Leveraging Multi-Level Modeling for Multi-Domain Quality Assessment

Maria Teresa Rossi*, Martina Dal Molin†, Ludovico Iovino*, Martina De Sanctis*, Manuel Wimmer‡
*Gran Sasso Science Institute, Computer Science Department, L'Aquila, Italy
{mariateresa.rossi,ludovico.iovino,martina.desanctis}@gssi.it
† Gran Sasso Science Institute, Social Sciences Department, L'Aquila, Italy
{martina.dalmolin}@gssi.it
‡ CDL-MINT, Johannes Kepler University, Linz, Austria
manuel.wimmer@jku.at

***Abstract*—Quality Evaluation Systems (QESs) are a class of software systems which receive evaluation requests and quality requirement specifications as inputs and produce results as outputs of an assessment process. This class of systems usually work with a *quality model* including quality definitions and metrics, and produce the output as a quantitative evaluation of a subject. QESs can be implemented using model-driven techniques and dedicated languages, for domain-specific evaluation of different subjects. However, applying traditional two-level metamodeling techniques for this scenario entails that every time a QES is required, a new modeling framework, with consequent engine for interpreting the newly defined models, must be re-developed from scratch. To overcome this repetitive process, in this paper, we propose a Multi-Level Modeling (MLM) approach for realizing the artifacts involved in the development phase of a QES which are reusable across multiple domains. We demonstrate the approach with running examples from three different application domains comprising different evaluation scenarios.**

***Index Terms*—Quality Evaluation Systems, Multecore, Multi-level Model Interpretation**

I. Introduction

Quality Evaluation Systems (QESs) are software systems providing quality evaluation results of given subjects, according to evaluation requests and quality requirements received as inputs. QESs basically carry out an evaluation process to accomplish their task, and may be implemented as a part of a higher-level system, as well as independent systems [1]. Such systems usually work with a *quality model* including quality definitions and corresponding metrics. In other words, QESs are measurement tools making use of evaluation techniques to produce quantitative evaluation of a subject as result.

Quality models have been used in software evaluation for decades as tools for assessing the degree to which a software product satisfies stated and implied needs [2]. A *quality model* is a model expressing quality as a set of characteristics, also called attributes, establishing relationships between them. A quality model relates to quality requirements and poses the bases for assessing the quality of a subject. A number of specialized tools have been proposed to assess the quality of different subjects, e.g., software systems [3], modeling artifacts [4]–[6], software architectures [7], to mention just a few. QESs have been often implemented for the domain-specific evaluation of different subjects. For instance, in our previous work [8], we have presented a reference architecture for Key Performance Indicators (KPIs) assessment of Smart Cities [9]. The system defined on top of it is realized as a QES where the subject of the evaluation is a smart city, and the quality model against which to perform the assessment is a KPIs definition model. It is worth to note that the assessment process shows similar characteristics even when the subject of the evaluation changes. The overall objective is commonly the quantitative assessment of a given subject passed as input, where the assessment is executed on top of a quality definition, required (and in best cases defined) by the user. Despite these commonalities, QESs are often re-implemented from scratch by using existing tools, e.g., spreadsheets, or developed as independent systems, e.g., Web or standalone applications. Domain-specific languages (DSLs) offer a degree of abstraction that helps in the definition process of both quality characteristics and evaluation subjects, so that the results of the process are more robust. This implies reducing the time to market, and facilitating the definition of the artifacts involved in the entire process. DSLs are often intended to be used by someone who is not a programmer and therefore they offer advantages similar to the usage of programming languages, but for domain experts. Although using traditional Model-Driven Engineering (MDE) approaches when realizing QESs for multiple subjects may already provide benefits, there is also a limitation that it is worth to be investigated. Indeed, there is an issue in applying traditional two-level metamodeling techniques for this scenario. Every time a QES is required, all the involved modeling artifacts, related to the subject of evaluation and quality definition, with consequent engine for interpreting these models, must be re-developed. Even if the benefits of traditional two-level metamodeling with respect to traditional code-centric techniques are still valid, the assessment process is analogous independently from the subjects under evaluations across different domains. Multi-level Modeling (MLM) provides an unbounded number of levels of abstraction [10], offering enough flexibility when specific patterns arise in modeling approaches [11].

In this paper, we use MLM to realize QESs for multiple domains. Specifically, we aim to transform a QES development approach based on two-level modeling into a multi-level based

one. Our goal is to define what we call a *multi-level framework* that allows to customize QESs for different domains. In this work, we present three different domains in which we applied the proposed approach and, subsequently, we show three different running examples for showing the usefulness of our proposal. In particular, by means of our running examples we demonstrate as an MLM approach guarantees the same expressiveness and modeling power of traditional two-level modeling techniques, while further pushing forward reuse and customization. As a consequence, our approach allows reducing time-to-market for the modeled systems and error-proneness.

Structure of the paper. The paper is organised as follows. Section II describes three candidate scenarios of application cases which are used as running examples throughout the paper. Section III presents our general architecture with the involved artifacts, explained in terms of the proposed hierarchies of models, and shows the results of the assessment process on the running examples. In Section IV we discuss related work whereas we draw some conclusions in Section V with an outlook on future work.

II. Background and Running Examples

Assessing the quality of a subject when supported by an automated tool is a process involving multiple artifacts and technologies. In our case, if we concentrate on MDE as technical space and inspired by the work presented in [8], [12], we define the assessment process as A_{req}:

$$A_{req}(Subject, Q_m) \rightarrow EQ_m \tag{1}$$

where A_{req} is the assessment request, given a $Subject$ and a quality definition expressed as a quality model Qm. The output of this request is an evaluated quality model EQm, where the requested quality characteristics are expressed quantitatively.

In this context, we have identified three possible case studies with commonalities in the request and in the expected result type, i.e., a quantitative analysis of the subject, that can be assessed by implementing a QES. These three case studies are used for the remaining of the paper as running examples.

A. Smart City KPIs evaluation

Smart governance exploits KPIs coming from standard guidelines (e.g., [13], [14]), to assess smart cities in terms of sustainability and smartness. KPIs assessment in smart cities is used to support decision-making processes in order to have a global vision of the city under analysis. An example of KPI that can be selected in this scenario, is called Green Areas (GA) in [13]. This particular indicator measures the green area in the city per 100.000 inhabitants. It is calculated as follows:

$$GA = \frac{TotalGreenArea}{\frac{1}{100000} \times CityPopulation} \tag{2}$$

The data needed to calculate KPIs in a smart city may come from different types of sources, e.g., open libraries, private repositories. Data is attached to the representation of the input subject for which the evaluation is performed, i.e., the smart city. KPIs are required in the evaluation as performance indicators of the subject and are part of the quality model passed as input to the evaluation request. Thus, we can define a relationship between the smart city and the subject using Definition (1) as:

$$instanceOf(SC, Subject) \wedge instanceOf(KPI_m, Q_m) \tag{3}$$

where SC is a definition of a smart city, and $instanceOf$ is the relationship of instantiation of a subject of the assessment process. The same relation persists between the requested KPIs, namely KPI_m, and the quality model, Q_m.

