International Journal of Software Engineering and Knowledge Engineering
Vol. 11, No. 6 (2001) 703–729
© World Scientific Publishing Company

A METRIC-BASED APPROACH FOR PREDICTING CONCEPTUAL DATA MODELS MAINTAINABILITY

MARIO PIATTINI*,‡, MARCELA GENERO*,§ and LUIS JIMÉNEZ†,¶
*ALARCOS Research Group,
†ORETO Research Group,
University of Castilla-La Mancha,
Ronda de Calatrava, 5, 13071, Ciudad Real, Spain
‡mpiattin@inf-cr.uclm.es
§mgenero@inf-cr.uclm.es
¶ljimenez@inf-cr.uclm.es

Received 1 February 2001
Accepted 1 August 2001

It is generally accepted in the information system (IS) field that IS quality is highly dependent on the decisions made early in the development life cycle. The construction of conceptual data models is often an important task of this early development. Therefore, improving the quality of conceptual data models will be a major step towards the quality improvement of the IS development. Several quality frameworks for conceptual data models have been proposed, but most of them lack valid quantitative measures in order to evaluate the quality of conceptual data models in an objective way.

In this article we will define measures for the structural complexity (internal attribute) of entity relationship diagrams (ERD) and use them for predicting their maintainability (external attribute). We will theoretically validate the proposed metrics following Briand *et al.*'s framework with the goal of demonstrating the properties that characterise each metric. We will also show how it is possible to predict each of the maintainability sub-characteristics using a prediction model generated using a novel method for induction of fuzzy rules.

Keywords: Information systems quality; entity relationship diagram maintainability; structural complexity metrics; maintainability prediction; theoretical validation; empirical validation; fuzzy classification and regression tree.

1. Introduction

In a marketplace of highly competitive products, the importance of delivering quality is no longer an advantage, but a necessary factor for success. It is generally accepted within the information system (IS) field that the quality of IS is highly dependent on decisions made early in the development life cycle. Generally, problems in the artifacts produced in the initial stages of IS development spread to the artifacts produced in later stages, where they are much more costly to identify and correct [1]. Conceptual data models are often an important task of this early

development stage. Moreover, conceptual data models lay the foundation of all later design work and also determine the information that can be represented by an IS [2]. Therefore, the quality of conceptual data models has a significant impact on the quality of the IS which is ultimately implemented [3], and an even greater impact if we take into account the size and complexity of current IS. Therefore, it is time to consider conceptual data model quality as a main goal to pursue, instead of a subproduct of information modelling or database creation processes.

There are different kinds of conceptual data models, such as traditional (some variants of entity-relationship models) and object-oriented models (UML and OMT diagrams). In this article we will focus on entity relationship diagrams (ERD) [4, 5] because in today's IS design world they are still the preferred method of conceptual modelling [6].

In practice, evaluation of the quality of conceptual data models takes place in an ad hoc manner, if at all. There are no generally accepted guidelines for evaluating the quality of conceptual data models, and little agreement even among experts as to what makes a "good" conceptual data model [7]. Wand and Weber [8] argue that for IS design to go from an art to a science, there is a need to find ways of formally evaluating designs rather than relying solely on the judgement of the designer. The truth is that the notion of quality in conceptual modelling is poorly understood and, in most literature only 'bread and butter' lists of properties have been provided [9–11]. Only a few comprehensive and structured quality evaluation frameworks have been proposed which attempt to address quality in conceptual modelling in a much more systematic way [12–15]. The reader who is interested in a deeper study of these frameworks, may find a summary in [16]. Although these frameworks contribute to our understanding of quality issues of conceptual modelling, so far they are on a high level of abstraction. Both Lindland *et al.* [12] and Krogstie *et al.*[13], underline the necessity of enriching their frameworks with quantitative and objective measures, to reduce subjectivity and bias in the assessment of the quality of conceptual data models. The early availability of metrics allows designers to measure the quality of conceptual data models in order to assess (and if necessary to improve) the quality of the IS from the early phases of their life cycle.

Within the field of software measurement, a plethora of metrics have been proposed for measuring software products, processes and resources [17–19]. Unfortunately, almost all the metrics proposed until now have focused on program characteristics or on detailed design, without paying special attention to conceptual data models. Some exceptions known to us are the metrics proposed by Eick [20], Gray *et al.* [21], Kesh [22] and Moody [23]. Although all of these metrics proposals are a good starting point to think about quality in conceptual modelling on a numeric scale, most of them are subjective, lack empirical and theoretical validation and some of them are not useful in practice. Thus, there is a need for metrics and quality models that can be applied in the early stages of IS design, and we are particularly concerned with those applied to ERDs, to ensure that those designs have favourable internal properties that will lead to the development of quality IS.

We will define metrics for measuring ERD structural complexity (internal attribute), with the goal of using these early metrics to predict ERD maintainability (external attribute), which indirectly influences the IS maintainability. These metrics can also act as a guide to improve ERDs and explore alternatives.

This paper is organised as follows: In Sec. 2, we define a set of metrics for measuring ERD structural complexity following the GQM paradigm [24–25]. In Sec. 3, we present the theoretical validation of the proposed metrics following the framework proposed by Briand *et al.* [26] with the goal of demonstrating the properties that characterise each metric. In Sec. 4, we describe a controlled experiment, carried out in order to ascertain the relationships that exist between the proposed metrics and maintainability, and also to obtain a prediction model for ERD maintainability from the metric values. Lastly, in Sec. 5, we draw our conclusions, and present future trends in the field of measurement of conceptual models.

2. Definition of Measures for ERD using GQM

The Goal/Question/Metric(GQM) paradigm [24–25] provides a framework for deriving measures from measurement goals. The measurement goal should be clearly connected with an industrial goal, so the measurement program responds to a software organization's needs. In GQM each metric is deduced using a top-down approach covering three levels: at conceptual level goals are defined, at operational level questions are defined, and at the quantitative level metrics are derived. The GQM approach results in a set of metrics whose utility is clearly justified [27].

2.1. *Goal*

In our case the goal is to analyse *ERD with* the purpose of *evaluating maintainability* from the viewpoint of the *database designer* or IS design in *software development companies/departments.* The goal is therefore defined in terms of the entities shown in Fig. 1.

Object of study:	ERD
Purpose:	Evaluating
Quality focus:	Maintainability
Viewpoint:	Database designer/IS designer
Environment:	Software development companies/departments

Fig. 1. Goal of ERD metrics.

2.1.1. *Object of study*

ERDs are the most popular conceptual data models, providing a graphical notation that allows us to represent any Universe of Discourse in a simple and easy way to understand.

2.1.2. *Quality focus*

The ISO/IEC 9126 [28] defines software quality as composed of six external characteristics of interest, namely, functionality, reliability, efficiency, usability, maintainability and portability. In turn, each of these quality characteristics is refined into sub-characteristics. The focus of this work is on ERD maintainability, because maintainability has been and continues to be one of the pressing challenges facing any software development department. To our knowledge, not all of the maintainability sub-characteristics proposed in that standard are suitable for ERDs. Therefore, we distinguish six sub-characteristics for maintainability:

- Understandability: the ease with which the conceptual data model can be understood.
- Simplicity: means that the conceptual data model contains the minimum number of constructions possible.
- Analysability: the capability of the conceptual data model to be diagnosed for deficiencies or for parts to be modified to be identified.
- Modifiability: the capability of the conceptual data model to enable a specified modification to be implemented.
- Stability: the capability of the conceptual data model to avoid unexpected effects from modifications.
- Testability: the capability of the conceptual data model to enable modifications to be validated.

External quality attributes as maintainability sub-characteristics can only be measured late in the IS life cycle. We therefore need to identify early quality indicators based, for example, on the structural complexity (internal attribute) of ERD. Thus, our purpose is to define metrics to quantify ERD structural complexity and afterwards to ascertain how each of these metrics is related to each of the maintainability sub-characteristics.

2.2. *Questions*

As ERD maintainability is influenced by structural complexity, which, in turn, depends on elements that compose an ERD (relationships, entities, and attributes), three important questions arise:

- How is ERD maintainability affected by the structural complexity caused by the number of entities?

— How is ERD maintainability affected by the structural complexity caused by the number of attributes?
— How is ERD maintainability affected by the structural complexity caused by the number of relationships?

2.3. *Metric definition*

Reasoning with expert database designers and trying to respond to the questions brought up by the GQM method, we have identified several metrics which deal with the structural complexity of an ERD. All the metrics can be applied at diagram level, and are classified in the following categories: Entity metrics, Attribute metrics and Relationship metrics.

2.3.1. *Entity metrics*

— NE METRIC. We define the Number of Entities metric as the number of entities within an ERD.

