Formalizing Data Quality and its Influence on Business Performance

---

Master's Thesis, November 19, 2013

Mark Zijdemans

Formalizing Data Quality and its Influence on Business Performance

MASTER'S THESIS

submitted in partial fulfilment of
the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER SCIENCE
TRACK INFORMATION ARCHITECTURE

by

Mark Zijdemans
born in Dordrecht, The Netherlands

Ministry of Defence

Web Information Systems Group
Department of Software Technology
Faculty EEMCS
Delft University of Technology
Delft, The Netherlands
http://wis.ewi.tudelft.nl

Central Data Management Office
Defence Materiel Organisation

Ministry of Defence
The Hague, The Netherlands
http://www.defensie.nl

Formalizing Data Quality and its Influence on Business Performance

Author:	Mark Zijdemans
Student id:	1217771
Email:	mark@zijdemans.com

Abstract

Organisations make decisions every day. Some of these are for example triggers within automated process, others are decisions made by persons. Both have at least one thing in common, they are based on the available data. The quality of this data may create a discrepancy between the set of actions taken and the set that should have been taken.

This work investigates the possibility to calculate the data uncertainty at decision level, based on knowledge gained by use of data quality metrics at the data level. The data quality research field contains a rich set of methods to determine the quality of the data within an organisation. This project sets out to show how to model this quality using data uncertainty, specifically using probabilistic databases. This in such a way the information the data quality metrics provide is maintained within this model. Furthermore, an overall system is designed in which the data and the uncertainty can be stored, decisions can be modelled and data used in these decision can be linked to the given decision.

This framework can then provide the data values needed for the decision accompanied with the uncertainty of those data values. Finally, this works aims to show that it is possible to predict a loss in business performance based on the data quality score, data retrieval steps and the decisions made.

Thesis Committee:

Chair:	Prof. dr. ir. G.J.P.M. Houben, Faculty EEMCS, TUDelft
University Supervisor:	Dr. ir. A.J.H Hidders, Faculty EEMCS, TUDelft
Company Supervisor:	Cdr. A.V. Braanker, Royal Netherlands Ministry of Defence
Committee Member:	Prof. dr. ir. J. van den Berg, Faculty TPM, TUDelft

1 Preface

"War is ninety percent information..."
Napoleon

This work started as a quest to answer the question: "How to get data quality information about the data used in decisions?". Of course this problem can be solved in numerous ways. Even more ways if one removes the requirement of data quality and just sets out to solve the problem of uncertainty during decisions. But this project has the requirement of fitting in the current efforts of the Netherlands Ministry of Defence and the Computer Science Department of Delft University of Technology.

Of course, as a research project this project will not provide something that can be used by a given organisation. In other words, this project will not provide a software tool that will solve all the problems. The results will show if it is possible to solve the problem and it will show that the taken approach will get us closer to a solution. It might also show the chosen way is a dead-end. The most important results this work hopes to achieve are:

1. That the Ministry of Defence learns from the project. That it is possible to determine the uncertainty of information at decision level.

2. That future students have a point to start from in continuing this work and hopefully create a ready to use software product which can model data quality and provide the uncertainty score at decision level.

The company supervising of this project was done by Cdr. A.V. Braanker of the Central Data Management Office (CBG) of the Royal Netherlands Ministry of Defence. I would like to thank him for keeping the project in-line with the needs of the organisation. And for teaching me how to communicate my ideas with colleagues and managers. Cdr. Braanker also showed me that succinct communication saves time and helps get thing done. Finally I would like to tell how much the insight in sentence and paragraph construction helps to convey ideas and facts.

Dr. ir. A.J.H. Hidders was the academic supervisor of this project. I thank him for making sure this project is still called an academic research project. He also helped in beforehand determining interesting research questions in light of the organisations needs and available research fields. Dr Hidders also helped with issues that arose during this project.

Others I would like to thank for their support during this project are, in no particular order: M. Gillissen (KPU), D. van Deuveren (KPU), D. Dirven (KVL), R. Anous (KVL), M. Wortelboer (MOD), P. Gelauff (Cendris), J. Berg (MOD), Maj. J. Bremen (MOD), R. van Mierop (MOD), R. van Dalen (MOD), M. Bakker (MOD), F. de Vries (MOD) and the rest of the colleagues at the CBG. The CBG is part of the Defence Materiel Organisation (DMO).

Contents

2 Executive Summary

Knowing the quality of data used in a decision is very important in order to pick the best option. Knowing which data must be cleansed, in order to improve the information needed for a given decision, is invaluable. The Ministry of Defence presented this problem, which this work sets out to solve within the scope of a master's thesis.

Besides the fact that the determined data quality score shows an organisation what data to cleans. A system that can be used to trace the root cause of the business performance problems due to data quality is very interesting. This way, a subset of the data can be cleansed using specific rules to directly improve business performance.

A system that can determine which option to pick in a given a decision, based on the available data and uncertainty information will help an organisation understand its data quality problem. Based on possible outcomes, it is even possible to predict gain and loss due to data quality issues.

The Ministry of Defence has huge amounts of data, the quality of that data is sometimes questioned. Mostly due to operational problems, of which the root cause seemed to be the lack of quality of the data.

The Ministry of Defence is improving its data management and data governance efforts. This is done by centralising the master data and its maintenance. A mature data quality project is part of this larger project. However, migrating the data introduces new issues.

This research project sets out to design an overall system which predicts the quality of the data at decision level based on the quality of the master-data-set and the data integration steps needed to retrieve the data needed for the decision. The core research is done by determining how to use data uncertainty techniques to model the data quality such that at every level, from data to decision, the quality of the data is correctly predicted.

For this research the use of data uncertainty techniques to model the data quality was chosen. Since the available tools for data uncertainty proved to contain very interesting possibilities in order to determine uncertainty of results of relational algebra operations. Applications of relation algebra operations are for example using SQL and a database system.

Data quality and data uncertainty are not the same, but by definition they have something in common, they are both related to the differences between the data-set and the real world object. A data quality score shows how well the data represents the real world object, as defined by the needs of the data client. Data uncertainty shows the probability that the data is correct. Data uncertainty can also be used to model an uncertain phenomenon, in this case the real world object is uncertain. This last property will not be used much in this project.

If it is possible to translate the data quality score to a data uncertainty score, a probabilistic database and probabilistic calculus in itself, could be used to determine the uncertainty at decision level. Another way to determine the quality of the data at decision level would be to check that data itself. However,

both the temporal nature of that data and the time it takes to run a data quality measurement set show that such an approach is not feasible. Data quality checks on master data, data which is more or less a constant, can run silently in the background. Combining this with the designed system and the quality of resulting data is determined on the fly.

More insight into the problems caused by data quality provides an incentive for data cleansing actions. However, cleansing the data is just one of the options to improve the data quality. Resilience against tainted input is another important option, in a general case this is less expensive than cleansing afterwards.

At the Ministry of Defence they employ over 400 rules to determine the quality of the data. For any given subset of that data, not all rules are used. Some of those rules are specific to certain data sets. The first attempt to model the quality using data uncertainty was to take the mean percentage of individual rule scorings.

The data quality monitoring tool of the Ministry of Defence provides a list of all the rules which apply to a given data set and the percentage of correct data-item in that set, according to the specified rule. Correct is in this case the fact that the rules could not determine if the item was incorrect. So "correct" in this case, does not mean "of perfect quality". The mean of all those rules is then determined by adding up all the percentages and divide that by the number of rules. The resulting score was shown not to be an uncertainty score, since it does not show the probability that any given data item within the set was correct, according to the specified rule. So the rules must be used in a different way, in order to determine the data uncertainty.

A data quality rule can be seen as a filter, separating the incorrect data item from the rest of the data. However, the rest of the data is not necessarily correct, without entering a discussion whether data in a database can be 100% correct, the Ministry of Defence assumes that a data item is correct if it is not found to be incorrect according to at least one of the rules. Additionally, the Ministry of Defence assumes the data is correct if the person responsible for the data claims it is. In this project the assumption is made that there is still uncertainty whether the data is correct, given it is not found to be incorrect.

Throughout the overall concept designed to solve the problem, uncertainty and quality are defined in different levels of that concept. First, there is the data level, this can be seen as the source data or database. At this level the Ministry of Defence measures the data quality. At the end of the process there is the decision level, this can be seen as the moment of decision making. The data available at this moment is created based on data available in source systems and data integration steps. Data integration is the process of combining data from different sources into a new data product. Sometimes this data product is referred to as "information".

Filtering, based on quality rules, can be done at a record level or individual value level. A set of data quality rules can thus be used to determine the percentage of incorrect items within the data set, at a given moment in time. If this value is somewhat constant, or a trend can be determined, then a prediction can be made of the score at a given moment. This percentage qualifies as a data

uncertainty score, since it represents the probability that a given item is correct.

A data quality score shows how well the data represents the real world. For example if 9 out of 10 items is shown to be correct and the 1 remaining item is shown to be incorrect, a score of 9/10, 0.9 or 90% would be a good way to show the quality of that data set. If, based on a sample set of historical trends, this score is a prediction of the quality of the current set, one does not know which items are actually incorrect. In this case the probability of picking an incorrect items is $1-0.9 = 0.1$. In other words, there is a 0.9 probability an item is picked which is correct, as far as the data quality rules can determine.

Determining the quality like this results in a lower score then before. Using the first method, the method of the mean data quality score, a data quality score of 98% was found. With the second method a score of 71% was found. However, by checking this with the result-set data, this second method was shown to provide better predictions of the uncertainty of the data.

However, a quality score of 50% after a few simple integrations seemed a bit too uncertain. The data is daily used by an organisation, given the fact that this organisation still exists, one can assume that it does not make one mistake in every two decisions. Besides the fact that not all decisions are negatively influenced by data that is not completely correct, there was still no relation defined between rules and the data client. Quality is in the eye of the beholder one might say.

The next step was to determine the data quality, solely based on rules which determine the incorrectness of the data actually used by the data client. In other words, a value is incorrect if it is incorrect according to a given data quality rule, such that both the data is used by the client and the rule expresses a need of the data client. Of course, for generic data cleansing efforts, every rule is important, but for a given data client this is not the case.

By using this new data quality score within the designed system, the predicted data quality at decision level was between 92% and 98%, which is very close to the performance problems encountered by the data client. This project shows that the result of the system is a good worst case prediction of the quality of the data used for a decision. More detailed data quality information and dependency information will make the prediction more precise. This could not be shown within this project, since that detailed information was not available at the time. Additionally, the determination and maintenance of such detailed quality information is probably quite difficult. More detailed quality information would, for example, encompass dependency information among items or individual quality scores.

Besides the fact that the determined score was shown to be close to what actually happens, such a system can be used to trace the root cause of the data quality problem to be solved, in order to improve business performance. Finally, the costs of possible mistakes made, due to lack of data quality, can be predicted by the designed system. This shows the costs of the data quality problem and can be used to create a budget to cleans the data.

3 Introduction

"Uncertainty is not simply the absence of knowledge..."
Walken et al. [28]

This section gives an introduction to the thesis, here the research questions are explained, the research fields are discussed and the problem is explained. First, the context of the project will be explained, this background section will explain the current situation at the Royal Netherlands Ministry of Defence (MOD). As well as the research fields with which this thesis is concerned, these fields will be data quality and data uncertainty. Within the paragraphs on data uncertainty the field of probabilistic databases and probability and statistics will be explained briefly. After this there is a section on the research objective. This objective will be explained in short. Within this section the supporting questions will be discussed, since one cannot just answer the overall question. These questions helped shape the rest of this document and answering these questions brought us closer to the objective. As mentioned earlier, not everything will be solved, the scope of this work will be set next. The contribution to science this work hopes to make is discussed in Section 3.4. And finally, the outline of this document is provided at the end of this introduction section.

3.1 Background

To place this project in the correct context, this section provides some background on the situation of the MOD and the current state of the research fields. It starts with a section on the MOD and the problem sketched by the MOD. Followed by an introduction into the research fields, which will be discussed in more depth later in this work.

3.1.1 Royal Netherlands Ministry of Defence

The problem as explained by the MOD is that they cannot review the quality of the data used at the moment of the decision. They can only review, at a separate moment, the quality of the source data. Knowing this information can greatly improve the way decisions are made. Data used during decisions is created temporary to support the decision maker, this data is sometimes called information since it is actionable. A system that determines the quality at decision level, must be able to determine the data quality at decision level, based on the process used to retrieve that data.

Once a system is in place that links the data quality of the source data to the quality of the data used in a decision, such a system could be used to improve the impact of data cleansing actions. Since it can be then shown which data is of influence to the quality of the result data. Additionally, if costs of making a wrong decision are known and the probability is known of making that mistake, it is possible to determine the costs of the data quality problem.

Throughout the overall concept designed to solve this problem, uncertainty and quality are defined in different levels of that concept. First, there is the data level, this can be seen as the source data or databases, at this level the MOD measures the data quality. At the end of the process there is the decision it self, the decision level. The data available at this moment is created based on data available in source systems and data integration steps. Data integration is the process of combining data from different sources into a new information product.

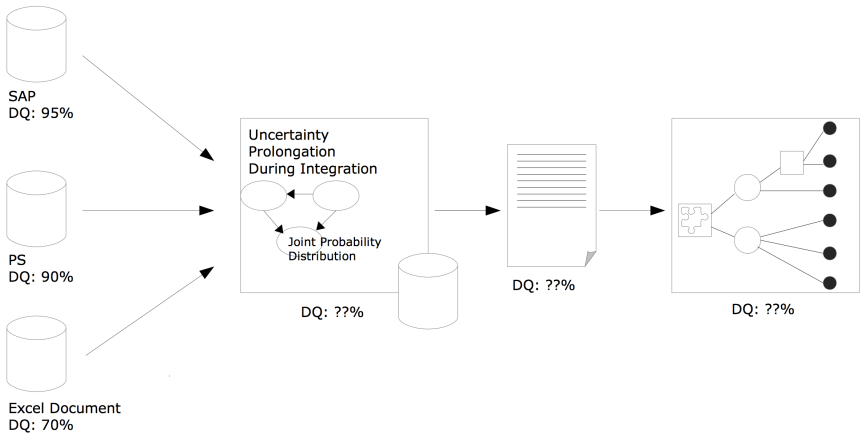

Figure 1: A sketch of the "context"

Figure 1 shows the context as defined by the MOD. It shows the problem of heterogeneous sources from which data is being combined into a report. This report can be seen as result data or information. Within this thesis we will use the term "data", however, information is a good term to indicate an actionable data product made from source data. In Section 5.1 the definition of data will be explained.

Using the "information" product of the data integration process, a decision can be made. In Figure 1 the decision looks quite complex, however, to be able to proof the concept of the solution this project will only use binary decisions. These are decision which have a "yes" or "no" answer and are more easy to model. This way it is not to complex to determine which option must be picked. This is also less complex, because with binary decisions only the situation in which one outcome is picked must be defined, in every other situation the other outcome is than to be chosen.

Within the figure several scores are shown and in some cases the score seems to unknown. This shows that of the source data the quality is determined. But after data integration steps, the quality of the result data is unknown. The later designed system will provide the scores which are missing here.

Like every organisation the MOD encounters the problems of data quality. This does not mean airplanes are crashing, but issues arise and problems are noticed. Sometimes the problems are noticed outside the organisation. In 2012, for example, members of the Dutch parliament asked questions about the current state of the information of the MOD concerning its assets and stock levels.

MOD is not just any organisation for this year (2013) the budget is about 7.7 billion Euros and the number of employees is about 60.000. One can image that every thing that goes on in a normal sized organisation, happens on greater scale in such a large organisation, even the problems. One of these problems is the quality of the data. This problem is currently of special interest since the MOD is centralising the maintenance of its data. All the data in the different decentralised information systems is being migrated to the central information system.

Problems that arise are not just the quality. Also the structure of the data, the capabilities of the software and the responsibilities of the users are changing with this new architecture. This thesis concerns itself with problems with data quality, so it will stick to those problems when related to data.

Some systems have already been migrated to the new centralised information system and the quality of this data is frequently checked. Systems that are going to be migrated, are checked beforehand, making sure the data which enters the central system is good enough. In earlier migrations the problems from legacy systems were migrated along with the data.

Improvements of the data quality, within the information system are initiated based on two triggers. The first trigger is the operational need, performance is less than optimal due to bad data and this relation is shown. Based on this the MOD has a team of people which are authorised and able to improve the quality of the data. Of course, this is guided by strict rules, since the MOD is not just any organisation, it is military organisation, so it is clear that lives depend on some parts of the information system. One of the main points is that it is better to go on a mission having the best data and system possibly available than to have no system available because it is being cleansed.

The other trigger is the weekly automated data quality tests ran on the data at the central information system. This is a test consisting of mote than 400 rules which are checked automatically. The results of these tests can be used to improve the quality of the data, by the maintainers or by the same team of people mentioned earlier.

Of course, the team of data cleaners at MOD is not the owner of the data, so they are not responsible for the data. The owner of the data is someone who is responsible for the content of the data system and most of the time an employee of the department which is the main user of that set of data.

3.1.2 Research Fields

As can now be determined, the research fields in which this research fits are data quality and data uncertainty. The work of Scannapieco et al. [5] provides a good survey into the field of data quality. In the paragraphs under the head-

ing named "Data Quality" a short overview of the field will be given. In an earlier study [34] the field of Data Uncertainty is surveyed, with the focus on Probabilistic Databases (PDBs) as the model for data uncertainty. After the paragraphs on data quality there is a short description on the field of data uncertainty.

Data Quality

The research field "Data Quality" is concerned with the quality of data. But for this to have any meaning one must define the quality attribute of data. Data is, according to the English dictionary, a fact. But to determine the quality of a fact is somewhat strange, since it is a fact. But data in information systems is a value representing a fact in the real world, there is no direct connection, it is a record of a real world object. So it can be wrong, incomplete or outdated.

Mostly this quality is translated into the notion "fit for use", this means that quality is not just dependent on the data object and the real world object, but also on the needs of the data client. Data quality consists of several dimensions. Numerous of these dimensions are found throughout literature, but in this thesis the ones presented by Scannapieco et al. [5] are used to discuss the concept of data quality. These are accuracy, completeness, consistency, currency, volatility, and timeliness. These will be explained more in depth in Section 5.2.

This set of dimensions is a generic set. Some organisations might have specific needs in dimensions. A dimension found in some situation is for example uniqueness. Within the uniqueness dimension one determines if the data-items are unique within the dataset. Other examples can be found in the work "Methodologies for Data Quality Assessment and Improvement" by Batini et al. [4]

For a given organisation some dimensions might be more important than others. The eventual data quality scores should be calculated with the relation of dimensions in mind. Furthermore, maintaining one dimension to improve on that specific score could influence the scoring of other dimensions.

Data Uncertainty

Knowledge of the correctness of data can be modelled via the quality attribute, but it can also be modelled using data uncertainty. Within the field of data uncertainty one sets out to determine the uncertainty of a data value, depicting the level of uncertainty or confidence placed in that data value.

The model most used throughout the field of data uncertainty is the probabilistic database. The recent work of Suciu et al. [27] provides a very good insight in the used models, semantics and query evaluation. In previous work [34] the field of data uncertainty was surveyed and specifically that of probabilistic databases, in order to create an overview of recent accomplishments. This concept of a probabilistic database was one of the main reasons to look into the field of data uncertainty for this research project.

Most important is the notion that data uncertainty is not born out of prob-

lems, but out of the fact that data represents values in the real world. That representation has an inherit imperfection, in comparison to that real world object. For example a lot of values in the real world are in the domain of real numbers. These values cannot be represented precisely in in information system. But one might know the range of possible real world values. The stored values represents the real world values, but are known to be imprecise. Modelling this knowledge of the imprecision is one of the things Data Uncertainty sets out to solve. Of course there are more cases of uncertainty when it comes to data within organisations.

