Integrating an Explicit Knowledge Model into Geographic Information Systems *

Miro Casanova, Thomas Wallet and Maja D'Hondt

System and Software Engineering Laboratory
Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels Belgium
E-mail: mcasanov | twallet | mjdhondt@vub.ac.be

Abstract

Geographic Information Systems (GIS) require the power of sophisticated database management systems for efficiently managing the persistency and consistency of huge amounts of data. A pool of software tools takes care of the production process, which consists of editing, visualising and selecting geographic data. However, the *quality* of the data – its well-formedness and integrity – is not so easily ensured. Quality is represented by declarative expressions that constrain the relations between concepts of the geographic domain knowledge. However, there exists a mismatch between this implicit model which takes into account the spatial and multi-dimensional character of the data, and the format in which the geographic data is actually stored in the database. Current practices have shown that hard coding these quality ensuring constraints using software engineering results in a loss of expressiveness and even accuracy. Therefore, in this research, we envisioned a co-existence of the usual components of a GIS with an explicit representation of the geographic domain knowledge and constraints, for which we employed knowledge engineering techniques. In this paper we illustrate and validate our ideas at the hand of the *GeoObjects* System, a GIS we developed together with our industrial partner, *TeleAtlas*.

1. Introduction

In this section we explain the state of the art in developing GIS's. We continue by showing that ensuring quality of geographic data is essential and that current practices have difficulties achieving this. Our contribution is to use an explicit model of the geographic domain knowledge and the quality ensuring constraints, thus combining knowledge engineering with sofware engineering techniques. Finally our industrial partner is introduced and the structure of the paper is layed out.

1.1 State of the Art

Digital geographic data, and especially data concerned with the road network, is used in sophisticated applications such as Geographic Information Systems (GIS) for environmental planning and control, alarm call and dispatch, and Fleet Management Systems, car navigation and geo-marketing. The data is modelled as a general, non-application specific planar-graph representation of the real world. On top of this model, a road network specific application model has been built.

Suppliers of geographic data are responsible for the production process of geographic data, which consists of capturing and storing the real-world geographic data. There are typically three important parts in the production process: software tools for editing, visualising and selecting the geographic data. The editing tool supports the insertion and correction of geographic data. The source material typically comes from satellite images, scanned maps, Global Positioning System data, but most frequently these data are entered manually. The visualisation tool is in charge of giving a graphical representation of the geographic data, displaying it as a map. The selection tool is responsible for selecting some part of the geographic data, corresponding to a region of the map that is represented by the data. Due to the spatial and multi-dimensional character of geographic data, this selection process is particularly complex and requires specific spatial query mechanisms. In the remainder of this paper we will use the term *software components* to refer to the database and production tools a GIS typically consists of.

In the past, the storage and efficient handling of spatial data were only possible in dedicated systems, especially built to serve as a GIS. Those systems provided their own spatial data types, spatial query functions, and a suite of software tools. Nowadays, most database suppliers are ex-

*This research, in collaboration with *Tele Atlas Data Gent NV*, is part of the project *GeoObjects (IWT 990025)*, funded by the *Flemish Institute for the Improvement of the Scientific-Technological Research in the Industry* (IWT).

tending their Data Base Management System (DBMS) with spatial data types and functions enabling spatial queries. Moreover, a DBMS has the enormous advantage of providing the typical database functionalities for persistency, consistency, transactions, optimised querying and so on. These functionalities are absolutely essential when developing large and efficient data-oriented software systems.

1.2 Quality of Geographic Data

The use of the source material in the production process described above indicates that many mistakes may creep into the geographic data. Therefore, one of the major challenges in the production process of geographic data is ensuring its quality, which we define here as the integrity and well-formedness of the geographic data, and should not be confused with consistency of the data. The importance of delivering high-quality geographic data is easily understood when thinking of the consequences of using data of poor quality in applications such as car navigation and alarm call and dispatch. Quality is defined by means of implicit constraints on the concepts and relations of the geographic domain knowledge.

There exists a mismatch between the geographic domain knowledge which takes into account the spatial and multi-dimensional character of the data, and how the geographic data is actually stored in the database. Moreover, when examining current practices we find that the domain knowledge is mostly never explicitely used or modelled, and that the constraints are described in natural language and manually implemented together with the production tools using software engineering techniques. This approach results in a tangling of the implicit quality constraints in the application code, hence causing them to be non-modular and hard to localise. This has significant consequences with respect to manageability, adaptability and reuse of the constraints. In addition to this, because the constraints are inherently declarative, they lose expressiveness and even some of their original meaning in a procedural medium.