2.3.2. *Attribute metrics*

— NA METRIC. The Number of Attributes metric is defined as the total number of attributes within an ERD, considering both entity and relationship attributes. This figure includes simple attributes, derived attributes, composite attributes and also multivalued attributes, each of which take the value 1 when calculating the metric.
— NDA METRIC. An ERD is minimal when every aspect of the requirements appears once in the diagram, i.e., an ERD is minimal if it does not have any redundancies. One of the sources of redundancy in ERDs is the existence of derived attributes. An attribute is derived when its value can be calculated or deduced from the values of other attributes. The Number of Derived Attributes metric is defined as the total number of derived attributes within an ERD.
— NCA METRIC. The Number of Composite Attributes metric is defined as the total number of composite attributes within an ERD.
— NMVA METRIC. The Number of Multivalued Attributes metric is defined as the total number of multivalued attributes within an ERD.

2.3.3. *Relationship metrics*

— NR METRIC. The Number of Relationships metric is defined as the total number of relationships within an ERD, considering only common relationships.
— NM:NR METRIC. The Number of M:N Relationships metric is defined as the total number of M:N relationships within an ERD.
— N1:NR METRIC. The Number of 1:N Relationships metric is defined as the total number of 1:N relationships (including also 1:1 relationships) within an ERD.

— NN-ARYR METRIC. The Number of N-ary Relationships metric is defined as the total number of N-ary relationships (not binary) within an ERD.
— NBINARYR METRIC. The Number of Binary Relationships metric is defined as the total number of binary relationships within an ERD.
— NIS_AR METRIC. The Number of IS_A Relationships metric is defined as the total number of IS_A relationships (generalisation/specialisation) within an ERD. In this case, we consider one relationship for each child-parent pair within the IS_A relationship.
— NREFR METRIC. The Number of Reflexive Relationships metric is defined as the total number of reflexive relationships within an ERD.
— NRR METRIC. Another source of redundancy in an ERD are redundant relationships. We define the Redundant Relationship metric as the number of relationships that are redundant in an ERD.

Figure 2 summarises the application of the GQM paradigm for deriving metrics for ERD structural complexity evaluation.

These are open-ended metrics [29], i.e., they are not restricted in an interval. Close-ended metrics (percentage metrics) could also be useful, such as the following: NRR/NR; NDA/NA; NM:NR/NR; N1:NR/NR, NBinaryR/NR, NN-AryR/NR, etc.

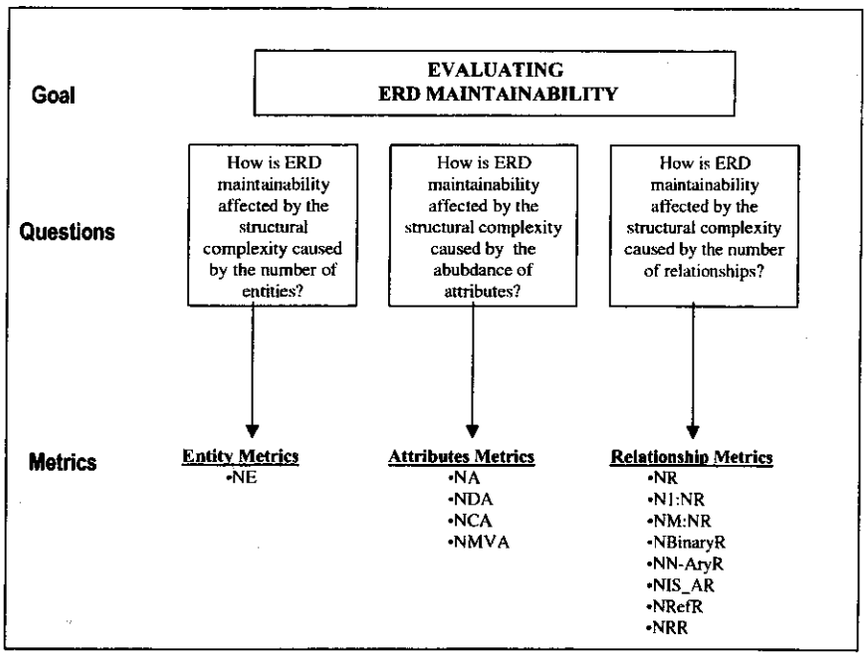


Fig. 2. ERD metrics obtained by the application of the GQM paradigm.

3. Theoretical Validation

The main goal of theoretical validation is to check if the intuitive or formal idea of the attribute being measured is reflected in the measurement. This is done by analysing the theoretical requisites which must be satisfied when measuring. Essentially, it is based on the analysis of the properties of the attribute that we wish to measure.

Even though several attempts have been made to establish how to carry out theoretical validation in software measurement, there is not yet a standard, accepted way of theoretically validating a measure. As Van den Berg and Van den Broek [30] said "a standard on theoretical validation issues in software measurement is urgently required".

Work on theoretical validation has followed two paths which, rather than alternative are complementary.

(1) Measurement theory-based approaches [31–33]: check for a specific measure if the empirical relations between the elements of the real world established by the attribute being measured, are respected when measuring the attributes.
(2) Property-based approaches [26, 34]. Aim to formalize the empirical properties that a generic attribute of software or a system (eg. the complexity or size) must satisfy in order for it to be used in the analysis of any measurement proposed for that attribute.

In the next subsection we present the property-based approach proposed by Briand *et al.* [26], and later improved by Morasca *et al.* [35], with the goal of establishing which mathematical properties must be achieved by the metrics in accordance with the internal software attribute they intend to measure: size, complexity, cohesion, coupling, length.

3.1. *Introduction to Briand et al.'s framework*

The Briand *et al.*'s framework [26] provides a set of mathematical properties that characterise and formalise several important measurement concepts: size, length, complexity, cohesion and coupling. This framework is based on a graph-theoretic model of a software artifact, which is seen as a set of elements linked by relationships. The different kinds of metrics, and the properties which identify each one, are applicable to modules and modular systems.

We describe the basic concepts of this framework and the set of properties for the metrics of five software attributes of interest.

A *System* is defined as a pair (E, R), where E is the set of elements of S, and R is a binary relation among the elements of $E(R \subseteq E_x E)$. From this point, we say that m is a module of S if and only if $E_m \subseteq E$, $R_m \subseteq E_m \times E_m$ and $R_m \subseteq R$. One extension of Briand *et al.* (1996) was made by Morasca *et al.* (1997) which allows not only binary relations.

The elements of a module are connected with elements of other modules of the system with input and output relations. So, the following two sets are defined:

$$\text{Input}R(m) = \{(e_1, e_2) \in R / e_2 \in E_m \wedge e_1 \in E - E_m\}$$

$$\text{Output}R(m) = \{(e_1, e_2) \in R / e_1 \in E_m \wedge e_2 \in E - E_m\}$$

$MS = (E, R, M)$ is a Modular System if $S = (E, R)$ is a system according to the previous definition and M is a collection of modules of S with no common elements (they are disjoint).

IR is the union of all the relations which relate the entities of a concrete module (intramodule relationship). According to this definition, R–IR is the set of relations among elements of different modules (intermodule relationship).

Table 1 lists the properties that the metrics of size, length, complexity, cohesion and coupling must fulfil.

It is important to recall that coupling and cohesion are only defined in the context of modular systems, whereas size, length and complexity are defined for all systems.

These properties can be used to guide the search for new product measures and help to avoid future confusion, often encountered in the literature, about which properties product measures should or should not have. Studying measure properties is important in order to provide discipline and rigour in the search for new product measures.

3.2. *Theoretical validation of the proposed metrics*

NA METRIC. For our purpose and accordingly with Briand *et al.*'s framework, we consider that entities and relationships are system modules, the attributes are the elements and relationships are represented by the relation "belong to", which reflect that each attribute belongs to an entity or to a relationship. We will demonstrate that NA fulfils all of the axioms that characterise size metrics (see Table 1), as follows:

1. *Nonnegativity.* One ERD can or cannot have attributes, i.e., it could happen that NA = 0 or NA > 0, but never NA < 0.
2. *Null value.* If we have no attributes NA = 0.
3. *Module additivity.* If we consider that an ERD is composed of modules, i.e., entities and relationships, the number of attributes of an ERD will always be the sum of the number of attributes of all of its entities and relationships, because each attribute of an ERD is an attribute of either an entity or relationship.

Table 1. Properties that characterize measurement concepts.

