Supporting Smart Cities Modeling with Graphical and Textual Editors

Francesco Basciani[1], Maria Teresa Rossi[2], and Martina De Sanctis[2]

[1] University of LAquila, Italy
francesco.basciani@univaq.it
[2] Gran Sasso Science Institute, L'Aquila, Italy
{mariateresa.rossi,martina.desanctis}@gssi.it

Abstract. Smart cities are characterized by their complex structures made by different dimensions (e.g., living, mobility) managed by different stakeholders (e.g., public/private administrations) that not always communicate with each other. Indeed, smart cities include a huge number of diverse aspects from different contexts, ranging from technical to societal aspects. Modeling every context is challenging but required. Towards this direction, in this paper, we present a modeling editor for smart cities realized by exploiting Model-Driven Engineering techniques. We started by designing the data analytics context, while leaving the extension to further contexts as future work. The editor shows both a graphical and textual view and is devoted to stakeholders for the design of smart city models in an *intuitive* and *easy* way.

1 Introduction

Smart cities are characterized by their complex structures made by different dimensions (e.g., living, environment, mobility) managed by different stakeholders (e.g., public and private administrations, service providers). Moreover, they are composed by different physical and technological infrastructures, service platforms, IoT devices. Due to the heterogeneity characterizing the smart cities domain, it might be challenging monitoring the functioning and evolution of the smart cities [7]. Moreover, the lack of communication among stakeholders might compromise the possibility of having a global vision of a smart city, thus causing difficulties for the public administration in defining, for instance, new public policies. Yet, due to the ongoing digital revolution [3], smart cities technologies are continuously evolving. This asks for a way to easily model smart cities environments allowing the integration of new technological systems.

The typical challenges of this domain can be handled by exploiting Model-Driven Engineering (MDE) techniques (e.g., [10]), supporting the creation of abstract and understandable models. In this artifact paper, we present a modeling editor for designing smart cities by focusing on the data analytics context.

It aims at supporting stakeholders in the creation of smart city models in an intuitive and easy way. The editor exhibits both a graphical and a textual view, and it fulfills the main requirements for modeling editors. Eventually, we show the usage of the editor and its main features.

2 Background

In this section, we report about modeling editors in different contexts, and we give an overview of the smart cities' domain. We further extract the requirements that, in our opinion, a modeling editor for smart cities should meet.

2.1 Modeling Editors

In the literature, modeling editors are used in different contexts.

In [5] the authors present OntoUML Lightweight Editor (OLED), a modeling environment to build, evaluate, and implement models written in OntoUML, which is an ontologically based language for Ontology-driven Conceptual Modeling. It allows the design of real-world contexts enabling the interoperability between different ecosystems. In the context of Petri Nets, a modeling development work-flow, supporting users from the model design to the final controller code generation is presented in [9]. The editor allows the interactive design and editing of Petri Net models, and it is deployed as a web service. In [4] the authors propose a graphical modeling editor to design accessible media players. Here, the focus is mainly in supporting accessibility through the identification of accessibility requirements used for the development of media players. Another modeling editor is reported in [8], where a tool in the context of Safety Analysis Modeling Language (SAML) is provided. It aims to be a solution to deal with the complexity of software-intensive systems.

As regards the smart cities domain, in [10] the authors present a Domain Specific Language (DSL) to model smart cities systems showing a high number of heterogeneous devices (e.g., sensors, platforms, communication protocols). Here, the DSL is a tool for smart city experts that do not have knowledge in software engineering. Thus, they developed a metamodel, a concrete syntax made by the combination of a textual and a graphic representation of the concepts defined in the metamodel, and a definition of the proposed DSL semantics. However, the authors only rely on the TreeEditor provided by Eclipse Modeling Framework (EMF), without developing a standalone editor by exploiting languages devoted explicitly to this, as we do in this work.

The reported works demonstrate, from one side the relevance of modeling editors in supporting the automatic management of models in diverse contexts. From the other side, they highlight that it does not exist a modeling editor for smart cities devoted to its heterogeneous contexts.