B. Research Institute Social Impact

In the context of Higher Education Institutions (HEIs), i.e., universities and research centers, the social impact is measured and assessed to demonstrate the positive direct and indirect impact on companies and societies generated by the research and third mission activities [15]–[17]. HEIs may generate positive impacts through a variety of activities, e.g., public engagement and technology transfer, and such impact may be evaluated using several impact dimensions, ranging from innovation, economic, technological and social. Given the increasing use of social media in HEIs, an interesting dimension to be considered is the distribution of social media content among the various departments of an institute. For example, Equation (4) reports the formula related to the calculation of the Content Distribution (CD) related to a department i of an HEI j.

$$CD_i = \frac{Media_i}{MediaTotal_j} \tag{4}$$

By using Definition (1), we can define the following relationship:

$$instanceOf(I, Subject) \wedge instanceOf(SM_m, Q_m) \tag{5}$$

where I is a representation of an institute to be evaluated (e.g., research center, school), and SM_m is a definition of a set of social metrics interesting for the candidate institute.

C. Covid-19 Risk

During the pandemic period, the Italian Ministry of Health defined 21 indicators [18] to monitor the transmission and impact of the virus in Italy. The monitoring was carried out at a regional granularity in order to assign a different level of risk to every region. One of these 21 indicators is the one that calculates the Intensive Care Unit (ICU) Occupancy rate (IO), as reported in Equation (6), by Covid patients ($Patients$) over the total number of available beds in ICU, namely $BedsICU$.

$$IO = \frac{Patients}{BedsICU} \tag{6}$$

The input data needed to calculate these risk indicators were collected by the different health institutes in the regions. In this scenario, the subject of the quality assessment is the region that has to calculate the 21 indicators composing, in this case, the instance of the quality model. The relationship between the region and the subject using Definition 1 is as follows:

$$instanceOf(Region, Subject) \wedge instanceOf(RI_m, Q_m) \quad (7)$$

where $Region$ contains the definition of the input parameters needed to calculate the risk indicators defined in RI_m.

D. Synopsis

The presented examples only show one measurement per case study, for sake of readability, but in reality these examples contain a large number of requested calculations with complex operations that are supported by the developed engine discussed in [12].

de Lara *et al.* identified a set of patterns explaining when MLM may be used successfully [11]. By analyzing these patterns, we found some of them in the scenarios exposed above and we discuss them in the following:

- *Type-object pattern*: this pattern allows the explicit modeling of types and their instances, where types are not static and may be dynamically included. As can be guessed from definitions (3), (5) and (7) this pattern persists in all three scenarios, since any modification to the *Subject*, for instance the addition of a new concept, reflects dynamically on the three considered subjects, i.e., smart city, institute and region. To give a concrete example in terms of MLM, a modification at the level $Subject$@2 in Fig. 2 is dynamically reflected at the level @1 in Fig. 2, where indeed the multi-level hierarchy for the subject definitions is modeled. A further example applying to all the considered domain refers to the dynamically addition of different *Data* types in the level @2, as soon as new data types are identified in the management of metrics. Then, this implies the creation of instances of the new data types in the lower levels (e.g., @0 in the multi-level hierarchy shaped in Fig. 2).
- *Dynamic features pattern*: this pattern enables the dynamic addition of new features to a type. Thus, it shares similarity with the previous one. It also persists in all the considered scenarios. This means that, by looking at Fig. 2, if we add a new feature to the *Subject* type at level @2, it is inherited by all the Subject's instances at the level @1.
- *Dynamic auxiliary domain concepts pattern*: it is a variant of the dynamic features pattern, in which, instead of the definition of features for dynamic types, the definition of dynamic entities in relation to dynamic types is allowed. In other words, as stated in [11], this pattern helps with the dynamic addition of new entities related to a type, as well as the instantiation of those entities, which should be correctly related to instances of the type. Also this pattern persist in all the scenarios. For instance, we may need an entity to describe the *repository systems* to which the *data sources* are connected. In this case we can dynamically add an entity called *Repository* and use it to describe the data repositories to which different *Source*s refer, by defining a relation between them.
- *Relation configurator pattern*: this pattern is related with the configuration of a reference type dynamically created, and its relative instantiation. It also persists in our scenarios. For instance, according to the example just discussed in the dynamic auxiliary domain concepts pattern, once we create the new *Repository* dynamic entity, we must consequently define a reference type to connect it to the *Source*s instances.
- *Element classification pattern*: this pattern permits the organization of dynamically created elements along subtyping and inheritance hierarchies. It also persists in our scenarios. Examples of this pattern can be easily guessed by looking at the examples given for the type-object pattern, where new elements were dynamically created and organized in the existing hierarchy in Fig. 2.

We can conclude that any case study requiring an assessment process which can be represented by Definition (1) is a good candidate for realizing a QES by exploiting the approach proposed in the next section.

III. Automated MLM-based Quality Assessment

This section introduces our approach for the qualitative assessment of subjects that can be represented with our MLM approach. The pre-requisites for using the proposed approach have been specified in Section II, where we have identified the patterns persisting in candidate quality assessment approaches. In the remainder of this section, we show how an MLM approach can leverage the definition and implementation of a QES.

We report the conceptual architecture of our approach in Fig. 1 and in the following we explore the constituent components and involved artifacts. In our approach, we use MultEcore [19], a modeling tool facilitating MLM on top of EMF (Eclipse Modeling Framework[1]). The tool allows in practice the instantiation of unlimited numbers of levels of abstraction. This means that we can instantiate multiple Ecore models as instances of other models coming from a more abstract level. MultEcore exploits text annotations to export the Ecore files and keep track of: (i) the root model from which the actual model is instantiated; (ii) the clabjects' ontological type coming from the higher levels of the hierarchy; (iii) the potency values of entities, references and attributes telling for how many levels they can be instantiated. Since MultEcore provides a graphical editor based on Sirius[2], it is possible to design models using a graphical representation showing all of the described annotations with different shapes and colours (e.g., blue circles for entities types, red boxes for entities minimum and maximum potencies: an example is shown in Fig. 2).

