Data model Maintainability

A comparative study of maintainability metrics

Thesis

Master

Software Engineering

ing. Maarten Wullink

UNIVERSITEIT VAN AMSTERDAM

Thesis supervisor: Dr. J. Vinju

SIDN

Stichting Internet Domeinregistratie Nederland

Supervisor: C. Toet

Availability: Public
Date: July 16, 2010

Preface

This thesis is the conclusion of the master program Software Engineering at the University of Amsterdam. The research has been performed at the Stichting Internet Domeinregistratie Nederland (SIDN) located in Arnhem. SIDN, founded in 1996 is responsible for the delegation of the .nl country code top-level domain (ccTLD) . The research has been done at the request of the IT department of SIDN.

Acknowledgements

The successful completion of this research project and the resulting thesis was made possible by the support of a number of people. I would like to thank my thesis supervisor Jurgen Vinju, his guidance and support during this project was of great help. My employer SIDN and specifically Cees Toet for providing me with the opportunity to complete this master program. Finally I would like to thank my colleague Krista Vermeulen for her help with reviewing this thesis.

Maarten Wullink

Arnhem, July 16, 2010

Abstract

Data models are found at the heart of almost every software system, they form the foundation of a software system. It is important to know if the foundation is able to support the weight of the software system which is built on top of it. If there are any weak spots or cracks in the foundation they must be detected and improved.

Most of the cost in the software lifecycle are made in the post delivery phase, when the software product must be maintained. Software which is hard to maintain is more expensive to maintain and to extend. It takes the maintainers more time to get to know the inner workings of the software system. Too much complexity makes it hard to extend and improve an existing software system. The earlier in the software life cycle at which problems are detected the cheaper it will be to fix these problems.

If the foundation of a software system is complex then this will negatively affect the software system which is built on top of the foundation.

This research looks at methods which can be used to objectively measure the maintainability of data models. For this purpose a number of existing proposals for data model evaluation methods have been used. Some methods measure more than maintainability alone. The ISO9126 standard was used to determine which elements of a measurement method affect maintainability.

Three existing methods and a new method proposed in this thesis have been used to determine the maintainability of two related data models. The new method was added because the existing methods are not able to pinpoint the areas with a high complexity.

With these four methods the data models for the DRS4 and DRS5 software system have been evaluated. DRS stands for the “Domain name Registration System”, it is used by SIDN to manage the “.nl” ccTLD. The DRS5 data model was found to be more complex than the DRS4 data model, mainly because the DRS5 data model contains more entities , attributes and relationships. This makes the data model harder to understand and to maintain.

The results of the methods have been validated with the use of maintainability metrics results for the source code of the DRS software system. In addition to the software source code metrics, the Gini coefficient was also used. With the Gini coefficient it is possible calculate the distribution of the results from the data model evaluation methods and the source code metrics. Because it is population size independent, it is suited to compare different data models with each other.

The results of the data model evaluation methods are supported by the results of of the source code analysis and by comparing the Gini coefficients of the different results. The conclusion is that it is possible to objectively measure the maintainability of a data model. With the use of Gini coefficient it even becomes possible to compare the results of different data models with each other. The newly proposed evaluation method also adds the possibility to find areas in a data model which are more complex than the average complexity of the data model.

Keywords:
Data model, maintainability, ISO9126, Gini coefficient

Contents

1. Introduction

This thesis describes the methods which can be used to objectively determine the maintainability of a data model. The aim of this thesis is to determine if it is possible to compare the maintainability of different data models with each other. And to use the maintainability methods to analyze a data model for elements with a high complexity.

Tom DeMarco once remarked “You cannot control what you cannot measure” [13], in order to make any valid claim about the quality of a software product, it is imperative to objectively measure the quality of the software product. “objectively” in this context is defined as measuring the quality in such a way the results do not depend on the opinion, background, assumptions or mood of product experts such as an architect, developers, customers and other experts. The only way to gather true objective measurements is by calculating measurement data directly or indirectly from the properties of the software product.

The quality of a software product is determined by two types of quality, internal and external quality. Internal quality can be evaluated by measuring the internal attributes of a software product. This can be done with the use of static analysis. The external quality can be evaluated by measuring the behavior of the product when it is executed. Complexity for example is an internal quality attribute, whilst usability is determined by the experiences of the user with the product, and is therefore an external quality.

This thesis will focus on measuring internal quality only, this internal quality can be objectively measured using the structural properties of a data model.

1.1. Motivation

Data modeling is a part of requirements engineering, it involves translating user requirements into a data model which captures the reality of the user. This data model describes the Universe of Discourse (UoD) for a problem area. The resulting data model forms the foundation of a software system. If this foundation is weak or contains cracks, this will affect the entire software system. Modifying or improving the foundation after the entire software system has been completed is a costly endeavour. Having a method to determine if the foundation is solid and flexible enough to support an evolving software system can be of great value. Fixing problems in the foundation before the rest of the software system has been built can be much cheaper compared to fixing the problems after the complete software system has been delivered.

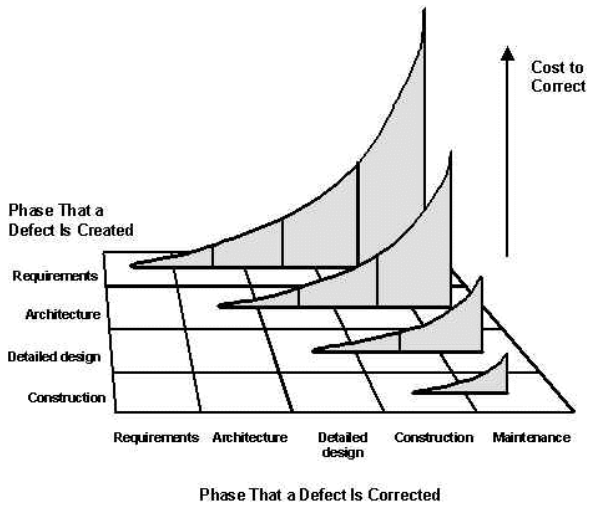


Figure 1 The costs of defects in development lifecycle

Earlier research done by Boehm [8] shows that relative to removing a defect which is discovered in the requirements phase, removing the same defect costs on average 3.5 times more during design, 50 times more at the construction phase and up to 170 times more during the maintenance phase after the product has been delivered [16]. see figure 1 for an overview of the costs in the software development lifecycle.

This exponential increase in the costs of removing defects in phases after the requirements phase shows the importance of catching defects early in the requirements phase. According to Moody "it is during this phase that the notion of software engineering as a craft rather than an engineering discipline is strongest"[1] . No guidelines are established which make clear what the properties of a "good" data model are. The interpretation of the quality of data models is left up to the data modeler. It is mostly determined by the background and experiences of the modeler [8]. This is where the core of the problem lies, Moody describes this as: " *the process of designing a data model is more of an art then it is an engineering discipline. For this to change, quality standards need to be defined, agreed and applied in practice.*" [7]. Without quality standards the definition of data model quality will be left up to the data modeler. Every data modeler will have different ideas on what quality is, this is dependent on the background of the data modeler, the training he has received and his experience.

The main goal of this thesis is to identify and test practical metrics that can be used to determine the maintainability of a data model. A secondary goal is to use the metrics to locate the problem areas of a data model, these are areas with a high complexity. A "practical" metric is defined as a metric which can be used easily, reliable, and cheaply in practice. This means it must be possible to determine the value of the metric from calculating the properties of a data model.

1.2. Research questions

The main research question of this thesis is:

How to compare the maintainability of data models?

To answer this main research question a number of sub-questions need to be answered.

1- What is "maintainability" in the context of a data model?
Hypothesis: Data model maintainability is determined by complexity.

2- What methods can be used to measure the maintainability of a data model?
Hypothesis: The maintainability can be measured by counting the structural elements and their relationships.

3- How to determine the effectiveness and correctness of a measurement method?
Hypothesis: The effectiveness can be determined by correlating the results of the data model metrics with the results of the source code metrics.

1.3. Related work

Research into data model maintainability has resulted into a number of proposals for methods and frameworks which can be used to measure data model quality, and maintainability as a sub characteristic of quality. Moody [1]has identified over 50 of these proposals. Not all of the proposals are validated by using them on actual data models. No research has been found which uses the existing proposals for a comparative evaluation, in order to be able to compare the results of the different methods.

1.4. Structure of this thesis

Chapter 2 describes the research method used for answering the research questions.
Chapter 3 presents the background required for understanding the research done in this thesis.
chapter 4 describes the different existing data model maintainability metrics which were found in the literature and were selected for this research.

Chapter 5 presents the results of the research, it describes how the different data model quality metrics were applied to existing data models. Chapter 6 presents an interpretation of the results of all the methods, the strengths and weaknesses of the methods are also reviewed.

Chapter 7 contains the conclusion and possible future work.

2. Research method

The research is divided into two separate phases. The first phase consists of a literature study. This literature study will result in an overview of existing and usable data model evaluation metrics. The second phase is where the practical research is done. The metrics identified in the first phase will be applied to actual data models in the second phase. This chapter will describe the approach for answering each of the research questions.

2.1. Research questions

2.1.1. Q1 What is "maintainability" in the context of a data model?

During the literature study a search will be done for papers which will help in determining what "maintainability" is and how it is used in the context of a conceptual data model. The links between data model maintainability and software engineering maintainability as described by ISO 9126 will also be investigated.

2.1.2. Q2 How can you measure the maintainability of a data model?

A study of the existing literature will result in a list of models, frameworks and methods for measuring the maintainability of a data model with the use of metrics. This list of metrics will be analyzed for usability. The criteria for a useable metric are: The metric must be an objective metric, meaning it must not depend on the opinion, experience etc of a person related to the model. The metrics must measure a structural property of the data model, these criteria make it easier to collect data with an automated technique.

2.1.3. Q3 How to determine the effectiveness and correctness of a measurement method?

This is the most difficult question, the data model maintainability metrics which are identified in answering Q2 are not easy to compare with each other. They use different approaches to maintainability and get different types of results. If we want to know how to determine if one method is better than another method, it is necessary to develop a method to compare the results of the different methods with each other.

To determine if the results of a measurement method are correct, the results will be compared with the results from a source code analysis. The assumption is that the maintainability of the source code is linked to the maintainability of the data model. If the maintainability of the source code is good, then the data model maintainability will also be good or at least sufficient.

The source code which will be used for this analysis is the source code of the DRS application. This application is designed to make use of the DRS data model. Only the analysis results of parts of the source code that are directly responsible for mapping the data model entities to Java classes and manipulating data will be used . The analysis consists out of a static analysis performed by the Software Improvement Group. The results for fan-in, fan-out and number of classes metrics of the source code will be used to compare with the results of the data model metrics.

The static analysis is not part of this thesis research this is an ongoing activity at the company where this research is performed. This thesis will only make use of the already collected results.

Another method for validating the data model metric results is by using a Lorentz graph and the Gini coefficient. The results of a method can be highly misleading if the distribution is a-symmetric. The Gini coefficient can also be used to compare the results of the different measurement methods with each other.

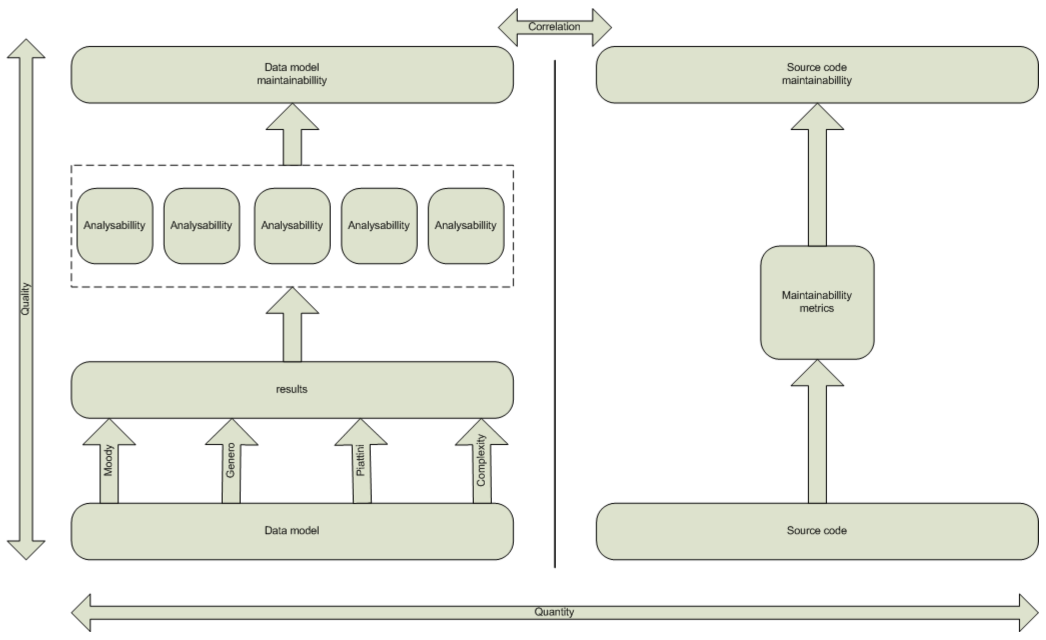


Figure 2 Research method overview

Figure 2 shows a graphical representation of the research method, On the left is the data model, evaluation of the data model leads to results which are used to determine maintainability. This data model maintainability is linked correlated to the maintainability of the source code.

2.2. Phases

The research done for this thesis is divided into a number of phases, see figure 3.

