



MIKADO: a smart city KPIs assessment modeling framework

Martina De Sanctis^{1,2} · Ludovico Iovino^{1,2} · Maria Teresa Rossi^{1,2} · Manuel Wimmer³

Received: 15 October 2020 / Revised: 30 April 2021 / Accepted: 28 June 2021 / Published online: 4 August 2021
© The Author(s) 2021

Abstract

Smart decision making plays a central role for smart city governance. It exploits data analytics approaches applied to collected data, for supporting smart cities stakeholders in understanding and effectively managing a smart city. Smart governance is performed through the management of key performance indicators (KPIs), reflecting the degree of smartness and sustainability of smart cities. Even though KPIs are gaining relevance, e.g., at European level, the existing tools for their calculation are still limited. They mainly consist in dashboards and online spreadsheets that are rigid, thus making the KPIs evolution and customization a tedious and error-prone process. In this paper, we exploit model-driven engineering (MDE) techniques, through metamodel-based domain-specific languages (DSLs), to build a framework called *MIKADO* for the automatic assessment of KPIs over smart cities. In particular, the approach provides support for both: (i) domain experts, by the definition of a textual DSL for an intuitive KPIs modeling process and (ii) smart cities stakeholders, by the definition of graphical editors for smart cities modeling. Moreover, dynamic dashboards are generated to support an intuitive visualization and interpretation of the KPIs assessed by our KPIs evaluation engine. We provide evaluation results by showing a demonstration case as well as studying the scalability of the KPIs evaluation engine and the general usability of the approach with encouraging results. Moreover, the approach is open and extensible to further manage comparison among smart cities, simulations, and KPIs interrelations.

Keywords Smart Cities · MDE · KPI · DSL · Smart Governance

1 Introduction

Despite the diverse definitions of smart cities [1], they all target the achievement of a sustainable economic, societal, and environmental development, while enhancing the quality of living for their citizens. In this context, *smart decision*

making for sure plays a central role [2]. It exploits *data analytics* approaches applied to collected data, coming from heterogeneous data sources and providers, with the aim of supporting smart cities stakeholders in understanding and effectively managing a smart city. However, decision making for smart cities is challenging due to their complex nature. Indeed, smart cities are characterized by different dimensions (e.g., mobility, education, environment), each managed by different stakeholders (e.g., public administrations, private institutions) who not always communicate with each other. This makes it difficult for public administrations to have a complete overview of the city.

Smart Governance aims to overcome this limitation. It concerns the use of technology in processing information and decision making enabling open, transparent and participatory governments [3], by supporting the knowledge sharing among the involved actors. The smart governance is performed through the management of *key performance indicators* (KPIs) [4] representing raw set of values that can provide information about relevant measures that are of interest for understanding the progress of a smart city.

Communicated by Zhenjiang Hu.

✉ Ludovico Iovino
ludovico.iovino@gssi.it

Martina De Sanctis
martina.desanctis@gssi.it

Maria Teresa Rossi
mariateresa.rossi@gssi.it

Manuel Wimmer
manuel.wimmer@jku.at

¹ Gran Sasso Science Institute, L'Aquila, Italy

² National Interuniversity Consortium for Informatics (CINI), Rome, Italy

³ CDL-MINT, Department of Business Informatics – Software Engineering, JKU Linz, Austria

KPIs are gaining relevance at European level. Indeed, the European Commission published an agenda [5] containing the so-called *sustainable development goals* (SDGs) to be achieved in 2030 to promote a smart,¹ sustainable, and inclusive growth of European cities. To this aim, on top of the SDGs, the International Telecommunication Union (ITU) drafted a list of all the KPIs for smart sustainable cities (SSCs), along with its collection methodology [6]. Other initiatives and international projects (e.g., [7,8]) also targeted the definition of smart cities KPIs to capture the performance of a city in multiple dimensions and to support a transparent monitoring and the comparability among smart cities.

Information and communications technologies (ICT) can help in managing different aspects of complex systems, such as smart cities (e.g., [9]). In particular, model-driven engineering (MDE) [10] techniques are widely used to represent complex systems through abstract models. Examples exist also in the smart city context (e.g., see [11,12]). Several European projects promoting KPIs definition and monitoring have been funded (e.g., see [7,13]). However, there is still an important cornerstone missing, namely a comprehensive methodology supporting (i) systematic and uniform modeling of smart cities and KPIs, (ii) automatic KPIs measurement on top of candidate smart cities, and (iii) intuitive representation and visualization of assessed KPIs. This lack of comprehensiveness represents an obstacle in the management of smart cities. It makes the knowledge sharing difficult which, in turn, negatively affects the smart city decision-making process. This is mainly due to different challenges that can be observed in real scenarios. KPIs reflect the degree of smartness and sustainability which, however, differs among different cities, thus implying the need for KPIs *customization*. Moreover, *KPIs evolve over time* [14]. New KPIs can be defined or existing ones can be implemented in slightly different manners. However, the currently available frameworks (e.g., online spreadsheets, Excel) for the KPIs calculation are still far from being flexible enough.² On the contrary, KPIs definition models are embedded in the frameworks allowing users to only get the results of their measurement. As a consequence, the KPIs evolution management is difficult to handle and it represents an error-prone task [15].

With these premises, we argue that it is necessary to develop a systematic methodology allowing smart city stakeholders to *define*, *measure* and *visualize* the KPIs of interest for their cities in order to efficiently assisting the decision-making process of the smart cities management. To tackle this challenge, we propose in this paper *MIKADO*—a Smart City KPIs Assessment Modeling Framework supporting (i) the uniform modeling of both smart cities and the KPIs, (ii) the

automatic calculation of KPIs, and (iii) graphical visualization of assessed KPIs by means of dynamic dashboards. The resulting approach provides a standard, but at the same time, customizable process for smart cities governance administrators. We aim at exploiting MDE techniques to provide *domain-specific languages* (DSLs) [16] as tools specifically devoted to the domain experts for the modeling of smart cities and corresponding KPIs, while delegating the KPIs measurement, with its complexity, to an *evaluation engine* implementing the KPIs calculations. The choice of the name *MIKADO* is related to the famous game in which *MIKADO* is the name for the most valuable stick.³ The connection with our approach lies in the fact that it can help to get the most value out of a bunch of “*data sticks*.”

The presented approach builds on top of previous work. In [17], we presented the flexible architecture of the tool, identifying both required and optional components and functionalities needed for realizing the automatic KPIs assessment. We also showed flexibility points allowing for different specification of the architecture (e.g., standalone, hybrid, online) through diverse technologies. Precursor work on the modeling editor components, showing both graphical and textual views, has been presented in [18]. The extensions of this work w.r.t. [17,18] are manifold. Firstly, we show how the evaluation engine has been implemented with a complete model-based approach; moreover, we show a new extension of our framework that exploits model-driven dashboards devoted to the visualization of the KPIs model once it is actualized with the results of the evaluation. Secondly, we further improved the *scalability* of our evaluation engine w.r.t. [17], by performing relevant code refactoring in the framework that reduced the computation time about 50%. We evaluated the approach with several smart cities also varying the KPIs definition in order to see how the framework implementation scales with real-world exemplary models. Moreover, we evaluated the *understandability* of the presented DSL for specifying KPIs, as the main prerequisite for the *usability* of the approach, by involving smart cities and/or KPIs experts. The results are positive and encourage to further improve the presented approach with a self-contained tool support. The process, methodology, and the components treated in this paper are core of the assessment process and they are needed to enable the architectural framework described in [17].

The rest of the paper is structured as follows: Sect. 2 describes the role of KPIs in smart decision making. In Sect. 3, we describe the overall approach for the automatic KPIs assessment and we expose the artifacts composing the proposed approach. Subsequently, in Sect. 4 we show the feasibility of the approach by applying it on a real-world case study. In Sect. 5, two evaluations of the proposed approach

¹ <https://sustainabledevelopment.un.org/sdgs>.

² Key performance indicators in Power Pivot at <https://bit.ly/37EFR9r>.

³ [https://en.wikipedia.org/wiki/Mikado_\(game\)](https://en.wikipedia.org/wiki/Mikado_(game)).

are reported. In Sect. 6, related work is reported, and finally, Sect. 7 concludes the paper and proposes future work.

2 Key performance indicators for smart decision making

KPIs [4] support smart governance managers in continuously evaluating smart cities by measuring their sustainability and smartness, thus significantly affecting the smart city decision-making processes. The aforementioned SDGs [5] and the KPIs released by the ITU [6] highlight the relevance that KPIs are gaining worldwide attention. In general, KPIs are defined as visual measures of performance, supported by a specific calculated field. “A KPI is then designed to help users quickly evaluate the current value and status of a metric against a defined target” is what Microsoft research reports online [19]. In the transition of cities to smart cities, KPIs are used as measurement that facilitate monitoring and help to predict and evaluate the transition process. In particular, the ITU KPIs [6] are chosen as potentially applicable to all cities, thus essentially representing *general guidelines* such that smart cities managers can *interpret* and *adapt* them to their managed smart cities.

Specifically, KPIs are elicited and defined by standardization bodies, such as ITU [6], along with their *collection methodology*, standard *definitions* and *formulae*. However, multiple KPIs sources exist. For instance, in this work we also consider CITYkeys [7] and DigitalAQ [8] (see Table 2 in Sect. 4.1)) for completeness purposes. To give a concrete example, the KPI called *Air Pollution (AP)* measures the air quality based on the values reported for specific pollutants. In particular, it is based on the *Air Quality Indexes (AQI)* measured from the registered air concentration of particulate matter (*PM*), nitrogen dioxide (NO_2), sulfur dioxide (SO_2) and ozone (O_3), which are calculated with respect to given *legal limits* established by the law (see formula (3) in Sect. 4.1). The pollutant showing the highest *AQI* (see formula (4) in Sect. 4.1) determines the *AP* KPI that is evaluated w.r.t. five evaluation classes, namely excellent, good, discrete, bad and terrible (see formula (5) in Sect. 4.1).

The complexity of smart cities, which actually are *systems of systems*, makes the smart decision making and the KPIs assessment challenging tasks. More precisely, *KPIs continuously evolve* [14] w.r.t. the evolution of cities’ needs and context. The selection of KPIs is driven by aspects that can change in time (e.g., data availability). This is also due to the *digital revolution* [20] which is taking place in these years. For instance, we can think of how the advent of electrical vehicles has changed the urban scenario, by carrying out new needs (e.g., finding the optimal positioning of car chargers) emerged only in the last few years. Another example may be related to the KPIs involving the network coverage (e.g.,

wireless broadband coverage KPI). After the advent of 5G, the formula for this KPI has been adapted, by including the 5G network coverage, in addition to 4G and broadband connections.

Moreover, smart cities may differ in several aspects, based on their stage of economic development, population growth, available services. Geographical implications must also be considered. Every country has different conditions and KPIs relevance can vary depending on the spatial granularity (e.g., small, medium and metropolitan cities). For instance, sustainable mobility is more developed in metropolitan cities than in small- and medium-sized cities.⁴ As a consequence, *different smart cities can be interested in specific KPIs*. This aspect strongly implies the need for *KPIs customization*, where, for customization we mean the selection of appropriate KPIs to be evaluated on a given smart city. In other words, the selection of KPIs must be driven by the subject smart city and its peculiarities, notwithstanding that KPIs definitions and formulae constantly adhere to those provided by the standardization bodies.

2.1 KPIs assessment frameworks for smart cities: process perspective

With these premises and by inspecting several smart city projects dealing with the performance evaluation in smart cities (e.g., [6–8]), we extracted the basic process used for the realization of frameworks for the automatic assessment of KPIs as illustrated in Fig. 1. Basically, it envisions five main phases referring to the frameworks *development* and *operation* stages. During the *analysis* phase, one of the main activity consists in identifying and defining the KPIs. This activity is fundamental in order to select the right indicators, and usually, it has to be validated with the aid of stakeholders. In this phase also data collection procedures have to be applied for the common and transparent monitoring in a way that the selected data and KPIs can be applied across multiple cities subjects of the evaluation. This leads again to the point that KPIs should be tailored for the city. When the KPIs are identified, the *implementation* phase can start and the framework supporting the assessment can be developed. Once in *preparation*, the assessment framework is ready to use. In this phase, input data can be collected, with the intent of inserting data related to both existing smart cities or future projects to be evaluated. At the *execution* phase, the automatic measurement of KPIs can be performed in order to get quality assessment results about the subject of the evaluation. Eventually, at the *visualization* phase, results can be provided in an accessible way (e.g., tables, graphics), such that to make them easier to interpret and understand.

⁴ Polis 4.0 - Smart City Index 2018 <https://bit.ly/3xnyHEF>.

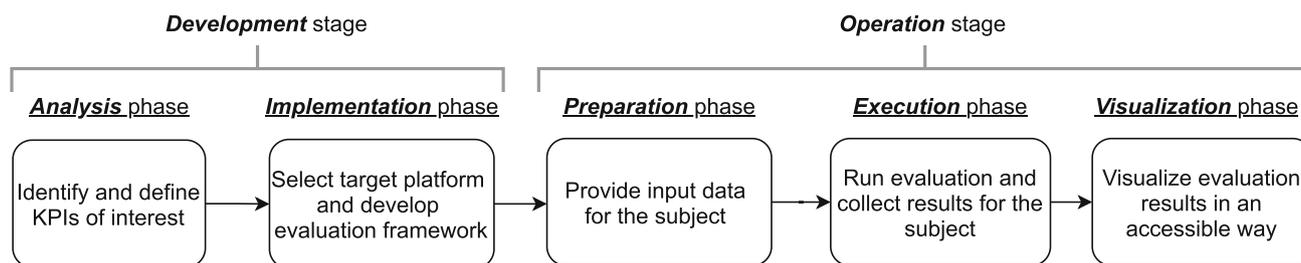


Fig. 1 Basic process for the development of frameworks for the KPIs assessment in smart cities

2.2 KPIs assessment frameworks for smart cities: requirements perspective

Analyzing different smart city projects dealing with the performance evaluation of smart cities further allowed us to extract a set of relevant requirements that a framework for the automatic assessment of KPIs must satisfy, in order to effectively and efficiently support decision-making processes. In particular, we noticed that these projects can be essentially grouped into three main categories, based on the type of implementation strategy and/or target platform they exploit, namely (1) *manual* approaches, where essentially the KPIs definition, data collection and KPIs measurement are manually performed; (2) *Spreadsheet*-based approaches mainly relying on the use of Excel spreadsheets and the Power Pivot feature.⁵ In these approaches, an external data source (e.g., a MS SQL database) is configured to host the required data. These data are then given as input to the KPIs formulae defined in Excel that will actualize the values and present the results. (3) *Web-based platforms* (e.g., [7]) for those approaches supported by an online platform exposing defined KPIs whose calculation is offered as platform functionalities. However, each of these categories shows some drawbacks and limitations, which helped us to derive the following requirements.

2.2.1 Automation

Some of the inspected projects perform a manual evaluation of the input subjects, where the obtained results are then filled into reports to be shown and discussed with the other stakeholders. Of course, even if this manual process can be precise, it is not scalable since it is not automated and when the data grows, the evaluation can suffer of procedural delays. For this reason, the first requirement we argue is that the assessment process has to be **automated**. Moreover, if an automated evaluation is available, the system can easily give the evaluation results in multiple output formats, e.g., Excel, textual files, CSV, in order to enable further elaboration.

⁵ <https://bit.ly/3dT1zwV>.

2.2.2 Separation of concerns

Another issue we observed is the lack of separation of concerns (SoC), especially in Excel spreadsheets. It is totally delegated to the user composing the spreadsheet to keep, for instance, separate the smart city modeling from the KPIs definition and formulae. As a consequence, if the spreadsheet is not well structured, changing the subject of the evaluation can lead to copy and paste activities, with all the related problems. In addition, often KPIs definition models are embedded in the frameworks allowing users to only get the results of their measurement, thus changes are not even allowed. SoC is also about the competences of the concepts related to the KPIs definition and application, that are two different fields.

2.2.3 Domain-specificity

Spreadsheets show several limitations [21] from this point of view. Although providing multiple functionalities supporting mathematical calculations, data analysis and modeling, they have not been conceived for the definition and measurement of KPIs. What we observed in spreadsheet-based approaches is that the definition and calculation of KPIs are highly coupled, therefore KPIs experts are forced to be aware of the specific language underlying the framework (e.g., macros and procedures in Excel). They must learn and be trained to compose formulae in Excel (or to build programs), which can be a tedious and time consuming task, if we consider that Excel formulae can be verbose and complex. For this reason, we argue that offering a domain-specific language would be convenient, also considering the SoC principle and the stakeholders distribution in a smart governance team.

