

DOCTORAL THESIS

---

Supporting Smart Cities  
Quality Evaluation Exploiting  
Model-Driven Engineering

---

PHD PROGRAM IN COMPUTER SCIENCE: XXXIV CYCLE

*Supervisor:*

Dr. Ludovico IOVINO  
ludovico.iovino@gssi.it

*Author:*

Maria Teresa ROSSI  
mariateresa.rossi@gssi.it

*Co-Supervisor:*

Prof. Manuel WIMMER  
manuel.wimmer@jku.at

March 20, 2023

# Declaration of Authorship

I, Maria Teresa ROSSI, declare that this thesis titled, 'Supporting Smart Cities Quality Evaluation Exploiting Model-Driven Engineering' and the work presented in it are based on co-authored papers published in Proceedings of International Conferences (ECSA, ICSOC, MoSC@STAF, MULTI@MODELS) and in the International Journal on Software and Systems Modeling (SoSyM).

I confirm that:

- The content of Chapter 3 is based on a work co-authored by Martina De Sanctis, Ludovico Iovino and Manuel Wimmer, which is under review to the International Journal on Software and Systems Modeling (SoSyM) [1].
- The content of Chapter 4 is based on a work co-authored by Martina De Sanctis, Ludovico Iovino and Manuel Wimmer, which have been published in the International Journal on Software and Systems Modeling (SoSyM) [2].
- The content of Chapter 5 is based on works co-authored by Francesco Basciani, Martina De Sanctis, Ludovico Iovino and Manuel Wimmer, which have been published in the International Journal on Software and Systems Modeling (SoSyM) [2], in the Proceedings of the 14th European Conference on Software Architecture (ECSA 2020) [3], in the Proceedings of the 1st International Workshop on Modeling Smart Cities (MoSC@STAF 2020) [4] and in the Proceedings of the 19th International Conference on Service-Oriented Computing (ICSOC 2021) [5].
- The content of Chapter 6 is based on works co-authored by Martina De Sanctis, Ludovico Iovino and Manuel Wimmer, which have been published in the International Journal on Software and Systems Modeling (SoSyM) [2], in the Proceedings of the 14th European Conference on Software Architecture (ECSA 2020) [3], and in the Proceedings of the 19th International Conference on Service-Oriented Computing (ICSOC 2021) [5].

- The content of Chapter 7 is based on works co-authored by Martina De Sanctis, Martina Dal Molin, Ludovico Iovino and Manuel Wimmer, which have been published in the Proceedings of the 8th International Workshop on Multi-Level Modelling (MULTI@MODELS 2021) [6].

Signed:

---

Date:

---



# *Abstract*

The concept of Smart City was coined in 2011 to define an idealized city characterized by automation and connection. As a consequence of the digital revolution, it further evolved detailing the multiple aspects characterizing smart cities themselves (e.g., sustainable mobility, environmental management, citizens inclusion). The European Commission published an agenda containing several objectives, called *Sustainable Development Goals* (SDGs), to reach in 2030 to face the crisis and promote smart, sustainable, and inclusive growth of European cities. On top of the SDGs, international projects targeted the definition of smart cities' Key Performance Indicators (KPIs), along with their collection methodology, to capture the performance of a city in multiple dimensions and to support transparent monitoring and the comparability among smart cities. In this context, the *Smart Governance* performed in smart cities is in charge of decision-making processes, by exploiting KPIs assessment in order to have a complete vision of the cities in terms of smartness and sustainability.

Overall, a framework for KPIs measurements in smart cities could be defined as a Quality Evaluation System (QES) that performs the assessment of a subject, i.e., the candidate smart city, w.r.t. some quality metrics, i.e., the selected interesting KPIs, as usually done for software quality analysis. However, despite the growing interest in smart cities evaluation and the existing guidelines on KPIs, no standard tools, languages and models to support systematic KPIs assessment processes do exist. This implies the lack of efficiency in the process of smart city evaluation and comparison that, in turn, affects the growth and improvement of smart cities. Moreover, this limitation hinders knowledge sharing among the smart city stakeholders, thus negatively affecting the smart city decision-making processes. These challenges are further exacerbated by the complex nature of smart cities themselves. In fact, when speaking about smart cities, multiple dimensions (e.g., mobility, economy, environment) come into play, together with their corresponding stakeholders (e.g., private companies, public administrations, service providers). These dimensions are very heterogeneous, making it difficult also the interconnection among them. Furthermore, different cities can show diverse features and peculiarities (e.g., size, economic growth), thus affecting the relevance that some KPIs might have in their specific context. This entails that the relevant sets of KPIs might differ among different cities, thus implying the need for KPIs *customization*. *KPIs evolve over time*, which means that new KPIs can be defined or existing ones can be implemented in slightly different manners. Unfortunately, the currently available frameworks (e.g., manual, spreadsheets-based, Web-based platforms) for KPIs assessment are still far from being flexible enough. To the contrary, they are tailored to a specific domain, or closed to customization, or rely on manual and error-prone tasks.

Model-Driven Engineering (MDE) techniques are widely used to represent complex systems through abstract models. In this dissertation, we propose *MIKADO*– a Smart City KPIs Assessment Modeling Framework. Our solution is an approach supporting (i) the uniform modeling of both smart cities and KPIs, (ii) the automatic calculation of KPIs, and (iii) graphical visualization of assessed KPIs by means of dynamic dashboards. *MIKADO* enables the continuous monitoring and evaluation of the KPIs' input parameters, by weaving open services and runtime models. The resulting approach provides a standard, but at the same time, a customizable process for smart cities governance administrators. *MIKADO* is characterized by domain-independence, indeed we demonstrate that it can be generalized into a QES for multiple domains, by exploiting the Multilevel modeling paradigm. We evaluated the approach in terms of (i) understandability of the *MIKADO* Domain Specific Languages, (ii) performance measured as execution time given variable models; and (iii) the latency.

## *Acknowledgements*

I would like to express my deepest gratitude to my supervisor Ludovico Iovino for his invaluable support, motivation and feedback. I also could not have undertaken this journey without my co-supervisor Manuel Wimmer for providing me with his knowledge and expertise.

I am deeply indebted to my tutor, co-author and friend Martina De Sanctis for her editing help, long and precious feedback sessions and moral support. This thesis would have not been possible without her supervision and inspiring energy.

This work was partially supported by the Centre for Urban Informatics and Modelling (CUIM).

# Contents

<b>Declaration of Authorship</b>	<b>i</b>
<b>Abstract</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>Abbreviations</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Thesis Context . . . . .	1
1.2 Problem and Challenges . . . . .	2
1.3 Research Questions and Solution Overview . . . . .	4
1.4 Research Activities and Publications . . . . .	5
1.4.1 Research Publications . . . . .	8
1.4.2 Other Publications . . . . .	9
1.5 Tools & Demo . . . . .	10
1.6 Thesis Outline . . . . .	11
<b>2 Background</b>	<b>13</b>
2.1 The Smart City Domain . . . . .	13
2.1.1 Smart Governance . . . . .	16
2.2 Quality Evaluation Systems . . . . .	17
2.3 Model-Driven Engineering . . . . .	18
2.3.1 Multi-level Modeling . . . . .	20
2.3.2 Models@Runtime . . . . .	21
2.4 Discussion . . . . .	22
<b>3 State of the Art and Motivation</b>	<b>23</b>
3.1 Smart Cities Evaluation through KPIs . . . . .	23
3.1.1 KPIs Assessment Frameworks . . . . .	25
Manual approaches. . . . .	25
Spreadsheets-based approaches. . . . .	26



	Web-based platforms. . . . .	27
	3.1.1.1 Requirements Perspective . . . . .	28
	3.1.1.2 Process Perspective . . . . .	29
3.2	Quality Evaluation Systems . . . . .	30
3.3	Modeling Smart Cities . . . . .	33
	3.3.1 Smart Cities Modeling Approaches Review: Main Findings . . . . .	34
	Problem Domain. . . . .	36
	Solution Domain. . . . .	37
	Scientific Community Domain. . . . .	38
3.4	Research Challenges . . . . .	40
3.5	Discussion . . . . .	43
<b>4</b>	<b><i>MIKADO</i>– a Smart City KPIs Assessment Modeling Framework</b>	<b>45</b>
4.1	Motivating Scenario: Smart Decision-Making . . . . .	45
4.2	The <i>MIKADO</i> Approach . . . . .	47
	4.2.1 Smart Cities Metamodel . . . . .	48
	4.2.2 Key Performance Indicators Metamodel . . . . .	49
	4.2.3 KPIs Assessment . . . . .	51
	4.2.4 KPIs Reporting . . . . .	51
4.3	Supporting KPIs evolution . . . . .	52
4.4	Discussion . . . . .	54
<b>5</b>	<b>Architecture and Implementation</b>	<b>55</b>
5.1	<i>MIKADO</i> Flexible Architecture . . . . .	55
5.2	Standalone Deployment Implementation . . . . .	59
	5.2.1 Modeling Component . . . . .	60
	5.2.2 Analysis Component . . . . .	63
	5.2.3 Data Visualization Component . . . . .	65
5.3	Hybrid Deployment Implementation . . . . .	66
	5.3.1 Weaving Open Services with Runtime Models for Continuous Smart Cities KPIs Assessment . . . . .	67
	5.3.2 Runtime Models Update by Continuous Monitoring . . . . .	70
5.4	Online Deployment Specification . . . . .	73
5.5	Discussion . . . . .	76
<b>6</b>	<b>Evaluation Results</b>	<b>78</b>
6.1	Demonstration Case . . . . .	78
	6.1.1 Selection of Real-World KPIs . . . . .	78
	6.1.2 Modeling of KPIs Definitions . . . . .	80
	6.1.3 Modeling of Smart Cities . . . . .	81
	6.1.4 KPIs Assessment through the Evaluation Engine . . . . .	82
	6.1.5 KPIs Visualization through Dashboards Generation . . . . .	83
	6.1.6 Supporting Smart Cities Comparison . . . . .	84
	6.1.7 Supporting Smart Cities and KPIs Evolution . . . . .	86
6.2	Understandability of <i>MIKADO</i> 's DSLs . . . . .	87
	6.2.1 Survey Setup and Execution . . . . .	87
	6.2.2 Survey Results . . . . .	88

	Threats to validity. . . . .	89
6.3	Performance of the <i>MIKADO</i> Framework . . . . .	90
6.3.1	Scalability of the Evaluation Engine . . . . .	91
	Experiment Setup. . . . .	91
	Experiment Results. . . . .	91
6.3.2	Empowering of the Evaluation Engine . . . . .	92
	Experiment Setup. . . . .	93
	Experiment Results. . . . .	93
6.3.3	Threats to validity . . . . .	95
6.4	Latency Analysis of Service-based Continuous KPIs Assessment . . . . .	95
	Experiment Setup. . . . .	96
	Results for RQ1. . . . .	97
	Results for <i>RQ2</i> . . . . .	99
	Threats to validity. . . . .	99
6.5	Discussion . . . . .	100
<b>7</b>	<b>Generalizability of the Proposed Framework</b>	<b>101</b>
7.1	Motivations . . . . .	101
7.2	Bringing <i>MIKADO</i> to Multilevel . . . . .	102
7.2.1	Leveraging Multi-Level Modeling for Multi-Domain Quality Assessment . . . . .	103
7.2.2	Running Examples . . . . .	104
	Smart City KPIs evaluation. . . . .	104
	Research Institute Social Impact. . . . .	105
	Covid-19 Risk. . . . .	105
7.3	Subjects Definition . . . . .	106
7.4	Quality Metrics Definition . . . . .	108
7.5	QES engine . . . . .	110
7.6	Assessment Results Graphical Representation . . . . .	112
7.7	Discussion . . . . .	114
<b>8</b>	<b>Conclusions and Future Work</b>	<b>115</b>
8.1	Future Work . . . . .	118
8.1.1	Leveraging Models@Runtime in a Digital Twin perspective . . . . .	119
8.1.2	Supporting Ethics Risk Traceability in Quality Assessment . . . . .	120

# List of Figures

1.1	Research Plan performed during the PhD program. . . . .	6
2.1	Boyd Cohen Wheel. . . . .	15
2.2	Structure of the KPIs proposed in [7]. . . . .	17
2.3	Relationship between MDE, MDD and MDA concepts. . . . .	20
2.4	Comparison between traditional and multi-level modelling [8] . . . . .	21
3.1	Basic process for the development of frameworks for the KPIs assessment. . . . .	30
3.2	Distribution of publications among the smart city dimensions. . . . .	36
3.3	Number of publications w.r.t. the type facets. . . . .	38
3.4	Boxplot with the TRL values w.r.t. the three facets. . . . .	39
3.5	Number of publications per year w.r.t. the publications type. . . . .	39
4.1	Overview of the <i>MIKADO</i> approach. . . . .	47
4.2	Smart City Metamodel. . . . .	48
4.3	KPIs Metamodel. . . . .	50
4.4	Example of gauge and range charts. . . . .	52
5.1	The Flexible Architecture for the KPIs Assessment in Smart Cities. . . . .	56
5.2	Standalone specification of the flexible architecture in Figure 5.1. . . . .	60
5.3	Textual representation of the model for the city of L'Aquila. . . . .	61
5.4	Graphical representation of the model for the city of L'Aquila. . . . .	62
5.5	Auto-completion feature. . . . .	63
5.6	Syntax error at modeling time. . . . .	63
5.7	Hybrid specification of the flexible architecture in Figure 5.1. . . . .	68
5.8	Sequence Diagram of the KPIs assessment process. . . . .	69
5.9	Publish-Subscribe pattern in our example . . . . .	71
5.10	Our tool in action for the smart city L'Aquila. . . . .	73
5.11	Online specification of our architecture . . . . .	74
6.1	Graphical Representation of the Smart City Model for the city of L'Aquila. . . . .	82
6.2	Examples of the input and output models during the assessment process. . . . .	83
6.3	KPIs Assessment Results over the Smart City of L'Aquila. . . . .	83
6.4	Overall View of the KPIs for the city of L'Aquila. . . . .	84
6.5	Detailed view of a single category of KPI. . . . .	85
6.6	Scenario 2 - Change KPIs type. . . . .	87
6.7	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. . . . .	89

---

6.8	Users preferences between our DSL and spreadsheets when considering a period of training of 1 week (left side) and 10 weeks (right side). . . . .	89
6.9	Users preferences w.r.t. the two proposed smart city's representations, graphical vs. tabular. . . . .	90
6.10	<i>Exp1</i> : increasing the number of evaluated smart cities. . . . .	92
6.11	<i>Exp2</i> : increasing the complexity in the calculation of each modeled KPI. . . . .	92
6.12	<i>Exp3</i> : make each KPI of type <i>range</i> . . . . .	92
6.13	<i>Exp4</i> : increasing the number of KPIs. . . . .	92
6.14	Execution times of the evaluation engine resulting from the experiments. . . . .	94
6.15	Data retrieving, SC Model update and evaluation engine execution times. Automated: 963 interactions. . . . .	97
6.16	Data retrieving, SC Model update and evaluation engine execution times. Baseline: 18 interactions. . . . .	98
6.17	Latency contributed by the three phases. . . . .	99
7.1	Overview of the quality assessment approach. . . . .	102
7.2	Overview of the Multi-level Quality Assessment System. . . . .	103
7.3	Multi-level hierarchy for Subjects definition. . . . .	107
7.4	Multi-level hierarchy for Quality Metrics definition. . . . .	109
7.5	Excerpt of the Dashboard evaluating Smart City KPIs. . . . .	112
7.6	Excerpt of the Dashboard evaluating Social Impact. . . . .	113
7.7	Excerpt of the Dashboard evaluating COVID Risk. . . . .	113
8.1	KPIs assessment process and involved roles. . . . .	121

# List of Tables

1.1	List of the resources related to the developed artifacts. . . . .	10
3.1	Evaluation of KPIs assessment frameworks. . . . .	29
5.1	Architecture flexibility in terms of required components and technologies. . . . .	75
6.1	Sources of the KPIs definition used in the experiment . . . . .	79
6.2	Evaluation and comparison of the subject smart cities. . . . .	85
6.3	Scenario-based testing for the evaluation of the <i>MIKADO</i> 's evolution support. . . . .	86
6.4	Increasingly complex scenarios for the scalability assessment. . . . .	93

# Abbreviations

<b>AP</b>	<b>Air Pollution</b>
<b>API</b>	<b>Application Programming Interface</b>
<b>AQI</b>	<b>Air Quality Index</b>
<b>BN</b>	<b>Bicycle Network</b>
<b>DSL</b>	<b>Domain-Specific Language</b>
<b>DT</b>	<b>Digital Twin</b>
<b>EGL</b>	<b>Epsilon Generation Language</b>
<b>EMF</b>	<b>Eclipse Modeling Framework</b>
<b>EOL</b>	<b>Epsilon Object Language</b>
<b>EVL</b>	<b>Epsilon Validation Language</b>
<b>Exp</b>	<b>Experiment</b>
<b>GA</b>	<b>Green Areas</b>
<b>GMF</b>	<b>Graphical Modeling Framework</b>
<b>GQM</b>	<b>Goal-Question-Metric</b>
<b>GPL</b>	<b>General-Purpose programming Languages</b>
<b>HEI</b>	<b>Higher Education Institution</b>
<b>ICT</b>	<b>Information and Communication Technology</b>
<b>IoT</b>	<b>Internet of Things</b>
<b>ITU</b>	<b>International Telecommunication Union</b>
<b>KPI</b>	<b>Key Performance Indicator</b>
<b>LCDP</b>	<b>Low-Code Development Platform</b>
<b>MA</b>	<b>Mobile Applications</b>
<b>M2M</b>	<b>Model-to-Model</b>
<b>M2T</b>	<b>Model-to-Text</b>
<b>MDA</b>	<b>Model-Driven Architecture</b>

---

<b>MDD</b>	<b>Model-Driven Development</b>
<b>MDE</b>	<b>Model-Driven Engineering</b>
<b>MLM</b>	<b>Multi-Level Modeling</b>
<b>ms</b>	<b>milliseconds</b>
<b>QES</b>	<b>Quality Evaluation System</b>
<b>QM</b>	<b>Quality Metric</b>
<b>RC</b>	<b>Research Challenge</b>
<b>RQ</b>	<b>Research Question</b>
<b>SC</b>	<b>Smart City</b>
<b>SCS</b>	<b>Smart City System</b>
<b>SD</b>	<b>Sequence Diagram</b>
<b>SDG</b>	<b>Sustainable Development Goal</b>
<b>SoC</b>	<b>Separation of Concerns</b>
<b>SSC</b>	<b>Smart Sustainable Cities</b>
<b>SV</b>	<b>Sales Volume</b>
<b>TM</b>	<b>Transport Monitoring</b>
<b>TRL</b>	<b>Technology Readiness Level</b>
<b>UML</b>	<b>Unified Modeling Language</b>
<b>UHI</b>	<b>Urban Heat Island</b>

# Chapter 1

## Introduction

In this Chapter, we give an exhaustive introduction of the work described in this dissertation. In particular, we describe the context in which the thesis has been developed in Section 1.1. Then, Section 1.2 explains the problem and research challenges that we want to face with our work. Thus, in Section 1.3 we report the research questions that guided the work and an overview of the developed solution is provided. Moreover, the research activities performed and the produced publications making up this dissertation are reported in Section 1.4. Further, in Section 1.5 we provide the references to the tools and demos of the developed framework. The Chapter is closed by the thesis outline in Section 1.6.

### 1.1 Thesis Context

The concept of Smart City (SC) was coined by the IBM's brand in 2011 [9] to define an idealized city characterized by automation and connection. It further evolved detailing the multiple aspects characterizing smart cities themselves, which must be considered to effectively make cities smarter, such as sustainable mobility, environmental management, citizens inclusion, and so on. Consequently to the birth of the concept, cities started a process of transition to become smarter. For this purpose, weaving Information and Communication Technology (ICT) and cities can be exploited in order to promote their sustainable development to improve human lives and preserve the environment. Consequently, the concept of SC has been gaining more and more relevance both at European and world-wide level. For instance, the European Commission published an agenda [10] containing several objectives, called *Sustainable Development Goals* (SDGs)<sup>1</sup>, to reach in 2030 to face the crisis and promote a smart, sustainable and inclusive growth of European

---

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



cities. In particular, the smart growth is related with the development of an economy based on *knowledge* and *innovation*. This type of economies aim to use technologies and information systems for the sharing and spreading of knowledge. However, the smart city domain brings a certain complexity. It covers different dimensions, such as mobility, environment, economy, governance, and so on. As a consequence, it is required a centralized management and monitoring of the different dimensions, supporting also the knowledge sharing among the involved actors and stakeholders, such as municipalities, public and private companies. Ideally, the *Smart Governance* should play this role. It is supposed to have a complete vision on the city in order to support decision-making processes. However, the dimensions of the smart city are managed by different stakeholders (e.g., public administrations, private institutions) that not always communicate with each other. This makes it difficult to public administrations to have a complete overview of the city. Moreover, the *digital revolution* [11] which is taking place in these years is bringing out new issues and needs in the cities (e.g., the availability of several big data sources). A support for the smart governance of smart cities is provided by Key Performance Indicators (KPIs) [12], representing raw set of values that can provide some information about relevant measures that are of interest for understanding the progress of a smart city. 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 [7]. Other initiatives and international projects (e.g., [13, 14]) 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. The process of KPIs assessment over smart cities can be mapped to a quality evaluation process usually used for software quality analysis [15]. Indeed, a framework providing KPIs measurements over a smart city can be defined as a Quality Evaluation System (QES) that performs the assessment of a subject, i.e., the candidate smart city, w.r.t. some quality metrics, i.e., the selected interesting KPIs.

## 1.2 Problem and Challenges

Despite all this interest in the definition of KPIs assesment over smart cities, no standard tools, languages and models do exist and a myriad of fragmented, disconnected, incompatible and heterogeneous solutions exists. This implies the lack of efficiency in the process of smart cities evaluation and comparison that, in turn, affects the growth and improvement of smart cities. This lack of a widely used approach for KPIs assessment, given the spread of smart city and its relevance at European and world-wide level, makes the smart cities KPIs assesment a process for its own sake, which does not expose its potentiality, since it does not support the fruition of generated data, the comparison

among several smart cities (i.e., because of the incongruence of the different used tools and technologies) which are, instead, of crucial importance. Moreover, this lack of comprehensiveness represents an obstacle in the management of smart cities. It makes the knowledge sharing among smart cities stakeholders difficult and this, in turn, negatively affects smart city decision-making processes. This is also due to the complex nature of smart cities. In fact, multiple dimensions (e.g., mobility, economy, environment) come into play, together with their corresponding stakeholders (e.g., private companies, public administrations, service providers). These dimensions are very heterogeneous and this makes it difficult also the interconnection among them. Moreover, the transition to smart cities might also be affected by the growing (and uneven w.r.t. the different dimensions) amount of investments in smart initiatives and the advent of new technologies.

As regards the nature of KPIs, these metrics reflect the degree of smartness and sustainability of smart cities. However, different cities can show diverse features and peculiarities (e.g., size, economic growth), thus affecting the relevance that some KPIs might have in their specific context. The relevant (sub-)set of KPIs might differ among different cities, thus implying the need for KPIs *customization*. Moreover, *KPIs evolve over time* [16]. 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<sup>2</sup>, Web-based platforms [13]) for the KPIs calculation are still far from being flexible enough. On the contrary, KPIs definition models are embedded in these 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 [17]. Moreover, this lack of flexibility implies that when changing the smart city to evaluate the whole framework or the great part of it has to be re-implemented from scratch. For instance, in Excel-based frameworks, when changing the subject of the evaluation, also the data given as input to the KPIs formulae, which are themselves defined in Excel, has to change. Further, in this case, the configuration of external data sources (e.g., a MS SQL database) hosting the required data requires re-configuration, by using advanced features of Excel. These adjustments have to be performed by a user with an high expertise of the advanced features of Excel, which not always correspond to the final user of the KPIs assessment framework. The same reasoning applies for QESs, since also this type of systems have been often implemented for the domain-specific evaluation of different subjects, also by means of spreadsheets.

These emergent requirements strongly affect the design and development of frameworks devoted to KPIs assessment over smart cities. With these premises, we come out with three main Research Challenges (RCs) that we intend to address with the work presented in this dissertation, which are:

---

<sup>2</sup> Key Performance Indicators in Power Pivot at <https://bit.ly/37EFR9r>

- RC1:** Provide a uniform way of modeling *smart cities* and their complexity.
- RC2:** Provide a systematic methodology allowing smart cities to *define, measure* and *visualize* the KPIs of interest in order to efficiently assisting the decision-making processes by also supporting:
- RC2.1:** *KPIs evolution* over time w.r.t. the evolution of smart cities needs and contexts;
  - RC2.2:** *KPIs customization* based on the specific smart city under evaluation in terms of selection of appropriate KPIs to be calculated.
- RC3:** Make the systematic assessment methodology *generalizable* such that to be applied in the quality assessment of subjects coming from different domains than the smart cities.

### 1.3 Research Questions and Solution Overview

This section describes the research questions that drive our research work and further gives an overview of the realized solution. In the literature, there exist examples showing how ICT can help in managing different aspects of complex systems (e.g., [18]). In particular, Model-Driven Engineering (MDE) [19] techniques are widely used to represent complex systems through abstract models. Examples exist also in the smart city domain (e.g., [20], [21]). With these premises, after an appropriate feasibility study we decided to exploit MDE techniques to develop a smart city KPIs assessment modeling framework.

Based on the before-mentioned research challenges, we derived the following Research Questions (RQs), which led the work behind this dissertation:

**RQ1:** How smart cities are modeled in the literature?

This RQ is based on RC1, and requires to investigate if there exist a standard to describe smart cities that can be reusable, machine readable and understandable also for users that have no expertise in programming. This is valuable to know, before defining a new uniform way of modeling smart cities, as needed in our envisioned solution.

**RQ2:** How can MDE help in the uniform modeling and automatic assessment of smart cities?

This RQ is related to RC2 and requires the understanding of how to model smart cities and KPIs in a systematic and uniform way, enabling the automatic calculation of the defined KPIs over the candidate smart city. Moreover, this RQ also requires

to investigate how graphical representations of the results can be generated, starting from the defined models. It includes two additional sub-questions:

**RQ2.1:** How can we efficiently monitor KPIs over time?

This RQ is connected to RC2.1 and requires the understanding of how KPIs change over time and how to support these changes in the corresponding model.

**RQ2.2:** How can we adequately monitor KPIs for different smart cities?

This RQ is connected to RC2.2 and requires the understanding of how KPIs change from a city to another and how to make the corresponding model customizable accordingly.

**RQ3:** To what extent does a model-based smart cities KPIs assessment approach can be generalized to other subjects of evaluation belonging to different domains?

This RQ is based on RC3 and requires the investigation of the commonalities between KPIs assessment processes and more generic quality evaluation processes, in order to understand if a model-driven KPIs assessment framework can be applicable in different domains than smart cities.

To address the challenges defined in Section 1.2 and to further answer the RQs just described above, we develop *MIKADO*— a Smart City KPIs Assessment Modeling Framework. In particular, our framework provides as main functionalities (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. In the development of the framework, we exploited MDE techniques to provide *Domain Specific Languages* (DSLs) [22] 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. Moreover, by applying the Multilevel modeling paradigm [23], we generalized the smart cities KPIs assessment framework transforming it into a QES for multiple domains.

## 1.4 Research Activities and Publications

This section describes the performed research activities and the produced publications making up this dissertation. An overview is reported in Figure 1.1. In particular, in the first lane the RQs that conducted the research stages are reported. Thus, in the second lane the research stages are shown in logical order. Lastly, in the third lane the

publications produced as output of the performed research stages are reported by their titles and publication venues.

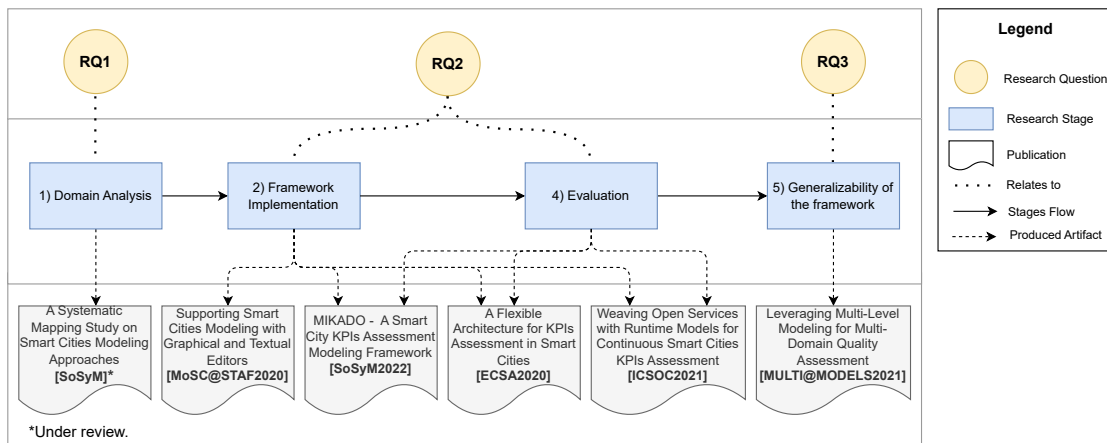


FIGURE 1.1: Research Plan performed during the PhD program.

In details, we describe here the research activities conducted in four main stages w.r.t. the RQs that motivate them and their produced publications, indicated with an identifier in square brackets and listed in Section 1.4.1.

1. *Domain Analysis*. During this preliminary stage, we performed a *Domain Analysis* of the smart city domain. This led us to run a systematic mapping study about *smart cities modeling*. The goal of this stage was to extract all the concepts and requirements needed to design a DSL for smart cities. This preliminary study concerns also the *understanding of KPIs*. In particular, the extraction of the logic behind the calculations of the interesting KPIs, in such a way to create corresponding models abstract enough to be used in different smart cities as well as easy to modify to deal with the evolution of KPIs. This stage has been structured into two sub-stages:

- (a) the *Study on Smart Cities*, in which a mapping study on smart cities modeling [SoSyM]\* to answer RQ1 was performed. The outcomes of this study supported the design of a *Smart City metamodel* for the modeling of smart cities with a particular focus on the data analytics context.
- (b) the *Understanding of KPIs* during which multiple KPIs sources have been inspected, namely, (i) ITU [7] and (ii) CITYkeys [13], which are two projects specifically targeting the definition of smart cities KPIs to capture the performance of a city in multiple dimensions; (iii) DigitalAQ [14], that is an European project 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). The study of the mentioned KPIs sources produced the design of a *KPIs metamodel* for the modeling of different KPIs definitions.

2. *Framework Implementation*. During this research stage, all the modeling artifacts required for the KPIs assessment framework were designed and developed. In particular, we can identify five sub-stages:
  - (a) The *Definition of the Architecture* supporting the planned model-driven KPIs assessment approach has been defined in a flexible fashion allowing different deployment style (e.g., standalone, online) [ECSA2020].
  - (b) The *Modeling Tools Development* on top of the metamodels defined during the first stage. In particular, the implementation of a *Smart City modeling tool* as a visual editor in which the user can design, under certain constraints, the model of the city [MoSC@STAF2020]. Secondly, the *KPIs modeling tool* to support the user in the selection and application of the relevant KPIs for the referring city through the definition of a *textual concrete syntax* for modeling KPIs.
  - (c) The *Evaluation Engine Development* as the core component of the approach devoted to the interpretation of the modeled smart cities and KPIs, the calculation of such indicators and the instantiation of the results in the model containing the KPIs definition.
  - (d) The *Reporting System development* regarding a code generation process that produces graphical dynamic dashboards has been implemented, enabling the intuitive representation and visualization of the KPIs assessed over a city. The outcome of the last three sub-stages (*b,c,d*) make the model-based KPIs assessment framework [SoSyM2022].
  - (e) Lastly, the *Development of an Extension of the Assessment Framework enabling KPIs Continuous Monitoring* has been proposed. In particular, the extension envisages the continuous monitoring of the KPIs' input parameters provided by open services, automating the data acquisition process and the continuous evaluation of KPIs. Specifically, this feature will turn smart cities models into digital twins, by weaving open services and runtime models [ICSOC2021].
3. *Evaluation*. Performance evaluations of the overall approach have been incrementally conducted for each framework releases. In particular, two aspects of the approach have been tested, namely, (i) the *scalability* of the evaluation engine in managing models of increasing size and complex KPIs [ECSA2020,SoSyM2022], and (ii) the *usability and understandability* of the presented approach, especially w.r.t.

spreadsheets-based approaches, through a survey study [SoSyM2022]. Moreover, even the latency of the execution time of the framework extension was tested w.r.t. the standalone baseline [ICSOC2021].

4. *Abstraction of the framework.* During this research activity, the application of the presented framework on other two different domains (i.e., research institutes social impact, covid-19 risk) by abstracting the approach as a Multi-Domain Quality Evaluation System have been proposed [MULTI@MODELS2021]. This application was carried out by exploiting the Multilevel modeling paradigm [23] in the development of a Quality Evaluation System that showed the *domain-independence* of the approach.

### 1.4.1 Research Publications

In the following, the research publications supporting this dissertation are listed, by also specifying the candidate's level of contribution according to the Elsevier's CRediT (Contributor Roles Taxonomy) author statement<sup>3</sup>:

[SoSyM]\* **Rossi, M.T.**, De Sanctis, M., Iovino, L., Wimmer, M.: *A Systematic Mapping Study On Smart Cities Modeling Approaches*. In: the International Journal on Software and Systems Modeling (SoSyM) (2022) (Submitted). CRediT: Conceptualization; Methodology; Validation; Formal analysis; Investigation; Resources; Data Curation; Writing; Visualization.

[MoSC@STAF2020] Basciani, F., **Rossi, M.T.**, De Sanctis, M.: *Supporting smart cities modeling with graphical and textual editors*. In: *1st International Workshop on Modeling Smart Cities @STAF2020 (2020)*.

A workshop paper [4] in which the smart city modeling editor component, showing both graphical and textual views, has been presented. CRediT: Conceptualization; Methodology; Validation; Investigation; Data Curation; Writing; Visualization.

[ECSA2020] 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 '20 (2020)*.

A conference paper [3] in which we presented a flexible architecture, identifying both required and optional components and functionalities needed to support the automatic KPIs assessment approach. CRediT: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Resources; Data Curation; Writing; Visualization.

<sup>3</sup> <https://www.elsevier.com/authors/policies-and-guidelines/credit-author-statement>

[SoSyM2022] *De Sanctis, M., Iovino, L., Rossi, M.T., Wimmer, M.: MIKADO – A Smart City KPIs Assessment Modeling Framework.* In: the International Journal on Software and Systems Modeling (SoSyM) (2021).

The paper [2] presents all the artifacts developed so far and composing the thesis approach. It has been selected to be presented at the MODELS 2021 conference as a **Journal First** paper. CRediT: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Resources; Data Curation; Writing; Visualization.

[ICSOC2021] *De Sanctis, M., Iovino, L., Rossi, M.T., Wimmer, M.: Weaving Open Services with Runtime Models for Continuous Smart Cities KPIs Assessment.* In: *The 19th International Conference on Service-Oriented Computing, ICSOC 2021.*

The paper [5] propose the extension of the framework that enables KPIs continuous monitoring. CRediT: Conceptualization; Methodology; Validation; Formal analysis; Investigation; Data Curation; Writing; Visualization.

[MULTI@MODELS2021] *Rossi, M.T., Dal Molin, M., Iovino, L., De Sanctis, M., Wimmer, M.: Leveraging Multi-Level Modeling for Multi-Domain Quality Assessment.* In: *MULTI 2021, The 8th International Workshop on Multi-Level Modelling, MODELS 2021.*

An *inter-disciplinary* workshop paper [6] presenting the abstraction of the assessment approach as a Multi-domain quality evaluation system. CRediT: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Resources; Data Curation; Writing; Visualization.

## 1.4.2 Other Publications

During the PhD program also other activities external to the thesis research have been conducted. These activities helped me in investigating the smart city domain from other perspectives, and produced further papers:

1. *Rossi, M. T., De Sanctis, M., Iovino, L., Rutle, A.: A Multilevel Modelling Approach for Tourism Flows Detection.* *MULTI@MODELS 2019: 103-112.* URL: <https://ieeexplore.ieee.org/document/8904730>.

CRediT: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Resources; Data Curation; Writing; Visualization.

2. *Iovino, L., De Sanctis, M., Rossi, M. T.: Automated Code Generation for NFC-based Access Control.* *MDE4IoT@MODELS 2019: 31-38.* URL: <http://ceur-ws.org/Vol-2442/paper5.pdf>. CRediT: Investigation; Writing; Visualization.



3. **Rossi, M. T.**, Greca, R., Iovino, L., Giacinto, G., Bertolino, A. : *Defensive Programming for Smart Home Cybersecurity*. In: *Workshop on Software Attacks and Defenses SAD (2020)*.  
URL: <https://ieeexplore.ieee.org/document/9229690>. CRediT: Conceptualization; Methodology; Formal analysis; Investigation; Writing; Visualization.
4. *Tuyishime, A., Izquierdo, J.L.C., Rossi, M.T. De Sanctis, M.: Modeling Linked Open Data (POSTER)*. In: *International workshop on MDE for Smart IoT Systems (MeSS'22) (2022)*. URL: <https://ceur-ws.org/Vol-3250/messpaper3.pdf>.  
CRediT: Investigation; Visualization.
5. *De Sanctis, M., Di Salle, A., Iovino, L., Rossi, M.T.: A Technology Transfer Journey to a Model Driven Access Control System*. In: *International Journal on Software Tools for Technology Transfer (STTT) (2023)*. CRediT: Investigation; Writing; Visualization.

## 1.5 Tools & Demo

In Table 1.1, we reported the references to the repositories where to find all the developed artifacts and some video demo showing specific functionalities of the developed framework.

Tools and Resources	Description	URL
Smart City Modeling	GIT repository collecting all the modeling artifacts composing <i>MIKADO</i> .	<a href="https://github.com/gssi/SmartCityModeling">https://github.com/gssi/SmartCityModeling</a>
Runtime KPIs Assessment	GIT repository of the <i>MIKADO</i> extension enabling KPIs continuous monitoring.	<a href="https://github.com/iovinoludovico/runtime-kpi-assessment">https://github.com/iovinoludovico/runtime-kpi-assessment</a>
Smart City Modeling Editor	GIT repository of the graphical/textual Smart City modeling editor.	<a href="https://github.com/gssi/MoSC2020">https://github.com/gssi/MoSC2020</a>
Smart City Dashboards Generation	Video demo showing the KPIs reporting system development.	<a href="https://www.youtube.com/watch?v=B-B3_O4L6xc">https://www.youtube.com/watch?v=B-B3_O4L6xc</a>
Continuous Monitoring	Video demo showing the continuous monitoring feature.	<a href="https://www.youtube.com/watch?v=2pK-PzOLvv4">https://www.youtube.com/watch?v=2pK-PzOLvv4</a>
Runtime Monitoring	Video demo showing the runtime assessment of a KPIs example through an IoT sensor.	<a href="https://www.youtube.com/watch?v=f8BFnUHSsQs">https://www.youtube.com/watch?v=f8BFnUHSsQs</a>

TABLE 1.1: List of the resources related to the developed artifacts.

In conclusion, this PhD thesis work has been developed in the context of the Centre for Urban Informatics and Modelling (CUIM)<sup>4</sup>, a national Italian project.