2.2 The Smart Cities Domain

Due to its continuously evolving nature, the smart city concept has not yet been clearly defined. In this work, we build on the definition given in [6], where a smart city is defined as *"an instrumented, interconnected, and intelligent city. Instrumented refers to sources of near-real-time real-world data from both physical and virtual sensors. Interconnected means the integration of those data into an enterprise computing platform and the communication of such information among the various city services. Intelligent refers to the inclusion of complex analytics, modeling, optimization, and visualization in the operational business processes to make better operational decisions."*. Indeed, smart cities are made by different dimensions belonging to different contexts, spanning from mobility to education, passing through the environment. Moreover, each smart city involves different stakeholders, such as those dealing with its management (e.g., public administrations, municipalities), those exploiting its services (e.g., citizens), and those acting as service providers for the city (e.g., transport companies, cultural organizations). The backbone of a smart city is further represented by several physical and technological infrastructures supporting both the management of the city and the monitoring of its functioning. These bring to the availability of a massive amount of data that must be stored, managed, integrated to extract useful information about the smart city. Data analytics has become a hot topic in the smart cities domain since it strongly supports the smart decision-making process performed by the smart governance authority. For instance, in [11] the authors expose the issues related to the presence of massive-scale infrastructures, thus of ever-increasing the amount of data.

2.3 Requirements for Modeling Editors

As inspired from [8], we list some general requirements that we think our modeling editor for smart cities must satisfy, as follows.

R1 *Modeling support* refers to the need of supporting users during the modeling phase (e.g., syntax highlighting auto-completion, code snippets).
R2 *Error detection*, both in the syntax and semantics at modeling time.
R3 *Automation of the model transformation and analysis* by means of model checkers that run into the editor.
R4 *Understandable representation* of the results of the model checkers.
R5 *Project versioning support* to allow the integration of external versioning tools (e.g., svn, git).
R6 *Staging and evolution support* of the realized models.
R7 *Modeling support of a great amount of data* generated by the smart cities infrastructure.

We used the listed requirements as a guideline for the development of a user-friendly smart cities editor to support the stakeholders of smart cities.

3 A Modeling Editor for Smart Cities

In this section, we describe the two artifacts making the modeling editor for smart cities. We started by specifying a subset of the smart city concepts, designed in a *smart city metamodel*, also defining the relations among them. On top of it, we built a *graphical* and *textual* editor to support smart cities modeling. Both editors are based on EMF, which means that in creating a smart city instance, both can act on the same model. This implies that whenever the changes made to the model are saved by one of the two editors, these are automatically reflected in the other editor, keeping the synchronization between the two editors. We highlight here that the provided metamodel, thus the editor, only refer to the data analytics context. Thus, the editor represents a trivial but exhaustive example of the support it would give to stakeholders. Eventually, both the metamodel and the editor are open for future extensions.

3.1 Smart Cities Metamodel

In Fig. 1 we show a portion of the smart city metamodel, specifically the one focusing on data analytics and infrastructures. The SmartCityModel is specified