Within this project the chosen model is that of the probabilistic database (PDB). Several recents solutions to this concepts are known, earlier work listed six projects which got recent academic attention, these are BayesStore [29], MayBMS [2, 3], MystiQ [12], Orion [9], PrDB [23] and Trio [32].

These different systems have semantics in common, the possible world semantics [11]. Actually the de facto semantics for a PDB is the Possible Worlds Semantics (PWS). This fits alongside the notion that a relational database represents only one possible world [10]. It is clear that the number of possible worlds grows with each uncertain value in the database. Naively iterating over all possible worlds when evaluating the query is shown to be intractable [30].

To represent the data and its uncertainty all mentioned approaches to PDBs use different models. These range from graphical probabilistic models to confidence scores and alternative tuples. Since all PDBs adhere to PWS, some equivalence is found in the models. The difference is in the efficiency of query evaluation, the types of probabilistic information that can be stored, the space needed to store the data and the complexity of the model itself. Every project was started with different goals in mind. Some can store probability distribution functions other just probability values. Some provide complex correlation support while other provide a more simple but more workable model.

One important aspect is lineage. Lineage is the knowledge where a current data values finds its origin. A combined probabilistic event is dependent on those events which result in the combined event. This is the same for combined data values. Researchers creating a PDB have the option to maintain the lineage information, which is needed for the probabilistic calculations, or they can do the calculation right away and "forget" the lineage information. It has been shown that storing the lineage information and later calculate the probability scores, of just the needed result values, saves time during evaluation. In some cases the probabilistic score is needed to provide the client with ranked results for example. In such a case the smart use of lineage information can not improve the efficiency. [34]

3.2 Research Objective

The overall question on which this research project is based is: "Is it possible to design a generic system to model uncertainty of data and decisions based on that data, in such a way it predicts the data uncertainty during decision-making? Where the uncertainty is determined by use of methods from the field

of data quality." This question was one of the motivations for the design of the overall system which models the link between data and decision in terms of uncertainty. But most parts of this system have already been studied, for example the concept of probabilistic databases.

This overall system will take the following information as input: the data, the uncertainty of the data, data integration steps and the decision. The first two terms are already explained, the third term "data integration" is used for all possible queries and data operations which are performed in order to present the needed data to the decision maker. We assume that SQL is sufficient for the data retrieval in most companies, especially when looking at daily made binary decisions. The information systems supporting these decisions are in place to do just that, so the retrieval operation is relatively inexpensive. Thus most likely a set of SQL queries.

A decision is in our case is a binary decision, either the answer is positive or negative, yes or no. Some values of the data will trigger a positive response, others a negative response. This way the uncertainty of the data can be used to determine uncertainty of the decision and thus of the probability of making a correct decision. The probability of false-positives and false-negatives can be determined. Finally, the costs of mistakes can be modelled and the system can determine the expected value of these costs.

This project sets out to answer the following research question: "How can data quality be modelled using data uncertainty techniques, such that it correctly models the uncertainty at data and decision level". By answering this question, organisations are able to use the power of probabilistic databases to help decision makers understand the uncertainty they are facing. This will apply to organisations, which are not ready for or cannot benefit from other approaches to determine uncertainty at decision level. Section 8 explains how this main question will be solved and provides the answer to this question.

A very interesting property of the overall system is that it provides the user with the means to trace the uncertainty back to the source. So if the user wants to have less uncertainty in the data, the user knows which sources need to be improved. Furthermore, the expected costs of a decision and the knowledge of the cause of the uncertainty provide the organisation with the money available to improve its data quality. A very simple but apt example is the following.

Given a decision to be made, of which triggers a positive answer, with a probability of 0.9 of being correct. Assuming the decision is formulated in such a way the probability of making a mistake is now $1 - 0.9 = 0.1$. The costs of making a mistake are €10,000.– and the benefits of making the correct decision are €10,000.–. This means there is a chance of 0.1 that the organisation loses €10,000, in other words gains €-10,000.– and a 0.9 chance it gains €10,000.–. This results in an expected value of $-10000 \cdot 0.1 + 10000 \cdot 0.9 = 8000$ which is still a gain, but €2,000.– less than generally is expected.

Quality of the data at a decision level can now be defined as a prediction of the costs, or gains, by making decisions based on the current data with the current level of data quality. If decisions like this are made every day, the organisation can spend some money on improving its certainty. How to translate

this into a budget is a question that can already be answered at the MOD.

In order to create this system, all the parts must be designed and created. Some parts of the system deserve their own scientific or engineering research project. These will be discussed in Section 10 on future research. To some extent we have designed a simple version of these "missing" parts to be able to proof the concept of this system and to do the measurements we needed. An example is the language to model the decision with its optional results, triggers, data input and costs. In Section 7.4 this designed language is explained in more detail. It is clear that it is sufficient to meet the requirements and is not a final product.

3.2.1 Supporting Questions

To solve the research question several supporting questions must be answered. Some questions provide background or scope for the main question. Other questions brought this project closer to the answer to the main question. This section lists the supporting questions and explains them.

1. What is known about the data of the MOD currently?
   (a) How is quality of data determined?
   (b) What is the domain and state of MOD? Of both Data and Data Quality.

To start a project concerning the data of a given organisation, one must understand the current situation of that organisation. This project first supporting question is to understand what the current situation of the MOD is. To eventually design a generic system one must compare this with the generic case. Next, the data quality efforts of the MOD are evaluated. Not to determine whether these are correct, but to understand how they tackle their problems. At the MOD a group of professionals devote themselves to the data quality issues and this is done by following the principles of data quality.

1. What research fields are interesting to understand when setting out to solve this question?

This question has already been answered, but is mentioned since it is an integral part of the raison d'être of this work. In Section 5 the research fields Data Quality and Data Uncertainty will be explained more elaborate. The following supporting questions support the main research question more directly.

1. How to define the relation between Data Quality and Data Uncertainty?
2. How to define the relation between Data Uncertainty and Probabilistic Databases?
3. Determine the influence of Data Quality on Decisions?

4. How to model the data quality scores of the MOD, using Data Uncertainty?

As explained data quality and data uncertainty will be described. Once this is done the relation between these two fields will be discussed. A better understanding of the relation was not clear before the experiments were completed. However, Section 5.4 follows the sections on data quality and uncertainty directly. Later during discussion of the research questions, hypotheses and the experiment results, more insight will be gained in this direction.

A more easy to answer question is that of the relation between data uncertainty and probabilistic databases. In recent work the tool mainly used for modelling data level uncertainty, is the probabilistic database. But it is still important to explain why this model is so appropriate.

A very important finding will be if and hopefully how the business performance is influenced by the lack of data quality. This work, however, focusses on decisions so we set out to determine the probability of making mistakes based on data of less than perfect quality. For this work we thus define business performance as the ratio of mistakes noticed after a set of business decisions.

Next, this project shows how to model the data of the MOD in a probabilistic database. Finding this is the goal of the main research question, but after that step we set out to make this more generic.

1. Does more detailed data quality knowledge improve the prediction?

2. Data Quality. An end in itself or a level of "fit for use"?

During experiments it became clear that the predictions made were sound, when looking at the complete data set. Even when random sample sets were examined in more detail, the determine uncertainty was shown to be correct. However, individual items might not have a correct uncertainty score. This is logical, since the detailed knowledge is either not available or removed in this more abstract view. To determine if information will not get lost by the use of the later designed system, tests are done to determine if more detailed data quality information improves the result.

The definition of data quality will be given in Section 5.2. This definition will explain that the needs of the data client are of influence on the way the quality score is determined. This work sets out to determine wether the designed system performs better if data quality measures are done with as much rules as possible or with only those rules which directly support the client.

The following list of questions guided the design of the overall system and are answered in Section 7.

1. What are input restrictions and requirements?

2. A language to model the decision

3. How to support data integration steps of an organisation?

4. Must complex data operations be supported?

5. Which probabilistic database can support this overall system?

6. How to support the decision maker with this system?

First for all, the input is partially defined, this would be generic data found in information systems. Over this data, quality scores are determined by the use of data quality metrics. The way to get this into the new system must be determined.

Next, a non-trivial artefact is designed, a language to model the decision. This decision must be modelled in such a way the system can retrieve the needed data. The language must model the needed data uncertainty operations. And the language must show how to provide the result data accompanied with the probabilities on false-negatives or false-positives. As explained this work focusses on binary decisions, which have a positive or negative outcome.

To determine the result data, operations must be performed. But how this must be done in the overall system must be determined. Additionally, what kind of operations are supported or can be supported is something that must be defined.

Later will become clear that one probabilistic database is used during the experiments, since that one was available, reliable and sufficiently complete for the project. Finally, this system must eventually support the decision maker. We set out to design a system that supports the decision maker.

3.3 Scope

In Section 3.2 the main objective is explained, this object is to find a way to model the quality of the data, such that the overall system can provide the uncertainty at decision level. This provides to outer scope of this project. However, this is this still too vague and too large.

Firstly, to create a system that can provide an outcome to a decision, the decision must be defined, as well as the possible outcomes. To make this possible and to refrain from designing a system that is too complex, only binary decisions will be supported.

In addition to binary decisions, also relative complex data operations will not be supported. Earlier we noted that we believe that supporting SQL would sufficient. This system will support everything that can be defined as a single query or a sequence of queries. These queries must be in SQL or TriQL, where TriQL is an extension to SQL used by the probabilistic database system that supports the overall system. Given a generic data retrieval action on relational databases, this setup will be sufficient, however, for the system to be useful with more complex data retrieval operations more research will be needed into defining such a set of operations.

The data quality information available at the MOD, is not enough to determine conditional probabilities and correlations. This will not be a requirement of both the supporting probabilistic database system and the overall system.

Furthermore, the overall system is designed with the goals of this research and the MOD in mind. This means that is a generic as possible, but at least meets the requirements of this research project. For example, not all data types are support, but only the set which is found at the MOD, we believe that such occasions do not influence the prove of the concept.

3.4 Contribution

After reading the parts of this thesis concerning the research questions, it is clear which parts are new to the scientific world. The most interesting part here is the concept on how to use the field of Data Uncertainty to model Data Quality in such a way that the Data Uncertainty methods and techniques can provide prediction about the decisions made with that data. As can be read in the concerning sections and the discussion & conclusion, these are correlations, but this project was unable to "contradict" the suspicions of an existing relation, with both the empirical results and deductive reasoning.

The most interesting knowledge gained during this research project is how to model the quality of the data, such that retrieving the data from the probabilistic database provides uncertainty scores consistent with the quality of the data. The concerning sections set out to explain how one must use data quality principles, for example how to define rules and how to score the data set accordingly and how to store this in the given probabilistic database.

This last paragraph looks like to contain somewhat of a circle. Of course, if one measures value "x", stores this value and then checks if it is still the same value, this does not prove anything. But data quality and data uncertainty are different ways of scoring the correctness of the data. So a correct quality score might not be the score that one must use in data uncertainty. To cope with this problem this research sets out to find how one can use data quality methods to determine uncertainty at decision level.

Furthermore, the data quality measurements are done once a week at the MOD. Other companies might do it more or less often. This means the scores are not completely related to the actual data used. This problem could manifest itself in such a way the measurements have nothing to do with the current data, but one can assume that a company measuring its data quality, does this in such a way that a trend analysis tells something about the current data set.

At an operational level, only the data is available. A specific data item which fails a test is not removed or locked. So the daily operations have no knowledge of the exact incorrectness of a given data value. They could benefit from a complete correct data set, but this is intractable. So they need the best data set they can get and they need to understand the uncertainty. This project aims at just that insight, providing both the only available answer and the uncertainty of that answer.

3.5 Outline

This work starts with a section on data within organisations. Here the generic concept will be discussed briefly and the situation of the MOD will be explained. Next it is time to relate the goal specified by the MOD to the data, data quality and data uncertainty. The main goal is to understand the influence of data quality at decision level. The scope of this project is too small to completely answer that question, but it sets out in just that direction and will get the MOD closer to that understanding, this will be done in Section 5.

Next, a section about the mainly used approach to data uncertainty modelling is discussed: the probabilistic database. The concept of the PDB is discussed in some depth in Section 6. The work divides the concept of data uncertainty into four segments: semantics, models, query evaluation and result presentation.

Before this work continues to the actual research done, the overall system used to perform these measurements will be discussed in Section 7. This section will explain that the overall system is the solution to the original problem presented by the MOD. This research project has its place within that overall system.

Section 8 is the core chapter in terms of the research project. This section contains the different research questions and hypotheses formulated during the research, the results and the conclusions made based on these results. It also shows how one hypothesis is transformed into the next, in order to finally come up with the best hypothesis fitting the evidence. Finally, this work ends with a discussion & conclusion section, to put findings and thoughts against one and another and to conclude the results of this research.

4 Data in Organisations

Until now data has been discussed as something that is used by every organisation and might be correct or not. This section will clarify on a more abstract level what data means for organisations and how data is maintained. The next section on Data, quality and uncertainty will briefly discuss what data is and what quality and uncertainty means in the context of data.

4.1 Organisations in General

Every organisation owns data, from client information to employee information or members, financial records etc. Most of this data is stored in tables, sometimes as simple as a spreadsheet, other times more complex relational database systems. Spreadsheets can become quite complex, mostly by then an organisation switches to a more sustainable solution.

All this data is needed to run the daily operations of an organisation, for example sending out bills, ordering stock, measuring business performance and many more. Knowing the goals data is used for, it is understandable that a lot of organisations treat data as assets nowadays. This principle is known as Data Governance. [22]

Data governance principles enforce business to look at data as assets. Not just value them, but also assign people to be responsible for the data and information systems. This includes maintaining the systems and also the quality of the data. This way others within the organisation are assured the data is correct and timely available. This accountability empowers those who are responsible to take action to improve the state of the data, an example is that they should be able to demand a certain quality of the data put into the system. When looking at other parts of an organisation, these responsibilities were already in place, purchasers demand a certain level of product quality, so should system administrators.

Other important parts of data governance, besides data quality, are policies, processes and risk management concerning data. The main goals of the methodologies in data governance is to make the enterprise more efficient, by utilising policies, processes and technology, data governance makes it possible to reuse data throughout the whole enterprise. There are even maturity levels along which a company can be measured how well it takes care of its data. Data governance is in a lot of the cases a requirement to exist as an organisation.

Data is of influence on the business performance, this and the fact that the dependency on data grows, forces organisations to look at things like data quality. A field like Data Quality is a natural place to start when intending to improve operations results based on data. The field is rich on methods and frameworks, which everybody can employ to their advantages. [5]

4.2 Royal Netherlands Ministry of Defence

This work focusses on the data situation as maintained by the Central Data Management Office (Centraal Bureau Gegevensbeheer - CBG). The CBG is a department of the Defence Materiel Organisation (DMO). The CBG maintains a large set of databases of several different organisations within the MOD. The goal is to centralise the data governance and maintenance efforts of the complete MOD.

A few years back this centralised data management system was put into action and every department could put its data into this system. As expected this first attempt failed, there were too many variables and this approach was not just difficult, but impossible. After a rollback and a revisit of the drawing board, standards were laid out to which the data should adhere before it could enter the new system.

The second good idea was to do this system per system and not all the departments at once. This way the data-professionals of CBG can support the departments when they get a chance to draw up their specific requirements they have for the system. Furthermore, they get support in cleansing and standardising the data.

The part of data governance efforts on which this work focusses is data quality. The data quality project of CBG consists of over 400 data quality rules to check the data within those dimensions mentioned earlier. Using advanced software products and these rules to measure the data quality the CBG gets a very precise overview of the data quality in their system.

The same checks are now used to measure the quality of data before it is placed under maintenance by the CBG. This way the data gets a first improvement before it enters the system and the system is kept at the same quality level.

As noted earlier there is another trigger which tells people to improve the data quality. This is the operational awareness of data problems. If data must be cleansed because important tasks need this, the data will be cleansed by a special data quality team.

5 Data, Quality and Uncertainty

Until now this work hinted on what data is, what quality of data means and what one can expect of uncertainty within data. This section aims to provide a more detailed explanation about data, data quality and data uncertainty. A superficial understanding of these concepts might be enough when one encounters the problems with the data and employs professionals to cope with these problems. But to understand the problems this work sets out to solve, more depth is needed.

5.1 Data

When searching the Oxford Dictionary for the term "data" one finds that it is a noun meaning: "facts and statistics collected together for reference or analysis". A very apt description, for this work it is important to note that data in this case is always a recorded value in a digital system. In other words, data in the context of this work are values in a database and of course especially those values that are used by the organisation. The Oxford Dictionary adds information of the origin of the word. It stems from the Latin word "datum" and is first found in the 17th century in philosophical texts.

For this thesis "data" is defined as followed:

Definition 1. *Data are values representing facts in the real world.*

Data is found throughout every organisation, but it differs in form. Most data is found in relational databases, a system which empowers the storage and retrieval of data values. The relations which can be defined make it a perfect system to model the real world objects which are represented by the data in the system. A Relational Databases consist of data-tuples with relations and attributes. The model of a relational database is called: "the relational model". [10] One of the reasons for this popularity is the fact that anything can be modelled using the relational model. Of course this might not be the most efficient model for all applications. [15]

As explained every organisation needs data and not just organisations, everyone benefits from the data stored throughout all different systems. Relational databases store the data highly structured, but there are other means to store data, namely semi-structured and unstructured. An example of semi-structured data is XML, these are structured but the data is part of sections of semantic chunks, text for example. In layman's terms the difference can be explained as follows: in a relational database contains specific data values, where XML nodes contain pieces of information. Of course, it is possible to define a highly structured XML document where the nodes only contains single values, but one of its powers is the possibility for this semi-structured data. Finally, a lot of data is found in an unstructured fashion. These are for example e-mails sent by a person or audio fragments, there is data inside but it can only be found with complete semantical understanding.

5.2 Data Quality

By now it is clear organisations want data of high quality, or they want to be robust enough to handle the data of less than perfect quality. To understand how well the quality of the data is within an organisation one must measure it. But what does one measure? Within data governance principles, data quality is an interesting concept and this field contains very interesting methodologies to determine that quality. With data quality measurements one measures how well the real world objects are represented within the database systems and if the data meets the quality needs of the data clients.

Data Quality is mostly explained as the level of fitness for use of the data. During the research project this definition is made more precise. However, this more precise definition is only appropriate when dealing with data uncertainty.

Definition 2. *Data Quality is the extend the data supports the correct outcome of the decision made based on that data.*

Data Quality is a mature and rich research field and can be an end in itself. This definition is by no means a generic new definition. Within this research this definition is both appropriate and found to be a good foundation to start modelling data quality using data uncertainty methods. This definition also shows that the quality is dependent on the use of that data. So it still adheres to "fit for use" definition, but it now also defines the quality as a percentage or ratio.

The work of Scannapieco et al. [5] called: "Data Quality, Concepts, Methodologies and Techniques", is a good survey of current knowledge and methods concerning data quality. Especially the defined dimensions seem to cover the generic spectrum of data quality and the work provides ways to calculate quality scores within these dimensions. As mentioned before, there are more dimensions defined in other work, some are equal, others need a different perspective to be useful, but the work of Scannapieco et al. provides a more generic and overall set.

This work explains a framework to guide the data quality efforts of an organisation. This book can help get the organisation to a mature data quality level. This section continues with a succinct part on data quality as defined by Scannapieco et al.

