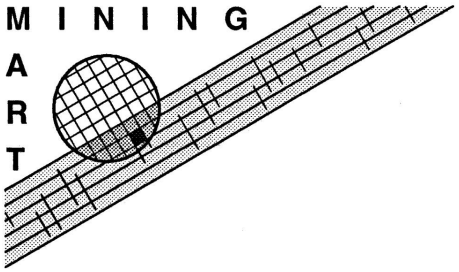


Enabling End-User Datawarehouse Mining
Contract No. IST-1999-11993
Deliverable No. D19

Problem Modeling Deliverable D19

Ronnie Bathoorn[1,2], Nico Brandt[1], Marc de Haas[1], and Olaf Rem[1]

[1] Perot Systems
Dept. Knowledge Engineering
NL-3821 AE, Netherlands
{Nico.Brandt, Marc.DeHaas, Olaf.Rem}@ps.net

[2] University of Groningen
Dept. of Computer Science
D-44221 Groningen, Netherlands
{r.w.bathoorn}@wing.rug.nl

June 21, 2001

Contents

Abstract

Domain understanding and description of activities are still a challenge in the context of the data mining process. This report presents a language that can be used to identify, organise and model the knowledge about the data and the data mining process and supports its reuse for similar situations.

Chapter 1

Introduction

A data-mining project starts with understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition. This task requires in depth knowledge of the business domain. A graphical representation of the domain enables a domain expert and a data-mining expert to get a better overview and gives her/him a basis for communication. This work package is named *problem modeling* because that is the essence of what is done while modeling the domain in which we want to find hidden patterns and relations. This report describes a language that can be used to define domain knowledge and presents, as example, a high level case that can be reused as design pattern for similar learning problem[1]. It is based on OMG's[2] *Unified Modeling Language* (UML). The UML represents a collection of best practices that have proven successful in the modeling of complex systems. This workpackage introduces extensions on UML that meet the specific needs for specification and visualisation of the data mining process.

1.1 Relation to other work packages

This work package builds further on the results of workpackage 5, in which classes of domain knowledge have been defined. Results of this workpackage will be used in the workpackages 12 and 16 in which libraries for a graphical user interface are build and in workpackage 10 for specifying prior data mining projects.

[1] A case base of prior data mining projects will be produced in WP10

[2] The Object Management Group (OMG) is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Their best-known specifications include *CORBA*, *OMG IDL*, *IIOP*, *UML*, *the MOF*, *CWM* and *Domain Facilities* in industries such as healthcare, manufacturing. Fore more information visit their website at http://www.omg.org/

Chapter 2

Domain Visualisation

Visualisation is about creating order in information chaos. By creating visualisations of tables, relations and dependencies between element in your data you get a much better view of the problem. With a little help from the GUI you can also build a data-mining set-up without having to know everything there is to know about data mining and databases. If you have a graphical representation in which it is possible to visualise your database model and any domain knowledge it would be possible to create a program which domain experts can use to add their domain knowledge to a picture of a database model. Then the data mining expert can use this domain knowledge in the data mining process right from the data model instead of having to handcode it into lots of configuration files for different mining engines.

We use the UML with additional extensions for specifying domain knowledge. The UML has grown into the de facto visual language for specifying, constructing and documenting the artefacts of information systems and is becoming more and more popular for modeling business processes. For reasons of acceptance it is important to comply with these standards. Furthermore, the object oriented methodology provides very useful features that enable compact and readable specifications.

A data mining process is more than domain analysis. In the end we will also need to execute pre-processing actions and run learning algorithms by using databases and programs. Programs and databases still require other languages. We need glue between these languages, a glue that makes a straightforward transformation from specification from domain analysis to implementation/execution possible[1]. For the time being we can take the very pragmatic approach by allowing the user to only use graphical representations that are high level or that have a direct translation to a

[1]At this point we have to notice that the UML has no formal semantics. This is a potential danger, since we don't have a sound axiomatization. This is a challenge that provides a lot of opportunities for further investigation (see [4]), but that is not in the scope of this workpackage or even this project.

fixed set of pre-defined operators, like those that are described in WP8 and WP9.

2.1 Graphical Representation

In the context of the data mining process the UML class diagrams can be used to give a static representation of the concepts we want to use in the learning process and the relation between these concepts. The standard visual UML elements class, association, aggregation, composition, inheritance and dependency will be used with their standard interpretation[2]. We introduce new stereotypes *<<base>>*, *<<aggregate>>*, *<<constructs>>* and *<<summarise>>* for data mining specific constructions. We also add some visual elements for expressing aggregation functions like *sum*, *maximum*, *minimum*, *average* and *time frame* on properties.

For modeling dynamic elements of the data mining process, like sequences of operations and iterations, we use UML sequence diagrams. For a better connection with the data mining process we introduce new visual elements for *data mining goal*, *analysis paradigm* and *domain model*. These sequence diagrams can represent very detailed sequences of operations, but could also be more high level. These high level diagrams can be seen as a case description and can be reused for a similar data mining goal.

