

Metrics for Evaluating the Quality of Entity Relationship Models

Daniel L. Moody

Simsion Bowles and Associates,
1 Collins St., Melbourne, Australia 3000.

email: dmoody@acm.org

Abstract. This paper defines a comprehensive set of metrics for evaluating the quality of Entity Relationship models. This is an extension of previous research which developed a conceptual framework and identified stakeholders and quality factors for evaluating data models. However quality factors are not enough to ensure quality in practice, because different people will have different interpretations of the same concept. The objective of this paper is to refine these quality factors into quantitative measures to reduce subjectivity and bias in the evaluation process. A total of twenty five candidate metrics are proposed in this paper, each of which measures one of the quality factors previously defined. The metrics may be used to evaluate the quality of data models, choose between alternatives and identify areas for improvement.

1 Introduction

The choice of an appropriate representation of data is one of the most crucial tasks in the entire systems development process. Although the data modelling phase represents only a small proportion of the total systems development effort, its impact on the final result is probably greater than any other phase (Simsion, 1994). The data model is a major determinant of system development costs (ASMA, 1996), system flexibility (Gartner, 1992), integration with other systems (Moody and Simsion, 1995) and the ability of the system to meet user requirements (Batini et al., 1992). For this reason, effort expended on improving the quality of data models is likely to pay off many times over in later phases.

Previous Research

Evaluating the quality of data models is a discipline which is only just beginning to emerge. Quantitative measurement of quality is almost non-existent. A number of frameworks for evaluating the quality of data models have now been proposed in the literature (Roman, 1985; Mayer, 1989; von Halle, 1991; Batini et al., 1992; Levitin and Redman, 1994; Simsion, 1994; Moody and Shanks, 1994; Krogstie, Lindland and Sindre, 1995; Lindland, Sindre and Solveberg, 1994; Kesh, 1995; Moody and Shanks, 1998).

Most of these frameworks suggest criteria that may be used to evaluate the quality of data models. However quality criteria are not enough on their own to ensure quality in practice, because different people will generally have different interpretations of what they mean. According to the Total Quality Management (TQM) literature, measurable criteria for assessing quality are necessary to avoid “arguments of style” (Zultner, 1992). The objective should be to replace intuitive notions of design “quality” with

formal, quantitative measures to reduce subjectivity and bias in the evaluation process. However developing reliable and objective measures of quality in software development is a difficult task. As Van Vliet (1993) says:

“The various factors that relate to software quality are hard to define. It is even harder to measure them quantitatively. There are very few quality factors or criteria for which sufficiently sound numeric measures exist.”

Of the frameworks that have been proposed, only two address the issue of quality measurement. Moody and Shanks (1994) suggest a number of evaluation methods, which in some cases are measures (eg. data model complexity) and in other cases are processes for carrying out the evaluation (eg. user reviews). Kesh (1995) defines a number of metrics for evaluating data models but these are theoretically based, and of limited use in practice. Most of the other frameworks rely on experts giving overall subjective ratings of the quality of a data model with respect to the criteria proposed.

2 A Framework for Evaluating and Improving the Quality of Data Models

This paper uses the framework for data model evaluation and improvement proposed by Moody and Shanks (1998) as a basis for developing quality metrics. An earlier version of the framework was published in Moody and Shanks (1994). This framework was developed in practice, and has now been applied in a wide range of organisations around the world (Moody, Shanks and Darke, 1998). The framework is summarised by the Entity Relationship model shown below.

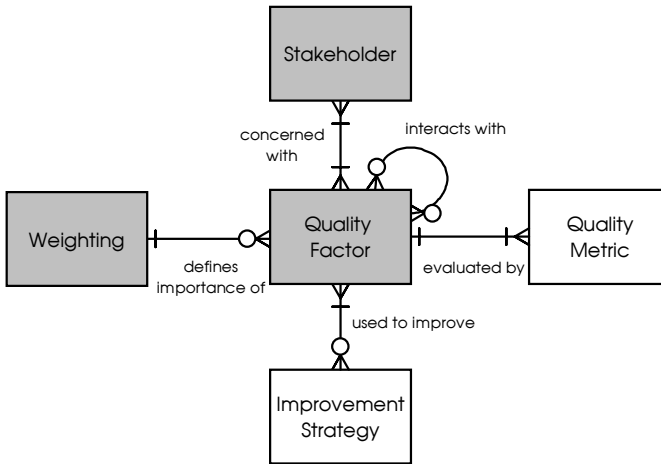


Fig. 1. Data Model Quality Evaluation Framework

- Quality factors are the properties of a data model that contribute to its quality. These answer the question: “What makes a good data model?”. A particular quality factor may have positive or negative interactions with other quality factors—these represent the trade-offs implicit in the modelling process.