Quality problems concerning data are noticeable every day. Companies which deliver multiple services to the same person, are sometimes unable to merge the duplicate data they keep on that person. Not all problems are purely due to the lack of data quality, but these duplicate records are a sign of quality issues. Solving data quality problem is also not just cleansing the data, sometimes policies are needed, new requirements or a new information architecture is needed to solve the problems.

The example of the duplicate client-data could be solved by correcting a misspelled name. But this might need a database redesign to accept more characters. Perhaps there are more systems and all use different identifiers for

client-data. So there might be no direct way to determine if two items represent the same real world object. The combination of several unique attributes of a person could help, but not in every case. For example if a policy prohibits communication of personal data, even within the company, merging data might not be possible. Examples of such policies are found throughout the world of banking, insurances and military organisations.

For the organisation to be efficient, company wide data integration is very important. This is only feasible if the data is of a certain quality, people must be able to trust the data. Even if somebody else is providing it, otherwise, they might want to maintain their needed data them selves. With poor data quality this becomes a difficult and expensive task, while costs and time could be saved when the data sources contain data of high quality.

Data quality as a research domain, exists for some time now. Within statistical research there were the first encounters of research into the quality aspect of data. Logically computer science and management studies followed, both fields noticing the influence of the quality of data first hand.

In this text until now data quality can be defined as the correctness of the data and the fitness for use. But it is a bit more complex. Data quality can be measured along several dimensions. Because some situations require data which is eventually correct, while other situations need the data right now and close to correct is good enough. Both serve the need of the client, so both should score relatively well. This cannot be done using one metric.

The dimensions as Scannapieco et al. have defined them are accuracy, completeness, time-related dimensions, and consistency. Accuracy depicts the closeness of the data to the correct value. Completeness of data shows if every needed value of the real world object is represented by the available data.

Two paragraphs back, the notion of "time" was mentioned. Data can be correct later on, or right now. But it must be correct at the moment the data client requests it. The time related dimensions; currency, volatility, and timeliness, provide a means to measure the quality of the data along these dimensions. Currency shows if the data value changes according to changes in the real world, the volatility dimension shows how often the data is actually updated. And the timeliness dimension is concerned with the fact if the data is available on time.

To finally cope with semantic rules of related data values, there is the consistency dimension. Data can be syntactically correct and accurate while still there are faults. If these semantic rules apply to the real world as well, these mistakes fit under accuracy, otherwise, if the rules only apply within the organisation, consistency is where the mistakes would be found. Data could be semantically correct, but not adhere to syntax or standards, the data might be accurate but not consistent.

These dimensions are not independent, improving the data along one dimension, might mean lower scores along another dimension. This all makes data quality to a somewhat complex domain.

The measurements and improvements are not straightforward. Checking the real world value might not possible. So scoring sometimes depends on theoretical

sets. With enough knowledge of the organisation, its data and its surroundings the data quality problems can be tackled. A mature approach to data quality helps organisations to solve a lot of their problems concerning data.

The set of definitions and the example measurements found in the work of Scannapieco et al. show that doing data quality checks at the earliest or lowest data level is better than at a higher, or information, level. Since the fact that these measurements take up time and some even take up a lot of time, these measurements are not well suited to run at the moment of decision making.

After implementing a measurement environment, an organisation can choose to cleans the data. However, another good option would be to use the same rules to define input validation and filters. This way the data quality problems are kept out of the system. This greatly improves the trust in the data set, since people know that no erroneous values are placed in the database anymore. However, some measurements need to be done with a complete data set, something that is not possible during input.

Most of the time, if the data is of less than perfect quality, people have placed that data in the system. One-to-one digital data transfer only goes wrong if there are no data exchange standards defined or the system itself contains bugs or errors. So generally speaking, people make mistakes, for the MOD this is no different. People will input data of good quality if it easy to do so, so one must make a combination of input filtering and an easy way to input data of high quality.

The quality dimensions discussed so far are mostly used independent of the production, use and storage of the data. While a lot of the data quality problems noticed are not independent of this larger context. Exception of completeness and the time-related data quality dimensions, the dimensions are of the intrinsic data quality category according to Strong et al. [26]. Accordingly, the completeness dimension and time related dimensions are of the contextual data quality category.

As Strong at el. define, the intrinsic data quality category means the dimensions are independent of the context. While the opposite is true for the contextual data quality category, these are completely dependent on the context. The context is defined by the production, maintenance, storage and use of the data.

Furthermore, there is the accessibility data quality category, containing dimensions which are to be used to determine how well the data can be retrieved as well as protected agains unauthorised access. And finally, the representational data quality category, where representation, understandability and interpretability are important dimensions to measure the data quality.

However, the intrinsic data quality category is defined to be independent of its context. In a sense that the measurement and rules can be defined solely based on the data. The understanding and the use of the context surrounding the data can greatly improve the data quality assessments relevance for people and systems involved with that data. As we will later determine, the data client is only interested in good data according to his definition of good and concerning the data he uses at that moment.

5.3 Data Uncertainty

Using data of which one knows the quality is not 100% means the client is accepting the data as it is. One might actually know if the data is correct or incorrect, but a lot of the time, based on the nature of the data, it is not known at the moment of use. There is a level of uncertainty about the correctness of the data. This is one type of uncertainty the field of data uncertainty aims to model. Data is defined by now, and so is quality, so now the meaning of "uncertainty" must be found. In contrast to the facts data represents, uncertainty claims that a value is vague, not precise or not definitely.

Definition 3. *Data Uncertainty is the probability the retrieved data correctly represents the real world value.*

This definition is apt, but somewhat simple and the rest of this section will explain data uncertainty in more detail. But this definition is sufficient and precise for use in this research.

According to Dalvi et al. [11] uncertainty exists in all systems, however, the reasons for these imprecisions are different. For example, sensory data has a variance, other values can't be stored precise enough and in some cases data is purposely imprecise for sensitivity reasons.

Within data uncertainty the imprecision is expressed as probabilities, probability distribution functions or other scores which adhere to the principles of probabilities, but might have a slightly different definition. Examples of this last case are belief, trust and confidence. These will be discussed later. This way of expressing uncertainty gave rise to the probabilistic database, the widely used model to cope with data uncertainty.

Uncertain data is not bad or wrong data, the uncertainty expresses a certain knowledge about the data. It shows the probability distribution of the magnitude of the discrepancy between the real world fact and the fact in the database.

Dalvi et al. explain in their work that using the right methods, valuable information can be retrieved from uncertain data. They also define a probabilistic database as a system that stores data, probabilistic information about that data and support querying the data and its probabilistic information.

Several different approaches to probabilistic databases exist. Some segments will be discussed in Section 6. All the approaches adhere to the generic definition of a probabilistic database, but use different data models, probabilistic representations and query evaluation. Cavallo et al [8] have formalised the definition for probabilistic databases and the work of Dalvi et al. [11] explains the principles in depth.

The next question one might have, is where the uncertainty comes from, bad maintenance? Bad record keeping? Generally speaking the information architecture of any company exists of sources, channels, agents and users. Such entities influence the uncertainty of the data retrieved from the information systems. A source might be maintained by another company, so the quality

needs might differ and this means there is reason to doubt the data to some extend. Channels might be compromised and introduce missing values. Agents are intermediate systems, or persons, which relay the data. This is prone to accidental and intentional changes.

Some approaches to data uncertainty maintain information like provenance, pedigree and lineage. This information can help in determining the uncertainty.

Lineage is the mainly used principle within a probabilistic database to point to the original data items used in the creation of the current data value. This, however, is only maintained within the probabilistic database. This information can be used to calculate the uncertainty of that current data item. Other approaches use the outside origin of the data to determine the uncertainty of the data, in this case the earlier mentioned sources, channel and entities are of importance.

Wang et al. [31] devised an interesting way to use provenance information to determine trust in a data value. This model actually is very similar how lineage works on a data level.

The model Wang et al. designed, makes it possible to determine trust in a network setting. Data flows through this network and is combined into the result the overall system provides. Different sources provide the "raw" data and intermediate agents process the data and passes it along. The model of Wang et al. uses provenance information, trust placed in values and nodes and the reputation of the different nodes to determine the trustworthiness of data. This way they aim to design a collusion resilient trust determination system. In earlier work [34] more details of this model are explained.

To compare data uncertainty models, and especially probabilistic databases, Keijzer et al. [13] point to decisiveness, density, precision and recall. Decisiveness shows to what extent the database system can provide any answer with at least some certainty, for operational usefulness even probabilistic databases must provide a value to work with. The level of uncertainty can be used by the client to either work with this value or to improve his knowledge.

Density explains the average number of alternative answers the system presents to the client. This means the system tells the user that it has more than one option to choose from, the uncertainty information tells the client the most probable answer. Precision and recall are more like data quality properties, these show how well the real world objects are represented. Precision is the ratio of correct answer in the complete answer set given by the system and recall is the ratio of the provided correct answer and the total set of actual correct answers in the real world.

So far it has been seen that uncertainty manifests itself as a probability, confidence, belief or trust. But what is the difference? The difference is the meaning of the uncertainty score. In the generic case of a probabilistic database a probability or probability distribution function is coupled with the data. But in a natural situation this might not be the desirable way to model the uncertainty.

For example, it will take a more complex model to fit in the natural mutual- and non-exclusion features, while using the notion of "confidence" the default situation is that there is no mutual exclusion. Within a company a conflicting

phone number, could also mean the person has two phone numbers.

Another requirement of probabilities is that there are a total number of outcomes to an events, of which the added probability is equal to one. However, the information needed might not be available, so PDBs either use probabilities and define what it means when scores do not add up to one or they use one of the more loosely defined terms and adhere to the needed subset of probability calculus.

The best explanation of confidence values comes from the Trio system. The confidence value [0, 1] depicts the probability of correctness of a value, a relation or a tuple. However, two defined values, relations or tuples are not mutual exclusive, unless specifically defined. The remainder between the confidence score and 1, means there might be no or an unknown value, relation or tuple. [32]

The mentioned notion of "belief" has a different meaning. This term was found in the work of Shafer [24] and used in the Dempster-Shafer Theory. Belief is like probability but the precise probability calculations needed with belief values are picked according to the meaning of "believe" as defines in that theory. Within this Dempster-Shafer theory these calculations are called belief-functions.

Trust is mostly somewhat different, it is a score based on for example the reputation of the providing agent. This score might be in the same domain [0,1] but cannot be used in calculations like probability calculations. Trust is somewhat like a quality score, however, it still depicts the level of uncertainty. The same is true for belief, confidence and probability, these shows the level of uncertainty of the data.

Concluding on the differences between probability, confidence, belief and trust one can determine that they all express a probability. However, how events are interpreted and operations picked to determine the certainty of a piece of data, is slightly different in each case. Also what entity the certainty is related to, is different in some cases.

Like data quality, data uncertainty has dimensions as well. The work of Walker et al. [28] shows a set of such dimensions. In the case of data uncertainty these dimensions show the nature of the uncertainty. Walker et al. speak of location, level and nature. Location provides information on the position of the uncertainty within the data retrieval process. Level is the degree of uncertainty and nature explains how the uncertainty came into play, either caused by the phenomena, the measurement method, the storage, the channel or the imperfection of the knowledge itself.

5.4 Relation between Data Quality and Data Uncertainty

Data quality is determined at a data level, based on the needs of the data client, these needs are formulated at the operational or decision level. The designed system determines data uncertainty at a decision level. So according to the definitions provided earlier and this situation, data quality and data uncertainty have quite some similarities. We can even define this relation for this research.

Definition 4. *The relation of Data Quality and Data Uncertainty: Data Quality and Data Uncertainty are equal from the perspective of the data client and the client's data needs. Both are probabilities of the correctness and "fitness-for-use", within the decisions of the data client*

Looking back at the explanation of data quality and data uncertainty there are similarities and there are differences. Both show a level of confidence one can place in the given data. Data uncertainty can models the probability a given value is correct. Data quality has a more definite score, either a piece of data contains mistakes or it doesn't, however, this is not true for every metric within data quality and not for every situation.

We believe several methods within the area of data quality can be used to provide the level of uncertainty of the data. One example would be, given that the data quality tests are run over a given time period and the given data value has not yet been checked. Earlier data quality scores might indicate that 3 item out of a 100 are wrong. Assigning 0.97 as the probability of this current item being correct, is a good way to show the uncertainty.

Once a data quality check is completed, one knows of that set which items are incorrect. This information can be modelled using uncertainty. A completely wrong item can have a probability of being correct of 0. This is somewhat unnatural but it is both the only answer a system can provide and the system knows it is incorrect. The meaning of that knowledge is for every organisation different. At the MOD there is a strict rule, "a piece of data is correct, if the person who manages both the data and the real world object, claims it is correct".

On the other hand there is the data which passes all data quality tests. This could be modelled as completely certain data, and get probability 1. It can also be be modelled as an other default high score, it can be assumed the data quality measurements are not perfect, so this highest possible score could reflect the imperfection.

Later in this thesis it will be shown that the relation between data quality and data uncertainty does exist and is indeed not straight forward. It is clear that a combination of certain data quality measurements, data quality rules, client needs and situational aspects provide a way to determine the best approach to model the quality using data uncertainty.

6 Probabilistic Databases

This section takes a look at the mainly used model within the field of data uncertainty, the probabilistic database (PDB). Data is not always completely certain, there are questionable sources, channels and agents. The phenomena or its measurement could have an inherent uncertainty. It might not be possible to store the precise value in the given system. But decisions depend on the data and the knowledge of the certainty of the data is very useful. Traditional relational databases consist of tuples and relations, which can be used to model the information [10]. PDBs adhere to the principles of relational databases and extend the definition by viewing all, or a sub set of the parts of the model as probabilistic events.

Before this section continues with explaining the different parts of probabilistic databases, the terms used must be defined. As much as possible the terms from the work of Suciu et al. [27] will be used with the same meaning. This means the term "tuple" is a data object, consisting of attributes, these attributes are data values. A tuple represents a real world object.

The concept of probabilistic databases was first formally described by Cavallo and Pittarelli in 1987 [8]. The concept is dividable in four distinct parts. These parts are small research fields of their own and advances are made in all of them. The distinct parts are semantics, data model, query evaluation and presentation. The first three are defined by Dalvi et al. [11] and the fourth is covered in all recent work to some extent.

The formal definition given by Cavallo et al. is shown next. This definition is based on the definition of a "normal" relational database, with the distinction that the finite set of relations is replaced with a finite set of probabilistic functions. Within the finite set of relations each relation is a sub-set of the domains, these domains consist of all the relations that can possibly be defined within a schema.

Definition 5. *A probabilistic database is a set* $PDB = \{P_1, \ldots, P_n\}$ *where each element in the PDB is a probabilistic system:* $P_i = (V_i, \Delta_i, dom_i, p_i)$*, where:*

1. V_i *is a non-empty set of distinct attributes,*
2. Δ_i *is a non-empty set of domains,*
3. dom_i *is a function that associates a domain with each attribute,*
4. p_i *is the probability distribution function over* V_i [8]

First, Section 6.1 will discuss the semantics of PDBs. Next there is a section on the data models used in different probabilistic databases. As will be seen the semantics are generally the same, but the data models differ from one PDB to another. The section on query evaluation shows how different PDBs evaluate the query. This chapter ends with a section on result formatting.

6.1 Semantics

Semantics within PDBs provide the meaning. Semantics provide a way to compare models and evaluation techniques, since almost all PDBs have the same semantics in common.

The relational model allows one to model one real world. Of course, one can model several possible objects, but for the model and the for evaluation of the query it is one possible world. A probabilistic database must be able to model several possible worlds. The de facto semantics for a PDB is defined by Dalvi et al. [11] as Possible World Semantics (PWS).

More formally this research defines a conventional database I and a probabilistic database as a discrete probability space $PDB = (W, P)$, where $W = \{I_1, I_2, \ldots, I_n\}$ is a set of instances, called possible worlds. And $P : W \to [0, 1]$ is such that $\sum_{j=1,n} P(I_j) = 1$ [11]. Antova et al. [3] defined the same semantics different, and call it World-set Decomposition (WSD). WSD assumes there is one world consisting of several possible subsets, it is clear that this semantics is equivalent but the concept is somewhat different.

Since the semantics of different PDBs are the same, there could be the possibility for a standard all encompassing data model. The first notion is, that such a model would be very complex. It would for example be impractical to implement all possible probabilistic distribution functions. Even more so, if a system support custom defined continuous probability distribution function and support combination operations with these. So with different goals in mind, different projects choose different solutions.

6.2 Data Models

The data model used within a PDB is different within most approaches, unlike the de facto standard for the semantics. As far as can be seen the researchers take a subset of the possible ways to define probability events over data. They pick this subset according to problems they intent to solve.

Antova et al. [1] explain that any practical data modal can only be designed with the next aspects in mind: Expressiveness, succinctness, efficient query evaluation and ease of use. These four aspects are true for the relational model as well. Expressiveness of the model is the fact whether the representation is closed under the operations, meaning all the operations can be achieved by use of the model. If there is little or no overhead in defining instances of the model, it is succinct. The efficiency with which the model can be queried, is an important feature and finally one must be able to use the system.

Within probabilistic databases data is stored using relations, tuple and/or values. Every model approaches this a bit different, but all use variations on the following: ?-relations, u-relations and x-relations. Meaning, respectively, possible relations, an uncertain (probabilistic) relations and a mutual exclusive uncertain relations. Probabilistic databases consist of ?-tuple, u-tuple and x-tuple, which have an equivalent definition as the relations mentioned above. [34]

The uncertainty at data value level is referred to as attribute uncertainty.

Attribute uncertainty has the same properties as both relation and tuple uncertainty. The only difference is that an attribute has only zero or one actual value in the real world. Unlike tuples and relations which can have zero or more actual counterparts in the real world. This is due to the fact an attribute is one property of one object. Within a relational database the fact is that it represents the real world complete, leaves freedom in how to use tuples, relations and attributes to model the real world. However, within an PDB a more strict notion of attributes, tuples and relations must be followed, since the probability calculus defined over the operations acts differently according to these types.

An important model used is Block Independent-Disjoint (BID). This model is interesting since a lot of generically designed efficiency improvements to query evaluation assume a data model consistent with BID. According to the definition made by Ré and Suciu [20], BID is a situation where a relation is partitioned into disjoint sets, or "blocks". This way disjoint tuples are modelled as disjoint events, so both the data model and the probabilistic calculations maintain a consistent set of events towards the end result.

As will be later explained the measurement tool built for this research project uses a specific PDB. This system is called Trio[1]. The work "Working Models for Uncertain Data" by Sarma et al. [21] explains the first design of this data model, which fits the principles of BID.

The basic building blocks of their data model are x-tuples, ?-tuples and a-tuples, where a-tuples are approximation tuples, these tuples contain a value close to the real world value. They define a complete model $\mathcal{R}^A_{prob}$ which encompasses these two types of tuples. They have designed several incomplete models as restrictions of this theoretical complete model. The definition is shown next.

$\mathcal{R}^A_{prob}$ relation is represented by:

1. A multi-set of a-tuples, $T = t_1, \ldots, t_n$.
2. A boolean formula $f(T)$. [21]

with:

Definition 6. *The set of instances $I(R)$ of an $\mathcal{R}^A_{prob}$ relation $R = (T, f)$ is the set of ordinary relations R that can be obtained as follows:*

1. Let σ be a satisfying assignment for $f(T)$. Consider the set of a-tuples T' in T for which σ assigns `True`.
2. Let R be an ordinary relation that includes one tuple from every a-tuple, as defined above, in T'. [21]

Figure 2 shows the comparison in expressive power of the incomplete models. And Figure 3 shows the closure of the incomplete models. Closure means the extent to which the model is closed under certain operation, a complete model is closed under every operation. If a model is closed under the needed operation, this model is good enough to get the job done.