In the next sections we will discuss the visual elements and their relation to the domain knowledge elements from WP5.

2.2 Data mining extensions for UML class diagrams

- Class and *<<base>>*
  For the visualisation you need *concepts* which are collections of data that represent a concept in the domain of your data. These concepts can be in different levels of abstraction with at the base tables in a relational-database or objects in an OO-database. Concepts are visualised as an UML class with a name and a couple of attributes that represent the data that is contained in the concept. A concept that is a direct representation of a stored table or object will have the stereotype *<<base>>*. An example is shown in figure 2.1. In this way a clear distinction can be made between concepts that do have a direct representation and concepts that do not. From these base concepts we can build higher level concepts or, alternatively, we can map a high level concept to one or more base concepts. Figure 2.2 shows

[2]At http://www.omg.org/technology/uml/index.htm the UML specification can be downloaded.

an example of a data model of molecules with the concepts Atom, Model and Bond. Furthermore it shows how the concept OH-Group can be constructed from this data model with the help of *dependency* relations and object diagrams. The objects are elements from the concepts Atom and Bond that, in a given structure, form the concept OH-Group.

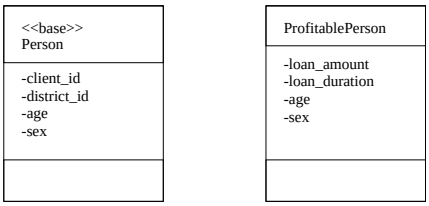


Figure 2.1: Concepts

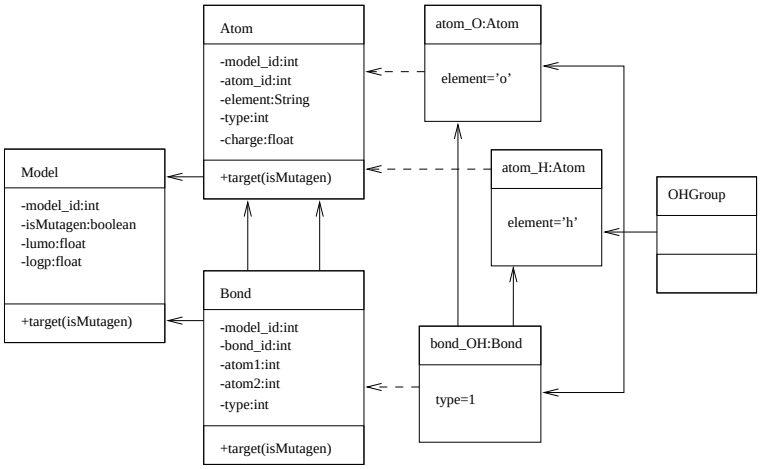


Figure 2.2: Structural model of an OH group

- Association
  Between our concepts we can have relations. Visually a relation is represented by an association (see figure 2.3). These associations have a multiplicity for both of the concepts involved and can have a reading direction. This reading direction is necessary in the data mining process to determine which data is accessible from where. An association between two concepts means that the data in those two concepts is somehow related.

- Aggregation & Composition
  Aggregation and composition are two special kinds of relations. If

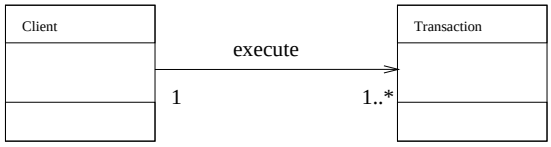


Figure 2.3: Association

there is an aggregation relation between concepts A & B it means that one instance of concept A can be related to a number of instances of concept B. In a *composition relation* concept A is build up of elements from concept B. This means that removing an element from A also removes all related elements. Aggregation is visualised by a diamond shaped box at the end of a relation and composition with a diamond shaped box filled with black (see figure 2.4). Notice that we also want to be able to specify a reading direction for these relations.

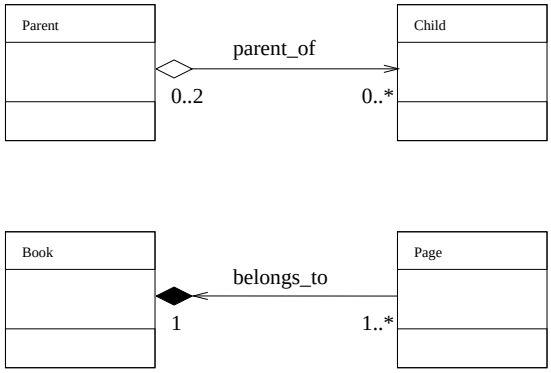


Figure 2.4: Aggregation (top) and composition (bottom)

- <<*summarise*>> and <<*aggregate*>>
  Summarisation is an important method in the data mining process where you have a concept that represents a summary of of groups in your data. For this summarisation you can use operations like *average*, *minimum*, *maximum*, *sum* and *predominant*. These operations are represented by small boxes behind the attributes (see figure 2.5). The attribute over which you are grouping the data gets a diamond behind it. The summarisation is represented by a dependency arrow with the stereotype <<*summarise*>>. The class representing the summarised concept has the stereotype <<*aggregate*>>. Notice that a concept with stereotype <<*aggregate*>> can have <<*summarise*>> dependencies with more than one concept.

