WiSPY: A Tool for Visual Specification and Verification of Spatial Integrity Constraints

Vincenzo Del Fatto
Faculty of Computer Science
Free University of Bozen-Bolzano
39100 Bolzano, ITALY
vincenzo.delfatto@unibz.it

Vincenzo Deufemia
Department of Computer Science
University of Salerno
84084 Fisciano (SA), ITALY
deufemia@unisa.it

Luca Paolino
Department of Research
Link Campus University
00162 Roma, ITALY
l.paolino@unilink.it

Sara Tumiati
South Tyrolean Municipality Consortium
39100 Bolzano, ITALY
sara.tumiati@gvcc.net

Abstract

Nowadays, most of tools for spatial data manipulation allow to edit information on maps without performing any integrity verification. On the other hand, data repositories such as the DBMS only permit few constraints to be defined by means of their Data Definition Languages and leave programmers to implement procedures for complex constraints. In this work we present the WiSPY system, a plugin of the GIS tool uDig for visually specifying and verifying complex spatial integrity constraints. WiSPY includes a visual environment for defining spatial data models with integrity constraints and for automatically generating the constraint checker. The latter is used by the WiSPY tool to verify the integrity of the data produced during the map editing process. The system has been validated on a real case study concerning the current regulation of the Public Illumination Plan (PIP) managed by an Italian municipality.

1 Introduction

The management of spatial data is one of the fields where companies and researchers have invested much money and time in the last decade. The result is that commercial products such as Autodesk Autocad Map 3D[1], Bentley Microstation[2], ESRI ArcGIS[3], or free and open source products such as Google Map[4], QGIS[5], GRASS GIS[6], uDig [22], have become part of the daily life not only for GIS users. This is because, they offer a large amount of features for spatial data manipulation, spatial analysis and reasoning functionalities that in many cases may support or simplify our activities. Despite the large amount of available features, these products lack of an adequate control during the editing phase, not allowing a solid constraint check. Also in the Database Management Systems (DBMS) field, commonly used product as Oracle and PostgreSQL, which offer spatial extensions, only allow basic functionalities to support constraint checking. Indeed, their Data Definition Languages (DDL) only support the management of simple topological constraints while advanced controls need to be coded. In this context, it appears to be desirable to provide a significant support in this phase in order to improve the quality of data and minimizing the implementation activities which often are annoying and repetitive.

In order to increase the dataset quality, correction operations can be performed both during the editing phase (on the fly) or after a data manipulation session (*a posteriori*). Both such approaches have pros and cons. On one hand, checking correctness during the editing phase has a direct effect on the data entry process, since the feedback is immediate, but the check can be performed only on a subset

[1] `http://www.autodesk.it/products/autocad-map-3d/overview`

[2] `http://www.bentley.com/it-IT/products/microstation/`

[3] `http://www.esri.com/software/arcgis`

[4] `https://maps.google.com/`

[5] `http://www.qgis.org/en/site/`

[6] `http://grass.osgeo.org/`

DOI reference number: 10.18293/VLSS2015-018

of data. On the other hand, an a posteriori check is performed on the whole dataset or on a selected subset of data, giving the possibility to apply a complete verification and to globally re-adjust geographic data. However, in this case the system returns feedback to the user just at the end of the manipulation process, making more difficult to handle possible errors.

To guarantee the effective verification of map constraints according to the designer requirements, in [7] a visual language parsing approach for constraint checking of input spatial data during the editing phase is presented. The integrity of data produced during the map editing process is guaranteed by a constraint checker automatically generated from a visual language grammar. In order to reduce the efforts for defining the constraints to be checked, a high-level data model is used to specify the user needs.

In this paper we propose a software system, named WiSPY, which consists of two uDig plugins that allow the user to:

- specify geographic models by means of the OMT-G visual modeling language;
- automatically translate the OMT-G models to the corresponding grammars;
- validate geographic incoming data against the constraint checker generated from the grammars.

The rest of the paper is organized as follows. Section 2 presents the case study based on the Public Illumination Plan which we used to validate the proposed system. Section 3 introduces the OMT-G modeling language used for specifying spatial integrity constraints (SICs, for short). Section 4 describes the WiSPY tool, while Section 5 shows its application to the considered case study. Section 6 discusses the work existing in literature related with our proposal. Conclusions and future work are given in Section 7.

2 Case Study

The case study presented in this section represents the current regulation of the Public Illumination Plan (PIP) managed by the South Tyrolean Municipalities Consortium (STMC), in South Tyrol, Italy. The STMC[7] is a cooperative founded in 1954 that includes among its members all the south Tyrolean municipalities and is mainly focused on legal practice, administrative training, labor legislation, and ICT services. The PIP is a complex set of regulations that can be difficult to interpret and apply correctly. In particular, these regulations define a complex set of lighting categories depending on the type of road the lamp is placed (urban or extraurban roads, pedestrian zone, bicycle paths), if the road is heavily busy or not, if there is a pedestrian passage, if there are crossing roads, and so on. The plan must be “safe for people and things” and implemented in order to limit light pollution. Light pollution is considered as a misdirected, excessive or obtrusive artificial light, causing a serious degradation of the natural nocturne light. A public administration must intervene in order to prevent such situations. In addition, a illumination system that involves wrong light bulbs, wrong lamp types or has an overestimation of the lamp power could create economical issues. The proposed WiSPY tool can help domain experts to better understand and effectively manage such a complex real world scenario.