[1] `http://infolab.stanford.edu/trio`

Model	$\mathcal{R}^A$	$\mathcal{R}_?$	$\mathcal{R}_?^A$	$\mathcal{R}_{\oplus\equiv}$	$\mathcal{R}_2$	$\mathcal{R}_2^A$	$\mathcal{R}_{sets}$
Building Block	a-tuple	tuple	a-tuple	tuple	tuple	a-tuple	tuple
Constraints	none	?	?	binary $\oplus$, $\equiv$	2-clause	2-clause	n-way choice

Figure 2: Nomenclature and Definition of Incomplete Models [21]

Closure-Model	$\mathcal{R}^A$	$\mathcal{R}_?$	$\mathcal{R}_?^A$	$\mathcal{R}_{\oplus\equiv}$,$\mathcal{R}_2$,$\mathcal{R}_2^A$,$\mathcal{R}_{sets}$
Union	Y	Y	Y	Y
$Select(ee)$	Y	Y	Y	Y
$Select(es)$	N	Y	Y	Y
$Select(ss)$	N	Y	N	Y
Intersection	N	Y	N	N
Cross Product	Y	N	N	N
Join	N	N	N	N
Difference	N	Y	N	N
Projection	Y	Y	Y	Y
Duplicate Elimination	N	Y	N	N
Aggregation	N	N	N	N

Figure 3: Closure of Incomplete Models [21]

An example of the Trio model is shown next, this model is directly based on the work of Sarma et al. [21]. In these examples it is assumed that all data is available, so if probabilities do not add up to 1, the remainder is the possibility the value does not exist at all. Additionally, the value # shows the confidence in that data value and the value (#), note the parenthesis, shows the probability.

```
[Amy, 12/23/04, SF, jay] 1.0
[Amy, 12/23/04, SF, {crow(0.6),raven(0.4)}] 0.8
```

results in:

```
I1(0.20):  [Amy, 12/23/04, Stanford, jay]
I2(0.48):  [Amy, 12/23/04, Stanford, jay],
           [Amy, 12/23/04, Stanford, crow]
I3(0.32):  [Amy, 12/23/04, Stanford, jay],
           [Amy, 12/23/04, Stanford, raven]
```

selecting only crow sightings:

```
[Amy, 12/23/04, Stanford, crow] 0.48
```

means:

```
I1(0.52):  empty
I2(0.48):  [Amy, 12/23/04, Stanford, crow]
```

The Trio system uses lineage to keep track of the needed probabilistic cal-

culations. Lineage information tells the system what the original data items were. This way the combined uncertainty can be determined afterwards. This approach to lineage used in Trio is best explained using the work of Benjelloun et al. [6] concerning x-relation.

In their work they explain a Lineage Database (LDB) as follows: "*given a query Q and an LDB (only the lineage version of the Uncertainty and Lineage Database (ULDB) system) D, $Q(D)$ is an LDB that extends D with one x-relation R_q and with lineage λ_{R_q} from R_q to $I(\overline{R})$. We write $Q(D) = D + (R_q, I(R_q), \lambda_{R_q})$.*" [6]

As can be seen in Figure 4 the λ function points to ids of data which are the source for the current data value. In the bottom table the lineage information points to id 21 of the top table and id 31 of the middle table. This way the uncertainty of the bottom table can be determined based on the uncertainty of the above two tables in the same way as discussed in the earlier examples. As can be seen all possible instances are preserved.

ID	**Saw(witness, car)**	
21	(Amy,Mazda) ‖ (Amy,Toyota)	?
23	(Betty,Honda)	

ID	**Drives(person, car)**
31	(Jimmy,Mazda)
32	(Jimmy,Toyota)
33	(Billy,Mazda)
34	(Billy,Honda)

ID	**Accuses(witness, person)**		
41	(Amy,Jimmy)	?	$\lambda(41,1)=\{(21,1),(31,1)\}$
42	(Amy,Jimmy)	?	$\lambda(42,1)=\{(21,2),(32,1)\}$
43	(Amy,Billy)	?	$\lambda(43,1)=\{(21,1),(33,1)\}$
44	(Betty,Billy)	?	$\lambda(44,1)=\{(23,1),(34,1)\}$

Figure 4: An example of a combination an uncertain `saw` x-relation with a `drives` relation into an `accuses` relation including the lineage [6]

6.3 Query Evaluation

In contrast with a traditional relational database, which models one world, a PDB models all the possible worlds. It is possible to query such a PDB by iterating over each possible world and return all the results form the different possible world. Each possible world is a situation where for each uncertain tuple, one optional value is chosen. All possible worlds are thus all possible combinations of all possible values. The number of all the possible worlds grows very large with only a few uncertain tuples. It is clear that query evaluation in this sense is computational intractable [30].

Not only is there the problem of the vastly big set of possible worlds, some of the interesting information available in a PDB is found with the relative uncertainty of tuples. Meaning one might want to find a set of tuples of which the uncertainty is relatively low. One may try for example the query: `SELECT * FROM names WHERE confidence>0.8`. This query will provide a result, however, this result might be empty, to big or just right, this depends on the tuple set and the confidence scores. This confidence score can be used as the minimal certainty of the data to be used. On the other hand, one might be interested in a possible world of which the system is most certain. Relative uncertainty is dependent on the current state of the system. To find items which are relatively certain, additional functions are needed.

Finding tuples of which the system is most certain, needs a different approach. The database management system (DBMS) needs some more understanding of the uncertainty score and needs to have algorithms in determining the trigger values at run-time for creating appropriate subsets. Within the field of PDBs these type of queries are called deterministic and non-deterministic, relatively.

In an earlier study of the field of Data Uncertainty [34] several interesting query evaluation algorithms are discussed. These techniques are based on a relative standard PDB model and are thus applicable on several existing PDBs. This work continues with the default evaluation techniques used in existing PDB projects.

Researchers mostly extend the language already used by the DBMS used within their project. They add functions and operators to cope with the uncertainty scores of tuples. The first example of this is the work of Dalvi et al. [12]. The operator they have designed is the $\approx$ operator, this operator is used to indicate to the system it must perform one of the available similarity functions. For every data type there is a different function, such that the result of the operation is based on the distance between discrete values of the data. Two interesting examples would be the distance between natural numbers or the Levenshtein distance. [18]

There is another issue why query evaluation is a bit more difficult, not all queries can be solved in polynomial time by the PDB. This property can be determined before hand. If a query can be solved by evaluation of the query and performing the defined operations and functions, this evaluation is extensional. On the other hand, if this approach on the given query is computationally hard, the lineage of the query can be evaluated. The query evaluation is than transformed into a problem of calculating the probability of a propositional formula. [27]

Another important aspect of queries is the difference between safe-queries and non-safe-queries. A safe-query allows the probabilistic information to be processed with in the relational algebra. For a non-safe-query this is not possible, the probabilistic information forces several runs of the database engine, which takes relatively more time to solve [11].

An approach to query evaluation with this in mind is to symbolically trace the tuple events during the evaluation and not do all the calculations at each

step. However, this is still computationally complex. Dalvi et al. have designed a method based on the knowledge that not all queries are safe. Parts of non-safe-queries can be rewritten to safe-queries. For the remaining part of the query they use heuristics to create a new query plan, avoiding large errors and using a Monte-Carlo simulation algorithm, which is expensive but has small errors.

This project uses the Trio system as a PDB for the measurements, the rest of this section sets out to show what queries might look like for Trio and what the different parts mean. Trio comes with its query language, names TriQL [6,32]. The current Trio version is build upon Postgresql. TriQL is an extension to SQL. The most important functions of TriQL are `Conf()`, `Maybe()` and `Lineage()`. `Conf()` is used to filter on confidence scores or to select the confidence scores. If in fact the confidence value is not needed, Trio does not calculate it. By using for example `Conf(*)` in the `SELECT` statement one forces Trio to add the confidence scores to the result. If the confidence of the result set is not yet calculated, it still shows if an attribute or tuple is uncertain.

The function `Maybe()` is used to determine wether an x-tuple, tuple with uncertainty, is a maybe-x-tuple. A maybe-x-tuple might not exist, in contrast with just the x-tuple which exist, but only the specific value is uncertain. Finally, the `Lineage()` function, this function is used to make the lineage part of the query, one can filter on the origin or request the lineage of a tuple.

A powerful feature of TriQL is the support of horizontal subqueries, which make it possible to query among tuple alternatives. As explained the combination of an alternative value of each tuple defines a possible world. With the use of horizontal subqueries alternatives of a tuple, are viewed like a table, over the rows of this table the algorithm can then iterate, like iterating over possible worlds. This can be used to determine the expected value of a sum query, but only within a subquery. This cannot be used for the overall query.

Next a simple example of querying a Trio system is shown. Here there is need to query a table named `saw`, and the result must consist of all cars which are not equal to "Mazda" but one alternative value of each tuple must be "Mazda". An example would be, a person saw a car, this was either a "Ford" or a "Mazda", so this query will find this uncertain tuple. Being certain of a "Mazda" is not found, as are the tuple representing the certain or uncertainty concerning other brands.

```
TriQL> SELECT car
TriQL> FROM Saw
TriQL> WHERE car <> 'Mazda'
TriQL> AND EXISTS [car = 'Mazda']
```

To explain this better, Figure 4 shows the database situation which could accompany this example. In this case the query would return a maybe-x-tuple containing "Toyota". The ID of this tuple is 21. Note "[" and the "]" which forces the evaluation algorithm to view this as a "horizontal subquery". As a response the Trio system provides a table with the result and the notion that some tuples are uncertain, if needed one can add the `Conf()` function to determine the confidence scores.

Earlier it is shown that iteration over all possible world is intractable [30]. To tackle the problems in query evaluation several researchers set out to design algorithms which improve the runtime of query evaluation. The literature study [34] looked at Continuous Probabilistic Sum Queries [17], which are sum functions on different events with a continuous probability distribution function, and Top-k Query Results [19, 25], those are results ordered by certainty of the data value.

The first shows one of the biggest problems, in order of number of possible worlds, the combination of events with continuous probability distribution function creates infinite possible worlds. If one represents these continuous function with discrete approximations, the number of possible world is still very large, but will become more manageable. They show that by using boundary values it is possible to prune the set of possible results, given the objective is to compare the result value to another value or to maximise the result. Secondly, is the research on top-k queries, where the requested result is a ranked list of answers based on their probability score.

Both Soliman et al. [25] and Ré et al. [19] have designed efficient algorithms to tackle this problem. Soliman et al. defined two different types of results. Firstly, U-Top$k$ query answers, this is a vector of tuples of length k. This vector contains all the most likely result tuples, where each tuple comes from one possible world. In contrast to this one they define U-kRanks, which sets out to find those data values with the highest the individual score, regardless if they come from the same possible world. Note that the first uses tuples and the second uses data values. A data value might be the attributes of a tuple, but the tuples come from one possible world, since the attribute value of a possible tuple in a possible world is mutually exclusive with the value of the same attribute of the same tuple in another possible world.

Ré et al. [19] set out to provide a new approach of efficient query evaluation, in order to get a set of best answers of size k. They use approximation algorithms to determine the results set and place it in the correct order. The resulting probability values are not correct but are approximation, but Ré et al. show this result is still in the correct order. If more precision is needed or approximation is not possible they use a minimum set of Monte-Carlo Simulations to determine the scores.

6.4 Result Formatting

This last section talks about the result of a given query. Result representation is an important feature of a PDB. As mentioned above, the Trio system provides a result which is of the same format as the content of the PDB. Just like a relational database would provide a table [10]. However, the confidence score of the tuples is not necessarily available, only when needed. In short, the format of the result is similar to the data format, but not completely the same.

Another example of the differences between the format within the database system and the result is the presence of the lineage information. Lineage is mostly of interest for the system itself to determine the needed probabilistic

calculations. However, this lineage information might be queried or requested by the user.

Dalvi et al. [11] are the first to talk about the issue of result formatting within PDBs. It is not just to be defined equal to the existing data, since a lot of information is stored in the probabilistic model, which might not the part that is queried. Looking at the recent PDBS and their data models, the format of the result is generally the same as the format used within. However, the probabilistic modelling is not provided, this is only used to determine the scores. It is possible to create tables from results in most PDBs, in this case they mostly add the lineage information to new table, or create a causal relation in their probabilistic model to make sure the probabilistic relation is maintained. [34]

Another less natural phenomena are the ranked results, based on the probability of each result. One could define the probabilistic information as separate of the data, while this now becomes an "attribute" on which the set is sorted. When retrieving such a result set, the probability of each value is something that is most interesting. While it is possible these scores are not modelled with the data. Every form of result formatting found is consistent with the needs served, which is more important than the question if the format is equal to the format used within the given PDB.

7 From Data to Decision, A Generic System

The original question presented by the MOD, as described in Section 3.1.1, requires more than this project can deliver. To solve the original problem stated by the MOD an overall system is designed, which solves the generic case of that problem. This designed system has parts which do not yet exists. Some of the newly formed questions are later solved in Section 8.

This section will explain the overall system which was designed to eventually solve the problem of the MOD. Most of this is already built as a first version of this system, in the section on future steps the still missing parts will be explained as well as the parts which need scientific investigation. The overall system can be defined as a simulation tool to determine the uncertainty at decision level.

7.1 The Overall System

While the name "The overall system" is not very exciting, it is a name that fits the software project. This system is designed because it provides the context needed to both determine what needs to be investigated and how these things would fit together.

This section starts by revisiting the subset of goals of the MOD and explaining the total set of goals this system aims to achieve. Following this there will be a section on the input of the system. Section 7.4 will explain the first attempt in designing a modelling language to define a decision. Following, there is a section on the data integration steps, these steps will take the input and transform it into the needed information for the decision.

Next, the used probabilistic database system is explained. This overall system uses this PDB like an engine or core. However, not every requirement can be met by the used probabilistic database, so additional features are designed, this is explained in the section on complex data operations.

This designed system helps decision makers in deciding what to do. So by definition of decision support systems, this system could be a decision support system, in addition to the existing set employed by a company. After this the future steps at a system level are discussed to show the current and future state of this software product.

7.2 Goals

The expressed need of the MOD was to know the quality of the data used at decision level. But the problem is that after integration steps the quality score is unknown, since there are no rules how to calculate quality scores over combined data.

After some discussions with the supervisors and some research into different fields there was reason to believe probabilistic databases are something that could help in this quest. Probabilistic databases do possess the rules to determine uncertainty of result data based upon input. The question was formulated

how to place data and its uncertainty into a PDB based on scores determined by data quality techniques.

The goal of the system became to show the uncertainty of result data used in a decision. Furthermore, it must show the chance of making a mistake, given it is a binary decision. If costs are defined, the expected value is a nice feature, this is fairly straightforward when the probabilities are known and for a given organisation money is of great interest. This project confines itself to binary decisions since this way it is possible to let the system decide which option is best and provide expected values.

To understand when a decision should get a positive or negative recommendation, trigger-values must be defined, such that the system can provide this recommendation based on the result data. The system defines triples consisting of the data item, an operation and a trigger-value. In Section 7.4 this will be discussed in more depth. These triples are called "triggers" within the overall system. The next important step to tackle is to get all the needed data into the system, together with the uncertainty information.

The system can do more than these goals require. During the project new insights were gained by talking to people, using the data and encountering data quality problems. One very interesting possibility that was taken into account when building this system was the possibility to trace back the origin of the data quality issues encountered at decision level. This way someone knows where to improve the quality in order to be more certain about the option to pick.

The notion of the costs of a mistake make it possible to determine a budget to improve the data quality. If one could make €1,000.– more each week by eliminating part of uncertainty, that money could be spent on improvement of the quality. Of course, one most wisely choose and measure the relation between the improvement in quality of the data and the lowering of costs one is expected to have.

7.3 Input Data

Given that the organisation in which such a system as this overall system will be used, already has a large amount of data in place, enforces the design to accept all standard data types. However, as will be discussed some data types will have extra possibilities, like the definition of probability distribution functions. This is seen separate from the acceptance of input data types. This means some datatypes are converted to character strings just for storage and the maintenance of its uncertainty information.

This section on the input starts with a section on the requirements, followed by what is designed and created in the system. Finally, this section ends with a discussion on the design.

7.3.1 Requirements

The most important feature at the input part of the system is that it should accept all data in a given organisation. It will not be designed as a replace-

ment for a current database systems employed, but as an input store for the PDB to perform the later defined data integration steps with the accompanying probabilistic calculations.

Since most modern PDBs are built upon an existing relational database or equivalent model, it is clear that the data present in an organisation will fit. At the MOD all the data encountered by this project was stored in relational databases.

7.3.2 Design

The difficult part was to create a universal input, which could take any input in table form, store it and use it within functions not defined within the PDB. In order to add information or the later explained alternatives, some semantics are needed of the data types, in order to determine neighbouring values. This system assumes some table definition is available and stores the integers and doubles accordingly, all other datatypes will be converted to text in the current version. In the following section the definition of probability distribution functions will be discussed, these are purely discrete at the moment, so the support of double values is limited to prevent the loss of information in case of currency for example.

As defined the system accepts doubles as a datatype, however, doubles in this case are discrete values. For every organisation it is important to support currency values, these are discrete values of the lowest possible currency unit. But it is stored as a value with two decimals behind the comma, a lot of the times database systems store this as doubles instead of a special discrete value which is presented like #.##. In this project for ease of use and ease of implementation doubles are used to support fractal values to some extend.

During import the uncertainty of the data must be defined. In Section 8 the best way to determine the uncertainty will be discussed. This current section assumes the uncertainty score is correctly defined and the user knows how to model the uncertainty.

The user is able to choose between attribute and tuple uncertainty. The user's decision should be based on the uncertainty that is determined during the data quality measurements. The Trio PDB will by default use the attribute uncertainty to determine the tuple uncertainty, if only tuple uncertainty is defined the attribute uncertainty will not exist as separate knowledge. The system provides several ways to transform the uncertainty score found into the optional ways to model this.

The models and accompanying transformations are:

1. Tuple uncertainty
2. Attribute uncertainty
3. Attribute uncertainty based on tuple uncertainty
4. Attribute uncertainty based on tuple uncertainty, with PDF

5. Attribute uncertainty based on tuple uncertainty, with dependency among attributes

6. Attribute uncertainty based on tuple uncertainty, with PDF for only numeric values

7. Table uncertainty

8. Relation uncertainty, with PDF: $P(R = r) = p$ & $P(R = NULL) = 1 - p$

Tuple and attribute uncertainty are implemented according to the explained definitions in Section 6.2. Relation uncertainty is equivalent to attribute uncertainty, but only applied to identifiers pointing to tuples in other tables. If the determined uncertainty is tuple uncertainty, but the user requires to store attribute uncertainty in light data integration steps for example. The system can determine an appropriate attribute uncertainty value based on the tuple uncertainty. This is based on the following equation: $\prod_{i=1}^{n} P(attr_i) = P(Tuple_{as-is})$, where n is the number of uncertain attributes.

The PDFs are, as discussed earlier, discrete. They are determined based on domain and granularity. Domain is the total domain of the histogram representing the PDF and granularity defines the size of the discrete intervals. Granularity was designed to support different discrete steps than naturally exist. This way the PDF is simulated by using the stored values, the alternatives. These alternatives are then used as the representatives of the interval defined by the granularity, which is thus the column width of the histogram. Within the later discussed Figure 5 these values are shown. The "D" together with the bar above, shows the domain. Additionally, the "G" and the accompanying bar is what the granularity means. Finally, the circle contains the original, known, data value.