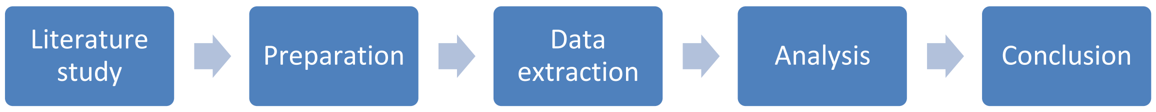


Figure 3 Research phases

2.2.1. Literature study

The literature study was performed by doing an exhaustive search of several online libraries such as: University of Amsterdam digital library, Springer link, ACM digital library, IEEE digital library and Google Scholar.
The search also included articles about regular (source code) software maintainability. The ISO/IEC 9126 standard was also studied. The articles which resulted from the search were sorted on relevance and importance. This resulted in a shortlist of articles which describe methods for measuring the maintainability of data models. For general background knowledge about relational theory a well known text book by Date [17] was used. The study also includes a search for papers describing the use of the Gini coefficient in the context of software engineering.

2.2.2. Preparation

The preparation phase is concerned with selecting data models which can be used in the data extraction phase. The data models used in this study are data models which are actually used in practice. They form the core of the domain name registration system of a leading European internet domain name registry. This domain name registration system (DRS) was recently completely rewritten. This study will look at the data models used by version 4 and version 5 of DRS. There are major architectural differences between both the data models, therefore it will be interesting to see if and how the maintainability was effected by those changes.

Both versions of the DRS data model are Oracle database schemas, the create scripts for all the structural elements of the data models will be collected. These scripts will be used in the data extraction phase to create a working database system which can be used to extract data. The extraction of measurement data will have to be done automatically, therefore we need working databases for all the data models. A database server will have to be set up for this.

To link the data model measurement results to the maintainability of the software which makes use of the data models, the analysis results of the DRS software will be collected. This software analysis is a static analysis of the source code. The analysis is performed by the Software Improvement Group (SIG). The software analysis measures many standard software metrics. These metrics include McCabe cyclomatic complexity, Fan-in, Fan-out, Module violations etc. The software metrics which can be used to validate the results of the data model metrics must be determined by studying the metrics in the data model quality methods.

2.2.3. Data extraction

In this phase the data models which were chosen in the preparation phase will be analyzed with the use of the evaluation methods which were selected in the literature study phase. For each of the data models a working database schema will be created. This schema can then be used to automatically collect data with the use of SQL queries.

Three evaluation methods were selected in the literature study, these are methods proposed by Moody [1,4,5,6,8], Genero [2,3,20] and Piattini [2,25]. A fourth evaluation method will also be used, this method is consists out of several metrics which are proposed by the author of this thesis. This method differs from the others in a sense that it will not produce results which describe the data model as a whole. These metrics will determine the individual complexity of the entities contained in the data model. The evaluation methods each contain a set of metrics. Not all of the of these metrics will be used to measure the DRS data models. Only those metrics which influence the maintainability of the data model will be used. The results will be collected and analyzed in the analysis phase.

2.2.4. Analysis
In the analysis phase the results from the previous phases will be used to answer the research questions. Validating the results of the data model metrics will be done with the help of the Software Improvement Group (SIG) measurement results for the DRS4 and DRS4 source code. With the help of Lorentz graphs and the Gini coefficient a correlation between the source code and data model results might become visible.

The source code of the DRS software which uses the DRS data model as its data source is analyzed by the SIG. The SIG has developed a maintainability model [19] which it uses to analyze software for its maintainability. The metric results of the DRS data models will be compared with the results of the analysis of the software. If the data model maintainability scores low it is presumed that this will also be the case for the software maintainability. Only those parts of the software which implement the database manipulation functionality will be used when correlating the results of the data model and the source code.

There is a SIG analysis done on the software of DRS4 and DRS5 system, this allows us to link the metric results for version 4 and 5 of the DRS data model, to the software metric results of version 4 and 5. If the DRS5 data model maintainability is higher or lower than the DRS4 data model maintainability, it is assumed that this will also be reflected by the software maintainability results.

To compare the results of the data models with each other and with the results of the software metrics, the Gini coefficient is used. The Gini coefficient can be used as a measure for the distribution of functionality in a software system. The Gini coefficient is population size independent, this makes it possible to compare software products (data models, source code) of different sizes with each other. A comparison of the Gini coefficients of the data models with the Gini coefficients of the source code will show if the source code maintainability is influenced by the data model maintainability. If the maintainability of both the data model and source code show the same changes then the source code results can be used to support the data model results.

2.2.5. Conclusion
In this phase the results from the analysis phase will be used to determine if the research questions have been answered.

2.3. Threats to validity

The biggest threat to the validity of this research will be the small scale of the research, only two data models were used. This is probably not enough to provide a conclusive answer to the main research questions. But the research was done within a limited time frame, there was not enough time to study more data models. In the literature study a number of articles were found which stated that future work is needed on data models (which are used in practice). It's my hope that although the limited scale of this research, it will succeed in providing a better insight into the problem of data model maintainability.

The used methods could all give a different picture of the data model maintainability, in which case it will be impossible to determine which method is correct. The assumption that the quality of the DRS5 data model is better than the quality of the DRS4 data model may turn out not to be true.

3. Background

This chapter contains a review of basic information obtained during the literature study. The environment where the research was done and the products that were studied are also described.

3.1. Metrics

The relational model was first proposed by Codd in 1970 [10], at the moment the relational model is the dominant model in the database market. This research will not look into metrics for other types of databases e.g. Object, XML databases etc. For years the only method for determining the quality of a data model was by the use of the "normal forms" defined by Codd [18]. Codd defined a number of normal forms of which the first three are the most well known. These normal forms can be used to remove redundancies from the data model, but in the process of removing the redundancies the data model is changed. The modified data model could turn out to be more complex to understand because it contains more entities and relationships.

Complexity in a data model can be compared with complexity in software products, and many years ago the eminent Edsger Dijkstra warned against the dangers of complexity "*The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility*" [16]
Dijkstra's warning proved right when its concerned with software products, studies have correlated complexity with low reliability and frequent errors. Using the McCabe and other complexity metric to identify en improve problem areas in software code can increase the reliability of a software product [11].

Using metrics to determine the quality of a data model is done for the following reasons [6];

1) Improving on the quality of an existing data model.
2) Choosing between different alternative data models.
3) Tracking the evolution of a data model.

There are two types of metrics: closed-ended metrics and open-ended metrics [2]. The results of a closed-ended metric can only fall within a certain range. The metrics proposed by Piattini which use a ratio of some part to the corresponding whole are an example of these metrics [2, 25]. The results of an open-ended metric are in a range where the upper or lower boundaries of the range are not absolutely fixed. An example would be the total number of attributes of a entity in an ERD.

In software products metrics can be divided into two groups, intra- and inter-module metrics. For data model metrics the metrics can be divided into intra-table and inter-table metrics. intra-table metrics measure the table in isolation, inter-table metrics measure the interaction between tables. Following this distinction two different types of metrics can be defined for relation databases: table-oriented and schema-oriented [15]

Measuring software quality is an activity used in the software lifecycle to assure the delivered software complies to the requirements set for certain quality attributes.
There are multiple software quality models describing quality attributes. Perhaps the most well known is the ISO/IEC 9126 [12] standard. ISO/IEC 9126 describes internal and external quality attributes for software products. Measuring those quality attributes is done with the use of so called "metrics".

3.2. Current state of data model metrics

The history of software metrics goes back to the late 1960's when LOC (Lines Of Code) or KLOC (thousands of Lines Of Code) were used to measure productivity and the defect rate per LOC/KLOC. Since then a very active research field has been established. For software products a number of well known metrics are available. The McCabe cyclomatic complexity[11] is a good example of a quality metric for software used to measure complexity. Other metrics focus on issues as cohesion between modules, the number of lines of code, variable span and lifetime etc [16].

When compared to software engineering metrics for source code which have been around for almost as long as the beginning of the software engineering discipline, metrics for data models are relatively newcomers. The first published papers about evaluating conceptual models are by Krogstie and Kesh [9]
Since then a lot of different frameworks and metrics have been proposed. In a study done by Moody 50 different frameworks or metric proposals were counted . This indicates that data model quality is still an active research field. There are many competing proposals which describe how data model quality should be defined and measured.

There is no consensus among the developers of the different proposals, which makes it very difficult to develop a common framework. Currently if someone would like to evaluate the quality of a data model that person would have to choose between all those different proposals.

While software quality metrics have been incorporated into the development lifecycle and have been recognized as a valuable indicator of the quality of a software product [14], the use of data model metrics is not commonplace yet. There are many proposals for quality frameworks, a recent study identified more than 50 different proposals. One of the reasons an "official" data model quality standard has not yet emerged may be caused by the fact that there are some many proposals causing a fragmented research field. [1]

Applying the metric models which are described in this thesis does not result in a single distinct number which must then indicate the quality of the examined data model. It is probably not possible to capture all the quality properties into one distinct number. Fenton described this as "The impossible holy grail" [23].
According to Fenton it is impossible to capture all the complexities of a data model into a single number[23]. Therefore no metrics model used in this research results into a single number representing a quality score.

3.3. Maintainability

Maintenance can account for 60% to 80% of the total cost in a typical software lifecycle
This cost factor is the driving force behind the software quality research field. Before searching for metrics which can measure data model maintainability, it is necessary to precisely define what maintainability actually is. The ISO/IEC 9126 describes a quality model for a software engineering product.

This quality model defines maintainability as: "The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in the environment, and in requirements and functional specifications." [12]

A data model can be seen as a software product, it is just as much an artifact of a complete product as the source code of a program is. Therefore it is logical to reuse this definition of maintainability when data models are the subject of investigation. The ISO/IEC 9126 quality model describes product quality, a product can have both internal and external qualities (figure 4).

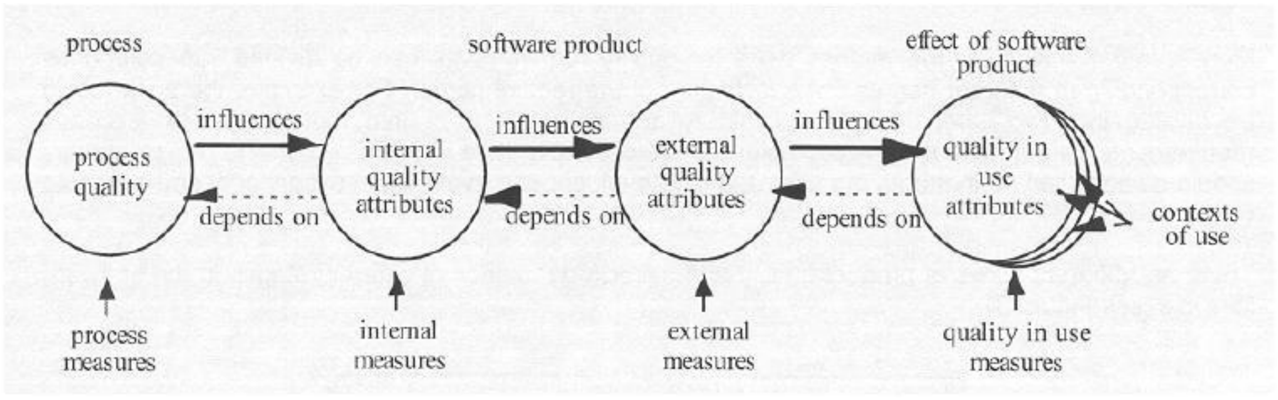


Figure 4 ISO/IEC 9126 quality lifecycle

Total product quality is determined by these two types of quality, internal and external quality. Internal quality can be evaluated by measuring the internal attributes of a software product. This can be done with the use of static analysis. The external quality can be evaluated by measuring the behavior of the product when it is executed.

The internal quality influences the external quality of a product, see figure 5. Complexity is for example an internal quality attribute, if the complexity of a product is high, then it is difficult to maintain and prone to errors. Product errors negatively influence usability which is an external quality attribute. The internal quality attributes for software can be evaluated by measuring internal attributes such as lines of code, complexity, variable span etc.

The data model quality measurement models which are reviewed and used in this thesis assume that measuring internal qualities of a data model will provide information about the quality of that data model. The ISO/IEC 9126 quality model contains six quality characteristics. Portability, maintainability, usability, reliability, efficiency and functionality. See figure 5.

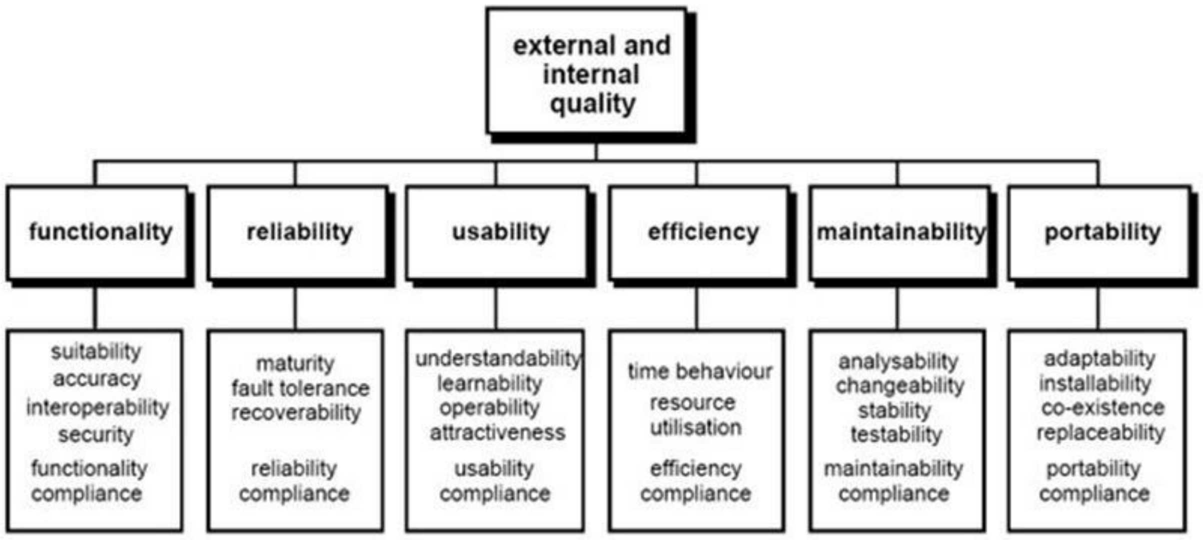


Figure 5 ISO 9126 quality model.