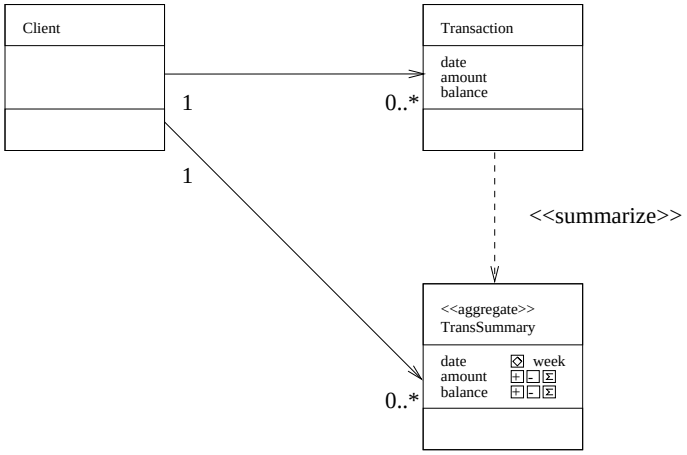


Figure 2.5: Summarisation

- *<<construct>>*
  New features can be constructed by taking data from different concepts and use it to compute a new attribute in a concept. The new feature is visualised by a circle attached to the concept with dependencies to all the tables that contain data that was used to compute it (see figure 2.6).

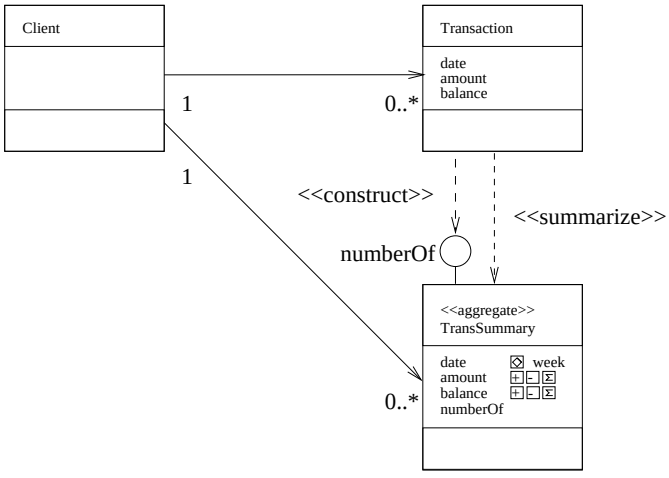


Figure 2.6: Feature construction

- Inheritance
  Concepts can be split up in a couple of sub concepts using the UML inheritance representation. In this way, for example, Clients could be divided in *Profitable Clients* and *Nonprofitable Clients* (see figure 2.7). This classification can be based on one or more attributes, but it could just as well be based on a complex model. For example the classification of Profitable Clients could be based on a model that has been build to compute the profit the bank has from a client based on a learning operator, like a decision tree or a neural network. In the case of classification based on the client having a credit card or not it could just be the attribute *hasCreditCard* in the base concept Client being 'Y' if the client has a credit card.

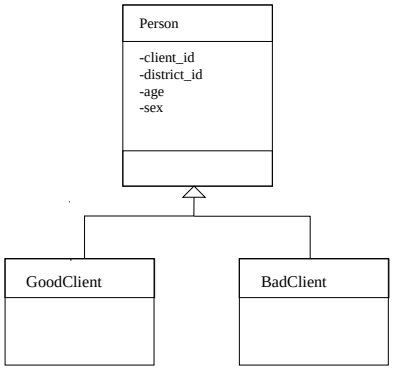


Figure 2.7: Inheritance

2.3 Data mining extensions for UML interaction diagrams

We use an interaction diagram to visualise the dynamic part of the data-mining process. In this diagram we show the sequence of the actions that need to take place to reach the mining goal. For this interaction diagram we define the following additional elements:

- Data mining Goal
  This element specifies what you want the data-mining environment to discover. This could be a function that predicts a numerical value or a decision tree for a nominal value. It also contains some specification as to when this answer is satisfying.

- Analysis Paradigm
  The class that the data-mining task belongs to, for example classification, regression or clustering is specified by this element. This

narrows down the number of algorithms the data mining environment could choose from to find its solution.

- Domain Model
  This element represents the whole class diagram in which information about the structure of the concepts is described and where the link between these entities in your model and their real world equivalents is specified.

data mining goal analysis paradigm domain model

Figure 2.8: Interaction Diagram extensions

With these extensions we can now build interaction diagrams that represent a data mining process for a certain mining task as seen in figure 2.9. In this interaction diagram we can make a coupling between concepts in our domain model and the tasks which should be performed on them. Thus creating a recipe for a data-mining problem. This interaction diagram can be used as a design pattern for a data mining problem. That is, we take an existing interaction diagram and map the concepts used in it to an other database model. In this way we get a new mapping between the data and the concepts in the sequence diagram.