Alternatives as designed in the Trio system are a very good way to define discrete probability distribution functions. The probability of two new alternatives is: $P(alternative_n) = (1 - P(original))/2$, where n indicates which alternative is determined. In the case of more alternatives the system further divides the $P(non\_original)$ into twice $2/6$ and twice $1/6$ for the other alternatives.

Figure 5 shows a visualisation of a set of alternatives as explained in the last paragraph. The system will tell the user that the answer is 10, with 90% certainty. However, according to the defined nature of that attribute, the alternative values are: 8, 9, 11 and 12. As explained what is put into the system is the value "10" with a score of 0.9 and the a domain of 2 and a granularity of 1. These values are to be viewed as settings of the system, the formal definition of "domain" would not agree with this use.

Based on the mentioned settings, the system defines the set of alternatives as stated above. The probability given to, for example, alternative "8" is based on the following calculation: $P(alternative_8) = ((1 - P(alternative_{10})/2) \cdot (1/6)$ since the area of the pillar at 8, is $1/6$ if the combined area of 8, 9, 11 and 12, by the definition used within this system. One can see this is just a rough

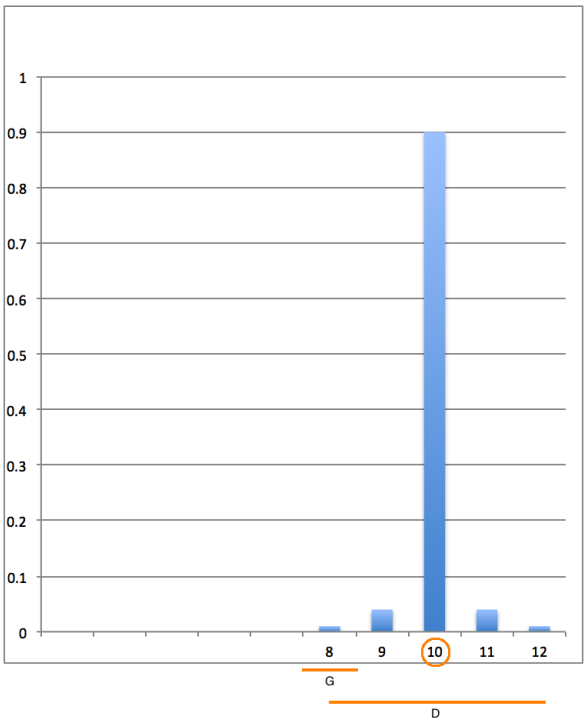

Figure 5: A visualisation of a histogram defined by alternatives

estimation of the real probability distribution, but it only has to serve the user in understanding the possible variations at decision level. The most important value is "10" and its uncertainty score.

First of all, this PDF support is made to mimic the real situation, for example, to account for misspelling or counting errors. And is designed as a tool for the user, who needs a simple approximation and can't define the real PDF. Later in the determination of the certainty of the outcome of the decision the resulting PDF will be used to determine the correct probability of the outcome. This means, in the example of Figure 5 that we are 100% certain the data value is below "13" and above "7". Furthermore, we are 95% sure that it is "10" or lower. Later in the design of the XDL it is shown that the design of how to determine the outcome of the decision, information is needed to know how to relate the PDF to the trigger-values.

The option to only define a PDF for numeric values, means that uncertain attributes of the type integer or double will get alternatives, based on domain and granularity. Non-numeric attributes will only get a confidence score. Note that Trio stores the uncertainty information as confidence scores. These values are probability values but they might not add up to one within a set of mutual

exclusive values, as explained this then means the value might not exists or is unknown.

Dependency among attributes can be chosen, this way the system enforces the fact that, if attribute one is correct, the others are as well correct. This behaves like tuple uncertainty, but the information remains at attribute level, so a single attribute has a correct uncertainty value, but the combinations have the same uncertainty as well. This option is not native to the Trio system, but implemented within the overall system.

Table uncertainty is something not natively supported by most PDBs. This functionality was implemented to serve as a form of benchmark. The calculations needed to determine the result set uncertainty are based on probabilistic calculus and the operations as defined in the Trio system. As stated this was implemented as a form of benchmark, because with the use of table uncertainty, the uncertainty can be modelled at table, data-set and data-source level. This way there are less complex calculations in determining the end result and this make it easier to validate the process.

Not all attributes of a given tuple are uncertain. Attributes maintained by the system itself might be assumed true. An example of this would be ids and relations. To serve this need the user can define a set of uncertain attributes during import. The uncertainty scores will only be related to these attributes. In the case of tuple uncertainty, this is somewhat cumbersome and changes the structure of the table, but it is quite easy to define only the id as certain and all the other attributes as uncertain.

The only missing natural way to determine attribute uncertainty, based on tuple uncertainty, is one which is essentially the same as number 1 but with weighted influence on the tuple uncertainty. The calculation would than be: $\prod_{i=1}^{n} w_i \cdot P(attr_i) = P(Tuple_{as-is})$, where w_i is the weighted influence of attribute i. However, implementing this function was not the problem, but creating a user friendly import function to support this seemed to take a lot of time and was left out of this project.

7.3.3 Discussion

During the experiments it will become clear which of these import functions will not be suitable, this functionality will then be removed from future versions. It is also be possible the project determines the differences are to small to be of any interesting influence, in this case the user will be free to pick that option which fits his situation best.

However, it is in no case appropriate to change the import function to improve decision certainty, but this will not be possible since the erroneous import functionalities will be removed, the difference will be insignificant or the resulting situation will not fit the organisation.

This way of accepting the possible inputs will serve both the research project as the generic overall system.

7.4 XDL, eXtensible Decision Language

This section will explain the designed decision modelling language. This language will tell the system which data integration steps to take, what data items are used to determine the answer to the decision. It will also compare the found values to the trigger values and provide the appropriate answer, in combination with the uncertainty of that answer. Finally, it is possible to define costs of making a mistake and the benefit of not making that mistake.

First, the requirements are defined, this explains what the goals of the language are and what parts are needed within the design to achieve these goals. This section ends with a discussion on the design.

7.4.1 Requirements

The language needs to define a decision and the data retrieval needed to make that decision. This language is based on a generic situation and validated with the situation of the MOD in mind.

Most importantly, it must define the decision to be made. It is given that this is a binary decision so two outcomes must be specified. These two outcomes are positive and negative, like yes and no, however, the meaning of these terms is for the system not relevant. What is most relevant is that is has a way to pick one outcome and determine the certainty, the other outcome can be defined as "if not outcome 1".

When defining outcomes the system must be pointed to data-items which influence the outcome. Values for these items must be defined to tell the system that an outcome must be picked when the data item has a certain value. However, sometimes an item must be lower than a certain value and sometimes equal or greater than, so this also must be defined in the language.

As stated in the paragraph above, single values or a set of single values can be checked to determine which option to pick within a decision. So the system must take steps to retrieve those single values. In the language there must be room to specify a query, in both SQL and the extension used by the PDB, to retrieve the needed data. However, since in a single query not everything can be retrieved, the language must be able to define a sequence of queries, where dependency among queries is possible.

Finally, it is important that the language is human-readably to some extend. This way the validation process later will be much less cumbersome, since the created documents are understandable, with some background knowledge of the system.

7.4.2 Design

The first version of this language was called SDL, as in Structured Decision Language. The format of that language was so strict the parsing was easily implemented but very sensitive to mistakes within the SDL instances, even the use of spaces and newlines was strictly defined. That language was 1 to 1 translated to an XML version, meaning only the structure was replaced by an

XML structure. This way the parser is much more robust and is built on solid libraries. This new language was creatively called XDL, eXtensible Decision Language. Note that this project does not aim to provide a new way to model a decision, the goal is to model a decision in order to orchestrate this system.

During research into existing decision languages the work of Evens [16] provided great insight. To understand decision modelling in light of statistics and data analysis this work is good place to start, especially due to the accompanying workable examples.

First, a sample of XDL will show the general structure and possible content. This sample will be used to explain all the parts of XDL in the following paragraphs.

```
<decision>
    <question>
        Should we buy, in order to have a sufficient stock level?
    </question>
    <options>
        <option>
            <longname>
                Yes, we must buy
            </longname>
            <name>
                yes
            </name>
            <benefit>
                100
            </benefit>
            <loss>
                200
            </loss>
            <trigger>
                (stock_level_1 =< 1200) [AND | OR (...)]*
            </trigger>
        </option>
        <option>
            <name>
                no
            </name>
            <benefit>
                0
            </benefit>
            <loss>
                10
            </loss>
            <trigger>
                otherwise
            </trigger>
        </option>
    </options>
    <data_integration>
        <result_set>
            <name>
                product_count
```

```
            </name>
            <query>
                SELECT *
                FROM products, stock
                WHERE products.id=stock.prod_id
            </query>
        </result_set>
        <result_set>
            <name>
                stock_level
            </name>
            <query>
                SELECT id, aantal AS total
                FROM product_count
            </query>
        </result_set>
        <information_item>
            <name>
                stock_level_1
            </name>
            <operation>
                SELECT total
                FROM stock_level
                WHERE id = 1
            </operation>
        </information_item>
    </data_integration>
</decision>
```

The root node of this XML document is the `<decision>` node, this contains everything needed to define the decision. The first interesting node is the `<question>` node, this is a human readable text and the essence of the decision. In this case "Should we buy, in order to have a sufficient stock level?" is mainly to support the user.

Next, the `<options>` node. Originally, this language is designed to support more than two options to pick from, but within this project the scope is set to binary decisions. So "questions" with answer "yes" or "no", 1 or 0, true or false.

First, one is able to define a `<longname>`, a verbose human readable answer to the question, if the trigger is met. The `<name>` node is unique identifier of this option and is used within the system and is required. Next, benefit and loss can be defined. Benefit is the gained value if this option is chosen and appears to be correct in the real world. Loss is the lost value if this option was mistakenly chosen. This information is used to determine the expected value of the decision result in a current uncertain data situation.

Finally, there is the `<trigger>` node within the option. This node contains a `triple`, a triple consist of three attributes, an information-item-name, a comparison operation and a trigger-value. The information-item-name points to an information-item available in the PDB or defined in the XDL document, this will be discussed later. The trigger-value is compared to the found value in the information item based to on the comparison operation. If the result is true, this option must be picked based on the available data. If no triggers result in a

"true" value, the option containing a trigger with the term "otherwise" is chosen. This, however, is not guided by the uncertainty knowledge.

After the explained "triple", one can see the following: `[AND | OR (...)]*`. This was originally designed in the SDL version and fits that language better in the sense of syntax. However, what this enforces is still valid in the XDL version. This explains that it is possible to define a more complex trigger based on more values. It is possible to combine a set of triples to which the data must adhere before the option is chosen. So one can define that several stock levels must be below a certain value before one starts to purchase more items.

Before an option can be picked the data needed for the triggers must be accumulated. This is done using data integration steps. An example can be found in the sample XDL document within the `<data_integration>` node. One can define two types of intermediate results, a result-set and an information-item. A results-set is a temporary database table and can be created using SQL, or the extension, TriQL. An information-item is a single data value, this value can be obtained using SQL, TriQL, a system-function or a combination of these. The system-functions are standard functions to obtain interesting information from a data-set, these currently are `SUM()`, `PRODUCT()`, `MEAN()`, `MEDIAN()` and `MINUS()`. These functions accept a table of one column and return a single value. To make the system somewhat more user friendly the information-item could be any table, but only the first value of the first row will then be used. In the next section these data integration steps will be discussed in more depth.

The most important aspects of the data integration node is that it essentially is a sequence of queries which are performed by the system. To get the needed result some of the queries are dependent on other queries within this set of queries. This is possible if the names of the table created are defined in the "name" node of a result set or information item. The system then creates a table with that exact name, so later queries can use that table in the FROM-clause. The system will check if the queries are in correct order and places them in a correct order if needed, however, circular dependency is not possible.

7.4.3 Discussion

Firstly, the language is clearly able to model a decision. It can define the possible outcomes and the benefit of picking each outcome correctly, as well as the penalty of picking each outcome by mistake. Furthermore, one can define a boolean condition for one of the outcomes, this way the system can determine which outcome to pick. These triggers that are defined are also a good basis for defining the probability distribution over picking the correct option, in the sense of probability of positive, false-positive, negative and false-negative

A sequence of queries can be ran and eventually the information items used in that boolean condition are created. Assuming the set of data integration steps eventually defines one information item for each items needed in the boolean condition, this will work. However, it is the responsibility of the user the define a correct XDL document.

For some parts of XDL it is currently quite clear they need some improve-

ment. The first thing that must be created is a solid way to define the positive and negative outcome. This is not important for the actual outcome of the system, but makes it possible to create a more formal and better report in the end, so the system can add semantics to "yes" and "no".

7.5 Data Integration Steps

Within the section on XDL the node `<data_integration>` has been briefly discussed as part op that language. This section will explain what this enforces the system to do. The data is assumed to be stored in the PDB, together with the uncertainty information. The requirements of the data integration feature will be discussed in Section 7.5.1. Followed by the design made to achieve these requirements. In the final discussion section the result and the requirements will be compared.

7.5.1 Requirements

By now the XDL document is parsed into a set of tasks which the system must perform in order to get to the decision. The data integration set, is a sub-set of all those tasks, but probably the largest one in any given instance.

The set of operations performed by the data integration step need to create a set of information items which can be used to determine the outcome of the decision. Looking at the underlying software, supporting this data integration module, it must use the probabilistic database and of course its content. The following section will explain the data integration process.

7.5.2 Design

A given data integration step either creates a result set or an information item. These two types are very similar, the main difference is that a result-set operation creates a new table. The storage within the PDB makes it perfect to keep track of uncertainty and lineage, since the Trio system takes care of this. While an information-set operation creates a single data item. This item is still stored as a table within Trio, again to use the power of Trio. This explains why both results-sets and information-items can be created using SQL and TriQL. An information item is handled as a single data-item to make sure the correct value is eventually chosen to use in decision making. The name "information-item" is chosen to indicate it is the result of data integration and to be used within the decision or as a single value in later data integration steps.

The functions mentioned in the Section 7.4.1 are performed under data integration. These functions take, as explained, a list of values and combine them accordingly. This means the input should be a table of one column, and the result is one row. The `MINUS()` function must be used carefully, since it is dependent on the first value encountered. Within these operations, the rules of probability calculations are strictly followed.

Another problem faced by these function was how to handle alternatives. As explained alternatives are like probability density functions and the theory of probability has rules how to perform such operations. [14] Some of the time this means a very large set of alternatives is created in the result.

7.5.3 Discussion

Section 7.4.1 ended with the notion of the fact that data integration is essentially a sequence of queries. By designing this part as a sequence of queries, we are sure that every process which retrieves its data by the use of queries and not some unknown software process, this system can mimic that retrieval process and thus perform the required uncertainty operations.

Two things are mainly of influence on the uncertainty calculation, the numbers of source items and the number of joins. Joins are the joins known from relational algebra. Translating these to terms known in the field of probability and statistics, source items would be events and joins would be combinations of those events. Due to this knowledge it is clear that data integration must define source data items and the operations to create the resulting information items. Supporting all the possible operations in all possible organisations is not the most important goal, more important is to at least support a sequence of queries, containing the correct set of sources and joins, which this system now does.

7.6 Probabilistic Database Support

The system uses Trio as a core for the tasks of managing the data and the probabilistic calculations. As explained Trio was chosen since it is the only available probabilistic database which supports most of the needed functionality. Furthermore, it was one of the few PDB projects which was actually implemented to such a level it could be integrated within the system. This also means it is currently the only one that can be used to support the system.

Trio is actually a set of functions for PostgreSQL in combination with the Trio parser, called TrioPlus. This parser translates the TriQL queries into the needed SQL queries to be performed by PostgreSQL. If one looks at the tables via PostgreSQL and not via TrioPlus, the expected tables are not there. Trio stores the data somewhat different in order to maintain the uncertainty and lineage information and still be able to efficiently query the data.

The default functions are loaded into PostgreSQL and are available, but without the parser these are not very useful. This is why it was needed to use the tool TrioPlus within the system to perform all data operations concerned with the PDB. Luckily TrioPlus is a command-line tool and can be used within a Java application. The response provided by a command-line tool is command-line text. To be able to use this within Java, a Trio-package was designed.

The most important class in the Trio-package is the Trio class. This class contains the `executeQuery()` and the `executeUpdate()` functions. ExecuteQuery gives the query over to TrioPlus and parses the response as a Trio-

Table object. This function is ideal for select queries. On the other hand the queries which do not provide a needed response, like update queries, can be performed using the executeUpdate function.

The mentioned TrioTable class is an object representing a database table, with uncertainty information. At the current version the lineage information is not included in this object. This is not a problem since the lineage is used and maintained only by the Trio system and it can still be queried and used but then as the response data itself. The TrioTable class consists of a TrioHeader class, maintaining the names of the column, the location of the uncertain attributes and the position of a special confidence column provided by Trio.

The other attribute of the TrioTable class is a set of rows, TrioRow classes to be precise. These represent, as expected, the rows of the table. Each row consists of a set of TrioAlternative classes, where a single TrioAlternative class represents a set of mutual exclusive alternative TrioRecord classes. These TrioRecord classes are the data values. Please note that the name TrioAlternative is not the best name to show the meaning of this class.

It is clear that the design is loosely based on the API of the Java PostgreSQL library, which made it more easy to use during development.

7.7 Complex Data Operations

This system must from some point on adhere to the principles of data uncertainty in every step. A complete system does not fit the scope of this project, but some indications that at least a lot of needed data retrieval operations within a given organisation can be performed, must be given.

This is one of the reasons why as much as possible is done by Trio, a proven probabilistic database system. Which was designed to serve business needs of organisations. Unlike some other PDBs, which are designed towards serving scientific needs or measurement systems.

With Trio relative simple uncertainty information of data can be stored, which is exactly what companies need. For the MOD this is quite enough to store the data and store its uncertainty information.

The fact is that the operations of the system are the same as defined in Relational Algebra. All accompanying probabilistic calculations are ready for use. This means that if a join is presented to the system, the resulting data set will get the correct confidence scores. Of course, the difficulty is not in defining the correct probabilistic calculations, since the events are clear, so the calculations are fairly straightforward. They are, however, great in numbers and the result is not known before hand. This is one of the reasons the system must sometimes iterate over a set possible worlds.

However, a SUM function which fits the needs of the MOD was missing. This one was implemented for the project, as explained. Other very useful functions, not existing in SQL, were indeed implemented by the Trio team. Examples of these are Lsum, Hsum and Esum. Respectively meaning, the lowest possible sum of the records, based on the alternatives, the highest and the expected value of the sum operation. The issue was that these functions cannot be used for the

final result of the query, only within horizontal subquery. This SUM function is a good example of a problem which can only be solved by iterating over all possible worlds. The other defined functions, Product, Mean, Median, Minus, work in relatively the same way. Additionally, as earlier explained, a sequence of SQL queries should suffice.

7.8 Decision Support

Eventually, the system needs to support the organisation, but this system must also be a tool for the decision maker. Since the main focus of an organisation is on both costs and quality, the decision maker needs additional data to understand the information presented to him. This section will focus on the needs of the decision maker.

By now it is time to define the term "information", in the last paragraph the term is used to indicate the data used by the decision maker to make his decision. Additionally, the decision maker used other data to understand his data, to avoid confusion the term "information" is used for the end result of the data integration steps. For this work information is defined as follows.

Definition 7. *Information is data which is actionable and directly used in decision making*

Within the overall system the term "information" is not of importance, it is just data. However, to make the situation clear how the data is used, the term information will show that the given piece of data is used within decision making.

In 1986 Elam et al. [7] reviewed literature on different Decision Support Systems (DSS). This work provides a clear and useful definition of a DSS.