The maintainability characteristic is divided into 5 sub-characteristics:

	sub-characteristics	**Definition**
1	Analyzability	The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.
2	Changeability	The capability of the software product to enable a specified modification to be implemented.
3	Stability	The capability of the software product to avoid unexpected effects from modifications of the software
4	Testability	The capability of the software product to enable modified software to be validated
5	Maintainability compliance	The capability of the software product to adhere to standards or conventions relating to maintainability.

Table 1 ISO 9126 sub-characteristic definitions

Some of the metrics used by the measurement methods use metrics which are not related to maintainability or cannot be evaluated by measuring the internal attributes of a data model. If this is the case then the metrics used by the method will be mapped onto the maintainability sub-characteristics to determine which of the metrics influence maintainability as defined by ISO/IEC 9126. Only those metrics which have a direct influence on the maintainability of a data model will then be used.

3.4. Gini Coeffecient

Software and data model metrics provide us the tools with which we analyze the different properties of a software product. This is very useful when comparing different software products with each other, it also makes it possible to monitor the evolution of software product. This information is then used by developers, project managers etc. to support their decisions. The problem with this is that the data which results from measurement using these metrics is often heavily skewed. [22] What this means is that the distribution of the data set is non-Gaussian. Analysis methods like calculating the "average" or "mean" assume that the distribution of data in a data set is Gaussian, if it is not this will result in an unreliable result.

One method to overcome this problem and calculate reliable results from data sets with a-symmetric distributions is to use the Gini Coeffecient (GC). The GC was developed by the Italian statistician Corrado Gini. It was mainly used in the area of economics, to determine the distribution of wealth in a society. It is a closed ended numeric value between zero and one. In the application of determining the distribution of wealth, zero means that all the wealth is equally divided, one means that one person has all the wealth in a society. How can a method for calculating the distribution of wealth in a society be of any use in the context of software metrics? It is possible to see the internal structures of a software product like classes, entities, relationships etc. as the population of a software product. It is then not unlikely to assume that the distribution of functionality and responsibilities is not evenly distributed among that population.

By replacing the wealth distribution data with the data resulting from metric measurements it is possible to calculate the GC for a software metric. An advantage of the GC is that is population size independent and bounded. This means the GC can be used for comparative analysis of software products

"Software systems tend to exhibit asymmetrically-shaped metrics data distribution profiles" [22] This is also known as the Pareto principle [22] also known as the 80-20 rule. When the Pareto principle is applied to software engineering this means that 80% of the complexity and functionality comes from 20% of the software constructs (classes, objects etc.) The other 80% of the code contains relatively simple support code and abstractions. This uneven distribution of complexity can cause unreliable results when measuring complexity metrics.

The Gini coefficient is based on the Lorentz curve which can be used to measure inequality within a population. See figure 6 for a graphical representation of the Gini coefficient. The diagonal line is the equality line value of the Gini coefficient, all values are evenly distributed on that line. Below the equality line we see thee Lorentz curve.

The Gini coefficient is defined as: $\mathrm{GC} = \frac{\mathrm{A}}{\mathrm{A+B}}$

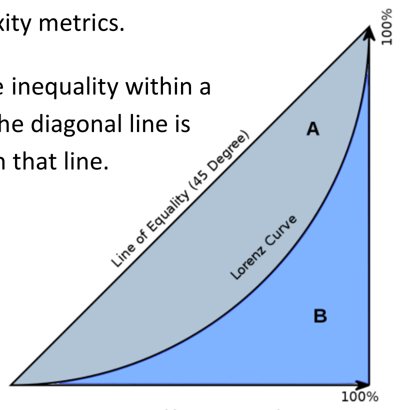


Figure 6 Gini coefficient definition

Where A is the area between the diagonal equality line and the Lorentz curve, B is defined as the area under the Lorentz curve, see figure 5. The Gini coefficient is population size independent it can be used when comparing software products with different population sizes.

3.4.1. Data model metrics and Gini coefficient

What makes the Gini Coefficient so useful ? It's the fact that the Gini Coefficient is population independent, this simple fact makes it possible to compare the maintainability results for different data models with each other. Even if the data models differ in size (number entities, attributes and relationships).

Not all metric results can be transformed into a Lorentz curve and a Gini Coefficient. The metric results must consist out of a population of values. If the metric only results in a aggregate value which is representative for the entire data model, then it's not possible to use the a Lorentz curve and Gini Coefficient. An example is the "NA" attributed defined by Genero [23], this results in a single value which represents the total number of attributes in a data model.

4. Data model metrics

In this thesis three different methods for evaluating the maintainability of data models will be reviewed and used in a comparative test with data models which are used in practice. After a literature study the following three methods were found and selected :

1) Moody
2) Genero
3) Piattini

These methods are designed to measure the maintainability of conceptual data models, the data models used for this research are logical data models. Some of the constructs of a conceptual model such a M:N (many to many) relationships are not present in a logical data model. In the logical data model these M:N relationships are translated into three separate entities with relationships between them. This means that not all of the metrics which are described in the evaluation methods can be used, if a metric is not used it is explained why it could not be used.

A fourth method is proposed in this thesis, the method can be used to determine what parts of a data model are more complex. This method is called:

4) Entity Complexity

4.1. Moody

Daniel L. Moody has described a framework [4,5,6,8] for data model evaluation. This framework consists of a number of quality factors (figure 6) which together could give a comprehensive quality indication of a data model. The ISO/IEC 9126 [12] standard describes six characteristics of software quality, one of those characteristics is maintainability. For this research we are interested in those metrics of the Moody framework that influence maintainability.

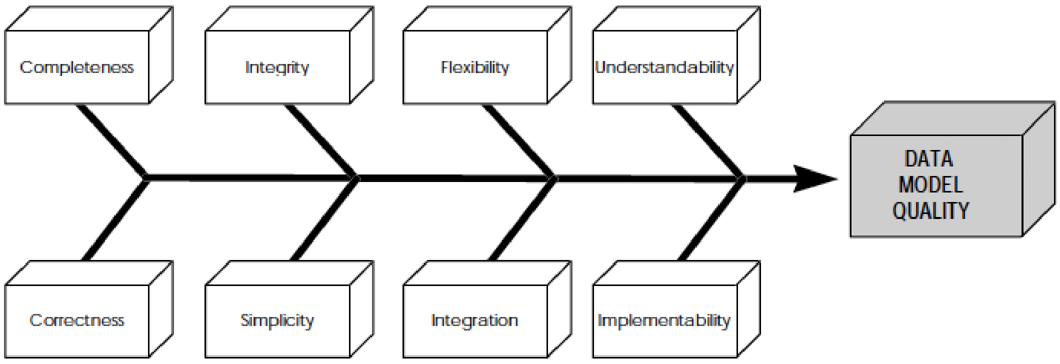


Figure 7 Moody quality factors

To find these metrics the ISO/IEC 9126 standard is used to determine which of the quality factors is related to maintainability as defined in the ISO/IEC 9126 standard. This thesis focuses on evaluating the maintainability of a data model, whilst the Moody model is design to give a general quality overview of a data model. Each of the quality factors is measured by one or more metrics, these metrics are not always objective metrics (a metric whose value is derived directly from data model properties by calculation)

The quality factors are defined by Moody [6] as:

Factor	***Description***
Completeness	The data model contains all the information required to support the required functionality of the system.
Integrity	The data model defines all the business rules which apply to the data.
Flexibility	The ease with which the data model can cope with business and/or regulatory change.
Understandability	The ease with which the concepts and structures of the data model can be understood.
Correctness	conformity to the rules of the data modeling technique, this includes diagramming conventions, naming rules, definition rules, rules of composition and normalization.
Simplicity	The data model contains the minimum required structures e.g. entities and relationships.
Integration	Consistency of the data model with the rest of the organizations data.
Implementability	The ease with which the data model can be implemented within time, budget and technology constraints of a project.

Table 2 Moody Quality factors

4.1.1. Mapping Moody to ISO 9126 maintainability

The sub-characteristics of maintainability as described by ISO/IEC 9126 can be mapped onto the Quality factors of the Moody method. With this mapping it becomes clear which of the quality factors are related to maintainability.

Moody↓ ISO →	*analysability*	*changebillity*	*stability*	*testability*	*compliance*
Completeness	-	-	-	-	+
Integrity	-	-	+	-	+
Flexibility	-	+	-	-	-
Understandability	+	+	-	-	-
Correctness	+	-	+	-	+
Simplicity	+	+	-	+	-
Integration	-	-	-	-	+
Implementability	-	-	-	-	-

Table 3: mapping of the Moody quality factors onto ISO/IEC 9126

Maintainability can be broken down into sub-characteristics. The analyzability, changeability and testability sub-characteristics of maintainability are influenced by complexity [15, 19]. The mapping of the Moody quality factors onto the ISO/IEC 9126 sub-characteristics of maintainability gives us the quality factors that are influenced by complexity and therefore will affect the maintainability of a data model. Table 3 shows the quality factors which are influenced by complexity and the type of metrics used to measure the quality factor.

After having mapped the quality factors onto ISO/IEC 9126 sub-characteristics to find out which quality factors determine the maintainability of a data model, the remaining quality factors that are not objectively measurable are removed. Moody has defined metrics for each of the quality factors [6]. Not all of those metrics are objectively measurable, see table 4.

Quality factor	*Metric*	*Objective*
Flexibility	Number of data model elements which are subject to change.	NO
	Probability adjusted cost of change.	NO
	Strategic impact of change.	NO
Understandability	User rating of understandability.	NO
	User interpretation of errors.	NO
	Developer rating of understandability.	NO
Correctness	Number of violations to data modeling standards.	NO
	Number of instances of entity redundancy.	NO
	Number of instances of relationship redundancy.	NO
	Number of instances of attribute redundancy.	NO
Simplicity	Number of Entities.	YES
	System complexity, entities + relationships.	YES
	Total complexity, entities + relationships + attributes	YES

Table 4 Metrics for quality factors

4.1.2. Usable metrics

The goal of this thesis is to find easy to use metrics which can be used cheaply and effectively and these metrics must be objective metrics. Objective metrics are cost effective, any metric that does not match these requirements will most probably not be used in practice [8].This is because collecting subjective metrics requires a lot of effort and will therefore consume more manpower and costs. Subjective metrics can only be collected by interviewing designers, stakeholder, customer etc. And they are inherently prone to bias.

Table 4 shows that only the simplicity metrics are objectively measurable. The Correctness metrics would seem to be objectively measurable but they are not, they describe unspecified violations of unspecified standards and semantic redundancies in the data model. Finding those semantic redundancies
requires analyzing the entire data model by hand, a very time consuming activity.
The Understandability metrics are not objective metrics, because they depend on the response of a stakeholder and are therefore influenced by the ideas, experience, and other forms of bias of the stakeholder. The only metrics remaining for measuring the maintainability are the metrics that belong to the simplicity quality factor. Simplicity is directly related to complexity which is known to have a direct influence on maintainability [19].

4.1.3. Moody simplicity quality factor

Moody has divided the simplicity quality factor into three separate metrics:

The number of entities (E)

This a very simple metric, it only counts the number of entities in a data model. This E metric can be compared with the number of source code files in a normal software engineering product. Increasing the number of separate containers which each hold information and/or logic about the system, makes it more difficult for developers and maintainers to get a good overview. In very big systems it could even be impossible for a single person to understand all the separate components and the interactions among them. The E metric is however limited in its use, it would be hard to compare the maintainability of two different data models only with this metric. A data model with twice as much entities is not necessarily twice as difficult to maintain. To compare the maintainability of two different data models (two different solutions for the same UoD) more information is required then the number of entities. The real value of this metric is in comparing versions of the same data model with each other. This way it is possible to track the evolution of the data model and determine if the data model becomes more complex because of the increasing number of entities.

The combined number of entities and relationships (E+R)

This metric builds on the E metric and adds the number of relationships to it. This results in a finer grained metric, which gives more insight in the true complexity of a data model than is possible with the E metric alone. This metric is based on complexity theory, according to which the complexity of a system (which includes database systems) is determined by the number of entities and the number of relationships between these entities.

The total number of data model structures ($aN^E+bN^R+cN^A$)

This metric builds on the E+R metric and adds the number of attributes of the entities. This creates a fine grained metric which incorporates all the constructs of a data model.

The research part of this thesis will use the simplicity metrics defined in the Moody model to determine the maintainability of the data models.

4.2. Genero

Genero [2, 3, 20, 23] describes a set of metrics which measure understandability, which is a key factor of maintainability [3].

The framework used by Genero to define the metrics is based on earlier research done in software engineering. In software engineering prediction models have been proposed for measuring external qualities like maintainability. Those models use structural software properties like coupling, cohesion and inheritance structures to determine software quality. The advantage of using structural software properties is that they are available at an early stage of development.

Briand [21] has suggested that the reason these structural properties have an impact on quality is because software that is composed out of many elements with relationships between the elements result in a high cognitive complexity (figure 5.3.1). It is this high cognitive complexity which causes negative effects in software such a higher error rate and difficulty in maintaining the code. This is simply because it is more difficult to understand. This matches the observation made by Dijkstra which is mentioned in the introduction of this thesis.