2.2.4 Usability

Notably manual and spreadsheet-based approaches exhibit usability issues. Manual approaches are definitely error-prone due to their nature. With spreadsheets, instead, usability issues arise, for instance, when a connected external source, such as a database, is used for injecting the data needed to calculate the KPIs about the subject. In this case,

also the database should offer a user-friendly UI to fill in the data. This might mean to spend other resources to build applications for data entry, with the result of having two separate systems, with all the related issues of possible inconsistencies [22]. For instance, if the database schema of the connected source changes, new inconsistencies can impact the formulae in the spreadsheet. For this reason, we highlight that the KPIs assessment framework has to offer usable interfaces or languages to specify required data and KPIs formulae.

2.2.5 Openness

As previously discussed, openness is mandatory to manage the KPIs evolution and customization. Spreadsheets exhibits a good degree of openness, in general. Indeed, if KPIs formulae need to be changed or customized, or evolve over time, they can support that, with of course the limits of the target platform (as for instance the limitation on the number of records that can be stored without degrading). To the contrary, Cloud or Web-based platforms are not usually released as open, in order to customize and extend the KPIs definition. In fact, they are often conceived as multi-tenant platforms and this often implies that customization is not supported, being out of the business plan. In other words, to accommodate multiple clients, Web-based platforms tend to be abstract, to cover as much as possible many clients at least to a certain extent. Single-user customization is not convenient in terms of costs and strategies. Consequently, the used KPIs models cannot be extended or customized to accommodate specific smart cities requirements. As a consequence, we argue that KPIs assessment frameworks should be evolutionary and open, meaning that KPIs formulae should be inspectable and modifiable.

2.2.6 Graphical

More advanced platforms can be built in order to support the process. Web-based platforms usually offer very powerful visualization tools showing the results of the KPIs evaluation, although they are not released as open. We argue that supporting a graphical visualization of KPIs evaluated over a given subject is quite relevant for the comprehensibility of the analysis especially for non-experts stakeholders and to ease the knowledge sharing among stakeholders.

2.2.7 Flexibility

Web-based platforms, in general, show the lack of active engagement of domain experts involved in designing, operating, and controlling activities. In addition, KPIs definition models are embedded in the frameworks allowing users to only get the results of their measurement. In fact, most of

them offer pre-packaged KPIs definitions, that the user can select in order to get the results for the subject. They also provide very intuitive forms for the data input. However, these platforms focus mainly on data visualization without offering the flexibility needed by an expert, for instance, for evolution or customization purposes. As an example, if a new activity is needed in order to process the output of the evaluation, we can rely only on the output format of the assessment.

2.2.8 Scalability

Last but not least, scalability is a mandatory requirement to guarantee the effectiveness of a running KPIs assessment approach that is applied over multiple and complex smart cities in order to measure hundreds of KPIs. Furthermore, in the smart governance context, simulation can be used to artificially manipulate smart cities data in order to see how KPIs results are affected. For this reason, scalability is an important issue, especially in a simulation environment, since KPIs evaluation should be performed in a timely manner. However, manual and spreadsheet-based approaches tend to show scalability issues when the size of the subject cities and/or the number and complexity of KPIs increase. Web-based platforms clearly show a better degree of scalability with the novel front-end development technologies and libraries, from one side, and back-end and persistence engines technologies, from the other side.

We sum up the needed requirements and how the currently available frameworks support (✓), partially support (~) or not support (-) them, in Table 1. In detail, the automation of the KPIs assessment process is fundamental to support automatic evaluation. It should further exhibit SoC in order to keep data and KPIs definition separated, also to better cope with the different stakeholders involved in the process. Domain specificity is a way to support usability too. Stakeholders should not be forced to use and learn technicalities in order to use the evaluation framework, but they have to be trained and supported to use constructs of the domain they already know. Usability of the framework is an important requirement. If the user has to learn a new language or technology to implement the assessment measurements, with verbose syntaxes or even programming languages, might not be acceptable. The use of DSLs may help to reduce this problem, since the domain specifics may support the learning with less training. Openness means that the framework should allow to inspect the KPIs definition, extend them, select the ones needed for specific requirements. Flexibility is about the flexibility of the framework to support different candidates, subjects of small and large scale. For instance, evaluations might be performed on entire cities, projects or proposals. Scalability means that the evaluation should scale also for large projects without suffering of performance degradation.

Table 1 Evaluation of KPIs assessment frameworks

Approach	Requirements							
	Automation	SoC	Domain specificity	Usability	Openness	Graphical	Flexibility	Scalability
Manual	–	✓	–	–	–	✓	~	–
Spreadsheets	✓	~	–	–	✓	✓	~	–
Web platforms	✓	✓	✓	✓	–	✓	–	✓
<i>MIKADO</i>	✓	✓	✓	✓	✓	✓	✓	✓

In conclusion, even though existing approaches are tool supported, we claim that an approach for the automatic KPIs assessment, relying on uniform but customizable models still does not exist. Due to these challenges and limitations, faced in the requirements listed above, we propose the *MIKADO* framework for the automatic KPIs assessment that is based on the uniform modeling of both smart cities and the KPIs selected to evaluate them. The approach supports the KPIs customization to different cities and requirements and the KPIs evolution over time, by exploiting MDE techniques. It is inspired by software quality assessment practices (e.g., [23]), and it basically enables a round-trip process for the performance analysis of smart cities and the results interpretation, assisting smart cities administrators to perform the decision-making process. In the next sections, we are demonstrating how *MIKADO* fulfills all the identified requirements, with a demonstration case and an evaluation based on different smart cities and KPIs definitions. In summary, as we will present next, several properties are realized by design such as automation, SoC, domain specificity, openness, and graphical (cf. Sect. 3). Flexibility is shown by a demonstration case (cf. Sect. 4), and scalability and usability are shown by experiments as well as by an expert survey (cf. Sect. 5).

3 The proposed KPIs assessment approach

In this section, we start by giving an overview of the proposed *MIKADO* approach, by describing the methodology we envisage for enabling an automatic KPIs calculation on top of smart cities under evaluation. We also sketch the main tools supporting the methodology and the relations among them. Then, we introduce the two main artefacts, namely the Smart City Metamodel and the KPIs Metamodel as well as the evaluation engine for calculating the KPIs.

All the described artifacts of our approach have been implemented and are openly available.⁶

3.1 MIKADO at a glance

Figure 2 gives an overview of the proposed approach. The box *modeling* highlights the design tasks we envisage and represents the scope of this work. In particular, it includes the modeling tool of the smart cities (*Smart City Modeling*) and the other one for the definition of KPIs (*KPIs modeling*). The smart city modeling is supported by a graphical and textual editor for the smart city design, implemented in [18], whereas the KPIs modeling can be performed through the use of a textual syntax (see icons on top of the modeling boxes in Fig. 2). These components have been realized after we have analyzed the smart city domain, its concepts and the relation among them. Furthermore, we performed an investigation about how KPIs can be measured, i.e., what type of calculations and data they require. With the proposed approach, we are able to define a *unique KPIs definition model*, valid for different smart cities, by using the KPIs modeling tool. This does not mean that the KPIs model is static. On the contrary, it is unique but not universal because its structure may change over time and the number of KPIs may change as well across the different smart cities, to manage both the KPIs *evolution* and *customization* needs.

Given a *Smart City Candidate*, its corresponding *smart city model* and the *KPIs definition model* conforming to the metamodels, which we will describe in the following, these will be used as input for an *Evaluation Engine* that will interpret and calculate the modeled KPIs for the candidate city. The latter component is part of the *Computation* box in Fig. 2. The evaluation engine produces a list of KPIs, identified in the KPIs definition model, and reports concrete values in the *evaluated KPIs model*. This model will be transformed into a KPIs dashboard through a code generation process and through a *Pretty Printing* component, which produces a detailed log of the evaluation. The log supports a fast feedback and debugging process. We grouped these instruments in a *Reporting* box in Fig. 2. The generated dashboards will represent a relevant instrument to guide the *decision-making* process performed by *Smart Governance Systems* for the candidate smart city. For instance, it supports the smart

⁶ <https://github.com/gssi/SmartCityModeling>.

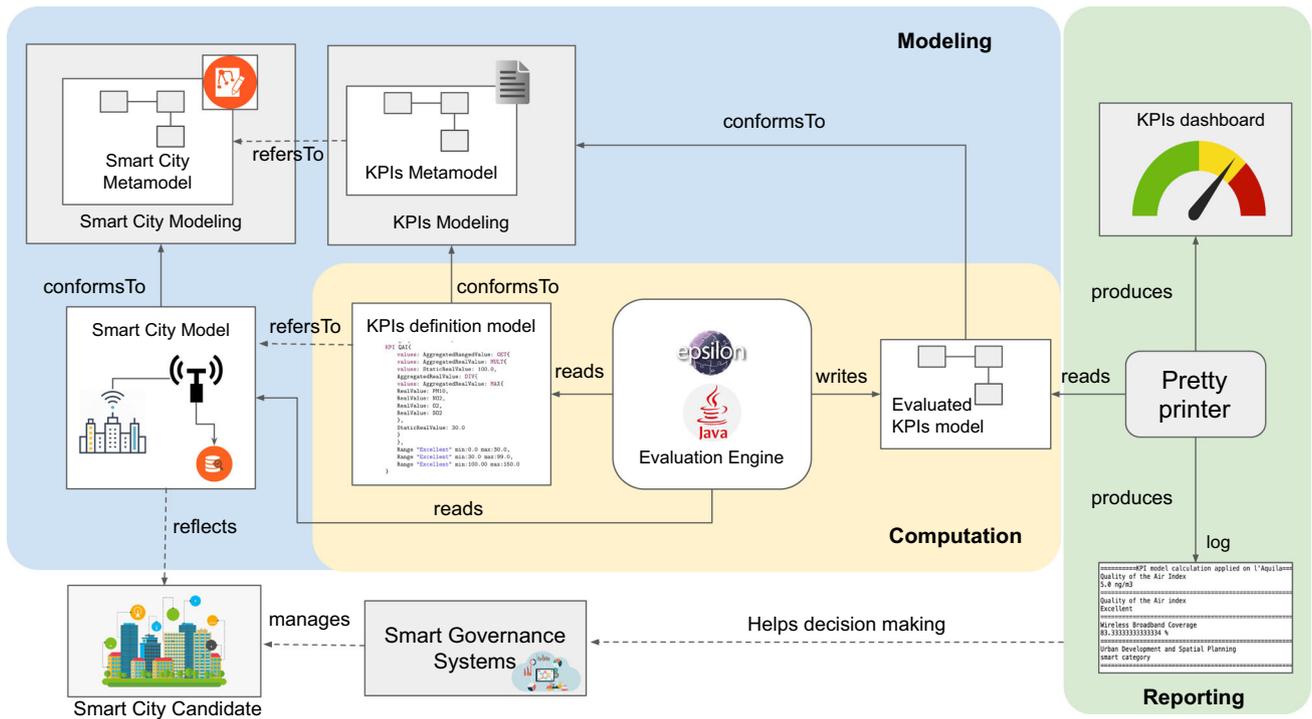


Fig. 2 Overview of the MIKADO approach

governance managers in looking for causes of smart cities problems highlighted by the KPIs.

3.2 The Smart Cities Metamodel

In this subsection, we present the *Smart City Metamodel*. In Fig. 3, we show the metamodel we devised to design smart cities. In particular, considering our interest in supporting decision making, we mainly target the data analytics portion of the metamodel. However, it has already been conceived to include additional aspects, such as IoT infrastructures and stakeholders.

The SmartCityModel is specified as a composition of *multiple* SmartCity entities, thus to model different smart cities. Each SmartCity can be, in turn, composed by 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. Therefore, 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 one or more MonitoringInfrastructures among those modeled in the Infrastructure package, from SocialMedia, from an OpenData dataset, or that we can obtain ProvidedData from a smart city Stakeholder. Here, the prominent Monitoring-Infrastructure we consider is the IoT network of the city, such as the infrastructure made by the sensors and actuators deployed around the city and enabling for different IoT sensing (e.g., traffic monitoring, smart grids). For ProvidedData, we intend data requested from third parties for a given reason (e.g., data from telco operators). 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. The *url* attribute can be used to automatically and continuously retrieve data from publicly available data sources, e.g., open data portals, social networks API, etc. In particular, SocialMedia data can be used, for instance, in the participatory governance context (e.g., in [7]). Lastly, the Stakeholder package contains the before mentioned Stakeholder component. It allows designers to model every type of

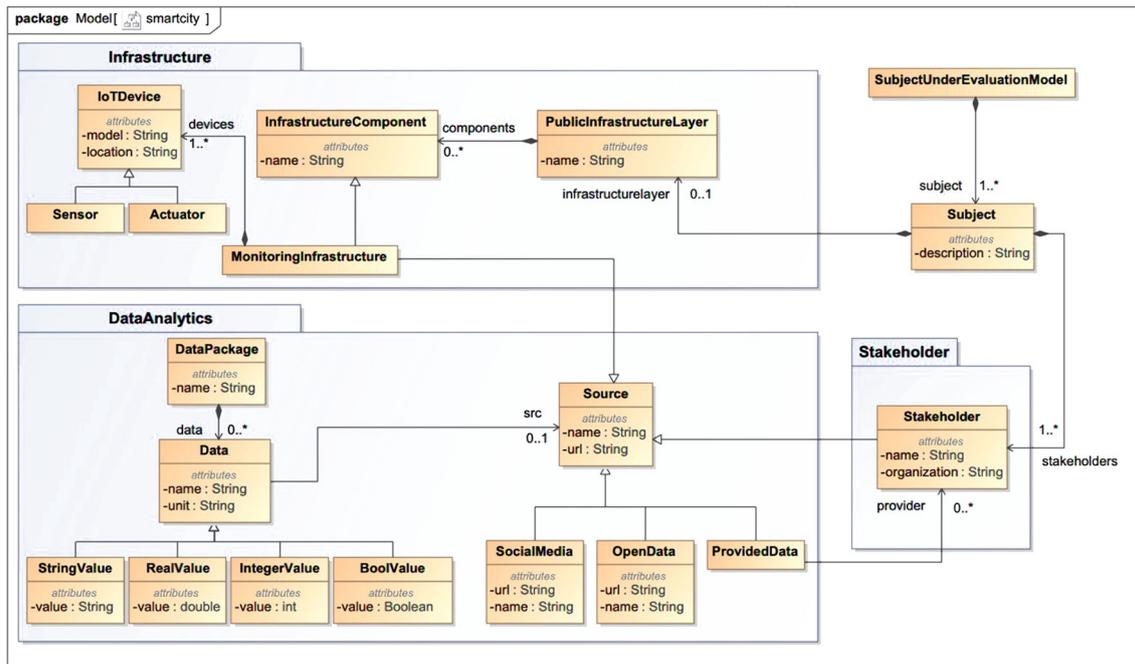


Fig. 3 Smart City Metamodel

smart cities stakeholders (e.g., private and public institutions, companies), which can act or not as data providers.

3.3 The KPIs Metamodel

We now describe the second metamodel artifact required for the KPIs definition. Figure 4 reports an excerpt of the KPIs Metamodel.

It is inspired from the metamodel presented in [23], used to define quality characteristics of modeling artifacts. According to the KPIs sources we analyzed for smart sustainable cities (e.g., [6–8]), KPIs are hierarchically organized. A Kpi-Model, indeed, is composed by multiple Dimensions that can also contain different sub-dimensions. Then, each Dimension is composed by different Category entities that, in turn, contain a set of KPIs. Each KPI is associated with only one category and each category is composed by *multiple* Parameters, referring to the specific parameters that can be used in the calculation of the KPIs pertaining to that category. Each KPI is described by the *name*, *description* and *unit* attributes and has a Value that, in turn, is associated to a ValueType. The *unit* attribute helps designers in specifying and measuring quantities [24], although the explicit representation in the type system is subject to future work. The ValueType is specialized into two main different types, namely RangedValue and CalculatedValue. The former is required to model those KPIs whose calculation methodology produces a value which has to be compared against a Range of values to get the final KPI measure. The Range com-

ponent is further described by the *rangeName*, *min* and *max* attributes. The latter, instead, refers to those KPIs for the measurement of which it is sufficient to perform a calculation. In particular, calculations can be made over different SingleValue that can be of different types (i.e., StaticRealValue, BoolValue, RealValue, IntegerValue, StringValue). Such single values are associated to a Parameter. Moreover, calculations can be made also on AggregatedValue of different types (i.e., AggregatedBoolValue, AggregatedRealValue, AggregatedIntegerValue, AggregatedStringValue, AggregatedRangedValue). Each type of AggregatedValue is defined by the attribute *operation* whose type is specified by the enumeration Operation, which defines the typical operations that can be used to calculate KPIs (e.g., MAX, AVG). Some of the types of SingleValue and AggregatedRealValue contain an attribute called *targetvalue* that can be instantiated with the desired value for the defined KPI.