- Stakeholders are people who are involved in building or using the data model, and therefore have an interest in its quality. Different stakeholders will generally be interested in different quality factors.
- Quality metrics are define ways of evaluating particular quality factors. There may be multiple measures for each quality factor.
- Weightings define the relative importance of different quality factors in a problem situation. These are used to make trade-offs between different quality factors.
- Improvement strategies are techniques for improving the quality of data models with respect to one or more quality factors.

A previous paper (Moody and Shanks, 1994) defined the stakeholders and quality factors relevant to data modelling, as well as methods for evaluating the quality of data models. This paper defines metrics for each quality factor.

Stakeholders

The key stakeholders in the data modelling process are:

- The business user, whose requirements are defined by the data model
- The analyst, who is responsible for developing the data model
- The data administrator, who is responsible for ensuring that the data model is consistent with the rest of the organisation's data
- The application developer, who is responsible for implementing the data model (translating it into a physical database schema)

Quality Factors

The proposed quality factors and the primary stakeholders involved in evaluating them are shown in Fig. 2 below.

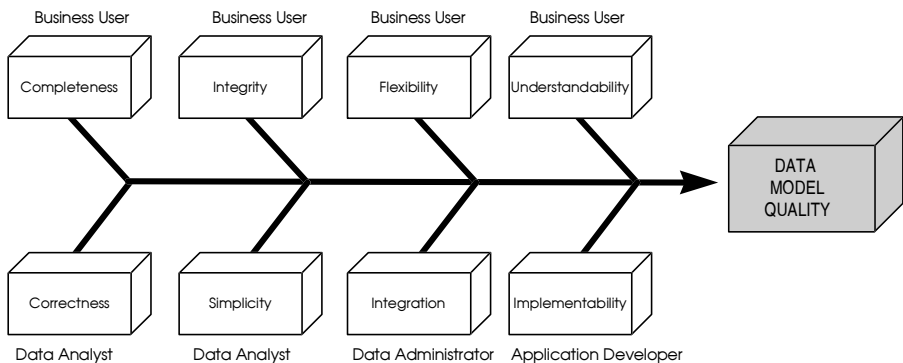


Fig. 2. Data Model Quality Factors

These quality factors may be used as criteria for evaluating the quality of individual data models or comparing alternative representations of requirements. Together they incorporate the needs of all stakeholders, and represent a complete picture of data model quality. The following sections define quality measures for each quality factor.

3 Completeness

Completeness relates to whether the data model contains all information required to meet user requirements. This corresponds to one half of the 100% principle—that the conceptual schema should define all static aspects of the Universe of Discourse (ISO, 1987). Completeness is the most important requirement of all because if it is not satisfied, none of the other quality factors matter. If the requirements as expressed in the data model are inaccurate or incomplete, the system which results will not satisfy users, no matter how well designed or implemented it is.

Evaluating Completeness

In principle, completeness can be checked by checking that each user requirement is represented somewhere in the model, and that each element of the model corresponds to a user requirement (Batini et al, 1992). However the practical difficulty with this is that there is no external source of user requirements—they exist only in people’s minds. Completeness can therefore only be evaluated with close participation of business users.

The result of completeness reviews will be a list of elements (entities, relationships, attributes, business rules) that do not match user requirements. Fig. 3 illustrates the different types of completeness mismatches:

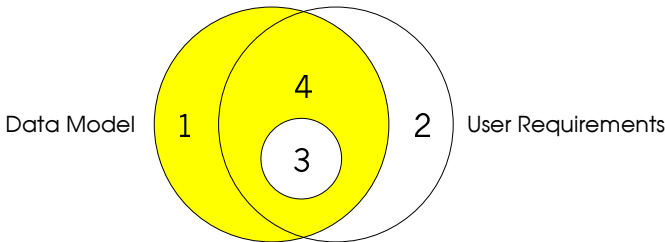






Fig. 3. Types of Completeness Errors

- Area 1 represents elements included in the data model that do not correspond to any user requirement or are out of scope of the system—these represent unnecessary elements. We call these *Type 1* errors.
- Area 2 represents user requirements which are not represented anywhere in the data model—these represent gaps or omissions in the model. We call these *Type 2* errors.
- Area 3 represents items included in the data model that correspond to user requirements but have been inaccurately defined. We call these *Type 3* errors.
- Area 4 represents elements in the data model that accurately correspond to user requirements.