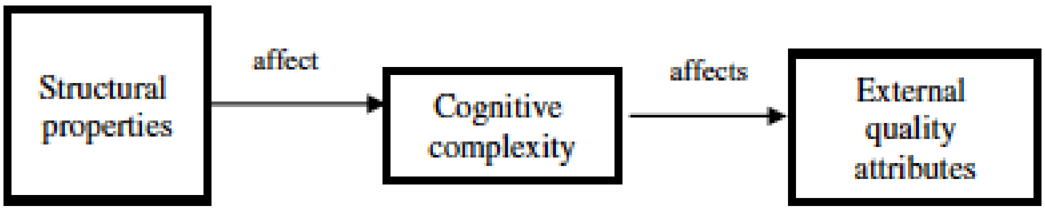


Figure 8 Relationships between structural properties, cognitive complexity and external quality attributes

Piattini, Genero and Jimenez [20] have used the ISO 9126 definition on maintainability. However they do not agree that all the maintainability sub-characteristics described in the standard are suitable for data models. They have defined their own sub-characteristics.

Sub-characteristic	***Description***
Understandability	The ease with which the conceptual data model can be understood.
Simplicity	The conceptual data model should contain the minimum number of constructs possible.
Analyzability	The capability of the conceptual data model to be diagnosed for dependencies or for parts to be modified to be identified.
Changeability	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.

Table 5 Genero maintainability sub-characteristics

In order to use the Briand model as a starting point, the data model is considered to be a software artifact. Genero takes the position that the quality of data model cannot be captured by a single metric. This is supported by research done by Fenton, the framework proposed by Genero uses twelve different metrics [23].

4.2.1. Metrics
The metrics that are proposed in the Genero method are presented in table 6.

Metric	*Description*	*Objective*
NE	The Number of Entities metric is defined as the number of entities within an ER diagram.	YES
NA	The Number of Attributes metric is defined as the total number of attributes defined within an ER diagram, This also includes relationship attributes. In this number all attributes are included (but not the composing parts of composite attributes). By convention, the NA metric does not count the attributes that a sub-type inherits from its super-type (i.e. these attributes are counted only once, as attributes of the super-type).	YES
NDA	The Number of Derived Attributes metric is defined as the number of derived attributes within an ER diagram. The value of NDA is always strictly less than the value of NA.	NO
NCA	The Number of Composite Attributes metric is defined as the number of composite attributes within an ER diagram. This value is less than or equal to the NA value.	YES
NMVA	The Number of Multi-valued Attributes metric is defined as the number of multi-valued attributes within an ER diagram. Again, this value is less than or equal to the NA value.	NO
NR	The Number of Relationships metric is defined as the total number of relationships within an ER diagram, excluding ISA relationships.	YES
NM:NR	The Number of M:N Relationships metric is defined as the number of M:N relationships within an ER diagram. The value of NM:NR is less than or equal to the NR value.	YES
N1:NR	The Number of 1:N Relationships metric is defined as the total number of 1:N and 1:1 relationships within an ER diagram. Also this value is less than or equal to the NR value.	YES
NN_AryR	The Number of N-Ary Relationships metric is defined as the number of N-Ary relationships within an ER diagram. Its value is less than or equal to the NR value.	YES
NBinaryR	The Number of Binary Relationships metric is defined as the number of binary relationships within an ER diagram. Again, the value is less than or equal to the NR value.	YES

Table 6 Genero metrics

The NDA metric measures the number of derived attributes, the value for a derived attribute is derived from the values of other attributes. In other words, a derived attribute is a calculated attribute. It can also be seen as redundancy in a data model. But because of performance requirements derived attributes are sometimes necessary. It is not possible to automatically detect if an attribute is a derived attribute using only the data model properties, for this a dataset would be required. This research will only use metrics which can be calculated directly from the data model properties. Because derived attributes the semantics of a derived attributed cannot be analyzed, this NDA metric will not be used.

The NMVA metric measures the number of multi valued attributes (MVA). A MVA is an attributes which (as the name might already reveal) can contain multiple different values. These different values are stored in one attribute instead of creating a one to many relationship between two entities to be able to store the values. The values in a MVA are usually in a character format where the individual values are separated with the use of separator markers such a comma, colon or dash symbol etc. Using only the data model properties it is not possible to detect whether an attribute is a multi valued attribute or a normal attribute. This metric will not be used for this research.

4.3. Piattini

The metrics proposed by Piattini [2, 25] are all closed-ended type metrics, They calculate a ratio between a part of the data model to the whole of the data model. This results into a number in the range of zero to one. The metrics are designed to measure the structural complexity of an entity relationship diagram (ERD). In the chapter about maintainability the link between maintainability and structural complexity was described. Because all the metrics in this method are designed for measuring maintainability, we do not need to eliminate metrics from the Piattini proposal that do not influence the maintainability of the data model. The only reason for not using one of the proposed metric s would be if the metric could not be objectively and automatically measured.

The use of closed-ended ratio type metrics offers some clear advantages when compared to the metrics proposed by Moody and Genero. First of all, being closed ended means that there is a built in scale which can be used to interpret the results of the measurements. Second, the use of ratio measurements makes it easier to compare the results of measurements done for different data models. Even if those data models have different populations of entities and relationships etc.

4.3.1. Metrics

These are the metrics proposed by Piattini:

Ratio of the number of relationships and entities

This metric measures the relation between the total number of relationships and the total number of entities in a ERD. This metric is influenced by the number of relationships per entity in an ERD diagram

$$RvsE = \left(\frac{N^R}{N^R + N^E} \right)^2$$

NR is the Total number of relationships in the ERD, NE is the Total number of entities in the ERD. NR + NE must be > 0. The result is zero when there are no relationships and it is one when there are many relationships and very few entities.

Ratio of the number of entity attributes and the number of entities

This metric measures the number of entity attributes and compares this with the number of entities in the ERD. .

$$EAvsE = \left(\frac{N^{EA}}{N^{EA} + N^E} \right)^2$$

NEA is the number of entity attributes, Ne is the total number of entities in the ERD.
There is a rule that NEA + NE must be > 0.
EAvsE is a closed-ended metric, it is zero when there are no entity attributes and 1 when there are many attributes and very few entities.

Ratio of the number of relationship attributes and the number of relationships

This metric measures the number of relationship attributes and compares this with the number of relationships in the ERD.

$$RAvsR = \left(\frac{N^{RA}}{N^{RA} + N^R} \right)^2$$

NRA is the number of relationship attributes, NR is the total number of relationships in the ERD. There is a rule that NRA + NR must be > 0.

Ratio of the number of M:N relationships and the total number of relationships

This metric measures the number of M:N relationships and compares this with the number of relationships in the ERD.

$$M{:}N\ Rel = \frac{N^{M:NR}}{N^{R}}$$

NM:NR is the number of M:N relationships, NR is the total number of relationships in the ERD. There is a rule that NR must be > 0.

Ratio of the number of 1:N relationships and the number of relationships

This metric measures the number of 1:N relationships (this also includes any 1:1 relationship) and compares this with the total number of relationships in the ERD.

$$1{:}N\ Rel = \frac{N^{1:NR}}{N^{R}}$$

N1:NR is the number of 1:N relationships, N^R is the total number of relationships in the ERD. There is a rule that N^R must be > 1.

Ratio of the number of N-ary relationships and the number of relationships

This metric measures the number of N-ary relationships and compares this with the total number of relationships in the ERD. This metric does not count any binary relationships.

$$\text{N-ary Rel} = \frac{N^{\text{N-aryR}}}{N^{R}}$$

$N^{\text{N-aryR}}$ is the number of N-ary relationships, N^R is the total number of relationships in the ERD. There is a rule that N^R must be > 1. This metric is a closed-ended metric, the result is zero when there are no N-ary relationships, when all of the relationships are N-ary the result is one.

Ratio of the number of binary relationships and the number of relationships

This metric measures the number of binary relationships and compares this with the total number of relationships in the ERD.

$$\text{Binary Rel} = \frac{N^{\text{BinaryR}}}{N^{R}}$$

N^{BinaryR} is the number of binary relationships, N^R is the total number of relationships in the ERD. There is a rule that N^R must be > 1. This metric is a closed-ended metric, the result is zero when there are no binary relationships, when all of the relationships are binary the result is one.

4.4. Entity Complexity

Determining the maintainability of a data model can be a valuable part of a software design lifecycle. Because as mentioned earlier in this thesis, most of the costs and effort are spent in the maintenance phase after the initial product has been delivered. Modifying the data model in the requirements and/or design phase to achieve a higher maintainability can potentially save a lot of needless effort and cost. A data model with a good maintainability score will allow an organization to quickly respond to changing business requirements.

The heretofore mentioned metrics for determining the maintainability of data models all use some of the properties of a data model and output a number which indicates the overall score for that metric for the entire data model. If the score is below par, any data modeler will want to modify the data model to improve its maintainability score. This however is a problem for these metrics, they only deliver an overall score, there is no way of tracking the results back to the individual data model structures. It is these structures that cause a lower than expected score that the data modeler is interested in. Another metric is required, one that will determine the maintainability of individual data model structures relative to the maintainability of the data model as a whole.

4.4.1. Metrics

For this thesis a set of new metrics is proposed, which can be used to determine which entities in a data model have a negative influence on the overall maintainability. These metrics are based on the fact [23] that the structural complexity of a data model is influenced by the number of relationships.

4.4.2. Entity influence metric

The Entity influence metric (EI) is determined by the number of relationships an entity is a part of. To calculate the EI first the average Overall Complexity (OC) must be determined. The OC is measured by dividing the number of relationships in a data model by the number of entities.

$$OC = \frac{N^R}{N^E}$$

N^R is the number of relationships in a data model, N^E is the number of entities in a data model.

Determining the EI for each entity is done by subtracting the number of relationships the entity is a part of from the overall complexity. This will result in a positive number if the entity has a low influence on the maintainability. A negative number would indicate a negative influence on the overall maintainability.

$$EI = OC - E^R$$

The EI must be calculated for every entity in the data model. OC is the overall complexity as calculated above, E^R is the number of relationships the entity is a part of.

4.4.3. Entity fan-out

The fan-out metric is already used for measuring complexity in software products, its used to measure the relationship between outgoing method calls and the container (class, file etc.) of their target. The same thing can be done in a data model, measuring the outgoing relationships of a entity will give a measure of its connectedness. It can be calculated with:

$$EFO = N^{R-out}$$

Where $N^{R\text{-out}}$ is the total number of foreign keys of a specific entity.

4.4.4. Entity fan-in

The entity fan-in metric (EFI) is the counterpart of the fan-out metric. It is also a measure of the level of connectedness of an entity. The EFI for a specific entity measures the total number of relationships which use to the entity as their target. In other words the EFI show the number of foreign keys in other entities which reference to the entity which is being measured. It can be calculated with:

$$EFI = N^{R-in}$$

Where $N^{R\text{-}in}$ is the total number of foreign keys in a data model which reference the entity which is being measured.

4.4.5. Validation

The metrics in this proposal must be empirically validated before it will be possible to use the metrics in practice. The research part of this thesis will describe to application of the metric onto actual data models. the results will be validated against the results of complexity measurements of software which is written to make use of the measured data models. The EFI and EFO metrics are incorporated in the EI metric, the results might show that the EFI and EFO have no added value. If that's the case then these metrics can be eliminated. The metrics will also be validated with the use of Lorentz Graphs and the Gini coefficient. With the help of Lorentz / Gini it should be possible to correlate the results from the data models with the results from the earlier source code analysis.

5. Results

In this chapter the results of the data extraction phase is presented. The four evaluation methods have been used to measure the maintainability of the DRS4 and DRS5 data models. For each method the evaluation results are presented, an analysis is made of the results and the weak points of the methods are discussed. The results are validated using either the results of source code analysis or with the use of Lorentz graphs and the Gini Coefficient.

5.1. Moody results

The data model quality method proposed by Moody[4,5,6,7,8] contained more metrics than the metrics which were used for evaluating the DRS data models. The metrics which were not used, were metrics which do not influence complexity and maintainability. Or metrics which cannot be measured objectively. The maintainability metrics defined by Moody were found to be:

- The number of entities.
- The sum of the number of entities and the number of relationships.
- The sum of the number of entities, the number of relationships and the number of attributes.

The results for the DRS4 and DRS5 data models are presented in table 7.

metric/data model	***DRS5***	***DRS4***	***Difference***
E (entities)	97	85	14,12%
E+R (entities + relationships)	218	189	15,34%
aE+bR+cA (entities + relationships + attributes)	1325	743	78,33%

Table 7 Moody maintainability results for DRS4 and DRS5

The metrics result into values which indicate the level of maintainability of a data model. The metrics measure basic properties of a data model, these properties have been shown [23] to directly influence complexity, which in turn is a major component of maintainability. A positive difference means a lower maintainability and a negative difference means an increased maintainability.

5.1.1. Analysis

The Moody results show an increase for all of the metric results when DRS4 is compared to DRS5. This increase would suggest an increasing complexity. However the Moody evaluation method does not suggest a method for analyzing the differences between data models. The results suggest a lower maintainability for the DRS5 data model when compared to the DRS4 data model. But this might not be the case, the metric results do not give us any insight into the distribution of the relationships and properties over the entity population. An entity with many relationships is more complex than an entity with few or even only zero or one relationship. Therefore just comparing the total number of entities and relationships of different data models does not provide enough detail.

The output of the Moody metrics results in an aggregate result, a distinct number which is valid for the entire data model. It is not possible to use the Gini coefficient for such a single number. To calculate the Gini coefficient a large data set is required, many versions of a data model will have to be analyzed with the Moody metric before a Gini coefficient can be calculated from the results.