We want to highlight here that we built both the *Smart City Metamodel* and the *KPIs Metamodel* after analyzing the KPIs documentations provided, for instance, by ITU [6] (see Sect. 4). This way, data types, data sources and operators correspond to those envisaged by the standardization bodies. This does not exclude that they may evolve or more complex ones (e.g., vectors, matrixes, non-standard operators) might be required, due to the KPIs evolution discussed in Sect. 2. To this aim, both metamodels can be easily extended by means of a single modeling step, i.e., adding a child element to the Data or Source elements, in the *Smart City Metamodel* or adding a child element to the SingleValue element in the *KPIs*

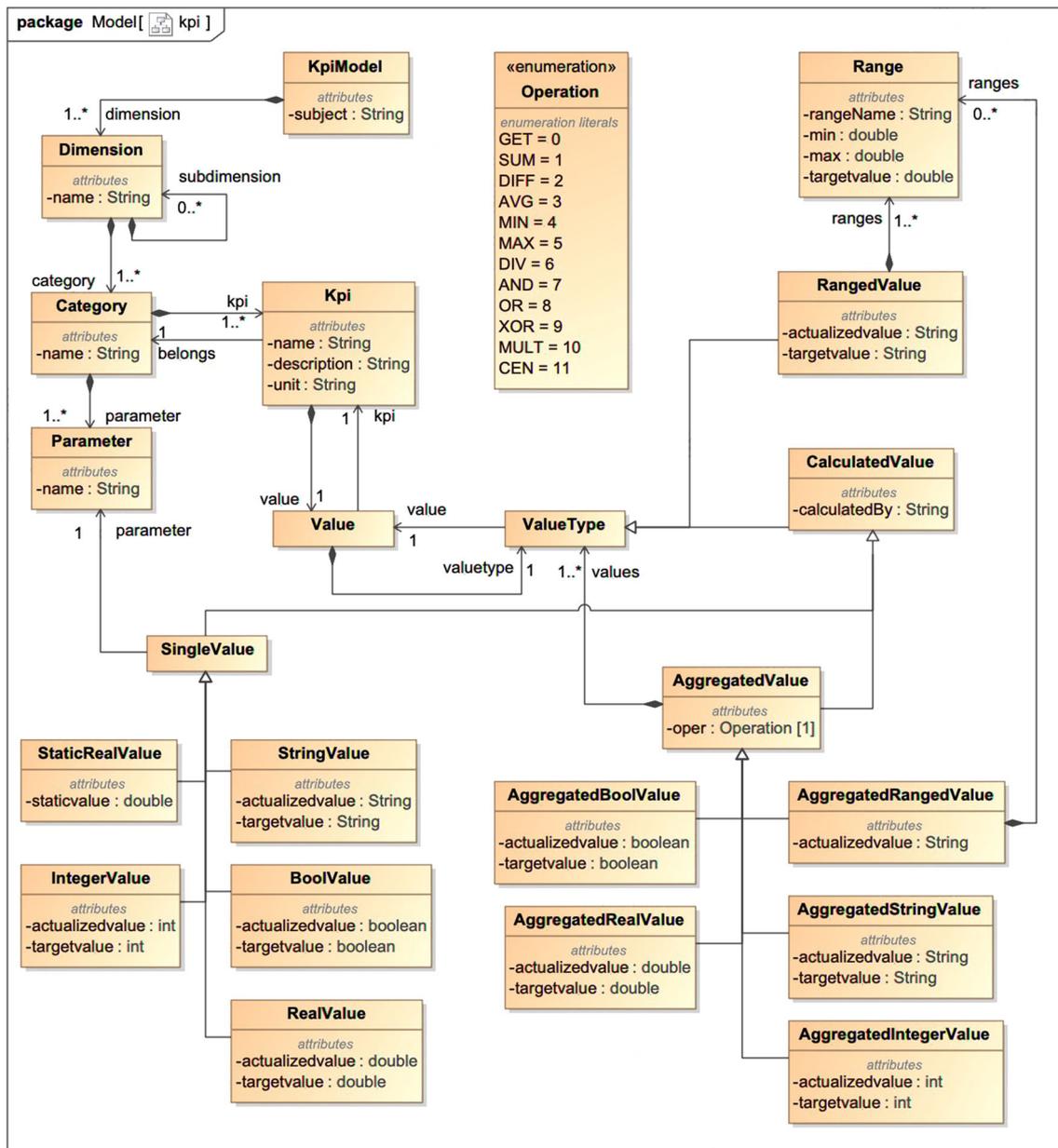


Fig. 4 KPIs Metamodel

Metamodel. Including a new operator would require to add it in the Operation enumeration in the KPIs Metamodel and further implement it in the evaluation engine. Considering all these aspects, the presented approach may result to be a perfect candidate for a multi-level modeling [25] realization, alternatively to the one provided in this work.

3.4 The evaluation engine

The computation component involved in the proposed KPIs assessment approach is the Evaluation Engine. It receives as input the KPIs definition model and the smart city can-

didate under study, in terms of the smart city model. It returns an evaluated KPIs model for the subject. The evaluation engine has been implemented with Epsilon, via the Epsilon Object Language (EOL) [26]. EOL is a language built on top of the eclipse modeling framework (EMF) that can be used both as a standalone generic model management language or as an infrastructure to build task-specific languages, as done for this work. The evaluation engine can be activated by a Java main method located in the SmartCityEvaluationEngine, whose code is reported in Listing 1. The main method of this class loads the input models, i.e., smart city model and KPIs Definition model passed as param-

ters (lines 3–6) and executes the EOL script at line 11. This code can be invoked by a plugin which can be activated by selecting the right models, based on the file extension.

```

1 public List<IModel> getModels(String smartcity, String
  kpiModel) throws Exception {
2     List<IModel> models = new ArrayList<IModel>();
3     models.add(createEmfModel("smartcity", smartcity,
4         "smart_city.ecore", true, false));
5     models.add(createEmfModel("kpi", kpiModel,
6         "kpi.ecore", true, false));
7     return models;
8 }
9
10 @Override
11 public String getSource() throws Exception {
12     return "epsilon/evaluation-engine.eol";
13 }

```

Listing 1 Snippet of the Java Evaluation Engine.

An excerpt of the invoked EOL script is reported in Listing 2. This main script takes as input the smart city model and, for each defined city (line 3), it calculates all the KPIs defined in the KPIs model (lines 5–9). It prints the description of the KPIs (line 6) and the result of the evaluation (line 7) in the console.

```

1 import "kpi-providers.eol";
2
3 for (smartc in smartcity!SmartCity.all) {
4     ("==KPI model calculation applied on "+
5         smartc.city+"==").println();
6     for (mykpi in kpi!Kpi.all) {
7         mykpi.description.println();
8         (smartc.collect(sc|sc.get(mykpi)).first+"
9             "+ mykpi.unit).println();
10    }

```

Listing 2 Snippet of the Main EOL Script.

This script uses a fundamental operation `get()` invoked on the current KPI that basically makes all the computation. This operation is defined in the EOL file called `kpi-providers.eol`, imported at line 1. For sake of completeness, we report also the imported EOL script in Listing 3 where the operation `get()` has been defined for each `ValueType`. For instance, lines 5–14 report the operation for the `SingleValue`, where the `actualizedValue` of the KPIs model stores the calculated value. In order to get the result, the engine retrieves the parameters needed for the calculation through name-based references between the KPIs definition model and the smart city model (lines 6–9). It is worth noting that if the smart city model does not contain the definition for the parameter involved, this operation will not actualize the result. The same for the `AggregatedValue` that recursively calculates the result until the single values. Lines [37–48] are the translations for the operations defined in the KPIs model, e.g., SUM, AVG, to the language implementing the engine. If other operations are needed, we just need to add other cases in this switch statement. For instance, we specified the CEN operation (line 46) to specifically measure the value of *one 100.000th of the city population*, which is often used in the calculation of several KPIs whose value is measured based on the city inhabitants (e.g., see Sect. 4.1, formula (1) for

the Green Areas KPI). Moreover, we added a check for those cities with less than 100.000 inhabitants, namely if the passed value (i.e., the number of inhabitants) is less than 100.000, the operation returns 1.

```

1 operation smartcity!SmartCity get(kpi:
  kpi!Kpi): Any{
2     return kpi.value.valueType.get(self);
3 }
4
5 operation kpi!SingleValue get(city:
  smartcity!SmartCity): Any{
6     self.actualizedvalue=city.data
7     collect(dat|dat.data).flatten()
8         .select(d|d.name==self
9     parameter.name).value.first;
10    return city.data.collect(dat|dat.data).
11    flatten()
12        .select(d|d.name==self
13    parameter.name).value.first;
14 }
15
16 operation kpi!StaticRealValue get(city:
  smartcity!SmartCity): Any{
17     return self.staticvalue;
18 }
19
20 operation kpi!AggregatedValue get(city:
  smartcity!SmartCity): Any{
21     self.actualizedvalue=self.values
22     get(city, self.oper);
23     return self.values.get(city, self.oper);
24 }
25
26 operation kpi!AggregatedRangedValue get(city:
  smartcity!SmartCity): Any{
27     self.actualizedvalue=self.ranges
28     select(r|r.min <= self.values.get(city,
29     self.oper).first and self.values.get(city,
30     self.oper).first <= r.max).rangeName.first;
31     return self.ranges.select(r|r.min <=
32     self.values.get(city, self.oper).first
33     and self.values.get(city,
34     self.oper).first <=
35     r.max).rangeName.first;
36 }
37
38 operation OrderedSet<kpi!ValueType> get(city:
  smartcity!SmartCity, oper:
  kpi!EEnumLiteral): Any{
39     return self.collect(s|s.get(city)).
40     getop(oper);
41 }
42
43 operation Any getop(op: kpi!EEnumLiteral): Any{
44     switch (op) {
45     case kpi!Operation#GET: return self;
46     case kpi!Operation#SUM: return self.sum();
47     case kpi!Operation#AVG: return
48         self.sum()/self.size();
49     case kpi!Operation#MIN: return self.min();
50     case kpi!Operation#MAX: return self.max();
51     case kpi!Operation#DIV: return
52         self.get(0)/self.get(1);
53     case kpi!Operation#MULT: return
54         self.get(0)*self.get(1);
55     case kpi!Operation#CEN: if(self.get(0) >=
56         100000) return self.get(0) / 100000;
57         else return 1;
58     default : "No operation provided
59         ".println(); }
60 }

```

Listing 3 Snippet of the EOL KPI provider.

3.5 Visualizing the evaluated KPIs model

This component is based on Picto [27], an Eclipse view for visualizing models via model-to-text (M2T) transformation to SVG/HTML. Briefly, in Picto, rule-based model-to-text transformations expressed in the Epsilon Generation Language (EGL) [28], an M2T transformation language, are used to transform models into hierarchically organized read-only graphical views, which are then rendered in an embedded Web browser. Views belong to a range of formats, such as SVG, HTML, PlantUML and Graphviz. Picto supports advanced features such as lazy view computation, layers, and composite views.

In our case, the views are generated based on three concepts, i.e., the overall KPIs model, dimensions and categories, and the graphical view generation is implemented in three EGL templates, i.e., KPIModel2Picto, Dimensions2Picto, Categories2Picto. These three templates generate HTML/Javascript code using the *Chart.js* library to render the charts of the indicators results.⁷ The templates correspond to the different views. In particular, the first one is the general view for all the KPIs declared in the model, then a view is dedicated to each dimension, and one for each category. What differs in between the three generated views is the level of nesting of the KPIs granularity, which in turn affects the way of navigating the KPIs. What is interesting is the different ways to represent KPIs in the graphical charts, which depend on both the KPIs value types and the actualized values returned by the assessment. For instance, examples can be:

- *gauges* (cf. left hand side of Fig. 5) are used to represent those KPIs calculated with an aggregation of real numbers, such as MIN, MAX, AVG, etc., as defined by the enumeration OPERATION in the KPIs Metamodel;
- *ranges* (cf. right hand side of Fig. 5) are used to represent those KPIs whose value belongs to a given range. The corresponding chart shows a label for each range options, where the one resulting from the assessment is highlighted;
- *progress bars* are used to represent KPIs of type integer;
- two *buttons* are used to represent Boolean KPIs with the one resulting from the assessment selected;
- *radar charts* are useful for comparing KPIs with the same numerical scale.

In the demonstration case discussed in Sect. 4, we report the dashboard for one of the evaluated smart cities (cf. Figs. 9 and 10).

Listing 4 reports a snippet of the code generator for creating the gauge. For each KPI whose return value is a real and

⁷ <https://www.chartjs.org/>.

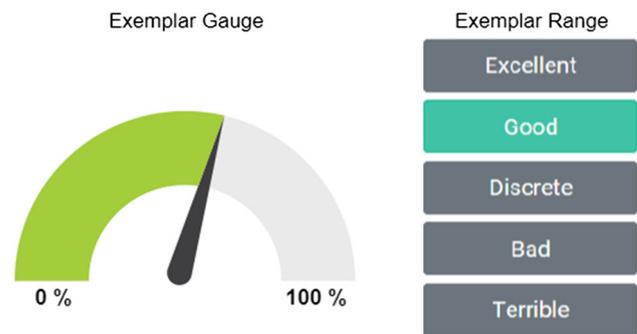


Fig. 5 Example of gauge and range charts

has a *target value* defined (as specified in the KPIs Meta-model in Fig. 4), it generates the needed Javascript code. Lines 12–19 set both the target value, e.g., the 100% value w.r.t. the KPI definition, and the value for the gauge indicator in percentage w.r.t. the target value. Lines 4–10 initialize the gauge graphs, with the options defined in lines 12–19. As can be seen from lines 21–22 for the other types of KPIs different code is generated. Moreover, the template may be customized in order to generate other types of representations. For instance, we are planning for future work to use radar charts to give an global overview of KPIs belonging to the same dimension or category.

Listing 4 Snippet of the EGL script for generating gauge chart.

```

1  ...
2  [%
3  var i=0;
4  for (kpi in category.kpi.flatten()) { [%]
5  [%if (kpi.value.getValue().isTypeOf(Real) and
6  kpi.value.getTargetValue().isDefined())[%]
7  if ($('#chart_gauge_[%=i%]').length) {
8  var chart_gauge_[%=i%]_elem =
9  document.getElementById('chart_gauge_[%=i%]');
10 var chart_gauge_[%=i%] = new Gauge(chart_gauge_[%=i%]_elem).
11 setOptions(chart_gauge_settings);
12 }
13 if ($('#gauge-text-[%=i%]').length) {
14 chart_gauge_[%=i%].maxValue = 100;
15 chart_gauge_[%=i%].animationSpeed = 32;
16 chart_gauge_[%=i%].set([%=kpi.value.
17 getValue()*100/kpi.value.getTargetValue(%)]);
18 chart_gauge_[%=i%].setTextField(document.
19 getElementById('gauge-text-[%=i%]'));
20 }
21 [
22 else if (kpi.value.valuetype.isTypeOf(AggregatedRangedValue)) { ... }
23 else if (kpi.value.valuetype.isTypeOf(BoolValue)) { ... }
24 i++;
25 }
26 %

```

4 Demonstrating MIKADO

In this section, we present a demonstration case in order to show an instantiation of the presented framework and how real-world KPIs are modeled and assessed on top of one or

more smart cities. The goal of this demonstration is to show the feasibility of the proposed methodology, the expressiveness of the specified DSL for KPIs modeling, and how the framework supports the comparison among multiple smart cities. In the following, we introduce a selected set of real-world KPIs which are subsequently modeled, computed, and visualized by *MIKADO*. Furthermore, we also show how *MIKADO* can support the comparison and evolution of KPIs for different cities.