The objective of completeness reviews is to eliminate all items of type 1, 2 and 3.

Proposed Completeness Metrics

The proposed quality measures for correctness all take the form of mismatches with respect to user requirements. The purpose of the review process will be to eliminate all such defects, so that the model exactly matches user requirements:

-  Metric 1. Number of items in the data model that do not correspond to user requirements (Type 1 errors). Inclusion of such items will lead to unnecessary development effort and added cost.
-  Metric 2. Number of user requirements which are not represented in the data model (Type 2 errors). These represent missing requirements, and will need to be added later in the development lifecycle, leading to increased costs, or if they go undetected, will result in users not being satisfied with the system
-  Metric 3. Number of items in the data model that correspond to user requirements but are inaccurately defined (Type 3 errors). Such items will need to be changed later on the development lifecycle, leading to rework and added cost, or if they go undetected, will result in users being unsatisfied with the system.
-  Metric 4. Number of inconsistencies with process model. A critical task in verifying the completeness of the data model is to map it against the business processes which the system needs to support. This ensures that all functional requirements can be met by the model. The result of this analysis can be presented in the form of a CRUD (Create, Read, Update, Delete) matrix. Analysis of the CRUD matrix can be used to identify gaps in the data model as well as to "prune away" unnecessary data from the model (Martin, 1989).

4 Integrity

Integrity is defined as the extent to which the business rules (or integrity constraints) which apply to the data are enforced by the data model¹. Integrity corresponds to the other half of the 100% principle—that the conceptual schema should define all dynamic aspects of the Universe of Discourse (ISO, 1987). Business rules define what can and can't happen to the data. Business rules are necessary to maintain the consistency and integrity of data stored, as well as to enforce business policies (Date, 1989; Loffman and Rush, 1991). All rules which apply to the data should be documented in the data model to ensure they are enforced consistently across all application programs (ISO, 1987).

Evaluating Integrity



Like completeness, integrity can only really be evaluated with close participation of business users. The rules represented by the data model may be verified by translating them into natural language sentences. Users can then verify whether each rule is true

¹ In the original version of the evaluation framework (Moody and Shanks, 1994) integrity was included as part of completeness, but has since been separated out as a quality factor in its own right.

or false. This is useful as a check on the integrity of the data model because business users often have difficulty understanding the constraints defined in data models, particularly cardinality rules on relationships (Batini et al., 1992). Many CASE tools can automatically translate relationship cardinality rules into natural language sentences, provided relationships have been named correctly.

Proposed Integrity Metrics

The proposed quality measures for integrity take the form of mismatches between the data model and business policies. The purpose of the review process will be to eliminate all such defects:

-  Metric 5. Number of business rules which are not enforced by the data model. Non-enforcement of these rules will result in data integrity problems and/or operational errors.
-  Metric 6. Number of integrity constraints included in the data model that do not accurately correspond to business policies (i.e. which are false). Incorrect integrity constraints may be further classified as:
 - too *weak*: the rule allows invalid data to be stored
 - too *strong*: the rule does not allow valid data to be stored and will lead to constraints on business operations and the need for user “workarounds”.

5 Flexibility

Flexibility is defined as the ease with which the data model can cope with business change. The objective is for additions and/or changes in requirements to be handled with the minimum possible change to the data model. The data model is a key contributor to the flexibility of the system as a whole (Gartner, 1992; Simson, 1994). Lack of flexibility in the data model can lead to:




- Maintenance costs: of all types of maintenance changes, changes to data structures and formats are the most expensive. This is because each such change has a “ripple effect” on all the programs that use it.
- Reduced organisational responsiveness: inflexible systems inhibit changes to business practices, organisational growth and the ability to respond quickly to business or regulatory change. Often the major constraint on introducing business change—for example, bringing a new product to market—is the need to modify the computer systems that support it (Simson, 1988).