5.1.2. Validation

To validate the Moody results, the results of the source code analysis are used. The source code was analyzed with the SIG model for maintainability [19]. The Moody metrics measure the number of entities, attributes and relationships. The source code metrics set contains measures for the number of classes and the fan-in and fan-out for each class.

Only the metric results of those parts of the source code responsible for mapping the data model to source code constructs and manipulating data in the database are used.

The source code measurement data for the fan-in and fan-out metric, has been used to calculate a Gini coefficient. This Gini coefficient will provide information about the distribution of the fan-in and fan-out. A higher a-symmetry in the distribution is caused by entities having more than average amounts of relationships. The Moody method suggests an increasing complexity in the DRS5 data model, table 8 shows the results of the source code metrics.

Data model	*classes*	*Fan-in (average)*	*Gini coefficient fan-in*	*fan-out (average)*	*Gini coefficient fan-out*
DRS4	67	16	0.619134	6	0.379389
DRS5	90	15	0.669061	5	0.533754
Difference	23	-1	0.049927	-1	0.154365
percentage	34,33%	-6,25%	8.06%	-16,67%	40,69%

Table 8 Source code results for DRS4 and DRS5

The source code metrics show a slightly lower average fan-in and fan-out. The Gini coefficient for the distribution of the fan-in and fan-out data has increased by respectively 8% and nearly 41% for the DRS5 fan-in and fan-out.

The number of classes in the source code does show a 34% increase, the data model shows a 14% increase in the number of entities. The E+R and the aE+bR+cA could be compared with the results for the fan-in and fan-out of the source code, with the assumption that the number of relationships affects the fan-in and fan-out for classes which model the entities. The The E+R and the aE+bR+cA however also include the number of attributes and entities, this means the fan-in and fan-out cannot be used to validate these metrics.

The results of the Moody metrics do not provide any information about the level of maintainability, it is only possible to compare the results with the results of versions of the same data models and even then it is not clear what the differences actually mean. For example what does an increase of 10 entities or relationships tell you about a data model? Is 10 bad? Or is it a relatively low increase? Moody defined no scale which can be used to interpret the differences in a meaningful way. The Moody model can only be used to detect the differences between two versions of a data model, it does not work well when one wants to compare two unrelated data models even if they are designed for the same UoD.

The results of the Moody metrics can only partly be confirmed by the results of the source code analysis, the only correlation is between the number of entities and the number of classes. They are both increased in DRS5. The number of classes has increased twice as much as the number of entities however.

Metric	*Validated*
E	Yes
E+R	No
aE+bR+cA	NO

Table 9 Moody metrics validation overview

5.2. Genero results

The metrics proposed by Genero [2,3] are much like the Moody metrics, they are designed to count the number of structural elements of a data model. Where Moody only counted the total number of relationships, this Genero method offers some finer grained metrics for counting relationships. Relationships are sub divided into the following types:

- One-to-many relationships.
- Many-to-many relationships.
- N-ay relationships.
- Binary relationships.

Table 10 shows the metric results for the DRS4 and DRS5 data models.

Metric	*DRS5*	*DRS4*	*Difference*
NE	97	85	14,12%
NA	1107	554	99,82%
NCA	0	0	0%
NR	121	104	16,35%
NM:NR	14	7	100%
N1:NR	121	104	16,35%
NN_AryR	0	0	0%
NBinaryR	121	104	16,35%

Table 10 Genero maintainability results for DRS4 and DRS5

Not all of the metrics could be used to measure the DRS4 en DRS5 data models, some of the metrics are defined at a conceptual level. There is no conceptual design available for the DRS4 and DRS5 data models, therefore the research was done with the logical data model of the database schemas of DRS4 and DRS5. At this level some of the conceptual structures are no longer present or are present in a in a different form.

The NCA metric measures the number of composite attributes. The DRS data models are Oracle database schemas and do not support composite attributes. The results for this metric is zero for both data models.

The many-to-many relationship is a relationship between two entities where the cardinality on both sides of the relationship is set to many. In practice this is not often used because this will cause redundancy in the data model. Most data model modelers will transform the many-to-many relationship into two separate one-to-many-relationships. For this construct an additional coupling entity is required between the original entities of the relation. This is also done in the DRS data models, this means the metric data cannot be calculated automatically anymore because semantic knowledge about the data model is required. For this research I have chosen to calculate the results by hand and include them in the results.

N-ary relationships with more than 2 participating entities which are modelled at the conceptual level are broken down to one-to-many relationships at the logical level of database design. The relationship is transformed by adding a new entity. This new entity splits the many-to-many relationship into two one-to-many relationships. This is a good solution to avoid redundancy in a data model, this however makes it impossible to measure this metric as by definition these relationships no longer exists in the DRS4 and DRS5 data model. The result for both data models for this N-ary metric is therefore 0.

The NBinaryR metric does not offer any more information than the NR metric, all the relationships in a logical data model for relational databases are binary relationships.

5.2.1. Analysis

When a comparison is made between the DRS4 and DRS5 results of the metrics, it is quite clear that the DRS5 data model contains more structural elements, see table 10. The number of attributes and the number of many-to-many relationships are respectively almost doubled and doubled.

All the metrics which could be measured show a clear increase when DRS5 is compared to DRS4, indicating an increased complexity and therefore a decrease in maintainability. This seems to support the results from the Moody method, as expected because the metrics used by the Genero method are very similar to those used by the Moody method.

The Genero method suffers from the same flaws as the Moody metric though. There are no established guidelines or range scales which can be used to interpret the results. Also the metrics do not provide any insight into the distribution of attributes and relationships over the entity population of the data model. If the Genero methods results in a bad score then a data modeler will want to know in what parts of the data model the problems occur. This method is not able to pinpoint the weak points of a data model.
The results like the Moody results are aggregate results which tell us something about the model as a whole, it is not possible to use this one number to calculate a Gini coefficient to learn about the distribution of the metrics results. This method can only be used to compare versions of the same data model, The results of a comparison between two different data models cannot be interpreted in a meaningful way.

5.2.2. Validation

To validate these metrics and determine their usefulness the results of the source code analysis are used. These results were also used for the validation of the Moody metrics, see table 8.

The results for the Genero metrics show an increase for all the metrics that were measured, The source code analysis also shows an increase in the number of classes. This could validate the increase in the number of entities. The increased number of relationships (NR) cannot be correlated with the increased average fan-out. These show opposite movements, the number of relationships is up 16% but the average fan-out is down 16% in DRS5. The Genero method can be only partially validated.

Metric	*Validated*
NE	Yes
NA	No
NCA	No
NR	No
NM:NR	No
N1:NR	No
NN_AryR	No
NBinaryR	No

Table 11 Genero validation overview

5.3. Piattini results

Piattini takes another approach to data model metrics than Moody and Genero have done, the metrics in this method are all ratio based metrics. The results are closed-ended, the lower bound is zero and the upper bound is 1. Closed-ended metrics like these have a build in scale. One could argue that depending on the measured functionality , the lower and upper bounds can be substituted with the labels "good" and "bad" this would make the results easier to interpret.

Table 12 shows the results for the Piattini metrics for the DRS4 and DRS5 data models.

metric/data model	***DRS5***	***DRS4***	***Difference***
RvsE	0.002197027	0.002517449	12,79%
EAvsE	0.000655265	0.001083748	39,54%
RAvsR	0	0	0%
M:Nrel	0.115702	0.067308	41,83%
1:Nrel	1	1	0%
N-aryrel	0	0	0%
Binary rel	1	1	0%

Table 12 Piattini maintainability results for DRS4 and DRS5

The RAvsR metric measures the number of relationship attributes. In the logical data models which were used for this research the relationships did not contain any attribute information anymore which could be measured. The results for this metric have been set to zero. One other observation, in a relational database model the N-ary relationships will have been transformed into two one-to-many relationships. The DRS4 and DRS5 did not contain any N-ary relationships, the results for this metric are set to zero for both data models.

5.3.1. Analysis

The RvsE metric shows an increase in the ratio of relationships to entities, the RvsE has increased 12,8% in DRS5. DRS5 introduced new entities, the RvsE does not give any information from which we can determine which of the entities are responsible for the rise in relationships. The EAvsE metric shows an increase in the ratio of attributes to entities. From the Genero results we know that the number of attributes in DRS is almost doubled in comparison to DRS4. The ratio of attributes to entities rises by 39,5% the rise in EAvsE is relatively limited because DRS5 also added additional entities. The value of the metric is however limited because it does not give any information about the distribution of the attributes over the population of entities. A newly introduced "god" entity with hundreds of attributes would not have been detected using this metric.

The result for the "1:Nrel" metric is 1, this means all the relationships in the data model are of the one-to-many type. But the "M:Nrel" metric is not zero!, this might not seem possible. This is due to the fact that the many-to-many relationships, although not physically present, can be recognized as logical constructs. For this research this has been done by hand but it should be possible to extract these logical many-to-many relationships automatically with a extraction tool that can recognize the relationship semantics..

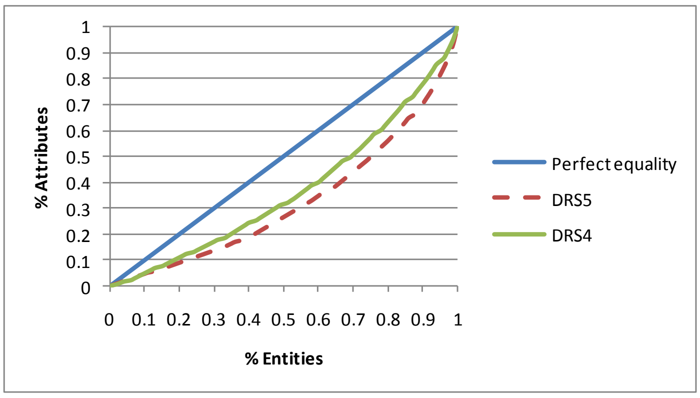
The results show a clear rise in complexity, all the metrics which could be measured have either remained the same of have risen considerable.

5.3.2. Validation

To validate the results of this method, an analysis of the distribution of attributes and relationships is used to determine if the ratios for the RvsE and EAvsE metrics are reliable. The ratios which are calculated in the method assume a symmetric distribution of the attributes and relationships, but this does not need to be true. Especially in data modeling we know that sometimes entities which are not decomposed correctly, can become quite large with many attributes. These entities can also have many relationships and will act as a spider in the data model web, these types of data models are very hard to maintain. To determine if the ratios which result from the metrics in this method portray an accurate picture of reality, a Lorentz curve is created to visually display the distribution of the attributes and relationships in the DRS4 and DRS5 data models.

5.3.3. Attribute distribution

The EAvsE metric measures the ratio between the number of attributes and the number of entities. A high ratio indicates many attributes per entity. According to complexity theory this means more complexity and this leads to a lower maintainability. See graph 1 for the attribute distribution Lorentz curve of DRS4 and DRS5.



Graph 1 Attribute distribution DRS4 and DRS5

With the visual aid of the Lorentz curve it is clearly visible that there is a change in the distribution of attributes between both versions of the DRS data model. The distribution in the DRS5 data model has become less equal, in DRS4 20% of the entities contain 40% of the attributes. When we take a look at the DRS5 data model this has increased, now 20% of the entities contain 50% of the attributes. These changes are not detected by the ratio calculations of the Piattini method. This is because the data in the result set is skewed and calculating averages over a skewed data set can result into a deceptive outcome.

Data model	*Gini Coefficient for attributes*
DRS4	0.263845
DRS5	0.351481
Difference	32,15%

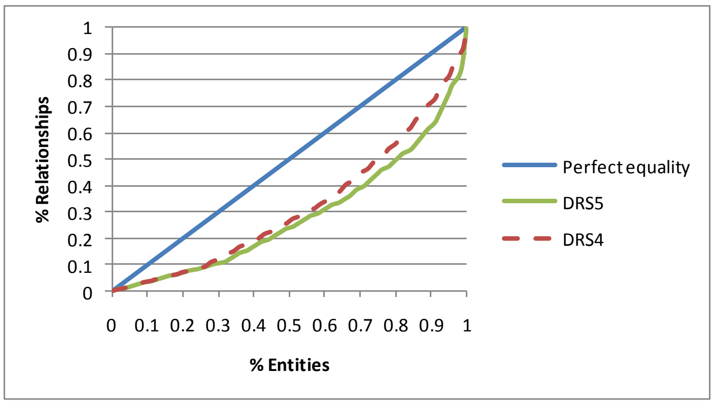
Table 13 Gini coefficient for attribute distribution

Table 13 shows the Gini coefficient for the attributes distribution in the DRS4 and DRS5 data models. The Gini coefficient has increased with over 32% indicating a shift in the distribution of attributes over the entities, the EAvsE metric results show an increase in the number of relationships per entity. The Gini coefficient supports the results of the EAvsE metric.

In a previous study [22] a change in Gini coefficient larger than 4% was found to indicate architectural shifts. Overall less then20% of the projects display a shift of more than 4% between releases. The Gini Coefficient for the attribute distribution shows an increase of 32% this is well above the threshold of 4% which has been found to be an indicator of architectural change .The Piattini method does show an increase in the number of entities and attributes but fails to link this to the major architectural shift which the Gini coefficient hints at.

5.3.4. Relationship distribution

The RvsE measures the ratio between the number of relationships and the number of entities, to validate the results of this metric, Lorentz curves and the Gini coefficient are used. See graph 2 for the relationship distribution Lorentz curve for DRS4 and DRS5.



Graph 2 Relationship distribution DRS4 and DRS5

The Lorentz curves for the distribution of relationships show an increased a-symmetric distribution for DRS5. The Gini coefficients for the Lorentz curves show an increase of 18,6% This means changes to existing entities and the introduction of new entities resulted in a situation where the relationships are less equally distributed over the population of entities. See table 14 for the Gini coefficients of the relationship distribution for DRS4 and DRS5.