4.1 Selected real-world KPIs

In order to demonstrate the *expressiveness* of the proposed KPIs DSL, we selected existing KPIs from multiple sources, specifically those reported in Table 2. We selected these sources because of the heterogeneity and completeness of the defined KPIs. In particular, ITU [6] and CITYkeys [7] are two projects specifically targeting the definition of smart cities KPIs. ITU defined 91 indicators for smart cities where each KPI has been selected throughout a review process by international experts, to capture the performance of a city in multiple dimensions. CITYkeys has been funded by the European Union HORIZON 2020 program. It has been executed with the involvement of multiple cities aiming for KPIs and data collection procedures definitions for common and transparent monitoring, and in addition, it targeted the comparability of smart city solutions across European cities [7]. DigitalAQ [8], instead, is an EU project, involving both public authorities and academia, aiming to help cities achieve sustainable economic growth through the integration of advanced technologies. It has been carried out in the city of L'Aquila (Italy) and, among other things, it also targeted the KPIs evaluation over the city.

In Table 2, we report the KPIs that we selected for our application example. In particular, we selected KPIs spanning across different dimensions, i.e., environment, infrastructure, transport, and requiring diverse calculations to be measured, to guarantee a wider coverage of the KPIs complexity. Moreover, the diverse sources use different ways to classify the KPIs. For instance, in ITU and DigitalAQ, KPIs are hierarchically classified in *dimensions* (e.g., *Economy*, *Environment*, and *Society and Culture*) that, in turn, contain multiple *sub-dimensions*. ITU further structures sub-dimensions in multiple *categories*. CITYkeys, instead, organizes KPIs in *themes* containing *sub-themes*. Eventually, each source shows a hierarchical KPIs organization, even though using a different terminology. This aspect has been taken into account when defining the KPIs Metamodel that is abstract enough to model KPIs coming from different sources.

In the following, we describe every KPI w.r.t. its dimension and calculations. We start by the ITU provided KPIs [6]. We can notice that the different KPIs sources show some

overlaps, as expected. Indeed, the *Green Areas* and *Bicycle Network* KPIs are reported by both the ITU and CITYkeys sources. The KPI *Green Areas* (GA) measures the green area in the city per 100.000 inhabitants [6]. It belongs to the dimension *Environment*, sub-dimension *Environment* and category *Public Spaces & Nature*. It is calculated as in (1), taking in input two parameters, i.e., the total area of green space in the city in hectares and the city's population.

$$GA = \frac{TotalGreenArea}{\frac{1}{100000} \times CityPop} \tag{1}$$

The KPI *Bicycle Network* (BN) measures the length of bicycle paths per 100.000 inhabitants [6]. Its calculation is similar to the one defined in (1) except that in this case we take into consideration the total length of bicycle paths in the city. This KPI belongs to the dimension *Economy*, sub-dimension *Infrastructure* and category *Transport*.

$$BN = \frac{BikePathLength}{\frac{1}{100000} \times CityPop} \tag{2}$$

The KPI *Air Pollution* (AP) measures the air quality based on the values reported for specific pollutants [6]. It is part of the dimension *Environment*, sub-dimension *Environment* and category *Air Quality*. The *Air Pollution* is based on the *Air Quality Index* (AQI) KPI for specific pollutants, such as particulate matter (PM), nitrogen dioxide (NO²), sulfur dioxide (SO²) and ozone (O₃). Usually, the AQI is calculated as in (3), where *p* refers to the pollutant, whereas the *legal limit* is established by the law.⁸

$$AQI_p = \frac{measured\ concentration_p}{legal\ limit} \times 100 \tag{3}$$

The worst AQI_p determines the *Air Pollution* KPI, as in formula (4):

$$AP = \max(AQI_{PM2.5}, AQI_{PM10}, AQI_{NO^2}, AQI_{SO^2}, AQI_{O_3}) \tag{4}$$

Both AQI_p and AP are measured as μg/m³ and evaluated w.r.t. five evaluation classes defined by the thresholds described in (5).

$$Index\ evaluation\ class = \begin{cases} Excellent & \text{if } Index < 30 \\ Good & \text{if } Index \geq 30 \wedge Index < 66 \\ Discrete & \text{if } Index \geq 66 \wedge Index < 99 \\ Bad & \text{if } Index \geq 99 \wedge Index < 150 \\ Terrible & \text{if } Index \geq 150 \end{cases} \tag{5}$$

⁸ <http://apollon-project.it/2019/12/10/indice-di-qualita-dellaria-iqa/>.

Table 2 Sources of the KPIs definition used in the experiment

KPI	ITU [6]	CITYkeys [7]	DigitalAQ [8]
Green Areas	✓	✓	
Bicycle network	✓	✓	
Air Pollution	✓		
NO2 emissions		✓	
PM2.5 emissions		✓	
Real-time transport monitoring			✓
# of mobile applications			✓

The following two KPIs belong to the *CITYkeys* source [7], as indicated in Table 2, and are thus structured in themes and sub-themes. We selected the *NO2 emissions* and *PM2.5 emissions* KPIs [7]. Both of them belong to the theme *Planet* and sub-theme *Pollution & Waste* and take as input the emissions of the corresponding pollutant p and the city's population:

$$Emissions_p = \frac{P}{CityPop} \quad (6)$$

Eventually, from the *DigitalAQ* source [8], we selected the following KPIs. The *real-time transport monitoring* (TM) refers to the (un)availability of real-time transport monitoring systems in the candidate city [8]; thus, it can be represented by a Boolean value. It belongs to the dimension *Infrastructure* and sub-dimension *Digital Infrastructure*.

Number of mobile applications (MA) indicates the number of mobile applications available in the city (e.g., food delivery, car sharing) [8], and it is obtained by collecting mobile applications information from different stores. The KPI belongs to the dimension *digital competencies* which is a sub-dimension of *competencies*.

4.2 KPIs definition and smart city models

In the following, we describe both the *KPIs model* and *smart city model* artifacts and then show how the evaluation engine performs the KPIs measurement, by analyzing the provided *evaluated KPIs model*. In this demonstration, we use the smart city case of L'Aquila (Italy) as subject. L'Aquila is an highly dynamic city due to the post-seismic reconstruction process still in progress. Applying our KPIs assessment approach in this context should represent a meaningful test case for evaluating our methodology, due to the continuous evolving nature of the city that should be captured by the calculated KPIs.

KPIs Definition Model After the KPIs selection, we defined our KPIs model based on their definition and formulae. This model defines the KPIs described above. We show here a snippet of the textual notation for the definition of a single KPI in Listing 5. In the Listing 1 in *Appendix A*, we report the definition of the other KPIs used in this demonstration case with the textual notation, since it makes the model easier to read and understand. We can see for every KPI the declaration of its Dimension and Category, as defined in the KPIs Metamodel (see Fig. 4). To give an overview on how KPIs can be modeled with the provided DSL, we describe how we defined the air pollution (AP) KPI in Listing 5, lines 5–50. We focus on the AP KPI since it shows more complex calculations, whereas we leave the interpretation of the remaining simpler KPIs in the *Appendix A* as an exercise for the interested reader. The AP KPI belongs to the *AirQuality* category, whose definition starts at line 4. In Lines 5–50, instead, the calculation of AP is defined as in formula (4), which in turn depends on the formula (3). For each input parameter required for the AP evaluation, i.e., the measured concentration of *PM2.5*, *PM10*, *NO²*, *SO²* and *O₃*, we calculate the percentage of their concentration in the air w.r.t. their legal limits, as for instance done in lines 8–14 for the *PM2.5* pollutant. In particular, given the pollutant measured concentration (line 11), it has to be divided (DIV operator at line 10) by its *legal limit* (modeled as *StaticRealValue* at line 12) and the resulting value is multiplied (MULT operator at line 8) for the *StaticRealValue* of 100. Then, the maximum value among those obtained for the measured pollutants is selected, by the MAX operator (line 7). Eventually, the AP KPI is evaluated against a ranged value modeled at lines 44–48 and corresponding to those in formula 5. The reported operators (e.g., GET, MULT, DIV, MAX in the Listing) are those defined by the enumeration *Operation* of the KPIs Metamodel in Fig. 4.

Listing 5 Air Quality KPI definition model shown in textual concrete syntax.

```

1  KPIModel mykpi;
2
3  Dimension Environment{
4    Category AirQuality{
5      KPI AP{
6        values: AggregatedRangedValue: GET{
7          values: AggregatedRealValue: MAX{
8            AggregatedRealValue: MULT{
9              StaticRealValue: 100.0
10             AggregatedRealValue: DIV{
11               RealValue: PM2.5,
12               StaticRealValue: 25.0
13             }
14           }
15           AggregatedRealValue: MULT{
16             StaticRealValue: 100.0
17             AggregatedRealValue: DIV{
18               RealValue: PM10,
19               StaticRealValue: 50.0
20           }
21         }
22         AggregatedRealValue: MULT{
23           StaticRealValue: 100.0
24           AggregatedRealValue: DIV{
25             RealValue: NO2,
26             StaticRealValue: 200.0
27         }
28       }
29       AggregatedRealValue: MULT{
30         StaticRealValue: 100.0
31         AggregatedRealValue: DIV{
32           RealValue: O3,
33           StaticRealValue: 180.0
34       }
35     }
36     AggregatedRealValue: MULT{
37       StaticRealValue: 100.0
38       AggregatedRealValue: DIV{
39         RealValue: SO2
40         StaticRealValue: 350.0
41     }
42   }
43 }
44 Range "Excellent" min:[0.0
45   max:]30.0,
46 Range "Good" min:[30.0 max:]66.0,
47 Range "Discrete" min:[66.0
48   max:]99.0,
49 Range "Bad" min:[99.0 max:]150.0,
50 Range "Terrible" min:[150.00
51   ]
52 }
```

Smart City Model At this point, the only missing artifact for enabling the KPIs assessment is a SmartCityModel. In Fig. 6, a portion of the graphical representation of the model of the city of L'Aquila is shown. It has been created with the graphical editor presented in [18]. In particular, the focus of the model is on the DataAnalytics package that models all the data collected from the corresponding providers and required to calculate the KPIs modeled above. This representation allows us to design the DataPackages instances, namely AirMonitoring, CityStatistics, BikePaths, GreenAr-

reas, TransportMonitoring and MobileApplication, and their providers as instances of Stakeholders and OpenData. In particular, the Web service BreezoMeter gives live air pollution,⁹ pollen, and fires information of a selected geographical area. Thus, here it is the provider of the real data composing the AirMonitoring data package (i.e., PM2.5, PM10, O3, NO2, SO2, CO2). Instead, the CityCouncil stakeholder provides information about the geographical extension of the city of L'Aquila (i.e., CityExt entity) and its total population (i.e., CityPop entity), participating in the CityStatistics data package. The open data instance PisteCiclabili.com provides the Italian bike paths at the provincial and municipal levels,¹⁰ thus the data composing the data package BikePaths (i.e., BikePathLength). Regarding the information about the green areas, we gain the TotalGreenArea data by the service Atlante Statistico dei Comuni designed here as an instance of open data.¹¹ The data about RealTimeTransportMonitoring is modeled in a data package called TransportMonitoring. The provider of this information is the stakeholder instance GSSI since it is the institute in charge of developing real-time transport monitoring systems and technologies.¹² Eventually, the data package MobileApplications contains the data MobileApplicationPS, that is the number of mobile applications provided by the GooglePlayStore.¹³

4.3 KPIs assessment through the evaluation engine

The SmartCityModel and the KPIModel described above constitute the input for the KPIs evaluation engine (see Fig. 2). A screenshot of the corresponding artifacts with their views are reported in Fig. 7. On the left side, there is the SmartCityModel that, in our scenario, models the smart city of L'Aquila. On the central panel, the KPIs model is shown, reflecting the AP KPI definition. Every element in the KPIs model tree owns its corresponding properties as shown in the property view displayed in the top-right panel. The aggregated value defined in the KPIs model (corresponding to line 6 of Listing 5) and measuring the AP KPI is actualized with the calculated value when the evaluation engine is executed. This is shown in the bottom-right side panel, which shows the property view after the execution. In the console as shown in Fig. 8, the results of the KPIs assessment for the city of L'Aquila are reported.

The output of the evaluation is proposed in two views, the console and the actualized quality model, containing all the calculated evaluations. The former supports a fast feedback

⁹ <https://breezometer.com/>.

¹⁰ <https://www.piste-ciclabili.com/>.

¹¹ <http://asc.istat.it/>.

¹² <https://www.gssi.it/>.

¹³ <https://play.google.com/store/apps?gl=it>.

Fig. 6 Graphical Representation of the smart city model for the city of L'Aquila

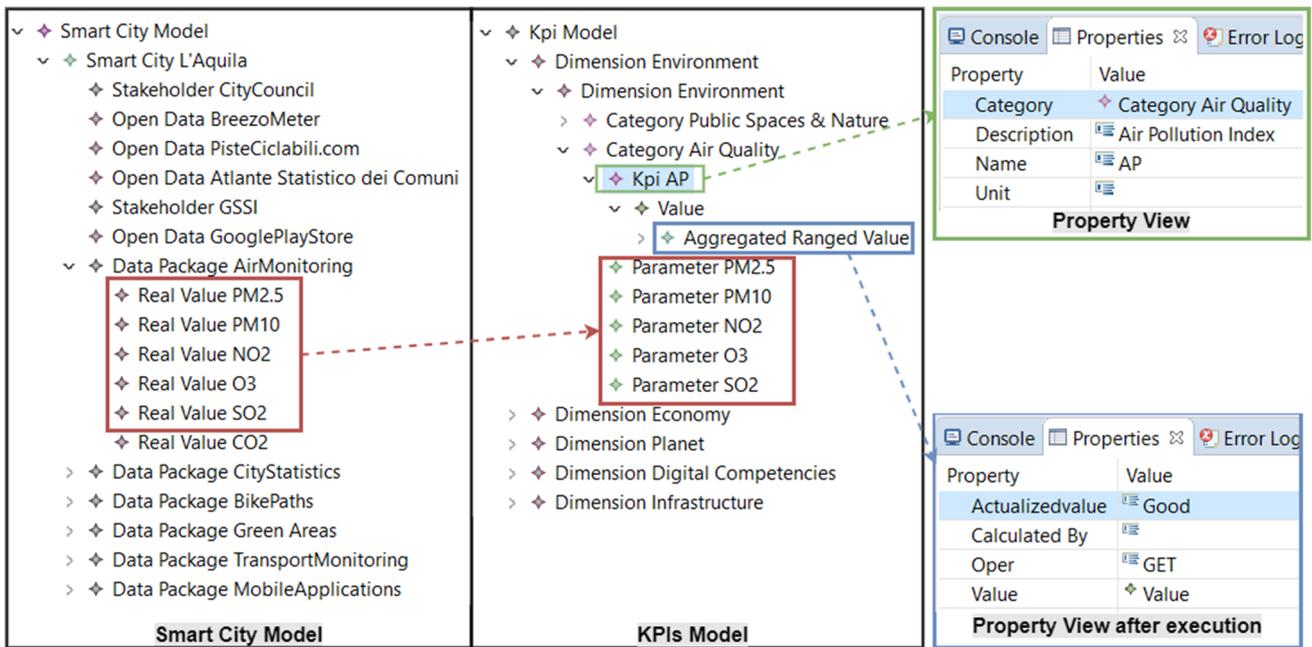
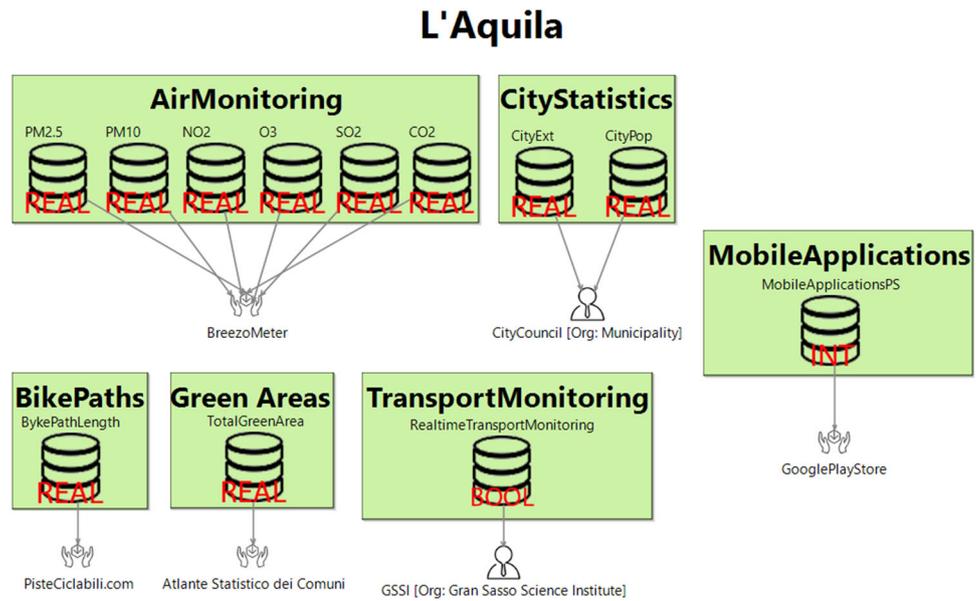


Fig. 7 Examples of the input and output models during the assessment process

and debugging process. The latter can be further processed in order to enable other activities. To this end, the actualized model resulting from the evaluation may be easily translated into data interchange formats, e.g., XML or JSON, in order to enable other applications to interact with it.