Size	NONNEGATIVITY. The size of a system is nonnegative. NULL VALUE. The size of a system is null if it has no elements. MODULE ADDITIVITY. The size of a system is equal to the sum of the sizes of two of its modules such that any element of the system is an element of either one module or the other.
Length	NONNEGATIVITY. The length of a system is nonnegative. NULL VALUE. The length of a system is null if has no elements. NONINCREASING MONOTONICITY FOR CONNECTED COMPONENTS. Let S be a system and m be a module of S such that m is represented by a connected component of the graph representing S. Adding relationships between elements of S does not increase the length of S. NONDECREASING MONOTONICITY FOR NON-CONNECTED COMPONENTS. Let S be a system and m_1 and m_2 be two modules of S such that m_1 and m_2 are represented by two separate connected components of the graph representing S. Adding relationships from elements of m_1 to elements of m_2 does not decrease the length of S. DISJOINT MODULES. The length of a system made of two disjoint modules m_1, m_2 is equal to the maximum of the lengths of m_1 and m_2.
Complexity	NONNEGATIVITY. The complexity of a system is nonnegative. NULL VALUE. The complexity of a system is null if it has no relations. SYMMETRY. The complexity of a system does not depend on the convention chosen to represent the relationships between its elements. MODULE MONOTONICITY. The complexity of a system is no less than the sum of the complexities of any two of its modules with no relationships in common. DISJOINT MODULE ADDITIVITY. The complexity of a system composed of two disjoint modules is equal to the sum of the complexities of the two modules.
Cohesion	NONNEGATIVITY AND NORMALIZATION. The cohesion of a [module $m = (E_m, R_m)$ of a modular system $MS = (E, R, M)$ | modular system $MS = (E, R, M)$] belongs to a specified interval: [Cohesion(m) $\in$ [0, Max] | Cohesion(MS) $\in$ [0, Max]] NULL VALUE. If there is no intramodule relationship among the elements of a (all) module(s), then the module (system) cohesion is null. [$R_m = \emptyset \Rightarrow$ Cohesion(m) $= 0$ | $IR = \emptyset \Rightarrow$ Cohesion(MS) $= 0$] MONOTONICITY. Adding intramodule relationships does not decrease [module | modular system] cohesion. COHESIVE MODULES. The cohesion of a module obtained by putting together two unrelated modules is not greater than the maximum cohesion of two original modules.
Coupling	When referring to module coupling, we will use the word coupling to denote either inbound or outbound coupling, and Outer$R(m)$ to denote either Input$R(m)$ or Output$R(m)$. NONNEGATIVITY. The coupling of a module [$m = (E_m, R_m)$ of a modular system $MS = (E, R, M)$ | modular system $MS = (E, R, M)$]] is nonnegative: [Coupling(m) $\in 0 \geq$ Coupling(MS) ≥ 0] NULL VALUE. The coupling of a [module $m = (E_m, R_m)$ of a modular system [modular system $MS = (E, R, M)$] is null if [Outer$R(m)$ | $R - IR$] is empty: [Outer$R(m) = \emptyset \Rightarrow$ Coupling(m) $= 0$ | $R - IR = \emptyset \Rightarrow$ Coupling(MS) $= 0$] MONOTONICITY. Adding intermodule relationships does not decrease coupling. MERGING OF MODULES. The coupling of a [module | modular system] obtained by merging two modules is not greater than the [sum of the couplings of two original modules | coupling of the modular system], since the two modules may have common intermodule relationships. DISJOINT MODULE ADDITIVITY. The coupling of a [module | modular system] obtained by merging two unrelated modules is equal to the [sum of the coupling of two original modules | coupling of the original modular system].

Following a similar reasoning as that used for the NA metric, it can be proved that the metrics NCA, NMVA and NDA are also size metrics.

NE METRIC. For our purpose and in accordance with Briand *et al.*'s framework, we consider that an ERD is a system composed of entities (elements) and relationships (relations). A module is composed of a subset of the ERD entities and ERD relationships. We will demonstrate that NE fulfils all of the axioms that characterise size metrics (see table 1), as follows:

1. *Nonnegativity.* The number of entities in an ERD is always greater than zero, so that NE can never be negative.
2. *Null value.* If there are no entities NE=0.
3. *Module additivity.* If we consider that an ERD is composed of modules with no common entities, the number of entities of an ERD will be always the sum of the number of entities of its modules.

NR METRIC. For our purpose and accordingly with Briand *et al.*'s framework, we consider that an ERD is a system composed of entities (elements) and relationships (relations). A module is composed of a subset of the ERD entities and ERD relationships. We will demonstrate that NR fulfils all of the axioms that characterise complexity metrics (see Table 1), as follows:

1. *Nonnegativity.* It is obvious that there is always a null or positive value of relationships. Then, NR ≥ 0.
2. *Null value.* If there are no relationships within an ERD then NR = 0.
3. *Symmetry.* The number of relationships does not depend on the convention used to represent the relationships.
4. *Module monotonicity.* According to the definition of this property, it is obvious that: being m_1 and m_2 any two modules of the ERD with no relationships in common, the value of NR(ERD) $\geq$ NR(m_1) + NR(m_2).
5. *Disjoint module additivity.* Effectively let m_1 and m_2 be any two disjoint modules such that ERD = $m_1 \cup m_2$. Let NR_1 and NR_2 be the number of relationships in the m_1 and m_2 modules. Obviously: NR = NR_1 + NR_2, because m_1 and m_2 are disjoint modules.

Following an analogous reasoning as that used for the NR metric, it can be proved that the metrics NM:NR, N1:NR, NBinaryR, NN-AryR, NIS_AR, NRefR and NRR are also complexity metrics.

Also, we have put the proposed metrics under theoretical validation in order to ascertain each scale type. Following Zuse's formal measurement framework [31] most of the proposed open-ended metrics are in ratio scale and the close-ended metrics are in absolute scale, which has a relevant importance in the types of operations and statistical analyses that can be applied to the measurement values.

4. Empirical Validation

As in other aspects of Software Engineering, proposing techniques and metrics is not enough, it is also necessary to carry out empirical validations to assure their utility in practice. Empirical validation is critical to the success of software measurement [18, 36–38].

Taking into account some suggestions provided in [39–40] about how to perform empirical studies in Software Engineering, we carried out a controlled experiment pursuing the following goals:

(1) ascertaining if any relationship exists between the metrics (NE, NA, NR, NM:NR, N_1:NR, NN-aryR, NBinaryR and NIS_AR) and each of the maintainability sub-characteristics: understandability, simplicity, analysability, modifiability, stability, and testability.
(2) establishing a prediction model for each of the maintainability sub-characteristics from metric values obtained at the early phases of IS life cycle.

We only consider the metrics (NE,NA,NR,NM:NR,N1:NR,NN-aryR, NBinaryR and NIS_AR) because the others take the value zero in most of the ERDs selected for the experiment.

In the remainder of this section, we show the experimental design, how we collect the experimental data, the technique used to analyse the empirical data, the results of the experiment and the threats to validity.

4.1. *Subjects*

The experimental subjects used in this study were 9 professors and 7 students enrolled in the final-year Computer Science course in the Department of Computer Science at the University of Castilla-La Mancha in Spain. All the professors belong to the Software Engineering field and they have on average five years of experience in the design of ERDs. By the time the experiment was done all the students had taken two courses on Software Engineering and one course on Databases, in which they learnt, in depth, how to build ERDs. Moreover, subjects were given an intensive training session before the experiment took place.

4.2. *Experimental materials and tasks*

The subjects were given twenty-four ERDs [41, 42] of the different universes of discourse, but general enough to be easily understood by each of the subjects. The complexity of the selected ERDs is different, because we selected them taking into account that they cover a wide range of the metric values. Each diagram has a test enclosed which includes the description of maintainability sub-characteristics, such as: understandability, simplicity, analysability, modifiability, stability and testability. Each subject had to rate each sub-characteristic using a scale consisting of seven

linguistic labels. For example, for understandability we proposed the following linguistic labels:

Extremely difficult to understand	Very difficult to understand	A bit difficult to understand	Neither difficult nor easy to understand	Quite easy to understand	Very easy to understand	Extremely easy to understand

We associate the numbers 1 to 7 to each linguistic label: The worst case takes the value seven (extremely difficult to understand), and the best case takes the value one (extremely easy to understand).

We also gave to the subjects a guide containing a detailed explanation about the ER notation used in the diagrams.

We allowed one week to do the experiment, i.e., each subject had to carry out the test alone, and could use unlimited time to solve it. After completion of the tasks the subjects were asked to complete a debriefing questionnaire. This questionnaire included (i) personal details and experience, (ii) opinions on the influence of different components of ERD, such as: entities, relationships, attributes, etc. on their maintainability.

4.3. *Experimental design and data collection*

The INDEPENDENT VARIABLE is the structural complexity of the ERDs, measured using the metrics proposed in Sec. 2.