The verification of the PIP is a typical task that a public administration is faced with and it is a complex task for different reasons, such as the difficulty of managing many types of geographic data involved in, as well as, the tight connection to the context they are inserted in. In fact, the simple containment spatial relationship is not sufficient to check a wide range of constraints that could depend on the context on which a lamp is being placed (type of road or area), the type of lamp itself, the type of illumination based on both lumen and lux. In addition, checking the correctness of the geographic data related to a street lamp could be tricky, because a lamp is normally represented as a point in space, while the constraints may need a polygon to be successfully checked. For example, checking the correct distribution of lamps along a road is not sufficient to compute the distance between the points of the lamps, it is also necessary to calculate the amplitude of the radiation given by the lux value.

The case study consists in the automatic verification of real geographic data related to the PIP of the South Tyrolean Municipalities. The data includes basic cartographic data (boundaries, hydrography, vegetation), roads, buildings, and of course the PIP. Based on the current regulations, a set of constraints suited to the validation of the chosen municipality’s PIP can be specified. The road types in the municipal boundaries are of type C (secondary extra urban road) and type F (local roads and bicycle paths). The following steps are necessary to determine which configuration is suitable for each road type:

- determine whether the road is of type C or F, including the speed limit;
- determine which technical illumination class is related to the road, in order to have the right luminance values;
- determine the type of lamp;
- determine the height of the poles;
- determine how to place the poles at the side of the road:
  - unilateral;

[7]`http://www.gvcc.net`

  - bilateral with alternate center;
  - bilateral with opposite center;
  - double centred (between the two carriageways);
- determine the distance of the poles.

All these factors must be taken into account during the verification process, and they depend on each others; determining the distance of the poles is the step that depends more on the other steps, whilst the first two steps are those that influence more the decision.

3 Visual Modeling Geographic Data embedding Integrity Constraints with OMT-G

Although existing data modeling approaches and tools can be adapted for geographic database design, most of them do not support certain aspects of the modeling process, such as the treatment of SICs. A visual language for modeling geographic data must be able to visualize different aspects of the data structure including numerous types of representations, such as point, line, polygon as well as non-spatial data; conventional as well as geo-referenced classes; different types of spatial relations, spatial constraints, spatial aggregation relationships.

Object Modeling Technique for Geographic Applications (OMT-G) is an object-oriented approach to model data for geographic information. Its notation is based on the classic OMT class diagram notation [5], and further extended to embrace also Unified Modeling Language (UML) concepts and notations [4]. OMT-G provides three types of primitives, based on the UML primitives for class diagrams, to model the geometry and topology of geographic data, providing support for topologic structures, network structures, multiple views of objects, and spatial relationships. These types are classes, relationships and SICs. These primitives allow also for the specification of alphanumeric attributes and associated methods for each class.

OMT-G offers three types of diagrams: the class diagram, that represents the classes involved in the model, as well as their relations; the transformation diagram, that permits the description of the transformation process of a class, if the class diagram indicates the need of multiple representation of it; the presentation diagram which describes how to represent the visual aspects of objects in the visualization.

OMT-G class diagrams are composed of conventional and geo-referenced classes. The first behave as UML classes and have no geographical properties. The latter include a geographical representation alternative, which specializes in two types of representations: discrete, associated with real world elements (geo-objects), or continuously distributed over the space (geo-fields). Geo-objects are represented with points, lines, polygons or network elements, whereas geo-fields correspond to variables such as soil type, relief and temperature. The relationships of a OMT-G class diagrams can be conventional, e.g. UML relationships, or georeferenced. The latter include topological relations (e.g. touch, in, cross, overlap, and disjoint), arc-node network relations and spatial aggregations.

OMT-G class diagram permits the derivation of the set of SICs that must be observed in the implementation. SICs can be classified in: *topological* (the geometrical properties), *semantic* (the semantic of the geographic feature), and *user-defined* integrity constraints, "business rules" and all those controls that are non-spatial. Topological integrity constraints include spatial dependencies, spatial associations, connectivity, and geo-field rules. Semantic integrity constraints include spatial association and disjunction rules. User-defined integrity constraints are obtained from methods that can be associated to the classes.

4 The WiSPY Tool

In this section we present WiSPY (Visual specification & Verification of SPatial integritY constraints), an extension of the uDig GIS tool [22] for enabling users to visually model geographic applications, also embedding SICs, and to verify the correctness of the input geographic data. WiSPY has been implemented by means of two plugins. In the following we present the architecture of the WiSPY tool and provide details about the implemented plugins.

4.1 The Architecture

Figure 1 shows the architecture of the proposed WiSPY tool. It has been implemented on top of uDig, which is a software program based on the Eclipse platform featuring full-layered Open Source GIS. In particular, uDig provides a complete Java solution for viewing, editing, and accessing GIS data. Since it is built on top of the Eclipse "Rich Client Platform", WiSPY has been developed in Java as two uDig plugins, namely OMT-G Editor and Constraint Checker.

The OMT-G Editor provides three different environments, one for each diagram the OMT-G data model provides. In the canvas of the class diagram editor, users specify their schema, adding classes and relationships chosen from the tool palette. The relationships selected from the palette can be inserted by clicking over the source class and dragging a line to the target class. As an example, Figure 2 shows simple OMT-G class diagram modeling *containment* constraints among Municipality, Lamp, and Road objects.