<sup>4</sup> <https://www.gssi.it/research-area/research-projects/cuim-project>

## 1.6 Thesis Outline

This thesis is organized in eight chapters. In Chapter 2 we give an overview of the background of the work described in this dissertation. Specifically, we start by describing the smart city domain, with a particular focus on the smart governance dimension. Then, quality evaluation systems are discussed. Lastly, we provide some background on MDE concepts by including also the multi-level modeling and models@runtime paradigms. Chapter 3 gives an overview of the current state of the art on smart cities assessment. In particular, we start by analysing the existing KPIs assessment frameworks from the perspectives of the process that they implement and the requirements they should address. Moreover, we give an overview on quality evaluation systems, by highlighting their limitations and potentialities. Then, we report the findings of an analysis about approaches dealing with smart cities modeling, before summarizing the identified limitations of the state of the art and deriving the research challenges that driven this work. In Chapter 4 we describe the developed model-based smart cities KPIs assessment approach, by first introducing the motivating scenario that justifies our work, i.e., smart decision-making in smart cities. Then, we start describing the proposed approach, namely *MIKADO*, and its constituent artifacts. In particular, we report the metamodels on top of which the approach relies, namely the smart city metamodel and the KPIs metamodel. We further explain how the KPIs assessment phase and the KPIs reporting process work, by also discussing how the presented approach supports KPIs evolution. Chapter 5 describes the architecture behind the *MIKADO* approach. Here, we show the architecture that we designed for KPIs assessment in smart cities in a flexible fashion, thus allowing different deployment style. In particular, we report a standalone deployment and an hybrid deployment implementations, presenting also the prototype of an extended *MIKADO* devoted to KPIs continuous monitoring. The online deployment pattern is discussed through hints of alternative styles and technologies. Instead, in Chapter 6 we show different types of evaluations performed incrementally while developing the *MIKADO* framework. Specifically, we show a demonstration case to prove the feasibility of the KPIs assessment approach. Then, we report the results of a survey collecting opinions about the understandability of the defined DSLs used to model smart cities and KPIs and two sets of experiments testing the performance of the evaluation engine's execution time. Lastly, the latency analysis of the service-based extension of the KPIs assessment approach is discussed. Chapter 7 proposes the abstraction of the KPIs assessment approach presented in Chapter 4 for the automatic quality assessment of subjects coming from different domains, further showing the domain-independence of *MIKADO*. In particular, we start by giving some motivations to the need of abstracting the assessment approach. Then, we provide an overview of the multi-level QES that we propose, by presenting different domains in which it may be applied. Thus, we propose

---

two multi-level hierarchies of models for the definition of subjects and quality metrics, respectively. Furthermore, we present a re-factoring of the evaluation engine artifact in order to support multi-level navigation of models and we show how the code-generation process devoted to graphical dashboards generation is also supported in the multi-level framework. Finally, in Chapter 8 we summarized the work presented in this dissertation, by highlighting how the developed model-based KPIs assessment approach answers to the stated RQs. Eventually, some future works are discussed whereas two already ongoing works are briefly presented.

## Chapter 2

# Background

In this Chapter, the background of the work described in this dissertation is provided. Specifically, Section 2.1 gives an overview of the smart city domain with a particular focus on the smart governance dimension. Then, quality evaluation systems are described in Section 2.2. Lastly, Section 2.3 provides some background on MDE concepts by including also the multi-level modeling and models@runtime paradigms.

### 2.1 The Smart City Domain

In the literature, we can find various definitions of smart cities [24]. Most of them are related with the development of cities in terms of sustainable economy, society, and environment. In this perspective, every initiative undertaken by the city has, as final objective, the improvement of the quality of living for its citizens, through the provisioning of *sustainable and smart services*, and the *proactive participation* of citizens (as well as tourists, commuters and any other final users of given services) in the *inclusive smart cities initiatives*. For this reason, one of the main goal of smart cities is to be transparent and to leverage on all sources of data useful for the purpose. For instance, one possibility is that of making use of an Internet of Things (IoT) sensor network in the city, to collect multiple types of different data (e.g., traffic flows, air quality, use of transportation means, tourists movements), in order to understand and even derive the different behaviours and needs of different types of users (e.g., citizens, tourists). From this knowledge, the implementation of initiatives aimed at improving the citizens quality of life can be conducted through the exploitation of ICT solutions integrated in the city as well as research activities and projects.

The smart city domain is characterized by its complex nature starting from the different dimensions composing it (e.g., [25], [26]). In particular, we refer to the work of Boyd

Cohen [26] that formalised six distinct smart cities dimensions, that we describe as follows:

- *Smart Economy*: it is characterized by the exploitation of ICT in economic activities. In other words, it refers to those activities aiming at creating an economic value for the city. For instance, initiatives promoting *smart tourism* that have economic implications [27].
- *Smart Environment*: it concerns the control and monitoring of environmental factors, such as pollution, planning of green areas, waste. Initiatives in this dimension target an effective and efficient use of public natural resources means (e.g., alternative energy sources) to avoid fossil fuels, to reduce carbon footprint, etc.
- *Smart Government*: it concerns the use of technology to enable open, transparent and participatory governments. The initiatives in this dimension are mainly focused in supporting decision-making processes for cities governments.
- *Smart Living*: it concerns the improvement of the quality of people's daily life and lifestyle. For instance, many initiatives in this dimension involve smart buildings, i.e., systems with appliances and services connected in a building's network (e.g., homes, hospitals), with the aim of optimizing resources consumption and management.
- *Smart Mobility*: it has the aim of improving local accessibility to mobility services and a smart and sustainable mobility, while also supporting social inclusion, reducing the environmental impact, etc. For this reasons, initiatives in this dimension mainly regard the development of sustainable, innovative and safe transport systems and applications.
- *Smart People*: it deals with the promotion of creativity, open-mindedness and participation in public life. This dimension includes initiatives about services for citizens (e.g., e-learning platforms, public administration online services), investment on and preservation of the human capital, and privacy-related solutions (e.g., privacy protection systems) [28].

This distinction, reported in Figure 2.1, was made to support the rankings of smart cities by framing their components in dimensions and sub-dimensions. In this way, the numerous and heterogeneous stakeholders are able to accurately benchmark the smart cities they are interested in at different depths and from different perspectives.

Moreover, the complexity and heterogeneity that clearly arise from the before-mentioned dimensions, highlights the involvement of different *stakeholders* playing diverse roles. In

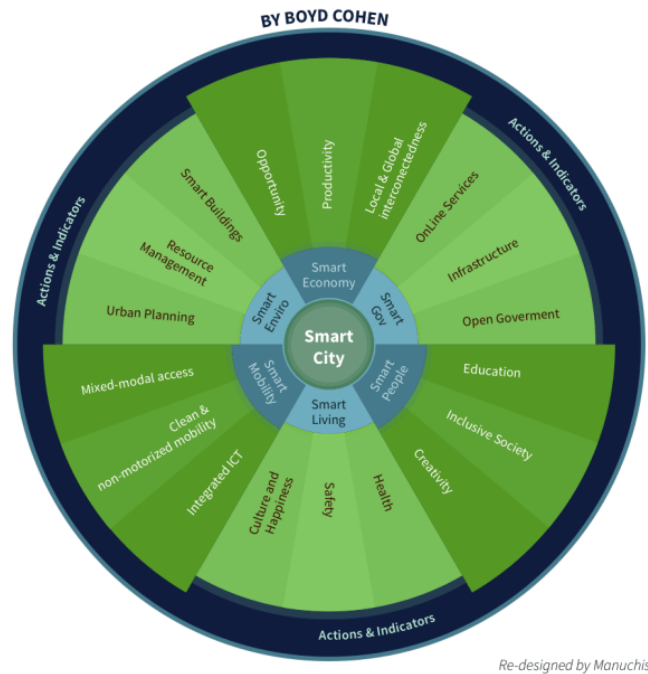


FIGURE 2.1: Boyd Cohen Wheel.

particular, in each dimension we can identify subjects, organized or not, that bear some interest in the development of the city [29]. From a general view, one of the main actor in the city is the *public administration* that has to face the limitations related to the city's sustainable growth, such as population growth, economic burdens, etc. The good quality of life is also a responsibility for the *companies in the public sector*. Another important role in the development of a healthier and more sustainable urban life is played by academic *researchers* that are in charge of creating new innovative solutions to face problems arising with globalization and climate change. Also *private companies* are playing an important role in the smart transition of cities. In particular, they are the main technology providers in the construction of infrastructures and smart solutions. Another important stakeholder of the city are the *citizens* themselves, since from their needs and problems, the whole smart transition process is guided. In this sense they play both a passive and an active role, in the sense that it is people's responsibility to make their issues a matter of public concern. In this perspective, the other actor that we can identify is the *social and third sector* which plays a community work in the interests of citizens.

The transition process of smart cities, indeed, has gained attention worldwide, producing an enormous amount of investments in innovative smart projects aimed at facing the ever-growing urbanization together with the urgent climate change. The already mentioned SDGs [10], are a clear demonstration of how much relevance the smartness process has for European cities. In fact, the European commission presented this list of objectives

as an urgent call for action by all countries, developed and in development, in a global partnership, to build smart strategies to improve health and education, reduce inequality and poverty, with a particular attention to the preservation of the environment.

### 2.1.1 Smart Governance

In this thesis, we particularly focus on the Smart Governance dimension. It is devoted to the development of initiatives whose aim is to make a government *open*, *transparent* and *participatory*. Usually, these initiatives include policies that use ICT to improve the collaboration between institutions and the smart city stakeholders [30]. Furthermore, it is important to distinguish between *electronic governance* and *smart governance* [31]. Electronic governance provides the application of technologies in the interaction between the government and the different stakeholders (e.g., citizens, businesses). Instead, for smart governance we mean the introduction of digital technologies in processing information and *decision-making*. In the literature, we find that the main aspects of a smart government should be *transparent governance* and *open data* [32]. These peculiarities should have the goal to create an accountable, collaborative and participatory government [33]. In this context, the adoption of ICT can provide opportunities to govern the complexity of urban systems in a more effective way by exploiting the availability of data, the automation in performing required calculations and the creation of crowd-participation platforms.

In the smart city domain, the smart governance exploits the assessment of Key Performance Indicators (KPIs) to support decision-making processes. Specifically, KPIs define raw set of values that can provide some information about relevant measures that are of interest for understanding the progress of a smart city in terms of sustainability and smartness. KPIs are elicited and defined by standardization bodies, such as the International Telecommunication Union [7], along with their *collection methodology*, *standard definitions* and *formulae*. However, multiple KPIs sources exist (e.g., CITYkeys [13], DigitalAQ [14], Minako *et al.* [34]). For instance, in Figure 2.2 we report an example of KPI to show how the authors have formalized KPIs in a structured way in the ITU documentation [7]. Since KPIs may be measured w.r.t. different granularities for different goals, they are defined on three levels, namely, dimensions, sub-dimensions and categories. In the reported example in Figure 2.2, the KPI *Green Areas* is shown. It measures the relation between green spaces and number of inhabitants. Specifically, this KPI belongs to the category *Public Space and Nature*, which, in turn, pertains to the sub-dimension *Environment* in the dimension *Environment*. In the reported documentation, further attributes are given, namely (1) *Definition* and *Rationale* describing the KPI meaning and how to interpret it; (2) *Methodology* giving hints on how to calculate

it; (3) *Unit* referring to the unit of measure to be used. Moreover, (4) the *Data Sources* attribute gives recommendations about where to retrieve data, such as the input parameters needed to calculate the specific KPI. Lastly, (5) the *SDGs Reference(s)* attribute points to specific SDGs to which the KPI may have impact.

Dimension	Environment				
Sub-Dimension	Environment				
Category	Public Space and Nature				
KPI Name	Green Areas				
KPI No.	EN: EN: PSN: 1C	Type:	Core	Type:	Sustainable
Definition / Description	Green area per 100,000 inhabitants				
Rationale / Interpretation / Benchmarking	<p>Green areas are important to the sustainability of a city. The benefits of green spaces include: capturing pollutants, reducing the “heat island” effect and providing recreational spaces.</p> <p>Green spaces can include parks, gardens, recreational areas, natural areas or other open green spaces.</p> <p>An improving trend and higher values are considered positive.</p>				
Methodology	<p>Calculate as:</p> <p>Numerator: Total area of green space in the city (hectares) (public and private).</p> <p>Denominator: One 100,000<sup>th</sup> of the city’s population.</p>				
Unit	Hectares / 100,000 inhabitants				
Data Sources / Relevant Databases	Data may be obtained through municipal parks and recreation departments, planning departments, aerial surveys or GIS data.				
SDG Reference(s)	SDG Indicator 11.7.1: The average share of the built-up area of cities that is open space for public use for all, disaggregated by age group, sex and persons with disabilities.				

FIGURE 2.2: Structure of the KPIs proposed in [7].

The different KPIs sources for sure share some commonalities in the way the KPIs are hierarchically structured and which metrics they measure but relevant differences in formats and denotations make them hard to combine and compare. For instance, Minako *et al.* [34] organize KPIs in layers, instead of dimensions like in the reported example in Figure 2.2. These documentations will be better discussed in Section 3.1.

## 2.2 Quality Evaluation Systems

With the rise of software development, the study of software quality also began. Research solutions to improve software quality are grown at the same rate with the demand for software products with increasing quality. An acceptable way to support quality management of software products have been the use of models [35]. 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 [36]. 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. This type of models were first developed by organisations and software industries because of their need to have specific quality models capable of specialised evaluation on individual components. For this reason, they are tailored to a particular application domain, and the importance of features can vary from model to model. On top of these quality models, Quality Evaluation Systems (QESs) are built, which are basically software systems providing quality evaluation results of given subjects. They perform subjects quality assessments w.r.t. evaluation requests and quality requirements contained in the quality models that they receive as inputs. QESs may carry out an evaluation process to accomplish their task, being implemented as a part of a higher-level system, as well as independent systems [15]. Indeed, the quality models given as inputs to such systems usually include 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.

## 2.3 Model-Driven Engineering

In many scientific contexts, methods of abstraction are used to represent complex aspects of reality, by using simplified representations in such a way to better manage complexity. These representations of reality are usually called *models*. In the context of software development this type of abstraction based on modeling is known as Model-Driven Engineering (MDE) [37]. Here, models are regarded as blueprints used to provide a complete and detailed specification of a system. These blueprints models can be further refined to create the system, through the use of code-generation techniques to minimize the coding tasks. In software engineering, abstraction is usually implemented with a computing-oriented focus, making it difficult to understand to people who have no language development expertise. MDE also combines *Domain-Specific modeling Languages* (DSLs) and transformation engines and generators [38]. In particular, the use of DSLs allows the involvement of domain experts in the development. DSLs are computer languages built with notations, constructs and abstractions peculiar to a particular application domain, offering substantial gains in expressiveness and ease of use over General-Purpose programming Languages (GPLs) for the domain in object, with corresponding benefits in terms of productivity and reduced maintenance costs. Thanks to their expressiveness, DSLs can be used also by users who have no expertise in programming and slightly low knowledge of the domain under definition [39]. DSLs formalize the structure, behavior,

and requirements of particular domains' applications. DSLs are defined by using meta-models, which concretize how concepts are inter-related in the modeled domain. The semantics and constraints associated with application domain concepts are also specified with metamodels. Transformations process models to produce other artifacts, e.g., other models, source code or other types of artifacts. Model transformations in combination with models (and consequently metamodels) define what is called *minimal MDE infrastructure*, but the defined models can be processed via model management operations. Model interpretation, for instance, is a way to process models programmatically, and trigger custom actions based on the model content. Model Validation instead, is the activity used to precisely define constraints and rules that cannot be defined at the meta-model level. All these model management operations can be chained or composed with automated tools, e.g., ANT or programming languages, to create more complex tasks. In order to automate these operations, at each step the produced model must be consumed by the following automatism, and this is possible only when some pre-requirements are respected. For instance, to compose model transformations, the output metamodel of each step must be contained in the input metamodel of the next step [40, 41]. This is a sort of compatibility check enabling the automated composition. Thus, *model transformations* envisage automated processes that take one or more source models as input and produce one or more target models as output according to a set of transformation rules. The automatising of these processes are very useful in the development of reverse engineering techniques, view generation, or refactoring [42]. Depending on the type and complexity of the model transformation process it can be named model-to-model (M2M) transformation, model-to-text (M2T) transformation, model merging, model linking, model synthesis, model mapping or model-to-code transformation, i.e., *code generation* [43]. In particular, code generation enables separation of concerns between application modeling and technical code, supporting maintainability and portability of software programs to different hardware and operating systems.

To sum up, a relevant feature of MDE approaches is that they enable the description of very complex systems with simplified models. In this way, we can think of MDE as a *reduction approach* because it allows systems to be described as an aggregation of complementary models. To better understand what is intended with model, we have to introduce the criterion of mapping. A model is something that maps with a real object or phenomenon. The classification of different types of models is based on the level of abstraction. In other words, we can have different models describing the same system at different levels of abstraction.

In the model-driven context, besides MDE, we have other important concepts. One of them is Model-Driven Development (MDD) that uses automation to generate code starting from the models. The other one is Model-Driven Architecture (MDA) that is an

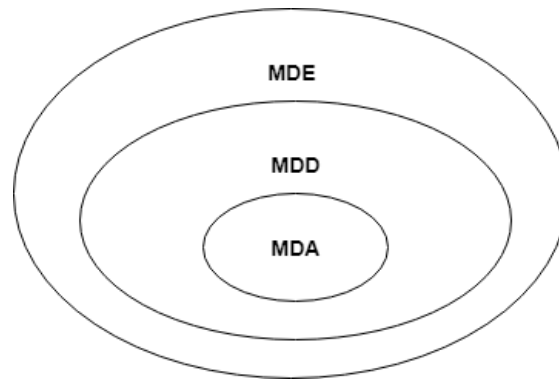


FIGURE 2.3: Relationship between MDE, MDD and MDA concepts.

approach for the execution of MDD using a set of standards. The relationship between these concepts and MDE is depicted in Figure 2.3. As we can see, MDE contains the other two concepts because it is utilised not only for development activities, but it also covers other domain-specific processes (e.g., analysis process).

Eventually, *model-centric approaches* differs from others that are more code-centric thanks to the concept of reuse of knowledge [44]. With a model-centric approach the business knowledge can be reused and only its implementation has to be reworked. Instead, in a code-centric approach the reuse implies that all the process has to be reiterated.

### 2.3.1 Multi-level Modeling

Traditional MDE approaches relying on two levels of modeling, envisage the description of a domain only at the meta-model level exploiting the natively meta-modelling facilities (e.g., type definition, type inheritance, data types, cardinalities). These facilities can not be used at the level of the model, thus, implying their explicit definition at the meta-model level anytime they are needed. To allow model elements to have a dual type/instance facet, multi-level modelling was introduced as a mean to define deep languages that span more than two meta-levels [45]. Indeed, multi-level modelling allows the definition of an unbounded number of levels of abstraction [23]. In this way model elements have a dual type-instance dimension [46]. This, not only enables flexibility in MDE when specific patterns arise in modeling approaches [45], but also supports the reduction of the complexity of models in certain scenarios, especially when the typical type-object pattern occurs. Multi-level modeling approaches are characterised by language support for expressing types and their instances. In particular, complex systems design can be widely supported by multi-level modelling thanks to features such as deep instantiation, potency, linguistic extensions [47, 48]. We report Figure 2.4 coming from [8] showing differences between traditional and multi-level modelling. In the former, in the meta-meta-model level the provided meta-modeling facilities defining the

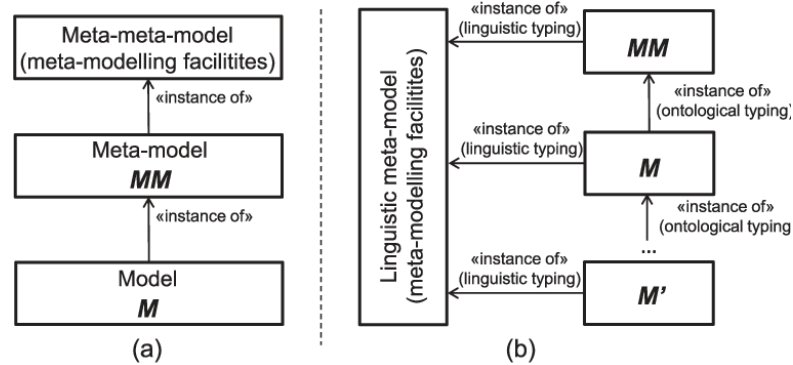


FIGURE 2.4: Comparison between traditional and multi-level modelling [8]

meta-modelling language are defined. This can be used only in the meta-model level. As regards multi-level modeling, in Figure 2.4 it is described thanks to the distinction between linguistic typing and ontological typing. Linguistic typing refers to the meta-modelling primitive used to create an element (e.g., model, class, field). Ontological typing refers to the classification of an element within a domain. It is worth noting that all linguistic meta-modelling facilities are available in all meta-levels. Meanwhile, the ontological typing is redefined and explicitly modelled in the linguistic meta-model, allowing the definition of an arbitrary number of meta-levels.

### 2.3.2 Models@Runtime

Model@runtime is a model-driven paradigm in which models are causally connected to the problem space. This concept emerged to face the problem related with software evolution that envisages the generation of a new solution by stopping the running system and replacing it with a new one. Further, runtime models can be used to observe dynamic state of the system and control it during execution. The role of runtime models in software evolution can be seen as a live development model that enables dynamic evolution [49]. In this paradigm, the model has a causal connection with a ever-changing system [50]. This principle assures that if the model is changed, the system will change correspondingly and transparently.

Models@runtime provides a unique model-based representation of the applications for both design- and run-time activities (i.e., for developers and operators) enabling an architectural pattern for dynamic adaptive systems that leverage models as executable artefacts supporting the execution of the system [51]. In this way, runtime models give an abstract representation of the underlying running system, supporting reasoning, simulation, and enactment of adaptation actions. Thus, the causal connection supporting the continuous evolution of the system with no strict boundaries between design-time and run-time activities, envisages that if there is a change in the running system this

will be automatically reflected in the model of the current system. Likewise, a change in the model influences the running system.

## 2.4 Discussion

In this Chapter we presented the domain of this thesis by giving an overview of the complex nature of smart cities with a particular focus on the smart governance dimension. We further gave evidence of how abstract is the way in which KPIs are released by standardization bodies. In particular, the reported KPI example in Figure 2.2 comes with a textual description even of its calculation formula and the indicated data sources for the needed input parameters to calculate it are not specific, but represent general indications.

Moreover, after the description of the concept of KPIs that are used for the quality assessment of smart cities, we introduce QESs that are commonly used to perform the evaluation of quality metrics of different subjects, mainly in the software development context. Here, we talked about quality models used to define quality metrics together with their attributes. Staying on the topic of models, we described the main concepts of MDE, by introducing also two specific paradigm, namely, MLM and models@runtime. We presented these modeling concepts since they compose the technical space in which the work presented in this thesis has been developed.

In the next Chapter we will discuss the state of the art about KPIs assessment in smart cities, QESs development and smart cities modeling, by highlighting the limitations identified in the literature.

## Chapter 3

# State of the Art and Motivation

In this Chapter, we investigate the current state of the art of smart cities assessment. In particular, we start by analysing the existing KPIs assessment frameworks, in Section 3.1. Moreover, we give an overview on quality evaluation systems, by highlighting their limitations and potentialities, in Section 3.2. Section 3.3 reports the findings of our systematic analysis about approaches dealing with smart cities modeling. The Chapter terminates with a summary of the identified limitations of the state of the art and the identification of the research challenges that led the work presented in this dissertation.

### 3.1 Smart Cities Evaluation through KPIs

As discussed in the previous chapter, the SDGs [10] provided by the European Commission as well as the KPIs defined on top of them by the ITU [7] and other entities, highlight the fact that KPIs assessment is gaining worldwide attention. In a more concrete perspective, KPIs can be seen as visual measures of performance, where a specific metric is calculated and graphically reported. A Microsoft research report [52] states that “*A KPI is then designed to help users quickly evaluate the current value and status of a metric against a defined target*”. To give a concrete example, the KPI *Green Areas* (GA), also described in Section 2.1.1, measures the green area in the city per 100.000 inhabitants [7]. It is calculated as in (3.1), taking in input two parameters, i.e., the total area of green space in the city, measured in hectares, and the city’s population.

$$GA = \frac{TotalGreenArea}{\frac{1}{100000} \times CityPopulation} \quad (3.1)$$

This way, KPIs can be used as measurement to support the *monitoring, prediction and evaluation* of the transition process of cities to smart cities. Moreover, the majority of

them (e.g., ITU KPIs [7], CITYkeys [13]) are 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.

One important aspect in the KPIs assessment process over smart cities is the *selection* of the appropriate set of KPIs that can help in understanding the performance of the city under analysis. Indeed, the set of useful KPIs may vary from city to city. To this aim, the authors in [53] define some principles that can be taken into account in the KPIs selection. These principles can be summarised in: comprehensiveness, comparability, availability, independence, simplicity, timeliness. Furthermore, the selection of the KPIs relevant for a smart city could be affected by different figures that may have an interest in the city, namely city officials and municipal administration, city residents and no-profit organizations, city services providers, and evaluation and ranking agencies (e.g., academia), among many. Since KPIs support the measurement of smart cities performances over time, they can be used in guiding policy assessments and enabling the comparison between different cities. Another aspect to be considered is related to the fact that KPIs for cities can change in time because of several causes, e.g., the evolution of the smart cities needs, the evolution of KPIs measurement logic. In fact, the selection of indicators is also driven by other aspects besides those listed in [53], that can change in time (e.g., data availability) [11]. For instance, the sustainability agenda [12], whose goal is to support the development of sustainable cities by providing a set of useful KPIs, was released in a first version in 2015 and an update followed in 2018, to deal with major changes impacting KPIs measurements and/or data restriction (e.g., GDPR, the new European data protection law).

Moreover, smart cities are different from one to another, depending on their geographical implications, stage of economic development, population growth, available services. In particular, regarding the geographical implications, given that every country has different conditions, 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<sup>1</sup>. Indeed, different smart cities can be interested in specific KPIs. For this reason, when dealing with KPIs assessment we face the need for *KPIs customization*, where, for customization we mean the smart city-driven selection of appropriate KPIs to be evaluated on that given smart city. In this perspective, the customization of KPIs must be driven by the smart city subject of the assessment and its peculiarities, notwithstanding that KPIs definitions and formulae constantly adhere to those provided by the standardization bodies.

---

<sup>1</sup> Polis 4.0 - Smart City Index 2018 <https://bit.ly/3xnyHEF>

The complex nature of smart cities, which actually are *systems of systems*, makes the smart decision-making and the KPIs assessment even more challenging tasks. Moreover, as just mentioned, *KPIs continuously evolve* [16] w.r.t. the evolution of cities' needs and context resulting from the *digital revolution* [11], which is taking place in the last years. For instance, the advent of electrical vehicles changed the urban scenario, by carrying out new needs (e.g., finding the optimal positioning of car chargers). Another example may be the advent of 5G that influenced the assessment of KPIs dealing with the network coverage (e.g., *Wireless Broadband Coverage* KPI) and previously involving only 4G and broadband connections.

In conclusion, despite the availability of different KPIs guidelines and the fact that, in general, the KPIs are measured through not too complex calculations, several challenges arise. Specifically, (i) the heterogeneity of the different guidelines providing diverse KPIs classification, (ii) the need for customizing KPIs over smart cities, and, (iii) the dynamicity of KPIs evolving over time, highlight the challenges of developing a smart cities KPIs assessment approach, dynamic enough to be applicable to any city, given any type of KPIs, and able to manage their evolution over time.

### 3.1.1 KPIs Assessment Frameworks

In the literature, KPIs are used in the assessment of different subjects coming from many heterogeneous domains (e.g., buildings renovation [54], manufacturing [55], logistic, about how to monitor the supply chain [56]). To this aim, despite we are interested in the assessment of smart cities, we also analysed the use of KPIs in other domains and contexts, to identify the used approaches and frameworks. After our investigation, we grouped the found frameworks devoted to KPIs assessment in three main categories, namely, *Manual approaches*, *Spreadsheets-based approaches* and *Web-based platforms*.

**Manual approaches.** These type of approaches are used when the calculations to be performed are few and very trivial. We are aware about them through our discussions with experts of the domain. However, not much documentation is provided in the literature for such approaches. This because, the KPIs are defined manually w.r.t. the personal expertise of the user who is commissioning the assessment. Moreover, also the collection of the input parameters needed for KPIs measurements is manually performed. In the approaches performing manual assessment, the obtained results are filled into reports to be shown and discussed with the other stakeholders. The format of these reports is unique for the specific final purpose of the KPIs assessment and can represent an issue for the comprehensibility of the analysis especially for non-experts stakeholders and for the knowledge sharing among stakeholders. Of course, even if the manual process can



be precise, it is not scalable since it is not automated and when the data grows or the calculations get more complex, the evaluation can suffer of procedural delays. Moreover, such approaches are definitely error-prone due to their nature and present some usability issues for non-domain experts.

**Spreadsheets-based approaches.** The approaches belonging to this category are characterised by the collection of data delegated to an external data source (e.g., a MS SQL database) and the calculation and visualization of KPIs performed on Excel spreadsheets exploiting features as Power Pivot<sup>2</sup>. The contexts in which this type of approaches are exploited are different. For instance, Abreu *et al.* [57] present a spreadsheets-based approach in an industrial context exploited for decision-making purposes. Here, 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. Also Jannach *et al.* [58] underline the problematic aspect of spreadsheets quality by presenting an approach of Model-Based Diagnosis of faults in spreadsheets. On the same trend, Luckey *et al.* [17] and Cunha *et al.* [59] propose approaches exploiting object-oriented modeling and MDE techniques to generate and evolve spreadsheets before using them, in order to reduce error-proneness in spreadsheets-based approaches. Besides the benefits coming from the exploitation of common and open spreadsheets (e.g., multiple functionalities supporting mathematical calculations and data analysis), they show several limitations [60]. In particular, KPIs assessment frameworks relying on the use of spreadsheets are strictly domain-dependent, thus negatively affecting reusability. Another issue we observed is related to the fact that it is totally in charge of the user composing the spreadsheets to keep separate the subject definition from the KPIs definition and formulae. This may raise a lack of separation of concerns (SoC) which could represent a problem if the spreadsheet is not well structured. In particular, changing the subject of the evaluation can lead to copy and paste activities, with all the related problems. Moreover, even 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. We may have usability issues even 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 [61]. For instance, if the database schema of the connected source changes, new inconsistencies can impact

---

<sup>2</sup> <https://bit.ly/3dT1zwV>

the formulae in the spreadsheet. It is worth noting that thanks to spreadsheets' good degree of openness, if KPIs formulae need to be changed or customized, or evolve over time, they can support that. These types of operations have to take into account the limitation on the number of records that can be stored without degrading. This limitation can affect the scalability of the framework when applied over multiple and complex smart cities in order to measure hundreds of KPIs. Moreover, with spreadsheets of big dimensions, also the readability is compromised.

**Web-based platforms.** In this category we can find approaches for KPIs assessment provided as online platforms. For instance, among Web-based framework, Bosch *et al.* [13] provide an online framework allowing users to insert the already calculated KPIs values, given a predefined set of smart cities KPIs, and to visualize them with graphical representations. However, the tool does neither envisage automatic calculation nor retrieving of data. Another example is proposed by Moustaka *et al.* [62] which present a framework to evaluate smart cities in terms of maturity by exploiting international standards for KPIs [63] and urban data. However, to execute the KPIs assessment the injection of urban data is still performed manually. One major limitation of Cloud or Web-based platforms is that, usually, they are not released as open. Moreover, being often conceived as multi-tenant platforms, they hardly support customization. Indeed, Web-based platforms tend to be abstract in order to accommodate multiple clients because the customization for a single-user is not convenient in terms of costs and strategies. On the contrary, as regards graphical visualizations, Web-based platforms usually offer very powerful tools showing the results of the KPIs evaluation, although they are not open to choose different types of graphical representations. Indeed, the lack of openness can be a problem, since usually there is no active engagement of domain experts involved in designing, operating, and controlling activities. In this way, KPIs definitions are embedded in the frameworks and the users can only get the results of their measurement in the pre-chosen graphical representations and export formats. Indeed, most of the found KPIs assessment frameworks offer pre-packaged KPIs definitions, that the user can select in order to get the results for the subject. The highlighted lack of openness raises the fact that this type of platforms are not flexible enough to support evolution or customization. Eventually, in contrast with the previous two categories of KPIs assessment framework, 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.

The analysis of the state of the art about frameworks for KPIs assessment supported us in drawing the fundamental requirements that these frameworks should meet, discussed in Section 3.1.1.1. Further, we derived the overall abstract development process that we

think should be followed to realize a KPIs evaluation framework, presented in Section 3.1.1.2.

### 3.1.1.1 Requirements Perspective

Looking at the drawbacks and limitations of the identified KPIs assessment frameworks categories, we derived the following requirements that these frameworks should meet, in order to overcome the limitations of existing approaches described above. Specifically, they are:

- *Automation* in the calculation of KPIs, in order to be able to perform complex evaluations with a lot of data and with multiple output formats (e.g., textual files, CSV).
- *Separation of Concerns (SoC)* as regard the distinction between the modeling of the subject under evaluation and the KPIs definition and formulae. We refer here also to the SoC between the competences of the concepts related to the KPIs definition and application.
- *Domain-Specificity* of the language used to define KPIs for domain experts and stakeholders.
- *Usability* of the KPIs assessment framework with user-friendly UI or languages to specify required data and KPIs formulae.
- *Openness* of KPIs formulae to inspection and modification in order to manage the KPIs evolution and customization.
- *Graphical* visualization of KPIs evaluated over a given subject in a comprehensible way for non-experts stakeholders and for knowledge sharing among them.
- *Flexibility* of the features provided by the framework, e.g., in the KPI formula definition, in choosing a different type of graphical representation of the results, in the export format of the results.
- *Scalability* of the framework w.r.t. the number of subjects under evaluation and/or KPIs involved in the assessment.

In Table 3.1 we report whether the three main categories of KPIs assessment frameworks support (✓), partially support (∼) or not support ( - ) the extracted requirements.

TABLE 3.1: Evaluation of KPIs assessment frameworks.

Approach	Requirements							
	Automation	SoC	Domain-Specificity	Usability	Openness	Graphical	Flexibility	Scalability
Manual	-	✓	-	-	-	✓	~	-
Spreadsheets	✓	~	-	-	✓	✓	~	-
Web platforms	✓	✓	✓	✓	-	✓	-	✓

In conclusion, we highlight here that none of the detected category provides a way of performing automatic KPIs assessment and, at the same time, of modeling both subjects and KPIs in an uniform and customizable way. In particular, as regard manual approaches, their main limitation stands in the fact that they are performed accurately on a subject and this make them not scalable both in terms of applicability on different subjects and performance with a lot of calculations and data. Moreover, performing such calculations manually increases the error-proneness of the approach. Instead, what resulted from the analysis of spreadsheets-based approaches is the of lack of SoC between the subjects modeling and the KPIs definition and formulae. Indeed, having the definition and calculation of KPIs highly coupled makes it hard for KPIs experts the change of the subject of the evaluation and also the modification of KPIs formulas, due to the specific language underlying the framework (e.g., macros and procedures in Excel). In this case, we can have also usability issues due to the lack of an user-friendly UI for certain operations (e.g., connect an external database to inject the data needed to calculate the KPIs about the subject). While, the main limitation of web-based platforms is the fact that they are not open and they do not provide customization features for KPIs formulas. Often, these platforms provide forms in which the user can insert the input parameter for a pre-packaged set of KPIs in order to get the report of the results. Although, they offer powerful tools of data visualization they do not support flexibility for evolution or customization.

### 3.1.1.2 Process Perspective

Given the requirements for KPIs evaluation frameworks arising from the limitations of existing approaches, we further derived the abstract development process that we envisage should be followed to realize these frameworks. In Figure 3.1 we report the abstract development process that we derived from our analysis.

We distinguished the phases of the process in two stages, i.e., *Development stage* and *Operation stage*. In particular, for the development stage we identified the *analysis* phase, during which the activities of identifying and defining the KPIs are performed. During this preliminary phase, domain experts are involved in order to select the right set of indicators and coordinate the corresponding data collection. This phase is conducted in

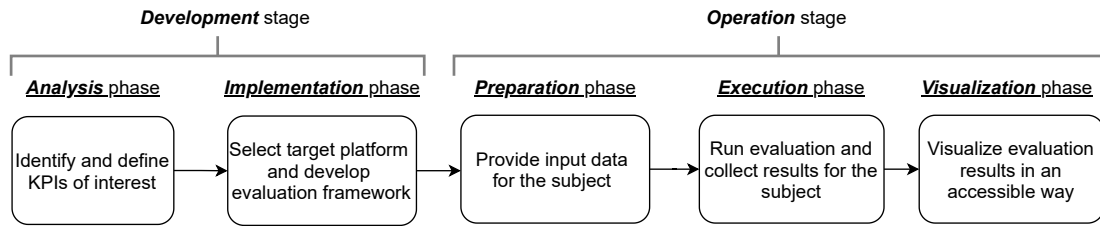


FIGURE 3.1: Basic process for the development of frameworks for the KPIs assessment.

such a way to be as transparent as possible and allow the application of the framework over multiple subjects of the same type. This could be possible if the framework would allow the possibility of customizing the KPIs definition for the different subjects. After the identification and definition of the KPIs we have the second phase of the development stage, namely the *implementation* phase. During this phase the development of the KPIs assessment framework starts w.r.t. the requirements defined in the previous phase. Then, we enter in the *Operation stage*, that, in turn, begins with a *preparation* phase. In this phase, the user has to retrieve and prepare the input data needed to use the framework. This input data will be used as the parameters of the assessment of the subject, thus they can be historical or hypothetical data. After finding the data to use for the evaluation, the *execution* phase can begin. The result of this phase will be the calculated KPIs values for the subject under analysis that can be reported through graphical representations during the next *visualization* phase.

It is worth mentioning in this context Low-code development platforms (LCDPs) [64, 65] as a possible solution to build a KPIs assessment framework devoted to users coming from different backgrounds. Indeed, LCDPs are characterized by easy-to-use visual environments that would allow smart cities' stakeholders with no coding skills to contribute in the process formalised in Figure 3.1. However, since these types of platforms are mainly used for specific types of applications, mostly in industry, they are usually built and provided as static black boxes, whose languages and behaviours are not customizable or open. The risk with LCDPs is similar to the one related with web-based platforms, namely, fostering usability by penalising flexibility and openness.

## 3.2 Quality Evaluation Systems

In the literature, we found several specialized tools devoted to the quality assessment of different subjects, e.g., software systems [66], modeling artifacts [67–69], software architectures [70]. Specifically, as introduced in Chapter 2, Quality Evaluation Systems (QESs) are basically software systems providing quality evaluation results of given *subjects*. They perform subjects quality assessments w.r.t. evaluation requests and quality

requirements contained in the *quality models* that they receive as inputs. QESs may carry out an evaluation process to accomplish their task, being implemented as a part of a higher-level system, as well as independent systems [15]. Indeed, the quality models given as inputs to such systems usually include quality definitions and corresponding metrics. QESs have been often implemented for the domain-specific evaluation of different subjects. Overall, the different QESs share some commonalities, namely the overall objective of assessing a given subject passed as input, where the assessment is executed on top of a quality definition, required by the user. Despite these commonalities, QESs are strongly domain-dependent, implying that they are typically re-implemented from scratch from a domain to another, by using existing tools, e.g., spreadsheets, or developed as independent systems, e.g., Web or standalone applications.

An example of measurement mechanism for evaluation in software development is given by the Goal-Question-Metric (GQM) approach by Van Solingen *et al.* [71]. 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). Still in the software development context, García-Munoz *et al.* [66] present a platform devoted to the assessment of the quality of software projects to estimate their quality, performance, management and costs. The authors state that in software development projects it is needed to merge quality metrics from different sources in such a way to have a wider range of monitoring for possible faults. For this reason they implemented their platform in such a way to allow integration of different analysis tools. Instead, Cardarelli *et al.* [70], present an approach for the evaluation of micro-services architectures w.r.t. a defined set of software quality attributes. Here, the authors decided to keep the definition of quality metrics in an independent dedicated ecosystem. In particular, both the modeling language used for defining the architecture and the quality attributes computed on the model, are independent from one another thanks to MDE techniques. As regards MDE technical space, Basciani *et al.* [67] propose an approach devoted to the definition of quality models supporting the quality assessment of modeling artifacts. The authors want to overcome typical limitations in this context, such as limited extensibility, artifact specificity, and manual assessment, which might lead to informal, subjective, and non-reproducible assessment processes. Meanwhile, van Amstel *et al.* [68] present a set of metrics for assessing the internal quality of ATL model transformations artifacts. The

authors highlight the fact that metrics alone are not enough for assessing quality and in the assessment process of the same collection of model transformations, it is needed the opinion of ATL experts. Lastly, Ambrus *et al.* [69] show a tool to measure the complexity of UML models (i.e., state charts) by proposing transformations to reduce the complexity. Here, it is highlighted how quality metrics are able to increase the quality of the system and save development costs as they might find faults earlier in the development process.

