in the table:

$$TS_i = TSSC + TSCC$$

We consider that all simple columns have a size equal to one, then the TSSC metric is equal to the number of simple columns in the table (NSA).

$$TSSC = NSA$$

And the TSCC is defined as the sum of each complex column size (CCS):

$$TSCC = \sum_{i=1}^{NCC} CCS_i$$

Being NCC the number of complex columns in the table.
The value for CCS is obtained as:

$$CCS = \frac{SHC}{NCU}$$

Being SHC the "size of the hierarchy (formed by the class and its antecesors)" above which the column is defined and NCU is the number of columns defined above this hierarchy. This expression reflects the fact that the effort to understand two complex columns decreases if both are defined over the same class.
The SHC may be defined as the sum of each class size in the hierarchy (SC):

$$SHC = \sum_{i=1}^{NCH} SC_i$$

being NCH the number of classes in the hierarchy.
The size of a class is defined as:

$$SC = \frac{SAC + SMC}{NHC}$$

being SAC the sum of the size attributes of the class, SMC the size methods of the class and NHC the number of hierarchies to which the class pertain.
The attributes of a class may also be simple or complex, then the SAC is defined as the sum of the simple attributes size (SAS, that have size equal to one like simple attributes) and the complex attributes size (CAS) in the class.

$$SAC = SAS + CAS$$

And the SMC is calculated with the version of the cyclomatic complexity of McCabe given by Li and Henry (1993):

$$SMC = \sum_{i=1}^{NMC} V_i(G)$$

being NMC the number of methods in the class

Complexity of references between tables. In object-relational databases, some characteristics of relational databases are preserved, for example referential integrity. So, for this kind of databases we can use the metrics related with referential integrity proposed for relational databases DRT and RD.

Example. Applying the metrics to the example showed in figure 3, we obtain the following values for the metrics: SC(class_person)=3,
SC(class_projects)=3,
SC(class_emp)=7
and SHC=13,CCS= 6.5,
TSCC=13,
TSSC=NSA=2
and finally TS=15.

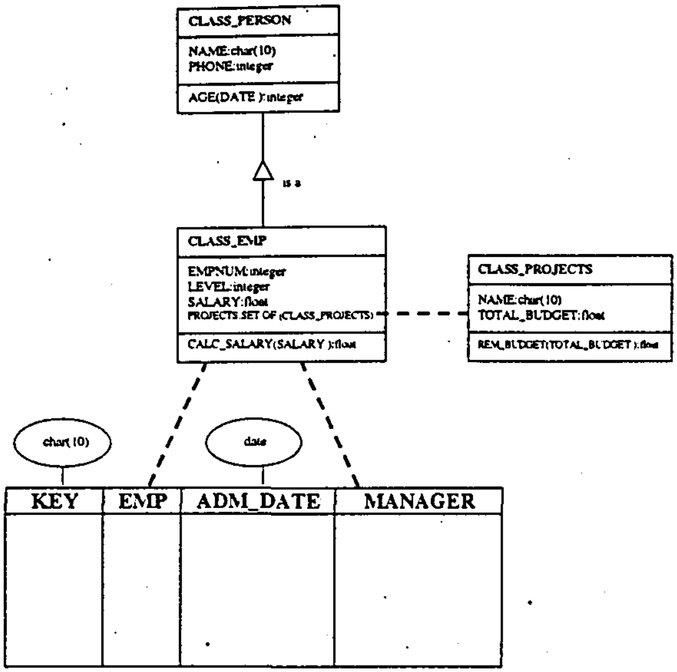


Figure 3. Example of a table with complex columns

5. Conclusions and future work

It is important that the software products, and obviously databases, are evaluated for every relevant quality characteristics, using validated or widely accepted metrics. These metrics could help designers, choosing between alternative semantically equivalent schemata, the most maintainable one. Because of this, we think it is very important to measure databases 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 conceptual modelling and the relational and object-relational databases and control its quality.

It is also important to validate the metrics from a formal point of view in order to ensure its utility. Several frameworks for measure characterisation have been proposed. Some of them (Briand et al., 1996; Weyuker, 1988; Briand and Morasca, 1997) are based on axiomatic approaches. The goal of this approach is merely definitional by defining formally desirable properties for measures for a given software attribute, so axioms must be used as guidelines for the definition of a measure. Others (Zuse, 1998) are based on measurement theory which specifies the general framework in which measures should be defined. Some of the presented metrics have been formalised from both points of view, axiomatic approach (Piattini et al., 1998; Piattini et al., 2000) and measurement theory (Calero et al., 1999a; Calero et al. 2000).