The OMT-G editor includes a function to derive a visual grammar modeling the SICs specified in the OMT-G data model. The Constraint Checker generated from the grammar by using the ANTLR parser generator[8] can be activated

[8]`http://www.antlr.org/`

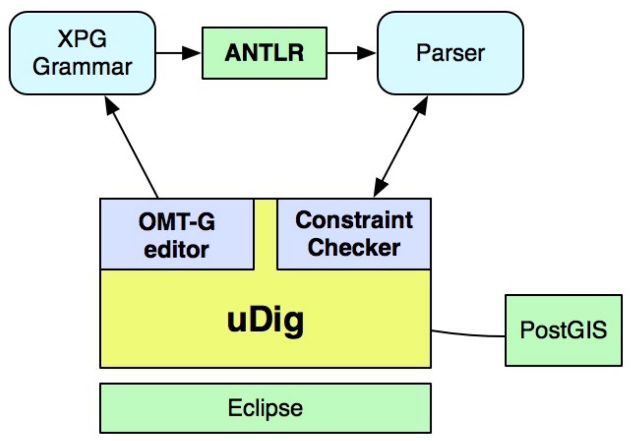


Figure 1: The architecture of WiSPY tool.

by the user during map editing phase. In particular, the input of WiSPY is a set of geographic data whose type is defined in the OMT-G data model. The output is the validation of the input data with respect to the SICs specified between the classes of the OMT-G data model. If a SIC is violated then a suitable error message is shown to user with information to recover from the violation.

4.2 Constraint Checker Generation

The WiSPY tool automatically generates a constraint checker able to verify the SICs defined by a OMT-G class diagram. In particular, WiSPY exploits the visual language compiler-compiler technique proposed in [6] for deriving a visual language parser through standard compiler-compilers, like YACC [11].

The OMT-G Editor allows users to develop their schema, adding classes and relationships chosen from the tool palette. The relationships can be annotated with SICs, which impose restrictions on the input data. In particular, the classes of the model represents the geographical objects that the user can place on the map, while the relationships specified between two classes define SICs on their instances. Such constraints are defined on the attributes of the involved classes. The editor provides the set of standard OMT-G spatial integrity rules (e.g., *contain relation*, *coincide relation*, *cross relation*, *touch*, *in*) as well as standard processes such as generalization and specialization. Moreover, the users can define new rules by specifying a set of conditions on the classes' attributes.

The constraint checker generation process consists of mapping the OMT-G class model into a visual grammar. To this end, we use the XPG grammar formalism [6], which is similar to context-free string grammars, where more general relations other than concatenation are allowed. In particular, an XPG textually describes a diagram by grammar productions that alternate (terminal and nonterminal) symbols with relations defined on the symbol attributes. Thus, the idea is to map the classes defined in the OMT-G schema, which represent the spatial objects to be placed on the map, into terminal symbols of the grammar, while the SICs defined between the spatial objects are modeled in terms of spatial relations among them [6]. In this way, the user can analyze the SICs specified for a particular application domain and, eventually, customize some of them interacting with the editor.

For instance, the containment constraint between *Road* and *Lamp* in Fig. 2 is modeled by the production:

Roads $\rightarrow$ ROAD $\langle contains \rangle$ LAMP

where *contains* is an empty production with associated a semantic action that verifies the satisfiability of the relationship [6]. ROAD and LAMP are terminal symbols having associated the set of attributes defined in the corresponding classes of the OMT-G model, e.g., *type* for ROAD. Such attributes are used by the *contains* production to verify the spatial constraint. The nonterminal symbol Roads has associated a set of attributes whose value is synthesized from the attribute values of ROAD and LAMP.

The WiSPY tool provides the implementation of semantic actions for a predefined set of constraints. However, as said above, the tool enables users to define their own constraints. In particular, the user can annotate a OMT-G relationship connecting two classes A and B with a boolean condition on the attributes of A and B. As an example, for the OMT-G diagram in Figure 2, a user could define the following illuminance constraint: *the lightning of lamps associated to urban highways is greater than 40SB*[2]. This constraint is named *illuminates* and is defined by the following boolean expression:

(Road.$type$='Highway' $\wedge$ Lamp.$lightning$ >40).

The constraint checker is obtained by giving as input to a compiler-compiler the grammar automatically generated from the OMT-G model. Since WiSPY uses the ANTLR parser generator to perform this task, we represent the XPG grammar into a format compatible with ANTLR. The use of

Fig. 3 shows the grammar constraint checker editor embedded into the uDig interface. In particular, in the right side of the interface (label D), the palette contains all the suitable tools for grammar checking, and the "Select Feature Set" operator is activated. By using this operator, users can select geographic features into the area of interest by using a simple rectangle selection tool on the standard uDig map view. After this operation the geographic data of interest are selected and highlighted in yellow in the map (see label A). In this example, the highlighted polygons represent areas, while the highlighted points represent lamps. In this case, only the selected geographic features are involved in the constraint check process. At the bottom left side of the interface (label B), the details about the selected features