A. Multi-Level Subject Definition

On the left hand side of Fig. 1, we have the artifacts involved in the definition of a subject. These models are represented

[1]https://www.eclipse.org/modeling/emf/

[2]https://www.eclipse.org/sirius

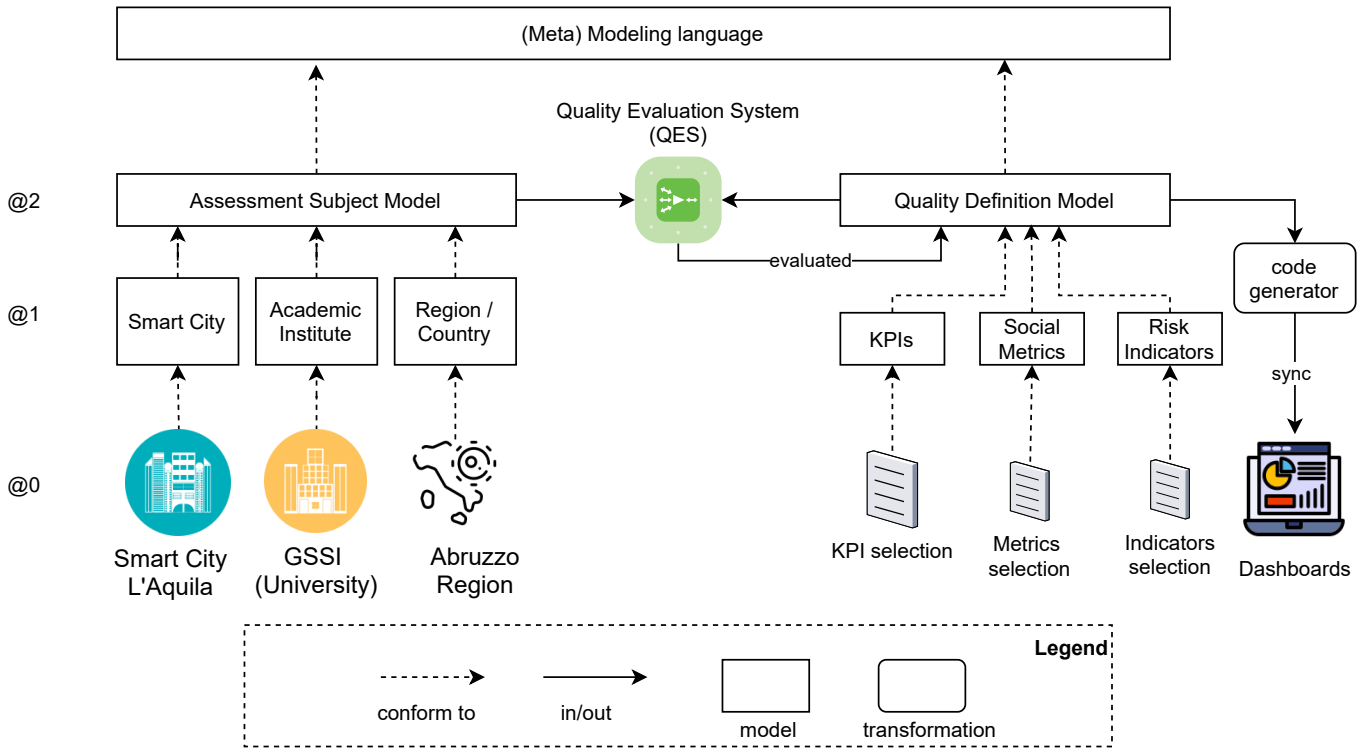

Fig. 1: Overview of the Multi-Level Quality Assessment Framework.

in the hierarchical organization reported in Fig. 2. Please note that they are not complete for sake of readability and understandability.

In this case we have a model for the definition of `Subject` that is also reported in Fig. 2 @2. In this model, a `SubjectUnderEvaluationModel` can be defined by adding a `Subject`. The model describes: (i) which are the `Sources` (e.g., `Stakeholders`) of the information collected about the subject; (ii) the type of the `Data` (i.e., `StringValue`, `IntegerValue`, `RealValue`, `BooleanValue`); and (iii) how the information is organized (i.e., `DataPackage`).

This model can be instantiated at the level @1 for our case studies. The `Subject` can be instantiated as different objects, i.e., `SmartCity`, `Institute`, or `Region`, by giving place to different models of three different domains, depending on the subject we want to evaluate. These three models at level @1 are reported as excerpts, so we only discuss here the relevant concepts and objects. In particular, for the smart cities domain, we instantiated the `Source` ontological type into three different data sources, namely `ProvidedData`, `OpenData`, `SocialMedia`. Moreover, we applied a linguistic extension introducing types (i.e., `PublicInfrastructureLayer`, `IoTDevice`, `Actuator`, `Sensor`) that allow the description of IoT systems such as `MonitoringInfrastructures` as data sources. With regard to the HEIs domain, we assume that the information is organized w.r.t. the various `Departments` of the institute and that they are collected by dedicated `Offices`. Meanwhile, for the regional risk monitoring domain, we assume to have two types of stakeholders, namely the `PublicAdministration` and `Hospitals`.

Level @0 represents real world objects / systems and in this case, we have defined 3 models (again partially reported). The first on the left-hand side represents a specific Smart City, i.e., `L'Aquila` in our running example. It is worth noting that we defined two data packages describing two collections of information (i.e., `CityStatistics`, `GreenAreas`). Besides the definition of the pure information indicating the `CityPopulation` and the `TotalGreenArea`, we annotated the providers of these information (i.e., `CityCouncil`, `AtlanteComuni`). In the center of the level @0 we have reported a model instantiating an institute (model at level @1), i.e., `GSSI`, that is the subject for the social impact scenario. Here, we assume that the information are organized w.r.t. the different departments (i.e., `ComputerScience`, `UrbanStudies`, `Mathematics`, `AstroparticlePhysics`) they describe. Moreover, the information collected for every department is about the contents shared on social media by them (e.g., `MediaCS` represents the social media content from the `ComputerScience` department). We assume that this type of data is provided by the `Administration` office. On the right, we have the model defining a specific region, i.e., the `Abruzzo` region, that is the subject of our assessment for the COVID scenario. `Abruzzo` is an instance of `Region` and collects information about `Healthcare`. The data provided by the `AgenziaSanitaria` is about the number of COVID

patients recovered in ICU (i.e., `Patients`) and the total number of beds in ICU (i.e., `BedsICU`).

