Using UML extensions for specifying domain knowledge for data mining

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

[1] Perot Systems Nederland B.V.,
P.O. box 2729, NL-3800 GG Amersfoort, The Netherlands
e-mail: {marc.dehaas, olaf.rem, ronnie.bathoorn}@ps.net
[2] University of Groningen
Dept. of Computer Science
D-44221 Groningen, Netherlands
e-mail: {r.w.bathoorn}@wing.rug.nl

Abstract. Multi-relational data mining algorithms that can mine on complex structured objects are now maturing. The construction of algorithms that can mine on structured objects is not beneficial unless end users are able to configure and run these algorithms conveniently, understand the results, and use the results to solve their business problems. The new methods for mining on structured objects require new ways for specifying the domain. In this discussion paper we present UML extensions for specifying domain knowledge.

1 Introduction

Traditional data mining algorithms, such as C4.5 (Quinan 1993 [11]) and others [2], can discover patterns that can be expressed in attribute-value languages which have the expressive power of propositional logic. These languages are limited and do not allow for representing complex structured objects and relations among objects or their components. In most relational data models these structured objects exist, and are interesting for learning purposes. Advances in the field of machine learning, such as multi relational data mining [9, 10] and ILP [12], have lead to techniques that can mine on structured objects. This enables the data mining expert to focus on the objects and relations that she/he wants to use in the learning process without the burden of propositionalizing/summarizing the data.

In this paper we present extensions of UML for specifying domain knowledge for a multi relational data mining environment. The use of UML (Unified Modeling Language) class diagrams for specifying domain knowledge is not new [6, 9, 8]. But we think that the extensions we propose are crucial for improvement and further automation of the data mining process. Next to the extensions of UML class diagrams we also propose the use of interaction diagrams for specifying the dynamic part of the domain knowledge.

UML is a standard in the object oriented analysis and design practice. This language is the most influential reference for the concepts of object oriented

information systems. There is a good connection between this design language and the conceptual 'lower level' coding languages for databases [5, 4].

The remainder for the paper is organized as follows: Section 2 describes the UML extensions. Section 3 concludes the paper.

2 UML extensions for multi-relational data mining

In the context of the data mining process 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. We introduce new stereotypes *<<base>>*, *<<aggregate>>*, *<<constructs>>* and *<<summarize>>* 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, UML sequence diagrams can be used. For a better connection with the data mining process we introduce new visual elements for *data mining goal*, *analysis paradigm* and *causal model*. These sequence diagrams can represent very detailed sequences of operations, but could also be more high level. These high level diagrams can be (re)used as a design pattern for a similar data mining goal.

2.1 Data mining extensions for UML class diagrams

- Class and *<<base>>*
  For the visualization 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 visualized 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 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 shows 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 which in a given structure form the concept OH-Group.
- Association
  Between our concepts we can have relations. Visually a relation is represented

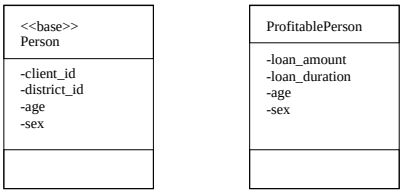

Fig. 1. concepts

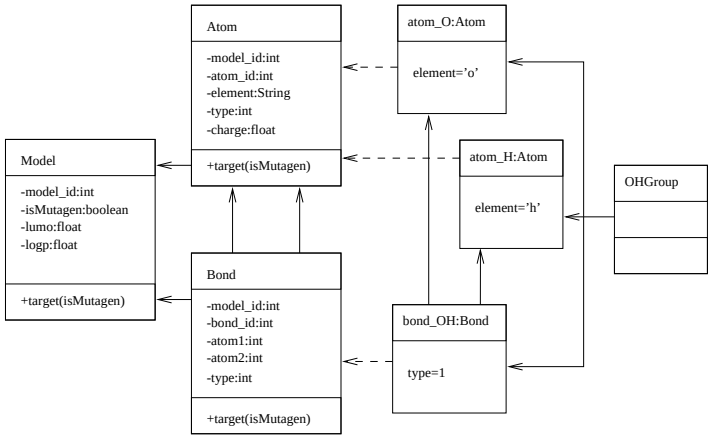

Fig. 2. structural model of an OH group

by an association (see figure 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.

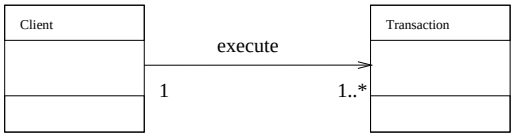

Fig. 3. association

– Aggregation & Composition
Aggregation and composition are two special kinds of relations. If 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 visualized by a diamond shaped box at the end of a relation and composition with a diamond shaped box filled with black (see figure 4). Notice that we also want to be able to specify a reading direction for these relations.