As seen in Cardarelli *et al.* [70], the use of DSLs in the abstraction of the definition process of both quality characteristics and evaluation subjects may support the robustness of the results of the process. Moreover, implementing a model-driven approach implies the reduction of the time to market, and the facilitation of the definition of the artifacts involved in the entire process. In this perspective, DSLs are often devoted to users which are not programmers and therefore they offer advantages similar to the usage of programming languages, but for domain experts. Indeed, every time a QES is required in a specific domain, 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.

It is worth highlighting, 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., [72]), we can find several quality models that present a hierarchical structure, as seen in the KPIs documentation. Lastly, Deissenboeck *et al.* [73] highlights one of the main issues in software quality assessment, 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.

Similarly to the smart cities assessment scenario, we can identify other scenarios in which quality assessment represent a not trivial task. For instance, a similar scenario is the social impact assessment. Activities and initiatives may generate positive impacts on the society and the subject under evaluation itself (e.g., public administration institution, private companies), thus, affecting human life and financial loss as happening for QES in the software development scenario. Moreover, depending on the nature of the subject under evaluation, the metrics to be used to assess the social impact have to be customized consequently. Also in this context of social impact assessment we have a variety of heterogeneous application scenarios (e.g., Higher Education Institutions (HEIs) [74–76],

finance [77]) that make the standardization of the assessment process complex. Related to the impact on human life, another scenario in which the quality assessment is directly affecting it is the Covid-19 risk assessment over geographical areas. During the pandemic period we have seen how the different nations of the world managed the monitoring of the spreading of the virus. Specifically, every subject defined a list of indicators and thresholds to map the level of risk w.r.t. different geographical areas. In this scenario, the different policies applied by every nation made it difficult to standardize the assessment process.

### 3.3 Modeling Smart Cities

We have already said that the smart city domain is made by different dimensions and involves diverse stakeholders. These are some of the aspects that contribute to make it so complex to handle. To reduce this complexity in the management of smart cities that clearly involves different processes given the different contexts and aspects, exploiting an high level of abstraction to generalize these processes might be helpful for the smart cities managers. Indeed this would support reuse and optimization. As discussed, MDE may result helpful in the abstraction of the different aspects of smart cities, and it has already been used in this context. For instance, DSLs have been exploited. To give a few examples, Rosique *et al.* [20] present a DSL to model Smart City Systems (SCSs). The authors face with the high heterogeneity of devices and protocols of communication that can help in creating a SCS. In this work, it is demonstrated how a DSL with concepts of the domain can be understandable also by experts with no knowledge in software engineering. Instead, Matar [21] exploits MDE to afford the issue involving the collaboration between the different stakeholders in the smart city domain. The use of DSLs in such contexts helps in terms of understandability and analysis in a more easy way w.r.t. General-purpose Programming Languages (GPLs) source code patterns that are more complex or usually not well defined [78]. The risk of using a GPL is to be too abstract and to struggle in the development phase.

As discussed, the main objective of this thesis, consists in realizing a KPIs assessment framework for smart cities that implements a systematic, reusable and customizable way for evaluating smart cities. In general terms, it can be seen as a QES, thus it relies on two main inputs artifacts, such as the subject of the evaluation and the quality model with the metrics to evaluate it. In order to use MDE techniques and approaches, thus to exploit all their benefits due to abstraction, model transformations and code generation, as a first step modeling artifacts for both smart cities and KPIs would be required. As regards KPIs, although a uniform way to model them does not exist, from our state



of the art analysis in Section 3.1 we found out that, according to the majority of the existing guidelines, they follow a common hierarchical structure, are already defined in an abstract way, and are measured according to basic operations. These aspects lead us to think that it should be feasible to provide a standard way to model them in order to be used as input for a QES. To the contrary, w.r.t. a smart city, it is more difficult to embed all its complexity in a modeling artifact to be used as input for a QES, that is the subject to evaluate. With these premises, we deeply investigated the literature to better understand the research status about *smart cities modeling* approaches, both as a whole or by considering their multiple dimensions, in an MDE perspective. This work eventually led to the realisation of a systematic mapping study on smart cities modeling [1]. For sake of space, in the following section we report only the main findings of our analysis, which of course have been used as a driver for the realization of the model-based framework presented in this dissertation, in the following chapters.

### 3.3.1 Smart Cities Modeling Approaches Review: Main Findings

To better understand the research status about smart cities modeling approaches, we decided to perform an analysis of the existing literature about the topic. Specifically, we performed our research investigation by following the process for systematic mapping studies in software engineering proposed by Petersen *et al.* [79]. The process we followed for accomplishing our analysis started with the definition of research questions delineating our review scope and focusing on the modeling of smart cities. In particular, we formulated the following six RQs belonging to three different domains, namely problem, solution and scientific community domains:

— **Problem Domain:**

**RQ1** *Which Smart City dimensions have been modeled the most?*

**RQ2** *Which application fields in the Smart City domain have been modeled the most?*

— **Solution Domain:**

**RQ3** *Which modeling approaches have been used to represent Smart Cities?*

**RQ4** *What is the maturity status of smart cities modeling approaches?*

— **Scientific Community Domain:**

**RQ5** *When did the contributions on modeling Smart Cities occur?*

**RQ6** *Where have the contributions been published?*

Starting from these RQs, we derived the search query used to conduct a literature search on different digital libraries. In particular, we composed the search query by combining the keywords *smart city* and *model* and their synonyms (e.g., smart cities, modelling, modeling) that identify the domain of the study and the investigated approaches. The automated searches were performed in four digital libraries<sup>3</sup> to guarantee a certain level of coverage and quality of relevant papers. After this search phase we obtained a first set of publications related to the topic, namely **1325** papers. This set then underwent a screening during which all the duplicates were removed and the defined inclusion/exclusion criteria were applied, aiming at identifying the publications not relevant to answer the research questions. After the screening of publications we get a set of **575** unique papers. On this set of publications, we performed a keywording phase using abstracts. From one hand, it allowed to detect further out of scope papers, by means of a *title analysis* focused on title, abstract, and keywords. From the other hand, it supported the classification of publications w.r.t. different aspects. In particular, the classifications performed on the selected set of publications aimed at finding out the maturity level of the detected studies. Specifically, we categorized the relevant publications according to the so-called *research type facets*, introduced by Petersen *et al.* [79], namely experience, opinion, philosophical, solution, validation, and evaluation. Furthermore, we categorized publications also according to (i) the *Technology Readiness Level* (TRL) scale<sup>4</sup>, (ii) the *smart city dimensions*, based on the Boyd Cohen Wheel standard [26], (iii) the *modelling approaches* exploited or proposed as contribution in the publications, (iv) the presence of a specific implemented *tool* based on the presented contribution (e.g., web applications, prototype, services), and (v) the *application field of the contribution* in the reference domain. Consequently, as output of the classification step we obtained a set of **106** relevant papers. The list of the analysed publications is available **here**.

We analysed this final set of relevant papers to better understand how smart cities have been modeled from the perspective of the different research communities and in the context of diverse application fields. The goal of our analysis is that of collecting the existing SC modeling approaches in order to provide an overview about the state of the art on smart cities modeling. In particular, we aim to identify possibly emerging SC dimensions, application fields, modeling approaches, publication venues, and to determine existing trends, if any, and potential future directions that SCs modeling should take. Furthermore, we want to give an overview on the maturity level of the analyzed works to understand to which extent the provided solutions are technically sound and robust.

In the following paragraphs, we summarize the main findings of our systematic mapping study on modeling approaches for smart cities. The findings are grouped according to

---

<sup>3</sup> The used digital libraries are: *IEEE Xplore*, *Scopus*, *Association for Computing Machinery (ACM)*, *dblp computer science bibliography* <sup>4</sup> <https://enspire.science/trl-scale-horizon-europe-erc-explained/>

the three research questions' domains. Of course, major details on both the setting of this studies and all the results are available in [1].

**Problem Domain.** In the problem domain, to answer *RQ1*, we were interested in which SC challenges are addressed and in which SC dimensions, thus to understand which are the ones that attracted more the interest of the research communities. It is worth noting that besides the dimensions coming from the Boyd Cohen wheel [26], we considered a further dimension specifically for *Smart Things* to keep track of those publications whose contribution lies mainly in the IoT world. In Figure 3.2, we reported the results for this analysis. In particular, we found out that the smart city dimensions that elicited mostly the research interest from the modeling perspective in the last decade are the *smart governance*, *smart environment* and *smart people* ones. For the other dimension we observed a quite uniform distribution. The main finding of this analysis is that, in the literature, there is an uneven coverage of the smart city's dimensions, from a modeling perspective. In particular, it highlighted a limited coverage for the *smart economy*, *smart mobility*, *smart living* and *smart things* dimensions. Considering the multi-dimensional nature of smart cities, this current weak coverage of some SC dimensions might highlight different interest of researchers and practitioners in the diverse dimensions of the smart city domain, considering that some of them received quite more attention.

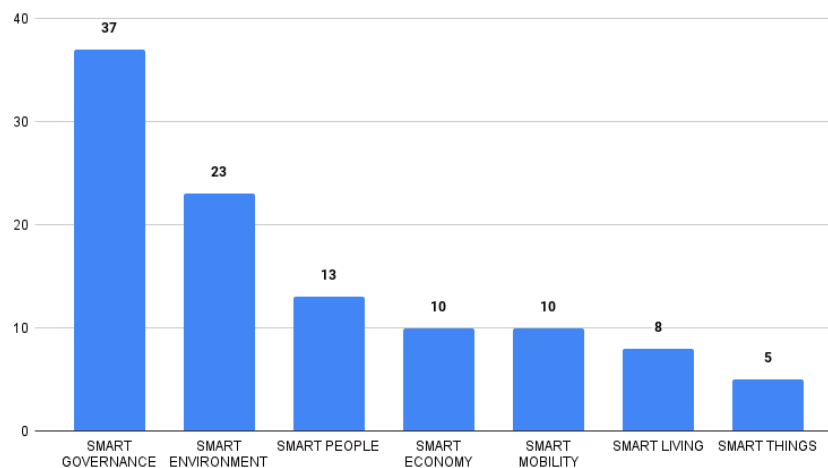


FIGURE 3.2: Distribution of publications among the smart city dimensions.

To answer *RQ2*, we also extracted the *application fields* from the publications set, and we derived a taxonomy of these *application fields* organized in different hierarchical levels. The classes used to classify the publications emerged during the keywording phase, thus the taxonomy arose in a bottom up fashion. We refer to [1] for the detailed taxonomy, while we report a summary here. From our analysis it resulted that smart city modeling has been largely exploited in the *Smart City Domain* field (50 publications), made by

*Smart City Design*, *Smart City Management* and *Smart City Administration* sub-fields. The application fields *ICT* and *Quality* closely follow with **36** and **25** publications, respectively, while the *Stakeholders* field closes the ranking with **16** publications. Indeed, from the obtained results about the application fields of SC modeling approaches we observed that the *Stakeholders* field, including *Participation* and *Communication* sub-fields, is still an unexplored area. This very marginal subset not only highlights a lack of research works dealing with inclusion, participation, communication and knowledge sharing devoted to smart city stakeholders, but also confirms another aspect that makes the smart city domain so complex, namely the actors who bear some interest in the city are many and heterogeneous.

**Solution Domain.** The inspection of the solution domain concerned the investigation of the *modeling approaches* currently used to design smart cities, in order to answer **RQ3**. In particular, we extracted the modeling kinds and techniques used in every publication to model the covered SC domain or subdomains. From our analysis, it emerged that *business*, *architecture* and *ontology* modeling are the most prominent detected approaches, with **15**, **14** and **12** publications, respectively. However, the distributions of publications among the detected model kinds is quite uniform. Thus, a prominent modeling approach is not emerged. This might be due to the complex nature of the smart city domain. Since most of the publications focus only on specific aspects and dimensions, then, the choice of the most appropriate modeling approach is driven by specific research goals.

During the inspection of the solution domain, we also investigated the status of the research works dealing with smart city modeling. Thus, to answer **RQ4**, we performed multiple analysis. First, we classified publications according to the *research type facets* [79], which are used to distinguish between empirical and non-empirical papers and to categorize them based on the proposed solution and level of validation. As reported in Figure 3.3, this analysis turned out that the majority of the publications (i.e., around 54%) belongs to the *Philosophical* and *Solution* types. These papers propose a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework or by proposing a novel solution for a problem. Meanwhile, only 24% of contributions are validated through experiments in laboratory environment (*Validation*) and 19% of them performed an evaluation research through the solution implementation and its evaluation (*Evaluation*). Furthermore, to understand the technological maturity of the contributions coming from the investigated publications, we calculated their *Technology Readiness Level (TRL)*<sup>5</sup> that is a metric used into the EU funded projects arena that, in turn, was defined by NASA in the 1990's as a means for measuring the maturity

<sup>5</sup> <https://enspire.science/trl-scale-horizon-europe-erc-explained/>

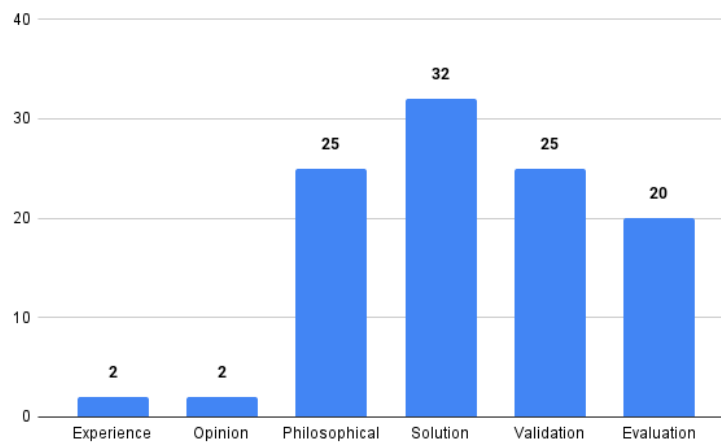


FIGURE 3.3: Number of publications w.r.t. the type facets.

of a given technology. We performed this analysis only on those publications classified as *Solution*, *Validation* and *Evaluation*. In particular, we measured the maturity level of the used technology, development and design. Specifically, the maturity level is measured according to a scale ranging from 0 to 9. The results of our analysis are shown in Figure 3.4 where we reported the distribution of the calculated TRL values for the three considered research facets. It emerged that for the solution facet we have mostly TRL values that go from 1 to 2, with the exception of a very few works showing a TRL equals to 1 and 3. For the validation facet, the TRL goes mostly from 3 to 8, with the median equals to 5. Lastly, the evaluation facet has TRL values mostly of 7 and 8. From the results, we can notice here that no contribution shows a TRL value of 9, namely presenting an actual system proven in operational environment, meaning that we have not found mature and stable systems or framework. Moreover, we investigated deeper the contributions and we observed that only 15 publications over 106 contribute with a developed tool, of both type open-source or proprietary.

Looking at the results of these two analysis, we can state that around the 96% of publications, present a contribution mostly in the form of a solution to a given problem, not yet evaluated (i.e., Solution facet), or a taxonomy or a conceptual model (i.e., Philosophical facet). In addition, the TRL measurement has shown that no work presents an actual system proven in operational environment, namely with a TRL of value 9. This is further proven by the very low percentage of papers (i.e., 15 over 106) contributing with a tool, where the majority of them have not yet been distributed in the real world but only tested in a sub-set of conditions and environments.

**Scientific Community Domain.** During the inspection of the scientific community domain, we analysed the distribution of publications over the years in order to answer

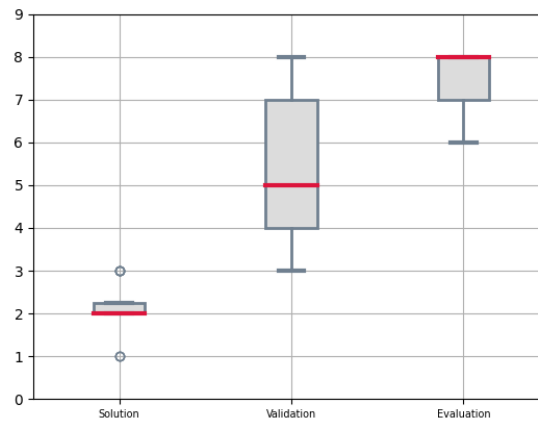


FIGURE 3.4: Boxplot with the TRL values w.r.t. the three facets.

**RQ5.** In this way we were able to observe the advent of the first contributions on the topic and the trend of publications over the years. In Figure 3.5, we reported the distribution of publications per year w.r.t. their type, i.e., *Articles* in scientific journals and *inProceedings* publications to scientific conferences. From the analysis it emerged that research about modeling smart cities started in 2011, simultaneously with the advent of the concept pointing out the smartness of cities [9]. From the publications type perspective, the results show that InProceedings publications appeared in 2013, and from 2014 they started to be increasingly much more than articles. Noteworthy, in 2018 we can observe a significant decrease for both publications types, while in 2020 a change in the trends of the two series can be seen, i.e., InProceedings publications started decreasing while articles started increasing.

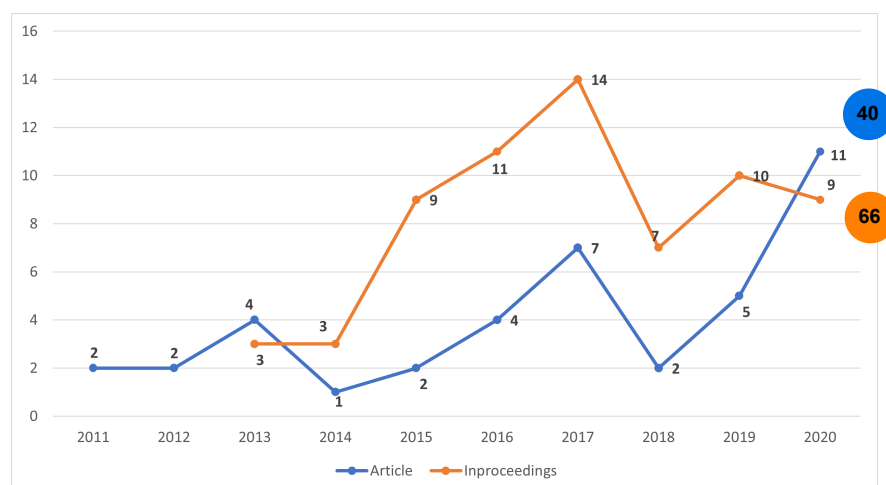


FIGURE 3.5: Number of publications per year w.r.t. the publications type.

Concerning the scientific community domain, to answer **RQ6**, we also inspected the

most prominent venues of both types of publications, i.e., inProceedings and articles. Specifically, for each publication we analysed its venue and also the information about the corresponding *Scientific Area*<sup>6</sup>. The analysis of inProceedings publications' venues resulted in more than 53 different conference venues. The four most prominent conference venues are: (i) the *IEEE International Smart Cities Conference (ISC2)* with 4 publications belonging to the *Computer Science* and *Social Sciences* scientific areas; (ii) the *Smart Cities Symposium (SCSP)* with 3 publications from the *Computer Science* and *Decision Sciences* scientific areas; (iii) the *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)* with 3 publications belonging to the *Computer Science* scientific area; (iv) the *Winter Simulation Conference (WSC)* in the scientific areas of *Computer Science*, *Engineering*, and *Mathematics*. The remaining venues show also multidisciplinary scientific areas. The articles contributions, instead, span over 32 different journals. For this publication type we inspected also the *Quartile* assigned to each journal, i.e., the journal impact index measuring the quality of scientific journals, ranging from Q1 (the highest ranked journals) to Q4 (the lowest ranked journals). The most prominent venues are *ISPRS International Journal of Geo-Information* and *IEEE Access* with 3 publications each, fitting into the *Q1* quartile. The first belongs to the *Earth and Planetary* and *Social Sciences* scientific areas. Instead, the second one to *Computer Science*, *Engineering*, and *Material Sciences* scientific areas. Thus, both papers' type refer to interdisciplinary communities, in line with the multi-disciplinary nature of smart cities, with a prevalence of *Computer Science*, *Engineering* and *Social Sciences*.

Our main finding in the scientific community domain is that it emerged a wide distribution of different conferences and journal venues where contributions on modeling SCs have been published. Moreover, also considering the contributions in the different scientific areas, separately, it does not emerge a reference venue/community. This observed multi-disciplinarity highlights again the complex nature of the smart city domain which is studied and modeled from different perspectives and by an heterogeneous group of researchers.

### 3.4 Research Challenges

In this Section, we extracted the Research Challenges (RCs) emerged from the limitations identified in the state of the art. These are the RCs we faced in developing a model-based framework devoted to KPIs assessment over smart cities. Indeed, our objective was to provide a comprehensive methodology supporting (i) systematic and uniform modeling

<sup>6</sup> We retrieved the scientific area from *Scimago* <https://www.scimagojr.com/>, i.e., a publicly available portal that provides journals and conferences rankings.

of smart cities and KPIs, *(ii)* automatic KPIs measurement, and *(iii)* intuitive reporting of assessed KPIs. Furthermore, we envisaged the need for an evaluation framework made in such a way that it can be reusable also for the assessment of subjects coming from different domains than smart cities.

Having an abstraction of the smart city would enable the smart governance to have a constant picture of the city itself. Moreover, such abstraction of the smart city could also easily support its performance monitoring that, in turn, would support the collaboration among the stakeholders involved in the decision-making processes. With these premises, and considering that one of our objectives was the systematic and uniform modeling of smart cities and KPIs, we first started from the inspection of the state of the art about existing smart cities modeling approaches, to then move to a deeper analysis on the KPIs side. In particular, we analysed the literature on smart cities modeling approaches w.r.t. different aspects. First, from the obtained results we observed an uneven coverage of the SC dimensions. As regards the application fields of the investigated publications, the stakeholders field, including participation and communication sub-fields, emerged as a quite unexplored research area. Looking deeper at the contributions reported in the analysed publications, we further observed that a prominent modeling approach of smart cities does not exist. Furthermore, looking at the maturity level of the contributions, the majority of publications present solutions to a given problem, not yet evaluated, or a taxonomy or a conceptual model. We also observed that no work presents an actual system proven in operational environment, meaning that we have not found mature and stable systems or framework. This is further proven by the very low percentage of papers contributing with a tool, where the majority of them have not yet been distributed in the real world but only tested in a sub-set of conditions and environments. Lastly, looking at the scientific community to which the analysed publications belong, we observed multi-disciplinary venues in-line with the complex nature of the smart city domain which is studied and modeled from different perspectives and by an heterogeneous researchers communities.

Considering the drawbacks emerged from our systematic mapping study on smart cities modeling approaches, and in view of the need of a smart cities model on top of which a KPIs assessment can be performed, we extracted the following research challenge (RC):

**RC1:** Provide a uniform way of modeling *smart cities* and their complexity.

After the investigation of the smart city domain in terms of modeling approaches, we performed an analysis of the state of the art about existing KPIs assessment frameworks, in order to detect possible re-usable and automatic frameworks. During this review, we analysed generic KPIs assessment frameworks that we categorized in manual approaches,



spreadsheets-based approaches and Web-based platforms. As regards manual approaches in which data retrieving and KPIs definition are performed manually, even if they can be precise, they suffer from lack of scalability. This is due to the fact that they are not automated and when the data grows or the calculations get more complex, the evaluation can suffer of procedural delays. Moreover, such approaches are definitely error-prone due to their nature and present some usability issues for non-domain experts. Instead, in the category of spreadsheets-based approaches we classified those frameworks for which the collection of data is delegated to an external data source and the calculation and visualization of KPIs performed on Excel spreadsheets. This type of frameworks benefit from a good degree of openness but suffer from relevant limitations. One of them is the lack of separation of concerns since the definition and calculation of KPIs are highly coupled. This implies that users must learn and be trained to compose formulae in Excel. Moreover, the verbose syntax and complex expert features (e.g., external data source configuration) of Excel can raise also usability issues. Another important limitation regards the number of records that can be stored without degrading. This limitation can affect the scalability of the framework when applied over multiple and complex smart cities in order to measure hundreds of KPIs. Meanwhile, Web-based platforms, in which the KPIs calculation is performed as a functionality of an online platform, clearly show a better degree of scalability with the novel front-end development technologies and back-end and persistence engines technologies. Indeed, they usually offer very powerful tools to represent the results of the KPIs evaluation in graphical dashboards. But one major limitation is that, usually, they are not released as open. Moreover, these platforms tend to be as abstract as possible to be used by a wide range of customers and this hardly affects customization. Due to the fact that KPIs definitions are embedded in the frameworks, the users can only get the results of their measurement in the pre-chosen graphical representations and export formats. This makes Web platforms not flexible enough to support evolution or customization of the KPIs definitions.

From the limitations just discussed, arisen from the analysed KPIs assessment frameworks, and considering the core objective of this work, we extracted the following RCs, which driven us in the development of our KPIs assessment framework:

**RC2:** Provide a systematic methodology allowing smart cities to *define, measure* and *visualize* the KPIs of interest in order to efficiently assisting the decision-making processes by also supporting:

**RC2.1:** *KPIs evolution* over time w.r.t. the evolution of smart cities needs and contexts;

**RC2.2:** *KPIs customization* based on the specific smart city under evaluation in terms of selection of appropriate KPIs to be calculated.

Eventually, since one of our objectives was that of applying the model-based KPIs assessment framework also in the assessment of subjects from different domains, we inspected the state of the art about Quality Evaluation Systems in the literature, in order to extract both potentialities and limitations of these tools. First, we observed that QES are developed for the assessment of different types of subjects coming from different domains. Then, we highlighted how quality assessment of smart cities can be re-conducted to a typical QES process, namely, they share the overall objective of assessing a given subject, which could be a smart city, passed as input, where the assessment is executed on top of a quality definition, such as a set of KPIs. Looking at the literature about QES in different domains, we observed that, despite the commonalities, QESs are strongly domain-dependent, implying that they are typically re-implemented from scratch from a domain to another, by using existing tools, e.g., spreadsheets, or developed as independent systems, e.g., Web or standalone applications. Indeed, every time a QES is required in a specific domain, 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.

Considering the limitations of typical QESs, which are not generalizable enough, thus not easily reusable in different domains, we extracted the following RC:

**RC3:** Make the systematic assessment methodology *generalizable* such that to be applied in the quality assessment of subjects coming from different domains than the smart cities.

### 3.5 Discussion

In this Chapter we presented the actual state of the art of smart cities assessment. First, we introduced smart cities evaluation through KPIs by giving a definition and an example of this type of metric. Then, three categories of typical KPIs assessment frameworks are described, namely, manual approaches, spreadsheets-based approaches and web platforms. In particular, we highlighted, for each category, benefits and limitations and we concluded that none of the detected category provides a way of performing automatic KPIs assessment and, at the same time, of modeling both smart cities and KPIs in an uniform and customizable way. Looking at these drawbacks we derived a set of requirements that KPIs assessment frameworks should meet and we formalised the abstract development process that we envisage should be followed to realize these frameworks. Secondly, we investigated the literature about QESs development to identify the commonalities between the different domains in which this type of tools are exploited. Lastly, we reported the results of an analysis of the literature to better understand the research

status about smart cities modeling approaches, both as a whole or by considering their multiple dimensions, in an MDE perspective.

According to the limitations identified in the state of the art we derived the research challenges that guided the research work reported in this dissertation. In particular, in the next Chapter we give an overview of the smart cities KPIs assessment approach that we developed in order to support smart cities and KPIs modeling, automatic assessment and real-time graphical representations generation.

## Chapter 4

# *MIKADO*— a Smart City KPIs Assessment Modeling Framework

In this Chapter we describe the developed model-based smart cities KPIs assessment approach by first introducing, in Section 4.1, the motivating scenario that justifies our work, i.e., smart decision-making. Then, Section 4.2 gives a general overview of the proposed approach, namely *MIKADO*. In Sections 4.2.1 and 4.2.2 the metamodels on top of which the approach relies are reported, namely the smart city metamodel and the KPIs metamodel. The KPIs assessment phase is reported in Section 4.2.3. Furthermore, the KPIs reporting process is explained in Section 4.2.4, by also expliciting how the presented approach supports KPIs evolution in sub-section 4.3. The Chapter ends with some discussions in Section 4.4.

### 4.1 Motivating Scenario: Smart Decision-Making

Smart governance managers monitor the performance of smart cities by means of the assessment of KPIs [12]. Measuring the *sustainability* and *smartness* of a city w.r.t. predefined objectives supports smart city decision-making processes. An example of the importance of monitoring smart cities goals stands even in the aforementioned SDGs [10]. Indeed, entities releasing lists of KPIs also highlight to which SDG each indicator contributes. In this way, a city can monitor its performance w.r.t. to the SDGs and implement actions to adjust or maintain its trend. For this reason, KPIs are build as *visual* measures of cities' aspects performances that support the evaluation and understanding of the current status and trend of a metric w.r.t. specific targets [52]. Understanding the performance of cities is of great relevance also for those cities that are involved in their transition to become smart cities. In this case, the measurement of KPIs supports the

prediction and evaluation of the transition process. It is worth noting that most of the standard lists of KPIs are defined in such a way to be applicable to all cities, enabling in this way benchmarking and rankings of cities. These features can be very helpful for the sharing of knowledge between different smart cities, for instance in the implementation of common smart projects for the resolution of similar problems.

The KPIs assessment and the consequently smart decision-making are not trivial tasks in smart cities that are characterized by a high level of complexity, whereby they can actually be thought as *systems of systems*. The crucial aspects that have to be taken into consideration in the context of KPIs assessment are several. First of all, we cannot ignore the fact that smart cities, together with their resources and needs, are continuously evolving. We also highlighted that certain cities may be in the beginning or in the middle of their smart transition. Indeed, following the behaviour of cities also *KPIs continuously evolve* [16]. Moreover, also the selection of the KPIs of interest for a city is influenced by factors that can change in time (e.g., data availability), in particular in these years during which the *digital revolution* [11] is taking place. For instance, in the last few years the advent of electrical vehicles changed the urban scenario, by carrying out new needs (e.g., finding the optimal positioning of car chargers). Also the advent of 5G changed the way how the Wireless Broadband Coverage is calculated in a city.

A second crucial aspect in the KPIs assessment regards the heterogeneity of smart cities. Indeed, they can differ not only for their dimensions (e.g., small, medium and metropolitan cities) but also for other aspects related to their stage of economic development, population growth, available services. However, the geographical implications can have a great influence on the selection of the set of KPIs of interest. Even in smart cities' rankings, the comparisons of cities as the results of KPIs assessment, are reported by differentiating among metropolitan cities, small and medium-sized cities<sup>1</sup>. Furthermore, different smart cities can be interested in specific KPIs. Thus, although all smart cities use KPIs definitions and formulae provided by the standardization bodies, they should have the possibility of selecting only the KPIs appropriate for their evaluation. Consequently, we highlight the need for *KPIs customization* claiming that the selection of KPIs must be driven by the subject smart city and its peculiarities.

We can conclude by saying that smart decision-making processes may be supported by smart cities benchmarking tools providing automatic KPIs assessment over different cities, thus implying a certain degree of customizability. Not disregarding an easy and intuitive usability both in the usage and in the visualization of cities' reports, in order to support the understanding of their performances and trends w.r.t. their goals and other similar cities.

---

<sup>1</sup> Polis 4.0 - Smart City Index 2018 <https://bit.ly/3xnyHEF>

## 4.2 The MIKADO Approach

Looking at the limitations of the smart decision-making scenario, at the drawbacks of existing KPIs assessment frameworks, as seen in Section 3.1.1, and at the consequent challenges highlighted in Section 1.2, we decided to exploit the benefits coming from the use of MDE techniques to develop a model-based approach devoted to the smart cities assessment. We reported in Figure 4.1 the main phases of our approach. Starting from

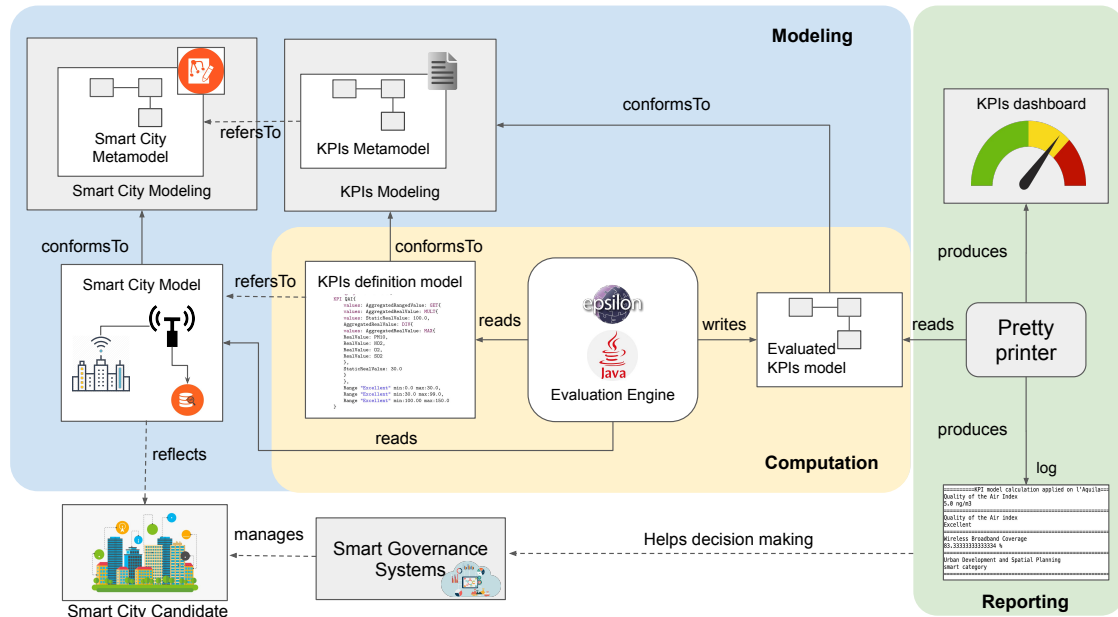


FIGURE 4.1: Overview of the MIKADO approach.

the *Modeling* box, here we find all the modeling artifacts devoted to design tasks. In particular, here we have modeling tools for the smart cities (*Smart City Modeling*) and for the definition of KPIs (*KPIs Modeling*). The *Smart City Metamodel* is the output of a study over the smart city domain, its concepts and the relation among them. Also the *KPIs Metamodel* comes from an investigation of the guidelines about how KPIs can be measured, to understand what type of calculations and data they require. After this analysis, we designed the KPIs metamodel in such a way that it allows the definition of a *unique KPIs Definition Model*, valid for different smart cities that can change over time in terms of nature and number of KPIs, to answer to both *evolution* and *customization* of these metrics.

Shifting the focus on the *Computation* box, we have the *Evaluation Engine* component that takes in input the *Smart City Model* of the *Smart City Candidate* and the *KPIs Definition Model*, both models conforming to the corresponding metamodels. This component interprets and calculates the modeled KPIs for the candidate city from the input

models and produces, via in-line transformations, the *Evaluated KPIs Model*, with the concrete values for the KPIs identified in the KPIs Definition Model.

The produced model is, in turn, the input for the *Reporting* box in charge of its transformation into a graphical dashboard through a code generation process, implemented via a *Pretty Printer* component, which produces also a detailed log of the evaluation. 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 governance managers in looking for causes of smart cities problems highlighted by the KPIs.

## 4.2.1 Smart Cities Metamodel

As seen in Section 3.3, during the analysis of the state of the art about smart cities modeling, a uniform way serving our scope did not emerged. To this aim, we decided to develop a DSL to describe smart cities in terms of data and their sources. In particular, we reported the defined *Smart City Metamodel* in Figure 4.2. Indeed, this metamodel has a

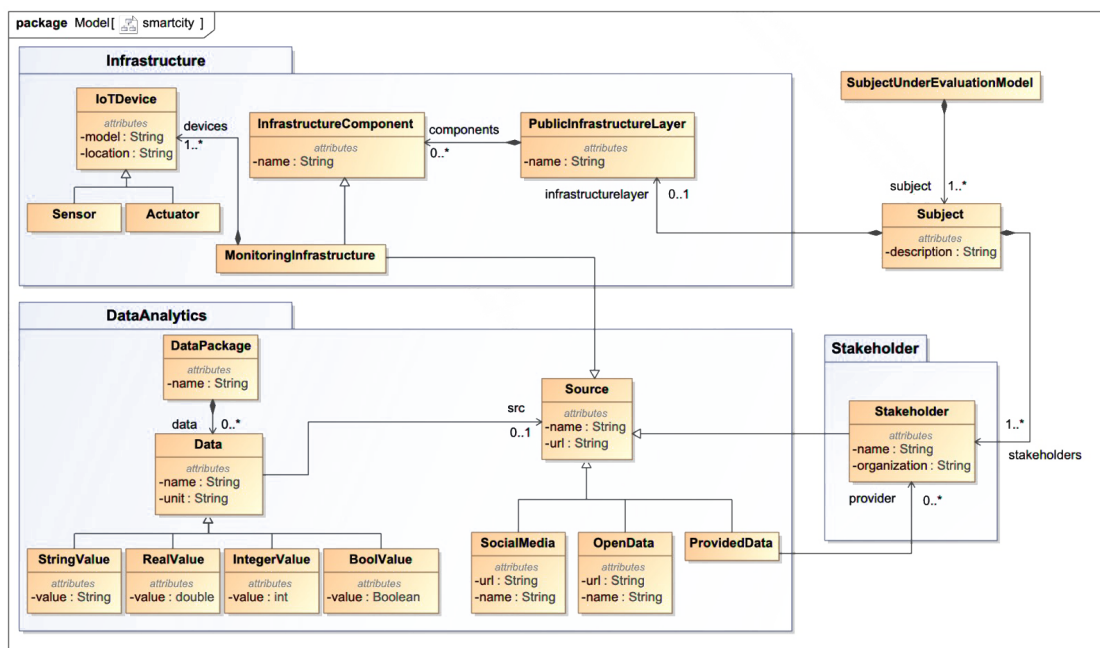


FIGURE 4.2: Smart City Metamodel.

particular focus on the data analytics context in order to support smart decision-making, in this way, the model instances of the smart city not only contain the data values useful for the KPIs assessments, but it also keep trace of the sources and relationships of the information collected in the city. In particular, the *SubjectUnderEvaluationModel* is specified as a composition of *multiple* *SmartCity* entities, to allow the modeling of 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`. In the data analytics package, we find concepts like `Source` that can be specialized into three different data sources, namely `ProvidedData`, `OpenData`, `SocialMedia`. From this sources may be generated 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, to allow the description of IoT systems, we defined, in the infrastructure package, another type of source, called `MonitoringInfrastructure` that can be composed of several `IoTDevices` instantiatable in `Sensors` and `Actuators`. This type of infrastructure is a specialization of the `InfrastructureComponent` composing the `PublicInfrastructureLayer` that can be developed in a city. Lastly, in the `Stakeholder` package we defined the `Stakeholder` concept to model every type of smart cities stakeholders (e.g., private and public institutions, companies), which can act or not as data providers.