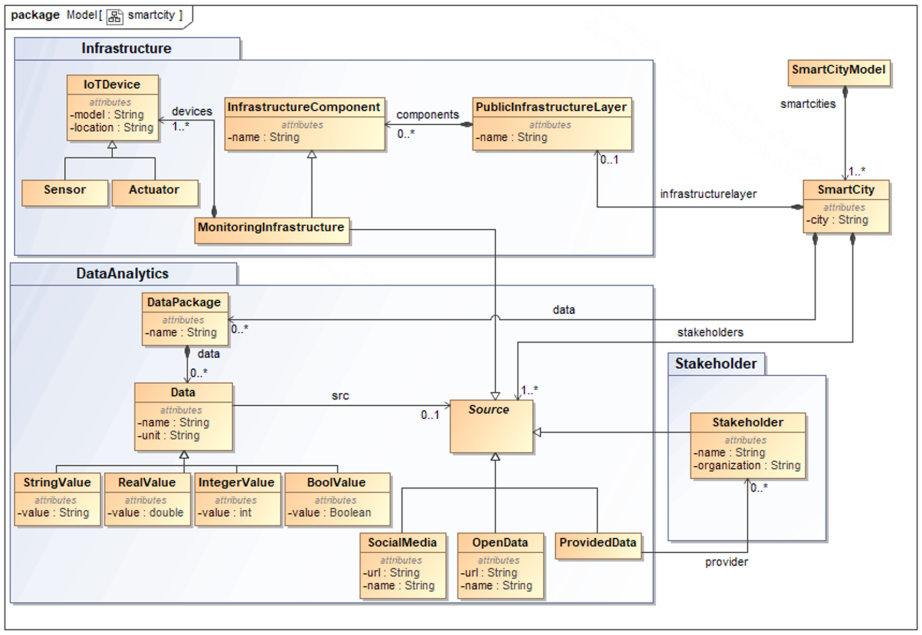


Fig. 1. Smart Cities Metamodel

as a composition of $1 \ldots *$ SmartCity, thus, to model different smart cities. Each SmartCity can be, in turn, composed of several entities, which have been organized in three packages, namely Infrastructure, DataAnalytics, and Stakeholder.

As regards the Infrastructure package, we refer to both the physical and organizational structures and facilities needed for the operation of a smart city. We assume that a smart city can rely on a PublicInfrastructureLayer that can be composed of different InfrastructureComponent, one for each infrastructure set up in the city. For instance, we modeled a MonitoringInfrastructure that can be composed of several IoT devices, modeled as IoTDevice and defined by the attributes *model* and *location*, giving details on the specific device and its physical position. IoTDevice can be further specialized in Sensor and Actuator. The DataAnalytics package defines the concepts required to describe the data related to a smart city. Thus, here we find the generic concept of Data, as part of a DataPackage, that is further specialized in different types of values (i.e., StringValue, RealValue, IntegerValue, BoolValue). Moreover, data may originate from different sources. As inspired by [1], we modeled a Source for each Data that, in turn, can be specialized in different types. More precisely, we consider that we can gain data from SocialMedia, from an OpenData dataset, or that we can obtain ProvidedData from third-parties, e.g., a smart city Stakeholder, for a given reason. Often, in this case, data are provided in a specific format depending on the request reason. Both the SocialMedia and OpenData components are defined by the attribute *url* pointing to the specific resource location of data. Lastly, the Stakeholder package contains the before mentioned Stakeholder component. It allows designers to model every type of smart cities stakeholders (e.g., private and public institutions, companies), which can act or not as data providers.

3.2 Smart Cities Editor

The smart cities metamodel presented in the previous section has been implemented as an EMF/Ecore[2] artifact. Both textual and graphical editors have been realized[3].

The textual editor has been developed by means of Xtext[4], which is an Eclipse project for developing DSLs. Starting from the specification of the grammar of the language, the Xtext framework supports the implementation of a full infrastructure, including parser, linker, type-checker, compiler as well as editing support for Eclipse. The editor provides modelers with typically expected features like syntax highlighting, code completion, and outlines.

The graphical editor is developed by relying on Sirius[5], which is an Eclipse project supporting the development of graphical modeling workbenches by leveraging the Eclipse Modeling technologies, including EMF and the Graphical Modeling Framework (GMF). A typical workbench developed with Sirius is composed of a set of Eclipse editors (diagrams, tables, and trees), which allow the users to create, edit and visualize graphical representations of EMF models. The editors are defined by a metamodel, which defines the complete structure of the modeling workbench, its behavior, and all the editing and navigation tools. The graphical editor comes with a palette containing tools allowing users to create

[3] https://github.com/gssi/MoSC2020
[4] https://www.eclipse.org/Xtext/
[5] https://www.obeodesigner.com/en/product/sirius

new model elements (see Fig. 2). Specifically, the palette contains tools to create *Entity* elements, *Attribute* within entities, and relationships between the same entities (*Relation*). For this work, the editor has a dedicated palette with specific tools w.r.t. the concepts defined in the smart cities metamodel. For each modeled element, in the Eclipse *Property view* it is possible to give a value to its attributes. It is important to remark that the graphical editor prevents problems of inconsistency between the tools in the palette. In other words, it is not possible to create elements that are not those expected from the language metamodel. For instance, the *Attribute* tool in the palette's *Entity* section can be inserted only within *Entity* elements. The editor also allows modelers to hide elements in the canvas to make it easier to view and develop big models.