A data mining process can be split up into multiple diagrams with different levels of detail. There can be a high level description of our data mining process showing only the high level actions, for example, a diagram as in figure 2.10 comparable to the data mining process described in the CRISP-DM model. A number of diagrams with low level descriptions can specify, in detail, sequences of actions of the data mining process.

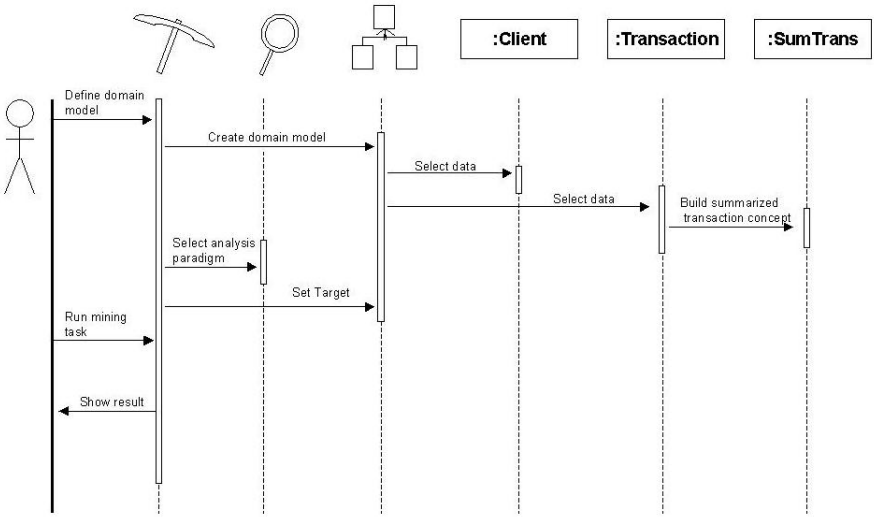


Figure 2.9: Interaction Diagram

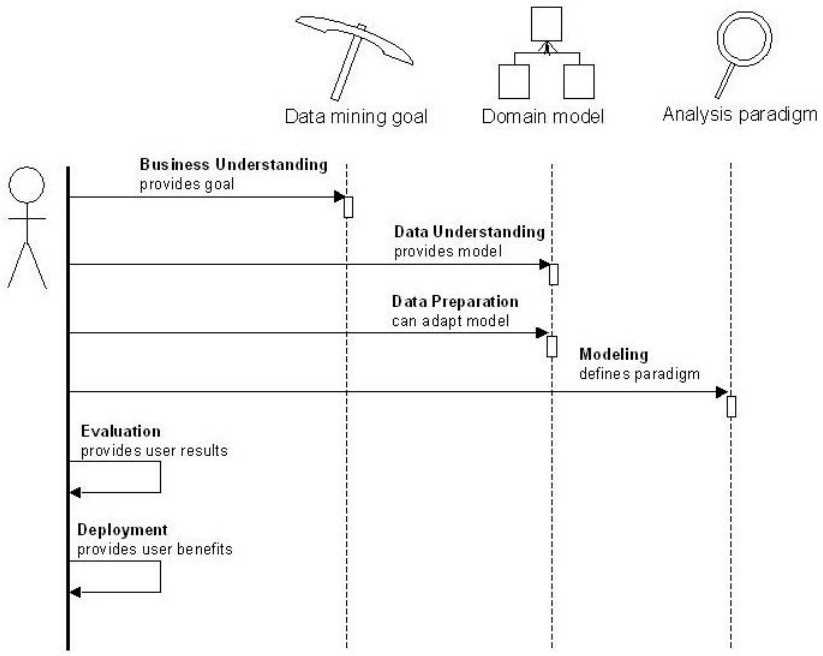
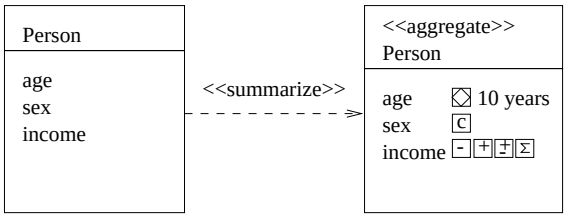


Figure 2.10: High level Interaction Diagram

2.4 Code generation from a visual language

Modeling the domain is only beneficial if we can use the domain specification to enhance the data mining process. In other words we want to produce some sort of code from class and interaction diagrams. Later on we can use this code to pre-process our data and start our data-mining algorithm. Before we can generate code from our diagrams we first need to determine what sort of code we want to produce. Do we only want to a create code sceleton[3] that defines the structure and the sequence of the data mining task? In that case the data mining expert has to provide the implementation. Another option is to generate execution ready code (SQL, stored procedures or calls to some external program) for that part of the specification that is complete and unambigious.