#### 4.2.2 Key Performance Indicators Metamodel

After the study on KPIs guidelines and documentations defined by standardization bodies, we defined a DSL to describe KPIs structures and calculation formulae coming from different sources. Specifically, we report the defined *KPIs Metamodel* for the KPIs definition in Figure 4.3. In this metamodel we defined the concepts resulted from the investigation that we did on KPIs documentations. Thus, data types, data sources and operators correspond to those envisaged by the standardization bodies that provide these KPIs lists. The fact that we used a specific metamodel does not imply that it cannot evolve with more complex concepts (e.g., vectors, matrixes, non-standard operators). According to the KPIs documentation taken into account (e.g., [7, 13, 14]), we modeled the `KpiModel` as a composition of multiple `Dimensions` containing themselves different sub-dimensions and `Category` entities. KPIs are associated with only one category and each category is composed by multiple `Parameters`. Each KPI has a `Value` that, in turn, is associated to a `ValueType` that can be specialized into two main different types, i.e., `RangedValue` and `CalculatedValue`. The former is required to model those KPIs whose calculation result has to be compared against a `Range` of values to get the final KPI measure. The latter, is used for those KPIs that only need to perform a calculation. This can be made over different `SingleValues` of different types (i.e., `StaticRealValue`, `BoolValue`, `RealValue`, `IntegerValue`, `StringValue`). Every single value is associated to a `Parameter`. Moreover, calculations can be made also on `AggregatedValues` of different types (i.e., `AggregatedBoolValue`, `AggregatedRealValue`, `AggregatedIntegerValue`, `AggregatedStringValue`, `AggregatedRangedValue`). In every aggregated value has to be defined an attribute *operation* whose type is specified by the enumeration `Operation`, which defines the typical



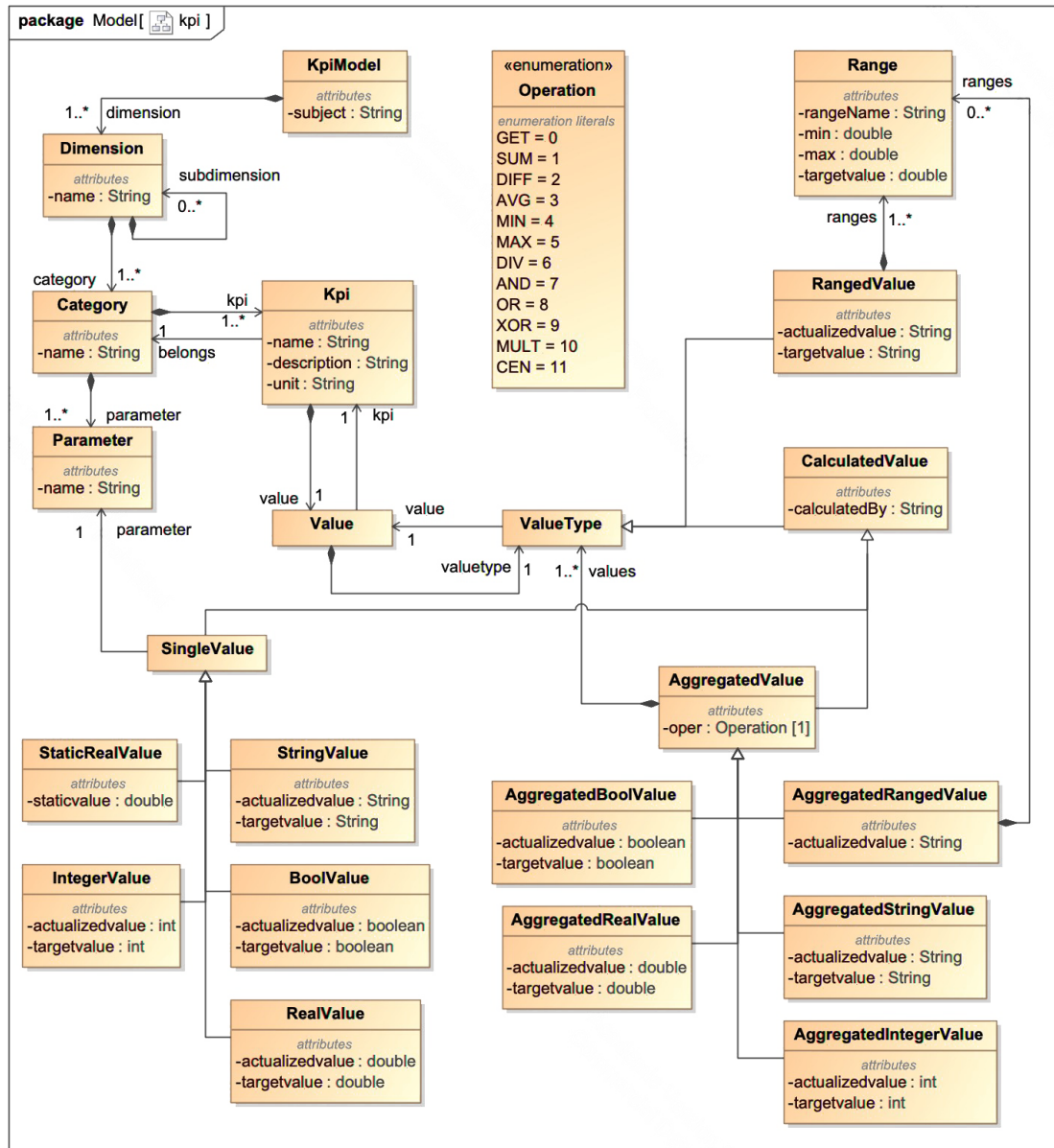


FIGURE 4.3: KPIs Metamodel.

operations that can be used to calculate KPIs (e.g., MAX, AVG). In some of the types of `SingleValue` and `AggregatedRealValue` the `targetvalue` attribute can be instantiated with the desired value for the defined KPI.

Once defined the input models of the KPIs assessment process by following the rules coming from the two metamodels just described, the computation phase devoted to the calculation of the defined KPIs over the smart city under assessment can begin, as we will see in the next Section.

### 4.2.3 KPIs Assessment

As shown in Figure 4.1, the **Evaluation Engine** is the computation component in charge of the calculations in the proposed KPIs assessment process. This component takes in input the KPIs definition model and the **Smart City model** of the candidate subject under analysis. The output of the execution of the engine is the **Evaluated KPIs Model** for the subject in which the KPIs values are actualized with the calculated ones via in-line transformations. The engine prints also a report of the KPIs assessment for the candidate city as a textual log, summarizing the results of the calculations for all the defined KPIs it assessed, by exploiting the required parameters in the smart city model. Indeed, the engine reads the KPIs definitions where all the parameters needed for the calculations are declared and search them into the smart city model by name-based references. If the engine does not find all the needed parameters for the calculation of a KPI, it does not execute the specific calculation.

The evaluation engine performs all the calculations thanks to the operations declared in the KPIs definitions. It translates the operations that it reads in the declared **AggregatedValues** into the language implementing the engine. The engine has been implemented in such a way that it would be easy to add new operations types. It is only a matter of adding a line of code to the evaluation engine script in which we implement the translation from the enumeration *Operation* element to the actual mathematical operation. For instance, looking at several KPIs guidelines we found that KPIs are often calculated w.r.t. *one 100.000th of the city population*; for this reason we defined a *CEN* operation that performs the division of the number of city inhabitants per 100.000.

### 4.2.4 KPIs Reporting

Besides the reporting of the results of the KPIs assessment as a textual log by the evaluation engine, the *Pretty Printer* component shown in Figure 4.1 produces also graphical dashboards to report the results. The generation of dashboards relies on model-to-text (M2T) transformations that we implemented in order to have a graphical visualization of the *Evaluated KPIs* model. In particular, we defined templates with rule-based model-to-text transformations that converts the models into SVG/HTML files.

We implemented the transformations in such a way that the generated dashboards are organized in views. These reflect three concepts, i.e., the overall KPIs model, dimensions and categories. For every view we implemented a M2T template, namely, for the general view of all the KPIs declared in the model, for the view of each dimension, and for each category. In this way the user can navigate the generated graphical dashboard at different

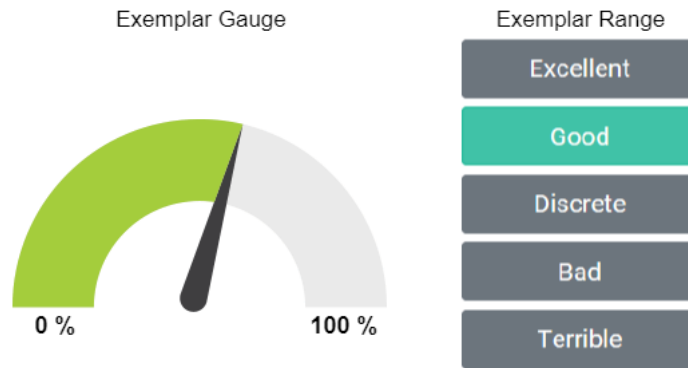


FIGURE 4.4: Example of gauge and range charts.

KPIs granularity. In the templates we defined some rules to generate different graphical charts depending on both the KPIs value types and the actualized values returned by the assessment. Essentially, we implemented four types of charts:

- *gauges* (left hand side of Figure 4.4) representing the KPIs calculated with an aggregation of real numbers (e.g., MIN, MAX, AVG);
- *ranges* (right hand side of Figure 4.4) representing the 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* representing KPIs of type integer (see Figure 6.4);
- two *buttons* representing boolean KPIs with the one resulting from the assessment selected (see Figure 6.4).

It is worth noting that for those KPIs represented by a gauge it is important to have a *target value* defined, since the percentage reported are calculated w.r.t. this value. This type of representation is very useful to understand the performance of the KPIs w.r.t. the pre-defined goals that a city may have. Also this artifact may be customized in order to generate other types of representations. For instance, radar charts can be implemented to represent multiple KPIs results to give a global overview of a specific KPIs dimension or category.

### 4.3 Supporting KPIs evolution

The evolution of KPIs may concern the update of input parameters or their sources, either their calculation. When this evolution process is about the data taken as input from the KPI, the smart city model has to be updated. This type of change might need

more than one modeling step in the smart city model to be implemented and this can affect also the KPIs definition model. We grouped the changes that can be performed on the smart cities models into three categories [80, 81]:

- **Additive changes:** when we add new modeling concepts related to smart cities (e.g., new stakeholders). This type of changes do not impact existing KPIs definition models. For instance, the addition of new data providers that in the KPIs definition will be available as parameters.
- **Subtractive changes:** when we remove one or more modeling concepts from the smart city model. This changes can impact the KPIs definition models. For instance, if we remove a data provider to which a parameter in the KPIs definition model is referring.
- **Structural changes:** regarding changes affecting the structural functioning of the assessment process. For instance, if we change the name of a data value in the smart city model then it should change also in the KPIs definition model.

Meanwhile, when there is a change in the structure or in the calculation of the KPI, it is only the KPIs definition model that has to change. Also in this case, we can implement the changes in few modeling steps on the KPIs definition model with the same categories of changes as explained before for the smart city models:

- **Additive changes:** when there is the advent of a new dimension, category, parameter or KPI for a city. For instance, if we are in the case of a new KPI two situations may happen, 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.
- **Subtractive changes:** when a KPI or a parameter become obsolete. In this case, it is only required to delete the KPI calculation or the parameter in the KPIs definition model.
- **Structural changes:** when we have to change the calculation logic of an existing KPI.

All the mentioned changes will be immediately reflected in the evaluation process [82] with the proposed approach, since they regard only the models given as input to the

evaluation engine and not the deployment of the framework. This aspect confirms the separation of concerns requirement that we extracted as one of the requirements that KPIs assessment frameworks should hold (see Section 3.1.1.1).

## 4.4 Discussion

In this Chapter we gave an overview of the model-based KPIs assessment approach over smart cities that we developed in order to support smart decision-making processes. In particular, the tool supporting our approach provides automatic KPIs calculations and intuitive reporting features, and is customizable on different cities.

Specifically, we described the two metamodels, i.e., the *Smart City Metamodel* and the *KPIs Metamodel*, designed to allow smart city designers and KPIs experts to model the desired assessment scenario. Moreover, we showed how they support customization of both smart cities and KPIs, and eventually, they can be extended with further concepts, if required by the domain experts. Furthermore, we reported how, on top of this metamodels, we developed the components devoted to KPIs calculations and reporting.

In the next Chapter, we will give more details about the implementation and the flexible architecture on top of which we developed *MIKADO*. In particular, we will provide alternative deployment patterns, to prove the flexibility and the technology-independence of the designed architecture.

## Chapter 5

# Architecture and Implementation

The architecture behind the *MIKADO* approach is described in this Chapter. In Section 5.1 we show the architecture that we designed for KPIs assessment in smart cities in a flexible fashion allowing different deployment style, namely *standalone*, *hybrid* and *online*. In particular, the standalone deployment implementation is reported in Section 5.2. A hybrid deployment style is described in Section 5.3, presenting also the prototype of an extended *MIKADO* devoted to KPIs continuous monitoring. Moreover, in Section 5.4 is specified the online deployment pattern and further technologies with which the different architectural components may be developed are proposed. Lastly, some discussions are reported in Section 5.5.

### 5.1 *MIKADO* Flexible Architecture

In Figure 5.1 we report the flexible architecture devoted for KPIs assessment in smart cities. We sketched the main functionalities by six macro-components that in the figure are represented with grey boxes. We also depicted the control- and data-flow between them via solid and dashed arrows, respectively. Specifically, data-flows concern the exchange of both models and raw data, usually persisted via XML or customization of an interchange format called XMI. Moreover, on the top of the figure we reported a white box in which we put the **Stakeholders** interested in the use of the realized solutions compliant with the architecture, e.g., *Municipalities*, *Smart Governance Team*, *Ranking Agencies*, *Researchers*. We put in the bottom-right side of the figure another group of stakeholders devoted to the design and implementation of the main software components and modeling artifacts, i.e., the *Developers Team* and the *KPIs Experts*. With the term *developers* we mean also modelers, DSL engineers and possibly software architects.

Every type of stakeholder can have different granting access, i.e., read, write, execute, depending on their profile.

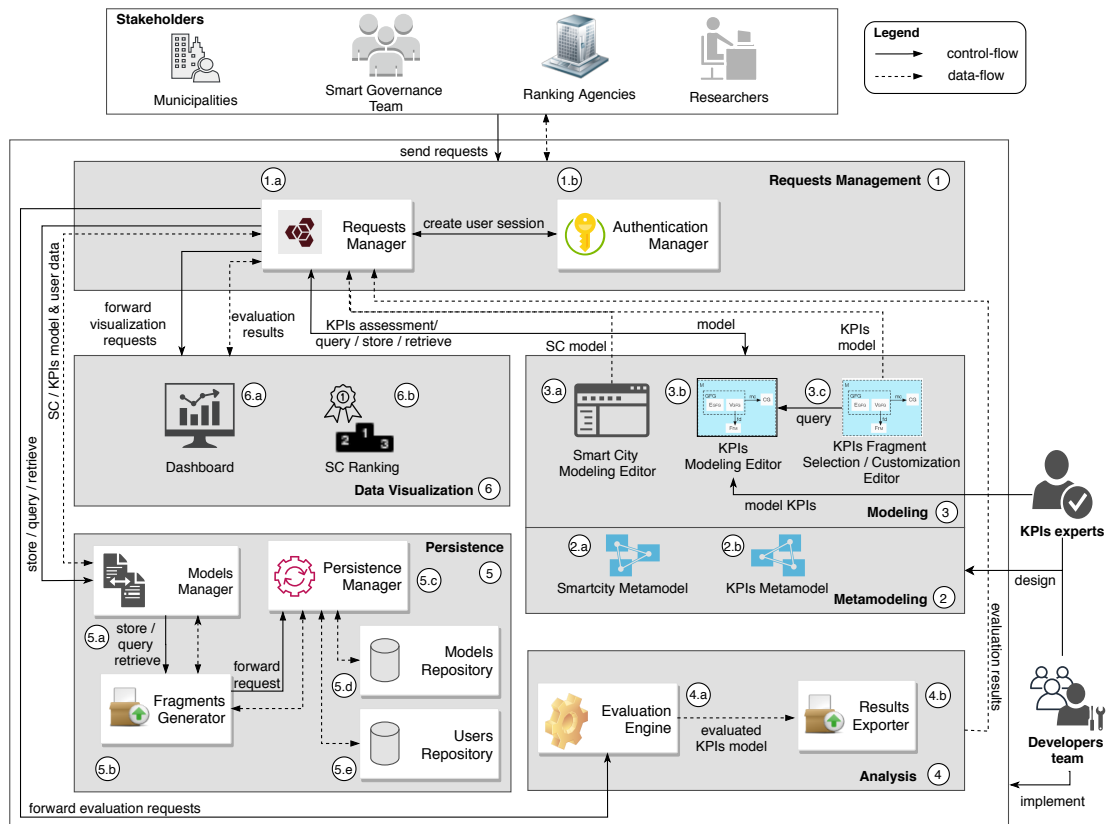


FIGURE 5.1: The Flexible Architecture for the KPIs Assessment in Smart Cities.

In Figure 5.1, the six main components, i.e., Requests Management, Modeling, Metamodeling, Analysis, Persistence, and Data Visualization, are labeled from (1) to (6), respectively. In the following we describe in detail each component and their responsibilities.

**Requests Management** component is the interface among the stakeholders and all the other components of the architecture. It is in charge of handling the users requests through its two sub-components, the Requests Manager (1.a) and the Authentication Manager (1.b). The latter is devoted to the requests concerning users registration, authentication and authorization. Every time there is the need to *create users sessions* it sends the request to the Requests Manager together with the information about the users access grants. This type of components are usually optional. In this case it depends on the deployment style. For instance, in standalone specifications of the architecture it is not necessary to deploy it. In online deployments it is good practice to have such component. Concerning the Requests Manager, it handles the requests from the users and the architecture's components and among components. The requests handled by this component are basically of three types:

1. *model* requests from the users that have the proper permissions to model the smart cities and to select/customize the KPIs to the **Modeling** component, containing the devoted modeling editors;
2. *store/query/retrieve* requests from the users that want to store/gather models concerning some previous interactions between them and the system or from users with read only permissions who want to get the available open data to the **Models Manager** in the **Persistence** component;
3. *forward visualization* requests coming from the system after the assessment or from the users that want to consult evaluation results to the **Data Visualization** component for the graphical visualization and interpretation of the evaluated KPIs.

Moreover, the *model* requests to the **Modeling** component can, in turn, give raise to further interactions: when the user is interacting with the modeling editors, she can decide to send a *query/store/retrieve* request from the **Modeling** component to the **Models Repository** via the **Requests Manager**, both to query/retrieve existing models and to store the produced models. Indeed, the **Requests Manager** is responsible for dispatching the users' KPIs assessment requests to the **Evaluation Engine** via the *forward evaluation requests* interaction.

**Metamodeling** component holds the two core metamodels of the model-based approach for the KPIs assessment, i.e., the **Smartcity Metamodel (2.a)** and the **KPIs Metamodel (2.b)**, and the corresponding tools used to model them. Thanks to this component the **Developers Team** and **KPIs Experts** manage and keep trace of the evolution of the two metamodels *design*. Notwithstanding, this is the first component to be developed and frozen in order to enable the whole KPIs assessment process, the metamodels have to be adapted accordingly to the evolving nature of both smart cities and KPIs. Consequently to evolution scenarios, coupled-evolution of the already modeled cities and/or KPIs definitions have to be performed in order to be compliant to the new metamodels [83].

**Modeling** component manages the implemented modeling editors devoted to granted users who need to model smart cities and model or select and customize KPIs. In particular, the **Smart City Modeling Editor (3.a)** provides both textual and diagrammatic concrete syntax to allow experienced and non-expert MDE developers to model a smart city. Indeed, it provides a graphical editor implemented in such a way that it can be used in a user-friendly manner also from smart governance managers, for instance. Instead, the **KPIs Modeling Editor (3.b)** has been implemented as a textual editor dedicated to KPIs Experts. They can use it, not only for the initial definition of the KPIs, but also to update them w.r.t. the evolving KPIs documentation and collection methodology provided at



European level. Moreover, besides the modeling of KPIs (*model KPIs*), smart cities managers might be interested in diverse KPIs, for instance, due to the smart cities needs and context, which are based on their stage of economic development, population growth, available services, etc. For this reason, in the modeling component we have the KPIs Fragment Selection/Customization Editor (**3.c**) that implements KPIs selection features in such a way to enable the customization of the KPIs set interesting for a city. This allows users to “*query*” the KPIs model in the KPIs Modeling Editor to select the KPIs of interest and further generate the so-called *model fragments* that can be seen as interesting object structures [84]. Indeed, the interaction of stakeholders with the modeling editors Smart City Modeling Editor and the KPIs Fragment Selection/Customization Editor produces the *Smart City model* and *KPIs model*, both conforming to the domain models in the Metamodeling component.

**Analysis** component manages the proper automatic KPIs assessment over smart cities. Here, we find the Evaluation Engine (**4.a**) in charge of the interpretation and calculation of the modeled KPIs for one or more candidate smart cities. It receives KPIs evaluation requests forwarded by the Requests Manager (*forward evaluation requests*) together with the *SC model* and *KPIs model* that, in turn, the Requests Manager receives from the Smart City Modeling Editor and the KPIs Fragment Selection/Customization Editor. As output, after executing the evaluation, it produces the *evaluated KPIs model*. This model is then sent to the Results Exporter (**4.b**), that exports the results in different formats (e.g., .csv, .xml or JSON files), w.r.t. the stakeholders’ needs (e.g., graphical visualization, textual interpretation, further elaborations) and the tool which might be used for their visualization and interpretation. Moreover, the *evaluation results* are also sent to the Requests Manager which forwards them to the user who submitted the KPIs evaluation request and/or to the Data Visualization component.

**Persistence** component takes care of the persistence of all the artifacts involved in the process of KPIs assessment, together with the stakeholders related data, such as their profiles, access grants and authentication data. In particular, it contains five components. First, the Models Manager (**5.a**) is the main interface of the Persistence component. All the *store/query/retrieve* requests coming from the Requests Manager, together with the accompanying data (e.g., the *SC/KPIs models* & *user data*) pass through this component who is also in charge of sending the corresponding replies back. The Fragments Generator (**5.b**) interacts with the Models Manager when there come specific requests of modeling artifacts and/or there is the need of generating model fragments from them. Then, the request for the artifact is forwarded to the Persistence Manager (**5.c**). This component is in charge of storing, querying or retrieving data from the appropriate repository, such as Models Repository (**5.d**) and Users Repository (**5.e**) after it receives requests and relative data arrive. In the Models Repository the artifacts are stored together with

the metadata about the users holding their ownership, meanwhile, in the Users Repository are collected the information about users profiles and granting access.

**Data Visualization** component in charge of the visualization and interpretation of the *evaluated KPIs model* via **Dashboard (6.a)** and **SC Ranking (6.b)** components. The dashboard implements the graphical transformations devoted to the generation of appropriate and easy to understand charts from the KPIs assessment results. Instead, the sc ranking component handles the comparisons among several cities whose models are made available and retrieved from the **Models Repository**, by showing graphical rankings between them. Indeed, the **Data Visualization** component receives the *evaluation results* from the **Analysis** component from the **Requests Manager** via *forward visualization requests*.

It is interesting to note that all the described components and their functionalities, apart from the **Data Visualization** component, are mandatory to allow architecture specifications to accomplish the quality evaluation process. On the contrary, to implement its functionalities, a component might not always require all its sub-components. Indeed, a subset of the sub-components may implement part of a functionality or several functionalities.

The presented architecture can be specified by using a combination of various technologies and deployment patterns. In Section 5.2, we present the *standalone specification*, where the architecture is entirely implemented as a standalone platform. In Section 5.3 we present the *hybrid specification*, where part of the architectural components are deployed online and part of them reside locally on the user's machine. In Section 5.4, instead, we give hints about the *online specification* that represent a further deployment enabled by our flexible architecture, where all the components can be completely deployed online.

## 5.2 Standalone Deployment Implementation

In this section, we present the deployment pattern that we specified to develop our framework prototype. We call this deployment pattern *standalone specification* because the architecture is completely implemented as a standalone platform relying on Eclipse, since it used as a standard platform in MDE because of its pre-packaged bundles for specific development paradigms. In particular, we used as target platform the *Eclipse Modeling Framework (EMF)*<sup>1</sup> supporting the engineering of DSLs [85]. The EMF core provides a metamodeling language, called *Ecore*, used for describing domain models, and runtime support for the models including change notification, persistence support with default serialization, and reflective APIs for manipulating objects in the models. In Figure 5.2, we reported all the components of the standalone specification of our

<sup>1</sup> <https://www.eclipse.org/modeling/emf/>

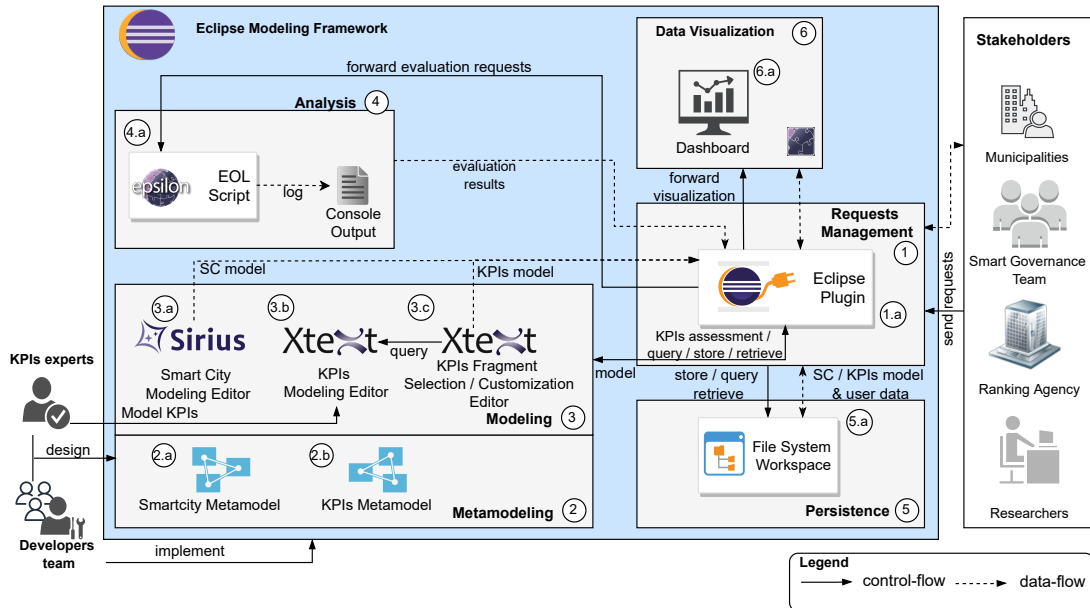


FIGURE 5.2: Standalone specification of the flexible architecture in Figure 5.1.

architecture. Starting from the Request component (1), the Requests Manager (1.a) is implemented as an Eclipse plugin organized with the extension point mechanism. It can be activated by file saving operations or even directly by menu entries in the editors. As regard the Metamodeling component (2), it has been defined on top of the EMF bundle, and collects the two main domain models ((2.a) and (2.b)). By means of the model generation feature, the Ecore metamodels produce the Java code for managing base editors and model manipulation operations.

### 5.2.1 Modeling Component

On top of the metamodeling layer, two editors have been implemented ((3.a) and (3.b))<sup>2</sup> supporting the definition of the involved models by providing operations allowing the editing of models with the possibility of filling them with their inherent model elements. The DSLs used in the editors implementation can be *graphical* or *textual* [86]. The former provide an intuitive GUI for modelers, the latter a support tool to define models as textual specification, which is better transposed by developers since textual DSLs resemble code development (see Figure 5.3). To allow both experienced and non-expert users (e.g., smart cities stakeholders) to use the editors for editing or creating new models, the modeling editors in the component (3) have been implemented with two different technologies. The Smart City Modeling Editor (3.a), besides providing a textual editor, has been also developed as a graphical modeling workbench for modeling smart cities to support non-expert smart cities stakeholders. The graphical editor relies on Sirius<sup>3</sup>, which

<sup>2</sup> <https://github.com/gssi/MoSC2020> <sup>3</sup> <https://www.obeodesigner.com/en/product/sirius>

```

1 smartCities{
2   city "L'Aquila"{
3     stakeholders {
4       organization "CityCouncil" organizationName "Municipality",
5       organization "TIM" organizationName "Internet and Telecommunication",
6       openData "BreezoMeter" [url : "https://breezometer.com/"],
7       openData "PisteCiclabili.com" [url : "https://www.piste-ciclabili.com/"],
8       openData "AtlanteStatisticoDeiComuni" [url : "http://asc.istat.it/asc_BL/"]
9     }
10    data{
11     dataPackage AirMontoring{
12       real "PM2.5" = 11.19("ng/m3") [BreezoMeter],
13       real "PM10" = 11.38("ng/m3") [BreezoMeter],
14       real "NO2" = 5.57("ng/m3") [BreezoMeter],
15       real "O3" = 18.28("ng/m3") [BreezoMeter],
16       real "SO2" = 0.17("ng/m3") [BreezoMeter]
17     },
18     dataPackage NetworkCoverage{
19       real "3GCoverage" = 300.0(km2) [TIM],
20       real "4GCoverage" = 300.0(km2) [TIM],
21       real "5GCoverage" = 100.0(km2) [TIM]
22     },
23     dataPackage CityStatistics{
24       real "CityExt" = 480.0(km2) [CityCouncil],
25       real "CityPop" = 69605.0(inhabitants) [CityCouncil]
26     },
27     dataPackage BikesPaths{
28       real "BykePathLength" = 86.0(km) ["PisteCiclabili.com"]
29     },
30     dataPackage GreenAreas{
31       real "TotalGreenArea" = 48.7235(hectares) [AtlanteStatisticoDeiComuni]
32     }
33   }
34 }
35 }

```

FIGURE 5.3: Textual representation of the model for the city of L'Aquila.

is an Eclipse project supporting the development of graphical modeling workbenches by leveraging the Eclipse Modeling technologies, including EMF and the Graphical Modeling Framework (GMF). Sirius allows the development of workbenches together with a set of Eclipse editors (diagrams, tables, and trees), to support the users in the creation, editing and visualization of graphical representations of EMF models. The editors are implemented on top of a metamodel, from which all the editing and navigation tools depend. Indeed, the graphical editor provides a palette containing tools allowing users to create new model elements (see Figure 5.4) coming from the metamodel. In particular, it supports the creation of *Entity* elements, *Attribute* within entities, and relationships between entities (*Relation*). In this case, the implemented SC modeling editor has a dedicated palette with specific tools w.r.t. the concepts defined in the smart cities metamodel. From the *Property view* envisaged from the Eclipse IDE, it is possible to give a value to the attributes of each modeled element. Relying on the metamodel, the graphical editor prevents problems of inconsistency because it does not allow the creation of elements that are not those expected from the language metamodel.

Since the use of the KPIs Modeling Editor (3.b) is envisaged for modelers supporting KPIs experts to specify KPIs and relative calculation formulas, it has been implemented with a textual concrete syntax by means of *Xtext*<sup>4</sup>, an Eclipse project for developing DSLs [87]. Thanks to the KPIs Modeling Editor, users may declare how KPIs are calculated in detail in an expressive and agile way. Moreover, the use of a DSL for KPIs definition enables the reuse and share of it. Using Xtext allows the implementation of a textual modeling

<sup>4</sup> <https://www.eclipse.org/Xtext/>

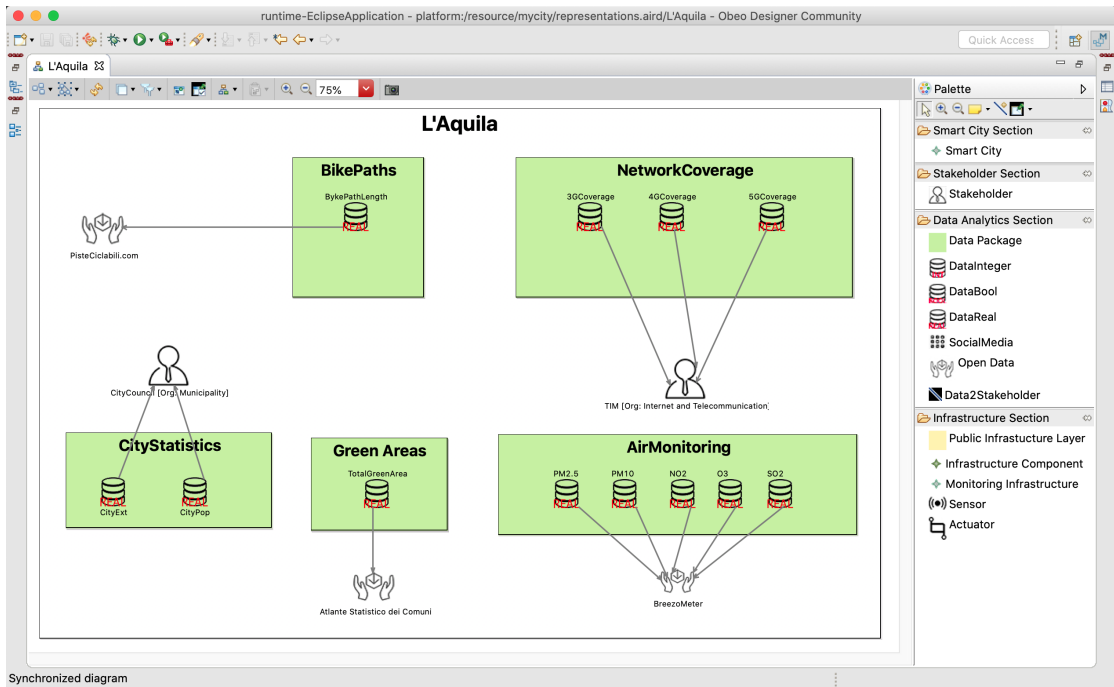


FIGURE 5.4: Graphical representation of the model for the city of L'Aquila.

editor with useful modeling feature such as syntax highlighting, code completion (see Figure 5.5), and outlines.

In addition to the basic checking features provided by the analysis tools coming from the modeling editors that monitor the structural correctness of the models w.r.t. the provided metamodels, it is possible to extend the system by specifying additional checks that modelers might want to add for the particular models at hand. In the development of our modeling editors we exploited the Xtend<sup>5</sup> programming language in the textual editor and the Aceleo Query Language (AQL)<sup>6</sup> in the graphical editor for the specification of further custom validation rules. Moreover, for the specification and evaluation of constraints on models of arbitrary metamodels and modeling technologies we used the Epsilon suite, and specifically Epsilon Object Language (EOL)<sup>7</sup> and the Epsilon Validation Language (EVL)<sup>8</sup>. With EVL it is also possible to manage interdependencies between constraints (e.g., if constraint A fails, the constraint B cannot be evaluated), to customize error messages to be displayed to the user, and to specify fixes (in EOL) that can repair inconsistencies (see an example in Figure 5.6).

<sup>5</sup> <https://www.eclipse.org/xtend/>

<sup>6</sup> <https://www.eclipse.org/acceleo/documentation/>

<sup>7</sup> <https://www.eclipse.org/epsilon/doc/eol/>

<sup>8</sup> <https://www.eclipse.org/epsilon/doc/evl/>

```

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

```

FIGURE 5.5: Auto-completion feature.

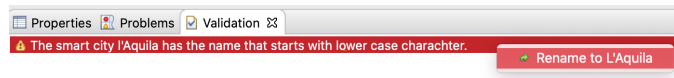


FIGURE 5.6: Syntax error at modeling time.

## 5.2.2 Analysis Component

After the definition of the SC model and KPIs model, the Analysis component (4) is invoked to trigger the KPIs evaluation phase. This process is managed by the Requests Manager (1.a) that is implemented as an Eclipse plugin organized with the extension point mechanism. It can be activated by file saving operations or even directly by menu entries in the editors. By selecting a SC model and a KPIs model (with specific extensions), a menu entry is enabled and the *EOL script* implementing the Evaluation Engine (4.a) is triggered. Thanks to EOL and the Epsilon framework [88] it is possible to create, query and modify models built on top of EMF. Basically, the EOL script is a file in the workspace of the project that will be invoked by the plugin which can be activated by selecting the right models, based on the file extension. In particular, the evaluation engine is executed via a Java main method, whose code is reported in Listing 5.1. In the main method of this class, the input models, i.e., Smart City Model and KPIs Definition model are loaded and passed as parameters (lines 3–6). At line 11, the EOL script devoted to KPIs calculations is executed.

```

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 @Override
10 public String getSource() throws Exception {
11     return "epsilon/evaluation-engine.eol";
12 }

```

LISTING 5.1: Snippet of the Java Evaluation Engine.

We report in Listing 5.2 an excerpt of the invoked EOL script. This 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). In the console, it prints the description of the KPIs (line 6) and the result of the evaluation (line 7).

```

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

```

LISTING 5.2: Snippet of the Main EOL Script.

All the KPIs computations are in charge of the fundamental operation `get()` at line 7 invoked on every KPI. We defined this operation in another EOL script (i.e., `kpi-providers.eol`), imported at line 1 and reported in Listing 5.3. Here, we defined the operation `get()` for each `ValueType`. At lines 5–10, the operation for the `SingleValue` is reported, where the calculated value is stored in the `actualizedValue` of the KPIs model. Before performing the calculation the engine searches for the needed parameters through name-based references between the KPIs definition model and the smart city model (lines 6–7). If the parameter indicated in the KPIs model is not defined in the smart city one, the operation will not be performed. As concerns `AggregatedValues`, for these types the result is recursively calculated until reaching the nested single values. The operations defined in the KPIs model (e.g., SUM, AVG) are translated into the language implementing the engine at lines 30–40. In case the rise of a new operation happens, it is only needed to add another case in this switch statement.

```

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.collect(dat|dat.data).flatten()
7     .select(d|d.name==self.parameter.name).value.first;
8   return city.data.collect(dat|dat.data).flatten()
9     .select(d|d.name==self.parameter.name).value.first;
10 }
11
12 operation kpi!StaticRealValue get(city: smartcity!SmartCity): Any{
13   return self.staticvalue;
14 }
15
16 operation kpi!AggregatedValue get(city: smartcity!SmartCity): Any{
17   self.actualizedvalue=self.values.get(city, self.oper);
18   return self.values.get(city, self.oper);
19 }
20
21 operation kpi!AggregatedRangedValue get(city: smartcity!SmartCity): Any{

```

```

22     self.actualizedvalue=self.ranges.select(r|r.min <= self.values.get(city ,
self.oper).first and self.values.get(city , self.oper).first <=
r.max).rangeName.first ;
23     return self.ranges.select(r|r.min <= self.values.get(city , self.oper).first
and self.values.get(city , self.oper).first <= r.max).rangeName.first ;
24 }
25
26 operation OrderedSet<kpi!ValueType> get(city : smartcity!SmartCity , oper :
kpi!EEnumLiteral): Any{
27     return self.collect(s|s.get(city)).getop(oper) ;
28 }
29
30 operation Any getop(op: kpi!EEnumLiteral): Any{
31 switch (op) {
32     case kpi!Operation#GET: return self ;
33     case kpi!Operation#SUM: return self.sum() ;
34     case kpi!Operation#AVG: return self.sum()/self.size() ;
35     case kpi!Operation#MIN: return self.min() ;
36     case kpi!Operation#MAX: return self.max() ;
37     case kpi!Operation#DIV: return self.get(0)/self.get(1) ;
38     case kpi!Operation#MULT: return self.get(0)*self.get(1) ;
39     case kpi!Operation#CEN: if(self.get(0) >= 100000) return self.get(0) /
100000; else return 1;
40     default : "No operation provided ".println() ; }
41 }

```

LISTING 5.3: Snippet of the EOL KPI provider.