Evaluating Flexibility

Flexibility is a particularly difficult quality factor to assess because of the inherent difficulty of predicting what might happen in the future. Evaluation of flexibility requires identifying what requirements might change in the future, their probability of occurrence and their impact on the data model. However no matter how much time spent thinking about what might happen in the future, such changes remain hard to anticipate. In this respect, evaluating flexibility has much in common with weather forecasting—there is a limit to how far and how accurately the future can be predicted.

Proposed Flexibility Metrics

The proposed measures for evaluating flexibility focus on areas where the model is potentially unstable—where changes to the model might be required in the future as a result of changes in the business environment. The purpose of the review process will be to look at ways of minimising impact of change on the model, taking into account the probability of change, strategic impact and likely cost of change. A particular focus of flexibility reviews is identifying business rules which might change.

-  Metric 7. Number of elements in the model which are subject to change in the future. This includes changes in definitions or business rules as a result of business or regulatory change.
-  Metric 8. Estimated cost of changes. For each possible change, the probability of change occurring and the estimated cost of making the change post-implementation should be used to calculate the probability-adjusted cost of the change.
-  Metric 9. Strategic importance of changes. For each possible change, the strategic impact of the change should be defined, expressed as a rating by business users of the need to respond quickly to the change.

6 Understandability

Understandability is defined as the ease with which the data model can be understood. Business users must be able to understand the model in order to verify that it meets their requirements. Similarly, application developers need to be able to understand the model to implement it correctly. Understandability is also important in terms of the usability of the system. If users have trouble understanding the concepts in the data model, they are also likely to have difficulty understanding the system which is produced as a result.




The communication properties of the data model are critical to the success of the modelling effort. However empirical studies show that in practice data models are poorly understood by users, and in most cases are not developed with direct user involvement (Hitchman, 1995). While data modelling has proven very effective as a technique for database design, it has been far less effective for communication with users (Moody, 1996a).

Evaluating Understandability

Understandability can only be evaluated with close participation with the users of the model—business users and application developers. In principle, understandability can be checked by checking that each element of the model is understandable. However the practical difficulty with this is that users may think they understand the model while not understanding its full implications and possible limitations from a business perspective.

Proposed Understandability Metrics

The proposed measures for understandability take the form of ratings by different stakeholders and tests of understanding. The purpose of the review process will be to maximise these ratings.

-  Metric 10. User rating of understandability of model: user ratings of understandability will be largely based on the concepts, names and definitions used, as well as how the model is presented. A danger with this metric is that it is common for users to grasp familiar business terms without appreciating the meaning represented in the model. As a result, they may *think* they understand the model while not really understanding its full implications for the business.
-  Metric 11. Ability of users to interpret the model correctly. This can be measured by getting users to instantiate the model using actual business examples (scenarios). Their level of understanding can then be measured by the number of errors in populating the model. This is a better operational test of understanding than the previous metric—it measures whether the model is actually *understood* rather than whether it is *understandable* (Lindland et al, 1994). This is much more important from the point of view of verifying the accuracy of the model.
-  Metric 12. Application developer rating of understandability. It is essential that the application developer understands the model fully so that they can implement it correctly. Getting the application developer to review the model for understandability is particularly useful for identifying where the model is unclear or ambiguous because they will be less familiar with the model and the business domain than either the analyst or business user. Many things that seem obvious to those involved in developing the model may not be to someone seeing it for the first time.

7 Correctness


Correctness refers to whether the model conforms to the rules of the data modelling technique being used. Rules of correctness include diagramming conventions, naming rules, definition rules and rules of composition (for example, each entity must have a primary key). Correctness is concerned only with whether the data modelling technique has been used correctly (syntactic or grammatical correctness). It answers the question: “Is this a valid model?”. Another important aspect of correctness, and a major focus of data modelling in practice, is to ensure that the model contains no redundancy—that each fact is represented in only one place (Simsion, 1994).