However, into the aspects of software measurement, the research is needed (Neil, 1994), from theoretical but also from a practical point of view (Glass, 1996). So, it is necessary to do experiments to validate the metrics. In this line we have also validate empirically some of the presented metrics (Calero et al., 1999b).

But the controlled experiments have problems (like the large number of variables that causes differences, deal with low level issues, microcosm of reality and small set of variables) and limits (do not scale up, are done in a class in training situations, are made in vitro and face a variety of threats of validity). Then, is convenient to run multiple studies, mixing controlled experiments and case studies. We are now working on this last kind of empirical validation with our metrics.

We are also adapting our framework to several RDBMS that followed the SQL:1999 model, simplifying the measures to be taken. In this way, we can give a more precise guide to the metrics usage (Churcher y Shepperd, 1995), and develop a tool for automatic metrics collection.

References

- 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.
- Briand L.C. and Morasca S. (1997). Towards a Theoretical Framework for Measuring Software Attributes. *Proceeding of the Fourth International, Software Metrics Symposium*, 119-126.
- Calero, C., Piattini, M., Ruiz, F. and Polo, M. (1999a), Validation of metrics for Object-Relational Databases, *International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (ECOOP99)*, (Lisbon ,Portugal. June 1999), 14-18
- Calero, C., Piattini, M., Polo, M., Ruiz, F. (1999b) Validating referential integrity as a database quality metric. *Proceedings of the First International Conference on Enterprise Information Systems, 45-50.*
- Calero, C., Piattini, M., Genero, M., Serrano, M. and Caballero, I. (2000). Metrics for Relational Databases Maintainability, *UKAIS2000*, Cardiff, UK.
- Card, D.N. and Glas, R.L. (1990). *Measuring Software Design Quality*. Englewood Cliffs. USA.
- Chidamber, S. and Kemerer, C. (1994), A metric suite for object-oriented design, *IEEE Trans. Software Eng.*, vol. 20,no. 6, June, 1994, pp. 476-493.
- Churcher, N.J. and Shepperd, M.J. (1995). Comments on "A Metrics Suite for Object-Oriented Design". IEEE Trans. on Software Engineering, Vol. 21(3). pp. 263-265.
- Codd, E.F. (1970). A Relational Model of Data for Larged Shared Data Banks. CACM, Vol.13 (6). pp. 377-387.
- Date, C.J. (1995). An Introduction to Database Systems. 6th. Addison-Wesley, Reading, Massachusetts.
- Elmasri, R. and Navathe, S. (1999). *Fundamentals of Database Systems*. Third edition. Addison-Wesley. Massachussets
- Fenton, N. (1994). Software Measurement: A Necessary Scientific Basis. *IEEE Transactions on Software Engineering*, Vol. 20(3). pp. 199-206.
- Fenton, N. and Pfleeger, S. L. (1997). Software Metrics: A Rigorous Approach 2nd. edition. London, Chapman & Hall.
- Glass, R. (1996). The Relationship Between Theory and Practice in Software Engineering. IEEE Software, November, Vol.39 (11). pp 11-13.
- 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
- Henderson-Sellers, B. (1996). Object-oriented Metrics - Measures of complexity. Prentice-Hall, Upper Saddle River, New Jersey.
- ISO, (1994). Software Product Evaluation-Quality Characteristics and Guidelines for their Use. *ISO/IEC Standard 9126*, Geneva.
- Li, H.F. and Chen, W.K. (1987). An empirical study of software metrics. IEEE Trans. on Software Engineering, Vol.13 (6). pp. 679-708.
- Melton, A. (ed.) (1996). Software Measurement. London, International Thomson Computer Press.
- Neil, M. (1994) Measurement as an Alternative to Bureaucracy for the Achievement of Software Quality. Software Quality Journal Vol.3 (2). pp. 65-78.
- Piattini, M., Calero, C., Polo, M. and Ruiz, F. (1998). Maintainability in Object-Relational Databases. . Proc of The European Software Measurement Conference FESMA 98, Antwerp, May 6-8, Coombes, Van Huysduynen and Peeters (eds.). pp. 223-230.
- Piattini, M., Calero, C., Polo, M. and Ruiz, F. (2000). Towards a metric suite for relational databases complexity, Accepted for IDPT 2000, June 2000, Dallas (USA)
- Pigoski, T.M. (1997). *Practical Software Maintenance*. Wiley Computer Publushing. New York. USA
- 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.). pp. 199-211.
- Weyuker, E.J. (1988). Evaluating software complexity measures. IEEE Transactions on Software Engineering Vol.14(9). pp. 1357-1365.
- Zuse, H. (1998). A Framework of Software Measurement. Berlin, Walter de Gruyter.