Data model	*Gini Coefficient for relationships*
DRS4	0.358888
DRS5	0.425668
Difference	18,61%

Table 14 Gini coefficient for relationship distribution

The RvsE metric shows an increase of almost 13% when the results for DRS4 and DRS5 are compared. This indicates an increase in the average number of relationships per entity. The Gini coefficient shows an increase of almost 19%, which indicates a more a-symmetric distribution of the relationships over the entities. The Gini coefficient supports the results of the RvsE metric.

Overall the Piattini method just like the other two methods from Moody and Genero lacks the ability to give precise information about the underlying causes for the differences in results between DRS4 and DRS5. Because this method is based on ratios, it can be used to compare the maintainability of completely different data models. But this method does not provide any information about the distribution of relationships in a data model, the results could be highly deceptive.

Calculating averages with a skewed data set causes would not show any clustering of attributes or relations for example. So called “god” entities with many relationships and entities cannot be detected using this method. The method does not provide the data modeler with any information about where the weak points of the data model can be found. This piece of information would be crucial for a data modeler seeking to improve a data model. As an indicator for maintainability the value of this method is limited to the the Rvse and the EAvsE metrics, see table 15.

Metric	*Validated*
RvsE	Yes
EAvsE	Yes
RAvsR	No
M:Nrel	No
1:Nrel	No
N-aryrel	No
Binary rel	No

Table 15 Piattini validation overview

5.4. Entity complexity results

The entity complexity method proposed in this thesis is an attempt to improve upon the existing methods. The existing methods all have the same flaw, if the results of these methods indicate there is a problem with the data model, it is then not possible using the results of those methods to pinpoint the problem areas in the data model. The entity complexity method aims to do just that.

5.4.1. Entity influence

This metric is defined as the difference between the Overall Complexity of the data model and the complexity of an individual entity, the difference is expressed as a percentage.

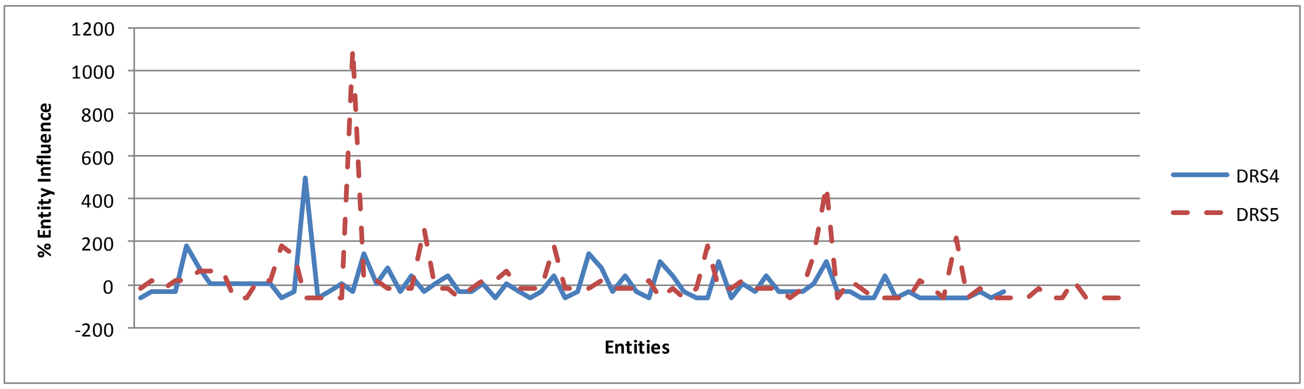
The Overall Complexity has increased slightly by almost 2% in DRS5, see table16.

Data model	*Overall Complexity (OC)*
DRS4	2.447059
DRS5	2.494845
Difference	1,95%

Table 16 Overall Complexity results

The overall complexity alone will not provide more information then was already available using the existing methods.

This is where the Entity Influence (EI) becomes useful. With this metric the individual influence of each entity on the overall complexity becomes clear. Two data models with almost the same overall complexity could have a completely different distribution of entity complexity values. The entity complexity for the DRS4 and DRS5 data models is plotted in graph 3.



Graph 3 Entity influence results

Looking at this graph it becomes clear that the distribution of the entity influence values shows distinct

changes when DRS4 is compared to DRS5. The DRS5 data model when compared to the DRS4 data model has four distinct spikes. These spikes indicate entities with a high complexity and pinpoint locations in the data model where there might be a problem with maintainability. The entity belonging to the highest spike has more than 1100% more complexity than the average entity.

The Top10 highest results for the entity influence metric are presented in table 17, the big spike in the DRS5 line of the graph can be identified as the "DRP_DEELNEMER" entity. This entity was already present in the DRS4 data model as the "DRS_DEELNEMERS" entity. This entity seems to have been modified heavily when the DRS4 data model was transformed into the DRS5 data model. The complexity of the entity relative to the average complexity of the data model is has gone from approximately 600% to 1100%. These results indicate that the complexity of the "DRP_DEELNEMER" entity is extremely high when compared with the rest of the data model. The number two in the DRS5 top10 is the "DRP_TRANSACTIE" entity. This is the entity called "DRS_AANVRAGEN" in the DRS4 version of the data model. The complexity of this entity has also more than doubled.

Table 17 shows the results of the entity complexity metric for the top10 entities, these are the 10 most complex entities in DRS4 and DRS5.

TOP 10 entity complexity DRS5	*relationships*	*complexity*	*TOP 10 entity complexity DRS4*	*#relationships*	*complexity*
DRP_DEELNEMER	30	1102.48%	DRS_DEELNEMERS	17	594.71%
DRP_TRANSACTIE	14	461.16%	DRS_AANVRAGEN	8	226.92%
DRP_DOMEINNAAM	9	260.75%	DRS_DOMEINEN	7	186.06%
DRP_TYPE_CORRESPONDENTIE	8	220.66%	DRS_LAYOUT	7	186.06%
DRP_CONTACTPERSOON	7	180.58%	DRS_PERSONEN	6	145.19%
DRP_GEBRUIKER	7	180.58%	DRS_ROL_PERSONEN	6	145.19%
DRP_NAMESERVER	7	180.58%	DRS_TYPE_AANVRAGEN	6	145.19%
DRP_CORRESPONDENTIE	6	140.50%	DRS_ABONNEMENTSPERIODES	5	104.33%
DRP_TAAL	6	140.50%	DRS_DOMEINNAAM_STATUSSEN	5	104.33%
DRP_BEOORDELING	4	60.33%	DRS_LANDEN	5	104.33%
Total	98		Total	72	

Table 17 Top10 entity complexity results

5.4.2. Entity Fan-in and Fan-Out

The entity complexity method also contains two metrics for measuring the fan-in and fan-out of an entity. These metrics were included in the method with the remark that if these metrics could not be validated or if the metrics do not provide added value, then these metrics can be removed from the method.

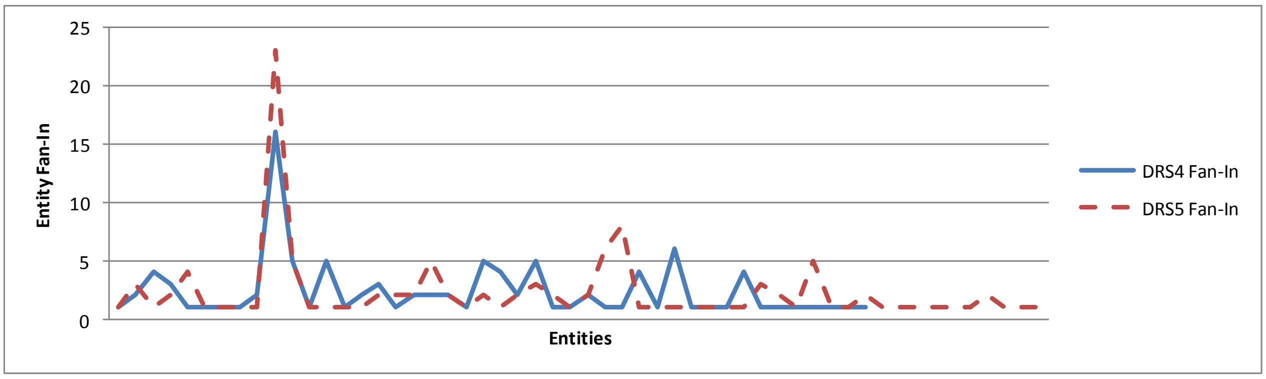
The entity fan-in is the data model equivalent of the fan-in metric which is used for measuring source code. The methods calls are replaced by relationships. The Fan-In and Fan-Out is already incorporated in the entity complexity metric, which uses the combined values of the fan-in and fan-out.

The results for the entity fan-in metric are presented in table 18.

TOP 10 Fan-In entities DRS5		*TOP 10 Fan-In entities DRS4*	
DRP_DEELNEMER	23	DRS_DEELNEMERS	16
DRP_TRANSACTIE	8	DRS_TYPE_AANVRAGEN	6
DRP_TAAL	6	DRS_DOMEINEN	5
DRP_DOMEINNAAM	5	DRS_DOMEINNAAM_STATUSSEN	5
DRP_GEBRUIKER	5	DRS_LANDEN	5
DRP_TYPE_CORRESPONDENTIE	5	DRS_PERSONEN	5
DRP_CONTACTPERSOON	4	DRS_AANVRAGEN	4
DRP_ARTIKEL	3	DRS_LAYOUT	4
DRP_NAMESERVER	3	DRS_TALEN	4
DRP_TYPE_BEPERKING	3	DRS_TYPE_CORRESPONDENTIE	4
Total	65	Total	58

Table 18 Top10 Fan-In overview

The results show only the top10 entities with the most incoming relationships. The number of incoming relationships for the top10 entities increased with 12% in the DRS5 data model. There is only one big difference between the DRS4 and DRS5 results. The "DRP_DEELNEMER" entity now has an additional 7 incoming relationships, this entity was called "DRS_DEELNEMERS" in DRS4 and the usage seems to have been modified. 7 more foreign keys reference to this entity. The results are also plotted into a graph, see graph 4.



Graph 4 Entity Fan-In results

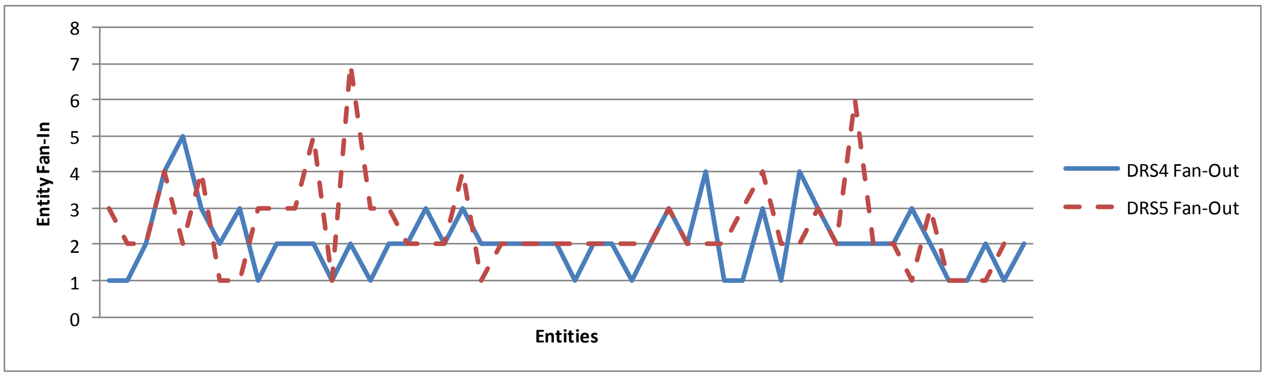
The results for the top10 fan-out metric also show an increase, the number of foreign keys have increased with 8. This equals to an increase of almost 23%. Fan-out measures the number of foreign keys of an entity. The number of foreign keys in a data model has been directly linked to the complexity of the data model [24].

The results for the entity fan-out metric are presented in table 19.

TOP 10 Fan-Out entities DRS5		*TOP 10 Fan-Out entities DRS4*	
DRP_DEELNEMER	7	DRS_ABONNEMENTSPERIODES	5
DRP_TRANSACTIE	6	DRS_AANVRAGEN	4
DRP_CORRESPONDENTIE	5	DRS_MUTATIES	4
DRP_BEOORDELING	4	DRS_ROL_PERSONEN	4
DRP_BEPERKING_VALIDATIE	4	DRS_ADRESSEN	3
DRP_DOMEINNAAM	4	DRS_CET_BRL	3
DRP_NAMESERVER	4	DRS_DOMEINNAAM_NAMESERVERS	3
DRP_ABONNEMENTSPERIODE	3	DRS_DOMEINNAAM_STATUS_VERLOPEN	3
DRP_CONTACTPERSOON	3	DRS_LAYOUT	3
DRP_CONTACTROL	3	DRS_RECHTSPERSONEN	3
Total	43	Total	35

Table 19 Top10 Fan-Out overview

See graph 5 for a plot of the results, here we see two distinct spikes in the DRS5 data. These spikes are bigger than any spike in the DRS4 data. These two spikes might be interesting for a data modeler to examine.



Graph 5 Entity Fan-Out results

The data which were used to generate the graphs and tables can be found in appendix A.

5.4.3. Analysis

The metrics in this Entity Complexity method are finer grained than the existing methods by Moody, Genero and Piattini. The results of the metrics do not consist of a single number representative for the entire data model. The results of the metrics contain one value for each entity. This data set can then be analyzed with the Lorentz curve and Gini coefficient to gain more insight into the distribution of the results. The results of the fan-in and fan-out for each entity do provide some additional value. With these metrics it is possible to detect spikes at a finer grained level than is possible with the more course entity influence metric.