Evaluating Correctness


Correctness is the easiest of all the quality factors to evaluate, because there is very little subjectivity involved, and no degrees of quality—the model either obeys the rules or it does not. Also, the model can be evaluated in isolation, without reference to user requirements. The result of correctness reviews will be a list of defects, defining where the data model does not conform to the rules of the data modelling technique. Many of these checks can be carried out automatically using CASE tools.

Proposed Correctness Metrics


The proposed quality measures for correctness all take the form of defects with respect to data modelling standards (syntactic rules). We break down correctness errors into different types or *defect classes* to assist in identifying patterns of errors or problem areas which may be addressed by training or other process measures. The purpose of the review process will be to eliminate all such defects:

 Metric 13. Number of violations to data modelling conventions. These can be further broken down into the following defect classes:

- Diagramming standards violations (eg. relationships not named)
- Naming standards violations (eg. use of plural nouns as entity names)
- Invalid primary keys (non unique, incomplete or non-singular)
- Invalid use of constructs (eg. entities without attributes, overlapping subtypes, many to many relationships)
- Incomplete definition of constructs (e.g. data type and format not defined for an attribute; missing or inadequate entity definition)

 Metric 14. Number of normal form violations. Second and higher normal form violations identify redundancy among attributes within an entity (*intra-entity redundancy*). Normal form violations may be further classified into:

- First normal form (1NF) violations
- Second normal form (2NF) violations
- Third normal form (3NF) violations
- Higher normal form (4NF+) violations

 Metric 15. Number of instances of redundancy between entities—for example, where two entity definitions overlap or where redundant relationships are included. This is called *inter-entity redundancy*, to distinguish this from redundancy within an entity (*intra-entity redundancy*—Metric 14) and redundancy of data with other systems (*external redundancy*—Metric 21). Fig. 4 summarises the types of redundancy and corresponding metrics.

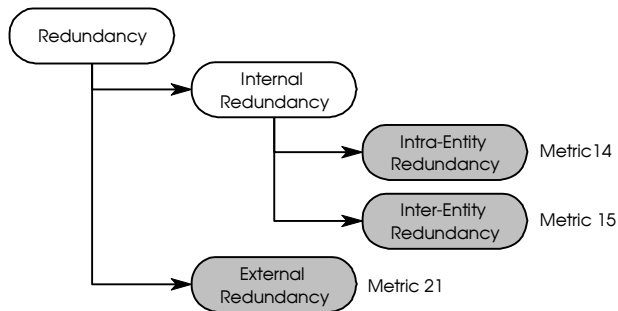


Fig. 4. Classification of Redundancy

8 Simplicity




Simplicity means that the data model contains the minimum possible constructs. Simpler models are more flexible (Meyer, 1988), easier to implement (Simsion, 1991), and easier to understand (Moody, 1997). The choice of simplicity as a quality factor is based on the principle of Ockham's Razor, which has become one of the cornerstones of the scientific method. This says that if there are two theories which explain the same observations, the one with the fewer constructs should be preferred (Dubin, 1979). The extension of this to data modelling is if there are two data models which meet the same requirements, the simpler one should be preferred.

Evaluating Simplicity

Simplicity is the easiest of all quality factors to evaluate, because it only requires a simple count of data model elements. This can be done automatically by CASE tools, or carried out manually. It takes no skill (apart from the ability to count!) and is totally objective. Simplicity metrics are particularly useful in comparing alternative data models—all other things being equal, the simpler one should be preferred.

Proposed Simplicity Metrics

Metrics for evaluating simplicity take the form of complexity measures. The purpose of the review process will be to minimise the complexity of the model while still satisfying user requirements. The following metrics represent alternative ways of measuring complexity of a data model—Metric 17 is recommended as the most useful of the measures proposed:

-  Metric 16. Number of entities (E). This is the simplest measure of the complexity of a data model. The justification for this is that the number of entities in the logical data model is a surrogate measure for system complexity and development effort. Symons (1988, 1991) found that in sizing of business (“data rich”) applications, the major determinant of software size (and development effort) was the number of entities.
-  Metric 17. Number of entities and relationships (E+R). This is a finer resolution complexity measure which is calculated as the number of entities (E) plus the number of relationships (R) in the data model. This derives from complexity theory, which asserts that the complexity of any system is defined by the number of components in the system and the number of relationships between them (Klir, 1985; Pippenger, 1978). Subtypes should *not* be included in the calculation of the number of entities because these represent subcategories within a single construct and generally do not translate into separate database tables. In addition, many to many relationships should be counted as three constructs, since when they are resolved, they will form one entity and two relationships (Shanks, 1997). This helps to standardise differences between different modelling styles.
-  Metric 18. Number of constructs (E+R+A). This is the finest resolution complexity measure, and includes the number of attributes in the calculation of data

model complexity. Such a metric could be calculated as a weighted sum of the form $aN^E + bN^R + cN^A$ where N^E is the number of entities, N^R is the number of relationships and N^A is the number of attributes. In practice however, such a measure does not provide any better information than Metric 17.