4.4 KPIs visualization through dashboards generation

In Fig. 7, we can notice that in order to understand how the indicators have been actualized and which are the values for

the subject smart city, we need to navigate the model looking for the value instantiated for a KPI. This operation is quite counter-intuitive for humans and requires an understanding of EMF's tree-based editor in order to inspect the model and collect the KPIs results for the subject smart city. For this reason, we provide another component (see Sect. 3.5) offering visualization support for the KPIs models which is fully integrated in MIKADO. Thus, the stakeholders can directly have an intuitive and user-friendly visualization of the KPIs' evaluation. A video demonstration of the tool is available at <https://bit.ly/models-tool-kpi>.

```

Epsilon
=====KPI model calculation applied on L'Aquila=====
Green Areas
48.7235 hectares/100.000inhabitants
=====
Air Pollution Index
Good
=====
Bicycle Network
86.0 km/100.000inhabitants
=====
NO2 emissions
8.002298685439265E-5 ng/m3
=====
PM2.5 emissions
1.607643129085554E-4 ng/m3
=====
# mobile applications
13 units
=====
Availability of Transport Monitoring
true yes/no
=====

```

Fig. 8 KPIs Assessment Results over the Smart City of L'Aquila

What we are able to produce with this component is set of views for the KPIs model under inspection. For instance, Fig. 9 shows the KPIs dashboard generated for the KPIs model elaborated for the city of L'Aquila, evaluated w.r.t. the KPIs model of which we showed a snippet in Listings 5.

When we inspect the KPIs model (cf. ② in Fig. 9), the view on the bottom (cf. ③ in Fig. 9) is automatically populated with multiple views. It shows all the KPIs elaborated for this city and the results w.r.t. the target value set in the model. We can see from the KPIs model that we declared KPIs in different dimensions and categories. All the KPIs are summarized in this view and for each KPI a separate view is also generated. On the left side of Fig. 9, a navigator view (cf. ④ in Fig. 9) is generated by Picto, showing all the dimensions, sub-dimensions and categories of the KPIs model. By selecting one of them, a dedicated view populated with the contained KPIs and corresponding results is shown. For instance, Fig. 10 reports the two KPIs belonging to the dimension planet and category pollution and waste. Two indicators are displayed, namely NO₂ (nitrogen dioxide) and PM_{2.5} (particulate matter) emissions with their corresponding values reported as percentage w.r.t. their target values specified in the KPIs model and corresponding to 100% in the gauges dashboards. These indicators are also quantified with units, specified in the KPI model (see Fig. 4).

4.5 Supporting smart cities comparison

In addition to the evaluation of a single smart city to demonstrate the feasibility of the approach, we further modeled two other Italian medium-sized cities, namely Bolzano and Matera. By this, we aim to show that evaluating different cities with our approach will also enable a comparison among them. In particular, in Table 3, we report the KPIs assess-

ments for the three selected cities. From the results in the table, we can observe that, concerning the KPI Green Areas (GA), Matera surpasses the other cities by far because of the presence of wide historical areas. We can see also the results for the Bicycle Network (BN) KPI measured in kilometers per 100.000 inhabitants. The cities of L'Aquila and Matera obtained results much lower than the one obtained by Bolzano. As it concerns the Air Pollution (AP) KPI, all the three smart cities have been evaluated with the class Good. For NO₂ emissions and PM_{2.5} emissions, Bolzano has the lowest values. Eventually, all the cities result to have a real-time transport monitoring (TM) system while, as it concerns the number of mobile applications (MA), Bolzano shows the highest number.

To conclude, we highlighted here that both the *Smart City Metamodel* and the *KPIs Metamodel* are designed to allow smart city designers and KPIs experts to model several relevant concepts, besides the ones we used in our demonstration scenario. They support customization of both smart cities and KPIs, and eventually, they can be extended with further concepts, if required by the domain experts.

4.6 Supporting evolutionary KPIs

Defining general and high-level KPIs is useful to make cities comparable and to rank them. However, based on their dimension (e.g., geographical extent, number of inhabitants), stage of economic development, population growth, etc., different smart cities may select appropriate KPIs among those available or discard those KPIs which are not significant for the given city. For instance, as a trivial example, the *shared bicycles* KPI measures the number of shared bicycles per 100,000 inhabitants. Its calculation might be irrelevant to those cities whose mobility infrastructure does not support shared bicycles services.

The given modeling approach allows designers to select relevant KPIs for the candidate smart cities, thus to enable a customization of the measured KPIs. This can be done through a dedicated editor as envisioned in [17], where we planned a *KPIs Fragment Selection/Customization Editor*. This editor will allow users to “query” the KPIs definition model to select and possibly customize given KPI definitions and generate *model fragments* [29].

A modeled smart city can evolve over time. For instance, when the source of a particular data point changes, we have to update the smart city model. Smart city changes can be of different types and, in this case, they might require more than one modeling step. This is due to the fact that a change in the smart city model may affect the KPIs definition model. Changes that can be performed on the smart cities models can be grouped into three categories [30,31]:

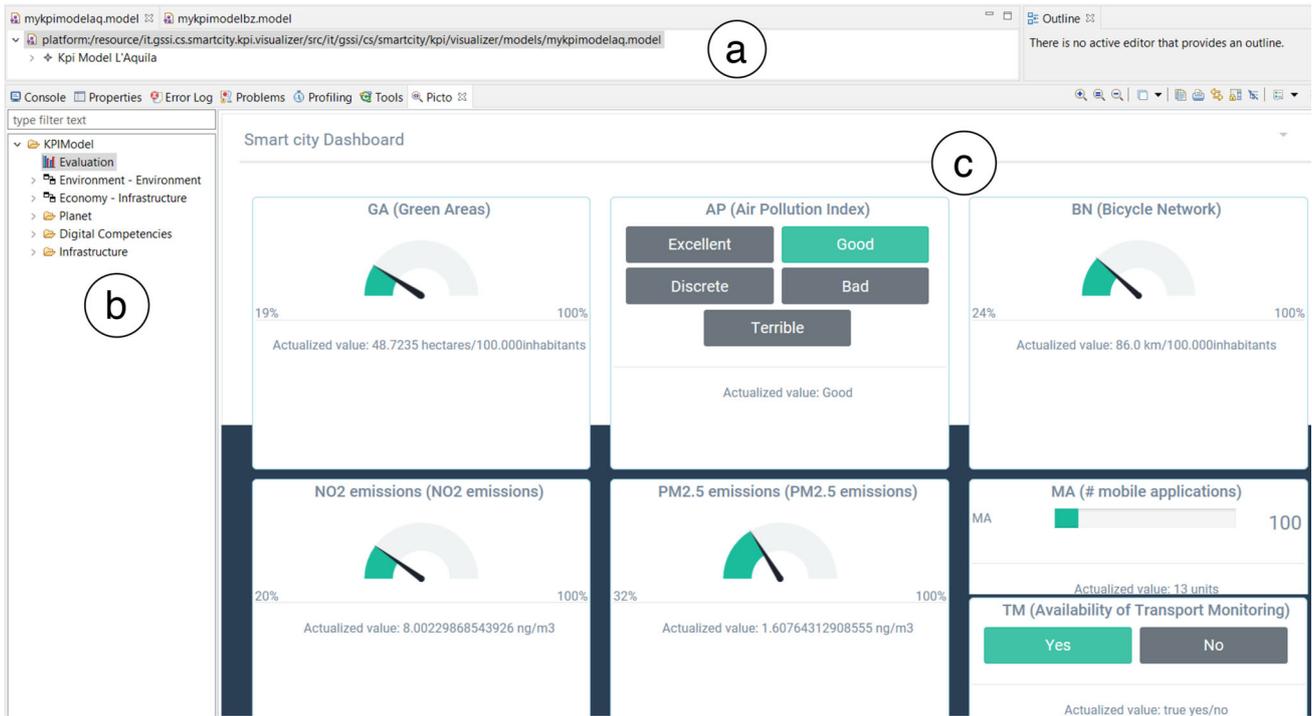


Fig. 9 Overall View of the KPIs for the city of L'Aquila

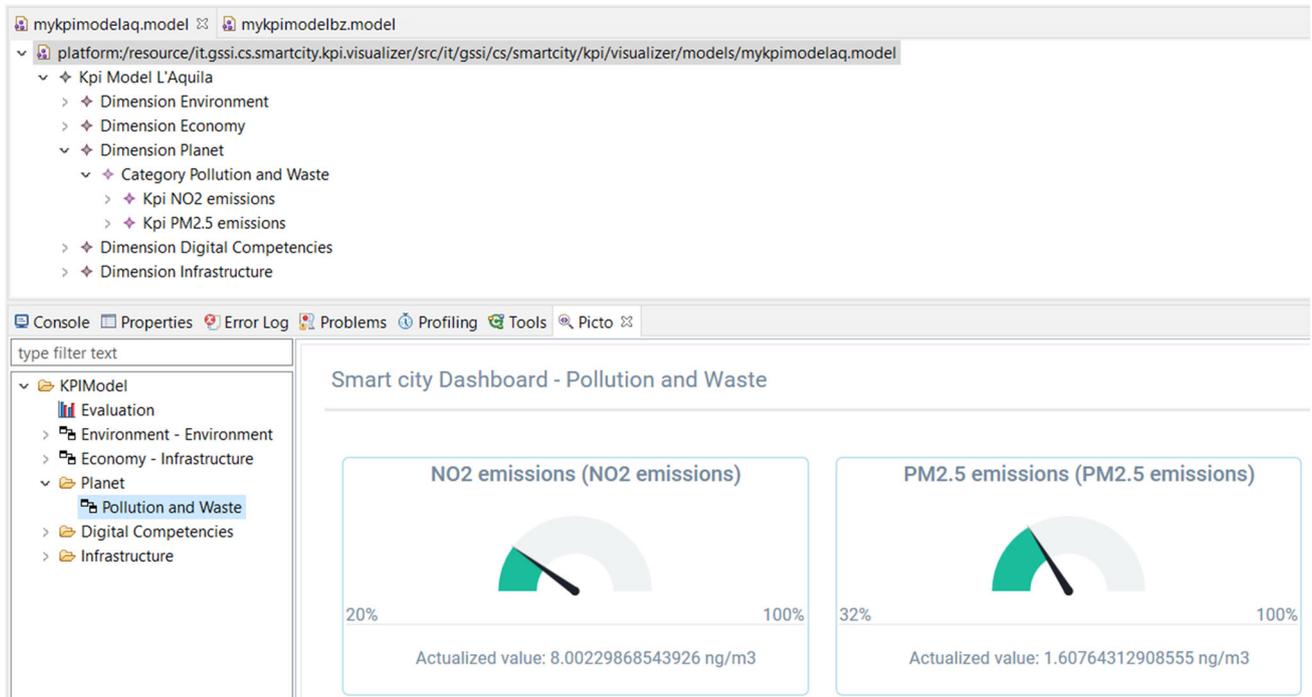


Fig. 10 Detailed view of a single category of KPI

Table 3 Evaluation and comparison of the subject smart cities

KPI	Subject cities		
	<i>L'Aquila</i>	<i>Bolzano</i>	<i>Matera</i>
GA	48.72 ha/100.000 inh.	176.26 ha/100.000 inh.	982.74 ha/100.000 inh.
BN	86.0 km/100.000 inh.	307.15 km/100.000 inh.	66.0 km/100.000 inh.
AP	Good	Good	Good
NO2	0.00008 $\mu\text{g}/\text{m}^3$	0.00002 $\mu\text{g}/\text{m}^3$	0.00004 $\mu\text{g}/\text{m}^3$
PM2.5	0.00016 $\mu\text{g}/\text{m}^3$	0.00014 $\mu\text{g}/\text{m}^3$	0.00023 $\mu\text{g}/\text{m}^3$
TM	YES	YES	YES
MA	13	34	21

- *Additive changes* adding new modeling concepts related to smart cities does not impact existing KPIs definition models. New data providers can be defined in the smart city model, with the effect that they will be available in the KPIs definition as parameters. New stakeholders as well as new data points may be defined which subsequently will be referable by the shown tracing mechanism.
- *Subtractive changes* Removing concepts from the smart city model may impact the KPIs definition models. For instance, removing a data provider affects the KPI definition if a parameter is referring to the removed data package.
- *Structural changes* A typical example is when the name of the parameter used in the KPIs definition model is changed in the smart city model.

Subtractive and structural changes impact the entire process since a static analysis is required in order to check if all the parameters used in the KPIs definition models are defined in the smart city model. This can be realized with validation rules, e.g., checking that a renaming operation performed on the smart city model does not affect the KPI definitions.

Concerning KPIs evolution over time, our approach also supports the modification, and thus, the evolution of KPIs definitions. By exploiting the relationship between the Smart City and KPIs Metamodels, we have a clear contract between them: the parameters defined in the KPIs model should be compatible with the *name*, *type*, and *unit* of the data sources defined in the smart city model. Thanks to the separation of concerns implemented by the approach, the KPIs definition is organized in sort of libraries where the modeler or KPIs expert defines how the smart city parameters can be composed to calculate the required metrics. This mechanism, supported by the textual editor, injects the textual definition into a model that will be used by the engine. Being a model, as any other structured model, it can be queried to select or deselect the needed fragment, implementing the selection mechanism explained above. In particular, also KPIs evolution can essentially be of the same categories as explained before for the smart city models whose implementation can

be realized via a few modeling steps on the KPIs definition model:

- *Additive changes* the rise of a new interesting KPI for the city is the typical example. When designers need to add the calculation of a new KPI, they can have two situations to deal with, i.e., the new KPI belongs to an existing dimension and category or the new KPI takes with it a new dimension and/or category. In the first case, designers have to model the KPI calculation in the corresponding dimension and category. In the second case, besides the KPI calculation, designers have to model the new dimension and/or category. Both situations require modeling operations only in the KPIs definition model. For instance, cities that had no cycling infrastructure at the time of their development, at a certain stage of their evolution may be interested in the calculation of the bicycle network KPI, previously mentioned in this paper, that before would have not been meaningful.
- *Subtractive changes* a typical example is the deletion of an obsolete KPI. When it is no longer interesting to calculate a KPI on a city, it is only required to delete the KPI calculation in the KPIs definition model. For instance, looking at the KPIs taken into consideration in our demonstration case, for the pollutant PM2.5 we calculate an ad hoc KPI called PM2.5 emissions (see lines 47–52 in Listing 1 in Appendix A) and also the air quality index AQI for the same pollutant (see lines 8–14 in Listing 5). This information could be redundant, thus some stakeholder may think of removing the PM2.5 emissions calculation from the KPIs definition model to save space and time for the KPIs assessment.
- *Structural changes* the modification of a KPI calculation logic is the typical case here. Evolution may refer to the calculations used to measure existing KPIs. For instance, the ranges for the air pollution evaluation already changed in the past, as described in the WHO air quality guidelines.¹⁴ Specifically, in 2015, the recommended limits for

¹⁴ <https://bit.ly/3dVT83X>.

the O_3 changed due to the correlation between daily mortality and lower ozone concentration. Further evolution steps may be expected for the future. In our demonstration case, the designers may promptly handle the change at line 33 of Listing 5. Moreover, let us think about the network coverage KPI. After the advent of 5G, the calculation for this KPI evolved, by including the 5G network coverage. This kind of evolution can be easily handled with our approach, by acting on the declaration of the KPI in the model.

The proposed approach is interpreter-based, thus supporting dynamic changes as they will be immediately reflected in the evaluation process [32]. Most of the traditional approaches, as afore discussed in Sect. 2, might need to re-implement and re-deploy the existing platforms depending on the performed evolution.

Coordinating and supporting evolutionary changes to both model types may be implemented with validation rules checking. This feature may be easily integrated in our approach by applying EVL [33] rules, checking if the above identified constraints are fulfilled, and in case they are not, resolutions can be performed with (semi-)automatic model repair rules. EVL indeed provides quick fixes as well as alerts that may guide and instruct the modelers during the resolution process.