### 5.2.3 Data Visualization Component

The component devoted to Data Visualization **(6)** has been built on top of Picto [89], an Eclipse view for visualising models via model-to-text (M2T) transformation to SVG/HTML. Picto supports rule-based model-to-text transformations with the Epsilon Generation Language (EGL) [90], a M2T transformation language, used to transform models into hierarchically-organised read-only graphical views. The generated views belong to a range of formats (e.g., SVG, HTML, PlantUML, Graphviz) that are rendered in an embedded Web browser. To implement the generation of our graphical Dashboard **(6a)** we implemented three EGL templates, i.e., KPIModel2Picto, Dimensions2Picto, Categories2Picto. These three templates exploit the library *Chart.js*<sup>9</sup> to generate HTML/-Javascript code to render the charts representing indicators results. In Listing 5.4 we show a snippet of the code generator for creating a gauge chart. This type of chart is generated for each KPI whose return value is a real and has a *target value* defined (as specified in the KPIs metamodel in Figure 4.3). At lines **12-19** 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 are set. Lines **4-10** initializes the gauge graphs, with

<sup>9</sup> <https://www.chartjs.org/>



the options defined in lines **12-19**. We can see at lines **21-22** the code generation for other types of KPIs.

```

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    } else if (kpi.value.valuetype.isTypeOf(AggregatedRangedValue)) { ... }
22    } else if (kpi.value.valuetype.isTypeOf(BoolValue)) { ... }
23    i++;
24  }
25 [%]

```

LISTING 5.4: Snippet of the EGL script for generating gauge chart.

### 5.3 Hybrid Deployment Implementation

The presented architecture in Figure 5.1 can be specified by using a combination of various technologies and deployment patterns. In this Section, we discuss an alternative deployment pattern that we used to develop a prototype of the extended KPIs assessment framework enabling continuous monitoring. In particular, we performed a refactoring of the standalone deployment that we define *hybrid* because part of the architectural components are deployed online and part of them reside locally on the user's machine. In this type of deployment we have an Internet layer that is in between the remote and the local components.

Indeed, some of the components are the same as in the standalone specification, such as components **(2)** and **(3)**. The Requests Manager **(1.a)** presents both a local and a remote instances. The local one is in charge of activating the editors and triggering the

remote KPIs evaluation by sending a request, via the internet, to the remote one that will forward the request and the received models to the Evaluation Engine (4.a). Even the Models Repository (5.d) that are managed by the Persistence Manager (5.c) remain the same as the standalone specification together with the Evaluation Engine (4.a) and Data Visualization (6) components. We will discuss the proposed hybrid deployments in details in the next sub-sections.

### 5.3.1 Weaving Open Services with Runtime Models for Continuous Smart Cities KPIs Assessment

In Figure 5.7 we reported the hybrid deployment pattern that we used to implement the KPIs assessment framework enabling continuous monitoring of smart cities.

In particular, we refactored and extended the standalone architecture by adding a *message-oriented middleware* connected to heterogeneous data sources and partially reusing the already implemented standalone specification. In this way, we evolved the framework in an automatic and service-oriented perspective in order to enable continuous monitoring of KPIs data sources and runtime update of models in the KPIs assessment for SCs.

In Figure 5.7 we reported the service-oriented architecture for the continuous KPIs assessment of SCs where the arrows shape the data-flow among components. We depicted in white and blurred grey the components coming from the standalone deployment, where the blurred grey ones required modifications to integrate the new architectural components. Meanwhile, the newly integrated components are colored in grey.

For a better understanding, in Figure 5.8 we depict the KPIs assessment process with a UML sequence diagram (SD). The diagram shows the high-level behaviour of architectural components, and the interactions among them. The highlighted part shows the newly integrated components and their tasks, while in grey, are reported the lifelines of the newly implemented components managing the continuous monitoring and model injection features. Lastly, the circled numbers ①–⑰ labeling the process steps in the SD are used in the following text to map those steps with the architectural components involved in their execution.

In the *front-end*, we have the KPIs Modeling Editor (implemented with Xtext [87]<sup>10</sup>) devoted to the selection and definition of the relevant KPIs for the SC under evaluation, through custom textual DSLs. The Smart City Modeling Editor (implemented with Sirius [91]<sup>11</sup>), instead, helps users to model the SC under evaluation through the exploitation of graphical functionalities. Lastly, the graphical Dashboard, obtained through

<sup>10</sup> <https://www.eclipse.org/Xtext/> <sup>11</sup> <https://www.eclipse.org/sirius/>

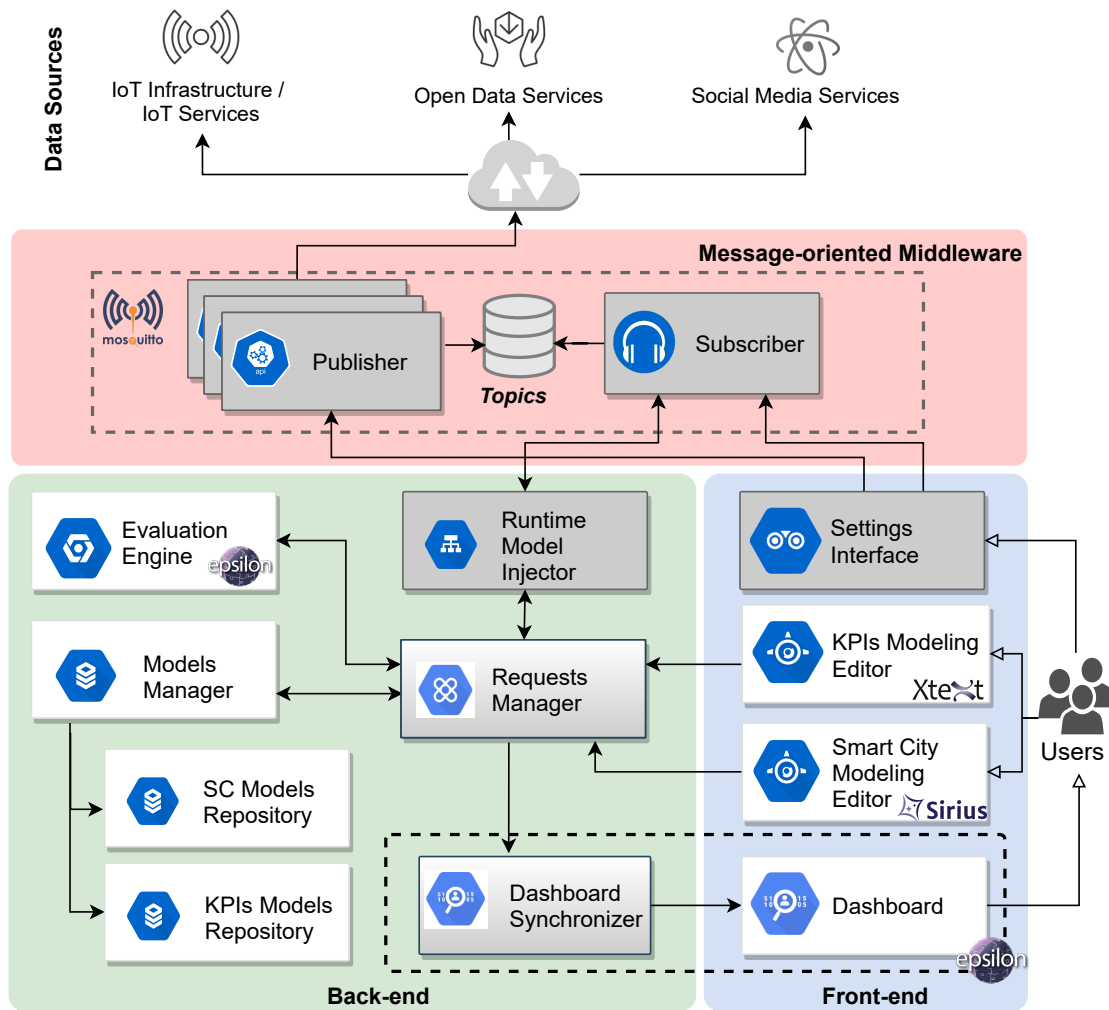


FIGURE 5.7: Hybrid specification of the flexible architecture in Figure 5.1.

model to code transformations (and visualized with Picto<sup>12</sup>) ⑦ allows the interpretation of the KPIs assessment results. The Dashboard Synchronizer converts the KPIs model instantiated after the assessment in an .html file ⑥, which is in sync with source files of the graphical Dashboard (modeled with Picto<sup>13</sup>) ⑦. This component has its own listener that every time the model changes the html file is reloaded, updating the views.

In the *back-end*, the Requests Manager is in charge of managing: (i) KPIs assessment requests ① to the Evaluation Engine (modeled with the EOL [88]<sup>14</sup>) that is responsible for performing the SC evaluation ③; (ii) visualization requests from the Evaluation Engine to the Dashboard component ⑤. In particular, the Dashboard Synchronizer converts the KPIs model instantiated after the assessment in an .html file ⑥, which is in sync with source files of the Dashboard. The synchronizer has its own listener that every time the model changes the html file is reloaded, updating the views. The Requests Manager also handles requests to the Models Manager to gather ② or store ④ the models needed in

<sup>12</sup> <https://www.eclipse.org/epsilon/doc/picto/>

<sup>13</sup> <https://www.eclipse.org/epsilon/doc/picto/>

<sup>14</sup> <https://www.eclipse.org/epsilon/>

the KPIs assessment process. The models manager handles the persistence of models in the SC Models Repository and the KPIs Models Repository.

We now describe the new components and the *message-oriented middleware* for the continuous monitoring. The Publishers and the Subscriber implement the classic publish/-subscribe communication pattern based on Topics (publish and subscribe to messages). The type of topics are the types of parameters the SC can handle in the evaluation, e.g., pollutants, weather, and so on. Specifically, the Publisher components can be multiple, considering the multiple data sources. They send calls to data sources, e.g., open services, to gather the input parameters needed for the KPIs calculations (12). Each

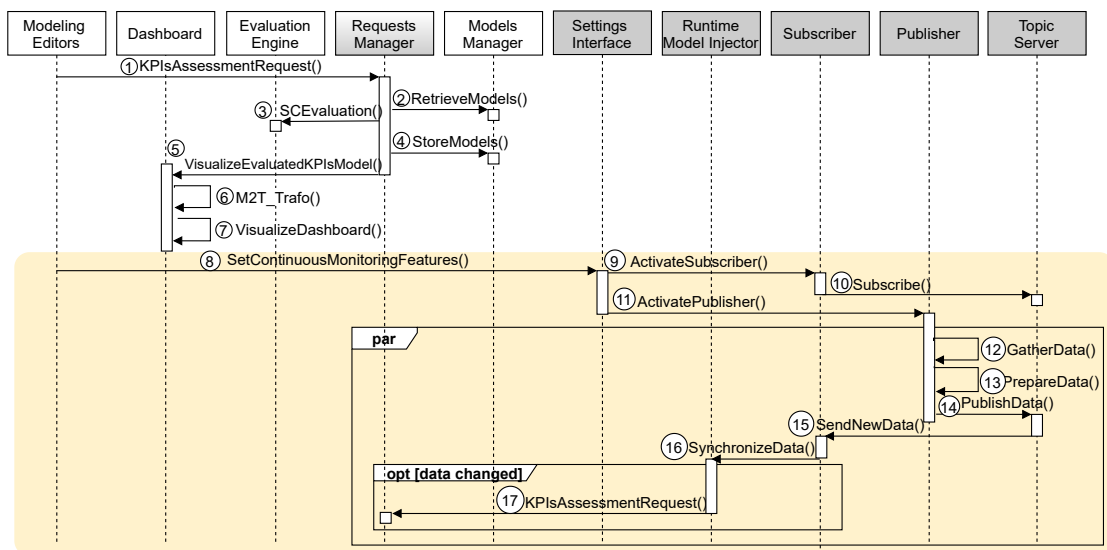


FIGURE 5.8: Sequence Diagram of the KPIs assessment process.

publisher cleans, prepares and aggregates the data (13), before publishing changed parameters on the assigned topic (14). These components are interfaces implemented, in turn, by specific Java classes. The topics are published as MQTT [92] messages with a specific structure, as in (5.1), allowing multi-city evaluation.

$$lat/\{latitude\}/long/\{longitude\}/parameter \quad (5.1)$$

Latitude and longitude are the GPS coordinates of the smart city under evaluation. This way we can provide multiple publishers as types of data gatherer for different cities, by matching the coordinates specified in the SC model. Then, the subscriber is able to distinguish which data intercept, by using the latitude and longitude of the smart city under evaluation. The Subscriber is devoted to the synchronization of the models with the data sources. It receives changed parameters through the subscription to the corresponding topic (10) and actions can be triggered, as specified in the following. The Runtime Model Injector is invoked by the Subscriber, when it receives new data from the

topics (15), and devoted to the retrieving of the SC models that need to be synchronised. It checks if the input parameters in the models are in line with the ones received by the Subscriber (16) by querying the model. It is also in charge of SC models *update*, that is the operation of filling the models with the received up-to-date data, through EOL [88] queries. The EOL scripts, basically, first query the SC model to get the needed parameters. Then, cross the parameters needed in the evaluation with the ones retrieved by the subscriber. For each retrieved parameter, the model injector checks if the corresponding value is still in line with the value of the subscribed topic. If this does not apply, then it injects the new parameter retrieved by the subscriber (an example of log is reported in Figure 5.10). Furthermore, the Runtime Model Injector interacts with the Requests Manager to trigger the KPIs assessment process (17) due to changed input parameters, requesting the last saved SC models (2). The overall monitoring process is enabled by the user that is responsible for setting up the continuous monitoring features, only for the dynamic parameters with the *runtime* attribute set to true (8) in the SC model, through the Settings Interface. It is used for defining the *topics* to which the Subscriber has to subscribe (9), and configure the APIs and the frequency with which they must be called in the Publishers (11). This setting is provided through a name-based convention, so if the parameters declared in the SC model are included in the topics, then the Runtime Model Injector will consider them in the process. The format required to publish KPIs parameters on the assigned topic has also to be defined.

### 5.3.2 Runtime Models Update by Continuous Monitoring

In this section, we demonstrate the evolved approach through a running example whose overview is shown in Figure 5.9. We applied the extended evaluation framework over a real smart city, i.e., L'Aquila. However, the framework can evaluate multiple SCs at a time. As *dynamic* KPIs we consider: *Humidity*, i.e., the percentage of the detected humidity level, *Air Pollution* (see formula (6.2)), *PM2.5*, *NO<sup>2</sup>*, measuring the *PM2.5* and *NO<sup>2</sup>* pollutants levels in the air, *Urban Heat Island (UHI)*, i.e., the difference in air temperature between the city and its surroundings (provided by two different sources), and *Green Area (GA)* as a less dynamic KPI measuring the green area hectares per 100.000 inhabitants. Four Publishers are considered as data sources for the input parameters of these KPIs. *BreezoMeter*<sup>15</sup> is a website offering *open API* exposing environmental data and insights to third-party applications. It can be queried via a specific *url* taking the latitude and longitude of a city as parameters, and it replies with JSON data providing values for different pollutants and other interesting info. *ISTAT*<sup>16</sup> is the national statistics institute providing *Web services* exposing statistics about, e.g., the census of

<sup>15</sup> <https://www.breezometer.com/> <sup>16</sup> <http://dati.istat.it>

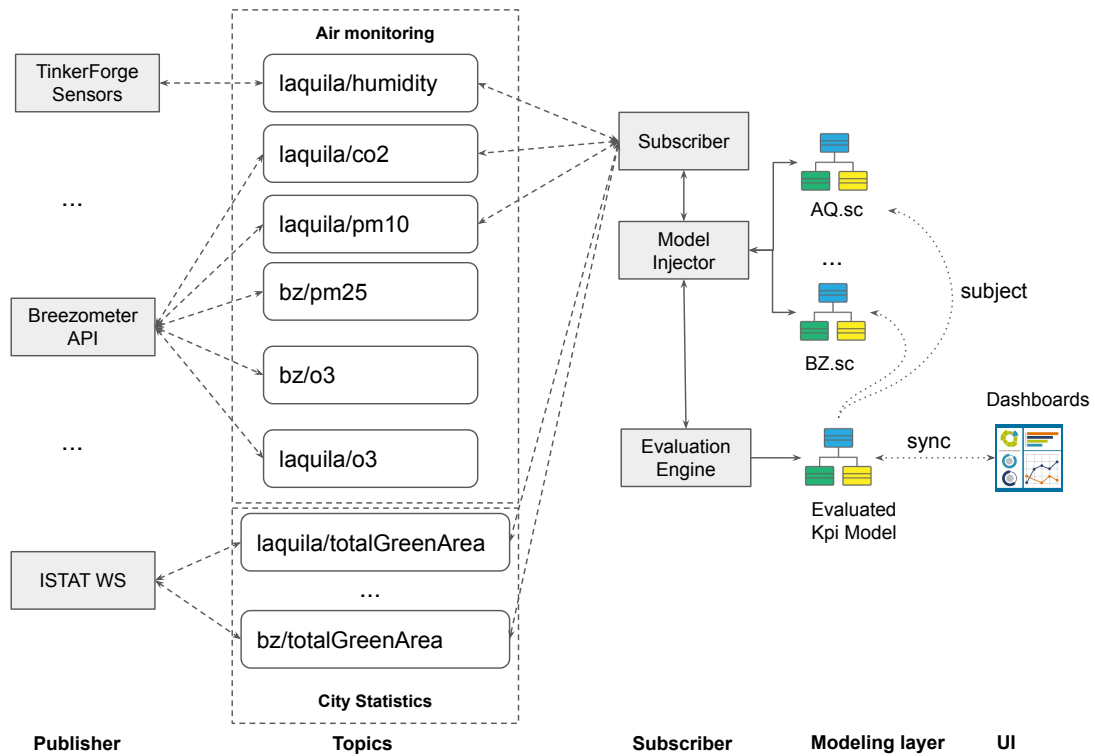


FIGURE 5.9: Publish-Subscribe pattern in our example

population, economic censuses. *OpenWeather* map<sup>17</sup> provides information about weather forecasts, air temperature, etc. for any coordinates via an open API. Lastly, an IoT infrastructure based on *Tinkerforge*<sup>18</sup> sensors that publishes data from a Tinkerforge Air Quality Bricklet<sup>19</sup>. Tinkerforge is a system of building blocks, based on pluggable modules and APIs, available for many programming languages. Using this board we have assembled a smart environment sensor in our department measuring multiple air quality parameters, e.g., humidity and temperature<sup>20</sup>.

The first three publishers collect different parameters values (e.g., PM2.5, totalGreenArea) through APIs calls and create topics for all of them. Anytime these values change, the publishers will send a new MQTT message. The Tinkerforge publisher, instead, continuously monitors specific parameters, i.e., humidity and temperature, for which a new message is published when the measured values change.

A snippet of the Java Tinkerforge publisher running on our board is reported in Listing 5.5. First, the connection with the sensor is established (line 2), then the smart city model is loaded (line 5) to get the *lat* and *long* (line 7) used to format the MQTT message. From line 10 a listener to monitor humidity changes is declared and the callback is set at line 35. The listener checks the humidity every 10 seconds and in case of

<sup>17</sup> <https://openweathermap.org>    <sup>18</sup> <https://www.tinkerforge.com/en/>    <sup>19</sup> <https://bit.ly/3xngllU>

<sup>20</sup> A picture of the assembled board is available at <https://bit.ly/3Y7HhVw>

changes, a new MQTT message is created and published (lines **21** and **29**). The message is both published and read in JSON. It is worth noting that the message string of the topic contains *lat* and *long* of the smart city. This is the distinguishing factor used by the subscriber to understand which *SC Model* must be updated. Similarly, the other publishers query the open APIs URLs to get the parameters values they relate to.

```

1 public void run() {
2     IPConnection ipcon = new IPConnection(); // Create IP connection
3     BrickletAirQuality aq = new BrickletAirQuality(UID, ipcon);
4     ...
5     SCModelLoader scmodel = new SCModelLoader(city);
6     try {
7         EolMap latlong = scmodel.getSmartCityLatLong();
8         ipcon.connect(HOST, PORT);
9
10        aq.addHumidityListener(new BrickletAirQuality.HumidityListener() {
11
12            @Override
13            public void humidity(int humidity) {
14                double humidityvalue = humidity/100.0;
15
16                try {
17                    MqttClient client = new MqttClient("tcp://localhost:1883",
MqttClient.generateClientId());
18                    MqttConnectOptions option = new MqttConnectOptions();
19                    option.setKeepAliveInterval(30);
20                    client.connect(option);
21                    MqttMessage message = new MqttMessage();
22
23                    Double value = humidityvalue;
24                    String messageJson="{\"Humidity\":\"+value+"\"}";
25                    message.setPayload(messageJson.getBytes());
26                    String topic = "lat/"+latlong.get("latitude").toString()+
27                        "/long/"+latlong.get("longitude").toString()+"/Humidity";
28
29                    client.publish(topic.toLowerCase(), message);
30
31                } catch (TinkerforgeException | MqttException e) { ... }
32            }
33        });
34        // Set period for all values callback to 1s (1000ms)
35        aq.setHumidityCallbackConfiguration(10000,true,'x',0,0);
36        ...
37    }
38    ipcon.disconnect();
39}

```

LISTING 5.5: Snippet of the Java Tinkerforge Publisher.

In the screen shown in Figure 5.10, the runtime logs of the BreezoMeter publisher and the subscriber are shown. The defined *KPIs Model* (top-left side) contains the KPIs we intend to evaluate on the subject city that are constantly synchronized with the dashboards

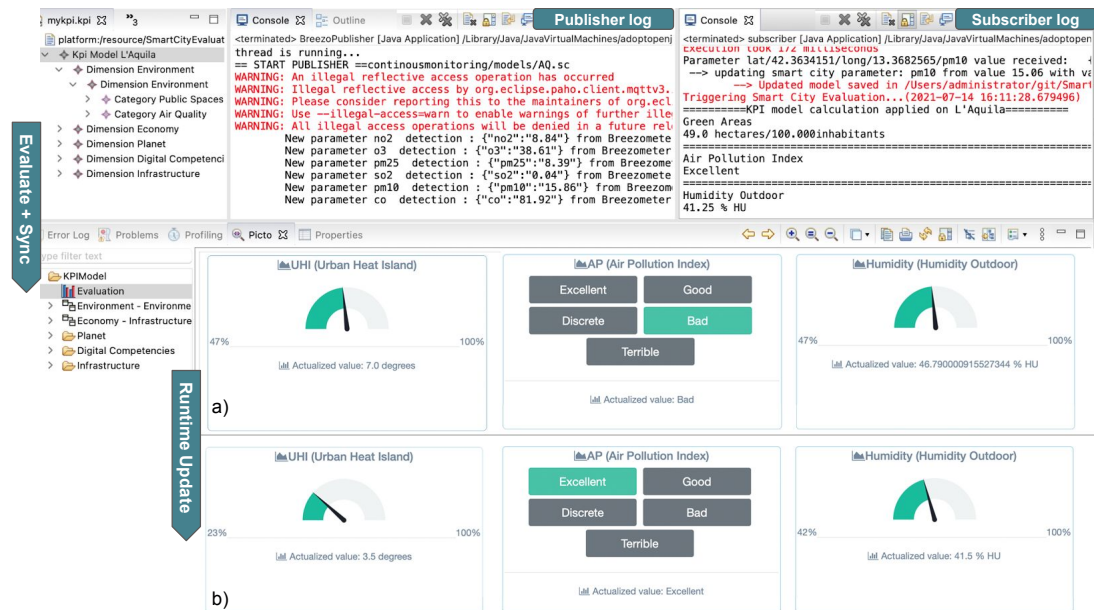


FIGURE 5.10: Our tool in action for the smart city L'Aquila.

(bottom side). Multiple publishers can be run at the same time, since every publisher is executed by a dedicated thread. The publisher log shows that new parameters for pollutants have been detected. As long as the publisher sends new retrieved values (it publishes on topics), the subscriber (subscribed to all topics) will receive the message, as shown by the subscriber runtime log. This message will be forwarded to the *Runtime Model Injector* that will inspect the retrieved values by comparing them with the ones in the *SC Model* under evaluation. In case they are misaligned, the model will be updated. Every time this happens, the evaluation phase is triggered (see the triggered evaluation in the subscriber log). The updated *Evaluated KPIs Model* after the evaluation is automatically synchronized with the dashboards, as can be seen from the bottom side of the screen, where gauges and other diagrams report the selected KPIs indicators evolution, at a given instant of time. For lack of space we report only a partial view of the entire dashboard. We can observe as the dashboard evolved from **a)** to **b)** highlighting that the *UHI* KPI decreased from 47% to 23%, the *AP* KPI improved from *Bad* to *Excellent*, and the *Humidity* decreased from 47% to 42%. These updates are instantaneous and transparent to the user that has to simply run the framework. A demo video showing the described running example is available at <https://youtu.be/2pK-PzOLvv4>.

## 5.4 Online Deployment Specification

We now describe an additional deployment pattern of our architecture, namely *online*. This deployment is currently under development, although it relies on a partial reuse



of the implemented standalone specification, thus we focus on the differences w.r.t. it. Figure 5.11 depicts how we plan to realize it and how the architectural components will be implemented. Being everything online, the main instrument to use the platform will

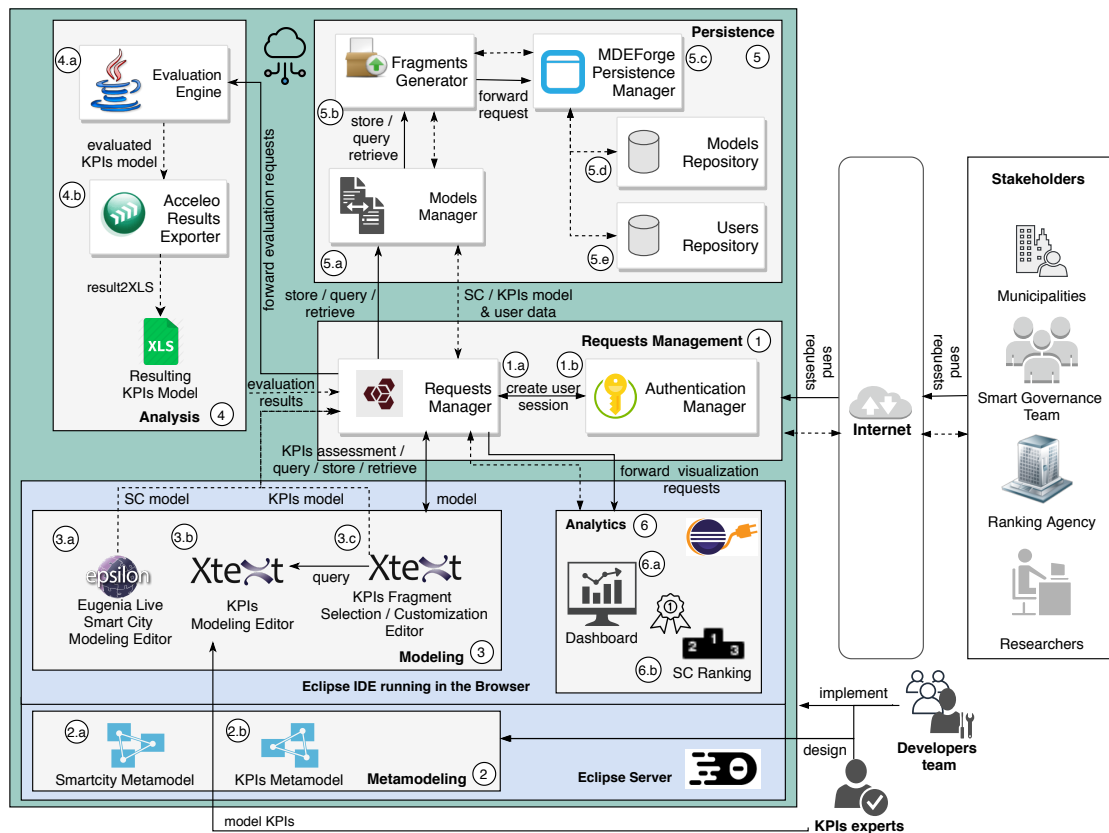


FIGURE 5.11: Online specification of our architecture

be the browser, that will provide access to a web application including the different components. More specifically, the Modeling component (3) and their editors are distributed online, thus executable through the web browser. For their implementation, the two candidate technologies allowing for running modeling editors in web browsers are *Eugenia Live* [93] and Xtext. From one side, Eugenia Live further supports on-the-fly model editing, that is why it represents the best candidate for the Smart City Modeling Editor (3.a). From the other side, since version 2.9, Xtext offers an interface for the integration of text editors in third-party web applications. For this reason, it will be again used for the KPIs Modeling Editor (3.b). The text editors will be implemented in JavaScript, and the language-related services will be implemented through HTTP requests to a server-side component. The modeling artefacts will not be stored locally but every time the modeler starts a new modeling phase the Persistence Manager will store the models in the proper repository, always via the Requests Manager. If the modeler has to resume a modeling work, the Authentication Manager is in charge of authenticate the user and retrieve the models with respect to the user's rights. The Persistence component (5) will be delegated

to MDEForge, and the **Analysis** component (4) will rely on the Java and Acceleo technologies. The evaluation results will be exposed by a dedicated API, that will feed the **Data Visualization** (6) view in the web browser. The advantage of an online environment is that it supports one of the main problems that slows down the path of MDE towards a standard: the reluctance in installing different tools, most of them academic, with all the related issues linked to safety and reliability. These aspects cannot be neglected in an industrial scenario. Moreover, collaborative repositories have been extensively proposed and investigated in MDE [94], highlighting multiple challenges. Among these, visibility of the repositories stored artifacts seem to be one of the hot topic in industry. Multiple resolutions have been proposed to tone down the reluctance in employing these tools, and extended visibility management seems to be needed [95] to assure that only artifacts intentionally shared will be visible to the community and not the models including intellectual property rights.

Besides the definition of different deployment patterns, we sketched a list of alternative technologies to use in the implementation of the various architectural components, to further support our claim that the proposed flexible architecture enables the discussed deployment alternatives. This type of analysis proves the *technology-independent nature* of the architectural components, enabling the use of diverse technologies, also w.r.t. the chosen deployment style. Indeed, in Table 5.1 we report some of the valid technological alternatives to implement the different architectural components w.r.t. the chosen deployment style.

TABLE 5.1: Architecture flexibility in terms of required components and technologies.

Components	Mandatory	Candidate Technologies
Requests Manager	✓	Eclipse plugin, Java
Authentication Manager	≈	Java
Smartcity Metamodel	✓	Ecore, Kermet[96], UML
KPIs Metamodel	✓	Ecore, Kermet, UML
Smart City Modeling Editor	✓	Sirius, Eugenia Live, Eugenia
KPIs Modeling Editor	✓	Xtext, EMFText[97]
KPIs Fragment Editor	✓	Xtext, OCL <sup>10</sup> , EMF-Fragments <sup>11</sup>
Evaluation Engine	✓	Java, EOL Script, ATL <sup>12</sup> , ETL <sup>13</sup>
Results Exporter	≈	Acceleo, JET <sup>14</sup> , EGL <sup>15</sup> , Xtend <sup>22</sup>
Models Manager	≈	Java, MDEForge
Fragments Generator	≈	EMF + Java + OCL
Persistence Manager	✓	MDEForge, Neo4EMF <sup>16</sup> , Relational DB, EMFJson <sup>17</sup>
Models Repository	✓	MDEForge, EMFStore <sup>18</sup>
Users Repository	≈	NoSql[98], Mysql <sup>19</sup> , MSSql <sup>20</sup>
Dashboard	≈	Spring <sup>21</sup> , other J2EE or JS-based frameworks
SC Ranking	≈	Spring, other J2EE or JS-based frameworks

In the table we indicated the level of binding of each architectural component. In particular, in the column **Mandatory**, the  $\checkmark$  symbol denotes that the corresponding component is mandatory, independently of the used deployment style, while  $\approx$  denotes that the component may be missing, e.g., because its offered functionality may be omitted without compromising the functioning of the approach (e.g., the **Data Visualization**) or because it is not required for the used deployment (e.g., the **Authentication Manager** in a standalone instance).

## 5.5 Discussion

In this Chapter, we show the architecture on top of which the *MIKADO* framework has been developed. In particular, we proposed different alternative *deployment patterns* that can be used to distribute systems implementing the proposed architecture, giving evidence of the *flexibility* of our architecture. Specifically, we described the standalone deployment pattern, by further giving details about the implementation of every component. Furthermore, we reported a hybrid deployment implementation describing a service-oriented architecture that enables smart cities continuous monitoring and runtime models update. In this perspective, the implemented KPIs assessment approach can be seen as a models@runtime evaluation. This model-driven emerging paradigm [49, 50] in which models are causally connected to the problem space may be very useful in the smart cities KPIs assessment. In particular, when changes are required in the KPIs specifications, the changes must be reflected on the results of the evaluation, thus, on the running system. In the case of *MIKADO*, the KPIs assessment approach may benefit from exploiting runtime models, since the KPIs definitions express the semantics of the generated dashboards. Being able to update KPIs models at every smart city assessment change enables the runtime update of graphical dashboards with no need for re-deployment, as in traditional web-based urban dashboards. In this way, the user has a constantly updated snapshot of the smart city she is monitoring. Lastly, the implementation of the architecture in a service-oriented fashion facilitates the integration of new

---

<sup>10</sup> <https://bit.ly/3cGvJAC> <sup>11</sup> <https://bit.ly/2z6pF5A> <sup>12</sup> <https://www.eclipse.org/at1/> <sup>13</sup> <https://bit.ly/2WZcYl6> <sup>14</sup> <https://bit.ly/3eQQef6> <sup>15</sup> <https://bit.ly/2Z9bzuR> <sup>16</sup> <https://github.com/neo4emf/Neo4EMF> <sup>17</sup> <https://emfjson.github.io/> <sup>18</sup> <https://www.eclipse.org/emfstore/> <sup>19</sup> <https://www.mysql.com/> <sup>20</sup> <https://bit.ly/353KD0P> <sup>21</sup> <https://spring.io/> <sup>22</sup> <https://www.eclipse.org/xtend/>

input parameters in the *SC Model* when the assessment of new KPIs are required. In this case, after the modeling of the new KPI in the KPIs model, we can add a publisher for every input parameter required by the new KPI definition. Then, we have to connect the new publishers with the parameters names in the *SC Model*. Eventually, we proposed an online alternative deployment pattern to overcome the limitations of a standalone framework. We further provided alternative technologies to implement the architectural components to confirm the technology-independence of the defined architecture.

While developing the overall *MIKADO* framework, we incrementally performed several types of evaluations on different components and of different aspects, from various points of view. The results of evaluations are shown in the following Chapter.

## Chapter 6

# Evaluation Results

In this Chapter, we show different types of evaluations performed incrementally while developing the *MIKADO* framework. Specifically, in Section 6.1 we show a demonstration case to prove the feasibility of the KPIs assessment approach. Section 6.2 reports the results of a survey collecting opinions about the understandability of the defined DSLs used to model smart cities and KPIs. Instead, Section 6.3 reports two sets of experiments testing the performance of the evaluation engine’s execution time, while in Section 6.4 the latency analysis of the service-based extension of the KPIs assessment approach is discussed. Section 6.5 closes the Chapter with some discussions.

### 6.1 Demonstration Case

We report in this section, a demonstration case in which we show an example of instantiation of the whole smart cities KPIs assessment process to prove the feasibility and expressiveness of the whole methodology.

#### 6.1.1 Selection of Real-World KPIs

The preliminary step in the KPIs assessment process concerns the selection of a set of KPIs interesting for the candidate city. We report in Table 6.1 a list of KPIs coming from multiple sources. In particular, ITU [7] provides a list of indicators for smart cities to capture the performance of a city in multiple dimensions. Meanwhile, CITYkeys [13] defines KPIs and data collection procedures for common and transparent monitoring of smart city solutions across European cities. DigitalAQ [14], instead, describes a set of indicators supporting the sustainable economic growth through the integration of advanced technologies, in particular for the city of L’Aquila (Italy).

We listed in Table 6.1, the KPIs that we selected for our demonstration case. As already discussed in Section 3.1, every KPIs source has a different ways of organising the KPIs. For instance, ITU and DigitalAQ, classify the KPIs in a hierarchical structure with *dimensions* (e.g., *Economy*, *Environment*, and *Society and Culture*) and *sub-dimensions*. ITU further structures sub-dimensions in multiple *categories*. Meanwhile, CITYkeys organizes KPIs in *themes* and *sub-themes*.

TABLE 6.1: Sources of the KPIs definition used in the experiment

KPI	ITU [7]	CITYkeys[13]	DigitalAQ[14]
Green Areas	✓	✓	
Bicycle Network	✓	✓	
Air Pollution	✓		
NO2 Emissions		✓	
PM2.5 Emissions		✓	
Real-time Transport Monitoring			✓
# of Mobile Applications			✓

Among the KPIs listed in Table 6.1, we report the definition together with the dimension and calculation formula for the KPI *Air Pollution (AP)*. Following, the ITU [7] documentation, it measures the air quality based on the values reported for specific pollutants. It belongs to the dimension *Environment*, sub-dimension *Environment* and category *Air Quality*. The calculation of this KPI relies on the *Air Quality Index (AQI)* for specific pollutants (i.e., Particulate Matter (*PM*), Nitrogen Dioxide (*NO<sup>2</sup>*), Sulphur Dioxide (*SO<sup>2</sup>*), Ozone (*O<sub>3</sub>*)). The calculation formula for the AQI is reported in (6.1), where *p* refers to the pollutant whereas the *legal limit* is established by the law<sup>1</sup>:

$$AQI_p = \frac{\text{measured concentration}_p}{\text{legal limit}} \times 100 \quad (6.1)$$

The worst  $AQI_p$  determines the *Air Pollution* KPI, as in formula (6.2):

$$AP = \max(AQI_{PM2.5}, AQI_{PM10}, AQI_{NO^2}, AQI_{SO^2}, AQI_{O_3}) \quad (6.2)$$

<sup>1</sup> <http://apollon-project.it/2019/12/10/indice-di-qualita-dellaria-iaq/>

Both  $AQI_p$  and  $AP$  are measured as  $\mu g/m^3$  and evaluated w.r.t. five evaluation classes defined by the thresholds in (6.3).

$$Index \text{ evaluation class} = \begin{cases} \textit{Excellent} & \text{if } Index < 30 \\ \textit{Good} & \text{if } Index \geq 30 \wedge Index < 66 \\ \textit{Discrete} & \text{if } Index \geq 66 \wedge Index < 99 \\ \textit{Bad} & \text{if } Index \geq 99 \wedge Index < 150 \\ \textit{Terrible} & \text{if } Index \geq 150 \end{cases} \quad (6.3)$$

### 6.1.2 Modeling of KPIs Definitions

After the KPIs selection, we passed to the modeling phase. First, we defined the KPIs Model based on the definitions and formulae of the identified KPIs. In Listing 6.1, we reported the textual annotation that is part of the produced KPIs model describing the AP KPI. At lines **3** and **4**, we defined the structure of the KPI, i.e., Dimension and Category, thanks to the DSL defined in the KPIs Metamodel in Figure 4.3. Lines **5–50**, translate the AP KPI formula (6.2), which in turn depends on the formula (6.1). In particular, the percentages of pollutants concentration in the air w.r.t. their legal limits is calculated, as for instance done in lines **8–14** for the  $PM_{2.5}$  pollutant. Thus, given the pollutant measured concentration (line **11**), it is 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 percentage 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 6.3. The reported operators (e.g., GET, MULT, DIV, MAX in the Listing) are those defined by the enumeration `Operation` of the KPIs Metamodel in Figure 4.3.

```

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 max:]30.0 ,
45     Range "Good" min:[30.0 max:]66.0 ,
46     Range "Discrete" min:[66.0 max:]99.0 ,
47     Range "Bad" min:[99.0 max:]150.0 ,
48     Range "Terrible" min:[150.00
49 }
50 }
51 }
52 }

```

LISTING 6.1: Air Quality KPI definition model shown in textual concrete syntax.