9 Integration

Integration is defined as the level of consistency of the data model with the rest of the organisation's data. In practice, application systems are often built in relative isolation of each other, leading to the same data being implemented over and over again in different ways. This leads to duplication of data, interface problems and difficulties consolidating data from different systems for management reporting (Moody and Simson, 1995). The primary mechanism for achieving corporate-wide data integration is a *corporate data model* (Goodhue et al., 1992). The corporate data model provides a common set of data definitions which is used to co-ordinate the activities of application development teams so that separately developed systems work together. The corporate data model allows opportunities for sharing of data to be identified, and ensures that different systems use consistent data naming and formats (Martin, 1989).

Evaluating Integration

Integration is assessed by comparing the application data model with the corporate data model (Batini, Lenzerini and Navathe, 1986). The result of this will be a list of *conflicts* between the project data model and the corporate data model. This is usually the responsibility of the data administrator (also called information architect, data architect, data manager), who has responsibility for corporate-wide sharing and integration of data. It is their role to maintain the corporate data model and review application data models for conformance to the corporate model.

Proposed Integration Metrics

Most of the proposed measures for integration are in the form of conflicts with the corporate data model or with existing systems. The purpose of the review process will be to resolve these inconsistencies.




Metric 19. Number of data conflicts with the Corporate Data Model. These can be further classified into:


- Entity conflicts: number of entities whose definitions are inconsistent with the definition entities in the corporate data model.
- Data element conflicts: number of attributes with different definitions or domains to corresponding attributes defined in the corporate data model.
- Naming conflicts: number of entities or attributes with the same business meaning but different names to concepts in the corporate data model (*synonyms*). Also entities or attributes with the same name but different meaning to concepts in the corporate data model (*homonyms*).



Metric 20. Number of data conflicts with existing systems. These can be further classified into:

- Number of data elements whose definitions conflict with those in existing systems e.g. different data formats or definitions. Inconsistent data item definitions will lead to interface problems, the need for data translation and difficulties comparing and consolidating data across systems.
- Number of key conflicts with existing systems or other projects. Key conflicts occur when different identifiers are assigned to the same object (eg. a particular customer) by different systems. This leads to *fragmentation* of data across systems and the inability to link or consolidate data about a particular entity across systems.
- Number of naming conflicts with other systems (synonyms and/or homonyms). These are less of a problem in practice than other data conflicts, but are a frequent source of confusion in system maintenance and interpretation of data.

 Metric 21. Number of data elements which duplicate data elements stored in existing systems or other projects. This is called *external redundancy* to distinguish it from redundancy within the model itself (Metrics 14 and 15). This form of redundancy is a serious problem in most organisations—empirical studies show that there are an average of ten physical copies of each primary data item in medium to large organisations (O'Brien and O'Brien, 1994).

 Metric 22. Rating by representatives of other business areas as to whether the data has been defined in a way which meets corporate needs rather than the requirements of the application being developed. Because all data is potentially shareable, all views of the data should be considered when the data is first defined (Thompson, 1993). In practice, this can be done by a high level committee which reviews all application development projects for data sharing, consistency and integration (Moody, 1996b).

10 Implementability




Implementability is defined as the ease with which the data model can be implemented within the time, budget and technology constraints of the project. While it is important that a data model does not contain any assumptions about the implementation (ISO, 1987), it is also important that it does not ignore all practical considerations. After all, there is little point developing a model which cannot be implemented or that the user cannot afford.

Evaluating Implementability

The implementability of the data model is assessed by the application developer, who is responsible for implementing the data model once it has been completed. The application developer provides an important “reality check” on what is technically possible and/or economically feasible. The process of reviewing the model also allows the application developer to gain familiarity with the model prior to the design stage to ensure a smooth transition.