B. Multi-Level Quality Definition

On the right hand side side of Fig. 1, we have the hierarchy of models used to define the other input of an evaluation request, i.e., the quality model. This hierarchy is organised in three levels, depicted in Fig. 3, where @2 defines the abstract language to define quality models, @1 allows the definition of different types of models for evaluating different aspects of quality, e.g., KPIs, social metrics and risk indicators. @0 defines the level for defining models instances of @1, so in our case we will define a selection of KPIs as well as metrics for social impacts and risk indicators. We could also consider that this level allows the definition of models containing all the possible aspects to be evaluated for an application domain, thus an additional level may be used to select some of the metrics with a sort of model slicing technique, to offer a query-selection on the entire quality model. This would permit to have a set of complete models, but also the possibility of selecting a subset of indicators we need for the selected subject. In detail, in Fig. 3, the model in level @2 provides the `Dimension` type to allow the organization of the `Metrics` and the input `Parameters` needed in the calculations. Every metric is associated to an output `Value`, that in turn has a `ValueType`. This type is specialized in `SingleValue` and `AggregatedValue`. The former is used in association with the parameters of calculations and can be of different types (i.e., `StaticRealValue`, `RealValue`, `IntegerValue`). The latter defines the typical operations that can be used in the metrics calculations (e.g., MAX, AVG) and can have different types (i.e., `AggregatedRealValue`, `AggregatedIntegerValue`).

At level @1, the described model has three instances, one for each domain of interest, among those described in Section II. In particular, in the `KeyPerformanceIndicators` model we instantiated the `Kpi` type and added other single and aggregated values (i.e., `BoolValue`, `StringValue`, `AggregatedBoolValue`, `AggregatedStringValue`, `AggregatedRangedValue`). Also the other two models at level @1 (i.e., `SocialMetrics`, `RiskIndicators`) are used to specify which types of metrics are used in the evaluation of the different domains.

We instantiated the models in @1 in level @0, by reporting the definition of a metric for each domain. In particular, the `GreenAreas` model describes how the KPI is calculated and which parameters it needs (see Equation (2)). Indeed, we associated to the considered dimension `Environment`, the parameters `TotalGreenArea` and `CityPopulation`. And then, we have an aggregated real value `DIV` associated to the value of the parameter `TGA` and another aggregated real value `CEN`, that in turn is associated to the other parameter value `CP`. We can see that this structure is repeated also in the other two models (i.e., `ContentDistribution`, `ICUoccupancy`). We have the description of the formulas used to calculate the metrics through the exploitation of the aggregated values specifying operations and parameters. This means that for the green areas KPI, the formula requires a division (`DIV`) between the total green area value and 1/100.000 (`CEN`) of the city population, as reported in Equation (2).

These models at level @0 are passed as parameter to the implemented QES we described above, with also the relative models on top of the hierarchy, so the engine can explore the hierarchical organization vertically by traversing the models.

Lastly, we remark here that, according to the MultEcore syntax, in both hierarchies in Figs. 2 and 3, red boxes report the potencies of the corresponding clabjects. In particular, the first one defines the *minimum potency*, i.e., from which level a clabject can be instantiated. The second one, instead, defines the *maximum potency*, i.e., up to which level you can instantiate the clabject. For instance, in the hierarchy in Fig. 2, the class `Source` at level @2 is annotated with a potency of 1-2-*, meaning that it can be instantiated from the level below (i.e., @1) and up to two levels downward (i.e., up to @0).

C. QES Engine

The quality evaluation system that we have implemented is a model interpreter, able to read the input models, i.e., the model representing the subject of the evaluation and the quality definition model, and produce in output an *evaluated* quality model. This engine has been implemented in [12] specifically for the Smart City case study (two-level modeling) and it is still under development for adapting it to the proposed MLM approach. This engine works as a model interpreter parsing the requests and actualizing the results on-the-fly. The result is based on the required quality model, actualized with quantitative evaluations, that will be synchronized with a code generator producing graphical dashboards. These dashboards (examples are available in Subsection III-D) can be used to practically inspect and analyze the results for decision making processes. Actually, the generated dashboards are very productive and useful since without this representation, the modeler would manually have to inspect the models to check the results, resulting in a very cumbersome and error-prone process. The generation of the dashboards is not part of this work and the interested reader is kindly referred to [12] for further details.

The QES in Fig. 1, as already discussed, takes as input two models, one representing the subject, and the other including the quality characteristics needed in the evaluation request. Listing 1 reports a few lines of the QES implemented in Java + Epsilon Object Language (EOL) [20]. EOL is the core expression language of the Epsilon framework. EOL is used as a general-purpose standalone model management language for automating tasks, in our case manipulating and querying the models.

```
1 var root=qm0!EClass.all.selectOne(c|c.name="Root");
2 var qm1packagename =
      root.eAnnotations.selectOne(ea|ea.source.
3         matches("om=[A-Za-z]*")).source.split("=").second;
4 var mm=qm1!EPackage.all.selectOne(p|p.name=qm1packagename);
5 var metricclass=
      mm.eClassifiers.selectOne(c|c.eAnnotations.source.
6         flatten().includes("type=Metric"));
```