Figure 2.11 shows an example of the SQL generated from a Person concept that is summarised into age categories of ten years. For these age categories we compute the number of men and women in them and statistics about their income.

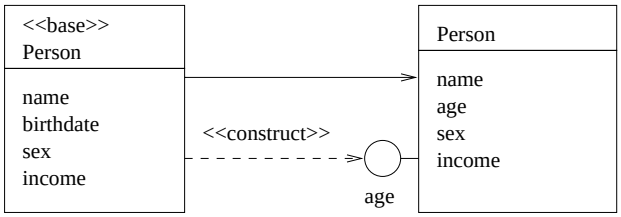



```
SELECT FLOOR(age/10), SUM(IF(sex='M',1,0)) maleCOUNT,
       SUM(IF(sex='F',1,0)) femaleCOUNT,
       MIN(income) incomeMIN, MAX(income) incomeMAX,
       AVG(income) incomeAVG, SUM(income) incomeSUM
FROM Person
GROUP BY FLOOR(age/10)
```

Figure 2.11: Summarisation to age categories and the equivalent SQL code

An example of feature construction is shown in figure 2.12 where the age of a Person concept is computed from the attribute *birthday* and the Oracle variable *SYSDATE* that holds the current date. For the construction of new features we could also use the operators from the M4-model. The only thing we need to do then is provide the relations to the concepts and fields that are needed to compute the operation and deliver its result in the new feature.

[3]This is what current UML modeling tools like *Rational Rose*, *Telelogic tau UML suite* and *TogetherSoft Control Centre* can do.




```
SELECT name, FLOOR(MONTHS_BETWEEN(SYSDATE-birthdate)/12) age, sex,
       city, income
FROM Person
```

Figure 2.12: Construction of the age field and the equivalent SQL code

2.5 Domain Knowledge Elements

The visual elements from the previous section have a relation to the Domain Knowledge Elements from WP5.

- Data Collection History
  This is part of the $<<base>>$ concepts which represent our stored data.
- Data Model
  These are models about the data used in the data-mining process and as such they are represented by all our visual elements. Where all the $<<base>>$ concepts together are the database model. By hiding all the less important concepts we can get representations of the business, conceptual model and analysis model.
- Causal Model
  This is a view of the domain model showing the connections between the $<<base>>$ concepts and the high level concepts on which you want to mine.
- Design Patterns
  A high level case build up of the most abstract concepts can be used as a design pattern for which you now only need to make the connections to your stored data.
- Analysis Paradigm
  The Analysis Paradigm can be found in the interaction diagram.
- Goal Specification
  The Mining Goal can be found in the interaction diagram.

- Background Knowledge
  Things like the best *time frame* to aggregate over or which aggregates have the most meaning in certain situations are all background information. Also known features and from which concepts they can be computed belong to this group. Associations and dependancies between concepts or stereotypes represent them.

- Integrity Constraints
  Integrity Constraints are contained in the concepts where they provide information over constraints on the attributes in this concept.

Chapter 3

Using visual language elements in a case

In this chapter we will focus on how the previously introduced language elements can be used in a case and how parts of this case can be reused for other cases. First we focus on the construction of the domain model and then we will look at the data mining process steps.

3.1 Constructing the domain model

The case we will be looking at here is credit risk assessment[1]. If a client of a bank requests a loan, the bank will want to make an estimate of the risk that the client doesn't repay the loan.

We will use a top down approach, meaning that first the concepts of the domain will be introduced and later on these concepts will be tied to their database representations. It should be mentioned here that the reverse process, a bottom up approach, is also possible. In that case base concepts would first be imported from a database and then the conceptual model would be constructed from these base concepts.

In order not to make this case too complex we will consider a simplified bank for which the following statements hold:

- clients of the bank can have one or more accounts;
- by requesting a transaction an account can be credited or debited;
- a client may have one or more permanent orders per account for recurring payments;

[1]The data model introduced here is based on the financial dataset that has been made available for the PKDD1999 and PKDD2000 conferences.

- a client may request one or more loans per account;
- per account two credit cards may be issued;

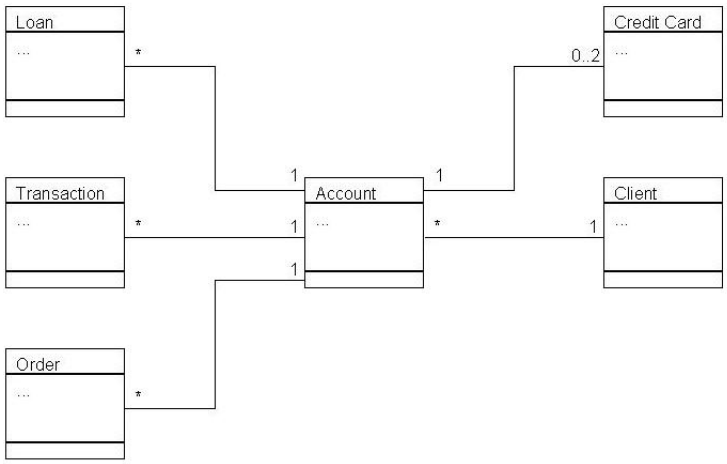


Figure 3.1: Conceptual model

Now a first version of a conceptual model can be constructed for this domain (see figure 3.1). The figure shows the concepts Client, Account, Transaction, Loan, Order and Credit Card. Furthermore the relationships between the concepts are shown.