The DEPENDENT VARIABLES are those of the maintainability sub-characteristics: understandability, simplicity, analysability, modifiability, stability and testability, measured according to the subject's rating.

We collected all the tests, controlling until they were complete. As all of them were complete and the subjects had, at least, medium experience in building ERDs (this fact was corroborated analysing the responses of the debriefing questionnaire) we consider their subjective evaluation to be reliable.

4.4. *Data analysis technique and results*

Due to the nature of the software development process and products, one cannot expect to use in Software Engineering the same measurement data analysis techniques that are used in "exact" sciences, e.g., physics, chemistry, nor obtain the same degree of precision and accuracy [43]. Statistic and numerical machine learning techniques are very dependent on the data set and the models and are not qualitative [44]. So, we need a machine learning technique that will allow us to build prediction models with two characteristics: they must be highly qualitative and more straightforward and intelligible to human beings.

Following this idea, Ebert [45] and Ebert and Baisch [46] proposed a fuzzy classification using a set of software quality metric values by unsupervised learning

process. But our proposal is different because we followed a supervised learning method, which is a novel data analysis approach based on a rule system with linguistic variables [47]. This approach, as a method of supervised learning, provides models that allow us to discover the most relevant conceptual relationships between the data we are analysing, where the accuracy of those models is sacrificed in favour of its simplicity and ease of understanding. In order to obtain these models we will use a method for induction of fuzzy rule systems, which is introduced in the next subsection.

4.4.1. *Method for induction of fuzzy rule systems*

This method for induction of fuzzy rule systems is a generalisation of the classical regression approach. In order to explain this method, first we make some introductory comments.

Let $S = \{s_1, \ldots, s_n\}$ be a set of data, which are defined by the value given by a set of variables $X = \{X^1, \ldots, X^d, Y\}$, $s_i = (x_i^1, \ldots, x_i^d, y_i)$. Let us suppose there is a function F which is only known at the points of S so that $F(s_i) = y_i$. The objective of regression, which is defined classically as a parametric function F', is to minimise the distance between the sample output values y_i and the predicted value $F'(s_i)$.

The regression problem differs from the classification problem in that the output variable can be continuous, where as in classification this is strictly categorical. From this perspective, classification can be thought to be a subcategory of regression. Recursive partition methods for classification problems such as ID3 decision tree have been applied, as a regression method by restrictions, in the CART program [48], developed in the statistical research community.

The CART program (classification and regression tree) is based on building a tree structure where the regions are defined by possible answers to a set of questions raised about the variables that define the problem. This approach creates a set of disjoint regions $SR = \{r_1, \ldots, r_p\}$ of the global domain of the problem. These partitions are obtained according to the kind of questions and answers that we have formulated. Our method generalises the kind of partitions obtained, using the method proposed in [49, 50] based on the fuzzy set theory [51]. In the fuzzy set theory the membership of an element x (which belongs to a domain X) to a fuzzy set A, is in the interval [0,1]. If $A(x)$ is the membership function, $A(x) = 0$ means that x is not a member of A, and $A(x) = 1$ means that x is a full member of A. $A(x)$ can also take other values between the interval [0,1], which graduate the membership of x to the fuzzy set A. We use Linares *et al.*'s idea [49] in order to define a fuzzy classification and regression tree (FCART), which produce a fuzzy rule system that represent our prediction model.

Hereafter, we will explain how to obtain the rules. Let A_T be a fuzzy set defined over the set S in the node T, such that $A_T{:}S \rightarrow [0, 1]$. In the root node this fuzzy set is A_{root}: $S \rightarrow 1$. We define the output value at node T by the membership value

of each point s_i, $A_T(s_i)$ and the output value in this point y_i.

$$F''(T) = \frac{\sum_{i=1}^{n} A_T(s_i)^m * y_i}{\sum_{i=1}^{n} A_T(s_i)^m}$$

where $AT(x)$ is defined by $\min_{j=1}^{d} A_T^j(x^j)$, and $A_T^j(x^j)$ is a membership of a fuzzy set defined over j-metric domain. The value of m is a real number greater than 1 which weights the contribution of the membership value.

The estimated error is defined by

$$\mathrm{Err}(T) = \frac{\sum_{i=1}^{n} (F''(T) - y_i)^2 * A_T(s_i)^m}{\sum_{i=1}^{n} A_T(s_i)^m}$$

Now, our problem is how to create the set of questions to split the node T. The questions were formulated for each variable, obtaining a binary fuzzy partition for each one. We supposed a binary fuzzy partition for the fuzzy set of node T by the fuzzy set A_T^j of variable j, this is $p_T^j = \{B(x), C(x)\}$, when $A_T^j(x) = B(x) + C(x)$. This partition creates two new nodes and two new fuzzy sets for them

$$A_{T_1}(s_i) = \min(A_T(s_i), B(x_i^j))$$

$$A_{T_2}(s_i) = \min(A_T(s_i), C(x_i^j))$$

IF variable$_j$ IS A_{T_1} THEN ...

ELSE IF variable^j IS A_{T_2} THEN ...

Following the CART program, we defined the proportion for $B(x)$ and $C(x)$ in relation to fuzzy set A_T^j as

$$P(T_1) = \frac{\sum_{i=1}^{n} B(x_i^j)}{\sum_{i=1}^{n} A_T^j(x_i^j)}$$

The quality of the partition can be estimated through the equation

$$C(T, p^j) = \mathrm{Err}(T_1) * P(T_1) + \mathrm{Err}(T_2) * P(T_2)$$

where p_j is the fuzzy partition of variable j.

We will select the p^* which minimises the value of $C(T, p_j)$. This approach to create the questions gives rise to a hierarchical fuzzy partition for each variable. This split process raises relevant metrics to define the model. The partition process

stops when a stop criterion is raised. In this case we look for larger estimated error. So that stop criterion can be

$$\text{ERROR} = \max_{T \in T} \text{Err}(T) \leq \varepsilon$$

where $\overline{T}$ is the set of leaf nodes of the tree, and each one represents a region for our solution. The output value y_i' for a input value s_i is

$$F'(s) = \frac{\sum_{T \in T} A_T(s)^m * F''(T)}{\sum_{T \in T} A_T(s)^m}$$

4.4.2. *Results of the experiment*

First, we establish a fuzzy rule system for understandability, a maintainability sub-characteristic. We use a learning set with X = {set of our metrics} and Y = {the values of understandability obtained in the experiment}. We have obtained 384 values (24 ERDs and 16 subjects) of this unknown function:

F(NE,NA,NR,NM:NR,N1:NR,NN-AryR,NBinaryR,NIS_AR) = Understandability.

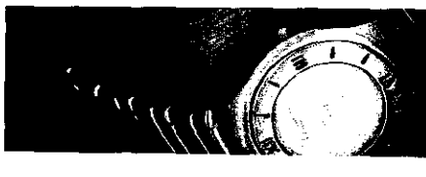
By the induction method described above we have obtained a prediction model composed of fuzzy rules, generated from the fuzzy tree obtained [49, 50].

A fuzzy rule is formed by the antecedent part (left part of the rule) and the consequent part (right part of the rule) to reflect causal-effect relation. The antecedent part is formed by aggregation of fuzzy statements as "X is A", where X is a metric value and A is a fuzzy set over metric domain.

In this example we have used a trapezoidal function for fuzzy sets, which are defined by four numbers. The fuzzy set $[a, b, c, d]$ has the following membership:

$$A(x, a, b, c, d) = \begin{cases} 0 & x \Leftarrow a \\ \dfrac{x-a}{b-a} & a < x < b \\ 1 & b \leq x \leq c \\ \dfrac{d-x}{d-c} & c < x < d \\ 0 & x \geq d \end{cases}$$

Table 2 shows the set of fuzzy rules that represents the understandability fuzzy model (they were generated using an automatic tool) where [MIN,MAX] represent the whole domain of every metric. The rows 0–14 represent the 15 rules obtained by the induction approach. For example rule 12 can be read as "*IF NA is in the fuzzy set [19.0,29.0,35.0,44.0] and N1:NR is in the fuzzy set [3.0,5.0,MAX,MAX] and NBinaryR is in the fuzzy set [6.0,8.0,8.0,10.0] THEN understandability (Y) is 3.87*". This rule gives a 0.12 error and has a data coverage percentage of 8.91%.



As our goal is to define a rule system with linguistic variables we now briefly introduce the concept of linguistic variables. A linguistic variable [47] is a tuple $(X, T(X), V, G, M)$ where X is the variable name (in our case metric names), $T(X)$ is the label set, V is the domain in which the variables are defined, M is a semantic rule that associates each label to a fuzzy set defined over the domain V. With our method once the fuzzy sets are inducted, we will label each fuzzy set (see Table 3) to establish the linguistic variables we will use.