### 6.1.3 Modeling of Smart Cities

During the modeling phase, also the `SmartCityModel` is defined. In Figure 6.1, we report a portion of the graphical representation of the model of the city of L'Aquila. Here, we modeled the `DataPackages` collecting the data required to calculate the KPIs and the corresponding providers. For instance, `BreezoMeter`<sup>2</sup> is a Web service providing live air pollution, pollen, and fires information of a selected geographical area. In the model, it is the provider of the real data composing the `AirMonitoring` data package (i.e., PM2.5, PM10, O3, NO2, SO2, CO2).

<sup>2</sup> <https://breezometer.com/>



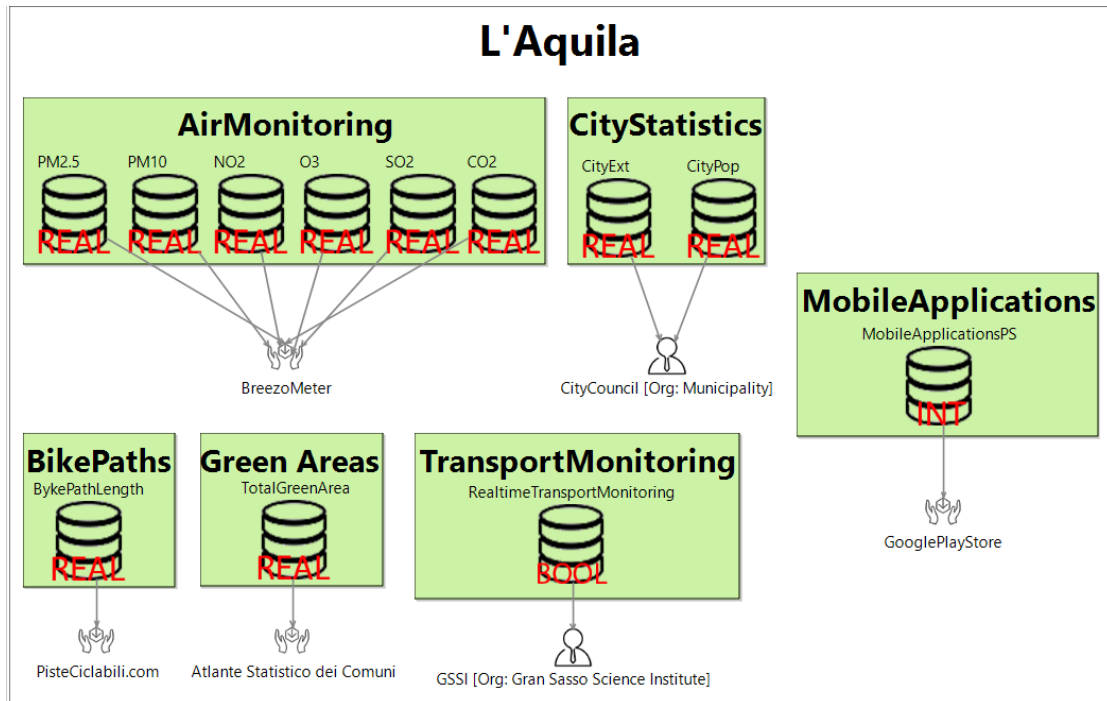


FIGURE 6.1: Graphical Representation of the Smart City Model for the city of L'Aquila.

#### 6.1.4 KPIs Assessment through the Evaluation Engine

After the definition of the two modeling artifacts describing the candidate smart city and the KPIs to be calculated on it, the assessment phase can be triggered. Indeed, the `SmartCityModel` and the `KPIModel` are given as input for the KPIs evaluation engine (Figure 4.1). In Figure 6.2, we report the three view of the considered models for this demonstration case, highlighting the mapping between the data values defined in the SC model and the parameters in the KPIs one. In particular, on the left side, there is the representation of the smart city of L'Aquila with the values collected in the air monitoring data package. On the central panel, the KPIs Model reflecting the *AP* KPI definition. Every element in the models tree owns its corresponding properties as shown in the property view displayed in the top-right panel. When the evaluation engine is executed, the aggregated value defined in the KPIs Model (corresponding to line 6 of Listing 6.1) and measuring the *AP* KPI is actualized with the calculated value, as shown in the bottom-right side panel, which shows the property view after the execution. In the console in Figure 6.3, we reported the results of the KPIs assessment for the city of L'Aquila.

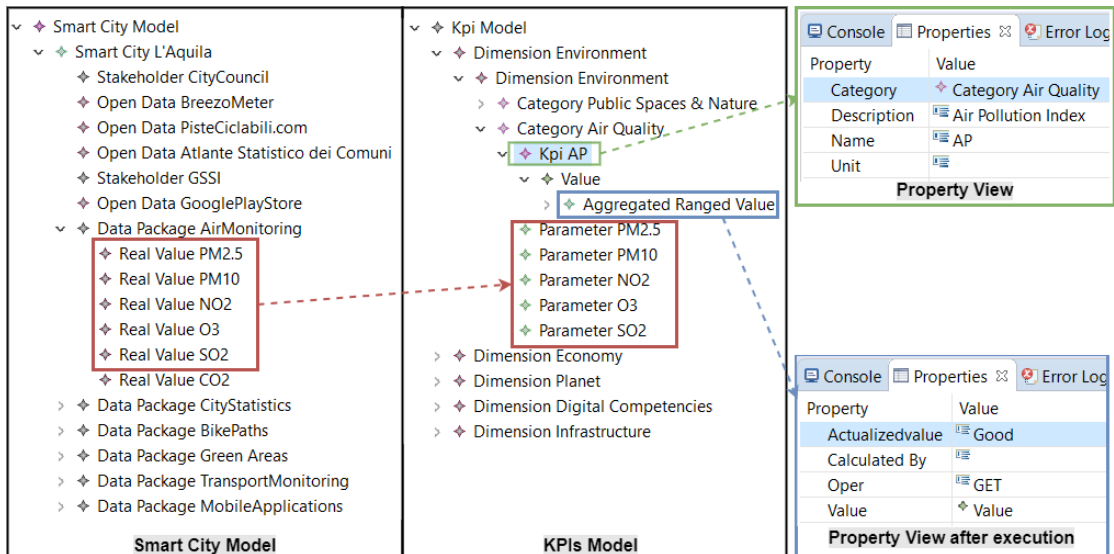


FIGURE 6.2: Examples of the input and output models during the assessment process.

```

Console Properties Error Log Problems Profiling Tools
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
=====

```

FIGURE 6.3: KPIs Assessment Results over the Smart City of L'Aquila.

### 6.1.5 KPIs Visualization through Dashboards Generation

Besides the visualization of the KPIs assessment via the console log and the actualized quality model, we provide another component (see Section 5.2.3) supporting the visualization and understanding of KPIs assessments results. This component is devoted to the production of a set of views for the KPIs model under inspection. For instance, in Figure 6.4 we reported the KPIs dashboard generated for the KPIs model elaborated for

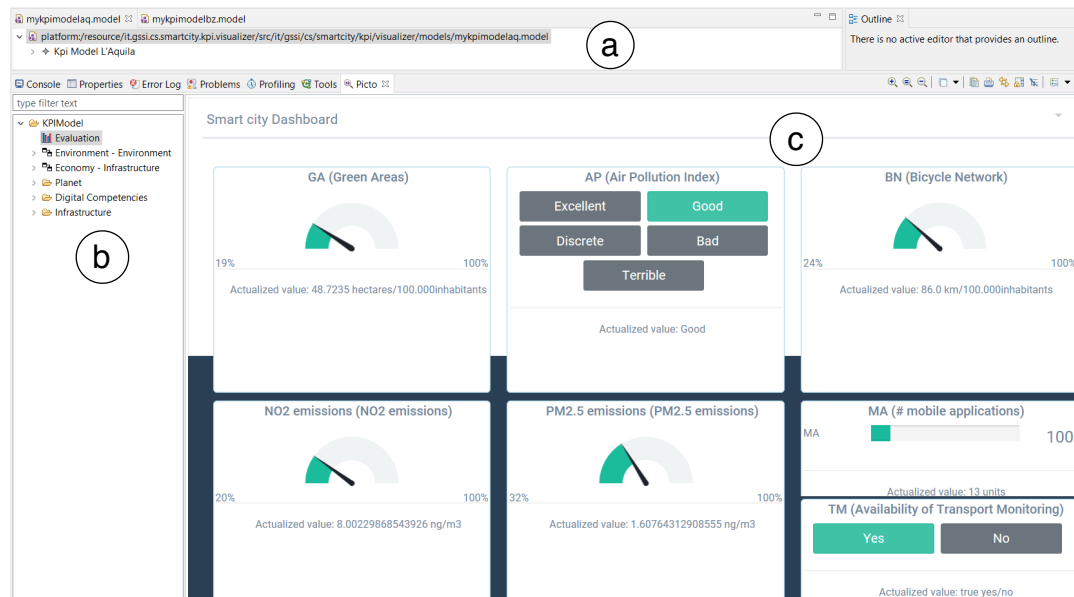


FIGURE 6.4: Overall View of the KPIs for the city of L'Aquila.

the city of L'Aquila, evaluated w.r.t. our demonstration case. In Figure 6.4, we show the correlation between the KPIs model ① and the view on the bottom ② automatically populated with multiple views. This one shows all the KPIs assessed for the city under evaluation and the results w.r.t. a target value set in the model. Indeed, in the KPIs model we declared KPIs in different dimensions and categories. On the left side of Figure 6.4, a navigator view ③ 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, in Figure 6.5 are reported the two KPIs belonging to the dimension Planet and category Pollution and Waste, namely NO<sub>2</sub> (Nitrogen Dioxide) and PM<sub>2.5</sub> (Particulate Matter) emissions with their corresponding values reported as percentage w.r.t. their target values specified in the KPI model and corresponding to 100% in the gauges dashboards. These indicators are also quantified with units, specified in the KPI model (see Figure 4.3).

### 6.1.6 Supporting Smart Cities Comparison

Another important aspect in smart cities KPIs assessment is the ability of making comparisons among multiple cities. For doing this, KPIs have to be general enough to be applicable to different cities. It is worth specifying that these types of comparisons would be significant when comparing similar cities, e.g., cities with the same dimension in terms of number of inhabitants. For instance, we modeled three Italian medium-sized cities, namely L'Aquila, Bolzano and Matera. In Table 6.2, we report the results of the KPIs

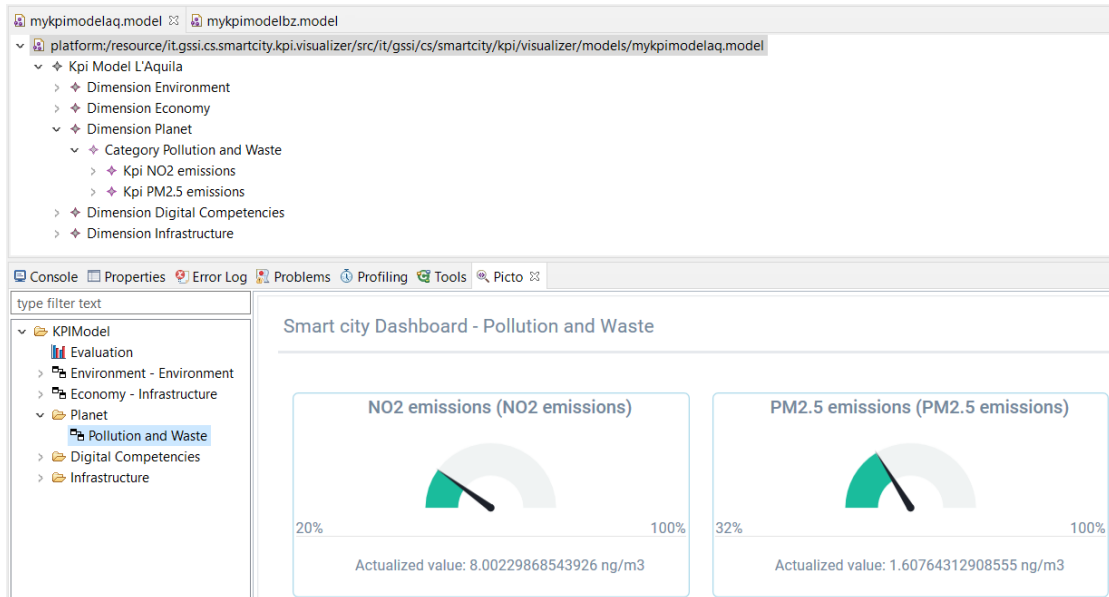


FIGURE 6.5: Detailed view of a single category of KPI.

assessments for the three selected cities that we performed with our framework. Looking at the results reported in the table we can make different reasoning about the candidate cities. For example, for the KPI Green Areas (GA), Matera surpasses the other cities by far because of the presence of wide historical areas. Also the results for the Bicycle Network (BN) KPI we can observe that the cities of L'Aquila and Matera obtained results much lower than the one obtained by Bolzano. Meanwhile, for the Air Pollution (AP) KPI, all the three smart cities have been evaluated with the class Good. For NO2 emissions and PM2.5 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.

TABLE 6.2: 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

### 6.1.7 Supporting Smart Cities and KPIs Evolution

In this subsection we sketched a slightly modified scenario-based testing [99] in which we identify a sample of scenarios and we check whether the implemented running system supports smart cities and KPIs models evolution and assessment with few modeling steps. For each scenario, an expected result is also identified and confirmed or not, as compared to the observed result. The scenarios and expected results are listed in Table 6.3, where we also categorized the scenarios with the three categories of changes reported in Section 4.3, namely subtractive, additive, and structural.

TABLE 6.3: Scenario-based testing for the evaluation of the *MIKADO*'s evolution support.

ID	Category	Scenario	Expected result	Observed result
S1	Subtractive	Remove KPIs	The KPI must be removed from the KPIs model and the generated dashboard.	After deleting a KPI from the KPIs model, it is not included anymore in the generated dashboard without affecting the assessment process.
S2	Structural	Change KPIs type	The KPI's type must be changed in the KPIs model and the generated charts must change accordingly, in the shape and resulting value.	The value of the updated KPI changed from a Real to a Boolean type in the KPIs model and the generated chart changed representation from donuts to multi-options.
S3	Structural	Change KPIs target value	The target value of a numeric KPI must be changed in the KPIs model and the generated chart must adapt the scale.	By increasing / decreasing the target value of a numeric KPI in the model, the generated chart updates the limits.
S4	Structural	Change KPIs unit measure	The unit declared for the KPIs must be updated in the generated charts.	By updating the unit for a KPI's definition, the generated chart includes the unit after the resulting actualized value.
S5	Additive	Add a data source	The new data source must be defined in the smart city model.	The definition of the new data source in the smart city model did not affect the KPIs definition and assessment process.
S6	Structural	Change a static value with no parameter	The static value must be changed manually in the KPIs model.	The value of the KPIs using the changed static value have been updated correspondingly after its manual change in the KPIs model.
S7	Additive	Add new data	The new data must be defined in the smart city model and connected to an existing or new defined data source. Also in the KPIs definition the parameter to be used must be updated.	After the definition of the new data in the smart city model, a new parameter is declared in the KPIs model with the same name of the new data and connected to the KPIs that use it in their calculation.

For instance, we applied the scenario (**S2**) in our demonstration case. Specifically, we changed the type of the GA KPI from a real to a boolean type, assuming that the indicator's aim is to verify whether in the city there are green areas or not. In Figure 6.6, we reported how the KPIs model and the generated dashboard appeared after the KPIs evolution.

All the listed scenarios in Table 6.3 envisage the re-iteration of the last two phases of the KPIs assessment process flow (see Figure 3.1), namely, the *Execution phase* involving

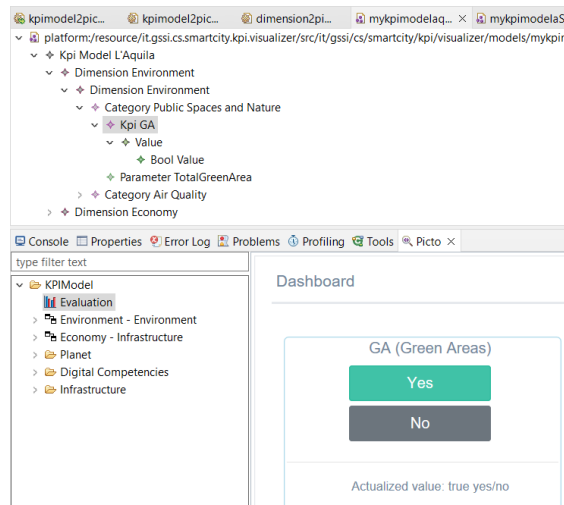


FIGURE 6.6: Scenario 2 - Change KPIs type.

the re-execution of the evaluation engine and collection of the assessment results and the *Visualization phase* with the re-generation of the graphical dashboards.

## 6.2 Understandability of *MIKADO*'s DSLs

In this section we reported an evaluation concerning the main modeling artifacts of the *MIKADO* approach, that is the Smart Cities and KPIs definition DSLs. We inspected the perceived understandability of the DSLs through an expert survey which is described in the following. All the artifacts used in the survey together with the anonymous results are available online<sup>3</sup>.

### 6.2.1 Survey Setup and Execution

To measure the understandability of our DSLs, we drew up an online survey<sup>4</sup> and sent it to different smart cities stakeholders. Since we consider spreadsheets as a common approach used in the KPIs evaluation domain, we structured the survey by proposing different KPIs expressed within spreadsheets as well as with our DSL. In particular, for each KPI, we compare the DSL definition with the spreadsheet formula, by asking an opinion for each definition with an open answer question, and to rank, with a likert scale (from 1 to 10), the understandability of the current definition. Moreover, we asked to the user, by means of multiple-choice questions, which of the two approaches may be more usable for her for modifying or defining new KPIs, considering a period of training of 1 week or a training period of 10 weeks. Eventually, we asked to the user, again

<sup>3</sup> <https://bit.ly/3kXIJ8x> <sup>4</sup> The survey is available at this [link](#)

via multiple-choice questions, to express her preference between the smart city graphical representation (as in Figure 6.1) and the spreadsheets tabular definition.

The survey has been submitted to 10 smart cities and/or KPIs experts. We decided to involve an heterogeneous group of participants with different perspectives and expertise about smart cities and their evaluation. In particular, the experts came 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 Smart City Looks Like Summer School 2019 edition<sup>5</sup>; (iii) the *smart city committee of the city of L'Aquila*<sup>6</sup>, nominated by the mayor and made by smart cities experts. It is worth specifying that the participants have not been trained about the DSL before receiving the survey, but they just had a brief introduction. However, we cannot exclude that they had background on the use of spreadsheets, given their diffusion and popularity.

### 6.2.2 Survey Results

In Figure 6.7, we reported 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. The results show that our DSL received better votes for all KPIs apart from the Mobile Applications (MA) one. This indicator is easily calculated by summing up the number of available mobile applications in the city, thus, in spreadsheets it does not require complex formulae, as in our DSL. On the contrary, all the other KPIs characterized by more complex calculations, resulted to be more comprehensible with our DSL. In Figure 6.7, we reported also the information about the number of responses of the type "*I don't know*" to the question that was asking to describe what the defined formula compute, through coloured dots in the chart. For the spreadsheet formulae we collected at least one "*I don't know*" for every KPI definition. Also for the Transport Monitoring (TM) KPI that is simply measured with a boolean value, the users found it easier to understand the spreadsheet formula.

As concerns the users preferences about using our DSL or the formulae in the spreadsheet, or even none of them, in Figure 6.8 we reported the collected results. When considering a period of training of 1 week (Figure 6.8 left side), 40% of the users affirmed that they would be in favor of using our DSL, and only a 20% of users expressed preferences to spreadsheets. Meanwhile, 30% of the users would like to use both of them, indistinctly. Meanwhile, when considering a period of training of 10 weeks (Figure 6.8 right side), 50% of the users would use both the DSL and the spreadsheets, indistinctly. For the remaining 50% of the users, 30% would prefer our DSL and 20% the spreadsheets.

<sup>5</sup> <https://cs.gssi.it/summerschool/> <sup>6</sup> <https://bit.ly/3xqKLFd>

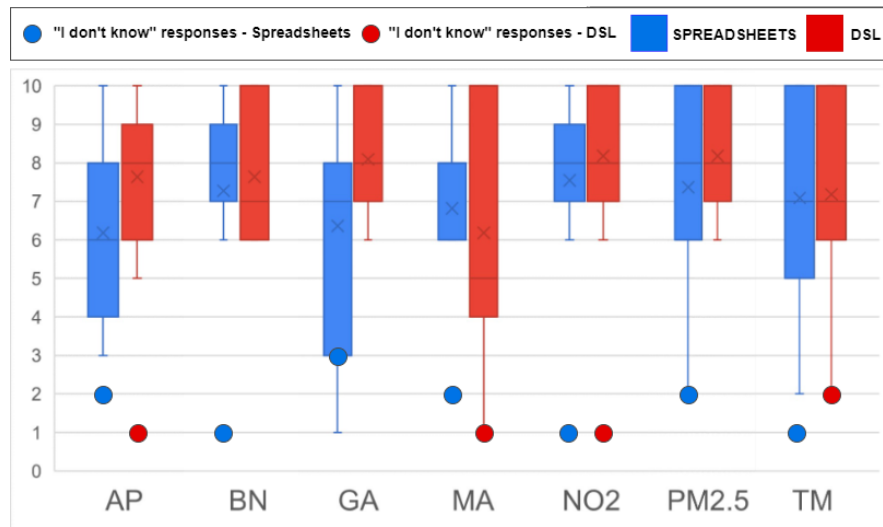


FIGURE 6.7: The boxplots report the votes given in the likert scales for each evaluated KPI w.r.t. our DSL and the spreadsheet formulae. The dots indicate the number of responses "I don't know" given for each KPI description.

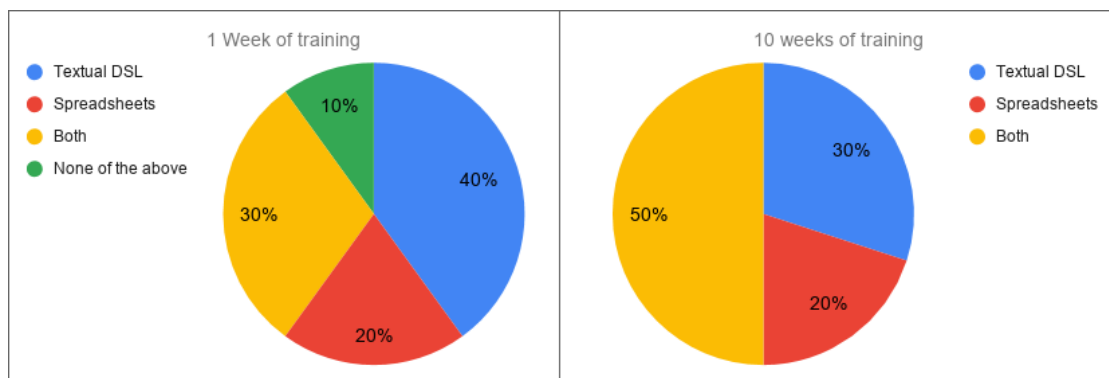


FIGURE 6.8: Users preferences between our DSL and spreadsheets when considering a period of training of 1 week (left side) and 10 weeks (right side).

Eventually, in Figure 6.9 we reported the users preferences w.r.t. the graphical smart cities representation used in our approach and the tabular one in the spreadsheets. It is shown that 40% of the users liked both representations, indistinctly.

The collected opinions tell us that our DSL is perceived more comprehensible w.r.t. spreadsheets-based solutions particularly when considering KPIs models showing more complex calculations. Moreover, as regards the smart city model, it is easily comprehensible also when represented in a spreadsheet. From the results, we can also argue that with a short training period the users would be more confident in using our DSL w.r.t. spreadsheets.

**Threats to validity.** A first threat to validity of this type of experimentation is related with the bias that the experts who participated in the survey may had by our



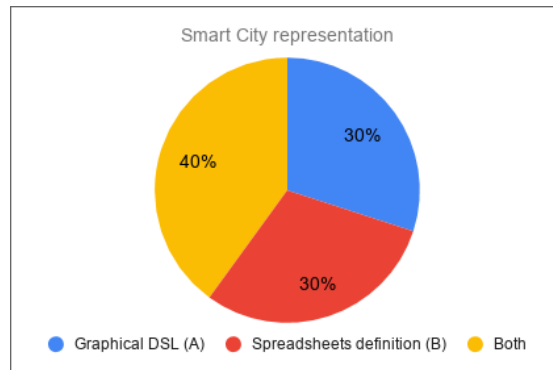


FIGURE 6.9: Users preferences w.r.t. the two proposed smart city’s representations, graphical vs. tabular.

expectations that may unintentionally leak from the questions. However, to overcome this problem we previously submitted the survey to senior colleagues in order to assess it in terms of independence from our expectations.

Instead, the limited number of participants to the survey may be a threat to the external validity. We tried to mitigate the shortage of participants by selecting potential participants with heterogeneous perspectives, i.e., business, academic, and smart city experts. This, in turn, may represent another threat to the external validity, since the non-homogeneity of the selected groups of participants, with different backgrounds, may be hard to compare. However, we observed quite concise and homogeneous answers, despite the diverse backgrounds of participants.

### 6.3 Performance of the *MIKADO* Framework

In this Section, we report an incremental evaluation of the scalability of the *MIKADO* approach w.r.t. the increasing size of smart city models and number of KPIs. Specifically, we performed two experiments measuring the execution time of the evaluation engine. In Section 6.3.1 we evaluate the approach as an EOL configuration, whereas in Section 6.3.2 we evaluate the approach extended with an automatic Java launcher. Both experiments fall in the area of performance evaluation of MDE artifacts [100], since the evaluation engine devoted to KPIs’ calculation can be seen as an interpreter of the proposed modeling language. Indeed, we focused on the measurement of the execution time of the evaluation phase only, without considering the modeling phase. For this reason, both input models used in the experiments have been automatically or manually filled without distinction, since we did not consider the execution time of the modeling phase. The goal of our experiments is that of checking the evaluation engine execution time (in milliseconds) w.r.t. the *size of the input models*, i.e., the number of elements in the models. All the

experiments were executed by running the standalone version of *MIKADO*, 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.

### 6.3.1 Scalability of the Evaluation Engine

**Experiment Setup.** We performed a preliminary evaluation for which we designed a smart city model, containing the instantiation of every concept of the metamodel, and a KPIs model initially made by one dimension with one category of 8 KPIs, thus to cover all the calculations defined in the KPIs metamodel (e.g. *SUM*, *AVG*). For this evaluation, we conducted four experiments each of them performing 10 execution runs of the evaluation engine. In the next paragraph, we incrementally describe each experiment and the obtained results.

**Experiment Results.** The results of the first experiment, *Exp1*, are reported in Figure 6.10. As mentioned in the experiment setup, during this experiment, for each run, we increment the complexity of the SC model by changing the number of modeled smart cities from 1 to 10. On this model, we measure the 8 KPIs in the KPIs model in each run. With these settings, the models size goes from 200 to 632 elements and the execution time goes from 16 ms to 79 ms. For the second experiment, *Exp2*, the results are reported in Figure 6.11. During this experiment the SC model is fixed and contains the definition of 10 smart cities. Instead, the KPIs model at every run increases the complexity of the KPIs calculations, by adding new nested operations, one KPI at a time. For this experiment, the size of the models goes from 638 to 676 and the execution time goes from 82 ms to 198 ms. Interestingly, despite the models size does not increase significantly, the execution time particularly increases in the last run. This is due to the fact that this run involves a KPI whose calculation combines the *range* operation, i.e., the most time-consuming one, with a basic operation (e.g., *AVG*). From this observation, we designed the third experiment, *Exp3*, such that, at every run, we add a *range* calculation in the definition of each KPI, one KPI at a time. In Figure 6.12, we reported the results of this experiment. Here, the size of models slightly increases from 692 to 803 elements (only 127 elements more than the last run in *Exp2*) while the execution time ranges from 306 ms to 1279 ms, by showing a considerable increase, thus confirming our prediction about the time consuming of calculations including the range operation. However, the overall execution time is still reasonable for the given models size. Eventually, in the fourth experiment, *Exp4*, at every run we increment by one the number of dimensions in the KPIs model, where each dimension includes 8 KPIs with complex calculations.

Consequently, the number of evaluated KPIs goes from 16 in the first run to 80 in the last one, always assessed on top of the 10 smart cities in the SC model. This means that in the last execution we assessed 80 KPIs over 10 smart cities, i.e., 800 KPIs in the same run. In Figure 6.13, we can see that the size of the models goes from 1124 to 3692 elements and the execution time ranges from 2426 ms to 12892 ms.

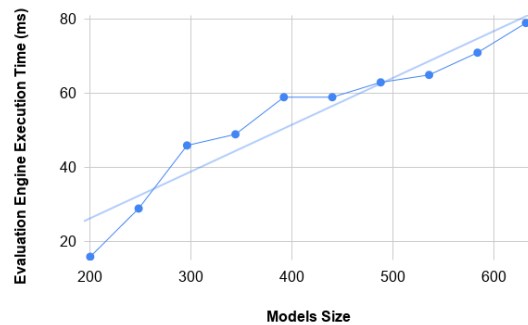


FIGURE 6.10: *Exp1*: increasing the number of evaluated smart cities.

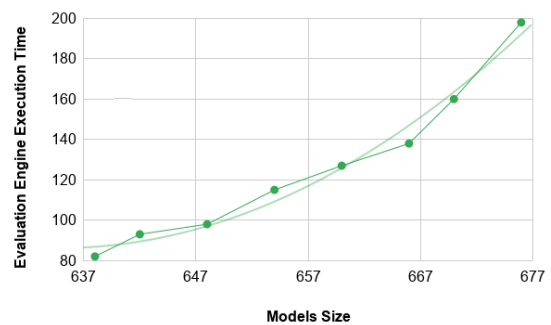


FIGURE 6.11: *Exp2*: increasing the complexity in the calculation of each modeled KPI.

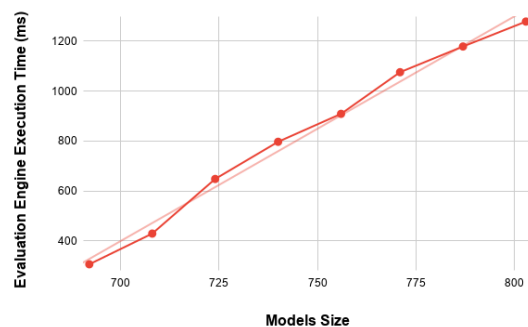


FIGURE 6.12: *Exp3*: make each KPI of type *range*.

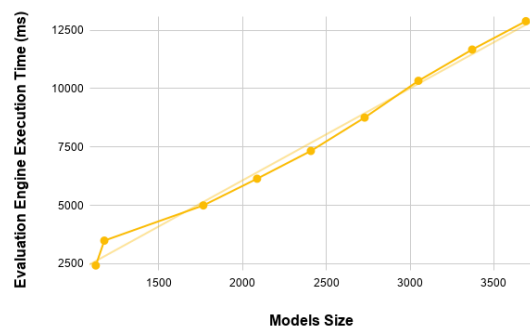


FIGURE 6.13: *Exp4*: increasing the number of KPIs.

In summary, this first set of experiments points out two main findings: (i) the efficiency in terms of the evaluation engine's execution time, i.e., the core component of our approach, since all the experiments show a linear or polynomial (of degree 2) increase of the execution time w.r.t. the increasing models size; (ii) promising scalability results showed by *Exp4*, indicating that the system takes 12.9 seconds for assessing 800 KPIs over 10 smart cities.

### 6.3.2 Empowering of the Evaluation Engine

As introduced above, in this section we evaluate the execution time of the evaluation engine after it has been extended with an automatic Java launcher. This means that the user has not to manually load the input models to the engine before running the evaluation, neither to store the evaluated KPIs model after the assessment.

**Experiment Setup.** For this second evaluation, we designed 4 increasingly complex scenarios, whose settings are reported in Table 6.4 and are similar from those of the previous evaluation. In particular, we performed also here 4 experiments with 10 execution runs of the evaluation engine. In each experiment, the SC model increases the number of modeled cities in every run. Meanwhile, the KPIs model increases its complexity from an experiment to another and remains fixed in every run.

Scenario	SCs	SC elements	KPIs	KPIs elements	Specific Computation
<b>EXP1</b>	1-10	49-481	8	151	Single calculations
<b>EXP2</b>	1-10	49-481	8	195	Nested calculations
<b>EXP3</b>	1-10	49-481	8	322	Range calculations
<b>EXP4</b>	1-10	49-481	80	3211	Multiple KPIs dimensions

TABLE 6.4: Increasingly complex scenarios for the scalability assessment.

Differently from the previous set of experiments, 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.

**Experiment Results.** We reported the results of the 4 experiments in terms of execution time in Figure 6.14. For each run of the experiments, the chart reports the average among the times of the 11 executions. The complete measured execution times are available online<sup>7</sup> and they show that, 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. In particular, in Figure 6.14 the blue line represents the results of the first experiment, *Exp1*, where the models size goes from 200 (49 SC elements plus 151 KPIs elements in Table 6.4) to 632 (481 SC elements plus 151 KPIs elements in Table 6.4) elements and the execution time goes from 72 ms to 124 ms. For the second experiment, *Exp2*, results are reported by the red line. In this case, 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 shapes the results of the third experiment, *Exp3*, with an execution time ranging from 202 ms to 654 ms. Here, we have an increase w.r.t. the previous two experiments. This highlights the complexity of calculations due to the range operation, as happened in the first set of experiments. However, the overall execution time is still reasonable for the given models size that goes from 371 to 803 elements. Eventually, in the fourth experiment, *Exp4*, whose results are represented

<sup>7</sup> <https://bit.ly/330iMzA>

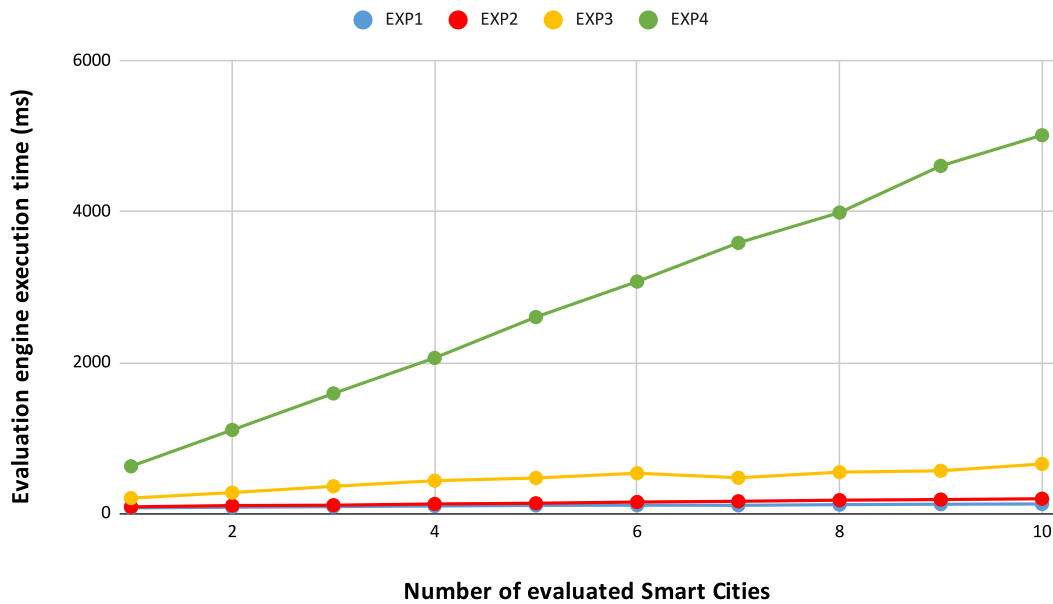


FIGURE 6.14: Execution times of the evaluation engine resulting from the experiments.

with a green line in Figure 6.14, 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). The execution time ranges from 624 ms to 5015 ms, with the models size going from 3260 to 3692.

In this second evaluation we can highlight two main findings: (i) the efficiency in terms of the evaluation engine’s execution time, since also in this case, all the experiments clearly show a linear increase of the execution time w.r.t. the increasing models size, without any peaks. Moreover, the reduction of the computation time of 50% compared to the experiments of the first set. 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. (ii) promising scalability results are shown by *Exp4*, indicating that the system takes approximately 5 seconds for assessing 800 KPIs over 10 smart cities.

Furthermore, we can observe that in Section 6.1.6, we modeled three real-world medium-sized Italian cities, namely L’Aquila, Bolzano, Matera, whose average model size is equal to 80, while the size of the KPIs model used for the comparison between them is 290 elements. This, from a qualitative perspective, considering that the smart city models size used in both evaluations is slightly higher (i.e., 49-481 in Table 6.4), 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.

### 6.3.3 Threats to validity

In both the reported performance evaluations, the assessment process strictly depends on data about the smart cities under evaluation. If this data are not coming from reliable providers but are estimated by domain experts, they can lead to incorrect KPIs measurements. However, the overall process implemented in the evaluation engine is not affected, since KPIs calculations are usually based on standard KPIs definitions. Starting from this observation, the experimentation performed for this evaluation may be internally biased from the parameters we used as input to calculate KPIs, which are realistic but not real, i.e., not related to specific cities. More complex data might lead to more time consuming calculations.

Despite we defined artificial models with the aim of instantiating all the entities in the smart city and KPIs metamodels, the fact that the results of the experiments have been obtained on a set of demonstration cases modeled by us could threaten the external validity of our experiment.

## 6.4 Latency Analysis of Service-based Continuous KPIs Assessment

In this Section we report an experimental evaluation of the service-based continuous KPIs assessment, presented in Section 5.3.1 as the hybrid deployment of the flexible architecture for KPIs assessment in smart cities. In particular, we compare the *hybrid deployment* with the *standalone* version (i.e., its baseline), where the KPIs parameters continuous monitoring and runtime models update were not available. To this aim, we defined the following research questions that guided the evaluation:

**RQ1:** What is the impact of continuous monitoring and runtime update of models on the evaluation framework? How does the presented approach compare with its baseline (see Section 5.2) w.r.t. the framework efficiency?

**RQ2:** What is the impact of KPIs *input data retrieving*, *models update* and *evaluation engine execution* tasks on the latency? How does the presented approach compare with its baseline (see Section 5.2) w.r.t. the impact to latency of these tasks?

**Experiment Setup.** To answer to **RQ1** and **RQ2** we have conducted a double experiment by comparing the *baseline framework* (see Section 5.2), requiring manual data retrieving and models filling, with the fully *automated framework*, enabling continuous monitoring and runtime models update presented in Section 5.3.1. We consider one subject smart city and 6 dynamic KPIs. As *dynamic* KPIs we consider: *Humidity*, i.e., the percentage of the detected humidity level, *Air Pollution* (see formula (6.2)), *PM2.5*, *NO<sup>2</sup>*, measuring the *PM2.5* and *NO<sup>2</sup>* pollutants levels in the air, *Urban Heat Island (UHI)*, i.e., the difference in air temperature between the city and its surroundings (provided by two different sources), and *Green Area (GA)* as a less dynamic KPI (see formula (3.1)). Three Publishers are considered as data sources for the input parameters of these KPIs. *BreezoMeter*<sup>8</sup> is a website offering *open API* exposing environmental data and insights to third party applications. It can be queried via a specific *url* taking the latitude and longitude of a city as parameters, and it replies with JSON data providing values for different pollutants and other interesting info. *OpenWeather* map<sup>9</sup> provides information about weather forecasts, air temperature, etc. for any coordinates via an open API. Lastly, an IoT infrastructure based on *Tinkerforge*<sup>10</sup> sensors that publishes data from a Tinkerforge Air Quality Bricklet<sup>11</sup>. Tinkerforge is a system of building blocks, based on pluggable modules and APIs, available for many programming languages. Using this board we have assembled a smart environment sensor in our department measuring multiple air quality parameters, e.g., humidity and temperature<sup>12</sup>.

In the following we describe the design of the two experiments, namely **Baseline** and **Automated**, where the former makes use of the standalone version of *MIKADO*, and the latter uses the hybrid version, namely the service-based solution, respectively.