Modeling tools have been designed also to give early feedback about the specified models. In particular, models are analysed by a set of checks each devoted to the discovery of possible issues. Even though the analysis tools include ready to use checks, it is possible to extend the system by specifying additional checks that modelers might want to add for the particular models at hand.

In particular, we added further custom validation rules in addition to the structural ones provided by the Ecore metamodel for the textual and graphical editor to give the user early feedback as they type and draw. Specifically, we used Xtend[6] programming language for specifying the checks on the textual editor, while for the graphical editor, we used validation expressions written through Acceleo Query Language (AQL)[7]. In addition to these, we also used the Epsilon suite, and specifically Epsilon Object Language (EOL)[8] and the Epsilon Validation Language (EVL)[9], that can be used to specify and evaluate constraints on models of arbitrary metamodels and modeling technologies. EVL also supports interdependencies between constraints (e.g., if constraint A fails, the constraint B cannot be evaluated), customizable error messages to be displayed to the user, and specifications of fixes (in EOL) can be invoked to repair inconsistencies (see an example in Fig. 6).

Morover, the language permits to handle the severity of validation result by means of: (i) **Constraints**: they are used to capture critical errors that invalidate the model; (ii) **Critiques**: they are used to capture non-critical situations that do not invalidate the model, but should nevertheless be addressed by the user to enhance the quality of the model.

4 Evaluation

To show the *usage* of the smart cities modeling editor, we used it to model two medium-sized Italian cities, i.e., Bolzano and L'Aquila, located in different areas of the country (i.e., north, center) and showing similarities and differences, as highlighted in the models. Then, we discuss how the editor meets the beforementioned requirements.

[6] https://www.eclipse.org/xtend/ [7] https://www.eclipse.org/acceleo/documentation/
[8] https://www.eclipse.org/epsilon/doc/eol/ [9] https://www.eclipse.org/epsilon/doc/evl/

Figs. 2 and 3 show the graphical and textual representation of the model for the smart city of L'Aquila, respectively. For the city of L'Aquila, an instance of

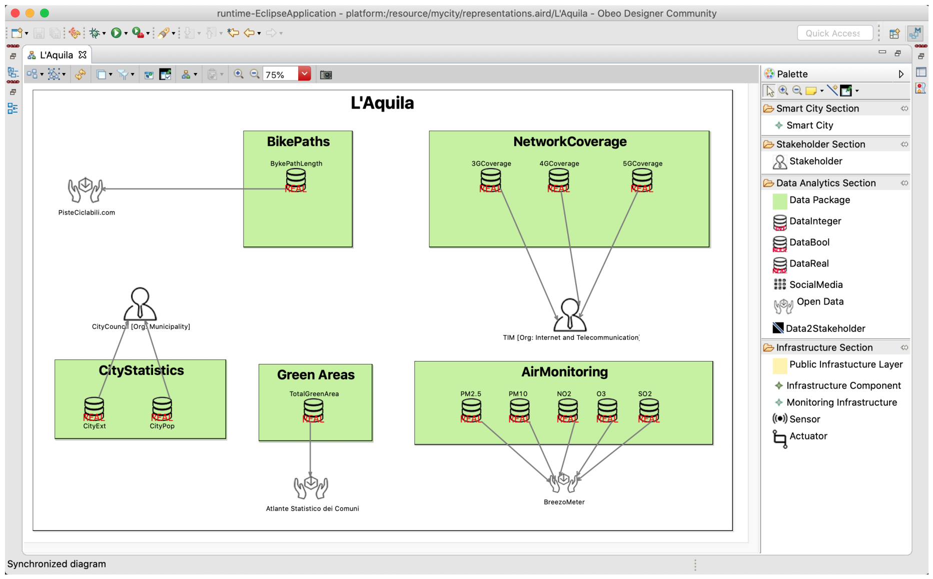


Fig. 2. Graphical representation of the model for the city of L'Aquila.