Table 4 shows the rules specified in Table 2 using linguistic variables where EVERYTHING is the complete domain of the metric.

Table 2. Understandability fuzzy model.

RULE	NA	N1:NR	NBinaryR	Y	ERROR	COV%
0	[MIN,MIN,MAX,MAX]	[1.0,2.0,3.0,5.0]	[6.0,8.0,MAX,MAX]	2.60	0.08	4.62
1	[MIN,MIN,MAX,MAX]	[0.0,1.0,1.0,2.0]	[MIN,MIN,MAX,MAX]	3.02	0.17	8.89
2	[MIN,MIN,MAX,MAX]	[1.0,2.0,3.0,5.0]	[MIN,MIN,3.0,5.0]	2.04	0.13	10.28
3	[MIN,MIN,MAX,MAX]	[MIN,MIN,0.0,1.0]	[6.0,8.0,MAX,MAX]	2.86	0.01	0.00
4	[MIN,MIN,19.0,29.0]	[3.0,5.0,MAX,MAX]	[6.0,8.0,8.0,10.0]	3.66	0.03	2.19
5	[MIN,MIN,MAX,MAX]	[MIN,MIN,0.0,1.0]	[3.0,5.0,6.0,8.0]	2.92	0.06	4.28
6	[MIN,MIN,MAX,MAX]	[MIN,MIN,0.0,1.0]	[MIN,MIN,3.0,5.0]	2.77	0.18	11.74
7	[MIN,MIN,MAX,MAX]	[3.0,5.0,MAX,MAX]	[10.0,11.0,MAX,MAX]	4.75	0.06	4.39
8	[MIN,MIN,MAX,MAX]	[3.0,5.0,MAX,MAX]	[8.0,10.0,10.0,11.0]	3.43	0.11	5.53
9	[MIN,MIN,19.0,29.0]	[3.0,5.0,MAX,MAX]	[MIN,MIN,6.0,8.0]	2.35	0.07	5.59
10	[19.0,29.0,MAX,MAX]	[3.0,5.0,MAX,MAX]	[MIN,MIN,6.0,8.0]	3.28	0.11	9.92
11	[35.0,44.0,MAX,MAX]	[3.0,5.0,MAX,MAX]	[6.0,8.0,8.0,10.0]	3.26	0.07	4.66
12	[19.0,29.0,35.0,44.0]	[3.0,5.0,MAX,MAX]	[6.0,8.0,8.0,10.0]	3.87	0.12	8.91
13	[MIN,MIN,19.0,29.0]	[1.0,2.0,3.0,5.0]	[3.0,5.0,6.0,8.0]	2.42	0.11	10.84
14	[19.0,29.0,MAX,MAX]	[1.0,2.0,3.0,5.0]	[3.0,5.0,6.0,8.0]	2.81	0.08	7.51

Where: the column RULE represents each rule number; the columns NA, N1:NR, NBinary_R are the fuzzy sets associated with each metric name; the column Y is the output or the consequent of the rules, it means the understandability; the column ERROR is the error produced when the rule is generated and the column COV% is the data coverage percentage of each rule, taking into account the sample data.

Table 3. Labels of fuzzy sets.

NA		N1:NR		NBINARY_R	
Fuzzy Set	Label	Fuzzy Set	Label	Fuzzy Set	Label
[MIN,MIN,19.0,29.0]	LOW	[MIN,MIN,0.0,1.0]	VERY LOW	[MIN,MIN,3.0,5.0]	VERY LOW
[19.0,29.0,35.0,44.0]	MEDIUM	[0.0,1.0,1.0,2.0]	LOW	[3.0,5.0,6.0,8.0]	LOW
[35.0,44.0,MAX,MAX]	HIGH	[1.0,2.0,3.0,5.0]	MEDIUM	[6.0,8.0,8.0,10.0]	MEDIUM
		[3.0,5.0,MAX,MAX]	HIGH	[8.0,10.0,MAX,MAX]	HIGH

Table 4. Understandability linguistic fuzzy model.

RULE	NA	N1:NR	NBinaryR	Y	ERROR	COV%
0	EVERYTHING	MEDIUM	MEDIUM or HIGH	2.60	0.08	4.62
1	EVERYTHING	LOW	EVERYTHING	3.02	0.17	8.89
2	EVERYTHING	MEDIUM	VERY_LOW	2.04	0.13	10.28
3	EVERYTHING	VERY_LOW	MEDIUM or HIGH	2.86	0.01	0.00
4	LOW	HIGH	MEDIUM	3.66	0.03	2.19
5	EVERYTHING	VERY_LOW	LOW	2.92	0.06	4.28
6	EVERYTHING	VERY_LOW	VERY_LOW	2.77	0.18	11.74
7	EVERYTHING	HIGH	HIGH	4.75	0.06	4.39
8	EVERYTHING	HIGH	MEDIUM or HIGH	3.43	0.11	5.53
9	LOW	HIGH	LOW or VERY_LOW	2.35	0.07	5.59
10	MEDIUM or HIGH	HIGH	LOW or VERY_LOW	3.28	0.11	9.92
11	HIGH	HIGH	MEDIUM	3.26	0.07	4.66
12	MEDIUM	HIGH	MEDIUM	3.87	0.12	8.91
13	LOW	MEDIUM	LOW	2.42	0.11	10.84
14	MEDIUM or HIGH	MEDIUM	LOW	2.81	0.08	7.51

Where: the column RULE represents each rule number; the columns NA, N1:NR, NBinaryR are the linguistic variables associated with each metric name; the column Y is the output or the consequent of the rules, it means the understandability; the column ERROR is the error produced when the rule is generated and the column COV% is the data coverage percentage taking into account the sample data.

We can read the rule 12 as "*IF NA is MEDIUM and N1:NR is HIGH and NBinaryR is MEDIUM THEN understandability is 3.87, i.e. Neither difficult nor easy to understand*".

This fuzzy rule system represents a prediction model for the understandability that is highly natural and closer to the human mind, therefore it is in accordance with our goals, presented in the beginning of this section.

Now, we will show an example of understandability prediction using the proposed model, presented above. For example, suppose that we want to answer the following question: "*What is the value of understandability, if we know that NA value is 20, N1:NR value is 4 and NBinaryR value is 4?*" This question, can be answered by inference following Table 5.

Using approximate reasoning [52] and with a simplified consequent, detailed in Table 5, we obtain the result shown in Fig. 3.

The value of understandability is $5.138/2.1 = 2.45$. This value is 55% very easy to understand and 45% a bit easy to understand (see Fig. 3).

We also want to highlight another very important characteristic of our method which is the identification of relevant metrics. According to our sample and analysing understandability, the relevant metrics are NA, N1:NR and NBinary_R because only these metrics change the EVERYTHING value obtained by induction methods.

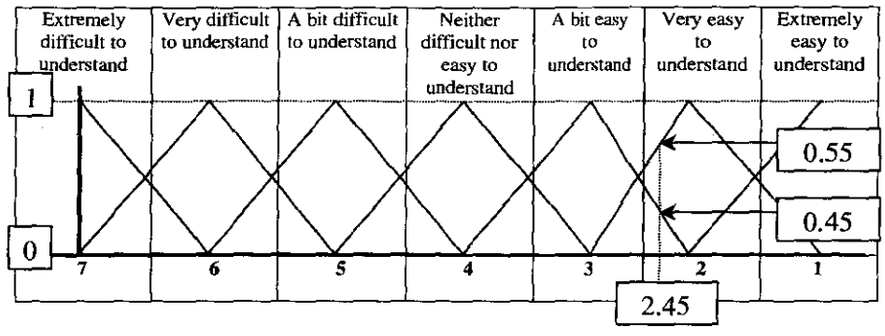


Fig. 3. Inference of a new understandability value.

Table 5. Inference of a new value of understandability.

RULE	NA(20)	N1:NR(4)	NBinaryR(4)	Y	MIN	MIN*Y
0	1	0.5	0	2.60	0	0
1	1	0	1	3.02	0	0
2	1	0.5	0.5	2.04	0.5	1.02
3	1	1	0	2.86	0	0
4	0.9	0.5	0	3.66	0	0
5	1	0	1	2.92	0	0
6	1	0	0.5	2.77	0	0
7	1	0.5	0	4.75	0	0
8	1	0.5	0	3.43	0	0
9	0.9	0.5	1	2.35	0.5	1.175
10	0.1	0.5	1	3.28	0.1	0.328
11	0	0.5	0	3.26	0	0
12	0.1	0.5	0	3.87	0	0
13	0.9	0.5	0.5	2.42	0.5	1.21
14	0.9	0.5	0.5	2.81	0.5	1.405
				SUM	2.1	5.138

Where: the column RULE is the rule number; the columns NA(20), 1:NR(4), BINARY_R(4) represent the degree of membership of each; the column Y represents the output or the consequent of the rules, it means the understandability; the column MIN represents the minimum of the degree of membership of the antecedents and the column MIN*Y represents the contribution of each rule to the final solution 5.138. This value is obtained by making a weighted average of the output (understandability) of each rule with its degree of membership.