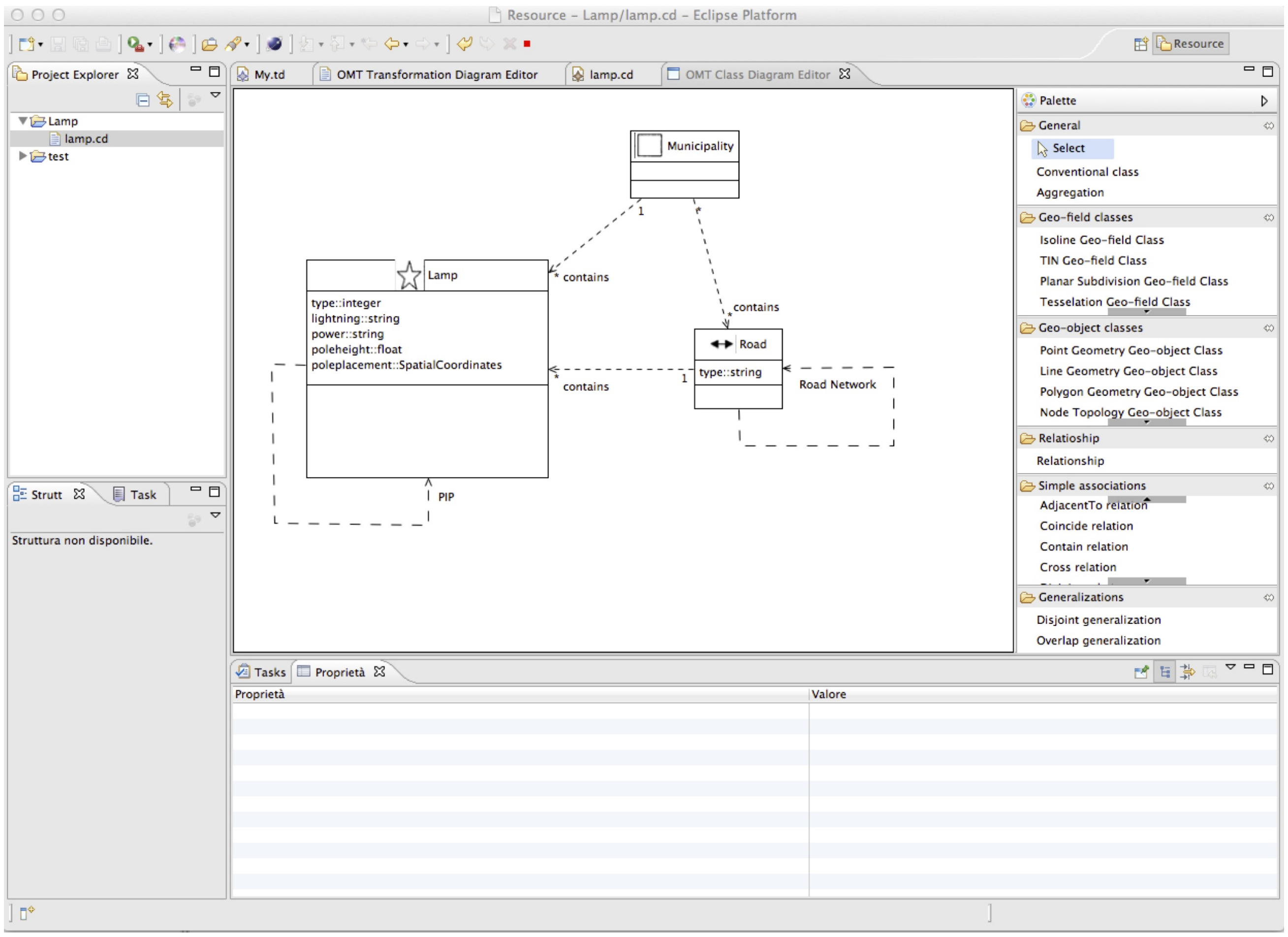


Figure 2: OMT-G interface for specifying the spatial integrity constraints.

are shown. Finally, at the bottom right side of the interface (label C), the output console of the validation process is shown.

4.3 Verification of SICs

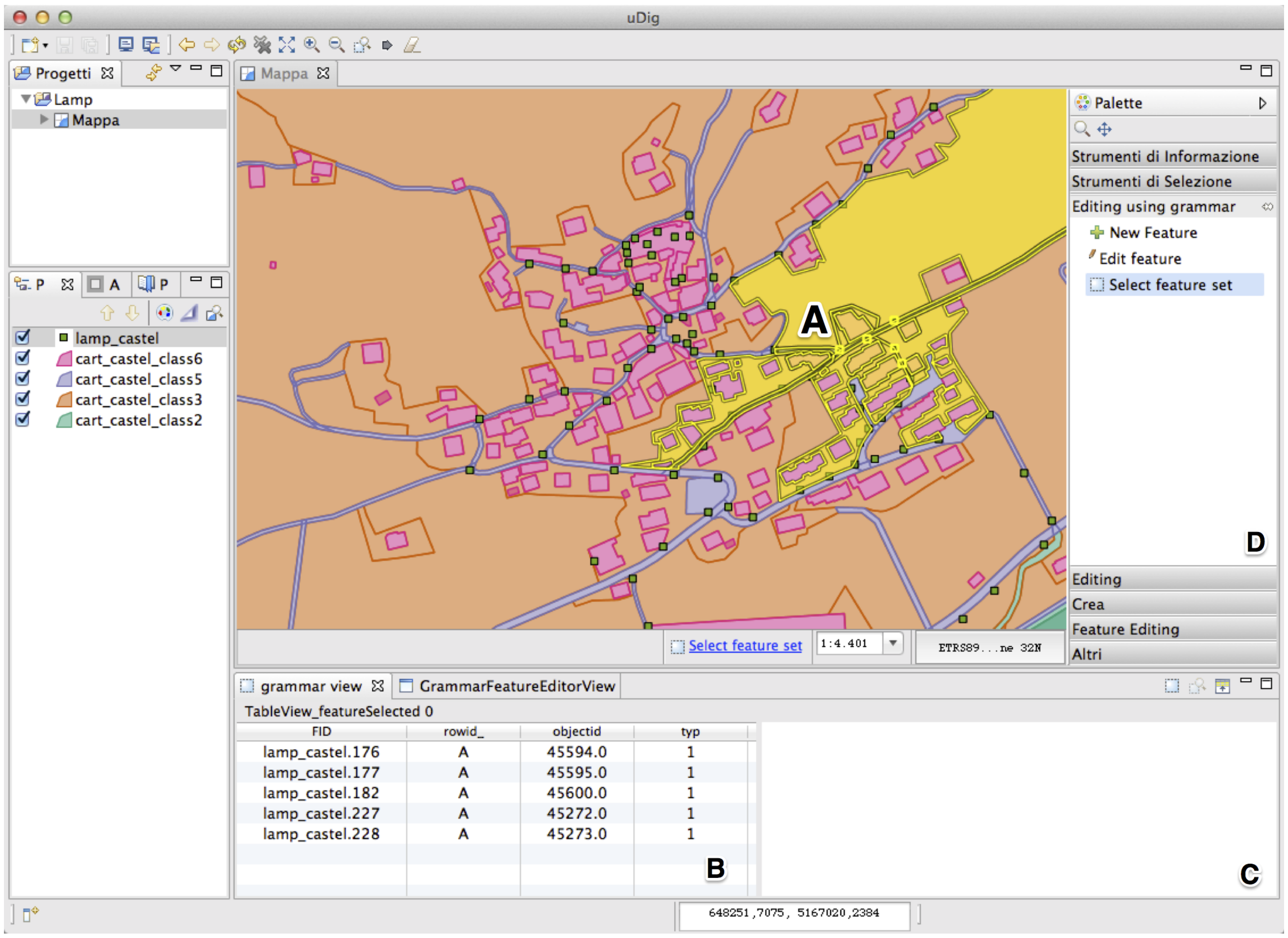
The parser generated with ANTLR is used by WiSPY to validate the input spatial data against the SICs specified in the OMT-G model. In particular, the parser analyzes the spatial objects positioned on a map driven by the relationships specified in the grammar. If the spatial objects violates a SIC then it yields a parse error.