Stakeholder called TIM (line 5 in Fig. 3) is the provider of the Data composing the DataPackage instance NetworkCoverage (lines 18-22 in Fig. 3). In particular, we modeled three instances of Data, i.e., 3GCoverage, 4GCoverage and 5GCoverage. Moreover, the Stakeholder CityCouncil (line 4 in Fig. 3), modeling the city municipality, is the provider of the Data in the DataPackage CityStatistics (lines 23-26 in Fig. 3). The statistics modeled here, i.e., CityExt and CityPop refer to the city area in km^2 and the total population. We modeled also some OpenData sources (lines 6-8 in Fig. 3) that provide the data composing the three DataPackage instances: AirMonitoring (lines 11-17 in Fig. 3), BikePaths (lines 27-29 in Fig. 3) and GreenAreas (lines 30-32 in Fig. 3). In particular, the service BreezoMeter[10] gives live air pollution, pollen, and fires information of a selected geographical area. PisteCiclabili.com[11] provides the Italian bike paths at the provincial and municipal levels. Regarding the information about the green areas, we gain the TotalGreenArea data by the service Atlante Statistico dei Comuni[12].

In Fig. 4 we report the graphical representation of the smart city of Bolzano model. We can see that, differently from the model for L'Aquila, in the data package NetworkCoverage we can have only the data instances 3GCoverage and 4GCoverage due to the lack of 5G coverage in this city. Regarding the data package GreenAreas, the provider is Open Data Alto Adige[13]. Eventually, in this model, we designed a data package, called SharedVehicles that is composed of the data about the units of SharedBicycles provided by the service Bici Bolzano.

[10] https://breezometer.com/ [11] https://www.piste-ciclabili.com/
[12] http://asc.istat.it/ [13] http://daten.buergernetz.bz.it/

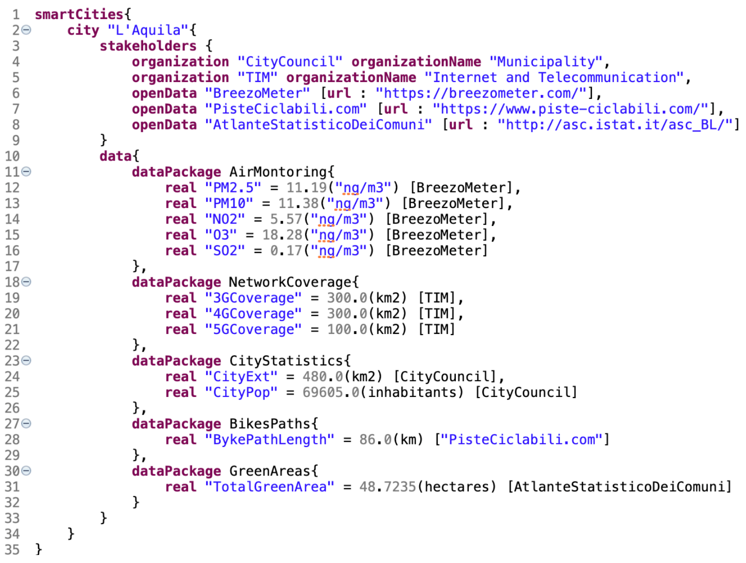

```
smartCities{
    city "L'Aquila"{
        stakeholders {
            organization "CityCouncil" organizationName "Municipality",
            organization "TIM" organizationName "Internet and Telecommunication",
            openData "BreezoMeter" [url : "https://breezometer.com/"],
            openData "PisteCiclabili.com" [url : "https://www.piste-ciclabili.com/"],
            openData "AtlanteStatisticoDeiComuni" [url : "http://asc.istat.it/asc_BL/"]
        }
        data{
            dataPackage AirMontoring{
                real "PM2.5" = 11.19("ng/m3") [BreezoMeter],
                real "PM10" = 11.38("ng/m3") [BreezoMeter],
                real "NO2" = 5.57("ng/m3") [BreezoMeter],
                real "O3" = 18.28("ng/m3") [BreezoMeter],
                real "SO2" = 0.17("ng/m3") [BreezoMeter]
            },
            dataPackage NetworkCoverage{
                real "3GCoverage" = 300.0(km2) [TIM],
                real "4GCoverage" = 300.0(km2) [TIM],
                real "5GCoverage" = 100.0(km2) [TIM]
            },
            dataPackage CityStatistics{
                real "CityExt" = 480.0(km2) [CityCouncil],
                real "CityPop" = 69605.0(inhabitants) [CityCouncil]
            },
            dataPackage BikesPaths{
                real "BykePathLength" = 86.0(km) ["PisteCiclabili.com"]
            },
            dataPackage GreenAreas{
                real "TotalGreenArea" = 48.7235(hectares) [AtlanteStatisticoDeiComuni]
            }
        }
    }
}
```

Fig. 3. Textual representation of the model for the city of L'Aquila.