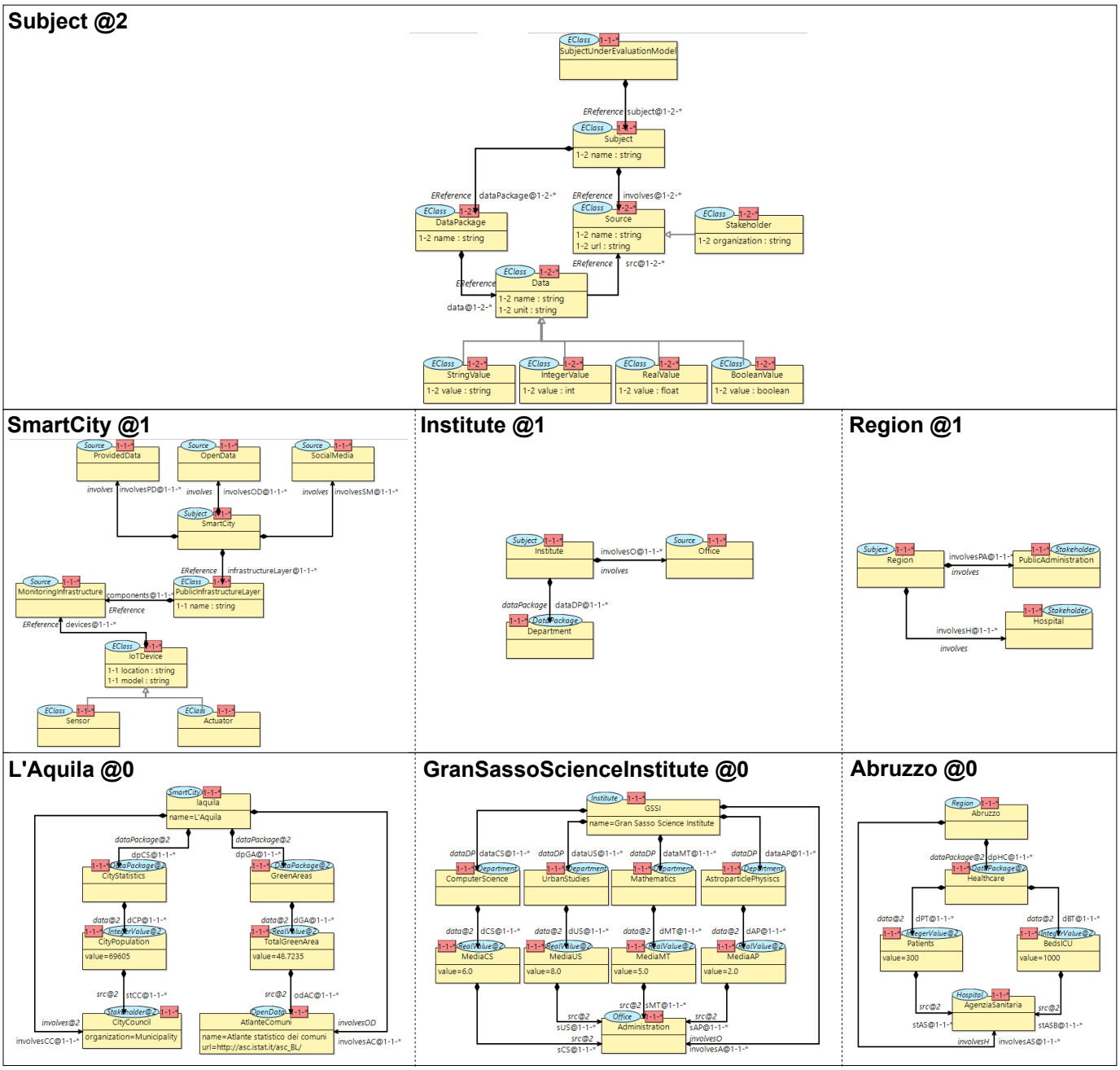

Fig. 2: Multi-Level Hierarchy for Subject Definitions.

```
7  var qm0metric=metricclass.name;
8  ...
9  for (metric in getMetrics(qm0metric)) {
10   ("Calculating..."+qm0metric+"-->"+metric.name).println();
11   var value=metric.getValue(subject);
12 }
13 ...
14 operation getMetrics(qm0metric:String){
15     var metrics=qm0!EClass.all.select(c|c.eAnnotations.
16         source.flatten().includes("type="+qm0metric));
17     return metrics;
18 }
19 ...
```

Listing 1: Snippet of the EOL-based implementation of the QES Engine.

This script is invoked from a Java launcher passing as parameter all three input models of the three levels @0 – @2 of both side of Fig. 1, for a total of 6 models. For instance the quality model is loaded by its `Root` at line 1. The object Root is a format that MultEcore uses as first instance containing all the objects in the model. In this script, we navigate the input models belonging to different levels by using the variables $qm0$, $qm1$ etc, referring to the level in the model's name. For instance, the expression $qm0!EClass.all$ will get all the objects instantiated at level @0 and so on. This model contains the definition of the KPIs or the SocialMetrics or