In conclusion, our approach can manage dynamic changes based on the interpreter strategy. Moreover, a (semi-)automated mechanism will enable the co-evolution of both smart cities and KPIs models which is left as subject for future work. In particular, co-evolution may be supported by checking the `refersTo` relationship between the smart city model and the KPIs model (see Fig. 2), similar as it is done for the `conformsTo` relationship between models and meta-models [34].

5 Evaluating MIKADO

In this section, we provide an evaluation of our approach which is twofold: (i) we evaluate the scalability of the evaluation engine in managing smart cities and KPIs models of increasing size, and (ii), we assess the usability and understandability of the presented approach, especially w.r.t. spreadsheet-based approaches.

5.1 Scalability of MIKADO's evaluation engine

The purpose of this evaluation is to study the scalability of the approach with an increasing size of smart city models and number of KPIs. Indeed, we recall here that the `SmartCityModel` allows for the modeling of multiple smart cities at a time, thus to simultaneously evaluate them (e.g., to

enable a comparison among smart cities, as done by ranking agencies). This means that the size of the `SmartCityModel`, together with the number of computations to evaluate the selected KPIs, can rapidly increase. For this reason, we aim to guarantee the effectiveness of the KPIs assessment approach when applied over multiple and complex smart cities to measure hundreds of KPIs.

5.1.1 Research question

We aim to answer the following research question (RQ).

RQ *Is the proposed assessment framework, in particular, the evaluation engine, scalable in terms of execution time?*

To answer this research question, we performed an experiment that falls in the area of performance evaluation of MDE artifacts [35], as our interpreter of the proposed modeling language can be considered as a kind of model transformation as described in the following. In particular, we run the experiment for replying to this research question with measurement of the execution time of the evaluation phase only. This means, in this experiment, the input models are ready for the evaluation and the modeling phase is considered as finalized, i.e., all the model elements have been automatically or manually filled. All the artifacts used in the experiments are available on github.¹⁵

5.1.2 Experiment setup

Scenarios. The goal of our experiments is that of checking the evaluation engine execution time w.r.t. the *size of the input models*, i.e., the number of elements in the models. In our case, the input models are twofold. Thus, we designed a smart city model, in which we instantiated every concept of the metamodel, and a KPIs model. In particular, the used KPIs model is initially made by one dimension with one category of 8 KPIs, thus to cover all the calculations defined in the KPIs Metamodel.

In particular, we designed 4 increasingly complex scenarios, whose settings are reported in Table 4. In *Exp1*, for each execution run of the evaluation engine, we increment the number of modeled smart cities (SCs) in the smart city model from 1 to 10 and we measure the 8 KPIs in the KPIs model for each of them. The size of the 2 input models, given by the sum of the smart city elements and the KPIs elements, goes from 200 (at the first run) to 632 (at the tenth run). In *Exp2*, we increased the complexity of the used KPIs operations, by adding new nested operations in every KPI. An example of a nested operation is reported in Listing 5, lines 8–14, where a `DIV` operator is nested into a `MULT` operator. Adding nested

¹⁵ https://github.com/gssi/SmartCityModeling/blob/master/reproduction_package.zip.

operations does not mean having more complex calculations. It might rather impact scalability for two main reasons: (i) it leads to an increase of the KPIs model size, and (ii) it leads to an increase of the KPIs model's depth that, in turn, may impact the KPIs model navigation time. This way, the size of the KPIs model increases from 151 elements in *Exp1* to 195 elements in *Exp2*. Then, as done for *Exp1*, we perform 10 runs measuring the new KPIs model on the smart city model where the number of modeled smart cities increases by one at each run. We designed *Exp3* by adding to every KPI definition in the KPIs model a *range* calculation, since we know from previous work [17] it is the most time consuming operation. An example of a range operation is reported in Listing 5, lines 6–50, where the GET operator used to calculate the AP KPI returns one of the ranged values defined from line 44 to line 48. This is due to the fact that AP is defined as an aggregated ranged value. In these settings, the KPIs model size further increases to 322 elements. Then, we perform 10 runs measuring the new KPIs model on the smart city model defining from 1 to 10 smart cities. Eventually, in *Exp4* we increased the complexity of the KPIs model used in *Exp3* by incrementing its dimensions in such a way to have a KPIs model with 10 dimensions and 80 KPIs. Then, we repeated 10 runs with the new KPIs model of size 3211.

For the design of the described scenarios, we defined artificial models with the aim of instantiating all the entities in the Smart City and KPIs Metamodels, thus to be sure of (i) involving all the model elements in the execution runs, comprising the most time consuming ones (i.e., nested operations in the KPIs model), and (ii) forcing the evaluation engine to navigate complete smart city models (i.e., models providing at least one instance for each metamodel entity).

Measurements. We executed the experiment by running the evaluation engine using a 6 core CPU running at 2.2GHz, with 16Gb memory. We run the evaluation engine on a machine with Windows 10, inside an Eclipse IDE of version 2020-06 (4.16) with Java 8, while the Epsilon version used was the 2.1.0.202006301809. Differently from [17], for this experimentation, the evaluation engine is launched multiple times from a Java program that loads and passes the input models to the engine and stores the evaluated KPIs model after the assessment. The execution time of the evaluation engine, reported in milliseconds (ms), is measured from when it receives the input models to when it returns the evaluated KPIs model (including the model persistence operation). In particular, for each of the 4 experiments described above, we performed 10 execution run, where each run has been repeated 11 times.

5.1.3 Experiment results

In Fig. 11, we show the results of the 4 experiments in terms of execution time. For each run of any of the experiments,

the chart reports the average among the times resulted from the 11 executions. The complete measured execution times are reported online and they show that,¹⁶ apart for the first run when the input models are initially loaded, there have been only minimal differences between the runs. This is also supported by the caching feature of Epsilon, as discussed later in this section.

In Fig. 11, the blue line represents the results of our first experiment *Exp1*. The models size goes from 200 (49 SC elements plus 151 KPI elements in Table 4) to 632 (481 SC elements plus 151 KPI elements in Table 4) elements and the execution time goes from 72 ms to 124 ms. The red line reports the results of *Exp2*. We can observe that the execution time goes from 86 ms to 193 ms w.r.t. an increase of the models size from 244 to 676 elements. The yellow line reports the results of *Exp3*, showing that the execution time ranges from 202 ms to 654 ms, by showing an increase w.r.t. the previous two experiments, thus highlighting the complexity of calculations due to the range operation. However, the overall execution time is still reasonable for the given models size that goes from 371 to 803 elements. Eventually, in *Exp4* (green line in Fig. 11) 10 dimensions made by 80 KPIs are considered. This means that in the last execution we assessed 800 KPIs in the same run (80 KPIs over 10 smart cities). Figure 11 shows that the execution time ranges from 624 ms to 5015 ms, with the models size going from 3260 to 3692.

5.1.4 Answer to the research question

Summarizing, these experiments point out two main findings. Firstly, the efficiency in terms of the evaluation engine's execution time, since all the experiments clearly show a linear increase of the execution time w.r.t. the increasing models size, without any peaks. Moreover, the computation time reduced of 50% compared to the experiments reported in [17]. This is due to relevant code refactoring performed in the framework, namely the embedding of the EOL evaluation engine in a Java program that manages the loading of input models, the multiple runs of the evaluation engine, and the persistence of the evaluated KPIs models resulting from the assessments. Moreover, the infrastructure makes use of the caching feature of Epsilon, which allows for loading on-demand only the models that have been changed between one run and another. Secondly, promising scalability results are shown by *Exp4*, indicating that the system takes approximately 5 seconds for assessing 800 KPIs over 10 smart cities.

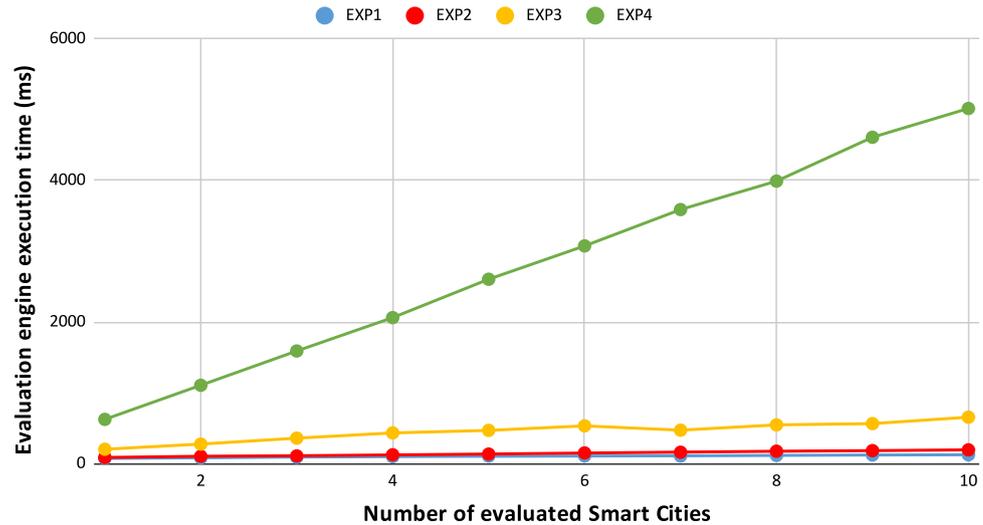
From a qualitative perspective, we performed the following observation. In Sect. 4, we modeled three real-world medium-sized Italian cities, namely L'Aquila, Bolzano, and Matera, to discuss how our approach supports comparison among different cities. In average, the smart city model size

¹⁶ <https://bit.ly/330iMzA>.

Table 4 Increasingly complex scenarios for the scalability assessment

Scenario	SCs	SC elements	KPIs	KPIs elements	Specific computation
EXP1	1–10	49–481	8	151	Single calculations
EXP2	1–10	49–481	8	195	Nested calculations
EXP3	1–10	49–481	8	322	Range calculations
EXP4	1–10	49–481	80	3211	Multiple KPIs dimensions

Fig. 11 Execution times of the evaluation engine resulting from the experiments



for these cities is equal to 80, while the size of the KPIs model used for the comparison is 290 elements. Considering that the smart city models size used in this experimentation is slightly higher (i.e., 49–481 in Table 4), we claim that in our framework we can easily compare 10 real-world medium-sized cities w.r.t. 80 KPIs, where 80 can be considered a valid upper bound, given that the average among the KPIs defined by ITU, CITYkeys and DigitalAQ is 76. We leave for future work the modeling and evaluation of metropolitan cities.

5.1.5 Threats to validity

Our approach strictly depends on data about the smart cities under evaluation. These data are needed for instantiating the parameters required by the assessment process. If these data are not open or providers are not making it available, we can rely on the support of domain experts for the estimation of these parameters for the smart cities under evaluation. Incorrect estimates can lead to incorrect KPIs measurements. However, the overall procedure implemented in the evaluation engine is not affected, since KPIs calculations are based on standard KPIs definitions, which are recognized by the community. This unavoidable problem about the availability of open data is an open issue in this domain that requires a wider investigation about useful data sources to consolidate the presented approach.

With these premises, the experimentation performed in this work may be internally biased from the parameters we used as input to calculate KPIs, which are realistic but not real, in the sense that they are not related to specific cities. More complex data might impact on the performance of the approach, if leading to more time consuming calculations. Moreover, also the time required to get this data in a real scenario, e.g., via APIs calls, has not been part of the calculation of the computation time. Lastly, in *Exp2* we evaluated the approach by adding a nested operation, i.e., the most time consuming one, in every KPI. However, we considered only one level of nesting even though we cannot exclude that some KPIs may require more nesting levels that might degrade the performance for their calculation.

A threat to the external validity is that the results have been obtained on a set of demonstration cases modeled by us. Indeed, we defined artificial models with the aim of instantiating all the entities in the Smart City and KPIs Metamodels and we are aware that this could also threaten the external validity of our experiment.

To increase the representativeness of the input models to our approach, more KPIs and smart cities experts should be involved in a wider experimentation. This is due to the fact that experts may customize existing KPIs and/or define new ones, going beyond the standard KPIs that we consider as representative, i.e., those reported by ITU and recognized at European level. Moreover, we are aware that the application of the approach to more complex smart cities (e.g.,

metropolitan cities) has not been performed. This would be fundamental to evaluate the generality of the approach, by applying it to cities which are distant in terms of dimension, growth, richness, etc. We leave this point as part of our future work.

5.2 Understandability of MIKADO's DSLs

The purpose of this evaluation is to study the usability of the presented approach. The main artefact the smart city domain experts have to use in the presented approach is the DSL to define the KPIs. As a first evaluation in this respect, we focus on the perceived understandability of the DSL. We consider understandability as a major building block for usability and leave for this study tooling aspects aside and as subject to future work. Specifically, as a future evaluation we aim to exploit Technology Acceptance Models (TAM) [36] in order to further measure the *perceived ease-of-use* and *perceived usefulness* indexes of the defined DSLs and the tools making *MIKADO*, so that to evaluate the user acceptance of our approach.

5.2.1 Research question

Given the premises stated above, we aim to answer the following research question (RQ).

RQ *What is the perceived understandability of the proposed KPIs DSL when investigated by smart city domain experts?*

To answer this research question, we performed an expert survey which is described in the following. All the artifacts used in the survey as well as the anonymous results are available.¹⁷

5.2.2 Survey setup and execution

To reply to the stated RQ, the KPIs DSL has to be investigated and judged by different smart cities stakeholders. To this end, we composed an online survey available at <https://forms.gle/dxidpC3z6XkvRpUo7> where we basically show different KPIs expressed as within spreadsheets as well as with our DSL. We used this setting since we consider spreadsheets as a common approach used in the KPIs evaluation domain. Furthermore, when reasoning about understandability it is easier to evaluate in a survey by having a relation to the state of the art.

For each KPI, we show its definition with both our DSL and spreadsheet formula; thus, for each definition we ask to describe what the defined formula computes with an open answer question, and to rank, with a Likert scale (from 1 to 10), the understandability of the current definition. Moreover,

we ask through multiple choice questions with which of the two approaches the user will be more confident to modify or define new KPIs, considering a period of training of 1 week or a training period of 10 weeks. Eventually, we ask, again via multiple choice questions, what smart city representation they preferred, between the graphical representation (as in Fig. 6) and the spreadsheets definition we implemented to make the formulae working.

We submitted the survey to 10 smart cities and/or KPIs experts. In order to involve an heterogeneous group of participants with different perspectives and expertise about smart cities and their evaluation, we contacted experts from: (i) *companies*, such as employees of ITU; (ii) *academia*, via two mailing lists of people involved in smart city projects, e.g., the mailing list of the <https://cs.gssi.it/summerschool/2019> edition; (iii) the *smart city committee of the city of L'Aquila*,¹⁸ nominated by the major and made by smart cities experts. Moreover, participants have not been trained about the DSL, but they just received a brief introduction, while we cannot exclude that they had background on the use of spreadsheets, given their diffusion and popularity.

5.2.3 Survey results

In Fig. 12, the chart reports the votes given in the Likert scales for each evaluated KPI (on the x-axis) w.r.t. the understandability of the two different formulae definitions. Here, we can see that our DSL received better votes, apart for the KPI mobile applications (MA). However, we can argue that the MA KPI is easily calculated by summing up the number of available mobile applications in the city, meaning that in spreadsheets it does not require complex formulae, as in our DSL. On the contrary, for all the other KPIs showing more complex calculations, our DSL resulted to be more comprehensible, by looking at the results. The colored dots in the chart in Fig. 12, instead, indicate the number of responses of the type “*I don't know*” to the question that was asking to describe what the defined formula compute. We can see that, differently from our DSL, for the spreadsheet formulae we registered at least one “*I don't know*” for every KPI definition. Interestingly, for the transport monitoring (TM) KPI that is simply measured with a Boolean value, the users found it easy to understand the spreadsheet formula.

Moreover, Fig. 13 reports the results referring to the questions about the users preferences on using one between our DSL and the formulae in the spreadsheet, or even none of them, when considering a period of training of 1 week or 10 weeks. We can see that in the first case, Fig. 13 left side, the 40% of the users claim that they would like to use our DSL, against a 20% of users preferring spreadsheets. A considerable percentage, namely 30%, would be in favor of using

¹⁷ <https://bit.ly/3kXIJ8x>.

¹⁸ <https://bit.ly/3xqKLFd>.