For the city of L'Aquila, it has not been possible to design such data package because of the lack of sharing vehicle services in the city. The described models show the usage of the editor through an intuitive graphical palette supporting users in modeling the different features of a smart city.

Editor's Evaluation Against Requirements. Here, we map the presented editor w.r.t. the requirements listed in section 2.3.

R1 ***Modeling support*****:** This requirement is explicitly satisfied by the textual editor that provides useful features like syntax highlighting, code completion, and outlines, as shown for instance in Fig. 5.

R2 ***Error detection*****:** As regards the detection of errors at modeling time, in our approach, we can define syntax rules, by exploiting EVL features, as already said in section 3.2. Fig. 6 reports an example of a syntax error detection (i.e., the name of a SmartCity is supposed to begin with an upper case character). Meanwhile, for semantic errors, the editor does not allow violations of the semantics declared in the metamodel, by construction.

R3 ***Automation of the model transformation and analysis*****:** This comes from a combination between EVL for the model analysis and error detection (see **R2**) and EOL for its ability to modify models. E.g., Fig. 6 shows the notification of an error and the possibility of fixing it via the editor, which can apply the performed modification directly in the model.

R4 ***Understandable representation*****:** The results of model checkers, implemented with EVL, are reported in the *Validation* view of the editor, as provided by Eclipse (see Fig. 6).

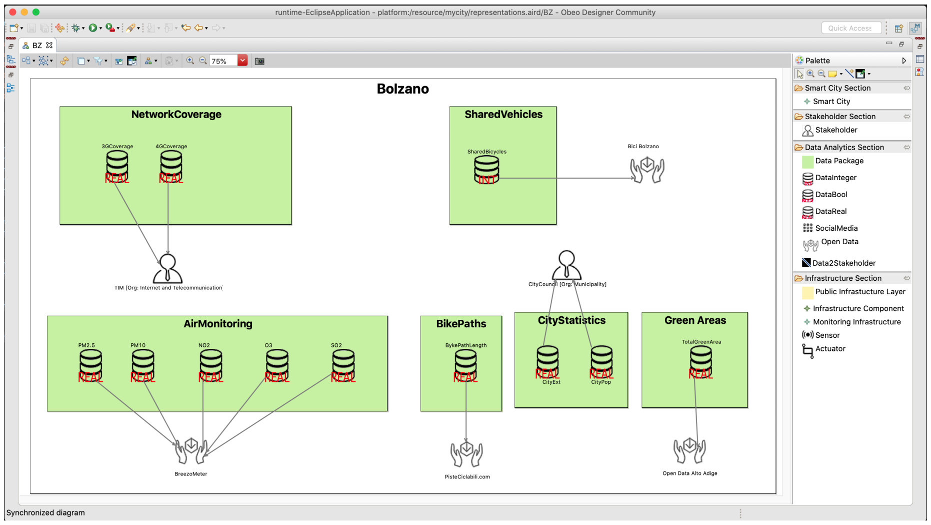
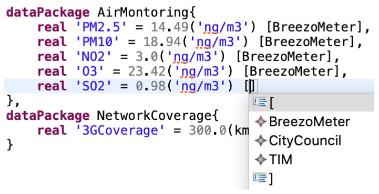
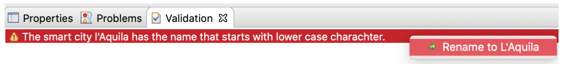


Fig. 4. Graphical representation of the model for the city of Bolzano.