1.3 Our Contribution

In this paper we argue that in addition to the aforementioned software components, i.e. the database and the production tools, a GIS also requires an explicit *knowledge model* and corresponding *constraints*, as shown in Fig. 1. The knowledge model is actually a representation of the domain knowledge, in this case the geographic data, whereas the constraints are a declarative, high-level, unambiguous, explicit and modular representation of the quality ensuring domain constraints. Hence, a GIS not only consists of software engineering components, but also incorporates domain knowledge for which knowledge engineering mechanisms

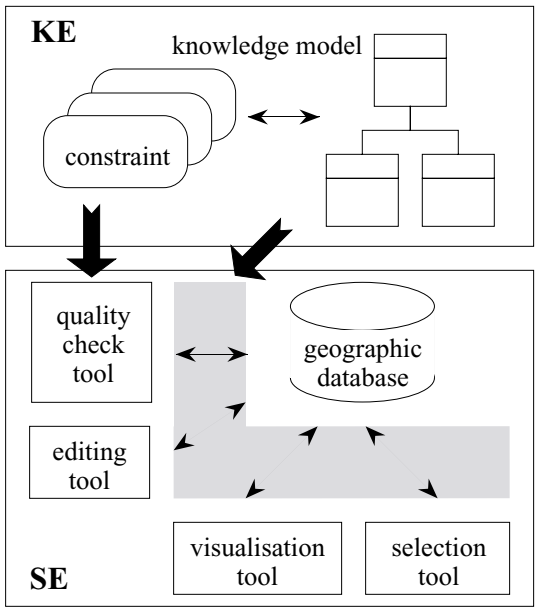

Figure 1. The general architecture GeoObjects system.

are needed.

The following section elaborates on the separation of the knowledge model and the constraints from the software components of a GIS. Section 3 then describes how the knowledge model and the constraints are linked to these other software components and make them operative. It is important to note that through this link, the high-level constraints are able to be checked on the geographic data in the database, and all the software tools use the concepts of the knowledge model instead of the implementation-dependent format of the database.

This research is the by-product of a project we are involved in with *TeleAtlas* – an important supplier of geographic data world-wide – as the other partner. In this context, the existing GIS of TeleAtlas was adapted to incorporate the ideas presented here. The resulting *GeoObjects* system, a real-world GIS which integrates knowledge engineering techniques and software engineering techniques as described above, will be used throughout the paper to illustrate and validate our ideas.

2. Explicit Knowledge Model and Constraints

As argued in the previous section, the model of the domain knowledge of geographic data should be modelled explicitly, and separated from the software components of a GIS. As a result, the constraints that dictate the quality of the geographic data can be expressed on this model in a

high-level and implementation-independent manner. The rest of this section first explains what the knowledge model contains and how it is modelled, and then illustrates the constraints and how they help to ensure quality.

2.1. Knowledge Model

The most widely accepted model for geographic data is the Geographic Data Files (GDF) standard [GDF], which has been created in order to improve the efficiency of capturing and producing road related geographic information. GDF achieves this efficiency by providing a common reference model on which clients can base their requirements and suppliers can base their product definition. The foundation of the GDF standard consists of a general, non-application specific planar-graph representation of the real world. On top of this model, a road network specific application model has been built. The last model describes real-world concepts in the domain of geographic road network data, as well as attributes of these concepts and relations between these.

To illustrate the concepts in the geographic domain knowledge that are described by the GDF standard, consider *RoadElement* and *Junction*. The relation between them is that a *RoadElement* has one or two *Junctions* and that a *Junction* has at least one *RoadElement*. Attributes of *RoadElement* are the average speed allowed of vehicles travelling along it, whether it is currently under construction or in the planning stage, and its direction of traffic flow. A *Manoeuvre* forms a link between a number of *RoadElements* and a *Junction*, where the order is important. A *Manoeuvre* indicates a certain path that can be followed by a vehicle. A *RestrictedManoeuvre* represents a manoeuvre that is restricted, which means a vehicle is only allowed to take this manoeuvre and no other. A *ProhibitedManoeuvre* indicates a manoeuvre that is prohibited. Figure 2 shows a real-world situation consisting of several *Junctions*, *RoadElements* and a *Manoeuvre*, whereas Fig. 3 depicts the corresponding part of the knowledge model. In this last figure it becomes apparent that in the particular case of the GeoObjects system, we used the *Unified Modeling Language* (UML) [FS99] for our knowledge model because it is suited for describing the geographic entities and relationships.