Definition 8. *"Decision Support Systems are computer-based systems whose objective is to enable a decision maker to devise high quality solutions to often only partially formulated problems"* [7]

7.8.1 Requirements

Part of the goal of the overall system is to augment the current decision support system with the knowledge of the uncertainty of the data. So the overall system must provide the uncertainty information about the result data it provides the decision maker. It must also show the decision maker which option is then the best one as far the data is concerned. Finally, the probability of making the correct decision must be provided. This final piece of information is not always the same as the uncertainty of the data on which the decision is based, the property of the data and the relation between the data and the trigger value is of influence on this, as explained in Section 7.3.

7.8.2 Design

To achieve the requirements explained in the last section, the system will provide all the needed data and will present this to the user. The system cannot provide different data values than the already existing database systems can. It will show the data which the database can provide. The system will determine the uncertainty of that data, as a confidence score.

Since the system will compare the result data to the trigger values, it can determine if the uncertainty is of influence to the result of the decision. In order for this to be a valid prediction the uncertainty information and the triggers must be used correctly.

A situation where the data is not very certain, but the influence of the uncertainty on whether or not the best option is chosen is non existing would be the following. An organisation can have a stock level of 100 as the trigger value for restocking, the database contains 500 as the current stock level, with a confidence of 0.9. In this situation the system cannot determine the possibility of making a mistake, since there is no information on what the value could be if not 500. If a PDF was defined over the data value "500", as simple as in Figure 5, the system can show that the chance of making a mistake is 0. In Section 7.3 was already explained how the PDF is used to determine the confidence in the outcome of the decision.

Finally, based on the confidence in the outcome of the decision, the system can calculate the expected value of the gain of the decision. Which in turn can be used to determine the costs of making a mistake.

7.8.3 Discussion

The system shows the decision maker the information needed to make the decision. This part of the system does not yet provide new possibilities to an organisation. This information is augmented with the confidence in that information, this is something new. This is a way of showing the influence of the lack of data quality on information used in decision making.

Furthermore, the system provides the confidence in the option to choose, if this is possible. This helps the decision maker understand the influence of the lack of data quality on business performance. This confidence information in the option chosen is then used to determine the costs of making a mistake. This information shows the costs of the lack of data quality.

The overall system, once fully implemented, will be a valuable addition to the existing decision support systems in place in organisations. Additionally, it will help data users in understanding and explaining the data quality problems of the organisation.

7.9 The Model in Short

To summarise how the designed system is build, the following list shows how the parts are related.

1. A decision outcome is purely based on the available data.
2. The PDF of this outcome, depicting the probability wether the outcome was correct and the probability it was mistakenly chosen is defined by the PDF of the used data values and the triggers defined by the user.
3. The PDF of the data values is determined by the use of the PDB, which in turn uses the data and probability information, which was put into the system by the user.
4. The probability information of the source data is determined by use of data quality metrics, as will be defined within Section 8.

7.10 Future Steps

This system is in no sense complete, one of the reasons is that the focus of this work is on the measurements that needed to be done to answer the research questions. The measurement tool is part of this overall system, with some changes which made it possible to run batches for example. This will be discussed in the next section. This section sets out to explain what needs to be done before this system can be used by a business user trying to learn more about his data.

First, of all the data input needs some work. Currently all data-set must be imported before the data integration steps are done. This is not very efficient. Currently the sources and their uncertainty information can be defined, but not the queries to determine the sub-set of the data needed. That way only the used data could be imported. Another way to do this would be to maintain the data sources or pointers to the original data and the uncertainty information and only grab the data at run-time. In the situation of the MOD this was not possible, since it was not permitted to query the actual databases.

The next issue is that the system only supports integers, text and doubles. In this case integers and doubles are treated the same, but to make sure the decimals remain intact the double support at database level was added. Every datatype has its characteristics, some of these define for example how discrete values are determined, which is interesting to define a PDF. An example is the difference between integers and text, when looking for an adjacent value, for integers this might be the two natural numbers next to the current one. For text a distance function must be defined, or chosen, and most of the time the set of items with distance "1" from the current value is relatively large. The current system support the distance as defined in natural numbers, for the integer values and just shows the distance for the text values. So at the moment another system must show what the possible values for a text are, given the current value and the defined distance. A string of 4 characters, in a 26 character alphabet, has more than a 100 direct neighbours, given the use of a Levenshtein distance. [18]

Another shortcoming of this system is that only Trio is supported. Though this is party due to the fact it was the only workable system, which supported the needs of the MOD. We hoped to have time to built an abstraction layer on

top of the probabilistic data layer. This endeavour is greatly dependent on a generic data model for probabilistic databases. This is a large research project in itself, in our eyes.

The set of available data operations during the integration steps is limited. We do not know if this set can serve all the existing needs. This looks to be a very big research issue. However, all the data operations with database systems an organisation can define currently using relational algebra, can be use with this system.

Functions to clean and merge data based on the uncertainty are available in Trio, or can be defined with TriQL or SQL. But these are not tested in depth. If all needs of a generic organisation are met with this system is unclear.

This project was concerned with determining if one can use data uncertainty techniques to determine quality or certainty of data at decision level. This means the implementation of the overall system was guided by this notion. This results in a system which could be more user friendly or operate more efficiently if time and effort are allocated for these features.

Finally, the decision modelling language is very simple. This language could be improved in order to be more efficiently parsed by the system. Or be improved to better model the decision and its components. However, this is another project in itself.

8 Modelling Data Quality

This section takes the reader through the set of research questions and hypotheses formulated during this research. It shows where the research started and why. It shows how it evolved to eventually finding an answer to the main question which could not be refuted by the available evidence.

First, this section will explain the measurement tool and the generic measurement setup. This explains how the overall system is used to run the needed steps to get from the source data to the information needed for a decision. What must be clear by now is that this tool does not provide a different answer that the original relational database system would provide, it adds the, hopefully correct, uncertainty information to that result.

The research was mostly done with the data-set which is owned by the Clothing and Personal Gear department of the MOD. In short this department will be called KPU. The master data-set of the KPU consist of about $60,000$ data-items.

The sections following the section on the measurement tool are the different research questions and hypotheses. All of them are here because they brought the research closer to the result, namely finding if it is possible to model the results of the data quality measurements, using data uncertainty, without losing the information gained from those measurements. If this is possible, finding out how this can be done is the next logical step. If this is possible for the case of the MOD, we believe it is possible for most organisations.

8.1 The Measurement Setup

As noted before, the measurement tool uses everything in the current system. It can be seen as a shell that starts the system automatically for each run within a batch. There is no need for user interaction and results are stored in output files. This measurement tool is used for every measurement-run within in this chapter.

The first difference between the measurement tool and the overall system is that the measurement tool uses the system. But if the measurement tool is defined like this, it is only a script that reads set-up files and then runs the system and points it to the needed XDL file and data sources.

For each measurement run the data source is defined, the confidence score of each attribute, tuple or table is defined. Due to the nature of the data quality measurements the confidence score of each attribute or tuple, depending on the exact setup, for a given data set is the same. Actual individual scores are unknown. In general this is fine, because a confidence score of 0.9 of 10 different tuples, means that 1 of these is expected to be incorrect, which is exactly what is found using the data quality measurements.

The measurement tool runs all experiments defined using the experiment files within the experiment folder. These files point to the data sources, define the uncertainty attributes of each table and the determined uncertainty scores. If the measurement takes a look at the influence of using the alternatives in

Trio, thus defining several probability distribution functions. The granularity and domain must be specified in the experiment setup file. In the experiment setup file the transformation to be used must be defined, these were explained in Section 7.3.

Both the measurement tool and thus the overall system are sufficient for the experiments and the situation at the MOD. The sequence of queries which are ran each time are a sufficient simulation of the data retrieval actions of the MOD. All the decisions made in the situation picked for validation are based on data which can be retrieved using a sequence of queries and no complex data retrieval operations outside this domain. Furthermore, as noted earlier in Section 7.5.2 to correctly simulate the decisions made the number of source items and the numbers of relational joins or operations is important, together with the correct data uncertainty score, based on the data quality.

Additionally, it is possible that the confidence in a picked option for a decision is different from the confidence in the data value on which the decision is based. This is due to the fact that the trigger value is somewhere within the domain of the PDF or completely outside the possible data values. This influences the confidence in picking the correct option, but the confidence in the known data value is still determined by the original data quality score and the data integration steps.

8.2 A One-to-One Translation

Every research project has to start with the evidence there is and the first research question one can image, which fits the evidence. So the first way of working we devised is to take the data quality score of the data set and use this score as a the data uncertainty score. The resulting uncertainty value might be a correct modelling of the data quality. Within this principle we left room to try out all the different type of attribute, tuple and table uncertainty, but we though these would be fairly equal. During the discussion of this first experiment al lot of examples will be shown. In the discussion of the later experiments we will only show important results.

8.2.1 Methodology

First of all, the data quality score was determined. This was done by taking the mean score of all the different rules. These rules, however, are a subset of the total set of rules defined by CBG. Only the rules which apply, according to CBG, to the current data-set were used, furthermore, rules that apply multiple times to a data item were skipped. At this time it was unclear how to correctly take those into account. Later, with the new found insight, these will be taken into account correctly.

Eventually, the exact rules do not matter, as long as the validation is done with the same rules in mind, for example, one cannot validate the results using one rule while the uncertainty was based on another rule. So, as long the rules

are consistently used, we can use a subset. The determined subset of rules is used in every future experiment, unless stated otherwise.

A few typical simple decisions were designed. They all need a few queries in order to retrieve the information from the imported data set. An example of these decisions is the one shown next.

```
<decision>
    <question>
        Is the value op the total stock of item_1 below 1200 euros?
    </question>
    <options>
        <option>
            <longname>Yes, we must buy</longname>
            <name>yes</name>
            <benefit>1</benefit>
            <loss>1</loss>
            <trigger>(stock_level_1 < 1200)</trigger>
        </option>
        <option>
            <name>no</name>
            <benefit>1</benefit>
            <loss>1</loss>
            <trigger>otherwise</trigger>
        </option>
    </options>
    <data_integration>
        <result_set>
            <name>product_count</name>
            <query>
                SELECT *
                FROM products, stock
                WHERE products.id=stock.prod_id
            </query>
        </result_set>
        <result_set>
            <name>stock_level</name>
            <query>
                SELECT id, (prijs * aantal) AS total
                FROM product_count
            </query>
        </result_set>
        <information_item>
            <name>stock_level_1</name>
            <operation>
                SELECT total
                FROM stock_level
                WHERE id = 1
            </operation>
        </information_item>
    </data_integration>
</decision>
```

This is a simple adaptation from the example from Section 7.4. Here actual new information is created from the source data. Every query used in tests is created with two requirements if at all possible. First, new data creation and second samples of the result could be manually checked. This query determines the value of the total stock of the item with `id=1`.

Validation of the result is done in two ways. First of all, the rules are used to determine which item actually passed or failed. This way the newly created data-set is "quality checked", this is not possible for the live data-set of the MOD, but we can do this for our experiment snapshot. Secondly, a few samples of 100 items are randomly chosen and evaluated, based on the rules and common sense. This last check is somewhat biased towards opinion, common sense is limited to "looks fine" and we cannot do consistency checks outside the data, however, it is done to verify wether the experiment ran correctly and because the suspicion exists that the data quality rules are to strict in relation tp the actual needs of the data-client (KPU).

8.2.2 Measurements

The eventual data quality rules set that applies correctly to the data set of KPU is of size 37. A lot less than the earlier mentioned 400, this is because a lot of rules are specifically designed for other data-sets. However, the rule set checks all or almost all the attributes and a lot of them look to support the data client needs.

Of all those rules we calculated the mean data quality score, not yet knowing if this is the best way to determine the quality score. But based on how this information is determined and used throughout the MOD this does not seem incorrect. The mean data quality score is 98%, a number which looks very nice at first sight. So we determine the data uncertainty score 0.98. Since this is the first experiment all results will be shown to provide the reader with the needed information, later on only the interesting figures will be shown.

The result is shown in Table 1. Note that for the PDFs the default settings are such that granularity is one and the domain is one. This means the alternatives are all the items that are 1 distance unit removed from the known value, for example, if the value is 5, than 4 and 6 are the extra alternatives.

Table 1 points to the next tables, this is because the result consists of a set of alternatives. The result data is found under "total*" in these tables, where the * means it is an uncertain attribute. The first one, Table 2, contains several equal values, these can be merged. This comes from the fact that other attributes of the tuples have alternatives as well, but these are not shown in the result. Trio maintains the different alternatives for this. The overall system only gets the resulting data item, on which it must base the result of the decision, so at that point it can merge equal values, and create a new table as shown in Table 3. Tables 4 and 5 are fairly straight forward.

In Section 7.3.2 it was explained that the confidence in the decision outcome might differ from the confidence in the data value. This can be seen in Table 1 where there is a difference between the value under "confidence" and under

Table 1: Results over experiments

Type	result-data	confidence	P(correct decision)	P(mistake)
Tuple uncertainty (TU)	1200	0.96	0.98	0.02
Attribute uncertainty (AU)	1200	0.92	0.96	0.04
AU based on TU	1200	0.96	0.98	0.02
AU based on TU, with PDF	See Table 2	See Table 2	0.98	0.02
AU based on TU, with dependency among attributes	1200	0.96	0.98	0.02
AU based on TU, with PDF for only numeric values	See Table 4	See Table 4	0.97	0.03
Table uncertainty	1200	0.96	0.98	0.02
Relation uncertainty	See Table 5	See Table 5	1	0

"P(correct decision)".

It must be noted that the very small confidence scores are not relevant. These can be easily ignored. It can also be argued that some of the really small values are the result of floating point errors. But to show the reader the complete response of the system, these values are left in. After merging such result tables, these small scores will mostly disappear. In future versions of the system insignificant results must be removed in order not to confuse the user, if they remain after merging.

Other decisions were also used in the experiments. All turning up with similar results showing the system does what is expected. And the built-in functions and the implemented functions all worked correct, in the context of probability calculations.

First of all, the resulting scores are in line with the expectations. This is because the PDB determines the correct probabilities of all resulting tuples. Even the mentioned SUM function determines a correct set of alternatives.

Very interesting to see is what a sum query does to the uncertainty of the resulting item. In the next example we have 1000 items with an individual uncertainty of 0.98. These items are combined using the following query:

```
SUM(
  SELECT kpu_transactiedata.stock_level
  FROM kpu_stamdata, kpu_transactiedata
  WHERE kpu_stamdata.matnr=kpu_transactiedata.matnr
  )
```

This results in the result shown in Table 6. `SUM()` will trigger the use of the sum function designed during this project, since the available sum-function in TriQL will not provide a result-set.

Table 2: All possible alternatives for the result of the example XDL. All uncertain attributes consists of 3 alternatives.

total*	confidence
601	2.23717005495E-7
600	4.47434010989E-07
600	6.62182248908E-05
599	1.51164724304E-09
599	2.23717005495E-07
601	2.23717005495E-07
601	3.31091124454E-05
600	6.62182248908E-05
600	0.0098
599	2.23717005495E-07
599	3.31091124454E-05
1202	1.48141429818E-07
1202	2.19242665385E-05
601	1.51164724304E-09
1200	4.38485330769E-05
1200	0.0064893860393
1198	1.48141429818E-07
1198	2.19242665385E-05
1202	2.19242665385E-05
1202	0.00324469301965
1200	0.0064893860393
1200	0.9604
1198	2.19242665385E-05
1198	0.00324469301965
1803	1.51164724304E-09
1803	2.23717005495E-07
1800	4.47434010989E-07
1800	6.62182248908E-05
1797	1.51164724304E-09
1797	2.23717005495E-07
1803	2.23717005495E-07
1803	3.31091124454E-05
1800	6.62182248908E-05
1800	0.0098
1797	2.23717005495E-07
1797	3.31091124454E-05

Table 3: Since the software known the integers are the same, it combines the results.

total*	confidence
1202	0.003288689694156818
1200	0.973422620611677
1803	3.355805810363304E-5
1800	0.00993288388379259
599	3.355805810363304E-5
601	3.355805810363304E-5
600	0.00993288388379259
1797	3.355805810363304E-5
1198	0.003288689694156818

Table 4: The set of resulting alternatives, when not all attributes are uncertaint.

total*	confidence
601	3.31091124454E-05
600	0.0098
599	3.31091124454E-05
1202	0.00324469301965
1200	0.9604
1198	0.00324469301965
1803	3.31091124454E-05
1800	0.0098
1797	3.31091124454E-05

Table 5: The set of alternatives, based on the fact that there might be no relation.

total*	confidence
-2	0.00657737938832
1200	0.973422620611
1	1.34232232415E-4
-600	0.0198657677676

Table 6: Result of the SUM query. Without alternatives

results	confidence
540359	1.682967357E-9

Table 7: A set of alternatives, with individual confidence scores. Based on a set of 100 items, with 3 alternatives and the example SUM query.

stock level	confidence
6263	1E-22
6270	2.86732036294E-06
6271	0.000140580592235
6268	4.17735650314E-10
6269	4.09472266023E-08
6266	1.5530746E-14
6267	3.0442903315E-12
6264	1.078E-19
6265	5.28231E-17
6280	4.17735650312E-10
6281	3.04429033148E-12
6282	1.5530746E-14
6283	5.28231E-17
6284	1.078E-19
6285	1E-22
6272	0.0045970782137
6273	0.0902995458185
6274	0.809919773371
6275	0.090299545818
6276	0.00459707821358
6277	0.000140580592239
6278	2.86732036294E-06
6279	4.09472266022E-08

If one uses alternatives, the built-in sum function in the overall system will determine and combine the alternatives to a new PDF, which could grow very fast in size with the number of data source items. To illustrate this a small example is given, this combines the alternatives of 100 items, each item has 3 alternatives.

```
SUM(
  SELECT kpu_transactiedata.stock_level
  FROM kpu_stamdata, kpu_transactiedata
  WHERE kpu_stamdata.matnr=kpu_transactiedata.matnr
    AND kpu_stamdata.matnr<101
  )
```

The result shown in Table 7 is a bit larger than earlier results. The implemented SUM function is relative naive and will run in exponential time with the number of tuples with alternatives.

The results in Table 5 are somewhat strange. Technically the system uses -1 to indicate a non existing relation. But the Trio system does not support

this trick. What we expected was a score like the attribute uncertainty, since it should behave like attribute uncertainty. We will skip the relation uncertainty until we find a better way to deal with this issue. Handling relations as uncertain attributes without alternatives is of course possible.

However, if we count the number of tuples in the results set that are incorrect according to the rule set, we find that only about 70% of the tuples is considered correct. This percentage drops even further after more data integration steps. This is much less than the determined 96%-98% as shown in Table 1. The more tuples are combined to create the result set, the lower the score gets, the actual percentage of wrong tuples in the result set grows as well. Creating decisions in which more data is combined, will provide a lower confidence score, but the scores found in validation are even lower.

On the other hand if we manually check a sample set of the result set, the percentage of correct ones is mostly close to 97%. However, this is based on what can be manually checked, and lacks consistency checks and accuracy check which are beyond our abilities. At the moment we take note from this, but do not take it into account just yet.

8.2.3 Discussion

First, if we consider the differences between actual incorrect tuples and predicted incorrect tuples, we must conclude that the calculation of the mean data quality score is not a correct way to go about this. It is now clear that for example rule number one could invalidate 50% of the data, while rule number two does also invalidate 50%. It is then possible that somewhere between 50% and 100% of the items are incorrect. So we need a new way to determine the total quality score that incorporates this knowledge.

8.3 A New Way to Determine the Uncertainty

After consideration of the conclusions sketched in Section 8.2 it is clear that a quality score that shows the percentage of correct items must be used as such. So the new way to determine the data quality of the data set is to determine that percentage.