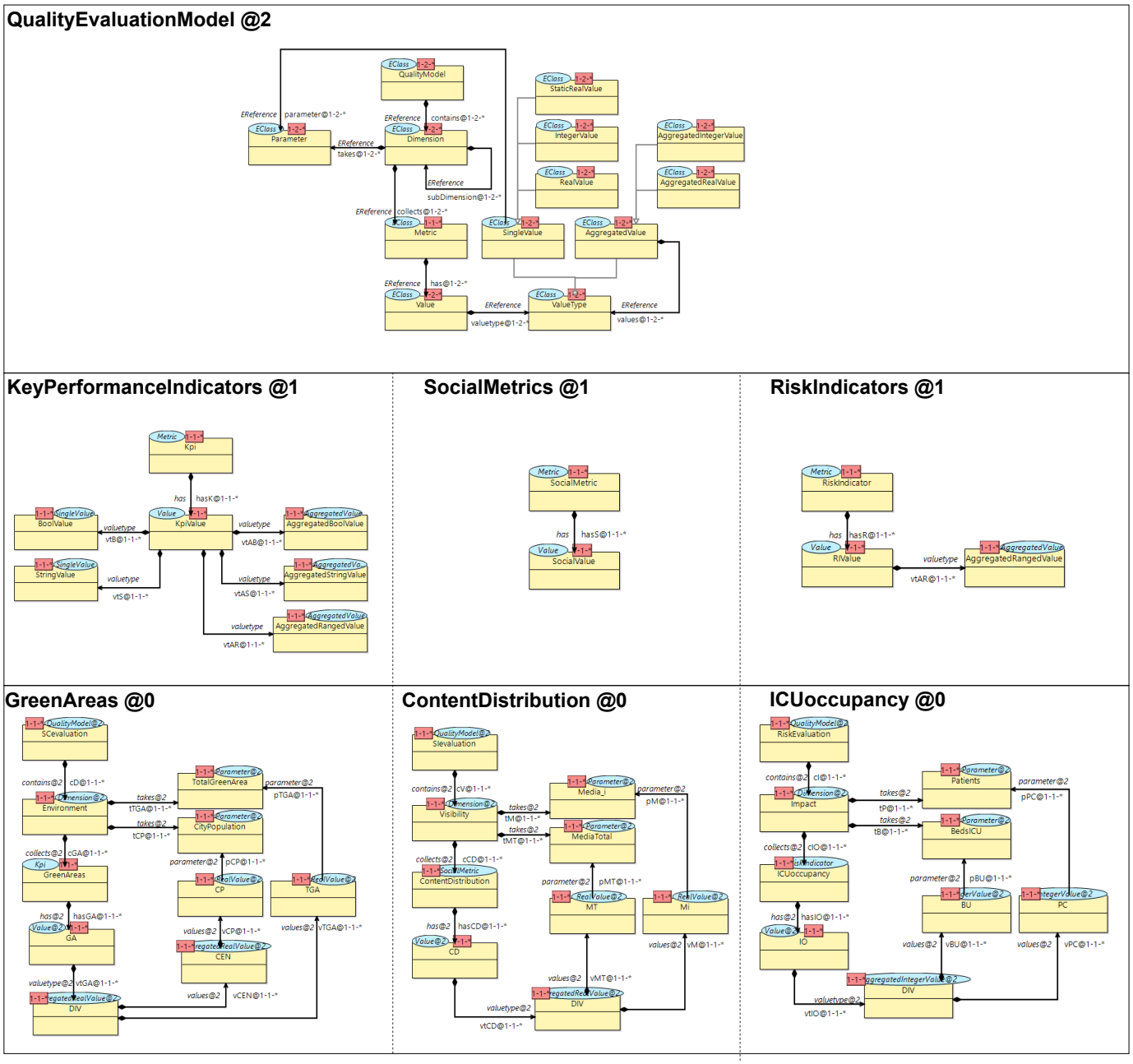

Fig. 3: Multi-Level Hierarchy for Quality Definitions.

the RiskIndicators, e.g., GreenAreas, or ContentDistribution or ICUoccupancy (see Fig. 3—models of level @0). Line 2 loads the package name used by the model at level @0. The operation of selection at line 3 gets the model at level @1 and it can be seen as a conformance retrieval between the two levels. In this case, the conformance relationship is persisted without a strongly typed relation like in two-level modeling but it is based on annotations, as we will see in the other lines. Indeed, this model query is executed using regular expressions, since MultEcore, as anticipated, uses String-based annotations for dynamic typing. For this reason, at line 4, we retrieve all the clabjects instantiating the `Metrics` at level @1. This operation will get the clabjects for KPIs, RiskIndicators and SocialMetrics, and whatever is defined at @1 as instance of metric. This embodies exactly what is defined as *Dynamic auxiliary domain concepts pattern* [11]. What is defined at level @1 as domain-specific metric is retrieved at lines 4 and 5. Then lines 7–10 applies the calculation for the metrics on the subject. If we launch this script on the models related to the case study in Subsection II-A, we receive in output what can be seen on top of Fig. 4. The console log demonstrates the dynamic binding of the defined KPIs as instances of Metrics. The same happens when passing the other models at level @0 in Fig. 3. The operation `getValue` will actualize the value

by using the defined operation in the given quality model[3]. This actualization of the result can be inspected directly in the model by looking at specific values of the metric (see `Value` class in Fig. 3). Examples of results obtained by executing the QES on the models explained in this section are shown in the next section.

D. Assessment Result for Running Examples

In this section, we show the results of the assessment of the three subjects explained in Section II with related assessment scenarios. In particular, we show some excerpts of the generated dashboards where the graphical charts support the understanding of the performances of the reported metrics. These dashboards are based on specific model to code transformation, generating HTML + Javascript files, synchronized with the @0 models. This means that every time the assessment is concluded, the dashboards are automatically updated. All these artifacts are included in a dedicated framework [12] developed as an Eclipse plugin. Indeed, the `Metric` has an attribute called `targetvalue` that indicates the optimum/desired value for the corresponding metric w.r.t. which the charts are generated. The running example in Section II-A has been applied on the city of L'Aquila, as drafted in the model in Fig. 2 @0. The result, reported in the generated gauge (part of the generated dashboard) in Fig. 4, shows the KPI *Green Areas* and the result of the measurement for the given smart city, i.e., 19%. Figure 5 is the result of the generated dashboard for the social impact scenario, applied to the GSSI (Gran Sasso Science Institute), Computer Science department, showing that it performs the 28% of the total social content distribution. Eventually, in Fig. 6, the generated gauge shows the evaluation of the COVID risk applied on the Abruzzo region. The ICU occupancy rate resulted 30% in that specific time of the assessment. These gauges are part of more complex dashboards that are automatically generated in sync with the assessed subject model.

It is worth noting that, although these three assessment results are quite trivial and might be considered meaningless, in the context of a wider evaluation (e.g., smart cities ranking, pandemic spreading) they contribute in giving a global view about the performance of the evaluated subjects. For instance, according to the three running examples described in Section II, the calculation of the Green Areas KPI on top of multiple smart cities would allow them to be ranked in terms of land use. Measuring the social media content distribution of multiple departments of a research institute would help to understand which one contributes the most or the least, if any, on the social impact of the institute. This information would help the departments, as well as the institute itself, to understand their generated impact on the society and where and how to invest in order to improve their contribution to the community. Eventually, applying the assessment of the ICU occupancy rate to multiple regions would allow the monitoring of the pandemic spread all over the country.

[3]For further details about the calculus, the interested reader is kindly referred to [12].

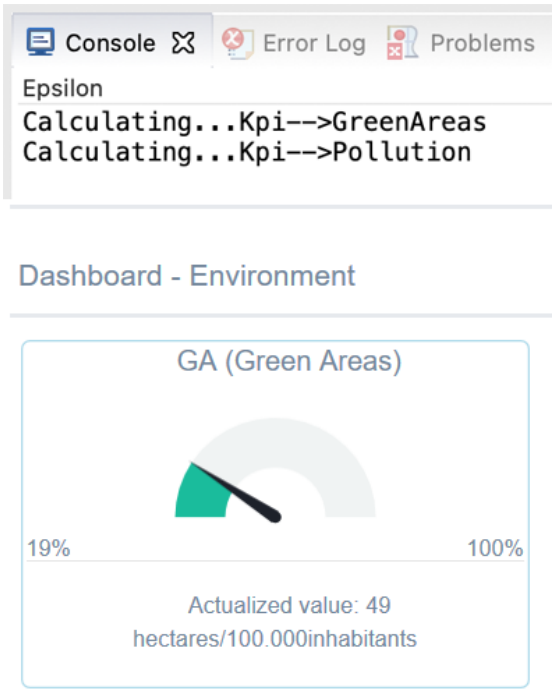

Fig. 4: Excerpt of the Dashboard evaluating Smart City KPIs.

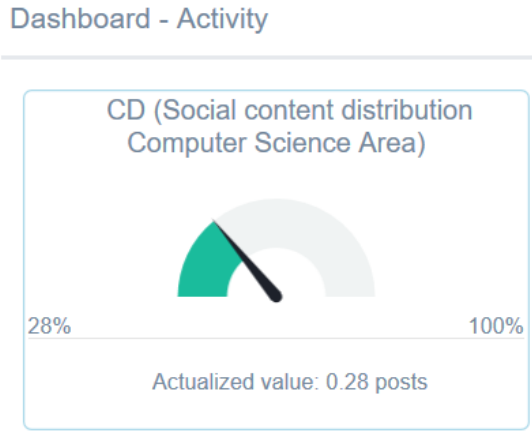

Fig. 5: Excerpt of the Dashboard evaluating Social Impact.