The bank also wants to be able to see which clients form a household. For this the concept Household is added to the model using the aggregation symbol. Further the bank is interested how demographic aspects as average income, unemployment and number of crimes per year influence client behaviour. This demographic data per district is obtained from an external company. Then as a last addition the bank would like summarised information about transactions per account per month. This includes information like the minimum, maximum and average balance and the number of transactions.
An adapted version of the conceptual model is shown in figure 3.2. Only the attributes have been shown that are relevant at this point.

The bank wants to be able to distinguish good clients from bad clients. In the respect of loans a Good Client is a client that has paid back (or still is paying back) his loan without any problems, whereas a Bad Client is a client that did have problems paying back his loan. This can be visualised by constructing a new attribute in the Client concept, which is called Loan_OK and

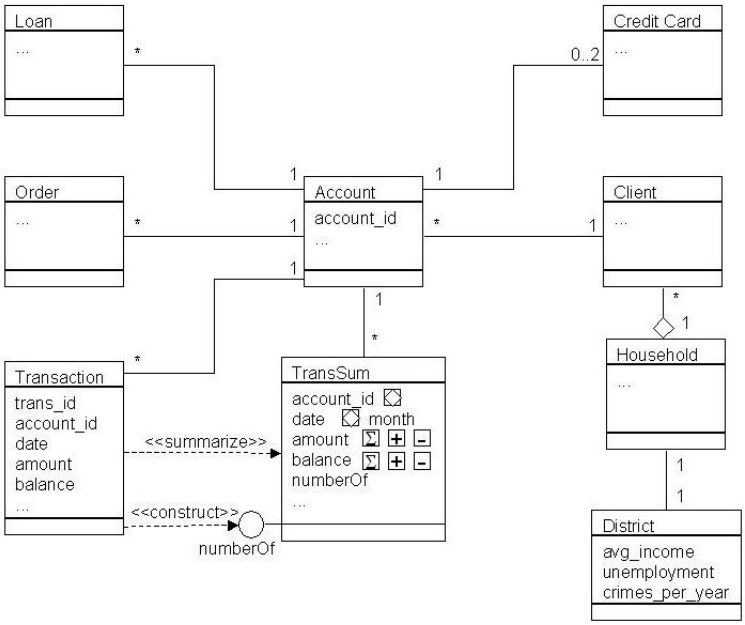


Figure 3.2: Updated conceptual model

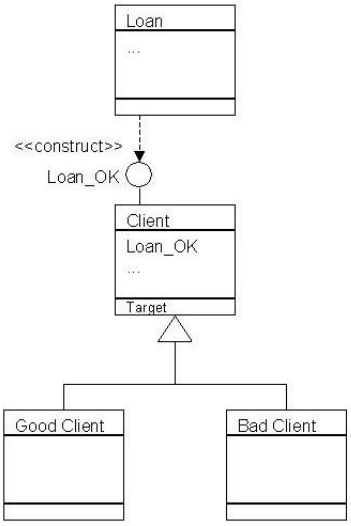


Figure 3.3: Classification of Client concept in Good and Bad Clients

that is derived from information from the Loan concept. Figure 3.3 shows the part of the conceptual model that has been updated for this. Note that

we don't have to show the *whole* model every time. It is allowed to split UML diagrams and focus on the part that is relevant at a certain point.
With these steps the conceptual model has been completed, a goal has been set and an analysis paradigm has been defined.

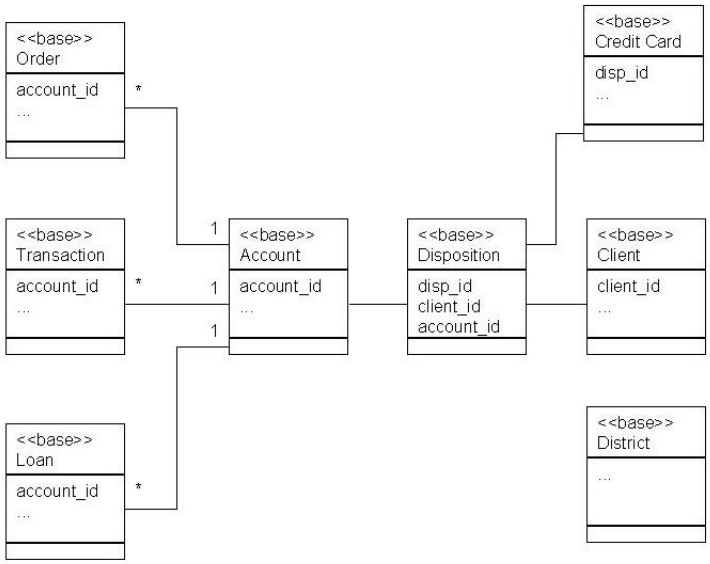


Figure 3.4: Database model

We will now look at the step of relating the concepts from the conceptual model to their representations (objects or tables) in a database. In this case the bank has a relational database and the concepts are stored as tables. It contains the concepts that are shown in figure 3.1. The demographic data is read from a different database. The resulting database model is shown in figure 3.4.