Fig. 12 The boxplots report the votes given in the Likert scales for each evaluated KPI w.r.t. our DSL and the spreadsheet formulae. The dots indicate the number of responses “I don’t know” given for each KPI description

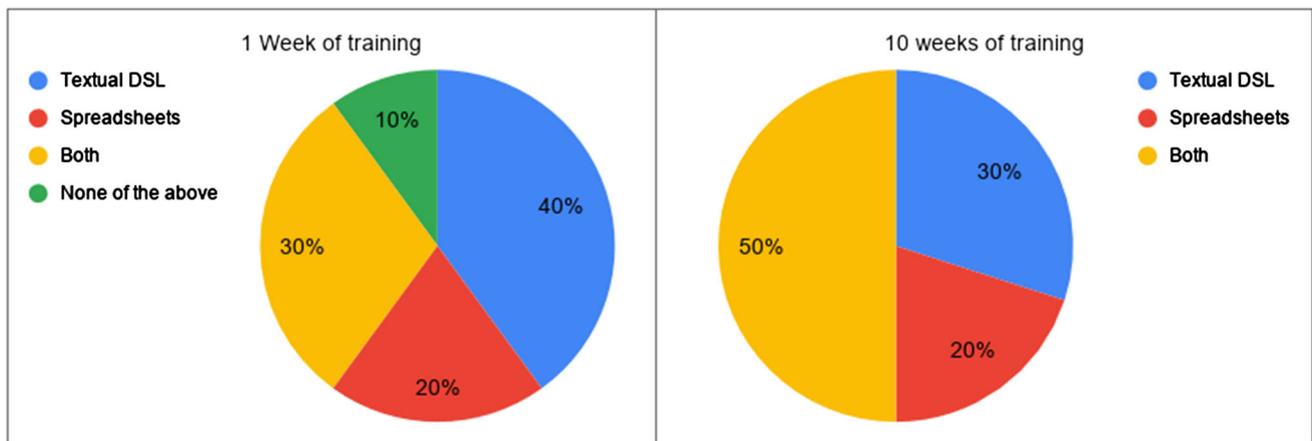
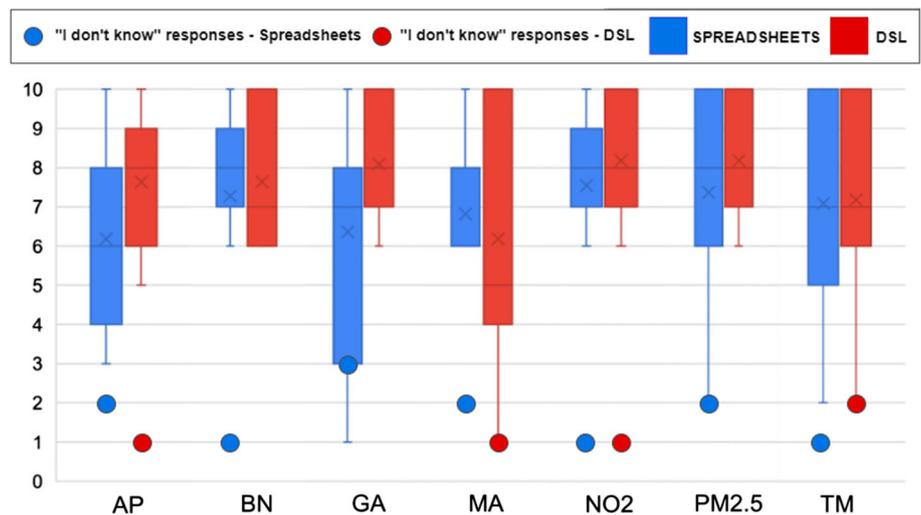


Fig. 13 Users preferences between our DSL and spreadsheets when considering a period of training of 1 week (left side) and 10 weeks (right side)

both of them, indistinctly. Meanwhile, considering a period of training of 10 weeks, Fig. 13 right side, the majority (50%) of the users claim that would use both the DSL and the spreadsheets, indistinctly. The remaining 50% is more in favor of our DSL (30%) than of the spreadsheets (20%).

Eventually, Fig. 14 reports the results about the users preferences w.r.t. the two smart cities representations that we provided, namely the graphical one used in our approach and the tabular one as done in the spreadsheets. The results show that 40% of them liked both representations, indistinctly.

5.2.4 Answer to the research question

From these preliminary results, it seems that our DSL is perceived as more comprehensible w.r.t. spreadsheet-based solutions particularly when considering KPIs models showing more complex calculations. This in spite of the fact that participants were certainly not aware of the DSL, while they probably had basic knowledge about spreadsheets. That being said and even though the spreadsheets-based solution

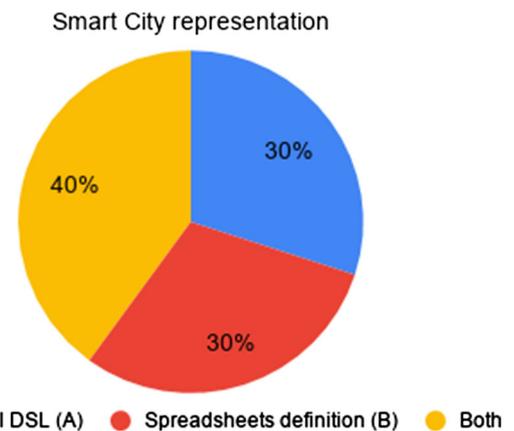


Fig. 14 Users preferences w.r.t. the two proposed smart city’s representations, graphical vs. tabular

did also score well, we can state that complex spreadsheets formulae are perceived more difficult to understand compared to the DSL-based solutions. Moreover, as regards the

smart city model, if not too complex, it is easily comprehensible also when represented in a spreadsheet. From these results, we can argue that with a short training period the users seem to be more confident in using our DSL w.r.t. spreadsheets.

5.2.5 Threats to validity

The preparation of the survey as discussed in Sect. 5.2.2 and its usage to evaluate the understandability of our DSL may represent a threat of construct validity. This is due to the fact that the experts who participated in the survey may be biased by our expectations that may unintentionally leak from the questions. To overcome this problem, we previously submitted the survey to senior colleagues in order to assess it in terms of independence from our expectations. This resulted in a number of iterations on the survey's definition to address the received feedback on the survey itself and to make it as independent as possible from our expectations.

A threat to the external validity is given by the limited number of participants to the survey. However, during the selection process, we aimed to contact potential participants with heterogeneous perspectives, i.e., business, academic, and smart city experts, might mitigate the shortage of participants. A second threat to the external validity follows from the previous one and relates to the non-homogeneity of the selected groups of participants, with different backgrounds, that may be hard to compare. However, in this experiment, especially as regards open-answer questions, we observed quite concise and homogeneous answers, despite the diverse backgrounds of participants. Moreover, none of the answers were out of topic or indicating misunderstanding of the questions by the participants.

6 Related work

In this section, we discuss related work concerning the application of MDE for Smart Cities as well as the application of KPIs for Smart Cities. Furthermore, we also discuss related approaches for quality assessment as well as for spreadsheet engineering which may also find further application in the field of smart cities.

6.1 MDE and smart cities

As already introduced, smart cities are actually considered *systems of systems*, made by different dimensions and involving several stakeholders. These features make them so complex to handle. The management of heterogeneous contexts and aspects has brought out the need of a high-level of abstraction of the different processes happening in the context of a smart city. MDE can be helpful, and it has been

already used in this domain. In [11], the authors present a DSL to model Smart City Systems (SCSs). They face with the high heterogeneity of services, devices, and communication protocols that can help in creating a SCS. They further show that a well-defined DSL in a given domain can be understandable also by experts with no knowledge in software engineering. Instead, in [12] the author exploits MDE to tackle the issue of limited collaboration between the different stakeholders in the smart city domain. The use of DSLs in such contexts supports understandability and analysis compared to general-purpose programming languages (GPLs) source code patterns that are more complex and often not well defined [37]. Indeed, the risk of using a GPL is to be too abstract and to struggle in the development phase. For these reasons, MDE-based tools and methods filling the gap between the high-level abstraction of domain concepts and the low-level abstraction of GPLs are proposed in [38], to deal with complex systems, such as those of smart cities.

6.2 KPIs and smart cities

Besides the ITU report [6], there are also other attempts of standardizing smart cities-related approaches and technologies. For instance, [13] mainly focuses on the development of standard indicators to measure technical and economic aspects for projects aimed at energy efficiency. Instead, in [7], KPIs and data collection procedures are defined to promote a transparent monitoring and comparability of smart cities solutions across European cities. Moreover, [39] proposes a methodology for the creation of a repository of all the possible KPIs for smart cities in order to support their selection for the interested users. In [40], the authors highlight the fact that to face smart cities evolution many organizations build models to contain the different KPIs of interest. They aim to overcome the difficulty in comparing different KPIs models by developing a framework for the analysis of smart cities models and by proposing a KPIs tree structure. In [41], the authors present a high-level model for the design of KPIs. In doing this they highlight the limitations in using KPIs and the complexity of smart cities systems. Here, the focus is on the use of KPIs in decision-making processes.

6.3 Quality assessment approaches

We already highlighted how quality assessment of smart cities can be re-conducted to quality assessment in software engineering. The quality of a software is important since it may affect several aspects, such as human life and financial loss. These two aspects, in particular, are relevant also in the smart cities domain because a correct evaluation of KPIs for a city can affect its improvement. As regards the quality evaluation applied in software engineering (e.g., [42]), we can find several quality models that present a hierarchical

structure, as the KPIs metamodel we defined for our demonstration case (Fig. 4). In [43], one of the main issue in software quality assessment is highlighted, i.e., the lack of standardization in modeling software quality metrics. The motivation relies in the variety of application scenarios. Similarly, a lack of standardization in the modeling of KPIs and their calculation for different cities can be observed. Moreover, in the MDE field the quality assessment has to face another relevant issue, such as the continuously evolution of languages [44]. Again, this aspect can be reflected in the KPIs assessment for smart cities, due to the discussed KPIs evolution. In addition, while defining our approach, we experienced one of the obstacle as in [45], typical in model-driven settings. More precisely, we refer to the fact that quality models, such as our KPIs Metamodel in Fig. 4, rely on partially defined subjects since they model quality issues/metrics, instead of concrete concepts/phenomena.

6.4 Spreadsheet engineering

Given the use of spreadsheets in the decision-making context also previously highlighted in this work, we further survey spreadsheets-based approaches that can potentially be used in the performance assessment of smart cities. In industrial contexts, spreadsheets are widely exploited for decision-making purposes [46]. In this work, the authors argue that the quality of the used spreadsheets is very important because decisions taken upon wrong spreadsheets-based assumption may have serious economical impacts on businesses. This drawback is further stated in [47], where the lack of rigorous quality assurance mechanisms in the development of spreadsheets is highlighted. In order to solve the problem, the authors present an approach of Model-Based Diagnosis of faults in spreadsheets. In [15], the goal of the authors is to reduce the known error-proneness shown by spreadsheet-based approaches. In this direction, they introduce a new way of object-oriented modeling to generate and evolve spreadsheets before using them. In particular, they make used of so-called *ClassSheets*, namely object-oriented classes, used to generate concrete spreadsheets as class instances in such a way to manage and reduce sources of errors before the spreadsheets generation. Similarly, MDE techniques are exploited also in [48] to build spreadsheets models easy to evolve and validate. These works, besides confirming the drawbacks of spreadsheets as already discussed in this work, highlight the potential of using MDE techniques, as widely done in our approach, already supporting model validation, error detection, code generation and evolution without the need of developing dedicated approaches to add these features to spreadsheets-based frameworks.

7 Conclusion and future work

In this paper, we presented *MIKADO*, an approach for the definition, automatic assessment, and visualization of KPIs over smart cities. In particular, the approach provides different artifacts for modeling smart cities and KPIs as well as an evaluation engine for the automatic KPIs calculation and an engine to render their visualization. This engine is also able to produce integrated KPI dashboards helpful in the decision-making process, actuated by the stakeholders. This work gives the complete realization of the overall approach described in Fig. 2 and is open to further extensions and improvements.

The goal of our approach is twofold: firstly, supporting smart cities administrators in evaluating the degree of smartness and sustainability, by providing them a global vision over their managed city through the automated and customized evaluation of KPIs they selected, and secondly, the approach is generic, and thus, equally applicable to different smart cities for their KPIs assessment. This enables a comparison among smart cities, as, for instance, periodically done by ranking agencies, by even increasing the trustworthiness of the approach. Understanding their city's level of smartness w.r.t. other cities may help administrators in understanding what are current strengths and weaknesses. Moreover, the round-trip nature of our approach, which goes from the KPIs evaluation, to the results interpretation back to the smart cities administrators, guarantees the stakeholders involvement and alerting.

As the current approach already showed promising evaluation results, we present in the following interesting lines for future work.

Smart Cities Comparison. Concerning the visualization part, the generated dashboards can be used to enable comparisons between different smart cities. The use of intuitive charts like those in Fig. 5 supports a fast and easy understanding of the indicators. For instance, by defining the same set of KPIs against which to assess a set of smart cities, the tool can be used to build smart cities rankings showing their level of *smartness*.

KPIs Continuous Monitoring and Forecasting. The runtime monitoring of KPIs is currently not handled by our approach that instead computes preliminary KPIs evaluation on the current status of smart cities only. However, we plan to extend our approach such that it is triggered at real-time by changing values in the KPIs input parameters, thus performing the KPIs evaluation periodically or in a reactive manner. Open data APIs and exposed Web services can be used as real-time sources. For instance, concerning our demonstration case, we plan to continuously inject model parameters by requesting data from a publicly available repository, i.e.,

OpenData L'Aquila.¹⁹ Moreover, it might be that real-world data required for the KPIs calculation is not available. When this happens, estimated data can be used. As future work, we are interested in investigating how to leverage both real-world and estimated data to create forecast models that can predict the expected KPIs of a smart city [49,50] and can be used in combination with our approach, to increase its accuracy.

Simulations and KPIs Interrelations. The smart city governance system, besides using the proposed approach to evaluate and make considerations about the calculated KPIs, can also exploit modeling tools to generate predictive and descriptive models of the city. This way, the infrastructure can be exploited also for simulation activities. Moreover, simulations may be performed also to investigate and evaluate the impact of planned changes on the KPIs for the city. For instance, if the mobility plan of the city includes the introduction of a new transportation mode (e.g., shuttle buses connecting peripheral areas), the city administrators might be interested in simulating its impact on, i.e., the number of private cars around the city. Moreover, a decrease in this value would also affect the *AP* KPI defined in formula (4). This leads to the analysis of KPIs interrelations, that is, in our opinion, enabled by our approach and may be performed through queries on the model artifacts we already provide.

Smart Cities and Digital Twins. Concerning simulation activities, it could be interesting to exploit *digital twin* approaches [51] in the smart city domain, due to their vision of simulation as a tool for merging the physical world and its virtual representation. On the one hand, making digital twin technology an integral part of model-based software engineering has been explored in [52]. On this basis, we can think to exploit digital twins in our approach to perform dynamic and real-time KPIs evaluation. On the other hand, using model-driven engineering for developing digital twins as described in [53] would allow to also exploit our models for deriving digital twins.

Historical Smart Cities Models. The presented tool may enable not only comparisons with other cities but also with values related to different periods of time on the same city. In this way, the stakeholders can reason about the trends of the different KPIs calculated over the city. This could be further supported, for instance, by integrating some existing tools to support versioning of KPIs models, e.g., TemporalEMF [54], in order to monitor the evolving indicators. Of course, since there is synchronization between the instantiated KPIs model and the dashboards view, the consequences of a change of a KPI value in the model can be immediately seen in the dashboard. This means that the tool can be used also to make simulations by changing the calculated KPIs values of the

candidate city, supporting stakeholders in making assumptions during decision-making processes.

Modeling and Reasoning on Decisions. Moreover, the presented approach implicitly supports the decision-making process, by providing KPIs value describing the city that can be exploited by smart city managers to reason about decisions to be taken. It would be interesting for future work to explicitly relate KPIs to decisions, as for instance done for software systems and architectures [55,56].

Further empirical studies. Further efforts will aim to model the complete list of KPIs provided by the standard guidelines we refer to (i.e., [6–8]). We also plan to evaluate the approach over a larger number of smart cities of different sizes in order to verify its accuracy. Finally, we plan to exploit Technology Acceptance Models (TAM) to evaluate the usability of our approach not only in terms of perceived understandability but also including the perceived ease-of-use and usefulness of the defined languages and tools.

Acknowledgements We want to thank Dimitris Kolovos, from the University of York (UK), for giving support for technical aspects of this project. This work was partially supported by the Centre for Urban Informatics and Modelling (CUIM) and the PON (Programma Operativo Nazionale Ricerca e Innovazione), AIM1880573 Cultural Heritage, National Projects at GSSI as well as by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG).