5.4.4. Validation

The entity influence metric results show an increasing complexity from DRS4 to DRS5. The overall average complexity also increased by almost 2%. To validate these results the Gini coefficients for the data models have been compared to the Gini coefficients of the source code. If the results for the data model can be supported by the results of the source code analysis then the entity complexity metrics are validated. Table 20 shows the Gini coefficient for the Fan-in, Fan-Out and Entity Complexity for the DRS4 and DRS5 data models.

	Fan-in	*Fan-Out*	*Entity Complexity*
DRS5	0.446434	0.245918	0.425668
DRS4	0.425262	0.225	0.358888
Difference	0.021172	0.020918	0.06678
Difference %	4.98%	9.30%	18.61%

Table 20 Gini coefficient for DRS4 and DRS5 data models

Table 21 shows the Gini coefficient for the Fan-in, Fan-Out and the sum of the fan-in and fan-out (used to validate the entity influence metric) for the DRS4 and DRS5 source code.

	Fan-In	*Fan-Out*	*Fan-in + Fan-out*
DRS5	0.669061	0.533754	0.636405
DRS4	0.619134	0.379389	0.542725
Difference	0.049927	0.154365	0.09368
Difference %	8.06%	40.69%	17.26%

Table 21 Gini coefficient for DRS4 and DRS5 source code

The Entity influence metric has been validated by its ability to identify the entities with the most complexity i.e. the most relationships. The spikes in the graph could be traced back to entities with a very high complexity compared to the rest of the entities. There is no need to further validate this metric with the results of the source code analysis, this will be done for the fan-in and fan-out metrics.

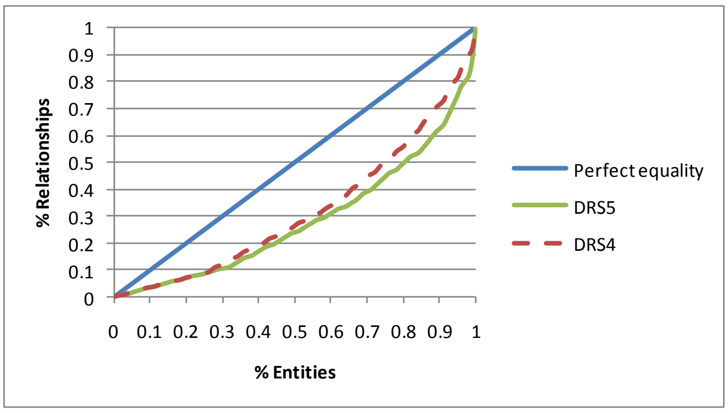
Both the data models and source code show an increase of the Gini coefficient for the all the metrics, indicating a more a-symmetrical distribution. The Gini coefficient for the entity influence metric has increased with almost 19% in the DRS5 data model. The Gini coefficient for the combined values of the source code fan-in and fan-out has also increased with over 17%. The source code metrics seems to show the same results as the data model metrics. This indicates that the data model influences the source code which is designed to make use of the data model.

The fan-in and fan-out metric results are supported by the results from the source code analysis, which also show an increase in the Gini coefficient, the fan-in and fan-out are also validated by the data model itself. The entities with the highest fan-in and fan-out count are by definition more complex than entities with a lower fan-in and fan-out. These entities have more influence on the maintainability than entities with a low fan-in and fan-out. See table 22 for an overview of the validation results for all the metrics in this method.

Graphs 6 shows the Lorentz curves for the entity influence metric, the graph shows the distribution of relationships becoming more a-symmetric in DRS5.

Metric	***Validated***
Entity influence	Yes
Entity fan-out	Yes
Entity fan-in	Yes

Table 22 Entity complexity validation overview



Graph 6 Entity complexity distribution

6. Interpretation of the results

In this thesis four methods have been used to determine the maintainability of the DRS data models, the results of these methods have been described in the previous chapter. This chapter will interpret these results and compare the methods with each other.

6.1. Moody

The Moody method contains three metrics which can be used to determine the maintainability. The other metrics were not objective or turned out not to contribute to maintainability after being mapped onto the ISO 9126 quality model. To validate the results of the Moody method, the results were compared to the results of the source code analysis. The results of this comparison showed that the increase in the number of entities, was supported by an increase in the number of classes in the source code. The other two metrics also included the number of attributes and relations. The results for these metrics show a large increase.

To support the results of these Moody metrics, the fan-in and fan-out metric results of the source code analysis were used. A comparison showed an decrease in fan-in and fan-out in the source code, the Moody metrics showed an increase. This means the second and third Moody metric could not be validated with the source code results. The conclusion must be that only the first metric which measures the number of entities could be validated. This metric shows an increase in complexity.

6.2. Genero

The Genero method was found to contain eight metrics which contribute to the maintainability of a data model. Of those, two metrics could not be used with the DRS data models because the constructs which are measured by those two metrics are not present in the data models. This means the Genero method is effectively evaluated with the use of the remaining 6 metrics. These metrics measure the number of entities, attributes and relationships. The difference when compared to the Moody metrics is that these metrics are finder grained with regard to relationships. The relationships are sub-divided into multiple types. The results of this method are validated with the use of the results of the source code analysis. The NE metric which counts the number of entities is supported by the increased number of classes in the source code. The other metrics which measure the attributes and different types of relationships could not be validated. The conclusion must be that only the first metric which measures the number of entities could be validated. This metric is the same metric as the first metric of the Moody metric and is a very basic measure.

6.3. Piattini

The method proposed by Piattini uses a different approach than the previous two methods, Piattini has proposed metrics which calculate a ratio between structural elements of a data model. This has some advantages compared to the methods of Moody and Genero. Using ratios makes it possible to compare the results for different data models with each other more easily. To validate the results of these metrics a different validation method is used than for the other two methods. To determine if the ratios which result from the metrics in this metric are accurate and valid, a Lorentz curve and GIni coefficient have been calculated to determine the attribute and relationship distribution. The Lorentz curves show a change between DRS4 and DRS5, the distribution has become less equal. The Gini coefficient which is derived from the Lorentz curve shows a quantified value for the change in distribution for the attributes and relationships. When the Gini coefficients for the attributes and relationships are compared with the RvsE and EAvsE metrics, we find that they have both increased. The conclusion is that the the RvsE and EAvsE metrics are supported by the Gini coefficients of the attribute and relationship distribution. The other metrics in the method cannot be validated.

6.4. Weak points

The three methods proposed by Moody, Genero and Piattini are all partially confimed, they can be used to determine some value of maintainability for a data model. The results from the methods all results into different types of output. The output of one methode cannot easily be compared with the output of another method. The methods also do not provide any help in interpreting the results, none of the methods describes a scale which can be used to interpret the results. If a method is used to compare two data models and the results for the two data models differ, then the method should provide information about where the differences are found and what they mean. This is not possible with the above mentioned methods. Therefore the most important weakness of these methods is the inability to pinpoint the problem areas of a data model. If the results of a method show that the maintainability is low, then the first question should be “why“. This “why” question cannot be answered by the analyzed methods. This was the reason for proposing a new method in this thesis. This new method the “Entity complexity” method does provide information about the weak points of a data model.

6.5. Entity complexity

The Entity complexity method is finer grained than the other metrics which in this case means that the results do not describe the model but the entities in the model. The results of the metrics when plotted in a diagram show spikes where the complexity of a structure is higher than the average complexity. In the case of the DRS data models big spikes were found in the diagrams. These spikes could be tracked back to individual entities in the data model, these are the entities a data modeler will be interested in. The other two metrics in this method are the fan-in and fan-out. These measure the incoming and outgoing relationships for each entity. The graphs for these metrics also show spikes. These spikes represent entities with either a high fan-in or fan-out. The fan-in and fan-out are sub metrics their results are incorporated in the entity complexity method. The spikes that were found in the fan-in and fan-out metric results were also found in the entity complexity result.

6.6. Summary

Table 23 shows an overview of the evaluation methods and the number of metrics of the method which could be validated.

Method	*# Metrics*	*# Validated metrics*
Moody	3	1
Genero	8	1
Piattini	7	2
Entity complexity	3	3

Table 23 Summary of method validation

There is no clear “winner”. All the methods used in this thesis gave the same indication of a decreasing maintainability, see table 24 for the results. The “Entity Complexity” method is the only method which can be used to determine the weak points of a data model. Some of the methods contained metrics which do not provide any added value and can be removed. Using the Gini coefficient to determine the distribution of relationships over the entities of a data model did turn out to be a success. With the use of the Gini coefficient it is possible to determine if the distribution of relationships becomes more a-symmetric. An increasing a-symmetric distribution indicates that some entities have many relationships and have become more complex.

Method	*Results for DRS5*
Moody	Decreased maintainability
Genero	Decreased maintainability
Piattini	Decreased maintainability
Entity complexity	Decreased maintainability

Table 24 Maintainability results of all the methods

All the methods indicate the same result, namely a lower maintainability for the DRS5 data model. This is supported by results of the source code analysis. From the results of this research the following metrics seem to have the best results and could be combined into a new method. See table 25 for a proposal of metrics which could be included in a new method. This method uses the metrics which gave the best results, which are objective and which can be calculated automatically.

Method	*Metric*	*Reason*
Moody and Gen+ero	Number of entities	Easy to calculate and can be compared with source code.
Piattini	Ratio of relationships to entities Ratio of attributes to entities	Easy to calculate and the results can be compared with the results for other data models.
Entity Complexity	Entity influence	Easy to calculate and provides information about the entities with a high complexity.

Table 25 Proposal for new maintainability method

7. Conclusions

The results of this research show that it is possible to determine the maintainability of a data model by evaluating the properties of a data model. This thesis research has looked at the existing methods for evaluating data models for their maintainability. Three methods were found which could be used easily, reliably and repeatedly to determine the maintainability of a data model. The application of these three methods results in a measure of maintainability. The problem is that these scores cannot be compared with each other and with other data models. It is also not possible to pinpoint the problem areas in a data model. To improve upon these existing methods, a new method the Entity Complexity method has been proposed in this research.

These four methods were used to measure the maintainability of the DRS data model, two versions of the DRS data model were used. The methods detected differences between the two data models. The DRS5 data model was more complex according to all the methods. This implies that the maintainability of the DRS5 data model is lower than the maintainability of the DRS4 data model.

7.1. Answers to the research questions

The research in this thesis was done in order to find the answer to the main research question. To help answer the main research question three sub-questions were designed.

7.1.1. Main question: How to compare the maintainability of a data models?

The methods for calculating the maintainability have been found in answering sub question Q1, these methods use the structural properties of a data model as their input. The output of these different methods are incompatible with each other, this makes it difficult to compare the results directly with each other. This research has used the Gini coefficient and results of a source code analysis to compare the results of the metrics. This made it possible to compare the results of different data models with each other. It was found that the results of the data model metrics were supported by the results of the source code analysis.

7.1.2. Q1: What is "maintainability" in the context of a data model?

This question was answered by using the existing definition described in the ISO 9126 model for software quality. This is a widely accepted definition, The Genero and Piattini and the Entity Complexity methods were designed to measure maintainability. The method proposed by Moody contained many "quality" metrics that did not relate to maintainability. The Moody and Genero method also contain metrics which are not objectively measureable. The metrics that did relate to maintainability were identified by mapping them onto the ISO 9126 model. The other metrics including the subjective metrics were not used in this research.

7.1.3. Q2: How can you measure the maintainability of a data model?

This question has been answered by searching for measurement methods in the existing literature. Three methods were found. These methods can be used to measure the quality of a data model. With the use of the ISO 9126 quality standard the definition of maintainability was determined. Metrics in the methods that did not influence the maintainability of a data model were not used when the method was applied to the DRS data models. The methods are based on the complexity theory which state that the more elements and relationships between those elements exist the harder it will become to understand the whole picture. This insight was already used for a long time in software engineering. Dijkstra already warned against too much complexity. The metrics that were used to measure the data model maintainability. All use the structural properties (entities, attributes and relationships) of a data model to determine the level of maintainability. Not all the metrics use them in the same way though, some just add up all the structures together to come up with a number, other metrics will calculate a ratio between two types of structures. This leads to different types of results for each method and makes it hard to compare the results of different methods with each other.

7.1.4. Q3: How to determine the effectiveness and correctness of a measurement method?

The results of the data model metrics which were used in this research have been validated with the following methods.

a) Results of the DRS source code analysis
b) Analysis of the distribution of complexity in the data models and source code with the Gini coefficient.

Determining if the results of a method are correct is done with the use of an analysis of the DRS source code. This source code analysis is not a part of this research, it is performed by the Software Improvement Group at the request of the host organization were this research was done.
The results of the data model measurements were supported by the results of the source code analysis. Both show an increasing complexity, complexity highly effect maintainability. Using the Gini coefficient to determine the distribution of complexity in both the data model and source code, it was found that the distribution has become more a-symmetric in both the data models and the source code.

7.2. Future work

The methods were used on only two data models and the validation of the results was done with the results of a source code analysis of only two versions of the DRS application. This small set of data models and applications means the results of this thesis need to be confirmed by repeating the research on more data models and applications.

In this thesis a new method for measuring data model maintainability was proposed, this method contained the best metrics from all the analyzed methods. This new method still needs to be validated.

Bibliography

[1] Moody D.L., Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering, Volume 55, Issue 3, Quality in conceptual modeling - Five examples of the state of art, December 2005, Pages 243-276.

[2] Piattini, M., Genero, M. and Calero, C. Data Model Metrics. In Handbook of Software Engineering and Knowledge Engineering: Emerging Technologies, World Scientific, 2002.

[3] Genero M, Poels G and Piattini M (2002) Defining and Validating Measures for Conceptual Data Model Quality. Lecture Notes in Computer Science 2348, 724-727.