Note that the database model contains a table Disposition that is not part of the conceptual model. This table states who is allowed to dispose over an account. Conceptually, however, this table is not needed. Another point to notice is that the District table does not have any relationships yet with other tables.
The concepts Loan, Transaction, Order and District can be related directly to their base representations, because they can be used directly without any transformation.

The following steps have to be taken in order to connect the remaining concepts to the base concepts:

- create the Account and Credit Card concepts using their base representations and the Disposition table;
- construct the Household concept from the Client table;
- connect the Household concept and District table by a primary - foreign key relationship;

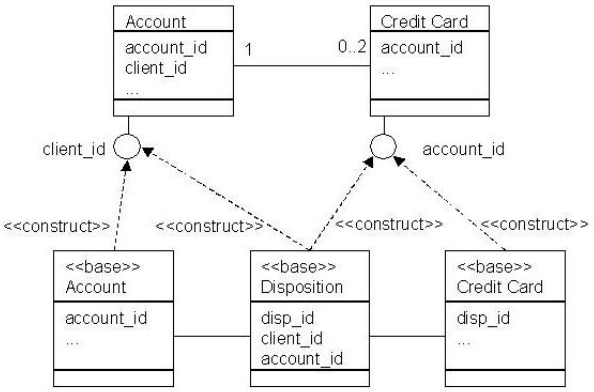


Figure 3.5: Creation of the Account and Credit Card concepts from their base representations

Figure 3.5 shows how the concepts Credit Card and Account can be formed from their base representations.

The Household concept can be created from the Client table. Clients who live on the same address will be assumed to share the same household. Let us assume that the address information is moved from the Clients table to the Household concept and that the postal code in the address can be used as a primary - foreign key with the District table. In this way the relationship between Household and District can easily be created. To make this example not too long we will skip the diagrams for creating the Household concept and connecting the Household concept to the District table. The diagrams that are needed for this are, however, comparable to figure 3.5.
With these steps the construction of the domain model has been completed.

It has been shown in this section how the domain model can be constructed and visualised. First the basic concepts in the domain have been defined.

Then some more concepts have been added. After that the target table has been chosen and a classification goal has been set. Finally it was shown how base representations can be connected to the previously defined concepts.

3.2 Capturing data mining process steps

Data mining process steps can be described using UML interaction diagrams and the extensions introduced earlier. These steps can be looked at from different detail levels. The high level steps that have been done up till now, can be visualised in an interaction diagram as shown in figure 3.6.

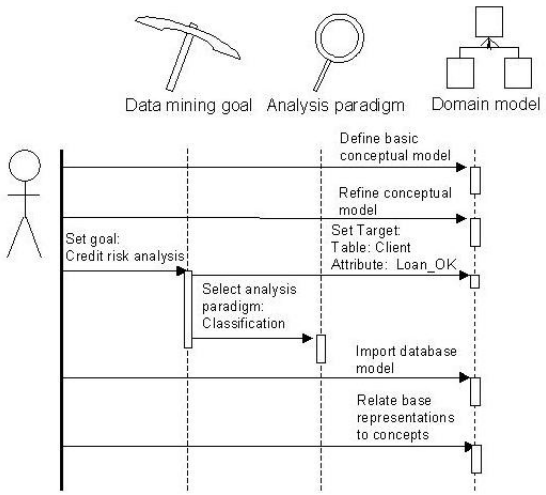


Figure 3.6: High level view of mining process steps

One can also use an interaction diagram for a more detailed view. A more detailed view of the step 'refine conceptual model' is shown in figure 3.7. It shows the involved concepts and actions.

If we would want to do cross-validation on the created domain model in our example, this could be depicted as shown in figure 3.8. Data from the Transaction concept is split into two datasets. The first dataset is used for building a knowledge model and the second one is used to estimate the validity and accuracy of the knowledge model.