2.2. Constraints

Even if the knowledge model is a high-level representation of real world geographic entities, it does not contain all the necessary domain knowledge to reach the desired degree of quality. It is not possible to describe all the existing relationships among the entities of the domain in just the knowledge model described above, only some very limited kind of domain restrictions. Due to this, an additional

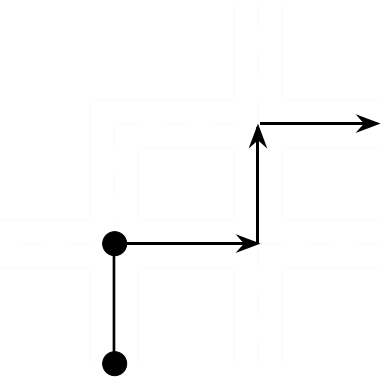

Figure 2. Real-world situation representing Junctions (black dots), RoadElements (the line connecting two dots) and a Manoeuvre (the sequence of arrows).

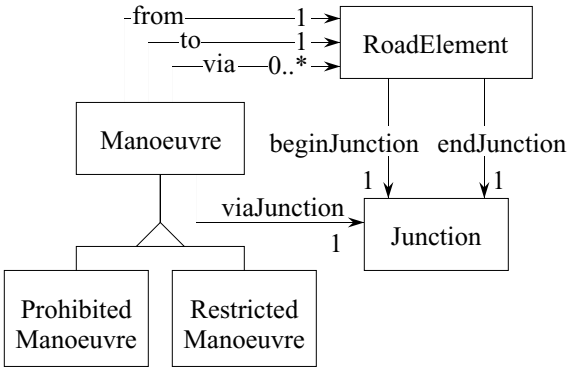

Figure 3. Part of the knowledge model representing the concepts Junction, RoadElement and different kinds of Manoeuvres, and the relations between them.

medium is needed for describing the constraints.

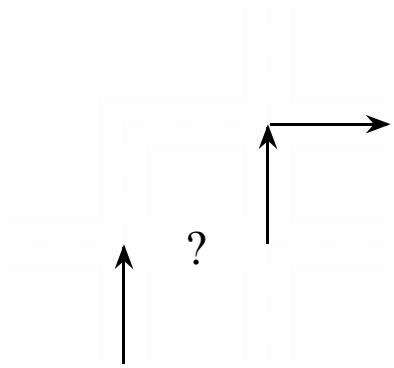

Figure 4. A non-continuous set of RoadElements.

Let us illustrate the requirements of the constraint language with an example. A *Manoeuvre* is the path that a vehicle has to follow for going from one point to another, so it is normal to model a *Manoeuvre* as a set of *RoadElements* (see Fig. 3). However, there is a restriction in the geographic domain that cannot be represented in the model, which is that the sequence of *RoadElements* should be continuous, i.e. that the end of a *RoadElement* should coincide with the beginning of the following one. Intuitively, this means that the vehicle, while performing the *Manoeuvre*, cannot ‘’jump” from one point to another, as seen in Fig. 4. As this situation might well be present in the geographic data due to inprecisions in the production process described in Sect. 1, there is definitely a need for constraints declaring that this situation is faulty. Constraint checking mechanisms, as described in Sect. 3, must then ensure that these constraints are true for all the data in the database. The constraint used to find situations such as the one depicted in Fig. 4 is

The RoadElements to which a Manoeuvre refers shall be a continuous set of RoadElements

which is related to the following constraint:

A continuous set of RoadElements is an ordered set of RoadElements. For each RoadElement in the set, except for the first and the last, the beginJuncion and endJunction must be the same Junction as the beginJunction or endJunction of the next and/or the previous.

A last example is:

A RoadElement shall not be the first RoadElement of a RestrictedManoeuvre and a ProhibitedManoeuvre in case both these manoeuvres refer to the same viaJunction.

A *RestrictedManoeuvre* is a path of *RoadElements* between a starting and an ending point that a driver is obliged to follow when he drives from this starting point to this ending point, whereas a *ProhibitedManoeuvre* is a path of *RoadElements* that it is forbidden to drive on. A *viaJunction* is a *RoadElement*’s *Junction* given as a passing point of a given manoeuvre. The knowledge model contains different classes (*RestrictedManoeuvre*, *ProhibitedManoeuvre*, etc.) and different relations between them (*viaJunction*, *from*, etc.) that represent the different elements of this constraint (see Fig. 3).