For the sake of brevity we do not show how to obtain the prediction model for the other maintainability sub-characteristics, but they can be built in a similar way to the understandability fuzzy model shown in Table 3. In Appendix A we show the fuzzy models (see Tables 6, 7, 8, 9 and 10) for the rest of the main-

tainability sub-characteristics. Analysing the fuzzy models for each maintainability sub-characteristic we can deduce that:

— Of all the metrics used, the only ones that are relevant for determining the totality of the maintainability sub-characteristics are NA, N1:NR and NBinaryR. This fact is deduced because these are the metrics that most frequently appear in each of the fuzzy models representing each maintainability sub-characteristic (see Tables 2, 6, 7, 8, 9 and 10).
— We can establish a hierarchy of the relevant metrics according to the number of examples that they can differentiate. By doing this we can see that the metric which makes the greatest contribution is NA followed by NBinary_R and finally N1:NR which allows us to make small refinements.

4.5. *Threats to validity*

Following several empirical studies [39,40] we will discuss the empirical study's various threats to validity and the way we attempted to alleviate them.

4.5.1. *Threats to construct validity*

The construct validity is the degree to which the independent and the dependent variables accurately measure the concepts they purport to measure. The dependent variables we used are maintainability sub-characteristics: understandability, simplicity, analysability, modifiability, stability and testability. We propose subjective metrics for them (using linguistic variables), based on the judgement of the subjects (see Sec. 3.2). As the subjects involved in this experiment have medium experience in ER design we think their ratings could be considered significant. For construct validity of the independent variables, we have to address the question: To which degree do the metrics used in this study measure the concept they purport to measure? Our idea is to use metrics presented in Sec. 2 to measure the structural complexity of an ERD. From a system theory point of view, a system is called complex if it is composed of many (different types of elements), with many (different types of) (dynamically changing) relationships between them [53]. According to this, we think that the construct validity of our independent variables can thus be considered satisfactory. In spite of this, we consider that more experiments should be made, in order to draw a final conclusion to assure construct validity.

4.5.2. *Threats to internal validity*

The internal validity is the degree to which conclusions can be drawn about the causal effect of independent variables on the dependent variables. The following issues have been dealt with:

— DIFFERENCES AMONG SUBJECTS. Using a within-subjects design, error variance due to differences among subjects is reduced. As Briand et al. [40] remark in

software engineering experiments when dealing with small samples, variations in participant skills are a major concern that is difficult to fully address by randomisation or blocking. In this experiment, we believe that even though the professors and students had not exactly the same degree of experience in modelling with ERDs, the experience they had was enough to do the required tasks.

— KNOWLEDGE OF THE UNIVERSE OF DISCOURSE AMONG ERDS. The ERDs were from different universes of discourse but they are general enough to be easily understood for each of the subjects such that we believe the knowledge of the domain does not affect the internal validity.
— ACCURACY OF SUBJECT RESPONSES. Subjects assumed the responsibility for rating each maintainability sub-characteristic. As they have medium experience in ERD design, we think their responses could be considered valid.
— LEARNING EFFECTS. All the tests in each experiment were put in a different order, to avoid learning effects. Subjects were required to answer in the order in which the test appeared.
— FATIGUE EFFECTS. On average the experiment lasted for less than one hour, so fatigue was not very relevant. Also, the different order of the tests helped to prevent this effect.
— PERSISTENCE EFFECTS. In order to avoid persistence effects, the experiment was carried out by subjects who had never done a similar experiment.
— SUBJECT MOTIVATION. All the professors who were involved in this experiment participated voluntarily, in order to help us in our research. We motivated the students who participate in the experiment, by telling them that similar tasks to the experimental ones could be required in exams, so they wanted to make the most of the experiment.
— OTHER FACTORS. Plagiarism and influence between students could not really be controlled. Students were told that talking to each other about the experiment was forbidden, but they did the experiment on their own without any control, so we had to trust them as far as that was concerned.

From the results of the experiment we can conclude that there is empirical evidence of the existing relationship between the independent and the dependent variables. But only by replicating controlled experiments, where the measures would be varied in a controlled manner and all other factors kept constant, could we really demonstrate causality.

4.5.3. *Threats to external validity*

The external validity is the degree to which the results of the research can be generalised to the population under study and to other research settings. The greater the external validity, the more the results of an empirical study can be generalised to actual software engineering practice. Two threats of validity have been identified

which limit the ability to apply any such generalisation:

— MATERIALS AND TASKS USED. In the experiment we tried to use ERDs and tasks which can be representative of real problems, but more empirical studies taking "real cases" from software companies must be done.
— SUBJECTS. To solve the difficulty of obtaining professional subjects, we used professors and students of advanced software engineering courses. We are aware that more experiments with practitioners and professionals must be carried out in order to be able to generalise these results. However, in this case, the tasks to be performed do not require high levels of industrial experience, so, experiments with students could be appropriate [38].

In general, in order to extract a final conclusion we need to replicate this experiment with a greater number of subjects, including practitioners. After carrying out the replication we will have a cumulative body of knowledge; which will lead us to confirm if the presented metrics could really be used as early quality indicators, and could be used to predict ERD maintainability.

5. Conclusions and Future Trends

The practical contribution of this paper is that it provides practitioners with a set of objective and automatically computed metrics for assessing the structural complexity of ERDs in the early phases of an IS life cycle. The proposed metrics were theoretically validated following Briand *et al.*'s framework [26].

We have carried out a controlled experiment for ascertaining which metrics are relevant for each maintainability sub-characteristic, and for building a prediction model for them based on the values of proposed metrics. That prediction model, built following a method for induction of fuzzy rule systems, is highly qualitative and very closed to human minds, and above all very easy to understand.

Although the results obtained in the experiment are encouraging, we are aware that it is necessary to do further empirical validation. Case studies with enterprises must be carried out, with the goal of assessing these metrics as predictors of maintenance efforts, and therefore, determining whether they can be used as early quality indicators. As Basili *et al.* [38] said it is necessary to replicate experiments in order to build knowledge through experimentation. This knowledge may help designers to make better decisions in the process of conceptual modelling.

We cannot disregard the increasing diffusion of the object-oriented (OO) paradigm in conceptual modelling. We are also working on metrics for OMT class diagrams [54] and UML class diagrams [55] but it is necessary to do further validation of them. In our knowledge, few works have been done for measuring models that capture the dynamic aspects of an OO software systems [53, 56]. As is quoted in [57] this is an area which needs further investigation. Thus, as a future work we will define metrics for dynamic diagrams, such as state diagrams or activity diagrams.

Furthermore, it is necessary not only to address maintainability, but also to focus on measuring other quality factors, such as that proposed in ISO/IEC 9126 [28].

Acknowledgements

This research is part of the MANTICA project, partially supported by CICYT and the European Union (1FD1997- 0168TIC), the DOLMEN project supported by CICYT (TIC2000-1673-C06-06) and the CIPRESES project supported by CICYT (TIC 2000-1362-C02-02).