- **Baseline:** We have involved four persons, one of the authors and three smart city domain experts. The experiment lasted 6 hours during which the three domain experts have been asked to occasionally monitor the 3 data sources, namely *BreezoMeter UI web interface*, *OpenWeather map* and the display of the *Tinkerforge Air Quality device*, providing the required parameters for the 6 considered KPIs. They simulate, for instance, three municipality's employees responsible for monitoring KPIs data sources while also accomplishing other tasks during their working day. They alerted the modeler (one of us) anytime they observed changes in the parameters. The modeler updated the *SC Model* accordingly, and run the evaluation engine to get the new results in the updated dashboards. They have been asked to measure and report the number of updates and the time required to perform them.

<sup>8</sup> <https://www.breezometer.com/>

<sup>9</sup> <https://openweathermap.org>

<sup>10</sup> <https://www.tinkerforge.com/en/>

<sup>11</sup> <https://bit.ly/3xngllU>

<sup>12</sup> A picture of the assembled board is available at <https://bit.ly/3Y7HhVw>

- **Automated:** For this experiment, the framework run on a Macbook Pro 2019, 2,3 GHz 8-Core Intel Core i9 processor, 32 GB 2667 MHz DDR4 RAM and 2TB SSD of storage. This laptop runs the Mosquito client, all the publishers, the subscriber and the evaluation engine. The publishers are executed locally, except for the Tinkerforge publisher (i.e., see Listing 5.5) which is executed on the assembled board. We run the experiment for 6 hour with the following configuration for the three publishers: (i) *BreezoMeter* publisher requests updates for all the pollutants every minute; (ii) *OpenWeather* publisher requests updates on the temperature every minute; (iii) *Tinkerforge* publisher requests updates for temperature and humidity every 1, 2, 5, and 10 seconds. Anytime (and only when) the *SC Model* is not up to date, the subscriber updates the parameters in the model with the new received values and it runs the evaluation engine that triggers the dashboard's update. All these activities have been automatically monitored and measured in a log.

**Results for RQ1.** To answer RQ1 we inspected the impact that continuous monitoring and runtime update of models on the KPIs assessment framework. To this aim, we reported the normalized trend of the execution times (reported as normalized milliseconds on the y-axis) corresponding to the data retrieving, *SC Model* update and evaluation engine phases in the Automated and Baseline experiments (*Exps*), in Figures 6.15 and 6.16 respectively.

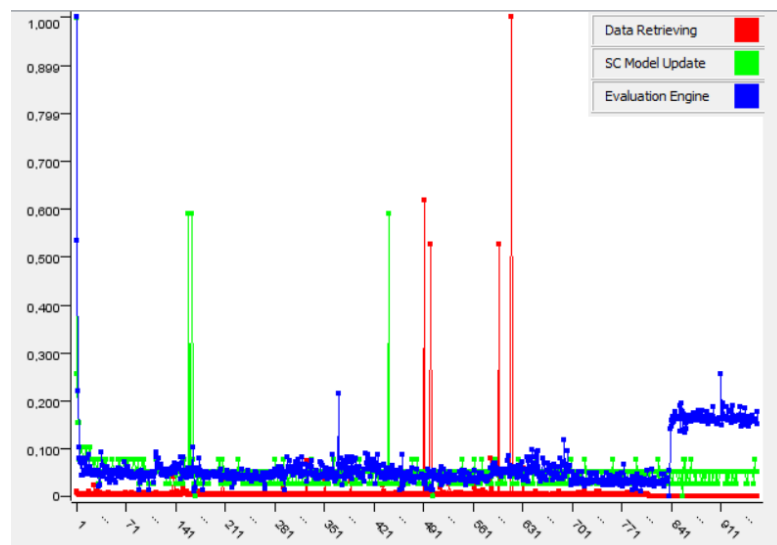


FIGURE 6.15: Data retrieving, SC Model update and evaluation engine execution times. Automated: 963 interactions.

Specifically, both *Exps* lasted 6 hours, performing a total of 963 interactions in the automated case and 18 in the baseline one, due to the fact that the baseline *Exp* envisages



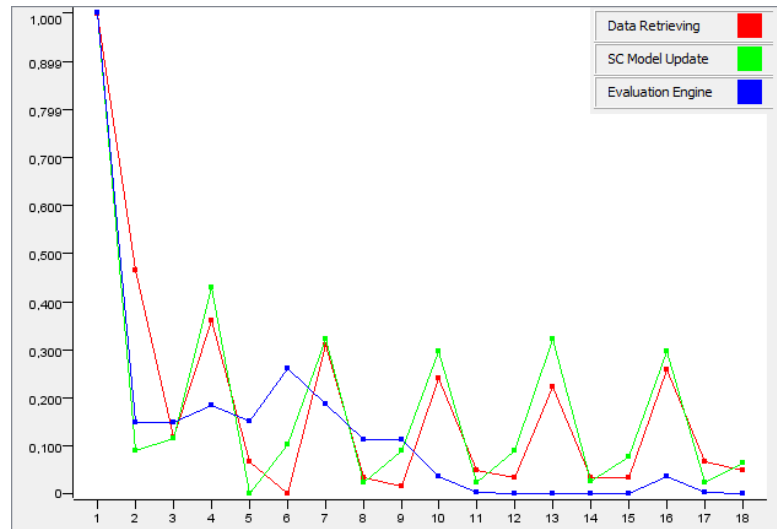


FIGURE 6.16: Data retrieving, SC Model update and evaluation engine execution times. Baseline: 18 interactions.

two manual tasks, namely the data retrieving and model updates, performed less frequently ( $\sim$  hourly) than the automated one. Each interaction can generate more than one *SC Model* update, since a single publisher can publish on more than one topic, as for instance the *BreezoMeter* publisher that publishes on 5 topics (see Figure 5.9). In Figure 6.15, we can see that the data retrieving is the less time consuming task, with a few peaks possibly due to delays on the network and/or servers workload. The *SC Model* update requires slightly more than data retrieving with also a few peaks that may depend on the number of updates to be performed. Lastly, the evaluation engine, after the first run taking more time due to the initial upload of different models, settles on a constant time, also due to caching features of the framework. However, this time increases during the last hour of the *Exp*. This deserves further analysis, although it might be due to a machine overloading. In Figure 6.16, instead, we can observe that the evaluation engine, which is an automated task in the Baseline *Exp* apart that it has to be launched manually, after the first run requiring the models upload, gradually settles on negligible time. The data retrieving and the *SC Model* update phases follow a common trend showing peaks in correspondence of the interactions of those publishers publishing on more than one topic, thus requiring more than one model update. Lastly, both data retrieving and model update take more time for the first run when the data sources interfaces are opened and the Eclipse platform is launched. Moreover, in Figure 6.16 is also visible the learning curve of the involved users leading to a gradual decrease of the time required for the three phases. Eventually, the Automated *Exp* exploiting continuous monitoring and runtime updates enables a larger number of interactions performed faster w.r.t. the Baseline approach. This increases the framework efficiency, while actually providing up-to-date data in real time.

**Results for RQ2.** Figure 6.17 shows to which extent the *data retrieving*, *SC Model update* and *evaluation engine* execution contribute to the framework latency, in both the Automated and Baseline *Exps*. Specifically, Figure 6.17 reports the average execution time for each of the three phases in milliseconds (ms), for both *Exps*. As expected, each of the three phases requires quite more time in the Baseline *Exp*, where the data retrieving and model update (performed manually) are the most time consuming ones. Moreover, the latency in the Baseline *Exp* could also depend from communication issues. For instance, more than one of our users can communicate with the modeler at the same time to report about changes in the parameters they were monitoring, implying to the modeler to coordinate the received information and perform multiple updates in the *SC Model*. The evaluation engine execution also contributes more to the latency in the Baseline *Exp*, since it has to be manually launched anytime changes are applied in the *SC Model* due to the evolution of the KPIs input parameters.

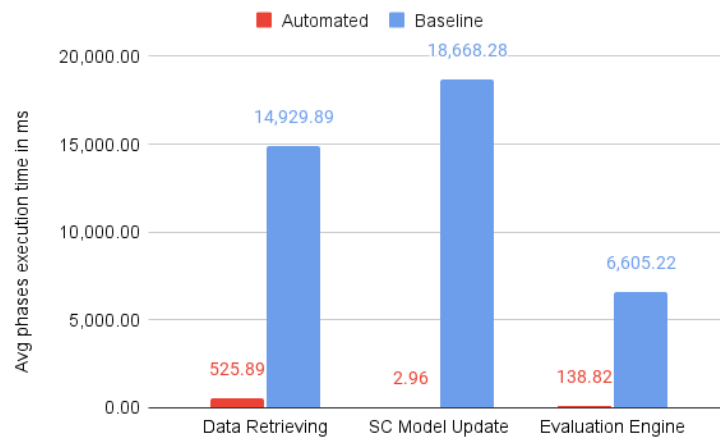


FIGURE 6.17: Latency contributed by the three phases.

**Threats to validity.** Concerning the refactoring enabling KPIs continuous monitoring, current limitations concern the generalizability and scalability of the approach. Although it has been applied on a single smart city to evaluate 6 KPIs, it uses techniques (e.g., PubSub pattern, open APIs) that can be generalized and extended to more complex systems, as long as there are accessible data sources to get real input parameters for *dynamic* KPIs to be measured on real smart cities. Indeed, PubSub is known to offer better scalability w.r.t. traditional client-server, by means of parallel operation, message caching [101]. Of course, the message-oriented middleware might add a network latency delay. However, keeping the data retrieving and update of models as manual tasks is impractical, considering the huge number of KPIs identified in standard guidelines by international institutes [7].

## 6.5 Discussion

In this Chapter, we show different types of evaluation of the *MIKADO* framework from different perspectives. First, we prove the feasibility of the overall KPIs assessment process envisaged by the proposed framework, by applying it on a real demonstration case. In particular, we documented all the steps needed to perform KPIs assessment over a smart city, from the selection of the KPIs to evaluate to the visualization of the assessment's results. Moreover, we show how smart cities comparisons are supported by computing the same set of KPIs. Secondly, we reported the results of survey that we submitted to a group of smart city and/or KPIs experts to investigate the understandability of the *MIKADO* DSLs. From the results, we saw that our DSLs are perceived more comprehensible w.r.t. spreadsheets-based solutions, specially when modeling complex KPIs. Thirdly, we assessed the performance of the evaluation engine execution time w.r.t. scenarios of increasing complexity, i.e., increasing size of the input models. The main findings of the reported experiments are two-fold: (i) the efficiency of the evaluation engine execution time, since in every experiment it showed a linear increase; (ii) promising scalability results, since in the last experiment the system took approximately 5 seconds to assess 800 KPIs over 10 smart cities. Lastly, we performed an evaluation of the service-based KPIs assessment deployment by comparing it with the standalone version. It shows how the evolved framework enables a continuous KPIs evaluation, by drastically decreasing ( $\sim 88\%$ ) the latency contributed by the data retrieving and models update phases, compared to the baseline framework.

In the next Chapter we propose an abstraction of the *MIKADO* framework in order to apply it in the assessment of different subjects than smart cities. In this way, we will evolve the proposed framework into a multi-level QES, by further prove the generalizability of the KPIs assessment approach.

## Chapter 7

# Generalizability of the Proposed Framework

Due to the challenges and limitations related to QES, highlighted in Section 3.2, and considering the benefits of a model-based KPIs assessment framework, in this Chapter, we propose the abstraction of the smart cities assessment approach presented in Chapter 4 for the automatic quality assessment of subjects coming from different domains, further showing the domain-independence of *MIKADO*. In particular, Section 7.1 starts with some motivations to the need of abstracting the assessment approach. Then, in Section 7.2 we give an overview of the multi-level QES that we propose, by presenting different domains in which it may be applied. In Sections 7.3 and 7.4 we propose the two multi-level hierarchies of models for the definition of subjects and quality metrics, respectively. Furthermore, in Section 7.5 we present a re-factoring of the evaluation engine artifact in order to support multi-level navigation of models. Section 7.6 shows how the code-generation process devoted to graphical dashboards generation is also supported in the multi-level framework. The chapter closes with some discussions in Section 7.7.

### 7.1 Motivations

The model-based KPIs assessment approach presented in this dissertation is based on the uniform modeling of both subjects under evaluation and the quality metrics selected to evaluate them. We have shown that it supports the customization of metrics w.r.t. different subjects and requirements and their evolution over time, by exploiting MDE techniques. Basically, it enables a round-trip process for the performance analysis of the assessed subjects and the results interpretation, supporting decision-making process. 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 by using existing tools, e.g., spreadsheets, or developed as independent systems, e.g., Web or standalone applications. For this reason, we decided to exploit Multi-Level Modeling (MLM) to realize QESs for multiple domains sharing commonalities in this respect, namely the overall objective of assessing a given subject passed as input, where the assessment is executed on top of a quality definition. Specifically, we aim to transform the KPIs assessment approach based on two-level modeling into a multi-level based one by defining a *multi-level framework* that allows to customize QESs for different domains. In particular, we state that a MLM approach guarantees the same expressiveness and modeling potentialities of traditional two-level modeling techniques, while further pushing forward reuse and customization. In this way we support the reduction of time-to-market for the modeled systems and error-proneness.

## 7.2 Bringing *MIKADO* to Multilevel

In this Section, we present the abstracted *MIKADO* approach for the development of QESs to support quality assessment in multiple domains. We report in Figure 7.1 an overview of the abstracted approach indicating the main phases together with their input and output elements. The approach relies on a uniform and customizable way of

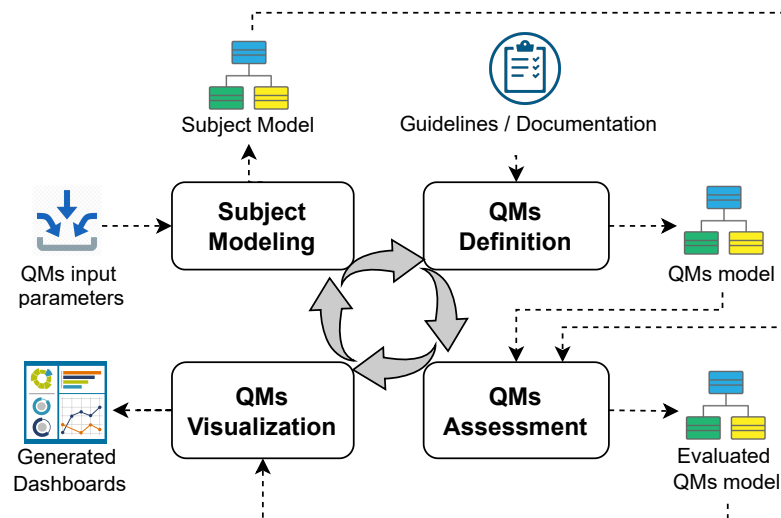


FIGURE 7.1: Overview of the quality assessment approach.

modeling of the subject under evaluation and the QMs that have to be calculated over it. We sketched the process behind the approach as a cycle because of the nature of decision-making processes. In particular, the overall quality evaluation approach consists in four main phases. The quality assessment of a subject starts from the *Subject Modeling* phase,

during which the subject under evaluation is modeled, by means of MDE techniques. In the *Subject Model*, Subjects are designed in terms of data sources (e.g., open data, IoT services) and data types. This way, the *Subject Model* provides the *input parameters* needed to calculate the QMs of interest, as we will see in the following. In the *QMs Definition* phase, by following *Guidelines/Documentation*, the user models or selects the relevant QMs for the subject under evaluation. In the *QMs Model* given as output, the calculation formulae of the selected QMs are defined by using DSL. The designed *Subject Model* and *QMs Model*, are the inputs for an evaluation engine that executes the *QMs Assessment* over the candidate subject. The assessment phase returns an *Evaluated QMs Model* reporting the QMs concrete values resulted from the assessment. The *Evaluated QMs Model*, in turn, is the input of the *QMs Visualization* phase during which *Dashboards* representing the QMs status are generated, through code generation.

### 7.2.1 Leveraging Multi-Level Modeling for Multi-Domain Quality Assessment

To allow the application of our approach in multiple domains we decided to exploit MLM to realise the artifacts involved in the development phase of a QES. In Figure 7.2, we sketched the quality assessment just described, by adding the typical annotations coming from MLM hierarchies, to explain how we enabled multi-domain quality assessment.

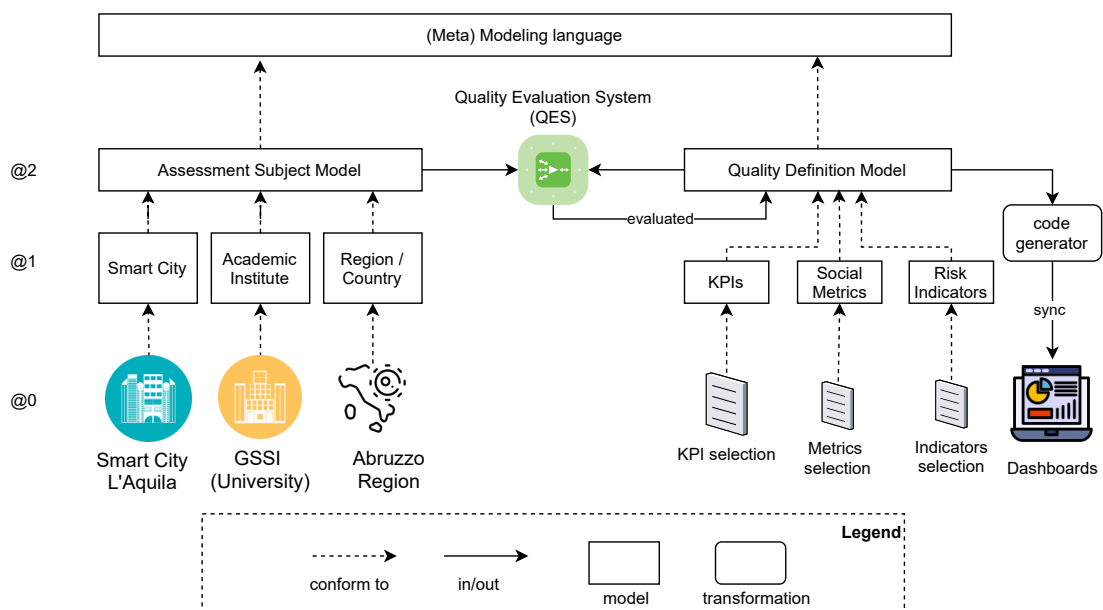


FIGURE 7.2: Overview of the Multi-level Quality Assessment System.

In particular, we sketched the core component of the approach devoted to the *QMs Assessment* as a *QES* that takes in input an *Assessment Subject Model* and a *Quality Definition Model* both conforming to a *(Meta) Modeling language*. Following the MLM

annotations at the left-side of the figure, at level @0 we propose three running examples for subjects and metrics conforming to a specific domain modeled at level @1 that, in turn, have to conform to the *Assessment Subject Model* and the *Quality Definition Model* defined at level @2.

Moreover, to prove the feasibility of the approach in a MLM perspective, we formalised an assessment process as  $A_{req}$ :

$$A_{req}(Subject, Q_m) \rightarrow EQ_m \quad (7.1)$$

where  $A_{req}$  is the assessment request, given a *Subject* and a quality definition expressed as a quality model  $Q_m$ . The output of this request is an evaluated quality model  $EQ_m$ , where the requested quality characteristics are expressed quantitatively. We used this formalisation to prove the feasibility of applying our approach in the three different domains reported in Figure 7.2 that we will use as running examples in the remainder of the chapter.

### 7.2.2 Running Examples

Besides the Smart City KPIs evaluation case, we choose two other running examples for which we identified 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.

**Smart City KPIs evaluation.** Starting from the smart city domain already described in Section 2.1, in the following we report an example of KPI called PM2.5 emissions [13] that we will use to map the KPIs assessment process with a typical QES. This indicator measures the ratio of the emissions of the pollutant to the number of inhabitants in a city. It is calculated as follows:

$$Emissions_{PM2.5} = \frac{PM2.5}{CityPopulation} \quad (7.2)$$

This particular KPI takes in input two parameters (e.g.,  $PM2.5$ ,  $CityPopulation$ ). Input parameters are collected in the representation model of the subject to be evaluated, i.e., the smart city. The calculation formulas for the KPIs selected for the evaluation are described in the quality model passed as input to the evaluation request. From this perspective, we can define a relationship between the smart city KPIs assessment and the assessment process formalised using Definition (7.1) as:

$$instanceOf(SC, Subject) \wedge instanceOf(KPI_m, Q_m) \quad (7.3)$$

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,  $Q_m$ .

**Research Institute Social Impact.** The second chosen running example concerns the context of Higher Education Institutions (HEIs), i.e., universities and research centers, where it is important to measure direct and indirect impact on companies and societies generated by the research and third-mission activities [74–76]. These measurements are done to prove that HEIs may generate positive impacts through a variety of activities, e.g., public engagement and technology transfer. This social impact can be evaluated w.r.t. different dimensions (e.g., innovation, economic, technological, social). Since economic investments are very important for research institutes, the economy impact dimension has a certain relevance in this domain. In particular, it is important to show the positive impact of economic investments. For instance, certain institutes measure social metrics calculating the Sales Volume (SV) after collaborations. Equation (7.4) reports the formula related to the calculation of the SV related to a department  $i$  of an HEI. Where  $SalesAC$  represent the sales after the collaboration and  $SalesBC$  the sales before the collaboration.

$$SV_i = SalesAC_i - SalesBC_i \quad (7.4)$$

It is worth noting here that for this type of institutes such social metrics are monitored per department in order to support decision-making initiatives to improve the performance of the whole institute. Also for this running example we can define a relationship with the Definition (7.1):

$$instanceOf(I, Subject) \wedge instanceOf(SM_m, Q_m) \quad (7.5)$$

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.

**Covid-19 Risk.** The third running example is related with the Covid-19 risk monitoring. Specifically, during the Covid-19 pandemic period, the Italian Ministry of Health drew up 21 indicators [102] to monitor the transmission and impact of the virus in Italy. In particular, every Italian region was measuring these indicators collecting the statistics data coming mostly from the local health system. Depending on the results of the measurements, a level of risk (e.g., red, orange, yellow, green, white) was assigned to every region. The aim of this type of regions monitoring was performed to lighten the burden of the pandemic over hospitals. For instance, one of the indicators calculates the Percentage of Positive Tests (PP) as reported in Equation (7.6), over the total number



of analysed tests, namely *TotalTests*

$$PP = \frac{PositiveTests}{TotalTests} \times 100 \quad (7.6)$$

In this context the monitoring is performed by region that represent the subject of the quality assessment, through the calculation of the 21 indicators that, in turn, compose an instance of the quality model. Thus, also for this running example we can define the relationship between the region and the subject using Definition 7.1:

$$instanceOf(Region, Subject) \wedge instanceOf(RI_m, Q_m) \quad (7.7)$$

where *Region* contains the definition of the input parameters needed to calculate the risk indicators defined in *RI<sub>m</sub>*.

### 7.3 Subjects Definition

Looking at the formalisations for the assessment processes for the running examples just described in equations 7.3, 7.5 and 7.7, we are able to map the detected subjects to the ones defined in Figure 7.2 on the left hand side at level @0. Starting from these three application domains we designed all the artifacts involved in the definition of a subject, i.e., the models describing the specific domains @1 and the model of the *Assessment Subject Model*. In Figure 7.3, we reported the multilevel hierarchy described through the MultEcore notation [103], a modeling tool facilitating MLM on top of EMF, containing the details of the models indicated in Figure 7.2. Please note that they are not complete for sake of readability and understandability. This models hierarchy comes from the abstraction of the smart city metamodel reported in Figure 4.2.

At level @2 we have a model for the definition of **Subject** corresponding to the *Assessment Subject Model* in Figure 7.2. This model allows the definition of a **SubjectUnderEvaluationModel** by adding a **Subject**. Moreover, it allows the description of: (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**).

At the level @1 we found the instantiations of the **SubjectUnderEvaluationModel** for the domains related to the running examples. In every instance the **Subject** is instantiated as a different object, i.e., **SmartCity**, **Institute**, or **Region**, for each of which we have different models of three different domains. These three models at level @1 are reported as excerpts, so we only discuss here the relevant concepts and objects. The concepts belonging

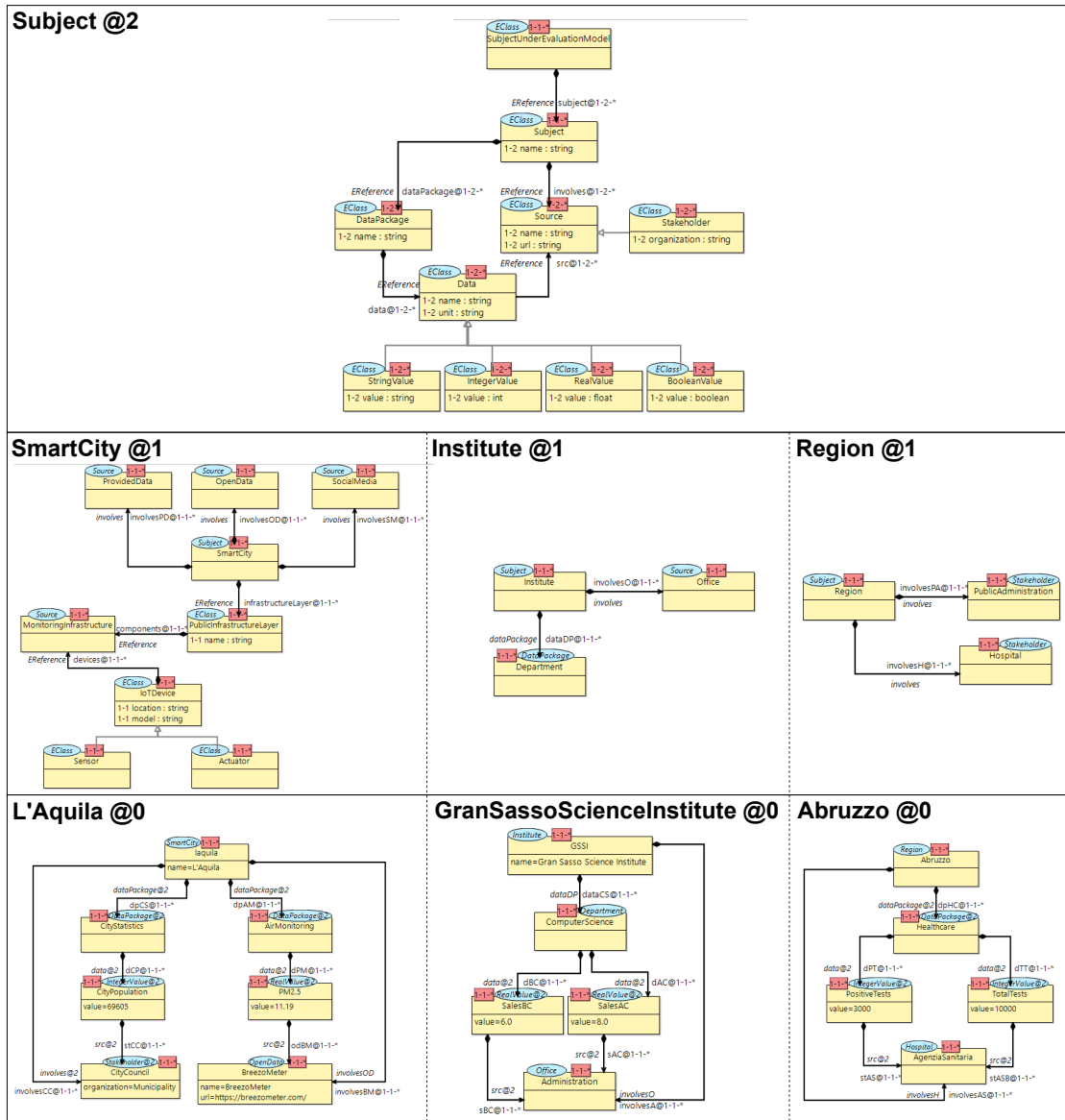


FIGURE 7.3: Multi-level hierarchy for Subjects definition.

to the smart city domain have been already discussed in Section 4.2.1. Concerning the research institute social impact domain, we assume that the the various Departments of the institute are in charge of collecting data through dedicated Offices. Meanwhile, for the regional risk monitoring domain, we assume to have two types of stakeholders, namely the PublicAdministration and Hospitals.

At Level @0 of the models hierarchy we reported the real world objects / systems corresponding to the three running examples in Figure 7.2, namely, the italian city of L'Aquila, the Gran Sasso Science Institute (GSSI), and the Abruzzo region. Please note that the models are partially reported also in this level, in particular in every model definition we reported the concepts needed to design the three specific running examples. It is in this level that we collected the input parameters with their values of the metrics that we want

to calculate for every running example. The first model on the left-hand side represents the specific Smart City of L'Aquila of our running example. In this model we defined two data packages containing two collections of information, namely, `CityStatistics`, `AirMonitoring`. In the first one we allocated the definition of the information indicating the `CityPopulation` and in the second the one about the `PM2.5` emissions. Every data is connected with the definition of its provider, respectively, `CityCouncil` and `BreezoMeter`. In the center of the level @0 the instance of a research institute (model at level @1) is reported, i.e., `GSSI`, that is the subject for the social impact running example. Here, we assume that the information are organized w.r.t. the different departments. For sake of space, here we reported only the instance of the `ComputerScience` department. For this specific department we assume to have two types of data, i.e., `SalesBC`, `SalesAC`. These entities contain the values for the sales volume before and after collaborations, respectively, for the `ComputerScience` department. Here, this type of data is provided by the `Administration` office. On the right, we have the model describing the `Abruzzo` region, that is the subject of the assessment for the last running example. `Abruzzo` is an instance of `Region` that contains information about `Healthcare`. The data in this case are provided by the `AgenziaSanitaria` and concern the detected number of positive COVID tests (i.e., `PositiveTests`) and the total number of tests performed (i.e., `TotalTests`).

## 7.4 Quality Metrics Definition

The artifacts reported on the right hand side side in Figure 7.2 are the ones used to define the other input of an evaluation request, i.e., the quality model. In Figure 7.4, the models hierarchy used to design the quality metrics definition artifacts is depicted. This models hierarchy comes from the abstraction of the KPIs metamodel reported in Figure 4.3. In particular, at level @2 we designed the abstract language to define quality models, level @1 allows the definition of different types of models for evaluating different aspects of quality w.r.t. the subject/domain that has to be evaluated, e.g., KPIs, social metrics and risk indicators. At level @0 are located the models instances of @1, specifically a selection of KPIs as well as metrics for social impacts and risk indicators. This level allows the definition of models containing all the possible metrics that can be evaluated in an application domain. This may support the definition of an additional level for the selection of a specific set of 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.

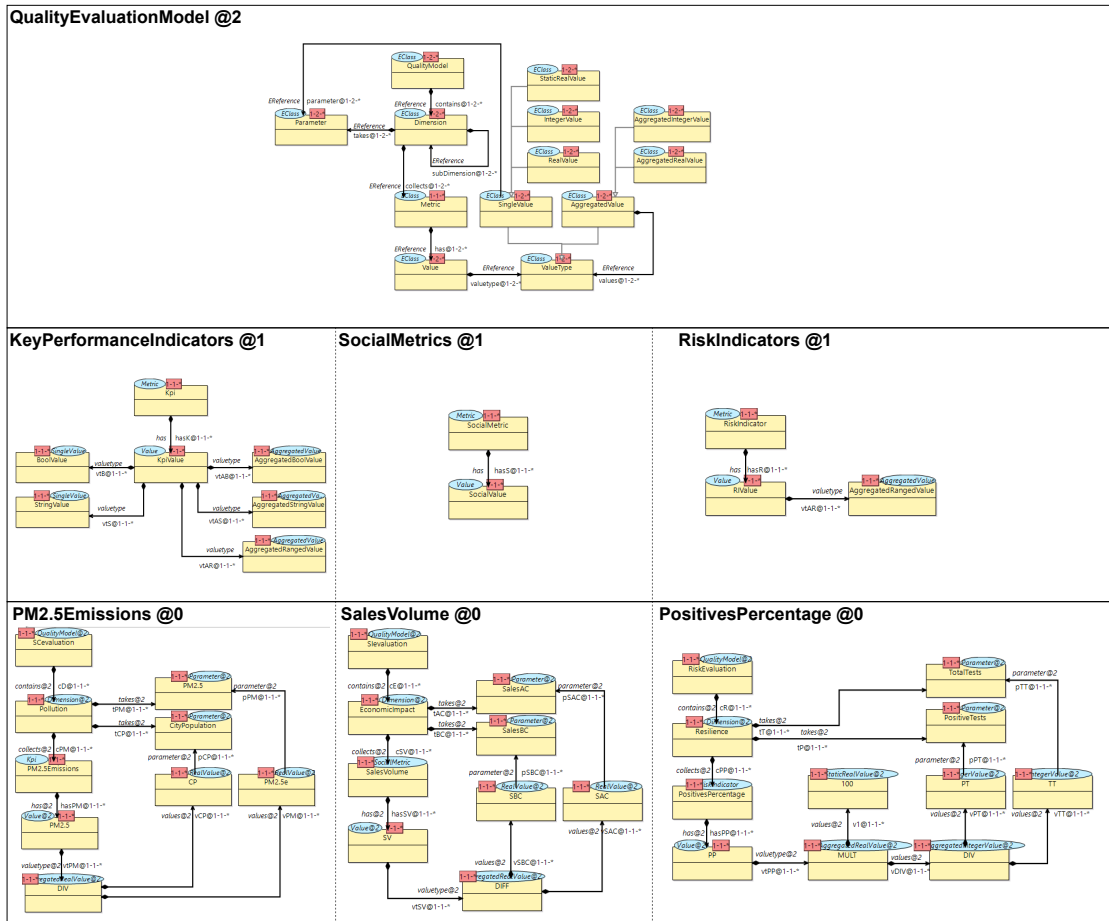


FIGURE 7.4: Multi-level hierarchy for Quality Metrics definition.

In detail, at level @2 the model allows the organization of the Metrics and the input Parameters needed in the calculations thanks to the Dimension type. 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, we find the instances of the described model, one for each domain of the three running examples. In particular, for the smart city domain we designed a KeyPerformanceIndicators model where we instantiated the Kpi type and added other instantiations of single and aggregated values (i.e., BoolValue, StringValue, AggregatedBoolValue, AggregatedStringValue, AggregatedRangedValue), that we described in Section 4.2.2. Also in the other two models at level @1 (i.e., SocialMetrics, RiskIndicators) we specified which types of metrics are used in the evaluation in the different domains.

At level @0, the instances of the models at level @1 report the definition of a metric for running example. In particular, the `PM2.5Emissions` model describes how the KPI is calculated and which parameters it needs (see Equation (7.2)). This particular KPI is associated to the dimension `Pollution`, and to the parameters `PM2.5` and `CityPopulation`. To define the calculation of this KPI, we defined an aggregated real value `DIV` associated to the value of the parameters `PM2.5e` and `CP`. This structure is repeated also in the definition of the considered metrics in the other two models (i.e., `SalesVolume`, `Positives-Percentage`). It is worth noting that for the description of the formulas used to calculate the metrics is exploited aggregated values type that is used for specifying operations and parameters. For instance, for the described `PM2.5` emissions KPI, the formula requires a division (`DIV`) between the detected value for the emissions of `PM2.5` and the number of city inhabitants, as reported in Equation (7.2).

Thanks to the multi level modeling perspective both hierarchy models (i.e., subject and quality metrics definitions) can be easily extended by means of a single modeling step, e.g., adding other instantiations of the `Data` or `Source` elements, in the subject definition at level @1, in case of the advent of a new type of data or data source, respectively; adding other instantiations of the `SingleValue` element in the quality metrics definition at level @1 to include another data type.

## 7.5 QES engine

The models at level @0 of both hierarchies in Figures 7.3 and 7.4 are the input artifacts of the implemented QES depicted in the center of Figure 7.2. The engine explores also the hierarchical organization vertically by traversing the models. As done for the *MIKADO* evaluation engine (see Section 5.2.2), we implemented the QES as a model interpreter that reads 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. Moreover, the calculation engine incorporated in the QES parses the requests and actualizes the results that are based on the metrics defined in the quality model with quantitative evaluations.