Fig. 4 shows the result of the constraint checking process in the WiSPY interface. In this example, points representing lamps are highlighted by using the "Select Feature Set" operator. Executing the constraint checking, the steps performed by the validation process are listed into the console view, located at the bottom of the interface. If an error occurs, it is reported to the user in the console view.

5 Checking SICs for Public Illumination Plans

The best way to illustrate how to apply our system to real problems is through an example on PIP case study. In this domain, lamps are spatial objects having associated the following information: localization of the lamp (municipality, hamlet, street, GPS coordinates), number of light points for every lamp, lamp type, type of light source, number of light sources for light point, electric power for each light source, year, overall electric power of the lamp, mounting typology (wall or pole), pole type, pole height, circuit and electric power panel, road classification and technical illumination classification. The roads are classified as:

- Category A: highways;
- Category B: high-speed extraurban roads;
- Category C: secondary extraurban roads;
- Category D: urban arterial roads;



TableView_featureSelected 0

FID	rowid_	objectid	typ
lamp_castel.176	A	45594.0	1
lamp_castel.177	A	45595.0	1
lamp_castel.182	A	45600.0	1
lamp_castel.227	A	45272.0	1
lamp_castel.228	A	45273.0	1

Figure 3: WiSPY main window with the additional tools for grammar parsing.

- Category E: urban district roads;
- Category F: local roads.

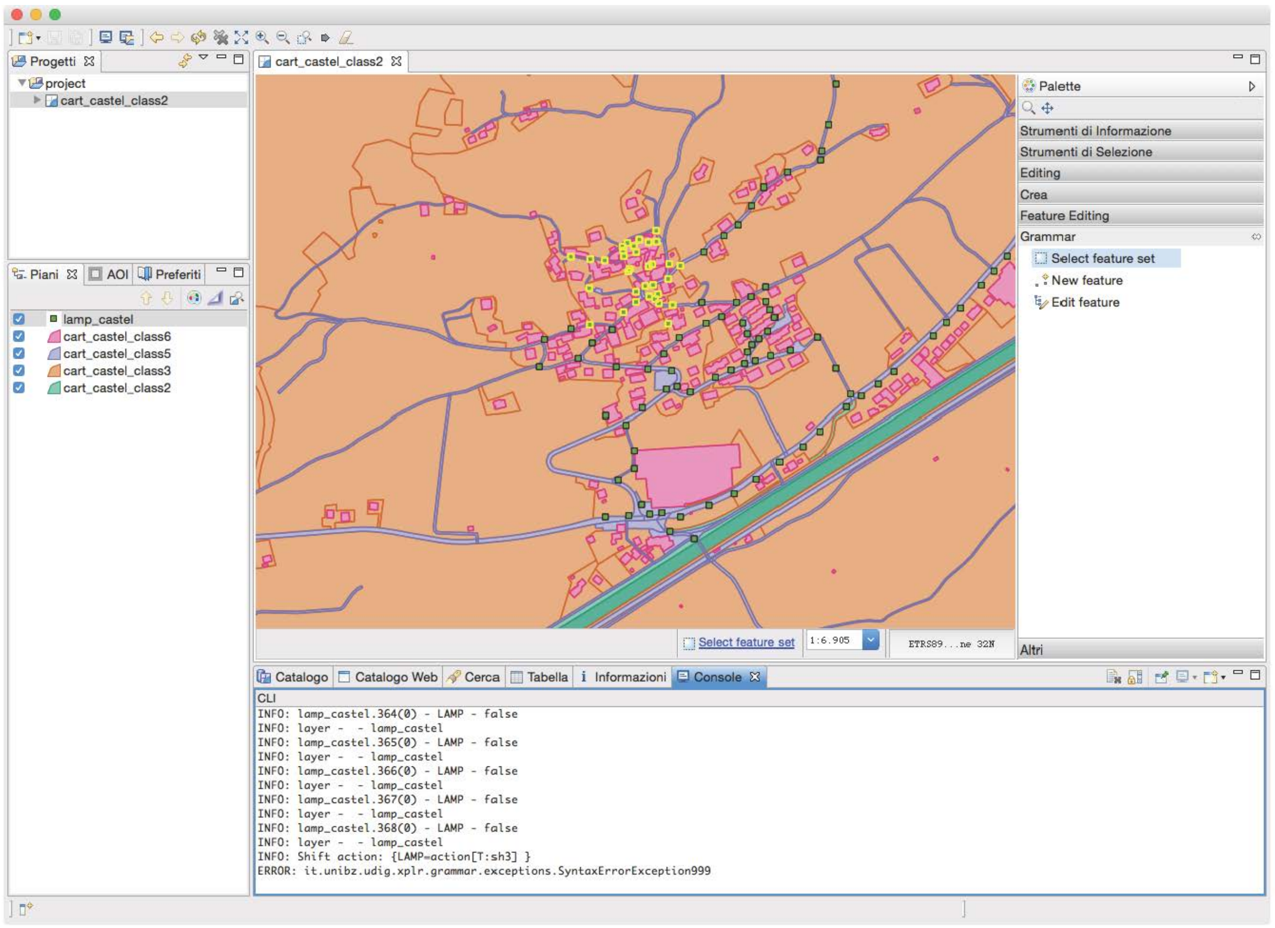
Road illumination varies depending on road classification and on the related technical illumination classification. This classification is the same specified in the UNI EN 13201 European normative, which states that the illumination level is based on the traffic intensity of the road and on the daytime. Therefore the technical illumination classification of a road may vary during the daytime. Since the geographic data used for the prototype are coming from a municipality far away from highways, the classification used in WiSPY is simplified as reported in Tables 1 and 2. The illumination parameters reported in Table 2 refers to:

- $\bar{L}(cd/m^2)$ is the average road surface luminance of a carriageway of a road expressed in candelas per square meter;
- U_o is the overall uniformity of road surface luminance;
- U_l is the longitudinal uniformity of road surface luminance;
- TI is the threshold increment, which measures the loss in percentage of visibility caused by the disability glare of the luminaries of a road lighting installation;
- SB^2 is the surround ratio of illumination of a carriageway of a road
- $\bar{E}$ is the hemispherical illuminance averaged over a road area expressed in lux.

Road Type	Road description	Speed limit (km/h)	Tech. Illum. Class.
C - Secondary extraurban roads	Extraurban road	70-90	ME3a
	Local extraurban roads	50	ME4b
F - Local roads	Local urban roads	30	S3
	Historic town center	-	CE4
F - Pedestrian and bicyle routes	-	-	S3

Table 1: UNI EN 13201 road classification (subset).

In order to illustrate how WiSPY is able to identify violations in the public illumination plan, in the following we



```
CLI
INFO: lamp_castel.364(0) - LAMP - false
INFO: layer -  - lamp_castel
INFO: lamp_castel.365(0) - LAMP - false
INFO: layer -  - lamp_castel
INFO: lamp_castel.366(0) - LAMP - false
INFO: layer -  - lamp_castel
INFO: lamp_castel.367(0) - LAMP - false
INFO: layer -  - lamp_castel
INFO: lamp_castel.368(0) - LAMP - false
INFO: layer -  - lamp_castel
INFO: Shift action: {LAMP=action[T:sh3] }
ERROR: it.unibz.udig.xplr.grammar.exceptions.SyntaxErrorException999
```

Figure 4: WiSPY interface showing the result of the constraint checking process.

Class	Detail type	Detail values
ME3a	Luminance of the road surface of the carriageway	1,0 $\overline{L}$(cd/m^2)
		0,4 U_o
		0,7 U_l
	Disability glare	15 TI in %
	Lighting of surroundings	0,5 SB^2
ME4b	Luminance of the road surface of the carriageway	0,75$\overline{L}$(cd/m^2)
		0,4 U_o
		0,6 U_l
	Disability glare	15 TI in %
	Lighting of surroundings	0,5 SB^2
S3	Horizontal illuminance	7,5$\overline{E}$ in lx
		1,5E_{min} in lx
CE4	Horizontal illuminance	10 $\overline{E}$ in lx
		0,4 U_o

Table 2: Considered UNI EN 13201 technical illumination classes.

sketch the grammar derived from the OMT-G model in Figure 2. In particular, the terminal symbols of the grammar correspond to the classes of the model, i.e., MUNICIPALITY, LAMP, and ROAD, while the productions are generated according to the class relationships specified in the OMT-G model:

1. Map $\rightarrow$ MUNICIPALITY $\langle contains \rangle$ Objects;

2. Objects $\rightarrow$ Lamps $\langle union \rangle$ Roads;

3. Lamps $\rightarrow$ Lamp $\langle pip \rangle$ Lamps

4. $pip \rightarrow \epsilon$
   SemanticAction: {
    if *distance*(Lamp,Lamps)<MIN_LAMP_DIST
      then
        parse.alert(*'CONSTRAINT VIOLATION'*,
         'Lamp'+Lamp.pos+' is too close to other lamps');
   }

5. Lamps $\rightarrow$ Lamp

6. Lamp $\rightarrow$ LAMP

7. Roads $\rightarrow$ Road $\langle touches \rangle$ Roads

8. Roads $\rightarrow$ Road

9. Road $\rightarrow$ ROAD

10. Road $\rightarrow$ ROAD $\langle contains, isIlluminated \rangle$ Lamp

11. $isIlluminated \rightarrow \epsilon$
    SemanticAction: {
    if (ROAD.*type*='C'$\wedge$ Lamp.lightning$\neq$0.5) $\vee$(...)
      then
        parse.alert('CONSTRAINT VIOLATION',
          'Lamp'+Lamp.position +' violates the '+
          'illumination of road ' + ROAD.name);
    }

The productions have associated semantic actions that analyze the values of the attributes associated to the spatial symbols and verify whether the PIP constraints are satisfied. In particular, the first production indicates that a map is composed of a municipality symbol (visually defined in Figure 2 with a polygon) containing within its area other objects. The latter can be Lamps and/or Roads as defined in production 2. The set of lamps in the municipality has to satisfy the constraint associated to the *PIP* relationship in Figure 2, which defines the compatibility constraints among lamps. As an example, the semantic action associated to production 4 checks if the lamps are positioned too close. In this case, a message is shown to the user. Thus, productions 3-6 define the nonterminal Lamps as a set of LAMP positioned within a MUNICIPALITY according to compatibility constraints. Similarly, productions 7-9 define the nonterminal Road as a set of ROAD symbols positioned within a MUNICIPALITY and related through a *touch* relationship. Productions 10 and 11 define the compatibility constraint between roads and lamps according to the classifications reported in Tables 1 and 2. In particular, each lamp is associated to road and has to satify the user-defined constraint *isIlluminated*. Notice that, for readability of productions, we have omitted the semantic actions that synthesize the attributes for the LHS nonterminal from the attributes of the RHS (non)terminals.