References

1. B. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.
2. J. Feng, "The 'Information Content' problem of a conceptual data schema and a possible solution", *Proc. 4th UKAIS Conference: Information Systems — The Next Generation*, University of York, 1999, pp. 257–266.
3. G. Shanks and P. Darke, "Quality in conceptual modelling: Linking theory and practice", *Proc. Pacific Asia Conf. on Information Systems* (*PACIS*'97), Brisbane, 1997, pp. 805–814.
4. P. Chen, "The entity-relationship model: Toward a unified view of data", *ACM Trans. on Database Systems* **1** (1976) 9–36.
5. T. Teorey, *Database Modeling and Design*, Third edition, Morgan Kaufmann, San Francisco, 1999.
6. R. Muller, *Database Design for Smarties: Using UML for Data Modeling*, Morgan Kaufman, 1999.
7. L. Moody and G. Shanks, "What makes a good data model? Evaluating the quality of entity relationships models", *Proc. 13th Int. Conf. on Conceptual Modelling* (*ER*'94), Manchester, England, December 14–17, 1994, pp. 94–111.
8. Y. Wand and R. Weber, "A model for systems decomposition", *Proc. AAANZ Conference*, Canberra, Australia, 1989.
9. C. Batini, S. Ceri and S. Navathe, *Conceptual Database Design: An Entity-Relationship Approach*, Benjamin-Cummings, 1992.
10. M. Reingruber and W. Gregory, *The Data Modelling Handbook: A Best-Practice Approach to Building Quality Data Models*, John Wiley, 1994.
11. M. Boman, J. Bubenko, P. Johannesson and B. Wangler, *Conceptual Modelling*, Prentice-Hall, 1997.
12. O. Lindland, G. Sindre and A. Solvberg, "Understanding quality in conceptual modelling", *IEEE Software* **11(2)** (1994) 42–49.
13. J. Krogstie, O. Lindland and G. Sindre, "Towards a deeper understanding of quality in requirements engineering", *Proc. 7th Int. Conf. on Advanced Information Systems Engineering* (*CAISE*), Jyvaskyla, Finland, June, 1995, pp. 82–95.
14. R. Schuette and T. Rotthowe, "The guidelines of modeling — An approach to enhance the quality in information models", *Proc. Seventeenth Int. Conf. on Conceptual Modelling* (*E/R* '98), Singapore, November 16–19, 1998, pp. 240–254.
15. L. Moody, G. Shanks and P. Darke, "Improving the quality of entity-relationship models — Experience in research and practice", *Proc. Seventeenth Int. Conf. on Conceptual Modelling* (*E/R* '98), Singapore, 1998, pp. 255–276.

16. M. Piattini, M. Genero, C. Calero, M. Polo and F. Ruiz, "Database quality", in Chap. 14, *Advanced Database Technology and Design*, eds. Mario Piattini and Oscar Díaz, Artech House, 2000.
17. A. Melton, *Software Measurement*, London, International Thomson Computer Press, 1996.
18. N. Fenton and S. Pfleeger, *Software Metrics: A Rigorous Approach*, 2nd Edition, Chapman & Hall, London, 1997.
19. B. Henderson-Sellers, *Object-Oriented Metrics — Measures of Complexity*, Prentice-Hall, New Jersey, 1996.
20. C. Eick, "A methodology for the design and transformation of conceptual schemas", *Proc. 17th Int. Conf. on Very Large Data Bases*, Barcelona, Spain, 1991.
21. R. Gray, B. Carey, N. McGlynn and A. Pengelly, "Design metrics for database systems", *BT Technology* **9(4)** (1991) 69–79.
22. S. Kesh, "Evaluating the quality of entity-relationship models", *Information and Software Technology* **37(12)** (1995) 681–689.
23. L. Moody, "Metrics for evaluating the quality of entity-relationship models", *Proc. Seventeenth Int. Conf. on Conceptual Modelling* (*E/R* '98), Singapore, November 16–19, 1998, pp. 213–225.
24. V. Basili and D. Weiss, "A methodology for collecting valid software engineering data", *IEEE Trans. on Software Engineering* **10** (1984) 728–738.
25. V. Basili and H. Rombach, "The TAME project: Towards improvement-oriented software environments", *IEEE Transactions on Software Engineering* **14** (1988) 758–773.
26. L. Briand, S. Morasca and V. Basili, "Property-based software engineering measurement", *IEEE Trans. on Software Engineering* **22(6)** (1996) 68–86.
27. J. McGregor, I.-H. Cho, B. Malloy and C. Hobatr, "Collecting metrics for CORBA-based distributed systems", *Empirical Software Engineering* **4** (1999) 217–240.
28. ISO/IEC 9126-1, Information Technology — Software Product Quality, Part 1: Quality Model, 1999.
29. T. Lethbridge, "Metrics for concept-oriented knowledge bases", *Int. J. of Software Engineering and Knowledge Engineering* **8(2)** (1998) 161–188.
30. Van Den Berg and Van Den Broek, "Axiomatic validation in the software metric development process", in Chap. 10: *Software Measurement*, ed. A. Melton, Thomson Computer Press, 1996.
31. H. Zuse, *A Framework of Software Measurement*, Berlin, Walter de Gruyter, 1998.
32. A. Melton, D. Gustafson, M. Bieman and L. Baker, "A mathematical perspective for software measures research", *IEEE Software Eng. J.* **5(5)** (1990) 246–254.
33. G. Poels and G. Dedene, *DISTANCE: A Framework for Software Measure Construction*, Research Report 9937, Department of Applied Economics, Katholieke Universiteit Leuven, 1999, 46 pp.
34. E. Weyuker, "Evaluating software complexity measures", *IEEE Trans. on Software Engineering* **14(9)** (1988) 1357–1365.
35. S. Morasca and L. Briand, "Towards a theoretical framework for measuring software attributes", *Proc. Fourth Int. Software Metrics Symposium*, 1997, pp. 119–126.
36. B. Kitchenham, S. Pflegger and N. Fenton, "Towards a framework for software measurement validation", *IEEE Transactions of Software Engineering* **21(12)** (1995) 929–943.
37. N. Schneidewind, "Methodology for validating software metrics", *IEEE Trans. on Software Engineering* **18(5)** (1992) 410–422.
38. V. Basili, F. Shull and F. Lanubile, "Building knowledge through families of experiments", *IEEE Transactions on Software Engineering* **25(4)** (1999) 435–437.

39. L. Briand, S. Arisholm, F. Counsell, F. Houdek and P. Thévenod-Fosse, "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions", Technical Report IESE 037.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1999.
40. L. Briand, C. Bunse and J. Daly, *A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs*, Technical Report IESE 002.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1999.
41. I. Ruiz and M. Gómez-Nieto, *Diseño y uso de Bases de Datos Relacionales*, Ra-Ma, 1997 (in Spanish).
42. A. De Miguel and M. Piattini, *Fundamentos y Modelos de Bases de Datos*, 2nd ed. Ra-Ma, 1999 (in Spanish).
43. S. Morasca and G. Ruhe, "Introduction: Knowledge discovery from empirical software engineering data", *Int. J. of Software Engineering and Knowledge Engineering* **9(5)** (1999) 495–498.
44. H. Sahraoui, M. Boukadoum and H. Lounis, "Using fuzzy threshold values for predicting class libraries interface evolution", *Proc. 4th Int. ECCOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, June 13, Cannes, France, 2000.
45. C. Ebert, "Rule-based fuzzy classification for software quality control", *Fuzzy Sets and Systems* **63** (1993) 349–358.
46. C. Ebert and E. Baisch, "Metrics for identifying critical components in software projects", in *Handbook of Software Engineering and Knowledge Engineering*, Vol. 1, 2001 (to appear).
47. L. Zadeh, "The concept of linguistic variable and its applications to approximate reasoning, Part I", *Information Sciences* **8** (1973) 199–249.
48. L. Breiman, J. Friedman, R. A. Olshen and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.
49. L. J. Linares, M. Delgado and A. Skarmeta, "Regression by fuzzy knowledge bases", *Proc. 4th European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, September, 1996, pp. 1170–1176.
50. M. Delgado, A. Gómez Skarmeta and L. Jiménez, *Int. J. of Intelligent Systems* **16** (2001).
51. L. Zadeh, "Fuzzy sets", *Information and Control* (1965) 338–353.
52. M. Sugeno, "An introductory survey of fuzzy control", *Information Sciences* **36** (1985) 59–83.
53. G. Poels and G. Dedene, "Measures for assessing dynamic complexity aspects of object-oriented conceptual schemes", in *Proc. 19th Int. Conf. on Conceptual Modeling* (*ER*2000), Salt Lake City, 2000, pp. 499–512.
54. M. Genero, M. E. Manso, M. Piattini and F. García, "Assessing the quality and the complexity of OMT models", *Proc. 2nd European Software Measurement Conference — FESMA 99*, Amsterdam, The Netherlands, 1999, pp. 99–109.
55. M. Genero, M. Piattini, and C. Calero, "Early measures for UML class diagrams", *L'Objet* **6(4)** (2001) 489–515.
56. G. Poels, "On the measurement of event-based object-oriented conceptual models", *Proc. 4th Int. ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, June 13, Cannes, France, 2000.
57. F. Brito e Abreu, H. Zuse, H. Sahraoui, and W. Melo, "Quantitative approaches in object-oriented software engineering", *Object-Oriented Technology: ECOOP'99*, Workshop Reader, Lecture Notes in Computer Science 1743, Springer-Verlag, 1999, pp. 326–337.

Appendix A

The following tables show the fuzzy models for the following maintainability sub-characteristics: simplicity (F2), analysability (F3), modifiability (F4), stability (F5), testability (F6).

Table 6. Simplicity fuzzy model.