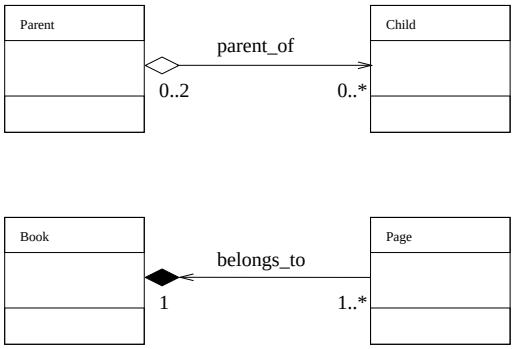

Fig. 4. aggregation (top) and composition (bottom)

– <<*summarize*>> and <<*aggregate*>>
Summarization is a special kind of aggregation where you have only one

concept and build a summary of the data of groups in your data. For this summarization you can use operations like *average*, *minimum*, *maximum* and *sum*. These operations are represented by small boxes behind the attributes (see figure 5). The attribute over which you are grouping the data gets a diamond behind it. The summarization is represented by a dependency arrow with the stereotype *<<summarize>>*. The class representing the summarized concept has the stereotype *<<aggregate>>*. Notice that a concept with stereotype *<<aggregate>>* can have *<<summarize>>* dependencies with more than one concept.

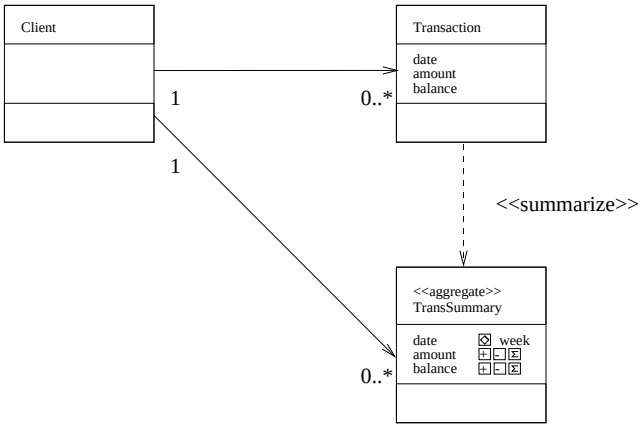

Fig. 5. summarization

– *<<construct>>*
New Features can be constructed by taking data from different concepts and using it to compute a new attribute in another concept. The new feature is visualized by a circle attached to the concept with dependencies to all the tables which data was used to compute it (see figure 6).

– 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 Good Clients and Bad Clients without having to specify directly on what criteria you want to divide this concept (see figure 7). In this way classification can be specified.

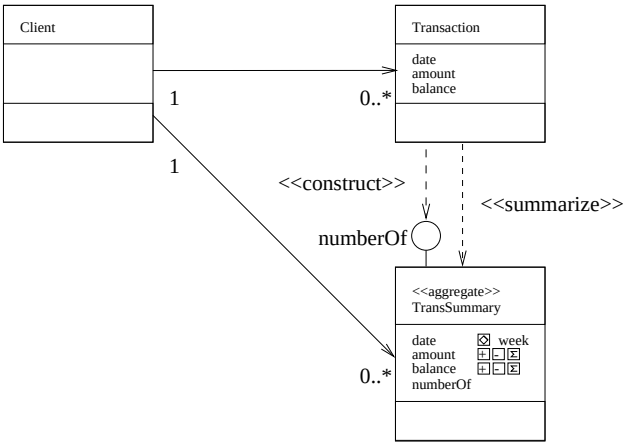

Fig. 6. feature construction

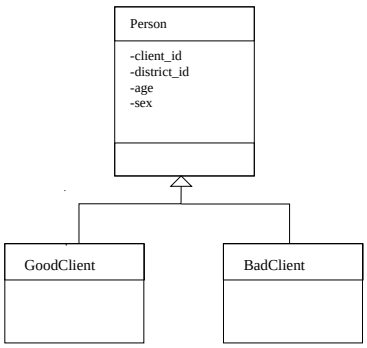

Fig. 7. inheritance

2.2 Data mining extensions for UML interaction diagrams

- Data mining Goal
  This element specifies what you want the dataming 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 datamining 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.
- Causal 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. With these real world equivalents being tables in a relational database, objects in an OO-database or data from other sources.