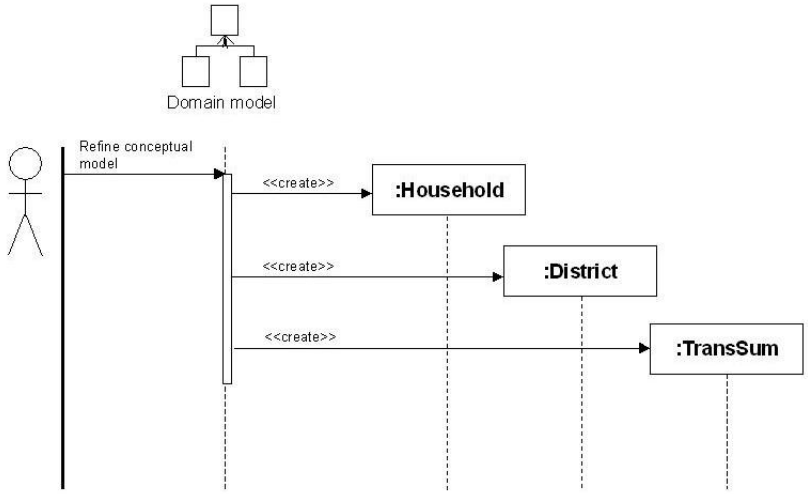


Figure 3.7: Lower level view of one mining process step

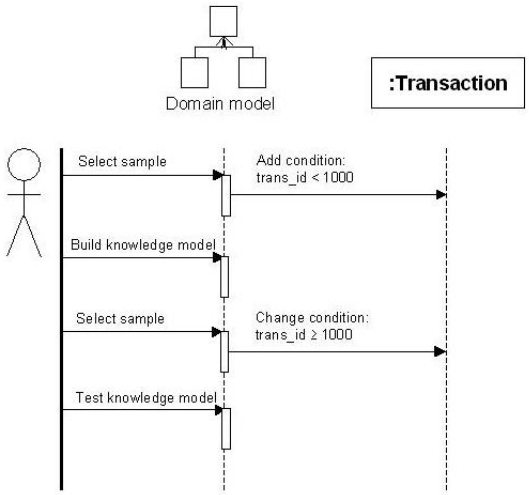


Figure 3.8: Cross-validation step

The three previously shown figures illustrate how a data mining process can be visualised. As stated before: high level diagrams make a coupling between concepts in the domain model and the tasks that should be performed on them. Therefore figures 3.6 and 3.8 together can be seen as a design pattern for credit risk assessment using cross-validation.

Chapter 4

Conclusion and Outlook

In this workpackage we have introduced a visual language that can be used to capture domain knowledge and define the data mining process itself. High level sequence diagrams capturing a data mining process can be reused in other data mining problems. Thus they can form a design pattern for handling certain types of data mining problems.
Visualizing domain knowledge and the data mining process will make communication easier between a data mining expert and a domain expert in a data mining project. The domain expert can focus his attention on the conceptual model of the domain and the data mining expert can use this model to map the concepts to the base concepts. Thus preprocessing steps can be defined in a visual way that helps the data mining expert to keep the overview of the data mining process. Some of the diagrams that define these preprocessing steps could be transformed directly into a formal language like SQL. Because UML is not a formal language, the application that provides the modeling functionality should, however, help the user to draw only diagrams that can be translated to a formal language.

A natural extension to the new techniques in data mining is mining in a distributed environment. Practical experience has shown that rich sources of information are provided by an increasingly large number of distributed and heterogeneous databases. We think that with the use of the UML extension that we have proposed in this workpackage, distributed database environments can be modeled and used for data mining. Concepts, like *consumer* and *product type* for example, can be specified on a global level. Mappings from these concepts can be specified to the different databases using <<*summerize*>> and <<*construct*>> dependancies, associations and inheritance. This task can be assisted by using techniques like attribute equivalence theory [5], quantitative measure of relevance [2], text mining, ontology learning and grammar induction [1].

Bibliography

[1] Adriaans P.W., Trautwein M.H., Vervoort M.R., *Towards high speed grammar induction on large text corpora*, in Proceedings of SOFSEM2 2000, 2000.

[2] Huan Liu, Hongjun Lu, JunYao, *Towards Multidatabase Mining: Identifying Relevant Databases*, IEEE Transactions on knowledge and Data Engeneering, IEEECS Log Number 105570, 2000

[3] Date C.J., Darwen H., *Foundation for Future Database Systems, The Third Manifesto*, second edition, 2000, Addison-Wesley.

[4] Haas E. de, *Logics for information systems*, ILLC Dissertation Series 2001-03, 2001

[5] Haihong Dai, *An Object Oriented Approach to Schema Integration and Data Mining in multiple databases*, Proceedings of the Technology of Object-Oriented Langueges and Systems-Tools 24, 1998.

[6] Knobbe A., Schipper A., Brockhausen P., *Domain Knowledge and Data Mining Process Decisions*, MiningMart Deliverable D5 IST research project, 2000.

[7] Morisio M., Travassos G.H., Stark M.E., *Extending UML to Support Domain Analysis*, The fifteenth IEEE International Conference on Automated Software Engeneering, 2000.

[8] Arno Knobbe & Arno Siebes & Hendrik Blockeel & Danil van der Wallen, *Multi-Relational Data Mining, using UML for ILP*, In Proceedings of PKDD 2000, 2000.

[9] Knobbe, A.J., Blockeel, H., Siebes, A., Van der Wallen, D.M.G. *Multi-Relational Data Mining*, In Proceedings of Benelearn '99, 1999.

[10] Quinan J.R., *C4.5 Programs for Machine Learning*, San Mateo: Morgan Kaufmann, 1993.