Following the discussion in Section 8.2.3 we believe the confidence score is that percentage of the data-set which at a given moment in time is correct according to all the DQ rules. This should be resilient to the fact shown in the example in Section 8.2.3, where two rules provide a DQ score of 50%. This means that the number of actual wrong items in the data set is still unknown.

However, at the moment of the data quality check, the labelling of correct and incorrect items could be seen as “certain”. But the situation remains that the data quality score is a week old, and the actual score of a given items is not available during decision making, for both source data and result data.

Table 8: Result of the given query.

results	confidence
45041.04	0.5041

8.3.1 Methodology

The methodology to check the result of this research question is similar to the one sketched in Section 8.2.1. The only difference is the new confidence scores accompanying the input data set.

8.3.2 Measurements

The results of the measurements are of the same structure are shown in Section 8.2.2. However, the data quality score is now 71%, so the confidence score is 0.71. When simply combining data from two data sets this means the new tuples will have a confidence score of 0.50.

Both checks using data quality rules and manual checking of samples revealed that the results provide by the database are correlated to the results found during validation. However, it seems to behave like a worst case scenario. We assume that if items of different data-sets are related, correct items tend to be related to correct items. We believe the same holds for incorrect items, such that this indeed is a worst-case prediction. This means we could say that if tuples are randomly related the scores as determined by the system are completely equal to the results determined during validation. The mean of the differences measured in the manually checked sample set is 0.05 and the highest score determined afterwards was: 0.6. The expected result based on the quality rules, was about 52% percent.

To illustrate one of the results we have chosen the next query:

```
SELECT (kpu_june_9001_double.stprs *
  kpu_june_oms.stock_level) AS result
FROM kpu_june_9001_double, kpu_june_oms
WHERE kpu_june_9001_double.matnr='1'
  AND kpu_june_oms.matnr='1'
```

This query combines two tables in which the tuple uncertainty is defined as: 0.71. The result is shown in Table 8. If more tables would be combined, this percentage drops even more.

8.3.3 Discussion

In a perfect situation we see that the determined confidence score is equal the actual data quality of that result set. A perfect situation means that in the combination of data sets, the quality of items in one set is not related to the quality of the data in the other set.

We saw this in biased sample sets, where with bias we mean that the sample set was picked from a subset of the total result set. For example, we can choose a set of relative expensive articles. This set of expensive articles is maintained with relative greater care. Besides the articles, the data concerning these articles is also relatively better taken care of. This shows that generalising data quality scores provides a result that behaves like a worst case scenario. As far as we could see in these tests, this worst case scenario is a good prediction of the problems faced. Since these differences could still be a result of a set of biased samples. Also the automated checks showed the results are very close in the case of these data-sets.

Another issue is the fact that some rules, for example a rule which determines the uniqueness of an item, points to both items as part of the problem. So this way the system is told that both are incorrect. If we take this into account while modelling the uncertainty, this will result in a better prediction. However, it we leave it this way, we must remember this when validating the results concerned with these items. For the pure model it does not matter, but from a business perspective it is important to know how to use such results. We believe that for this system one of the duplicate items should be marked as correct and the others as incorrect, bur for cleansing actions one should take them all into account.

Furthermore, a confidence score of 0.71 means, in less precise words, that the business is using data of which at least one out of four items is incorrect. An organisation with such quality of decision input will not be in business very long. We contacted the KPU office and inquired if they could provide the percentage of mistakes made due to data quality issues. This is not a very simple question to answer. However, they showed that in their outgoing deliveries between 0 and 2 percent of the deliveries are returned in any given period. Even if the data in the systems used at the KPU, so the including systems which are not maintained by the CBG, are of perfect quality, the predicted result is very different. We are not able to perfectly check and relate the results at the decision level in this project. However, we will take some time to explore this difference and the closeness of this information received from the KPU to the original data quality score from Section 8.2. There is no relation between these values, but it was the first trigger to have a closer look.

8.4 The Influence of Detailed Data Quality Knowledge

The show the reader how to improve the precision of the predictions made by this model. This section will show if it is possible to improve the predictions by improving the quality and uncertainty information within the system. The user must find a balance between maintainable uncertainty knowledge and prediction precision.

In the last section it was deduced that it might be good to have more detailed data quality information and thus more detailed probabilistic information about the data set. To measure this we have created a simple data-set consisting of several tables, such that we could follow all calculations.

8.4.1 Hypothesis

Our hypothesis in this case is: **The more detailed the data quality information, the better the predictions get.**

8.4.2 Methodology

After creation of the data-set with the confidence scores, several queries to combine and filter the data were run. The lineage of the resulting data was used to verify what exactly had happened. What we hope to see is that more detailed data quality information, will force the system to create more detailed uncertainty information about the result set.

8.4.3 Measurements

The main thing that must be said about the measurement results is that they are perfectly in line with the probability calculations defined in the Trio model [33]. That said, the result set shows that now for subsets of the data the uncertainty is better defined, since we know the data quality of the subsets, or even of the individual items.

Results of an example decision are shown in Table 9. The set of queries in that decision are:

```
product_count :=
  SUM(
    SELECT *
    FROM test_data_products, test_data2_count
    WHERE test_data_products.id =
              test_data2_count.prod_id
    )

stock_level :=
  SELECT id, (prijs * aantal) AS total
  FROM product_count

result :=
  SELECT total
  FROM stock_level
  WHERE id IN (1; 2; 3; 4)
```

The confidence score of the data items related to each other, in the source set are 1, 0.98, 0.9 and 0.71, in the same order as given in Table 9.

8.4.4 Conclusion

More detailed information provides an end result which better fits the actual uncertainty of the result set. This means that the more knowledge put into the system, the better the predictions it provides. This situation, however, had no

Table 9: Result of the given set of queries.

total	confidence
2200	1
1200	0.92236816
600	0.6561
200	0.25411681

dependency along different input sets, meaning that one item's uncertainty is not related to the other item's uncertainty to which it is related according to the relational database. It was not possible to model the possible related uncertainty between data sets, in order to correct for the small differences noted in Section 8.3.2.

What we can conclude is that a perfect situation results in a very precise prediction, and a situation with more dependency involved, which we can't model, will only get a worst case prediction. A perfect situation in this case is a situation where no dependencies exist among the data objects at all. What actually happens is that with the current uncertainty information, we always model a perfect situation.

The best way would be determine the quality of only that data object used during decision, this is not possible as defined by the project's problem. If we could model every detail, the prediction would be better, however, this is not feasible. So as far as we can deduce here, this system provides the next best thing, a worst case prediction of the uncertainty of the data object used during a decision, given we can correctly model the data quality inline with the business needs of KPU.

More detailed data quality information will make the prediction even more precise. However, this could not be shown within this project, since that detailed information was not available. Additionally, the determination and maintenance of such detailed quality information is probably quite difficult for any given organisation. And such detail might not even be more supportive to the decision maker, than the current level of detail.

8.5 Lack of Business Performance Due to DQ

After the side step taken in the hypothesis in Section 8.4 we take a look at the big difference between uncertainty of the results found in Section 8.3 and the performance of the KPU department. Of course an organisations process has some robustness against some lack of data quality and as shown in Sections 8.2 & 7.3.2 not all of the time a decision using wrong data will be affected by the fact that the data is incorrect, it could still be that the correct decision is made. But the mentioned difference is very big. This section aims to predict the loss of business performance due to mistakes made by imperfect data, of course, business performance of processes which are clearly affected by the data and decisions modelled within the experiment.

8.5.1 Hypothesis

As explained the scores received form KPU show a worst case scenario of 98% correct actions. So we hypothesise that: **The lack of DQ results in an uncertainty at decision level in the range of** $[0, 0.02]$**.** Of course given this current case at the KPU.

8.5.2 Methodology

To get an understanding of the problems the KPU encounters during their work we took a detailed look at the set of items sent back due to mistakes. We hope to find a percentage of mistakes made due to data quality at some level.

8.5.3 Measurements

As stated we have received a list of 2274 distinct items, which were wrongly provide to personnel of the MOD and in total the number of mistakes made was 39998. This set of items covers deliveries in the span of one month. Some data items are multiple times in this list, because some people received the same, wrong item. Why the items were returned to KPU is not specified, but something was wrong, either a wrong address, a wrong item, wrong size or perhaps a wrong number of units.

If we take the unique NATO stock number for each returned item and compare this to the original data set and mark the items that do not confirm to at least one rule, 1137 of those 2274 items are marked as incorrect. The number of total sendings with data quality issues was 23304, of those 39998. This means that with half the returned items, there is a data quality issue related. This does not mean that the data quality issue was the reason for the performance problem, but the fact that so many items involved with a business performance issue are related to a data quality issue is very interesting for our system.

We must note that one rule with which 45 items of the almost 40000 fail, is that the NATO stock number is not correct or unique. But we believe this "noise" is acceptable.

8.5.4 Conclusion

If we define the range of the KPU issues we found $[0, 0.02]$, so if we assume the provide ratio in the last section is normal, the problems due to the lack of data quality would range from $[0, 0.01]$ and if we define the range of the predicted number of issues by the system, using tuple uncertainty, this is $[0, 0.04]$. Note we now assume the found 0.96 in Section 8.2.2 is a worst case scenario.

If we compare these ranges they could be related. However, the determined score used in Section 8.2 was shown to be incorrect. What we did find was the when we looked at the result set, not very much seemed incorrect, a confidence score of 0.50 seems to be incorrect. However, this score in Section 8.3.2 was validated using the criteria defined in the data quality measurement.

We think there might be a difference in the definition of "fit for use" between the data quality measurement tools and the actual business needs of the KPU.

8.6 "Fit for use"

Looking at the data quality rules which apply to the KPU data set, we see that all of them are very good to be used in the efforts to improve the quality of the data set. However, we have seen that the mentioned data quality score of 70% is very low. A score that low would have a very negative impact on business performance. This negative impact is not shown in the performance of KPU.

We believe the difference exists because of two facts. First of all, the data-set maintained at the CBG is not the only used data-set during operations. This data is augmented with the data system of the KPU which maintains orders and stock levels. However, the product definitions, think of shoe size, come from the set maintained at the CBG. This data could clearly be involved in wrong deliveries.

Furthermore, the rules do not all adhere to the notion that the data is fit for use, use by the data client that is. We try to show that if only a subset of the rules is used, which actually support the business processes that are used by the data client, in this case to send articles to personnel. The data quality score and thus the data uncertainty will better predict the business performance loss based on data quality.

8.6.1 Methodology

First of all, this way of removing rules has the risk of creating a set of rules which results in the wanted prediction. We avoided changing the set according to result, but we have determined three different subsets.

In creating the first subset, we were very strict in keeping rules in the subset and we started with the rules set which were found Section 8.5. These rules were found by looking at the mistakes made, the data which was involved and then which rules marked that data as incorrect. As stated back then, that does not mean that that rule-issue combination influences the performance. Rules like: "No empty English translation" were removed, since in this case they have no influence at all.

The second set was also created by starting with the total rules-set and their detailed description defined by the CBG. In this set we removed some more rules which, according to common sense, cannot not influence the performance. And the final set was creating by removing rules which should not influence the performance of the KPU.

We must note that in order for the overall system to provide answers on which every business can depend, more detailed research should be done in which every detail is followed from data to result. However, this does not fit in the scope and time of this project, we only show the correlation between prediction and what actually happened.

8.6.2 Measurements

The size of the total data-set maintained at the CBG consists of about $60,000$ data-items. The first rule set showed that 2814 items are not correct. This results in a DQ score of 95.4%. The second set showed that 1168 items are wrong, which relates to a DQ score of 98%. And finally the third set indicates a DQ score 99.1%, based on 521 incorrect items. If we use these scores in our overall system we get results like the one shown in Section 8.2 and the final prediction is close to what actually happened.

8.6.3 Discussion

This way of predicting the mistakes made is very close to that experienced by the KPU. Still there are some factors which are of unclear influence, it remains however a prediction based on the available knowledge and the resulting confidence in the correct value is based on a data quality score based on the entire data set. But these results look promising in favour of the overall system.

8.7 Equal Uncertainty.

All measurements, except those in Section 8.2 , were done using tuple uncertainty, which was based on the data quality score. This way the Trio system has the uncertainty per data object and within the data object the attribute uncertainty is dependent on the tuple uncertainty, such that if the tuple is correct, the attributes are too.

It might be that the differences between types of uncertainty modelling, as defined in Section 7.3 are not relevant for the prediction. Of course the result can be very different, but this should be the result of incorrect use of the model. The reason to choose one should be how well it fits the business situation and the business needs. Most of the time data objects are combined into new data objects, both tuple and attribute uncertainty work perfectly in that situation. The different ways of using attribute uncertainty are options a user can pick from, to get some more insight into the predictions, however, these are all equivalent by definition.

Attribute uncertainty based on the rules and thus the data quality score, will give the same uncertainty score over the tuples as tuple uncertainty, if we define the data quality in the same way we did earlier. The rules are concerned with a subset of the tuple's attributes. In the following section we reason about how the different types of uncertainty work and what the influence on the result would be.

8.7.1 Deduction

For example, if there is only one rule concerned with one attribute and this results in a 90% score. The other attributes score 100%, since they are not found to be incorrect. This results in a tuple uncertainty score of 90%, since $0.9 \cdot (1)^n = 0.9$. If we use this rule as we did earlier the number of items that

are considered wrong is also 90% and the tuple uncertainty score would be 0.9 as well.

In this case, however, if the complete tuple is used within the decision than both ways provide the correct score, but if only some attributes are requested the uncertainty defined over those attributes is more precise. Otherwise, when using tuple uncertainty and requesting one attribute which was assumed to be correct, this will show a 90% uncertainty score, while this score is not related to this particular attribute. It is possible to use tuple uncertainty and define which of the attributes are certain and uncertain. In this case the result will be again correct.

If two attributes are uncertain and each of them gets a 0.9 confidence score, this means the tuple confidence derived from the defined attribute confidence will be $0.9 \cdot 0.9 = 0.81$. This actually defines the certainty of that tuple, which is the probability that that tuple is completely correct, as far as can be determined. The derived confidence might not be correct, since it is possible that these attributes are correlated. This way using only tuple uncertainty will be best for the business situation.

8.7.2 Discussion

This explanation shows why a user of this tool can choose between tuple uncertainty and attribute uncertainty. It also shows how to do this. If the data quality metrics are used to determine the data uncertainty, in the way it was shown in the earlier hypothesis and research questions, the use of tuple or attribute uncertainty is equal, given that the chosen one is correctly used. Due to the limited support for correlations, with attributes that show dependency, some loss of information in the overall system will occur, but a more abstract scoring will be resilient against this.

9 Discussion & Conclusion

This work aims to provide answers to the questions posed in the section on the research objective and supporting questions within Section 3. In those sections it was explained that an organisation like the MOD uses data quality techniques to search for problems concerning the data, improve the quality of the data and to determine the general state of the data.

The real problem in layman's terms is that information on the quality of the data is not presented to people using the data during decisions. Besides the fact that information is just not presented, the quality of the original data is different than that of the combined data set used at a given moment. An organisation could present the quality score along with a data item, but what happens to that score when combining the data?

Measuring the quality of the newly created data is an option, but not very practical, since that data is temporary and the measurement tools need relatively much time to determine the score. It is also not always possible to determine the quality of some data items.

This thesis could be divided into two parts, the first part containing Sections 4, 5 and 6, which explains the context and the research fields of this work. The second part shows the design project of the overall system and the research project done to find the answer to the questions. First, we will discussion the questions answered in the first part of this thesis.

1. What is known about the data of the MOD currently?
2. What research fields are interesting to understand when setting out to solve the main question?
3. How to define the relation between Data Quality and Data Uncertainty?
4. How to define the relation between Data Uncertainty and Probabilistic Databases?
5. Determine the influence of Data Quality on Decisions?

In short this was the problem sketched at the MOD and is more detailed explained in Section 3. On a higher level two goals were described in Section 1. These explain the knowledge this work hopes to provide the MOD as well as creating a place to start for future work. We believe this work shows the MOD the potential gain of using a system like the one explained to determine the problems at an operational level, using the data quality of the master data set. Given the nature of the designed system, it is possible to trace back the root cause, within the data quality, of the business performance problems.

As can be deduced from the paragraphs above, we believe it is possible to model the data quality using data uncertainty techniques. In such a way that the quality information does not get lost. This work shows a relation between the predicted uncertainty and the real issues during decisions and operations. It can be concluded that, it is possible to design a generic system to model

uncertainty of data and decisions made with that data, in such a way it predicts the uncertainty during decision-making, where the uncertainty is determined using measurements from the data quality research field.

The field of data uncertainty was chosen since the widely used model in this field, the PDB, looked very promising to support our first vision on how to solve the problems. A PDB is able to store data and a score related to that data. It also can combine the scores, during query evaluation in order to provide a score for the newly created data.

During this research it is shown that data quality and data uncertainty are not the same. Data quality shows how well the currently represents the real world, data quality is also related to the use of the data. Data quality efforts must, by definition, support the business needs of the organisation using that data.

In contrast to data quality, data uncertainty shows to what extend a data item is correct or certain. Note the similar words used, however, the difference is that with data uncertainty one knows the probability that a given data item is correct, but wether it is actually correct or not is unknown. Data quality aims to show which items are correct and which aren't. Of course, within both definitions there is room for overlap. Furthermore, data uncertainty is by definition not concerned with the needs of the data client.

Next, the questions that are answered using Sections 7 & 8 are shown. After this list, these question will be discussed and concluded, the explanation of the questions is found in Section 3.2.1.

1. How to model the data quality scores of MOD, using Data Uncertainty?
2. Does more detailed data quality knowledge improve the prediction?
3. Data Quality. An end in itself or a level of "fit for us"?
4. What are input restrictions and requirements?
5. A language to model the decision
6. How to support Data Integration steps of an organisation?
7. Must complex data operations be supported?
8. Which probabilistic database can support this overall system?
9. How to support the decision maker, with this system?

Once data quality is determined, things still happen to that data that change the data quality. One must think of cleansing, maintenance, updates of data values etc. Another important use of data is to derive information by combining data. All these steps have in common that they create new data, but without an indication of the quality or certainty. This work showed that by using data quality and data uncertainty techniques the certainty of the resulting data can be determined.

How well this determination can be done depends on how detailed the data quality information is and how well the data integration steps are defined. But with the case of the MOD a close worst case prediction could be established. In other words, this work showed that by defining data quality as the percentage of correct items in a data set, based on rules which determine the "fitness for use" of that data. The found score is the uncertainty score of the items in that data set. After data integration steps, the confidence one may put in a new data, or information, item is a new score. At decision level this information can provide probability of making a mistake, which is invaluable for determining the cost of the data quality problem.

Within this research we found that if one adheres to the made definition of data quality and data uncertainty as in Section 4, one can model the data quality using data uncertainty methods. This must be done such that the data quality score defines the probability that the item is correct, as far as can be determined. This way the system contains the best data uncertainty information, which represents the data quality. Continuing on this notion, more detailed data quality information, will help to improve the precision of the overall system.

Within the field of data quality it is clearly defined that data quality measurements should be done from the data user perspective. This means that quality is actually defined by the user of the data. However, the user group does not consist of only the end user, or decision maker in a given process. A user could also be the person responsible for the database systems. But for this concept to work, the subset of the data quality measurements done, must be geared towards the decision makers needs. The measurement outcomes must reflect the quality of the data used and be guided by how the data is used.

The overall system, designed and used throughout this project, supports the needs of this project. It is not a ready product for a generic organisation. The system is shown to be able to model the decision properly and predict the correct uncertainty at the decision level. This way the system provides the decision maker with the data available and the knowledge how "well" that data is.