Fig. 8. Interaction Diagram extensions

With these extensions we can now build interaction diagrams that represent a datamining process for a certain mining task as seen in figure 9. In this interaction diagram we can make a coupling between concepts in our causal model and the tasks which should be performed on them. Thus creating a recipe for a datamining problem. If we use this interaction diagram the other way around we get a design pattern for a datamining problem. That is, we take an existing interaction diagram and create the concepts used in it for our database model. In this way we get a mapping between our data and the concepts in our data mining process.

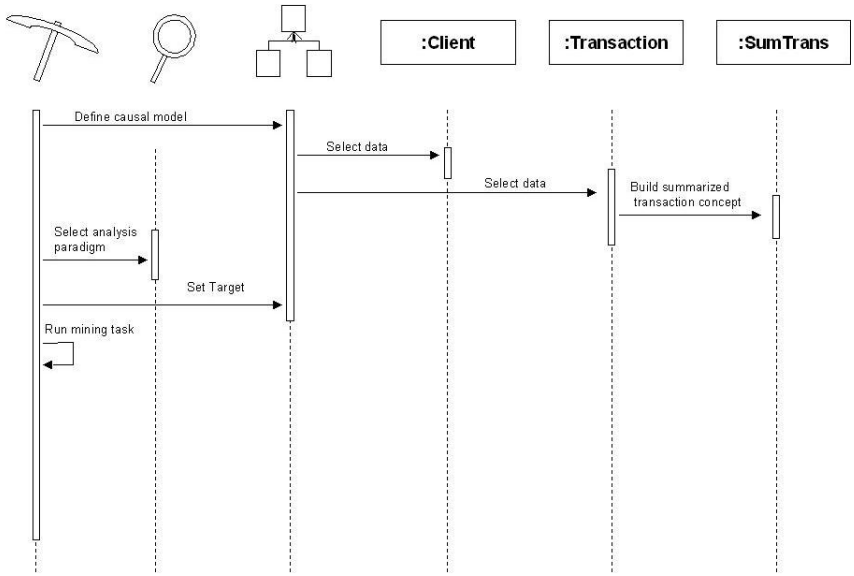

Fig. 9. Interaction Diagram

3 Conclusion

The advances in mining of new data formats and structures require new methods for users to configure and run these algorithms conveniently, understand the results, and use the results to solve their business problems. UML is becoming more popular in a wide range of application domains. UML has grown into the de facto visual language for specifying, constructing and documenting the artifacts of information systems. It takes advantage of the structured features offered by the Object Oriented methodology. The multi-relational algorithms require a structured specification of the data environment/domain. We think that using UML with some data mining specific extensions is a promising direction for building future data mining environments. We are just starting to get feedback from the practice using this approach. The concepts of the language presented in this paper are now being tested in the MiningMart IST research project.

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 paper, 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*>> dependencies, associations and inheritance. This task can be assisted by using techniques like attribute equivalence theory [6], quantitative measure of relevance [3], text mining, ontology learning and grammar induction [1].

References

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. Agrawal R., Srikant R., *Fast algorithms for mining ssociation rules*, in VLDB'94 pp. 487-499.
3. Huan Liu, Hongjun Lu, JunYao, *Towards Multidatabase Mining: Identifying Relevant Databases*, IEEE Transactions on knowledge and Data Engineering, IEEECS Log Number 105570, 2000
4. Date C.J., Darwen H., *Foundation for Future Database Systems, The Third Manifesto*, second edition, 2000, Addison-Wesley.
5. Haas E. de, *Logics for information systems*, ILLC Dissertation Series 2001-03, 2001
6. Haihong Dai, *An Object Oriented Approach to Schema Integration and Data Mining in multiple databases*, Proceedings of the Technology of Object-Oriented Languages and Systems-Tools 24, 1998.
7. Knobbe A., Schipper A., Brockhausen P., *Domain Knowledge and Data Mining Process Decisions*, MiningMart Deliverable D5 IST research project, 2000.
8. Morisio M., Travassos G.H., Start M.E., *Extending UML to Support Domain Analysis*, The fifteenth IEEE International Conference on Automated Software Engeneering, 2000.
9. Arno Knobbe & Arno Siebes & Hendrik Blockeel & Daniël van der Wallen, *Multi-Relational Data Mining, using UML for ILP*, In Proceedings of PKDD 2000, 2000.
10. Knobbe, A.J., Blockeel, H., Siebes, A., Van der Wallen, D.M.G. *Multi-Relational Data Mining*, In Proceedings of Benelearn '99, 1999.
11. Quinan J.R., *C4.5 Programs for Machine Learning*, San Mateo: Morgan Kaufmann, 1993.
12. Srinivasan, A., King, R.D., *Feature construction with ILP: a study of quantitative predictions of biological activity by structural attributes*, In Proceedings of ILP '96, 1996.