Given the declarative nature of the constraints, we use UML’s accompanying *Object Constraint Language* (OCL) [KW99] with some adaptations [CWD00] as the declarative language for the constraints in the GeoObjects system. OCL provides the appropriate expressiveness to write constraints in a high-level language that can reason on concepts of the knowledge model. The previous constraint can then be expressed in OCL as follows:

```
context RestrictedManoeuvre inv:
ProhibitedManoeuvre.allInstances
    ->forall(p : ProhibitedManoeuvre |
       self.viaJunction = p.viaJunction
       implies self.from <> p.from)
```

In the following section we explain how these constraints written in OCL are managed by the GeoObjects system so that they can be used to ensure the quality of the geographic data in the database.

3. Integrating the Knowledge Model and Constraints into the Software Components

In this section we present how we integrated the knowledge model and the constraints into the software components. We first show how the knowledge model is made operational by implementing it in an API, and how the other software components employ it to access the geographic data in the database. Furthermore we describe how the constraints are integrated into this by a quality check tool that uses executable check routines, which are generated from the constraints.

3.1 Integrating the Knowledge Model

Since the knowledge model is obviously compatible with the object-oriented paradigm, it is implemented using software engineering techniques as part of the object-oriented API, which will be the connection between the geographic data and the rest of the software components of the GIS.

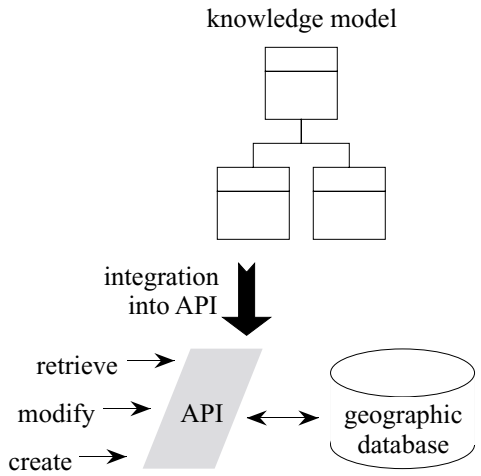

Figure 5. Knowledge model integration.

When a software tool uses the API to retrieve some geographic data from the database, the classes representing the domain knowledge concepts in the knowledge model are instantiated, and the resulting objects are initialised with the database results. Hence, it can be said that the geographic database's API is a software component that really contains the knowledge model. Fig. 5 schematically presents where the knowledge model is integrated into the GeoObjects system.

The presence of the API makes the extraction and manipulation of geographic data from the database transparent for the rest of the software components of the GIS, such as the selecting, editing, visualising and quality checking tools. As a result the set of software tools that compose the GIS can access the data no matter how it is stored in the geographic database. Moreover, the fact that the construction of the API is based on the knowledge model's concepts gives the advantage of dealing directly with a high-level representation of the geographic data.

3.2 Integrating the Constraints

The constraints should be checked at different moments on the GIS to ensure that the geographic data fulfills the requested quality criteria. Therefore it is crucial to integrate the constraints into a software component in the GIS that can use the API to access to the database. This component is the *quality check tool* (Fig. 6). Since the constraints are specified on the level of the knowledge model in a first order predicate logic language, they will need to be translated to a form more usable by this quality check software tool.

However, having those constraints expressed in OCL is not very useful for the quality check tool, because they are at that moment non-executable. Moreover, the constraints

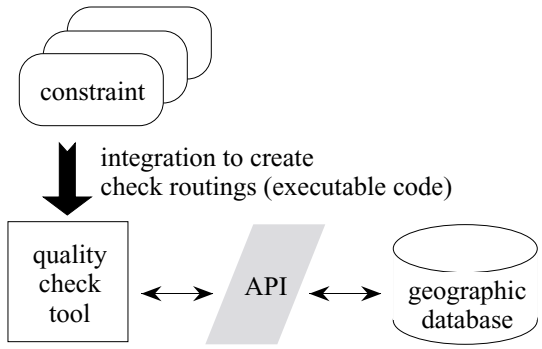

Figure 6. Constraints integration.