Proposed Implementability Metrics

Proposed measures of implementability all take the form of ratings by the application developer. The purpose of the review process will be to minimise these ratings:

-  Metric 23. Technical risk rating: estimate of the probability that the system can meet performance requirements based on the proposed data model and the technological platform (particularly the target DBMS) being used
-  Metric 24. Schedule risk rating: estimate of the probability that the system can be implemented on time, based on the proposed data model
-  Metric 25. Development cost estimate: this is an estimate of the development cost of the system, based on the data model. Such an estimate will necessarily be approximate but will be useful as a guide for making cost/quality trade-offs between different models proposed. If the quote is too high (exceeds available budget), the model may need to be simplified, reduced in scope or the budget increased.

11 Conclusion

This paper has proposed a comprehensive set of metrics for evaluating the quality of data models based on the set of quality factors proposed by Moody and Shanks (1998). A total of twenty five candidate metrics are identified, with eighteen secondary metrics which may be used to classify defects in more detail. It is not expected that all of these metrics would be used in evaluating the quality of a particular data model. Our aim in this paper has been to be as complete as possible—to suggest as many metrics as possible as a starting point for analysis. Selection of the most appropriate metrics should be made based on their perceived usefulness and ease of calculation.

Further Research

The next step in this research is to validate and refine these metrics in practice. This will help to identify which metrics are most useful. It is proposed to use *action research* as the research paradigm for doing this. Action research (Checkland and Scholes, 1990) is a research method in which practitioners and researchers work together to test and refine principles, tools, techniques and methodologies that have been developed to address real world problems. It provides the ability to test out new methods in practice for mutual benefit of researchers and practitioners. Moody, Shanks and Darke, 1998 (in this conference) describe how action research has already been used to validate the framework and the quality factors proposed.

Further research is also required to develop strategies for improving the quality of data models once a quality problem has been identified. Definition of improvement strategies would complete the specification of the framework.

References

1. AUSTRALIAN SOFTWARE METRICS ASSOCIATION (ASMA) (1996): *ASMA Project Database*, Release 7, November, P.O. Box 1287, Box Hill, Victoria, Australia, 3128.
2. BATINI, C., CERI, S. AND NAVATHE, S.B. (1992): *Conceptual Database Design: An Entity Relationship Approach*, Benjamin Cummings, Redwood City, California.
3. BATINI, C., LENZERINI, M. AND NAVATHE, S. (1986): A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, 18(4), December: 323-364.
4. CHECKLAND, P.B. and SCHOLLES, J., *Soft Systems Methodology in Action*, Wiley, Chichester, 1990.
5. DATE, C.J. (1989):, *Introduction to Database Systems (4th Edition)*, Addison Wesley.
6. DUBIN, R. (1978): *Theory Building*, The Free Press, New York.
7. GARTNER RESEARCH GROUP (1992): "Sometimes You Gotta Break the Rules", *Gartner Group Strategic Management Series Key Issues*, November 23.
8. GOODHUE, D.L., KIRSCH, L.J., AND WYBO, M.D. (1992): The Impact of Data Integration on the Costs and Benefits of Information Systems, *MIS Quarterly*, 16(3), September: 293-311.
9. HITCHMAN, S. (1995) Practitioner Perceptions On The Use Of Some Semantic Concepts In The Entity Relationship Model, *European Journal of Information Systems*, 4, 31-40.
10. INTERNATIONAL STANDARDS ORGANISATION (ISO) (1987), *Information Processing Systems - Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO Technical Report 9007.
11. KESH, S. (1995): Evaluating the Quality of Entity Relationship Models, *Information and Software Technology*, 37 (12).
12. KLIR, G.J. (1985): *Architecture of Systems Problem Solving*, Plenum Press, New York.
13. KROGSTIE, J., LINDLAND, O.I. and SINDRE, G. (1995): Towards a Deeper Understanding of Quality in Requirements Engineering, *Proceedings of the 7th International Conference on Advanced Information Systems Engineering (CAISE)*, Jyvaskyla, Finland, June.
14. LEVITIN, A. and REDMAN, T. (1994): Quality Dimensions of a Conceptual View, *Information Processing and Management*, Volume 31.
15. LINDLAND, O.I, SINDRE, G. and SOLVEBERG, A. (1994): Understanding Quality in Conceptual Modelling, *IEEE Software*, March.
16. LOFFMAN, R.S. AND RUSH, R.M (1991): Improving Data Quality, *Database Programming and Design*, 4(4), April, 17-19.
17. MARTIN, J. (1989): *Strategic Data Planning Methodologies*, Prentice Hall, New Jersey.
18. MAYER, R.E. (1989): Models for Understanding, *Review of Educational Research*, Spring.
19. MEYER, B. (1988): *Object Oriented Software Construction*, Prentice Hall, New York.
20. MOODY, D.L. AND SHANKS, G.G. (1994): What Makes A Good Data Model? Evaluating the Quality of Entity Relationship Models, in P. LOUCOPOLIS (ed.) *Proceedings of the Thirteenth International Conference on the Entity Relationship Approach*, Manchester, December 14-17, 94-111.