In conclusion, MLM in this case allowed us to work with different domains in which the subject of the assessment and the required measurements are different and can be all defined with our approach. This approach has been demonstrated with 3 of the possible case studies but it can be further applied to other domains to assess the expressiveness of the approach.

IV. Related Work

In this work, we proposed the quality assessment of different subjects from three specific domains. However, we can find some similarities with quality assessment in software engineering. Indeed, the quality of a software is of relevant importance since it may affect several aspects, such as human life and financial loss. These two aspects, in particular, are relevant also

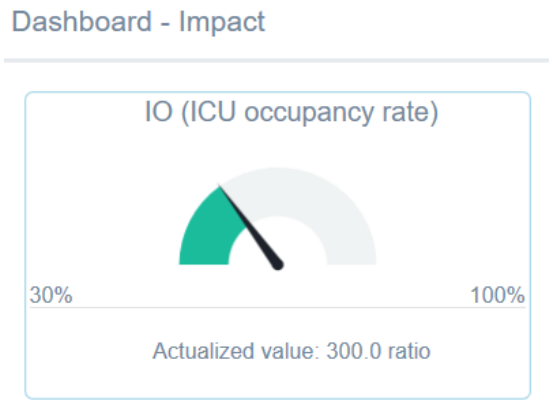

Fig. 6: Excerpt of the Dashboard evaluating COVID Risk.

in our candidate domains, since the related quality assessments aim to improve people quality of life in different aspects (e.g., sustainability, economy, health). An example of measurement mechanism for evaluation in software development is given by the Goal-Question-Metric (GQM) approach by Van Solingen *et al.* [21]. It represents a top-down approach based on goals and models applied to all phases of the software development process to evaluate the quality of both processes and products. Specifically, the GQM approach starts from the definition, by the organization exploiting the approach, of the goals for itself and its projects. Then, the defined goals must be traced to the data that operationally define those goals. Lastly, it provides a framework for interpreting the data with respect to the stated goals. Thus the resulting measurement model is organised as a hierarchy made by three levels, namely conceptual (i.e., goal), operational (i.e., question) and quantitative (i.e., metric). As regard quality evaluation applied in software engineering (e.g., [22]), we can find several quality models that present a hierarchical structure, as the one we defined in our quality model through the insertion of the type `Dimension` (Fig. 3). In the software quality assessment context one of the main issue is the lack of standardization in modeling software quality metrics, as highlighted by Deissenboeck *et al.* [23]. This is due to the fact that the application scenarios are very heterogeneous. Similarly, we can observe a lack of standardization in the modeling of quality metrics and their calculation in all the candidate domains of our scenarios. Moreover, in the MDE field the quality assessment has to face the continuous evolution of languages as described by Ma *et al.* [24]. This aspect can be reflected in the quality assessment, due to the evolution of metrics (e.g., new metrics can appear, existing metrics can be modified especially in their calculation formulas). For instance, standards defining KPIs to assess smart cities continuously evolve over time [25], accordingly to the evolution of cities. Lastly, the readability of quality models expressed in two-level modeling languages is typically counter intuitive since they model quality issues and metrics, which might be difficult to understand (e.g., [26]). Our `QualityEvaluationModel` in Fig. 3 aims to overcome the beforementioned limitations through the exploitation of MLM and its intrinsic characteristics. Indeed, using MLM supports both the readability of quality models, due to the multiple levels of abstractions, as well as the management of their evolution thanks to the patterns discussed in Section II-D.

In the literature we found several spreadsheets-based approaches used for quality assessment, also applicable in the three candidate domains discussed in this work. For instance, in industrial contexts, spreadsheets are widely exploited for decision making purposes as made by Abreu *et al.* [27]. In this work, the authors highlight the importance of having spreadsheets of high quality since decisions taken upon wrong spreadsheets-based assumption may have serious economical impacts on businesses. The problematic aspect of spreadsheets quality is highlighted also in the work by Jannach *et al.* [28]. In this case, in order to solve the problem, the authors present an approach of Model-Based Diagnosis of faults in spreadsheets. Meanwhile, Luckey *et al.* [29] introduce a new way of object-oriented modeling to generate and evolve spreadsheets before using them, in order to reduce error-proneness in spreadsheets-based approaches. Similarly, Cunha *et al.* [30] exploit MDE techniques to build spreadsheets models easy to evolve and validate. Besides the benefits coming from the exploitation of MDE techniques in spreadsheets-based QES development, such types of systems rely on a two-level modeling approach. This aspect makes them still too specific to their application domain, thus negatively affecting reusability. The reusability of QESs is an important feature since, as also shown in the literature, there are very heterogeneous application domains for evaluation systems, besides the ones considered in this paper. Examples of diverse application domains follow. Lee *et al.* [31] propose a hierarchical model with financial and non-financial performance measures to assess and monitor business performance of employees in a company. In addition Cao *et al.* [32] present an approach to design project performance evaluation systems in a manufacturing company. The aim here is to help managers in reviewing a project identifying points that can be improved.

In conclusion, in this work, we show how leveraging MLM w.r.t. traditional two-level modeling approaches for realizing QESs, is feasible and may further support evolution and reusability of quality models and QESs.

V. Conclusion

MLM offers advantages with respect to traditional two-level modeling approaches, in specific cases. This paper presents a MLM approach for implementing a QES that can be used in different application domains. We focus on three domains: smart city KPIs assessment, research institute social impact, and COVID risk impact evaluations. The presented approach has been implemented as a prototype and future work will be dedicated to adapt and migrate the current two-layer modeling approach and related engine to the proposed multi-level one. Moreover, since the approach is fully automated, we need to test it with a larger number of subjects and requested evaluation parameters. Eventually, we are also interested in

exploring such kind of frameworks, which we call multi-level frameworks. It seems they allow one to benefit from the advantages of specific frameworks developed for one metamodel (customization) as well as from the advantage from general frameworks (e.g., consider model transformation frameworks as a prominent example) which are applicable to all metamodels (reuse at scale).

ACKNOWLEDGMENT

This work was partially supported by the Centre for Urban Informatics and Modelling - National Project - GSSI, the PON (Programma Operativo Nazionale Ricerca e Innovazione) projects, AIM1880573 Cultural Heritage and Smart, Secure and Inclusive Communities - National Projects - GSSI, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG), and the Austrian Science Fund (P 30525-N31).