need to have access to the database, for retrieving the data corresponding to the domain knowledge concepts they reason about. Therefore we developed a specific translation engine (called BREK), that automatically translates constraints written in OCL into *check routines*, which are executable pieces of code. The check routines also contain the appropriate API calls for retrieving the correct data before checking if it complies to the quality. The translation engine carries out different operations, and uses compiling techniques to realise this translation [AU77]. It starts with lexical and syntactical analysis of the OCL constraints, and generates the corresponding abstract syntax tree. This syntax tree is type-checked to be sure that it complies to the OCL syntax and to make sure that the navigation via the concepts and relations in the knowledge model are correct. The final operation consists in the automatic generation of source code out of the syntax tree and some of the type information. Figure 7 shows the control flow of these operations. Briefly, the code generation is based on generating loops on the instances of the classes that are involved in the constraint and are retrieved by the API from the database. For every association, method or attribute of an instance, a method of the corresponding object given by the API is called. Finally, the result is stored as a boolean, which represents whether the checked data has been consistent with the constraint or not. Thus, the check routines are actually compilable code modules that are generated for each OCL constraint.

The quality check tool manages these check routines, which enables the user to select the appropriate constraints to be checked and select a part of the geographic database, and then run the corresponding check routine on the selected data.

4. Conclusion

We have argued that software engineering mechanisms are not sufficient neither adequate by themselves to en-

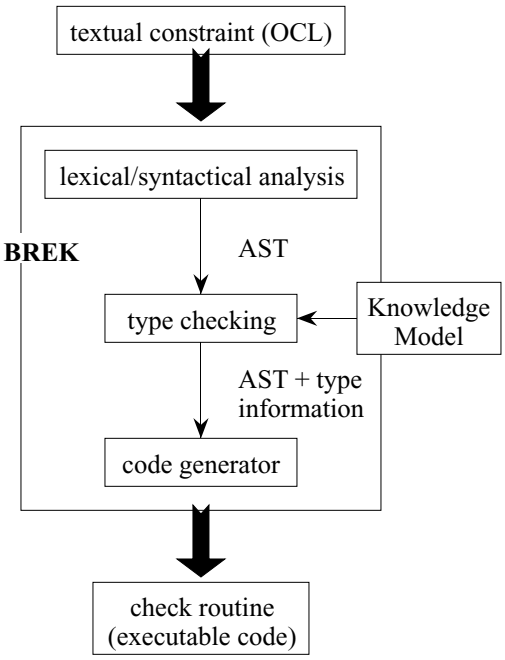

Figure 7. The control flow in the BREK.

sure a high degree of quality of the geographic data. We have shown as well that knowledge engineering techniques are suitable, in cooperation with software engineering techniques, to achieve our final goal. We validate our ideas by implementing them in the GeoObjects system that has been explained throughout the paper.

We use a knowledge model to explicitly represent geographic domain knowledge, and constraints to specify restrictions on the knowledge model in order to guarantee quality. The benefit of this in comparison with the current practices in building GISs is that we have a high-level representation of the geographic domain knowledge and constraints, and as a consequence allows it to be used as a fundamental part in the process of ensuring quality. Furthermore, having modular and high-level constraints is better than having them hard coded in the application code of the GIS, to facilitate maintenance and evolution.

The knowledge model is implemented as part of the API, thus inserting the domain knowledge into the software components of the GIS. This makes manipulating the geographic data transparent for the rest of the software components of the GIS. Since each software component of the GIS has to use the API in order to retrieve geographic data from the database, it implies that they use geographic data expressed in a high-level manner as opposed to the low-level query normally performed on a database. Consequently, if the software components are written in a high-level manner, they will be easier to evolve, maintain or reuse.

References

[AU77] Alfred V. Aho, Jeffrey D. Ullman. ‘’Principles of Compiler Design”. Addison-Wesley, 1977.

[CWD00] M. Casanova, T. Wallet, M. D’Hondt. “Ensuring Quality of Geographic Data with OCL and UML”. In Proceedings of the 3rd International Conference on the Unified Modeling Language : Advancing the Standard. UML 2000, York, UK. Pages 225-239. Springer, 2000.

[GDF] “The Geographic Data Files Standard”. Committee for Road Transport and Traffic Telematics of the Comité Européen de Normalisation.

[KW99] A. Kleppe, J. Warmer. “The Object Constraint Language: Precise Modeling with UML”. Addison-Wesley, 1999.

[MS95] Mark Stefik. ”Introduction to Knowledge Systems”. Morgan Kauffman, 1995.

[FS99] Martin Fowler, Kendall Scott. ”UML Distilled Second Edition: A Brief Guide To The Standard Modeling Language”.Addison-Wesley. 1999.