RULE	NE	NA	NBinaryR	F2	ERROR	COV%
0	[6.0,9.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	3.70	0.1791	15.61
1	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	[MIN,MIN,2.0,3.0]	3.10	0.0619	4.37
2	[6.0,9.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	[2.0,3.0,3.0,5.0]	3.16	0.0083	0.00
3	[MIN,MIN,5.0,6.0]	[MIN,MIN,MAX,MAX]	[2.0,3.0,3.0,5.0]	2.66	0.1590	7.95
4	MIN,MIN,5.0,6.0]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	2.73	0.0639	5.86
5	[5.0,6.0,6.0,9.0]	[MIN,MIN,19.0,29.0]	[3.0,5.0,6.0,8.0]	2.83	0.1837	11.46
6	[5.0,6.0,6.0,9.0]	[19.0,29.0,MAX,MAX]	[3.0,5.0,6.0,8.0]	3.55	0.0909	8.42
7	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	[10.0,11.0,MAX,MAX]	4.21	0.1369	8.26
8	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	[8.0,10.0,10.0,11.0]	3.91	0.1005	6.01
9	[5.0,6.0,6.0,9.0]	[19.0,29.0,MAX,MAX]	[2.0,3.0,3.0,5.0]	3.40	0.0036	0.00
10	[5.0,6.0,6.0,9.0]	[MIN,MIN,12.0,15.0]	[2.0,3.0,3.0,5.0]	2.39	0.0043	0.00
11	[5.0,6.0,6.0,9.0]	[17.0,19.0,19.0,26.0]	[2.0,3.0,3.0,5.0]	2.94	0.0097	0.00
12	[5.0,6.0,6.0,9.0]	[12.0,15.0,17.0,19.0]	[2.0,3.0,3.0,5.0]	2.97	0.2080	13.76
13	[MIN,MIN,6.0,9.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,8.0,10.0]	3.77	0.0994	6.64
14	[6.0,9.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	[6.0,8.0,8.0,10.0]	4.22	0.1039	9.72

Table 7. Analysability fuzzy model.

RULE	NA	N1:NR	NBinaryR	F3	ERROR	COV%
0	[35.0,44.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.76	0.1123	8.68
1	[19.0,29.0,29.0,32.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.23	0.1740	13.79
2	[17.0,19.0,19.0,26.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.05	0.1766	11.83
3	[33.0,34.0,35.0,44.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.52	0.1358	8.80
4	[12.0,15.0,15.0,16.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	2.55	0.1293	9.38
5	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,MAX,MAX]	3.38	0.0128	0.00
6	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	3.64	0.1049	7.87
7	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,3.0,5.0]	2.64	0.1190	7.48
8	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,MAX,MAX]	3.41	0.0058	0.00
9	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	2.32	0.0214	2.26
10	[29.0,32.0,33.0,34.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,6.0,8.0]	3.74	0.0485	4.00
11	[29.0,32.0,33.0,34.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,MAX,MAX]	4.56	0.1336	10.58
12	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,2.0,3.0]	2.12	0.0019	0.00
13	[15.0,16.0,17.0,19.0]	[MIN,MIN,3.0,5.0]	[2.0,3.0,3.0,5.0]	2.82	0.1883	13.27
14	[15.0,16.0,17.0,19.0]	[3.0,5.0,MAX,MAX]	[2.0,3.0,3.0,5.0]	2.87	0.0056	0.00

Table 8. Modifiability fuzzy model.

RULE	NA	N1:NR	F4	ERROR	COV%
0	[35.0,44.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.81	0.0976	8.68
1	[12.0,15.0,15.0,16.0]	[MIN,MIN,MAX,MAX]	2.74	0.1151	9.38
2	[33.0,34.0,35.0,44.0]	[MIN,MIN,MAX,MAX]	3.76	0.0962	8.80
3	[MIN,MIN,12.0,15.0]	[3.0,5.0,MAX,MAX]	3.19	0.0169	0.00
4	[MIN,MIN,12.0,15.0]	[1.0,2.0,3.0,5.0]	2.86	0.1173	7.10
5	[MIN,MIN,12.0,15.0]	[MIN,MIN,1.0,2.0]	3.34	0.1243	8.09
6	[15.0,16.0,17.0,19.0]	[3.0,5.0,MAX,MAX]	3.29	0.0119	0.00
7	[15.0,16.0,17.0,19.0]	[1.0,2.0,3.0,5.0]	2.47	0.0883	7.32
8	[15.0,16.0,17.0,19.0]	[MIN,MIN,1.0,2.0]	3.11	0.1301	8.62
9	[29.0,32.0,33.0,34.0]	[MIN,MIN,3.0,5.0]	3.64	0.0097	0.00
10	[29.0,32.0,33.0,34.0]	[3.0,5.0,MAX,MAX]	4.38	0.1518	13.77
11	[19.0,29.0,29.0,32.0]	[MIN,MIN,3.0,5.0]	3.24	0.0747	7.17
12	[19.0,29.0,29.0,32.0]	[3.0,5.0,MAX,MAX]	3.83	0.0785	6.63
13	[17.0,19.0,19.0,26.0]	[MIN,MIN,3.0,5.0]	2.93	0.0915	8.93
14	[17.0,19.0,19.0,26.0]	[3.0,5.0,MAX,MAX]	3.77	0.0538	2.90

Table 9. Stability fuzzy model.

RULE	NA	N1:NR	NBinaryR	F5	ERROR	COV %
0	[17.0,19.0,19.0,26.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.36	0.1497	11.83
1	[35.0,44.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.79	0.0980	8.68
2	[19.0,29.0,29.0,32.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.42	0.1641	13.79
3	[12.0,15.0,15.0,16.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	2.83	0.1187	9.38
4	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,MAX,MAX]	3.75	0.0051	0.00
5	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[6.0,8.0,MAX,MAX]	3.53	0.0104	0.00
6	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	2.88	0.0369	2.26
7	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[3.0,5.0,6.0,8.0]	3.64	0.1043	7.87
8	[MIN,MIN,12.0,15.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,3.0,5.0]	3.12	0.1529	7.48
9	[29.0,32.0,33.0,34.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	4.29	0.1378	14.58
10	[33.0,34.0,35.0,44.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.80	0.0954	8.80
11	[15.0,16.0,17.0,19.0]	[MIN,MIN,MAX,MAX]	[MIN,MIN,2.0,3.0]	2.77	0.0052	0.00
12	[15.0,16.0,17.0,19.0]	[3.0,5.0,MAX,MAX]	[2.0,3.0,3.0,5.0]	3.41	0.0060	0.00
13	[15.0,16.0,17.0,19.0]	[1.0,2.0,3.0,5.0]	[2.0,3.0,3.0,5.0]	2.88	0.0734	5.19
14	[15.0,16.0,17.0,19.0]	[MIN,MIN,1.0,2.0]	[2.0,3.0,3.0,5.0]	3.74	0.1233	8.08

Table 10. Testability fuzzy model.

RULE	NA	N1:NR	F6	ERROR	COV%
0	[35.0,44.0,MAX,MAX]	[MIN,MIN,MAX,MAX]	3.75	0.0783	8.68
1	[12.0,15.0,15.0,16.0]	[MIN,MIN,MAX,MAX]	2.69	0.0914	9.38
2	[MIN,MIN,12.0,15.0]	[3.0,5.0,MAX,MAX]	3.21	0.0115	0.00
3	[33.0,34.0,35.0,44.0]	[MIN,MIN,MAX,MAX]	3.74	0.0848	8.80
4	[15.0,16.0,17.0,19.0]	[3.0,5.0,MAX,MAX]	3.37	0.0078	0.00
5	[MIN,MIN,12.0,15.0]	[1.0,2.0,3.0,5.0]	2.92	0.0905	7.10
6	[MIN,MIN,12.0,15.0]	[MIN,MIN,1.0,2.0]	3.52	0.1102	8.09
7	[15.0,16.0,17.0,19.0]	[1.0,2.0,3.0,5.0]	2.49	0.0661	7.32
8	[15.0,16.0,17.0,19.0]	[MIN,MIN,1.0,2.0]	3.31	0.1149	8.62
9	[19.0,29.0,29.0,32.0]	[MIN,MIN,3.0,5.0]	3.12	0.0611	7.17
10	[19.0,29.0,29.0,32.0]	[3.0,5.0,MAX,MAX]	3.62	0.0620	6.63
11	[17.0,19.0,19.0,26.0]	[MIN,MIN,3.0,5.0]	3.07	0.0866	8.93
12	[17.0,19.0,19.0,26.0]	[3.0,5.0,MAX,MAX]	3.76	0.0252	2.90
13	[29.0,32.0,33.0,34.0]	[MIN,MIN,3.0,5.0]	3.65	0.0079	0.00
14	[29.0,32.0,33.0,34.0]	[3.0,5.0,MAX,MAX]	4.40	0.1047	13.77