R5 ***Project versioning support*:** This requirement is indirectly satisfied by the use of Eclipse as a development framework. Indeed it already supports the integration of external versioning tools (e.g., *eGit*[14] plugin).
R6 ***Staging and evolution support*:** The evolution is supported by EMF Compare framework[15], integrated with the Eclipse Teams API, which allows for the collaborative work on models using GIT, CVS, and SVN. It provides general support for comparison and merges facility for models, showing, with a generic comparison engine, the differences between two models. It is possible to see and analyze the evolution of a model as all the differences with the starting model are highlighted graphically. EMF Compare shows both the elements on which the changes were made and the type of the change itself (e.g., addition, renaming)[16].
R7 ***Modeling support of the great amount of data*:** The implemented editor provides commands, allowing us to easily hide/show parts of the designed models. This function can improve the understandability of the models when high-numbers of components are designed (e.g., multiple devices, data packages).



```
dataPackage AirMontoring{
    real 'PM2.5' = 14.49('ng/m3') [BreezoMeter],
    real 'PM10' = 18.94('ng/m3') [BreezoMeter],
    real 'NO2' = 3.0('ng/m3') [BreezoMeter],
    real 'O3' = 23.42('ng/m3') [BreezoMeter],
    real 'SO2' = 0.98('ng/m3') []
},
dataPackage NetworkCoverage{
    real '3GCoverage' = 300.0(km
}
```



The smart city l'Aquila has the name that starts with lower case charachter.

Fig. 5. Auto-completion feature. **Fig. 6.** Syntax error at modeling time.

[14] https://www.eclipse.org/egit/ [15] https://www.eclipse.org/emf/compare/overview.html
[16] See an example at https://github.com/gssi/MoSC2020/wiki/EMF-Compare-example.

5 Conclusions and Future Works

In this paper, we presented a modeling editor for designing smart cities, by starting from the infrastructure and data analytics contexts. In the near future, we plan to model further smart cities concepts, thus finalizing the smart city metamodel and the editor to make it ready for being evaluated with real stakeholders. Moreover, to further support stakeholders, we plan to extend the editor such that it shows different model views. Eventually, the presented editor will be part of a smart cities quality assessment system, which is under development.

Acknowledgment

This work was partially supported by the Centre for Urban Informatics and Modelling - National Project - GSSI.

References

1. Berntzen, L., Johannessen, M., El-Gazzar, R.: Smart cities, big data and smart decision-making understanding "big data" in smart city applications. In: Twelfth International Conference on Digital Society and eGovernments (ICDS 2018) (2018)
2. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Addison Wesley (2003)
3. C. Selada, C. Silva and A. L. Almeida INTELI Inteligncia em Inovao, Centro de Inovao: Urban Indicators and the Smart City Agenda (December 2016)
4. Gonzlez-Garca, M., Moreno, L., Martinez, P., Min, R., Abascal, J.: A model-based graphical editor to design accessible media players. JOURNAL OF UNIVERSAL COMPUTER SCIENCE (01 2014)
5. Guerson, J., Sales, T.P., Guizzardi, G., Almeida, J.P.A.: Ontouml lightweight editor: A model-based environment to build, evaluate and implement reference ontologies. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop. pp. 144–147 (2015)
6. Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., Williams, P.: Foundations for smarter cities. IBM Journal of Research and Development **54**(4), 1–16 (2010)
7. Hefnawy, A., Bouras, A., Cherifi, C.: Relevance of lifecycle management to smart city development. International Journal of Product Development **22**, 351 (2018)
8. Lipaczewski, M., Struck, S., Ortmeier, F.: Saml goes eclipse combining model-based safety analysis and high-level editor support. In: 2012 Second International Workshop on Developing Tools as Plug-Ins (TOPI). pp. 67–72 (2012)
9. Pereira, F., Moutinho, F., Ribeiro, J., Gomes, L.: Web based iopt petri net editor with an extensible plugin architecture to support generic net operations. In: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society. pp. 6151–6156 (2012)
10. Rosique, F., Losilla, F., Pastor, J.A.: A domain specific language for smart cities. In: 4th International Electronic Conference on Sensors and Applications. vol. 2, p. 148 (2018)
11. Schleicher, J.M., Vgler, M., Dustdar, S., Inzinger, C.: Enabling a smart city application ecosystem: Requirements and architectural aspects. IEEE Internet Computing **20**(2), 58–65 (2016)