Funding Open access funding provided by Gran Sasso Science Institute - GSSI within the CRUI-CARE Agreement.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: snippet of the KPIs definition model with textual concrete syntax

```

1  KPIModel mykpi ;
2
3  Dimension Environment {
4      Category Public Spaces & Nature {
5          KPI GA {
6              values : AggregatedRealValue : DIV {
7                  RealValue : TotalGreenArea
8                  AggregatedRealValue : CEN {
9                      RealValue : CityPop
10                 }
11             }
12         }
13     }

```

¹⁹ <http://opendatalaquila.it>

```

12     }
13   }
14 }
15
16 Dimension Economy{
17   Dimension Infrastructure{
18     Category Transport{
19       KPI BN{
20         values: AggregatedRealValue: DIV{
21           RealValue: BykePathLength
22           AggregatedRealValue: CEN{
23             RealValue: CityPop
24           }
25         }
26       }
27     }
28   }
29 }
30
31 Dimension Planet{
32   Category Energy & Mitigation{
33     KPI CO2 emissions{
34       values: AggregatedRealValue: DIV{
35         RealValue: CO2,
36         RealValue: CityPop
37       }
38     }
39   }
40   Category Pollution & Waste{
41     KPI NO2 emissions{
42       values: AggregatedRealValue: DIV{
43         RealValue: NO2,
44         RealValue: CityPop
45       }
46     }
47     KPI PM2.5 emissions{
48       values: AggregatedRealValue: DIV{
49         RealValue: PM2.5,
50         RealValue: CityPop
51       }
52     }
53   }
54 }
55
56 Dimension Digital Competencies{
57   Category Competencies{
58     KPI MA{
59       values: AggregatedIntegerValues:
60         SUM{
61           IntegerValue: MobileApplicationsPS
62         }
63     }
64   }
65 }
66 Dimension Infrastructure{
67   Category Digital Infrastructure{
68     KPI TM{
69       values:
70         BoolValue:
71           RealtimeTransportMonitoring
72     }
73 }

```

References

1. ITU-T focus group on smart sustainable cities: smart sustainable cities: an analysis of definitions, 2014. Available at: <https://bit.ly/324U929>
2. Visvizi, A., Lytras, M.D., Damiani, E., Mathkour, H.: Policy making for smart cities: innovation and social inclusive economic growth for sustainability. *J. Sci. Technol. Policy Manage.* **9**(2), 126–133 (2018)
3. Mutiara, D., Yuniarti, S., Pratama, B.: Smart governance for smart city. *IOP Conf. Ser. Earth Environ. Sci.* **126**, 12–73 (2018)
4. Science Communication Unit, UWE, Bristol. Science for environment policy: indicators for sustainable cities, April 2018. In-depth Report 12. Produced for the European Commission DG Environment. Available at: <https://bit.ly/3aMjgMK>
5. European Commission: Europe 2020 A European strategy for smart, sustainable and inclusive growth, March 2010. Available at: <https://bit.ly/2R8siwl>
6. International Telecommunication Union (ITU): Collection Methodology for Key Performance Indicators for Smart Sustainable Cities, 2017. <https://www.unecce.org/fileadmin/DAM/hlm/documents/Publications/U4SSC-CollectionMethodologyforKPIfoSSC-2017.pdf>
7. Bosch, P., Jongeneel, S., Rovers, V., Neumann, H.-M., Airaksinen, M., Huovila, A.: Citykeys indicators for smart city projects and smart cities (2017) Available at: <https://nws.eurocities.eu/MediaShell/media/CITYkeystheindicators.pdf>
8. Romeo, S., Di Gregorio, M., Alfonso, U., Tozzi, A., Tarquini, F., Tomassoni, F.: Digital cities challenge - assessment report for the city of l'aquila (2019). Available at: <https://bit.ly/32YBgjC>
9. Ferro, E., Caroleo, B., Leo, M., Osella, M., Pautasso, E.: The Role of ICT in Smart City Governance. Presented at the (2013)
10. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, 2nd edn. Morgan and Claypool Publishers, San Rafael (2017)
11. Rosique, F., Losilla, F., Pastor, J.A.: A domain specific language for smart cities. In Proceedings of the 4th International Electronic Conference on Sensors and Applications (2018)
12. Abu-Matar, M.: Towards a software defined reference architecture for smart city ecosystems. In Proceedings of the IEEE International Smart Cities Conference (ISC2), pp. 1–6, (2016)
13. Marijuan, A.G., Etmnan, G., Moller, S.: Smart cities information system: key performance indicator guide v.2.0 (2017). Available at: <https://smart-cities-marketplace.ec.europa.eu/resources/1441>
14. Hara, M., Nagao, T., Hanno, S., Nakamura, J.: New key performance indicators for a smart sustainable city. *Sustainability* **8**(3), 206 (2016)
15. Luckey, M., Erwig, M., Engels, G.: Systematic evolution of model-based spreadsheet applications. *J. Vis. Lang. Comput.* **23**, 267–286 (2012)
16. Fowler, M.: Domain-Specific Languages. Pearson Education, London (2010)
17. De Sanctis, M., Iovino, L., Rossi, M.T., Wimmer, M.: A flexible architecture for the key performance indicators assessment in smart cities. In Proceedings of the 14th European Conference on Software Architecture (ECSA) (2020)
18. Basciani, F., Rossi, M.T., De Sanctis, M.: Supporting smart cities modeling with graphical and textual editors. In Proceedings of the 1st International Workshop on Modeling Smart Cities (MoSC) (2020)
19. Microsoft Research. Key Performance Indicators (KPIs) in Power Pivot (2019). <https://bit.ly/37EFR9r>
20. Catarina, S., Carla, S., Ana L.A.: INTELI – Inteligência em Inovação, Centro de Inovação. Urban Indicators and the Smart City Agenda (2016). Available at: https://pocacito.eu/sites/default/files/POCACITO_PolicyBrief_No-5_Urban_Indicators_1612.pdf
21. Barns, S.: Smart cities and urban data platforms: designing interfaces for smart governance. *City Cult. Soc.* **12**, 5–12 (2018)
22. Curino, C.A., Moon, H.J., Zaniolo, C.: Graceful database schema evolution: the prism workbench. *Proc. VLDB Endow.* **1**(1), 761–772 (2008)

23. Basciani, F., Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: A tool-supported approach for assessing the quality of modeling artifacts. *J. Comput. Lang.* **51**, 173–192 (2019)
24. Burgueño, L., Mayerhofer, T., Wimmer, M., Vallecillo, A.: Specifying quantities in software models. *Inf. Softw. Technol.* **113**, 82–97 (2019)
25. Colin, A., Thomas, K.: The essence of multilevel metamodeling. In Martin, G., Cris, K., (eds) *Proceedings of the International Conference on the Unified Modeling Language (UML)*, pp. 19–33. Springer, Berlin (2001)
26. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: The epsilon object language (EOL). In Arend, R., Jos, W. (eds.) *Proceedings of the European Conference on Model Driven Architecture-Foundations and Applications (ECMFA)*, pp. 128–142. Springer, Berlin (2006)
27. de la Vega, A., Kolovos, D., Cooper, J.: Efficient generation of graphical model views via lazy model-to-text transformation. In *Proceedings of the 23rd ACM/IEEE International Conference on Model-Driven Engineering, Languages and Systems (MODELS)* (2020)
28. Rose, L.M., Paige, R.F., Kolovos, D.S., Polack, F.A.C.: The epsilon generation language. In *Proceedings of the European Conference on Model Driven Architecture-Foundations and Applications (ECMFA)*, pp. 1–16. Springer, Berlin (2008)
29. Brottier, E., Fleurey, F., Steel, J., Baudry, B., Traon, Y. L.: Metamodel-based test generation for model transformations: an algorithm and a tool. In *Proceedings of the International Symposium on Software Reliability Engineering*, pp. 85–94 (2006)
30. Dennis, W., Ludovico I., Di Ruscio, D., Alfonso P.: Translational semantics of a co-evolution specific language with the emf transformation virtual machine. In *Proceedings of the International Conference on Theory and Practice of Model Transformations (ICMT)*, pp. 192–207. Springer, Berlin (2012)
31. Tihamer, L., Daniel, B., Anantha, N., Gabor K.: A novel approach to semi-automated evolution of DSML model transformation. In *Proceedings of the International Conference on Software Language Engineering (SLE)*, pp. 23–41. Springer, Berlin (2009)
32. Svetits, M., Zdun, U.: Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Softw. Syst. Model.* **15**(1), 31–69 (2016)
33. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: The epsilon transformation language. In *Proceedings of the International Conference on Theory and Practice of Model Transformations (ICMT)*, pp. 46–60. Springer, Berlin (2008)
34. Martin, W., Jürgen, E., Johannes, S., Angelika, K., Elisabeth, K., Wieland, S., Manuel, W.: CARE: a constraint-based approach for re-establishing conformance-relationships. In *Proceedings of the Tenth Asia-Pacific Conference on Conceptual Modelling (APCCM)*, pp. 19–28 (2014)
35. Kolovos, D.S., Rose, L.M., Matragkas, N.D., Paige, R.F., Guerra, E., Cuadrado, J.S., de Lara, J., Ráth, I., Varró, D., Tisi, M., Cabot, J.: A research roadmap towards achieving scalability in model driven engineering. In Di Ruscio, D., Kolovos, D.S., Matragkas, N. (eds.) *Proceedings of the Workshop on Scalability in Model Driven Engineering*. ACM (2013)
36. Viswanath, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: toward a unified view. *MIS quarterly*, pp. 425–478 (2003)
37. Memik, M., Heering, J., Sloane, A.: When and how to develop domain-specific languages. *ACM Comput. Surv.* **37**, 316–344 (2005)
38. Benoît, C., Bruel, J.-M.: Applying model-driven engineering to the development of smart cyber-physical systems (2017)
39. Angelakoglou, K., Nikolopoulos, N., Giourka, P., Svensson, I.-L., Tsarchopoulos, P., Tryferidis, A., Tzovaras, D.: A methodological framework for the selection of key performance indicators to assess smart city solutions. *Smart Cities* **2**(2), 269–306 (2019)
40. Estrada, E., Maciel, R., Negrón, A.P.P., López, G.L., Larios, V., Ochoa, A.: Framework for the analysis of smart cities models. In: Jezreel, M., Mirna, M., Álvaro, R., Adriana, P., Marco, P.-C. (eds.) *Trends and Applications in Software Engineering*, pp. 261–269. Springer, Berlin (2019)
41. Orłowski, C., Ziółkowski, A., Orłowski, A., Kaplanski, P., Sitek, T., Pokrzywnicki, W.: High-level model for the design of KPIs for smart cities systems. *Trans. Comput. Collect. Intell.* **25**, 1–14 (2016)
42. Samadhiya, D., Wang, S.-H., Chen, D.: Quality models: role and value in software engineering. In *Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE)*, pp. 320–324 (2010)
43. Deissenboeck, F., Juergens, E., Lochmann, K., Wagner, S.: Software quality models: Purposes, usage scenarios and requirements. In *Proceedings of the ICSE Workshop on Software Quality*, pp. 9–14 (2009)
44. Ma, Z., He, X., Liu, C.: Assessing the quality of metamodels. *Front. Comput. Sci.* **7**, 558–570 (2013)
45. Giraldo, F.D., España, S., Pastor, O., Giraldo, W.: Considerations about quality in model-driven engineering. *Softw. Qual. J.* **26**, 1–66 (2016)
46. Rui, A., Jácome, C., João, F., Pedro, M., Alexandre, P., Joao, S.: Smelling faults in spreadsheets. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME)*, pp. 111–120 (2014)
47. Dietmar, J., Thomas, S., Konstantin, S.: Toward interactive spreadsheet debugging. In *Proceedings of the First Workshop on Software Engineering Methods in Spreadsheets*, pp. 3–6 (2014)
48. Cunha, J., Fernandes, J., Mendes, J., Saraiva, J.: Embedding, evolution, and validation of model-driven spreadsheets. *IEEE Trans. Softw. Eng.* **41**, 241–263 (2015)
49. Thomas, H., Assaad, M., Francois, F., Gregory N., Jacques, K., Traon, Y.L., Jézéquel, J.-M.: Model-driven analytics: connecting data, domain knowledge, and learning. (2017) [arXiv preprint arXiv:1704.01320](https://arxiv.org/abs/1704.01320)
50. Zafar, M.N., Azam, F., Rehman, S., Anwar, M.W.: A systematic review of big data analytics using model driven engineering. In *Proceedings of the International Conference on Cloud and Big Data Computing (ICCBDC)*, pp. 1–5. ACM (2017)
51. Stefan, B., Rosen, R.: Digital twin—the simulation aspect. In *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*, pp. 59–74. Springer, Berlin (2016)
52. Madni, A., Madni, C., Lucero, S.: Leveraging digital twin technology in model-based systems engineering. *Systems* **7**(1), 7 (2019)
53. Francis, B., Benoit, C., Romina, E., van den Brand, M., Wimmer, M.: Towards model-driven digital twin engineering: current opportunities and future challenges. In *Proceedings of the International Conference on Systems Modelling and Management (ICSMM)*. Springer, Berlin (2020)
54. Gómez, A., Cabot, J., Wimmer, M.: TemporalEMF: a temporal metamodeling framework. In *Proceedings of the International Conference on Conceptual Modeling (ER)*, pp. 365–381. Springer, Berlin (2018)
55. Svetits, M., Zdun, U.: Architectural design decisions for systems supporting model-based analysis of runtime events: a qualitative multi-method study. In *Proceedings of the IEEE International Conference on Software Architecture (ICSA)*, pp. 115–124 (2018)
56. Jugel, D., Schweda, C., Zimmermann, A.: Modeling decisions for collaborative enterprise architecture engineering. In *Proceedings of the Advanced Information Systems Engineering Workshops*, pp. 351–362, (2015)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Martina De Sanctis is Assistant Professor at the Computer Science department of the Gran Sasso Science Institute (GSSI), in L'Aquila, Italy, where she has previously been postdoctoral researcher. Her research interests include behavioral and architectural adaptation of service- and IoT-based systems, collective aspects and modeling of multi-agent systems, dynamic adaptations and its application to several domains, i.e., mobility, smart cities, IoT, eHealth. She received a Ph.D. in Computer

Science at the Doctoral School in Information and Communication Technology (2018), from the University of Trento and Fondazione Bruno Kessler (FBK) in Trento. From 2013 to 2018, she was PhD fellow and researcher at FBK, at the Distributed Adaptive Systems research unit where she was working on approaches for the dynamic adaptation of service-based systems, with focus on automated service composition, and their application in different domains. During her PhD studies she actively participated in European Projects in the large-scale collective systems (ICT-FET Proactive project) and digital industry (EIT Digital project) sectors. She has been previously working in companies as software developer in the business sectors of Geographic Information Systems (GIS) and Web-based software applications.



Ludovico Iovino is currently Assistant Professor at the GSSI—Gran Sasso Science Institute, L'Aquila—in the Computer Science scientific area. His interests include model-driven engineering (MDE), Model Transformations, Metamodel Evolution, code generation and software quality evaluation. Currently, he is working on model-based artifacts and issues related to the metamodel evolution problem. He has been included in program committees and organization of numerous conferences as STAF and MOD-

ELS. He also organize the models and evolution workshop from 3 years. He is part of different academic and national projects related to Model Repositories, model migration tools, Eclipse Plugins and technology transfer.



She received a master degree in Business Informatics at the University of Pisa, Italy, where she completed an internship at the Knowledge Discovery and Data Mining Laboratory (KDD Lab) of the National Council of Research (CNR) of Pisa, as part of her master thesis project on Mobility Data Analytics.



[.jku.at/manuel-wimmer/](https://www.se.jku.at/manuel-wimmer/)

Maria Teresa Rossi is a PhD student at the Computer Science department of the Gran Sasso Science Institute (GSSI), in L'Aquila, Italy, working for the Center for Urban Informatics and Modeling (CUIM) group. Her research interests include model-driven engineering and its application in the Smart City domain. She received a master degree in Business Informatics at the University of Pisa, Italy, where she completed an internship at the Knowledge Discovery and Data Mining Laboratory (KDD Lab) of the National Council of Research (CNR) of Pisa, as part of her master thesis project on Mobility Data Analytics.

Manuel Wimmer is full professor leading the Institute of Business Informatics-Software Engineering at the Johannes Kepler University Linz, and he is the head of the Christian Doppler Laboratory CDL-MINT. His research interests comprise foundations of model engineering techniques as well as their application in domains such as tool interoperability, legacy modeling tool modernization, model versioning and evolution, and industrial engineering. For more information, please visit [https://www.se](https://www.se.jku.at/manuel-wimmer/)