21. MOODY, D.L. AND SIMSION, G.C. (1995): Justifying Investment in Information Resource Management, *Australian Journal of Information Systems*, 3(1), September: 25-37.
22. MOODY, D.L. (1996a) "Graphical Entity Relationship Models: Towards A More User Understandable Representation of Data", in B. THALHEIM (ed.) *Proceedings of the Fourteenth International Conference on the Entity Relationship Approach*, Cottbus, Germany, October 7-9, 227-244.
23. MOODY, D.L. (1996b) Critical Success Factors for Information Resource Management, *Proc. 7th Australasian Conference on Information Systems*, Hobart, Australia, December.
24. MOODY, D.L. (1997): "A Multi-Level Architecture for Representing Enterprise Data Models", *Proceedings of the Sixteenth International Conference on the Entity Relationship Approach*, Los Angeles, November 1-3.
25. MOODY, D.L. and SHANKS, G.G. (1998): What Makes A Good Data Model? A Framework for Evaluating and Improving the Quality of Entity Relationship Models, *Australian Computer Journal* (forthcoming).
26. MOODY, D.L., SHANKS, G.G. and DARKE, P. (1998): Improving the Quality of Entity Relationship Models—Experience in Research and Practice, in *Proceedings of the Seventeenth International Conference on Conceptual Modelling (ER '98)*, Singapore, November 16—19.
27. O'BRIEN, C. AND O'BRIEN, S. (1994), "Mining Your Legacy Systems: A Data-Based Approach", *Asia Pacific DB2 User Group Conference*, Melbourne, Australia, November 21-23.
28. PIPPENGER, N. (1978): Complexity Theory, *Scientific American*, 238(6): 1-15.
29. ROMAN, G. (1985): A Taxonomy of Current Issues in Requirements Engineering, *IEEE Computer*, April.
30. SHANKS, G.G. (1997) Conceptual Data Modelling: An Empirical Study of Expert and Novice Data Modellers, *Australian Journal of Information Systems*, 4:2, 63-73
31. SIMSION, G.C. (1988): Data Planning in a Volatile Business Environment, Australian Computer Society Conference on Strategic Planning for Information Technology, Ballarat, March: 88-92.
32. SIMSION, G.C. (1991): Creative Data Modelling, *Proceedings of the Tenth International Entity Relationship Conference*, San Francisco, 112-123.
33. SIMSION, G.C. (1994): *Data Modelling Essentials*, Van Nostrand Reinhold, New York.
34. SYMONS, C.R. (1988): Function Point Analysis: Difficulties and Improvements, *IEEE Transactions on Software Engineering*, 14(1), January.
35. SYMONS, C.R. (1991): *Software Sizing and Estimating: MkII Function Point Analysis*, J. Wiley and Sons.
36. THOMPSON, C. (1993): "Living with an Enterprise Model", *Database Programming and Design*, 6(12), March: 32-38.
37. VAN VLIET, J.C. (1993): *Software Engineering: Principles and Practice*, John Wiley and Sons, Chichester, England.
38. VON HALLE, B. (1991): Data: Asset or Liability?, *Database Programming and Design*, 4(7), July: 13-15.
39. ZULTNER, R.E. (1992): "The Deming Way: Total Quality Management for Software", *Proceedings of Total Quality Management for Software Conference*, April, Washington, DC, April, 134-145.