The parser automatically generated from the previous grammar is able to analyze the municipality, road, and lamp symbols positioned by the user on a map, as shown in Figure 4, and verify whether the previously described SICs are violated. As an example, when the parser analizes the lamps positioned on a map it applies productions 3 and 4 trying to reduce the LAMP terminal symbols into Lamps nonterminal symbols. If a lamp is too close to a lamp already analyzed (the spatial coordinates of the lamps previously analyzed by the parser are associated to Lamps' nonterminal symbol) then violation message is shown to the user. When a SIC is violated by two or more geographical objects WiSPY shows a message with the information on the objects involved in the violation and the type of violation.

Thw WiSPY approach simplifies the specification and verification of SICs since the geographic application domain can be easily modeled with OMT-G class diagrams, the SICs can be specified as visual relationships between classes and customized using boolean conditions on attribute values, and the constraint checker can be automatically obtained from the annotated OMT-G model. In this way, the user can easily customize/add new SICs and rapidly prototyping new constraint checkers.

6 Related Work

The quality of spatial databases is an open problem in the field of geographic information systems and, in the last few decades, many efforts have been done to deal with implementation and management issues [15, 23]. In the following, we highlight the most important features of these works.

A constraint solver of spatial data based on programming logic has been presented in [1, 2]. The constraint system is able to handle the basic spatial types such as points, lines and polygons as well as the constraints in terms of equalities and inequalities, memberships, metric, topological and structural constraints. The system also provides a suitable theory for managing constraints and a set of transformation rules. The latter handle a special kind of constraints used for consistency checking, enabling an optimized and efficient resolution of spatial constraints.

In [12] a dimension graph representation is used for maintaining the spatial constraints among objects in an Euclidean space. The constraint consistency checking problem is transformed into a graph cycle detection problem on dimension graph.

The process for discovering inconsistencies in geographical dataset described in [19] consists of three steps: error definition, error checking, and error correction. Basically, the first step consists of the execution of some computational geometry algorithms, while the third one is solved by applying the first order calculus predicates.

In [14] a system developed for automatically maintaining topological constraints in a geographic database is presented. This system is based on extending to spatial data the notion of standard integrity maintenance through active databases. Topological relationships, defined by the users, are transformed into SICs, which are stored in the database as production rules. A similar approach is also introduces in [3].

An automated constraint checking procedure has been introduced by Udagepola *et al.* [21] to check constraint violations at compiling time before updating the database. It is based on a data structure called Semantic Spatial Outlier R-Tree (SSRO-Tree).

In [17] Rigaux *et al.* presented Dedale, a constraint-based spatial database system relied on a linear constraints logical model. This system provides both an abstract data model and a user declarative query language based on SQL in order to represent and manipulate geometric data in arbitrary dimension. A different approach which combines relational and constraint data models is used in [10], where a three-tier constraint database architecture is presented. The latter increases the level of abstraction between the physical data and its semantics by introducing an additional layer to the classical relational data model architecture (logical and physical layer), which allows to manage both constraint-based and geometric data representations in the same layer of abstraction, in opposition to the pure constraint databases, where all data are represented in terms of constraints.

The framework presented in [20] allows the definition of hierarchical descriptions of abstract regions. To this aim, the framework exploits attributed grammars which can be translated by a compiler of compiler to a parser for abstract regions. Once generated, the parsers can be used for evaluating whether the incoming regions are consistent with the specified patterns. Basically, the abstract region candidates that were identified by the parsing rules can be evaluated to check if they conform to the definition provided by the user.

On the commercial side, Oracle® Spatial[9] allows spatial constraint checking by using either the PL/SQL language or by defining the constraint within the table procedure. ArcGIS[10] provides users with a button bar where it is possible to visually define simple constraints. More complex constraints have to be implemented by specific languages.

A significant part of the proposed WiSPY tool concerned with the visual definition of spatial constraints. The choice we made for this purpose is using the OMT-G modelling language. Similar to other approaches, it uses some visual formalisms for describing the spatial objects composing the geodatabase and others for connecting the objects specifying the relationships existing among them. We have chosen OMT-G [4] for its capability of explicitly specifying the constraints in associations and attributes [9], which is a limitation of the models extending UML [18], such as Ext. UML [16] and GeoFrame [8]. Moreover, OMT-G seems to be the most simply and user-friendly notation for non-expert constraint designers. Along this line, in [13] Lizardo and Davis presented a tool which provides various consistency checks on the integrity of the defined schema, and includes a function that maps OMT-G geographic conceptual schemas into physical schemas, including the SICs. Although, it seems very similar to our approach, it is based on SQL constraints which considerably limits the power of constraint checking.

[9]`https://docs.oracle.com/cd/E18283_01/appdev.112/e11830/sdo_intro.htm#insertedID0`

[10]`https://sites.google.com/site/ochaimwiki/geodata-preparation-manual/how-to-check-topology-using-arcgis`

7 Conclusions

In this paper we have proposed a system to support users in the automatic verification of SICs in geographic applications by exploiting visual language parsing. We have demonstrated, by implementing the WiSPY tool, that the visual language parsing is suitable for identifying violation in the PIP case study and for solving ambiguities that may arise in their interpretation. We have motivated our choice of having an entirely visual system, and highlighted its advantages. This choice represents the major difference between our proposal and the related work.