In Listing 7.1 we reported few lines of the QES implemented in Java plus EOL [104].

```

1 var root=qm0!EClass.all.selectOne(c|c.name="Root");
2 var qm1package= 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=qm1package);
5 var metricclass= mm.eClassifiers.selectOne(c|c.eAnnotations.source.
6     flatten().includes("type=Metric"));
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 7.1: Snippet of the EOL-based implementation of the QES Engine.

We implemented a Java launcher that invokes the script and defines the input parameters of the QES, i.e., the three models of the three levels @0 – @2 of both side of Figure 7.2, for a total of 6 models. MultEcore allows the navigation of the models through textual annotation. For, instance, the object Root is a format that MultEcore uses as first instance containing all the objects in the model. Indeed, the quality model is loaded by its Root at line 1. As regard this script, we use some variables referring to the level in the model's name (e.g., *qm0*, *qm1*) to navigate the input models belonging to different levels. For instance, the expression *qm0!EClass.all* refers to all the objects instantiated at level @0 and so on. The loaded quality model contains the definition of the KPIs or the SocialMetrics or the RiskIndicators, e.g., PM2.5Emissions, or SalesVolume or PositivesPercentage (see Figure 7.4—models of level @0). At line 2 the package name used by the model at level @0 is loaded. Meanwhile, at line 3 the operation of selection retrieves the model at level @1 as a conformance retrieval between the two levels. Since MultEcore uses String-based annotations for dynamic typing, in this case, the conformance relationship is persisted without a strongly typed relation like in two-level modeling. Indeed, at lines 4 and 5, we retrieve all the clobjects instantiating the Metrics at level @1. This operation returns the clobjects for KPIs, RiskIndicators and SocialMetrics, and whatever is defined at @1 as instance of metric. Then, at lines 7–10 the calculation for the metrics on the subject are applied. After the launch of the script on the models, we receive as output a message in the console as the one reported on top of Figure 7.5. The console log proves the dynamic binding of the defined KPIs as instances of Metrics. The same happens when passing the other models at level @0 in Figure 7.4. The operation *getValue* actualizes the value by calculating the defined operation in the given quality model. This actualization of the result can be inspected directly in the model by looking at specific values of the metric (see *Value* class in Figure 7.4).

## 7.6 Assessment Results Graphical Representation

As already anticipated in Section 7.2 the results of the assessment are reported in graphical dashboards that are based on specific model to code transformation, generating HTML pages with their embedded Javascript files, as described for the *MIKADO* data visualization component in Section 5.2.3. The synchronization happens with the instantiated quality metrics models belonging to the @0 level of the hierarchy (Figure 7.4). In this way, every time an assessment is performed, the dashboards related to the corresponding subject are automatically updated.

To allow decision-making reasoning, every **Metric** is equipped with an attribute called **targetvalue** that provides the optimum/desired value for the corresponding metric. Indeed, these values are used also in the graphical representation in order to show how the subject is performing w.r.t. a specific metric's target value. In the following, we show some excerpts of the generated dashboards with the graphical charts representing the results of the assessments for the three running examples. In particular, the first running example has been applied on the city of L'Aquila, as drafted in the model in Figure 7.3 at level @0. The result, reported in the generated gauge (part of the generated dashboard) in Figure 7.5, shows the KPI *PM2.5 emissions* and the result of the measurement for the given smart city, i.e., 32%. In Figure 7.6, we reported the result of the generated

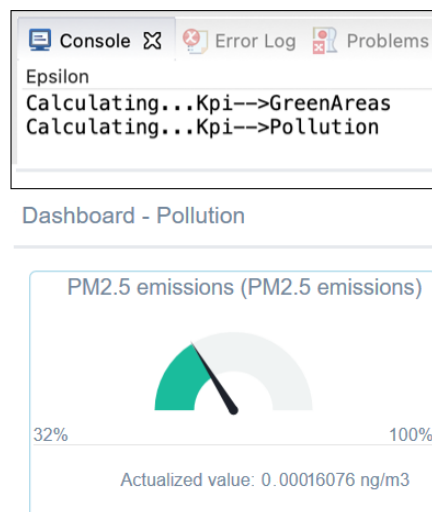


FIGURE 7.5: Excerpt of the Dashboard evaluating Smart City KPIs.

dashboard for the social impact scenario belonging to the second running example, in which we assessed the GSSI (Gran Sasso Science Institute), Computer Science department, showing that we had an increase of 50% of the sales volume after collaboration. Eventually, in Figure 7.7, the generated gauge shows the evaluation of the COVID risk applied on the Abruzzo region as third running example. The percentage of positives resulted 30% in that specific time of the assessment. These gauges are part of more

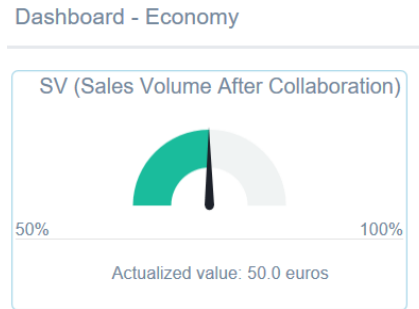


FIGURE 7.6: Excerpt of the Dashboard evaluating Social Impact.

complex dashboards that are automatically generated in sync with the assessed subject model.

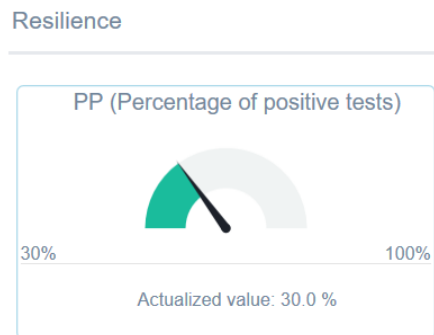


FIGURE 7.7: Excerpt of the Dashboard evaluating COVID Risk.

It is worth noting that, although the three presented running examples are quite trivial and represent fictional scenarios, in the context of a realistic evaluation with real data they may contribute in giving a global view about the performance of the evaluated subjects in different applications (e.g., smart cities ranking, pandemic spreading). For instance, in the context of the first running example, the calculation of the PM2.5 emissions KPI on top of multiple smart cities would allow them to be ranked in terms of pollution. Moreover, measuring the sales volumes after a collaboration of multiple departments of a research institute could be helpful for future investments in external collaborations. This would be of support for the departments, as well as for the institute itself, to understand their generated impact on the society and knowing where and how to invest would improve their contribution to the community. Eventually, the calculation of the percentage of positive tests over multiple regions allowed the monitoring of the pandemic spread all over the country.



## **7.7 Discussion**

This Chapter showed how exploiting MLM allowed us to apply our assessment framework in different domains from smart cities. Although, the subject of the assessment and the required measurements were different, they could have been all defined with a multi-level approach. To this aim, we proposed other two domains upon which apply the abstracted framework, i.e., research institute social impact and covid-19 risk. To model the new domains scenarios, we proposed two multi-level models hierarchy, devoted to the definition of subjects under evaluations and quality metrics. Then, even a refactoring in the evaluation engine has been developed in order to enable the navigation of multi-level hierarchies of models. Eventually, we reported the graphical representation of the assessment results of the three scenarios taken as running examples from different domains.

Notwithstanding the proven generalizability of the proposed approach, in the next Chapter we draw some conclusions and we report further extensions of the framework on which we are already working.

## Chapter 8

# Conclusions and Future Work

This dissertation has been developed in the context of smart cities assessment. Consequently, we inspected the actual state of the art on the different aspects relating this context, as discussed in Chapter 3. In particular, we start by analysing the existing KPIs assessment frameworks, from the perspectives of the process that they implement and the requirements they should address. Moreover, we give an overview on quality evaluation systems, by highlighting their limitations and potentialities. Lastly, we performed an analysis about existing approaches dealing with smart cities modeling. From the analysis of the state of the art we extracted the main research challenges that led the work presented in this thesis. Specifically, the identified challenges deal with the need of a uniform way of modeling smart cities and their complexity, to support the need of a systematic methodology allowing smart cities to *define*, *measure* and *visualize* the KPIs of interest in order to efficiently assisting the decision-making processes. Such methodology should also support (i) *KPIs evolution* over time w.r.t. the evolution of smart cities needs and contexts, and (ii) *KPIs customization* based on the specific smart city under evaluation, in terms of selection of appropriate KPIs to be calculated. Lastly, we highlighted the need to make the systematic assessment methodology *generalizable* such that to be applied in the quality assessment of subjects coming from different domains than the smart cities. To face the extracted research challenges, we proposed a model-based solution for smart cities KPIs assessment, namely *MIKADO*. It provides modeling artifacts for the definition of smart cities and KPIs in a uniform way, an evaluation engine for the automatic KPIs calculation and an engine to render their visualization (see Chapter 4). Moreover, we showed how the architecture behind the developed framework is flexible enough to allow different deployment patterns, i.e., standalone, hybrid and online specifications, and technologies (see Chapter 5). Indeed, by following a hybrid deployment pattern we extended *MIKADO* in a service-oriented fashion to enable KPIs data sources continuous monitoring and runtime up-to-date models (see Section 5.3.1). The

overall approach has been validated (see Chapter 6) from different perspectives. First, the feasibility of the approach by applying it on a demonstration case (see Section 6.1). Moreover, we assessed the understandability of the smart cities and KPIs DSLs through an experts' survey (see Section 6.2). Furthermore, we evaluated the performance of the KPIs calculations execution time w.r.t. scenarios of increasing complexity (see Section 6.3). Lastly, also the service-based continuous KPIs assessment extension has been validated and compared with the standalone version (see Section 5.3.1). The results of the various evaluations prove not only the feasibility of the approach and its ability of supporting smart cities comparisons but also a high understandability of both the smart city and KPIs definition DSL. Furthermore, concerning the performance of the evaluation engine's execution time, the obtained results highlighted (i) the efficiency of the evaluation engine execution time, since in every experiment it showed a linear increase; (ii) promising scalability results, since the system took approximately 5 seconds to assess 800 KPIs over 10 smart cities. Also for the service-based KPIs assessment deployment of the framework we saw how it enables a continuous KPIs evaluation, by drastically decreasing ( $\sim 88\%$ ) the latency contributed by the data retrieving and models update phases, compared to the baseline framework. Eventually, looking at the commonalities between the KPIs assessment process implemented by the *MIKADO* framework, and a typical quality evaluation process, namely an input subject to be assessed and a set of quality metrics to calculate over it, we decided to abstract the developed KPIs assessment framework. Specifically, we abstracted *MIKADO* by exploiting MLM into a QES applicable in different domains from smart cities (see Chapter 7), by further proving the domain-independence of our approach. Thus, we showed the feasibility of the MLM approach by applying it in the assessment of *research institutes social impact* and *Covid-19 risk*.

In summary, we report below how the developed model-based approach devoted to KPIs assessment over smart cities is able to answer to the research questions coming from the detected challenges of the state of the art and stated in Chapter 1.

**RQ1:** *How smart cities are modeled in the literature?*

To answer this RQ, we performed an analysis of the state of the art about smart cities modeling from which it resulted that most of the modeling approaches only cover specific aspects of the smart city domain without providing enough abstraction to model all the smart city dimensions. Consequently, a prominent model kind did not emerge. Moreover, we found relevant application fields not covered (e.g., stakeholders inclusion and communication). Lastly, we found out that the maturity level of the analysed contributions appeared to be low and a scientific reference community is not emerging. Thus, being aware of the fact that no standards to

describe smart cities exist, we defined a new uniform way of modeling smart cities thanks to a DSL provided both in a textual and graphical semantics. Specifically, the smart cities DSL we defined, from one side focuses mainly on those aspects relevant for the KPIs assessment, while from the other side it is easily extensible to add other aspects required for managing the heterogeneity of smart cities. In this way, we allow multiple types of users to model a smart city by using an intuitive language which is also extensible, re-usable and machine readable.

**RQ2:** *How can MDE help in the uniform modeling and automatic assessment of smart cities?*

To answer this RQ besides the analysis of approaches to model smart cities and the consequent definition of a DSL, we further analysed KPIs guidelines and documentations. From this analysis we extracted the main structures and concepts used to describe KPIs. Thanks to the power of abstraction of MDE, we defined an abstract and textual DSL for KPIs definitions which appeared to be understandable by experts with different backgrounds. Moreover, on top of these modeling artifacts we developed a model interpreter that automatically performs the defined KPIs calculations. In this way, MDE allowed us to develop a model-based KPIs assessment framework in which the subject to be evaluated and the metrics to assess are modeled separately, guaranteeing the right level of SoC. Furthermore, a code generation process implemented the translation of the evaluated KPIs models into dynamic graphical dashboards. Specifically, M2T transformations enable the accessibility of modeling artifacts to a wide range of users. In our case, we exploited model transformations to dynamically generate intuitive graphical dashboard to support the KPIs assessment reporting. Still in the MDE technical space, the models@runtime paradigm can be exploited to enable continuous monitoring of data sources providing input parameters needed for KPIs calculations. In this way, models would be constantly updated with up-to-date parameters values and, consequently, with up-to-date assessed KPIs values. Eventually, also with a view to future extensions of the *MIKADO* framework to bring it online, we have seen that there exist MDE tools (e.g., Eugenia Live<sup>1</sup>, Xtext<sup>2</sup>, Epsilon Live<sup>3</sup>) allowing to integrate modeling editors on the web browsers.

**RQ2.1:** *How can we efficiently monitor KPIs over time?*

To answer this RQ, we inspected the literature to understand how KPIs evolve over time. On top of this inspection, we defined both the smart city and KPIs DSLs. The definition of a systematic and uniform way of modeling through abstract DSLs allowed us to support KPIs evolution, such as the advent of

<sup>1</sup> <https://github.com/louismrose/eugenia-live>

<sup>2</sup> <https://www.eclipse.org/Xtext/>

<sup>3</sup> <https://github.com/epsilon-labs/playground/>

new parameters or the change of the calculation logic, with few modeling steps in the smart city and/or KPIs models.

**RQ2.2:** *How can we adequately monitor KPIs for different smart cities?*

To answer this RQ, thanks to the abstract DSL, we can define a unique KPIs model, valid for different smart cities. This does not mean that the KPIs model is static. On the contrary, it is unique but not univocal 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. Moreover, we showed how to apply the KPIs assessment framework also to other cities than the one proposed as demonstration case, proving, not only, the flexibility of the approach, but also, its ability of enabling smart cities comparisons.

**RQ3:** *To what extent does a model-based smart cities KPIs assessment approach can be generalized to other subjects of evaluation belonging to different domains?*

To answer this RQ, we inspected which are the commonalities between the developed model-based KPIs assessment process and a typical quality evaluation process, namely they share the overall objective of assessing a given subject passed as input, where the assessment is executed on top of a quality definition. Thus, we abstracted *MIKADO* by transforming it in a domain-independent QES through MLM and we further showed that it is generalizable to other subject of evaluation belonging to different domains. To this aim, we applied the multi-level framework to three running examples, i.e., the smart city KPIs evaluation, the research institute social impact, and Covid-19 risk assessment.

## 8.1 Future Work

Although the presented approach already showed promising evaluation results, we discuss in the following interesting lines for future work dealing with open issues we want to face in the future. First, since smart cities benchmarking is often used to produce rankings, we plan to further extend *MIKADO* with specific features enabling *Smart Cities Ranking*. The generated graphical dashboards can be used to enable comparisons between different smart cities. Thus, we want to integrate graphical functionalities to further allow smart cities rankings showing their level of *smartness* by defining the same set of KPIs against which to assess a set of smart cities. Moreover, since the modeling tools composing the proposed approach can also be used by the smart governance to generate predictive and descriptive models of the city, we want to exploit the framework also for simulation activities like, for instance, to evaluate the impact of planned changes on the KPIs

for the city. Thus, we plan to add features enabling not only *Simulations* but also to observe *KPIs Interrelations*. Eventually, besides the comparison with other cities, also the comparison with values related to different periods of time on the same city would be interesting. In this way, an overview of the trends of the different KPIs calculated over the city would be available for consultations. This could be further supported, for instance, by integrating some existing tools to support versioning of *Historical Smart Cities and KPIs Models*, e.g., TemporalEMF [105], in order to monitor the evolving indicators.

Besides this planned future work, we are already working on some extensions of the *MIKADO* framework to face its limitations, such as its dependency on the Eclipse IDE and the lack of traceability of ethical concerns in KPIs assessment processes. In the following sub-sections, we give more details about the extensions on which we are working.

### 8.1.1 Leveraging Models@Runtime in a Digital Twin perspective

In this section we report an extension of the proposed framework on which we are currently working, representing the online deployment of the flexible architecture presented in Section 5.1, i.e., where every component is deployed online. We decided to move forward an online deployment of *MIKADO* to overcome the typical limitations of a standalone implementation, such as the need for constant updates, implying the involvement of the user in the download and check of new versions of the entire bundle every time a new update is released. Furthermore, the Eclipse platform can be counter-intuitive for non-expert users using it for the first time due to possible issues with the host OS and the setup and settings. Moreover, we decided to deploy an online version of *MIKADO* in a digital twin perspective. A digital twin (DT) is a virtual model designed to reflect a physical entity, that may be used to represent a subject system for multiple purposes [106]. The concept of digital twins has been also applied to smart cities, coining the term "Urban Digital Twin" [107]. Indeed, smart cities continuously produce data that can be used as real-time feeding of the digital twin representation which can be the subject of our KPIs evaluation framework. Thus, we want to exploit distributed runtime models to implement and deploy a service-based quality evaluation system for smart cities represented as urban digital twins. To this aim, our plan is to further extend the smart cities assessment tool presented in this dissertation, by exploiting and combining models@runtime and urban digital twins. Specifically, we want to develop an evolution of the KPIs assessment framework based on (micro)services, i.e., *MIKADO@runtime*. We use the microservices term when referring to the distributed deployment of the new tool, where each provided service implements one single functionality. Furthermore, in last few years multiple cloud-based solutions have been proposed to avoid client-side

installations, e.g., Eclipse Theia<sup>4</sup>, Epsilon Playground<sup>5</sup>. These technologies also avoid possible compatibility problems and support sustainability, with an efficient use of distributed resources [108]. It is worth noting that the existing digital twins platforms are mostly business-oriented solutions, proprietary, and general purpose not offering support for modeling specific domains, e.g., smart cities. Basically, we are working on the realization of an Urban DT Low-Code Development Platform (LCPD) exploiting the paradigm of models@runtime and (micro)services for a quality evaluation system for smart cities represented as urban digital twins. The refactored architecture will allow for efficient urban digital twins through continuous evolution of the twins without the need for redeployment.

### 8.1.2 Supporting Ethics Risk Traceability in Quality Assessment

In this section we report another extension on which we are currently working dealing with the traceability of ethical concerns in KPIs assessment processes. The motivations behind this extension rely on the fact that ethical risks in KPIs evaluation are often hidden in the KPIs definition and experts will get the results without considering them. Moreover, detect ethical risks in an evaluated KPI is not trivial since ethical issues may be not directly linked to the KPI itself. To the contrary, they may be generated from the parameters used to calculate the KPI and/or from the sources of these parameters. Unintentionally, a non-expert operator can run into malicious sources or simply lack the necessary knowledge to identify ethical issues about used data. Specifically, the KPIs are calculated by assembling and aggregating input parameters retrieved from data sources that may suffer from both *trustworthiness* and *ethical* issues as the concerns of *security*, *human privacy*, and *discrimination*. For instance, the KPI Air Pollution (see Section 6.1) may give rise to human security concerns, since the collection of air quality data (i.e., *pollutants measured concentrations*) from malicious data sources may wrongly influence people about the choice of going to certain areas of the city [109], putting their health at risk. Procedures and standards, as well as policies and decisions that could represent a perceived violation of an ethical value should be highlighted when KPIs results are proposed to stakeholders. Often KPIs experts are not considering ethical aspects, when defining the KPIs, also because the role and expertise of a KPIs expert may not coincide with Ethical background.

With these premises, we highlight the importance of involving *Ethics Experts* in the KPIs assessment process of smart cities. This is essential because of both the ethical issues that may arise, and the credibility issues that the governance of a smart city may consequently occur, in the context of a transparent government. To this aim, we want

<sup>4</sup> <https://theia-ide.org/> <sup>5</sup> <https://www.eclipse.org/epsilon/live/>

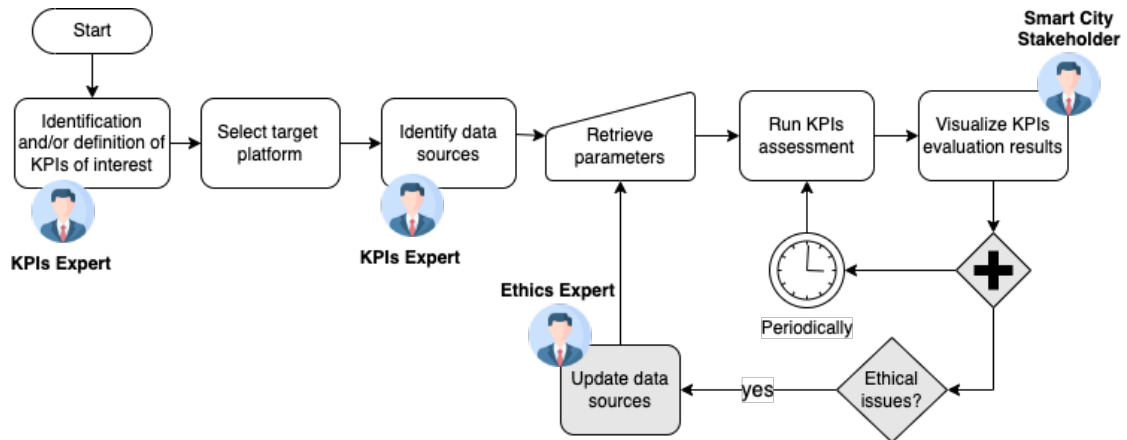


FIGURE 8.1: KPIs assessment process and involved roles.

to extend *MIKADO* to *alert* the involved stakeholders when the evaluated KPIs may suffer from ethical problems, through specific highlights in the automatically generated dashboards. To better support this process and overcome these limitations, we want to develop a traceability mechanism that can help to understand better KPIs which can be affected by ethical aspects. This could help the KPIs expert to re-design the KPI definition and data sources and to better position the result of a KPI when highlighted with ethical risk. In particular, we want to extend the KPIs assessment process by including an analysis of selected KPIs, used parameters, and data sources, performed by an ethics expert and concerning ethical and trustworthiness issues. The conceptual representation of our solution, including all the involved roles, is sketched in Figure 8.1. Here, we can see how we plan to extend *MIKADO* with a sub-process (in grey in Figure 8.1) enabling the highlighting of potential ethical issues corresponding to the evaluated KPIs. This additional analysis involves an *Ethics Expert* and it triggers, in turn, an iteration of the KPIs assessment by exploiting new data sources for the parameters retrieval, with the final objective of removing possibly any ethical issues associated with the considered KPIs.



# Bibliography

- [1] Maria Teresa Rossi, Martina De Sanctis, Ludovico Iovino, and Manuel Wimmer. A systematic mapping study on smart cities modeling approaches. *Software & Systems Modeling*, 2022. Submitted.
- [2] Martina De Sanctis, Ludovico Iovino, Maria Teresa Rossi, and Manuel Wimmer. MIKADO – A Smart City KPIs Assessment Modeling Framework. *Software & Systems Modeling*, 2021. URL <https://link.springer.com/article/10.1007/s10270-021-00907-9>.
- [3] Martina De Sanctis, Ludovico Iovino, Maria Teresa Rossi, and Manuel Wimmer. A flexible architecture for the key performance indicators assessment in smart cities. In *Proceedings of the 14th European Conference on Software Architecture (ECSA)*, 2020.
- [4] Francesco Basciani, Maria Teresa Rossi, and Martina De Sanctis. Supporting smart cities modeling with graphical and textual editors. In *Proceedings of the 1st International Workshop on Modeling Smart Cities (MoSC)*, 2020.
- [5] Martina De Sanctis, Ludovico Iovino, Maria Teresa Rossi, and Manuel Wimmer. Weaving open services with runtime models for continuous smart cities kpis assessment. In *The 19th International Conference on Service-Oriented Computing, ICSOC*, 2021.
- [6] Maria Teresa Rossi, Martina Dal Molin, Ludovico Iovino, Martina De Sanctis, and Manuel Wimmer. Leveraging multi-level modeling for multi-domain quality assessment. In *MULTI 2021, The 8th International Workshop on Multi-Level Modelling @MODELS 2021*, 2021.
- [7] 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>.

- [8] Juan de Lara and Esther Guerra. Refactoring multi-level models. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 27:1 – 56, 2018.
- [9] Umberto Rosati and Sergio Conti. What is a smart city project? an urban model or a corporate business plan? *Procedia - Social and Behavioral Sciences*, 223: 968 – 973, 2016. doi: <https://doi.org/10.1016/j.sbspro.2016.05.332>. URL <http://www.sciencedirect.com/science/article/pii/S1877042816304128>.
- [10] European Commission. Europe 2020 A European strategy for smart, sustainable and inclusive growth, March 2010.
- [11] Catarina Selada, Carla Silva and Ana Luísa Almeida 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](https://pocacito.eu/sites/default/files/POCACITO_PolicyBrief_No-5_Urban_Indicators_1612.pdf).
- [12] 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>.
- [13] Peter Bosch, Sophie Jongeneel, Vera Rovers, Hans-Martin Neumann, Miimu Airaksinen, and Aapo Huovila. Citykeys indicators for smart city projects and smart cities, 2017. Available at: <https://nws.eurocities.eu/MediaShell/media/CITYkeystheindicators.pdf>.
- [14] Saverio Romeo, Mario Di Gregorio, Ubaldo Alfonso, Anna Tozzi, Francesco Tarquini, and Federica Tomassoni. Digital cities challenge - assessment report for the city of l'aquila, 2019. Available at: <https://bit.ly/32YBgiC>.
- [15] Motoei Azuma. Software products evaluation system: quality models, metrics and processes—International Standards and Japanese practice. *Information and Software Technology*, 38(3):145–154, 1996. doi: [https://doi.org/10.1016/0950-5849\(95\)01069-6](https://doi.org/10.1016/0950-5849(95)01069-6).
- [16] Minako Hara, Tomomi Nagao, Shinsuke Hanno, and Jiro Nakamura. New key performance indicators for a smart sustainable city. *Sustainability*, 8(3):206, 2016.
- [17] Markus Luckey, Martin Erwig, and Gregor Engels. Systematic evolution of model-based spreadsheet applications. *Journal of Visual Languages & Computing*, 23: 267–286, 10 2012.
- [18] Enrico Ferro, Brunella Caroleo, Maurizio Leo, Michele Osella, and Elisa Pautasso. The Role of ICT in Smart City Governance. In *Proceedings of the International Conference for e-Democracy and Open Government (CeDEM)*, 2013.

- [19] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice, Second Edition*. Morgan & Claypool Publishers, 2017.
- [20] Francisca Rosique, Fernando Losilla, and Juan Angel Pastor. A domain specific language for smart cities. In *Proceedings of the 4th International Electronic Conference on Sensors and Applications*, 2018.
- [21] Mohammad Abu-Matar. Towards a software defined reference architecture for smart city ecosystems. In *Proceedings of the IEEE International Smart Cities Conference (ISC2)*, pages 1–6, 2016.
- [22] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.
- [23] Fernando Macías. Multilevel modelling and domain-specific languages, 2020. URL <https://arxiv.org/abs/1910.03313>.
- [24] ITU-T Focus Group on Smart Sustainable Cities. Smart sustainable cities: An analysis of definitions, 2014. Available at: <https://bit.ly/324U929>.
- [25] Taewoo Nam and Theresa A. Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, pages 282–291, 2011.
- [26] Boyd Cohen. Methodology for 2014 smart cities benchmarking, 2014. URL <https://www.fastcompany.com/3038818/the-smartest-cities-in-the-world-2015-methodology>.
- [27] Ulrike Gretzel, Marianna Sigala, Zheng Xiang, and Chulmo Koo. Smart tourism: foundations and developments. *Electronic Markets*, 25, 08 2015. doi: 10.1007/s12525-015-0196-8.
- [28] Antoni Ballesté, Pablo Pérez-Martínez, and Agusti Solanas. The pursuit of citizens’ privacy: A privacy-aware smart city is possible. *IEEE Communications Magazine*, 51, 06 2013. doi: 10.1109/MCOM.2013.6525606.
- [29] Julien Carbonnell. Smart-city: Stakeholders roles and needs., 2019. URL <https://julienarbonnell.medium.com/smart-city-stakeholders-roles-and-needs-8e3679764d2a>.
- [30] Hans Jochen Scholl and Margit Christa Scholl. Smart governance: A roadmap for research and practice. In *iConference 2014 Proceedings*, pages 163—176, 2014.
- [31] Gabriela Vialea Pereira, Petera Parycek, Enzoc Falco, and Reinoutc Kleinhans. Smart governance in the context of smart cities: A literature review. *Information Polity*, 23(2):143–162, 2018.

- [32] Dewi Mutiara, Siti Yuniarti, and Bambang Pratama. Smart governance for smart city. *IOP Conf. Series: Earth and Environmental Science*, 126:012–073, 2018.
- [33] Nuno Vasco Lopes. Smart governance: A key factor for smart cities implementation. In *2017 IEEE International Conference on Smart Grid and Smart Cities (ICSGSC)*, pages 277–282, July 2017.
- [34] Minako Hara, Tomomi Nagao, Shinsuke Hanno, and Shinsuke Nakamura. New Key Performance Indicators for a Smart Sustainable City. *Sustainability*, 206(8), 2016.
- [35] José P. Miguel, David Mauricio, and Glen Rodríguez. A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications*, 5(6):31–53, nov 2014. doi: 10.5121/ijsea.2014.5603. URL <https://doi.org/10.5121/ijsea.2014.5603>.
- [36] Oleksandr Gordieiev, Vyacheslav Kharchenko, Nataliia Fominykh, and Vladimir 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)*, pages 223–232. Springer, 2014.
- [37] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.
- [38] Douglas C. Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY*, 2006.
- [39] Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.
- [40] Dennis Wagelaar. Composition techniques for rule-based model transformation languages. In *International Conference on Theory and Practice of Model Transformations*, pages 152–167. Springer, 2008.
- [41] Jesús Sánchez Cuadrado and Jesús García Molina. Approaches for model transformation reuse: Factorization and composition. In *International Conference on Theory and Practice of Model Transformations*, pages 168–182. Springer, 2008.
- [42] S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *IEEE Software*, 20(5):42–45, 2003. doi: 10.1109/MS.2003.1231150.

- [43] Deniz Cetinkaya and Alexander Verbraeck. Metamodeling and model transformations in modeling and simulation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 3043–3053, 2011. doi: 10.1109/WSC.2011.6148005.
- [44] Margarida Afonso, Regis Vogel, and Jose Teixeira. From code centric to model centric software engineering: practical case study of mdd infusion in a systems integration company. In *Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MBD-MOMPES'06)*, pages 10 pp.–134, 2006.
- [45] Juan De Lara, Esther Guerra, and Jesús Sánchez Cuadrado. When and how to use multilevel modelling. *ACM Trans. Softw. Eng. Methodol.*, 24(2), 2014. ISSN 1049-331X. doi: 10.1145/2685615.
- [46] Colin Atkinson and Thomas Kühne. Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359, Jul 2008. ISSN 1619-1374. doi: 10.1007/s10270-007-0061-0. URL <https://doi.org/10.1007/s10270-007-0061-0>.
- [47] J De Lara, E Guerra, and Jesús Sánchez Cuadrado. When and how to use multilevel modelling. *ACM Transactions on Software Engineering and Methodology*, 24:1–46, 2014.
- [48] Colin Atkinson and Thomas Kühne. The essence of multilevel metamodeling. In Martin Gogolla and Cris Kobryn, editors, *Proceedings of the International Conference on the Unified Modeling Language (UML)*, pages 19–33. Springer, 2001.
- [49] Nelly Bencomo, Robert B France, Betty HC Cheng, and Uwe Aßmann. *Models@run. time: foundations, applications, and roadmaps*, volume 8378. Springer, 2014.
- [50] Hui Song, Gang Huang, Franck Chauvel, and Yanshun Sun. Applying MDE Tools at Runtime: Experiments upon Runtime Models. In Nelly Becomo, Gordon Blair, and Franck Fleurey, editors, *Proceedings of the 5th International Workshop on Models at Run Time*, Oslo, Norway, October 2010. URL <https://hal.inria.fr/inria-00560785>. to be published.
- [51] Nicolas Ferry and Arnor Solberg. *Models@Runtime for Continuous Design and Deployment*, pages 81–94. 12 2017. ISBN 978-3-319-46030-7. doi: 10.1007/978-3-319-46031-4\_9.
- [52] Microsoft Research. Key Performance Indicators (KPIs) in Power Pivot, 2019. <https://bit.ly/37EFR9r>.

- [53] ITU-T Focus Group on Smart Sustainable Cities. Key performance indicators related to the sustainability impacts of information and communication technology in smart sustainable cities, March 2015.
- [54] Angeliki Kylili, Paris A. Fokaides, and Petra Amparo Lopez Jimenez. Key performance indicators (kpis) approach in buildings renovation for the sustainability of the built environment: A review. *Renewable and Sustainable Energy Reviews*, 56: 906–915, 2016. ISSN 1364-0321. doi: <https://doi.org/10.1016/j.rser.2015.11.096>. URL <https://www.sciencedirect.com/science/article/pii/S1364032115013635>.
- [55] Patrick Hester, Barry Ezell, Andrew Collins, John Horst, and Kaleen Lawsure. A method for key performance indicator assessment in manufacturing organizations. *International Journal of Operations Research*, 14:157–167, 11 2017.
- [56] Dhanya Jothimani and S.P. Sarmah. Supply chain performance measurement of third party logistics. *Benchmarking An International Journal*, 21:944–963, 09 2014. doi: 10.1108/BIJ-09-2012-0064.
- [57] Rui Abreu, Jácome Cunha, João Fernandes, Pedro Martins, Alexandre Perez, and Joao Saraiva. Smelling Faults in Spreadsheets. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME)*, pages 111–120, 2014.
- [58] Dietmar Jannach, Thomas Schmitz, and Konstantin Schekotihin. Toward interactive spreadsheet debugging. In *Proceedings of the First Workshop on Software Engineering Methods in Spreadsheets*, pages 3–6, 2014.
- [59] Jácome Cunha, João Fernandes, Jorge Mendes, and Joao Saraiva. Embedding, evolution, and validation of model-driven spreadsheets. *IEEE Transactions on Software Engineering*, 41:241–263, 03 2015.
- [60] Sarah Barns. Smart cities and urban data platforms: Designing interfaces for smart governance. *City, Culture and Society*, 12:5–12, 2018.
- [61] Carlo A Curino, Hyun J Moon, and Carlo Zaniolo. Graceful database schema evolution: the prism workbench. *Proceedings of the VLDB Endowment*, 1(1):761–772, 2008.
- [62] Vaia Moustaka, Antonis Maitis, Athena Vakali, and Leonidas Anthopoulos. CityDNA Dynamics: A Model for Smart City Maturity and Performance Benchmarking. In *Proc. of the 6th Int. Workshop: Web Intelligence and Smart Cities*, 2020. doi: 10.1145/3366424.3386584.

- [63] ISO 37120:2014. Sustainable development of communities — Indicators for city services and quality of life, 2018. <https://www.iso.org/standard/62436.html>.
- [64] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178, 2020. doi: 10.1109/SEAA51224.2020.00036.
- [65] Davide Di Ruscio, Dimitrios S. Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. Low-code development and model-driven engineering: Two sides of the same coin? *Softw. Syst. Model.*, 21(2):437–446, 2022.
- [66] Javier García-Munoz, Marisol García-Valls, and Julio 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)*, pages 463–470. Springer, 2016.
- [67] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. A tool-supported approach for assessing the quality of modeling artifacts. *Journal of Computer Languages*, 51:173–192, 2019.
- [68] Marcel F van Amstel and MGJ 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.
- [69] Tamás Ambrus and Melinda 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.
- [70] Mario Cardarelli, Ludovico Iovino, Paolo Di Francesco, Amleto Di Salle, Ivano Malavolta, and Patricia Lago. An extensible data-driven approach for evaluating the quality of microservice architectures. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1225–1234, 2019.
- [71] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. Goal Question Metric (GQM) approach. *Encyclopedia of software engineering*, 2002.
- [72] 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)*, pages 320–324, 2010.
- [73] 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*, pages 9–14, 2009.

- [74] Monia Anzivino, Flavio A Ceravolo, and Michele Rostan. The two dimensions of italian academics' public engagement. *Higher Education*, 82(1):107–125, 2021.
- [75] Lorenzo Compagnucci and Francesca Spigarelli. The third mission of the university: A systematic literature review on potentials and constraints. *Technological Forecasting and Social Change*, 161:120284, 2020.
- [76] Birgitte Gregersen, Lisbeth Tved Linde, and Jorgen Gulddahl Rasmussen. Linking between danish universities and society. *Science and public policy*, 36(2):151–156, 2009.
- [77] Juan Piñeiro-Chousa, Aleksandar Šević, and Isaac González-López. Impact of social metrics in decentralized finance. *Journal of Business Research*, 158:113673, 2023. ISSN 0148-2963. doi: <https://doi.org/10.1016/j.jbusres.2023.113673>. URL <https://www.sciencedirect.com/science/article/pii/S0148296323000310>.
- [78] Marjan Mernik, Jan Heering, and Anthony Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37:316–344, 2005.
- [79] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008, Workshops in Computing*. BCS, 2008. URL <http://ewic.bcs.org/content/ConWebDoc/19543>.
- [80] Dennis Wagelaar, Ludovico Iovino, Davide Di Ruscio, and Alfonso Pierantonio. 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)*, pages 192–207. Springer, 2012.
- [81] Tihamer Levendovszky, Daniel Balasubramanian, Anantha Narayanan, and Gabor Karsai. A novel approach to semi-automated evolution of DSML model transformation. In *Proceedings of the International Conference on Software Language Engineering (SLE)*, pages 23–41. Springer, 2009.
- [82] Michael Szvetits and Uwe Zdun. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, 15(1):31–69, 2016.
- [83] Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. What is needed for managing co-evolution in mde? In *Int. Workshop on Model Comparison in Practice*, page 30–38. ACM, 2011.



- [84] E. Brottier, F. Fleurey, J. Steel, B. Baudry, and Y. L. Traon. Metamodel-based test generation for model transformations: an algorithm and a tool. In *Int. Symp. on Software Reliability Engineering*, pages 85–94, 2006.
- [85] Dimitrios S Kolovos, Richard F Paige, Tim Kelly, and Fiona AC Polack. Requirements for domain-specific languages. In *Workshop on Domain-Specific Program Development*, 2006.
- [86] Pedram Veisi and Eleni Stroulia. AHL: Model-Driven Engineering of Android Applications with BLE Peripherals. In *Int. Conf. on E-Technologies*, pages 56–74. Springer, 2017.
- [87] Lorenzo Bettini. *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing, 2016.
- [88] Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. The Epsilon Object Language (EOL). In *European Conf. on Model Driven Architecture-Foundations and Applications*, pages 128–142. Springer, 2006.
- [89] Alfonso de la Vega Dimitris Kolovos and Justin Cooper. 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.
- [90] Louis M Rose, Richard F Paige, Dimitrios S Kolovos, and Fiona AC Polack. The Epsilon Generation Language. In *Proceedings of the European Conference on Model Driven Architecture-Foundations and Applications (ECMFA)*, pages 1–16. Springer, 2008.
- [91] V. Viyović, M. Maksimović, and B. Perisić. Sirius: A rapid development of dsm graphical editor. In *Int. Conf. on Intelligent Engineering Systems (INES)*, pages 233–238, 2014.
- [92] Roger A Light. Mosquito: server and client implementation of the MQTT protocol. *Journal of Open Source Software*, 2(13):265, 2017.
- [93] Louis M Rose, Dimitrios S Kolovos, and Richard F Paige. Eugenia live: a flexible graphical modelling tool. In *Extreme Modeling Workshop*, pages 15–20, 2012.
- [94] Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. Collaborative repositories in model-driven engineering. *IEEE Software*, 32:28–34, 2015.
- [95] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. Model repositories: Will they become reality? In *Cloud-MDE@MoDELS*, 2015.

- [96] Jean-Marc Jézéquel, Olivier Barais, and Franck Fleurey. Model driven language engineering with kermeta. In *Generative and Transformational Techniques in Soft. Eng. IV: Int. Summer School*, pages 201–221. Springer, 2009.
- [97] Florian Heidenreich, Jendrik Johannes, Sven Karol, Mirko Seifert, and Christian Wende. Model-based language engineering with emftext. In *Generative and Transformational Techniques in Soft. Eng. IV: Int. Summer School*, pages 322–345. Springer, 2013.
- [98] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. NoSQL databases. *Lecture Notes, Stuttgart Media University*, 20:24, 2011.
- [99] Baikuntha Narayan Biswal, Pragyan Nanda, and Durga Prasad Mohapatra. A novel approach for scenario-based test case generation. In *2008 International Conference on Information Technology*, pages 244–247. IEEE, 2008.
- [100] Dimitrios S. Kolovos, Louis M. Rose, Nicholas Drivalos Matragkas, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan de Lara, István Ráth, Dániel Varró, Massimo Tisi, and Jordi Cabot. A research roadmap towards achieving scalability in model driven engineering. In Davide Di Ruscio, Dimitris S. Kolovos, and Nicholas Matragkas, editors, *Proceedings of the Workshop on Scalability in Model Driven Engineering*. ACM, 2013.
- [101] Ian Sommerville. *Software engineering, 8th Edition*. Addison-Wesley, 2007. ISBN 9780321313799.
- [102] Ministero della Salute. Tabella 21 indicatori, 2020. [https://www.salute.gov.it/imgs/C\\_17\\_notizie\\_5152\\_1\\_file.pdf](https://www.salute.gov.it/imgs/C_17_notizie_5152_1_file.pdf).
- [103] Fernando Macías, Adrian Rutle, and Volker Stolz. MultEcore: Combining the Best of Fixed-Level and Multilevel Metamodelling. In *Proceedings of the 3rd International Workshop on Multi-Level Modelling (MULTI)*, volume 1722 of *CEUR Workshop Proceedings*, pages 66–75, 2016.
- [104] Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. The Epsilon Transformation Language. In *Proceedings of the International Conference on Theory and Practice of Model Transformations (ICMT)*, pages 46–60. Springer, 2008.
- [105] Abel Gómez, Jordi Cabot, and Manuel Wimmer. TemporalEMF: A temporal meta-modeling framework. In *Proceedings of the International Conference on Conceptual Modeling (ER)*, pages 365–381. Springer, 2018.
- [106] Dirk Hartmann and Herman Van der Auweraer. Digital twins. In Manuel Cruz, Carlos Parés, and Peregrina Quintela, editors, *Progress in Industrial Mathematics:*

- Success Stories*, pages 3–17, Cham, 2021. Springer International Publishing. ISBN 978-3-030-61844-5.
- [107] Fabian Dembski, Uwe Wössner, Mike Letzgus, Michael Ruddat, and Claudia Yamu. Urban digital twins for smart cities and citizens: The case study of herrenberg, germany. *Sustainability*, 12(6):2307, 2020.
- [108] Felipe Albertao, Jing Xiao, Chunhua Tian, Yu Lu, Kun Qiu Zhang, and Cheng Liu. Measuring the sustainability performance of software projects. In *2010 IEEE 7th International Conference on E-Business Engineering*, pages 369–373. IEEE, 2010.
- [109] Francesca Righetti, Carlo Vallati, and Giuseppe Anastasi. Iot applications in smart cities: A perspective into social and ethical issues. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 387–392, 2018. doi: 10.1109/SMARTCOMP.2018.00034.