We can conclude that it is possible to show the quality of the information used at decision level and predict the business performance impact to some extend. Furthermore, the prediction of the costs is a very interesting attribute the show what the costs of the data quality problems are. These calculations are based on solid operation from the field of probability and statistics but deserve more research.

Finally, with the knowledge gained in this research we can now create a new definition of data quality. This definition is from the perspective of business performance:

Definition 9. *Data Quality is an indicator of the business performance, given that the data is used to make decisions that guide the business processes.*

10 Future Research

This section explains some interesting questions, which could not be answered within the scope of this project. But in the future these questions should help improve the system, the model or the way to determine different scores and settings used within models like the one described in this work.

From a business perspective it is very interesting to determine the expected value of the loss or gain of a given decision more precisely. If that is better predicted, the determination of the data quality improvement budget can be more precise. A step further would be to design a method to determine this budget in the case of a sequence of decisions. We believe that a useful system can be designed which uses the principle of Markov-chains to determine the possible loss or gain of a given sequence of possible events. Markov-chains can be used to create a joint probability distribution using random walks over the directed graph depicting possible next states.

Trio does not completely support the needs the system has, but it is sufficient in this stage of the system. However, in the future it might be beneficial to have completely tailored PDB which serves all the needs of the overall system.

This project showed a correlation between the determined data uncertainty and the business performance. However, it would be very interesting to set up a project which can determine the actual causal relation in a given situation. This is probably highly dependent on the actual organisational situation. If a generic case can be made for this, that would be invaluable for the acceptation of systems like this one in organisations.

An important next step would be to determine how this system behaves in a completely different environment. We believe the results seen here are not influenced by the organisational environment, but it would be helpful in tuning the system. Additionally, experiments with other data-sets and more complex data integration steps would help to understand how the system behaves in other environments.

What would make a system like this more useful is not to maintain the data itself, but to only maintain the uncertainty of the data. The data itself is maintained elsewhere. In other words, just point to the original data and store the uncertainty of that data. This helps this system in becoming a tool to add to a set of data management tools already in place.

In addition to making the system work with external databases, it is important it can work with more data types and not convert everything to default data types. However, the most interesting data types are probably character strings and numbers.

Earlier in the process of this research the question was formulated, if it might be necessary to extend SQL to support all possible data retrieval actions performed within an organisation. It was explained not to be a very interesting endeavour. However, it might be very interesting to look at the possibilities to use the data integration defined using RDF Gears[2] to determine the uncertainty of the end result. This way this model could augment a tool like RDF Gears and this could greatly improve the use of this model within organisations.

[2] http://wis.ewi.tudelft.nl/imreal/u-sem/rdfGears

11 Bibliography

[1] L. Antova and T. Jansen. Fast and simple relational processing of uncertain data. *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE*, pages 983–992, 2008.

[2] L. Antova, C. Koch, and D. Olteanu. World-set decompositions: Expressiveness and efficient algorithms. *Database Theory-ICDT 2007*, 403:265–284, 2006.

[3] L. Antova, C. Koch, and D. Olteanu. $ ${10^{(10^{6})}} $ $ worlds and beyond: efficient representation and processing of incomplete information. *The VLDB Journal*, 18(5):1021–1040, 2009.

[4] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, 41(3):1–52, July 2009.

[5] C. Batini and M. Scannapieco. *Data quality: concepts, methodologies and techniques*. Springer, 2006.

[6] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom. An Introduction to ULDBs and the Trio System. *IEEE Data Engineering Bulletin*, pages 1–12, 2006.

[7] J. Brennan and J. J. Elam. Understanding and validating results in model-based decision support systems. *Decision Support Systems*, 2(1):49–54, Mar. 1986.

[8] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. *... of the 13th International Conference on Very ...*, pages 71 – 81, 1987.

[9] R. Cheng and S. Singh. U-DBMS : A Database System for Managing Constantly-Evolving Data. pages 1271–1274.

[10] E. F. Codd. A relational model of data for large shared data banks. 1970. *M.D. computing : computers in medical practice*, 15(3):162–6, 1970.

[11] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt (extended version). *ACM*, 2008.

[12] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, June 2006.

[13] A. de Keijzer and M. van Keulen. Quality measures in uncertain data management. *Scalable Uncertainty Management*, pages 104–115, 2007.

[14] F. Dekking, C. Kraaikamp, H. Lopuhaä, and L. Meester. *A Modern Introduction to Probability and Statistics*. Springer, 1 edition, 2005.

[15] R. Elmasri and S. Navathe. *Fundamentals of database systems*. Addison-Wesley, 6 edition, 2010.

[16] J. R. Evans. *Statistics, Aata Analysis & Decision Modeling*. Pearson Prentice Hall, 3th edition, 2007.

[17] N. Hubig, A. Züfle, and T. Emrich. Continuous probabilistic sum queries in wireless sensor networks with ranges. *Scientific and Statistical . . .*, pages 96–105, 2012.

[18] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, 10, 1966.

[19] C. Re, N. Dalvi, and D. Suciu. Efficient Top-k Query Evaluation on Probabilistic Data. *2007 IEEE 23rd International Conference on Data Engineering*, pages 886–895, 2007.

[20] C. Re and D. Suciu. Materialized Views in Probabilistic Databases For Information Exchange and Query Optimization. *VLDB*, (September):23–28, 2007.

[21] A. Sarma and O. Benjelloun. Working models for uncertain data. *ICDE Data Engineering*, 2006.

[22] S. Sarsfield. The Data Governance Imperative. *IT Governance Limited*, 2009.

[23] P. Sen, A. Deshpande, and L. Getoor. PrDB : managing and exploiting rich correlations in probabilistic databases. pages 1065–1090, 2009.

[24] G. Shafer. Dempster-Shafer Theory. *Encyclopedia of artificial intelligence*, 76:330–331, 1992.

[25] M. a. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top-k Query Processing in Uncertain Databases. *2007 IEEE 23rd International Conference on Data Engineering*, pages 896–905, 2007.

[26] D. Strong, Y. Lee, and R. Wang. Data Quality in Context. *Communications of the ACM*, 40(5):103–110, 1997.

[27] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool Publishers, synthesis edition, 2011.

[28] W. Walker and P. Harremoës. Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support. *Integrated Assessment*, 4(1):5–17, 2003.

[29] D. Wang. Extracting and Querying Probabilistic Information in BayesStore. *http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-131.pdf*, 2011.

[30] D. Wang and E. Michelakis. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *Proceedings of the VLDB Endowment*, 1(1):340–351, 2008.

[31] X. Wang, K. Govindan, and P. Mohapatra. Collusion-resilient quality of information evaluation based on information provenance. *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 395–403, June 2011.

[32] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. *Technical Report*, 2004.

[33] J. Widom. Trio A System for Data Uncertainty and Lineage. *Managing and Mining Uncertain Data*, pages 113–148, 2008.

[34] M. Zijdemans. Literature Survey: Modelling Uncertainty in Data. Technical report, Delft University of Technology, 2013.

12 Appendix A. Details of the System

So far the system was explained in text, in this appendix the important parts of the software implementation are explained in a bit more detail. As explained the overall system is geared towards a generic solution to data quality at a decision level, based on data quality at the source.

Figure 6 shows a generic situation where a decision is based on a report. The report is created using data available in the database systems. Both this situation and the problem described by the MOD let us to design and start building a generic overall system.

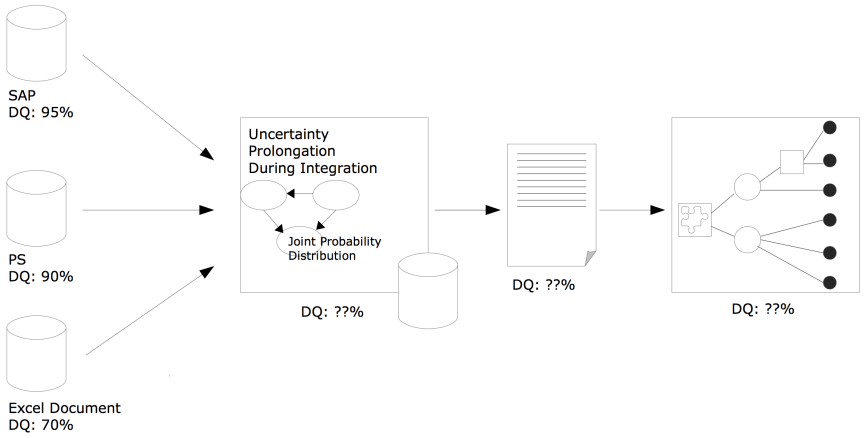

Figure 6: A sketch of the "context"

The sketch shown in Figure 6 contains the source databases on the left. SAP is the name of the company which delivered some of the information systems to the MOD. Because of this, SAP is the given name of the database systems within the new information system environment of the MOD. Which software is actually used for critical of confidential systems is unknown to us. Other data sources are locally maintained data systems which contain data which is not permanent of nature, for example the PS in Figure 6. Also some data still resides in uncurrated sources, like a spreadsheet document. Finally, there are the legacy systems, with which we are not concerned.

In short the designed system should be able to maintain data, assign uncertainty scores to that data and "parse" a decision in order to present the user with the answer and additional uncertainty information. Furthermore, the measurements needed to be done using most of the modules, in order to proof the concept of the system and saving time in not having to write two software products.

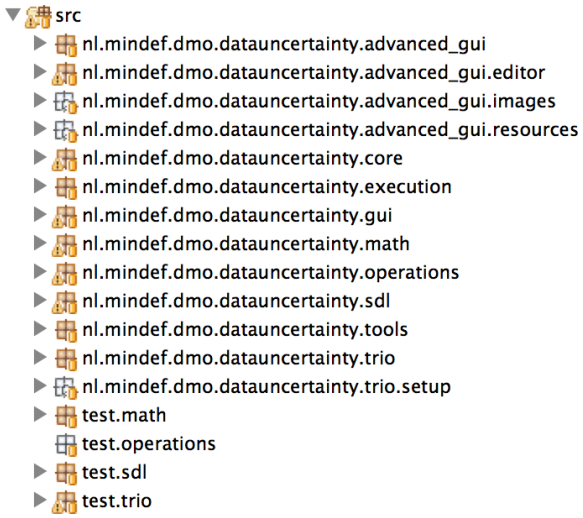

Figure 7: The package structure

First of all, Figure 7 shows the package structure of the Java project in Eclipse. The software is written in Java since this was the safest option concerning expertise available and unknown future options.

The system design started with the `nl.mindef.dmo.datauncertainty.core` package, this package contains the import functionality to import complete data-sets. As explained this needs to be improved to only import needed data, since it takes a lot of time to import large data-sets, of which only a few items might be needed. Furthermore, this `importDatabase` class is used both by the measurement runs as the later described graphical interface.

This `nl.mindef.dmo.datauncertainty.core` further contains the start up class for the measurements and several settings classes to point the software to the databases and for setting up the measurement environment.

The next interesting thing in the package structure is the relative small set of test packages. Every function that returns a value which can be used in unit testing and which does not heavily relies on other functions is tested using unit testing. Other functions were tested individually, since they mostly change the state of the current database. But looking at the package list, the "gui" is not yet tested automatically, the "core" package contains few testable code and the "execution" package is completely dependent on the "operations" and "sdl" packages.

Continuing top-down in the package structure as displayed in Figure 7, the next package is `nl.mindef.dmo.datauncertainty.execution`. This package only contains the `Executor` class, this class takes a Decision object as input and orchestrates the needed operations to determine the end result, it also creates the textual result to be shown in the command line clients.

This Executor class runs all the needed data integration steps and determines which type of probability distribution the end result should have. This executor also calculates the probability of the false-positives or the false-negatives.

The `nl.mindef.dmo.datauncertainty.gui` package, as well as the `nl.mindef.dmo.datauncertainty.advanced_gui` package will be discussed later. So the next package is `nl.mindef.dmo.datauncertainty.math`, this package contains the different probability distributions that are supported within the system and it contains a helper class "Calc", which contains some needed mathematical functions.

The operations which were not provided by the Trio system, but were still needed are implemented within this project and are placed in the `nl.mindef.dmo.datauncertainty.operations` package. In the section on the overall system it was already explained that these functions are: sum, product, minus, mean and median. All these operation follow the standard rules laid out in the theory of probability and statistics.

The largest package is the `nl.mindef.dmo.datauncertainty.sdl` package, this contains all the classes which are needed to create runtime counterparts of the elements specified in an XDL document. The XDL document is extensively explained in Section 7.4, so that will not be repeated here. The objects created to represent the elements within the XDL document are placed in a tree like relation.

Figure 8 shows the internal structure of this package. It shows the Decision class as the main class in this package, which contains a Question class. As explained earlier this Question class contains just a human readable String that represents the decision being made. Furthermore, this Decision class contains the options, represented by the Option class. Next the Decision class contains the trigger objects which are the representations of values the data should adhere to before an option is presented. This Trigger class contains zero or more Triple classes. These Triples are the combination of trigger value, real value and operation. This operation defines the relation between value and trigger value. For example, the operation could define a greater than relation.

Finally, the Decision class contains the DataIntegration class, a set of sorted information items and result sets. These Result_Set and InformationItem classes contain the data operation that need to be done the eventually create the final piece of information needed on which the decision is based.

An XMLParser class, also in the `nl.mindef.dmo.datauncertainty.sdl` package, creates the needed object as defined in the XDL document. These objects themselves parse the content of the elements as provided to them during initialisation. This way the XMLParser just parses the XDL document and the respective classes take care of parsing their content.

Finally, there are the two different GUI packages and the `nl.mindef.dmo.datauncertainty.tools` package contains no classes of interest at this moment. The first GUI was a very simple interface in order to get the system to be able to have a graphical user interface and to be directed by this interface. After understanding and implementing all the needed functions a more advanced user interface project was started. However, this took a lot of time and the focus of this project was on the measurements, so the advanced interface is currently not yet completed. The simple GUI also misses some later created functions.

A snapshot of the software can be requested via the e-mail address of the candidate, shown on the first pages of this document.

The Trio system as used within this project can be downloaded from the original site. In this version some very small changes are made in the TrioPlus code, to make the system work in the project environment, these changes are available in the snapshot.

The rest of the used products are Ubuntu Linux, Postgre-SQL, Eclipse and of course the needed Python and Java Libraries. The most recent, at time of writing, stable versions of these are used.

13 Appendix B. Lessons Learned

To close this project a reflection on the problems faced throughout the period is of great importance. This is why this thesis ends with a section on "Lessons Learned". Not all of these lessons learned are the results of a negative experience, almost all of these are methods used during this project and happened to work quite well in this case.

The first lesson was how important it is to manage expectations, it is better to tell something twice than understanding a few weeks later that both parties misunderstood each other.

In addition to the expectation management, it is important to exchange some form of formal documents stating the decisions made in the project, this leaves room to assume that those points are "fixed" and agreed upon.

The next issue which arose several times, was the fact that when explaining some part of a large project, it is always a good idea to first bring your audience into focus. This way they know exactly what you mean when explaining the core topic of your current talk. If this is omitted, your audience might be trying to understand the context and in the mean time they do not listen carefully to the current story.

Most important, and of course known by many, is to first carefully search for existing library and documentation. Chances are you are not the first one facing a given problem. Of course doing everything your self gives a great feeling of control, however, the extra time and effort are not worth it.

Another important decision is to first do the next task on your list, even when there is still time until the deadline of the current task. Wait with adding new tasks. Chances are a later deadline might be planned to soon. So only when you are absolutely sure you have time to add additional work, you can do this. When failing to meet a deadline, one must act immediately to remedy the problem.

Additionally, to your own deadlines, create deadlines for others on which you depend. However, it might not be wise to tell a superior officer that he or she has a deadline, this might be understood wrongly. But, one must create contingency plans when others fail to meet your project plan or quality restrictions. This way your work can always move forward.

It is better to have to do re-work, than to plan for everything. When planning for every possible future decision, the project will never actually start. So it is probably better to start anyway and later re-do some of the work that is found to be wrong or to remove work that is found to be unnecessary.

An issue encountered by this project was that the formal thesis was written at the end. But it might be better to write chapters at the moment of the research concerned with that topic. The upside of the taken approach is that the overall thesis has a more natural flow, but the amount of time needed to write such a thesis is a bit to much to do in one long period at once, this could make people trying to rush the work. Luckily in this case every step and decision was documented in great detail in notes made throughout the project, but it takes a lot of time reading all those, with the risk of missing some of them.

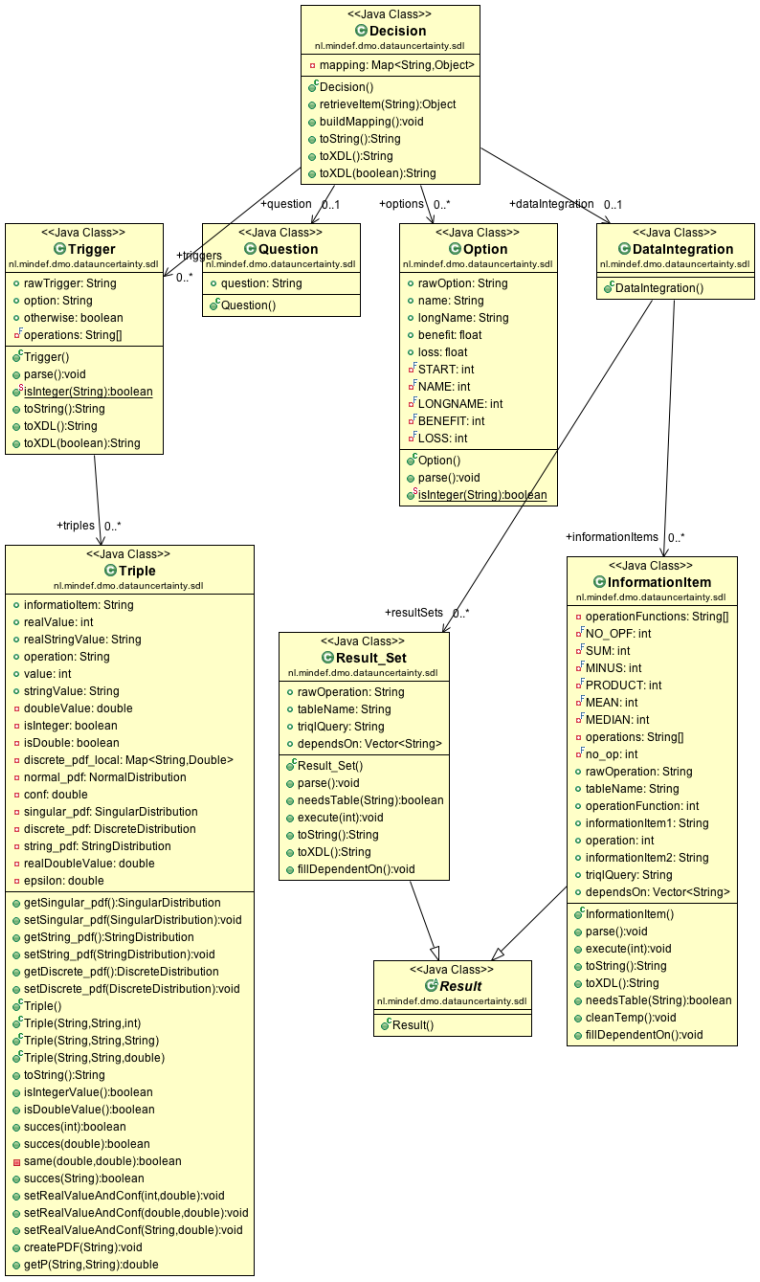

Figure 8: Class diagram of the SDL package