Our future work will focus on the extension of the current prototype in a fully functional product. Moreover, we will concentrate our efforts on finding appropriate solutions to present the information provided to the user, feedback and solutions, in a flexible and supportive manner.

References

[1] J. M. Almendros-Jiménez. Constraint logic programming over sets of spatial objects. In *Proceedings of the 2005 ACM SIGPLAN Workshop on Curry and Functional Logic Programming*, WCFLP '05, pages 32–42, New York, NY, USA, 2005. ACM.

[2] J. M. Almendros-Jiménez and A. Corral. Solving constraints on sets of spatial objects. In M. V. Hermenegildo and D. Cabeza, editors, *Practical Aspects of Declarative Languages, 7th International Symposium, PADL 2005, Long Beach, CA, USA, January 10-11, 2005, Proceedings*, volume 3350 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2005.

[3] A. Belussi, E. Bertino, and B. Catania. Manipulating spatial data in constraint databases. In M. Scholl and A. Voisard, editors, *Advances in Spatial Databases, 5th International Symposium, SSD'97, Berlin, Germany, July 15-18, 1997, Proceedings*, volume 1262 of *Lecture Notes in Computer Science*, pages 115–141. Springer, 1997.

[4] K. A. V. Borges, C. A. Davis, and A. H. F. Laender. OMT-G: an object-oriented data model for geographic applications. *GeoInformatica*, 5(3):221–260, 2001.

[5] K. A. V. Borges, A. H. F. Laender, and C. A. Davis. Spatial data integrity constraints in object oriented geographic data modeling. In C. B. Medeiros, editor, *ACM-GIS '99, Proceedings of the 7th International Symposium on Advances in Geographic Information Systems, November 2-6, 1999, Kansas City, USA*, pages 1–6. ACM, 1999.

[6] G. Costagliola, V. Deufemia, and G. Polese. Visual language implementation through standard compiler-compiler techniques. *J. Vis. Lang. Comput.*, 18(2):165–226, 2007.

[7] V. D. Fatto, V. Deufemia, and L. Paolino. Map integrity constraint verification by using visual language parsing. *JDIM*, 6(4):332–341, 2008.

[8] J. L. Filho and C. Iochpe. Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems*, GIS '99, pages 7–13, New York, NY, USA, 1999. ACM.

[9] A. Friis-Christensen, N. Tryfona, and C. S. Jensen. Requirements and research issues in geographic data modeling. In W. G. Aref, editor, *ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, Atlanta, GA, USA, November 9-10, 2001*, pages 2–8. ACM, 2001.

[10] D. Q. Goldin. Taking constraints out of constraint databases. In B. Kuijpers and P. Z. Revesz, editors, *Constraint Databases, Proceedings of the 1st International Symposium on Applications of Constraint Databases, CDB'04, Paris, June 12-13, 2004*, volume 3074 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2004.

[11] S. Johnson. *YACC: Yet Another Compiler Compiler*. Bell Laboratories, Murray Hills, NJ, 1978.

[12] X. Liu, S. Shekhar, and S. Chawla. Consistency checking for euclidean spatial constraints: a dimension graph approach. In *12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2000), 13-15 November 2000, Vancouver, BC, Canada*, page 333. IEEE Computer Society, 2000.

[13] L. E. O. Lizardo and C. A. D. Jr. OMT-G designer: A web tool for modeling geographic databases in OMT-G. In M. Indulska and S. Purao, editors, *Advances in Conceptual Modeling - ER 2014 Workshops, ENMO, MoBiD, MReBA, QMMQ, SeCoGIS, WISM, and ER Demos, Atlanta, GA, USA, October 27-29, 2014. Proceedings*, volume 8823 of *Lecture Notes in Computer Science*, pages 228–233. Springer, 2014.

[14] C. B. Medeiros and M. Cilia. Maintenance of binary topological constraints through active databases. In *Proceedings of the 3rd ACM International Workshop on Advances in Geographic Information Systems, Baltimore, Maryland, December 1-2, 1995, in conjunction with CIKM 1995.*, page 127, 1995.

[15] L. Plümer and G. Gröger. Achieving integrity in geographic information systems maps and nested maps. *GeoInformatica*, 1(4):345–367, 1997.

[16] R. Price, N. Tryfona, and C. S. Jensen. Extended spatiotemporal UML: motivations, requirements and constructs. *J. Database Manag.*, 11(4):14–27, 2000.

[17] P. Rigaux, M. Scholl, L. Segoufin, and S. Grumbach. Building a constraint-based spatial database system: model, languages, and implementation. *Inf. Syst.*, 28(6):563–595, 2003.

[18] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004.

[19] S. Servigne, T. Ubeda, A. Puricelli, and R. Laurini. A methodology for spatial consistency improvement of geographic databases. *GeoInformatica*, 4(1):7–34, 2000.

[20] J. Steinhauer, T. Wiese, C. Freksa, and T. Barkowsky. Recognition of abstract regions in cartographic maps. volume 2205 of *Lecture Notes in Computer Science*, pages 306–321. Springer, 2001.

[21] K. P. Udagepola, L. Xiang, L. H. Wei, and Y. X. Zong. Efficient management of spatial databases by data consistency and integrity constraints. *WSEAS Transactions on Computers*, 5(2):447–454, 2006.

[22] uDig. User-friendly desktop internet gis. `http://udig.refractions.net/`, 2004.

[23] M. F. Worboys. A unified model for spatial and temporal information. *Comput. J.*, 37(1):36–34, 1994.