REFERENCES

[1] M. Azuma, "Software products evaluation system: quality models, metrics and processes—International Standards and Japanese practice," *Information and Software Technology*, vol. 38, no. 3, pp. 145–154, 1996.

[2] O. Gordieiev, V. Kharchenko, N. Fominykh, and V. Sklyar, "Evolution of Software Quality Models in Context of the Standard ISO 25010," in *Proceedings of the Ninth International Conference on Dependability and Complex Systems (DepCoS-RELCOMEX)*. Springer, 2014, pp. 223–232.

[3] J. García-Munoz, M. García-Valls, and J. Escribano-Barreno, "Improved metrics handling in SonarQube for software quality monitoring," in *Proceedings of the 13th International Conference on Distributed Computing and Artificial Intelligence (DCAI)*. Springer, 2016, pp. 463–470.

[4] F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "A tool-supported approach for assessing the quality of modeling artifacts," *Journal of Computer Languages*, vol. 51, pp. 173–192, 2019.

[5] M. F. van Amstel and M. van den Brand, "Quality assessment of ATL model transformations using metrics," in *Proceedings of the 2nd International Workshop on Model Transformation with ATL (MtATL)*, 2010.

[6] T. Ambrus and M. Tóth, "Tool to Measure and Refactor Complex UML Models," in *Proceedings of the Fifth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA)*, 2016.

[7] M. Cardarelli, L. Iovino, P. Di Francesco, A. Di Salle, I. Malavolta, and P. Lago, "An extensible data-driven approach for evaluating the quality of microservice architectures," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1225–1234.

[8] M. De Sanctis, L. Iovino, M. T. Rossi, and M. Wimmer, "A flexible architecture for key performance indicators assessment in smart cities," in *Proceedings of the European Conference on Software Architecture (ECSA)*. Springer, 2020, pp. 118–135.

[9] 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.

[10] F. Macías, "Multilevel modelling and domain-specific languages," 2020. [Online]. Available: https://arxiv.org/abs/1910.03313

[11] J. De Lara, E. Guerra, and J. S. Cuadrado, "When and how to use multilevel modelling," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 2, 2014.

[12] M. De Sanctis, L. Iovino, M. T. Rossi, and M. Wimmer, "MIKADO – A Smart City KPIs Assessment Modeling Framework," *Software & Systems Modeling*, 2021.

[13] International Telecommunication Union (ITU), "Collection Methodology for Key Performance Indicators for Smart Sustainable Cities," 2017, https://www.unece.org/fileadmin/DAM/hlm/documents/Publications/U4SSC-CollectionMethodologyforKPIfoSSC-2017.pdf.

[14] P. Bosch, S. Jongeneel, V. Rovers, H.-M. Neumann, M. Airaksinen, and A. Huovila, "Citykeys indicators for smart city projects and smart cities," 2017, available at: https://nws.eurocities.eu/MediaShell/media/CITYkeystheindicators.pdf.

[15] M. Anzivino, F. A. Ceravolo, and M. Rostan, "The two dimensions of italian academics' public engagement," *Higher Education*, vol. 82, no. 1, pp. 107–125, 2021.

[16] L. Compagnucci and F. Spigarelli, "The third mission of the university: A systematic literature review on potentials and constraints," *Technological Forecasting and Social Change*, vol. 161, p. 120284, 2020.

[17] B. Gregersen, L. T. Linde, and J. G. Rasmussen, "Linking between danish universities and society," *Science and public policy*, vol. 36, no. 2, pp. 151–156, 2009.

[18] Ministero della Salute, "Tabella 21 indicatori," 2020, https://www.salute.gov.it/imgs/C_17_notizie_5152_1_file.pdf.

[19] F. Macías, A. Rutle, and V. Stolz, "MultEcore: Combining the Best of Fixed-Level and Multilevel Metamodelling," in *Proceedings of the 3rd International Workshop on Multi-Level Modelling (MULTI)*, ser. CEUR Workshop Proceedings, vol. 1722, 2016, pp. 66–75.

[20] D. S. Kolovos, R. F. Paige, and F. A. Polack, "The Epsilon Transformation Language," in *Proceedings of the International Conference on Theory and Practice of Model Transformations (ICMT)*. Springer, 2008, pp. 46–60.

[21] R. Van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, "Goal Question Metric (GQM) approach," *Encyclopedia of software engineering*, 2002.

[22] D. Samadhiya, Su-Hua Wang, and Dengjie Chen, "Quality models: Role and value in software engineering," in *Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE)*, 2010, pp. 320–324.

[23] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *Proceedings of the ICSE Workshop on Software Quality*, 2009, pp. 9–14.

[24] Z. Ma, X. He, and C. Liu, "Assessing the quality of metamodels," *Frontiers of Computer Science*, vol. 7, p. 558–570, 2013.

[25] M. Hara, T. Nagao, S. Hannoe, and J. Nakamura, "New key performance indicators for a smart sustainable city," *Sustainability*, vol. 8, no. 3, p. 206, 2016.

[26] F. D. Giraldo, S. España, O. Pastor, and W. Giraldo, "Considerations about quality in model-driven engineering," *Software Quality Journal*, vol. 26, pp. 1–66, 2016.

[27] R. Abreu, J. Cunha, J. Fernandes, P. Martins, A. Perez, and J. Saraiva, "Smelling Faults in Spreadsheets," in *Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME)*, 2014, pp. 111–120.

[28] D. Jannach, T. Schmitz, and K. Schekotihin, "Toward interactive spreadsheet debugging," in *Proceedings of the First Workshop on Software Engineering Methods in Spreadsheets*, 2014, pp. 3–6.

[29] M. Luckey, M. Erwig, and G. Engels, "Systematic evolution of model-based spreadsheet applications," *Journal of Visual Languages & Computing*, vol. 23, p. 267–286, 10 2012.

[30] J. Cunha, J. Fernandes, J. Mendes, and J. Saraiva, "Embedding, evolution, and validation of model-driven spreadsheets," *IEEE Transactions on Software Engineering*, vol. 41, pp. 241–263, 03 2015.

[31] H. Lee, W. Kwak, and I. Han, "Developing a business performance evaluation system: An analytic hierarchical model," *The Engineering Economist*, vol. 40, no. 4, pp. 343–357, 1995.

[32] Q. Cao and J. J. Hoffman, "A case study approach for developing a project performance evaluation system," *International Journal of Project Management*, vol. 29, no. 2, pp. 155–164, 2011.