[4] Moody, D. L., Metrics for evaluating the quality of entity relationship models, Lecture Notes In Computer Science; Vol. 1507, Proceedings of the 17th International Conference on Conceptual Modeling, Pages: 211 – 225, 1998

[5] Moody, D. L. and Shanks, G. G. 1994. What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In Proceedings of the13th international Conference on the Entity-Relationship Approach (December 13 - 16, 1994)

[6] Moody, D. L., Measuring the quality of data models: An Empirical evaluation of the use of Quality Metrics in practice, Proceedings of the 11th European Conference on Information Systems, ECIS 2003, Naples, Italy 16-21 June 2003

[7] Moody D. L., Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering, Volume 55, Issue 3, Quality in conceptual modeling - Five examples of the state of art, December 2005, Pages 243-276

[8] Moody D.L., Graeme G. Shanks, Improving the quality of data models: empirical validation of a quality management framework, Information Systems, Volume 28, Issue 6, September 2003, Pages 619-650, ISSN 0306-4379, DOI: 10.1016/S0306-4379(02)00043-1.

[9] Someswar Kesh, Evaluating the quality of entity relationship models, Information and Software Technology, Volume 37, Issue 12, 1995, Pages 681-689

[10] E. F. CODD, Relational Model of Data for Large Shared Data Banks, IBM Research Laboratory, San Jose, California, 1970

[11] T. McCabe, A Complexity Measure, IEEE Transactions on software engineering, Volume SE-2, NO. 4, December 1976.

[12] ISO/IEC, ISO/IEC Standard 9126: Software Product Quality, International Standards Organization (ISO), International Electrotechnical Commission (IEC), 2001.

[13] De Marco, T.: Controlling software projects. Management, measurement & estimation. Barry W. Boehm , Yourdon Press, Englewood Cliffs (1982)

[14] Kuipers, T., and Visser, J., Maintainability Index revisited – position paper. In 11th European Conference on Software Maintenance and Reengineering, 2007.

[15] Piattini, M., Calero, C., and Genero, M. 2001. Table Oriented Metrics for Relational Databases. Software Quality Control 9, 2 (Jun. 2001), 79-97

[16] McConnell S., Code Complete, 2nd Edition. Redmond, Wa.: Microsoft Press, 2004, chapter 19-20

[17] C.J. Date, An Introduction to Database Systems 8th Edition, 2004

[18] Codd, E.F. ,Further Normalization of the Data Base Relational Model, Randall J. Rustin(ed) Data Base Systems, Courant Computer Science Symposia Series 6, Englewood Cliffs, NJ, 1972

[19] Heitlager, I., Kuipers, T. and Visser, J., A Practical Model for Measuring Maintainability. In 6th International Conference on the Quality of Information and Communications Technology, 2007, pp. 30-39.

[20] Mario Piatini, Marcela Genero, Luis Jimenez, A metric-based approach for predicting conceptual data models maintainability, International Journal of Software Engineering and Knowledge Engineering, Volume: 11, Issue: 6(2001) pp. 703-729

[21] L. Briand, J. Wüst, H. Lounis, Replicated case studies for investigating quality factors in object-oriented designs, Empirical Software Engineering 6 (volume 1) (11-58) – 2001

[22] R. Vasa, M. Lumpe, P. Branch, and O. Nierstrasz, “Comparative Analysis of Evolving Software Systems Using the Gini Coefficient,” in Proceedings of 25th IEEE International Conference on Software Maintenance (ICSM ’09). Edmonton, Alberta: IEEE Computer Society, Sep. 2009, pp. 179–188.

[23] Genero, M., Poels, G., and Piattini, M. 2008. Defining and validating metrics for assessing the understandability of entity-relationship diagrams. Data Knowl. Eng. 64, 3 (Mar. 2008), 534-557.

[24] Coral Calero, Mario Piattini, Marcela Genero, Empirical validation of referential integrity metrics, Information and Software Technology, Volume 43, Issue 15, 23 December 2001, Pages 949-957

[25] Piattini, M., Measuring the quality of entity relationship diagrams. ER2000 Conference, 2000.

[26] C.J. Date , An introduction to database systems, Pearson Education; 8 edition (7 Aug 2003)

Appendix A – Raw data

Data model structural property totals

Data model	# entities	# relationships	# attributes
DRS5	97	121	1107
DRS4	85	104	554

DRS4 entities Fan-in

DRS_DEELNEMERS	16
DRS_TYPE_AANVRAGEN	6
DRS_DOMEINEN	5
DRS_DOMEINNAAM_STATUSSEN	5
DRS_LANDEN	5
DRS_PERSONEN	5
DRS_AANVRAGEN	4
DRS_LAYOUT	4
DRS_TALEN	4
DRS_TYPE_CORRESPONDENTIE	4
DRS_ARTIKELEN	3
DRS_FOUTMELDINGEN	3
DRS_AANVRAAG_STATUSSEN	2
DRS_DEELNEMER_STATUSSEN	2
DRS_FACTUURREGELS	2
DRS_GEBRUIKERS	2
DRS_GEBRUIKER_ROLLEN	2
DRS_INSTANTIE_JUR_BEPERKINGEN	2
DRS_NAMESERVERS	2
DRS_ROL_PERSONEN	2
DRS_AANVRAAG_FACTURERINGEN	1
DRS_BEDRIJFSREGELS	1
DRS_CORRESPONDENTIES	1
DRS_CORRESPONDENTIE_ELEMENTEN	1
DRS_DEELNEMER_STATUS_VERLOPEN	1
DRS_DOMEINNAAM_STATUS_VERLOPEN	1
DRS_FACTUREN	1
DRS_FUNCTIES	1
DRS_INSTANTIES	1
DRS_RECHTSVORMEN	1
DRS_REDEN_WIJZ_DEELNMR_STATUS	1
DRS_ROLLEN	1
DRS_SOORT_JURIDISCHE_BEPERKING	1
DRS_TEKSTEN	1
DRS_TYPE_ABONNEMENTEN	1
DRS_TYPE_ADRESSEN	1
DRS_TYPE_BEDRIJFSREGEL	1
DRS_TYPE_DEFAULTS	1

DRS_TYPE_MUTATIES	1
DRS_TYPE_NAMESERVERS	1
DRS_TYPE_REGISTRATIES	1
DRS_TYPE_TEKSTEN	1
DRS_TYPE_TELECOM_ADRESSEN	1
DRS_TYPE_X_HEADERS	1

DRS4 entities Fan-out

DRS_AANVRAAG_DOMEINNAMEN	1
DRS_AANVRAAG_FACTURERINGEN	1
DRS_AANVRAAG_STATUS_VERLOPEN	2
DRS_AANVRAGEN	4
DRS_ABONNEMENTSPERIODES	5
DRS_ADRESSEN	3
DRS_BEDRIJFSREGELS	2
DRS_CET_BRL	3
DRS_CORRESPONDENTIES	2
DRS_CORRESPONDENTIE_ELEMENTEN	2
DRS_CORR_ELEMENT_NAMEN	1
DRS_CRE_AVG	2
DRS_DEELNEMERS	1
DRS_DEELNEMER_HANDLE	1
DRS_DEELNEMER_STATUS_VERLOPEN	2
DRS_DEFAULT_WAARDEN	2
DRS_DOMEINEN	2
DRS_DOMEINNAAM_NAMESERVERS	3
DRS_DOMEINNAAM_STATUS_OVG	2
DRS_DOMEINNAAM_STATUS_VERLOPEN	3
DRS_DSP_RWS	2
DRS_FACTUREN	2
DRS_FACTUURREGELS	2
DRS_FOUTMELDING_TEKSTEN	2
DRS_FTE_GRL	2
DRS_GEBRUIKERS	1
DRS_GRL_GBR	2
DRS_INSTANTIE_JUR_BEPERKINGEN	2
DRS_IP_ADRESSEN	1
DRS_JURIDISCHE_BEPERKINGEN	2
DRS_LAYOUT	3
DRS_LYT_TKN	2
DRS_MUTATIES	4
DRS_NATUURLIJKE_PERSONEN	1
DRS_PERSONEN	1
DRS_RECHTSPERSONEN	3
DRS_RECHTSVORMEN	1
DRS_ROL_PERSONEN	4

DRS_ROL_PERSOON_DOMEINNAMEN	3
DRS_SUBREPORT	2
DRS_TAG_PER_DOMEINSTATUS	2
DRS_TAG_PER_TYPE_AANVRAAG	2
DRS_TEKSTEN	2
DRS_TELECOM_ADRESSEN	3
DRS_TYPE_AANVRG_JUR_BEPERKING	2
DRS_TYPE_ABONNEMENTEN	1
DRS_TYPE_MUTATIES	1
DRS_WHOIS_LOGS	2
DRS_WHOIS_ONTHEFFINGEN	1
DRS_X_HEADERS	2

DRS5 entities Fan-in

DRP_DEELNEMER	23
DRP_TRANSACTIE	8
DRP_TAAL	6
DRP_DOMEINNAAM	5
DRP_GEBRUIKER	5
DRP_TYPE_CORRESPONDENTIE	5
DRP_CONTACTPERSOON	4
DRP_ARTIKEL	3
DRP_NAMESERVER	3
DRP_TYPE_BEPERKING	3
DRP_BEPERKING	2
DRP_FACTUURREGEL	2
DRP_FOUT	2
DRP_FUNCTIE	2
DRP_GEBRUIKERSROL	2
DRP_INSTANTIE	2
DRP_LAND	2
DRP_RECHTSVORM	2
DRP_REGISTRY	2
DRP_TYPE_BEPERKING_ACTIE	2
DRP_TYPE_DEELNEMERSTATUS	2
DRP_TYPE_TRANSACTIE	2
DRP_AANVRAAG_FACTURERING	1
DRP_BEDRIJFSREGEL	1
DRP_CORRESPONDENTIE	1
DRP_CORR_BESTEMMING	1
DRP_CORR_RICHTING	1
DRP_CORR_SOORT	1
DRP_DOMEINNAAM_REGISTR_PER	1
DRP_DYNAMISCH_VELD	1

DRP_EMAIL_LAYOUT	1
DRP_FACTUUR	1
DRP_GESLACHT	1
DRP_KLASSEATTRIBUUT	1
DRP_REDEN_DEELNEMERSTATUS	1
DRP_TRANSACTIEBRON	1
DRP_TRANSACTIE_ELEMENT	1
DRP_TYPE_ABONNEMENT	1
DRP_TYPE_ADRESOPMAAK	1
DRP_TYPE_BEDRIJFSREGEL	1
DRP_TYPE_BEOORDELING	1
DRP_TYPE_BEOORDELING_RESULT	1
DRP_TYPE_CONTACTROL	1
DRP_TYPE_DEELNEMERCONTACT	1
DRP_TYPE_DEELNEMER_ADRES	1
DRP_TYPE_FACTUURPERIODE	1
DRP_TYPE_GEBRUIKERSACCOUNT	1
DRP_TYPE_INVULHULPWAARDE	1
DRP_TYPE_MUTATIE	1
DRP_TYPE_NAMESERVER_IP	1
DRP_TYPE_NOTITIE	1
DRP_TYPE_TRANSACTIEBERICHT	1
DRP_TYPE_TRANSACTIESTATUS	1
DRP_TYPE_TRANSACTIE_ELEMENT	1

DRS5 entities Fan-out

DRP_ABONNEMENTSPERIODE	3
DRP_AUTHORITATIEF_NAMESERVER	2
DRP_BEDRIJFSREGEL	2
DRP_BEOORDELING	4
DRP_BEPERKING	2
DRP_BEPERKING_VALIDATIE	4
DRP_BERICHTENQUEUE	1
DRP_CODE_OMSCHRIJVING	1
DRP_CONTACTPERSOON	3
DRP_CONTACTROL	3
DRP_CONTACT_ADRES	3
DRP_CORRESPONDENTIE	5
DRP_CORRESPONDENTIE_DATA	1
DRP_DEELNEMER	7
DRP_DEELNEMER_ADRES	3
DRP_DEELNEMER_CONTACTPERSOON	3
DRP_DEELNEMER_FUNCTIE	2
DRP_DEELNEMER_STATUS_VERLOOP	2
DRP_DEELNEMER_TRANSACTIE	2

DRP_DOMEINNAAM	4
DRP_DOMEINNAAM_REGISTR_PER	1
DRP_DOMEINNAAM_TRANSACTIE	2
DRP_EMAIL_DYNAMISCH_VELD	2
DRP_EMAIL_LAYOUT	2
DRP_FACTUUR	2
DRP_FACTUURREGEL	2
DRP_FCFS	2
DRP_GEBRUIKER	2
DRP_GEBRUIKERSROL_FUNCTIE	2
DRP_GEBRUIKER_GEBRUIKERSROL	2
DRP_GEKOPPELDE_BEDRIJFSREGEL	3
DRP_INGESCHREVEN_BEPERKING	2
DRP_INSTANTIE_TYPE_BEPERKING	2
DRP_INVULHULPWAARDE	2
DRP_MUTATIE	3
DRP_NAMESERVER	4
DRP_NAMESERVER_IPADRES	2
DRP_NAMESERVER_TRANSACTIE	2
DRP_NOTITIE	3
DRP_RAPPORT_LAYOUT	2
DRP_TRANSACTIE	6
DRP_TRANSACTIEBERICHT	2
DRP_TRANSACTIE_ELEMENT	2
DRP_TYPE_ABONNEMENT	1
DRP_TYPE_CORRESPONDENTIE	3
DRP_TYPE_MUTATIE	1
DRP_TYPE_TRANSACTIE	1
DRP_WHOIS_ONTHEFFING	1
DRP_XML_LAYOUT	2