GRANT AGREEMENT No 609035
FP7-SMARTCITIES-2013

Real-Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications

Collaborative Project

Smart City Reference Datasets and Key Performance Indicators (KPIs)

Document Ref.	D2.3
Document Type	Report
Workpackage	WP2
Lead Contractor	ERIC
Author(s)	Athanasios Karapantelakis (Editor,ERIC), Daniel Puschmann (UNIS), Daniel Kuemper (UASO), Thorben Iggena (UASO), Muhammad Intizar Ali (NUIG), Feng Gao (NUIG), Sefki Kolozali (UNIS), Hongxin Liang (ERIC)
Contributing Partners	UNIS, ERIC, UASO
Planned Delivery Date	M18
Actual Delivery Date	27/02/14
Dissemination Level	Public
Status	Final
Version	V2.3
Reviewed by	Michelle Bak Mikkelsen (AA), Daniel Kuemper (UASO), Konstantinos Vandikas (ERIC) and Nazli Farajidavar (UNIS)

Table of Contents

1. Introduction

This document presents the work that has been carried out in the context of Activity 2.3 of the Citypulse EU FP7 project. The goals of this activity as outlined in the Citypulse project plan [1] were to deliver a set of reference datasets for performance evaluation of Smart City Frameworks (SCFs)[1]. In particular, the tasks inside this activity were the following:

- Task 1: Provide a set of smart city-related reference datasets
- Task 2: Provide reference Key Performance Indicators (KPIs) and performance evaluation methods for Smart City Frameworks (SCFs)

The rest of this report presents the deliverables of the two tasks in greater detail.

2. Smart City Reference Datasets

This section is organized into two parts. The first part introduces the ontology developed to semantically annotate smart city datastreams. The second part describes the published datasets.

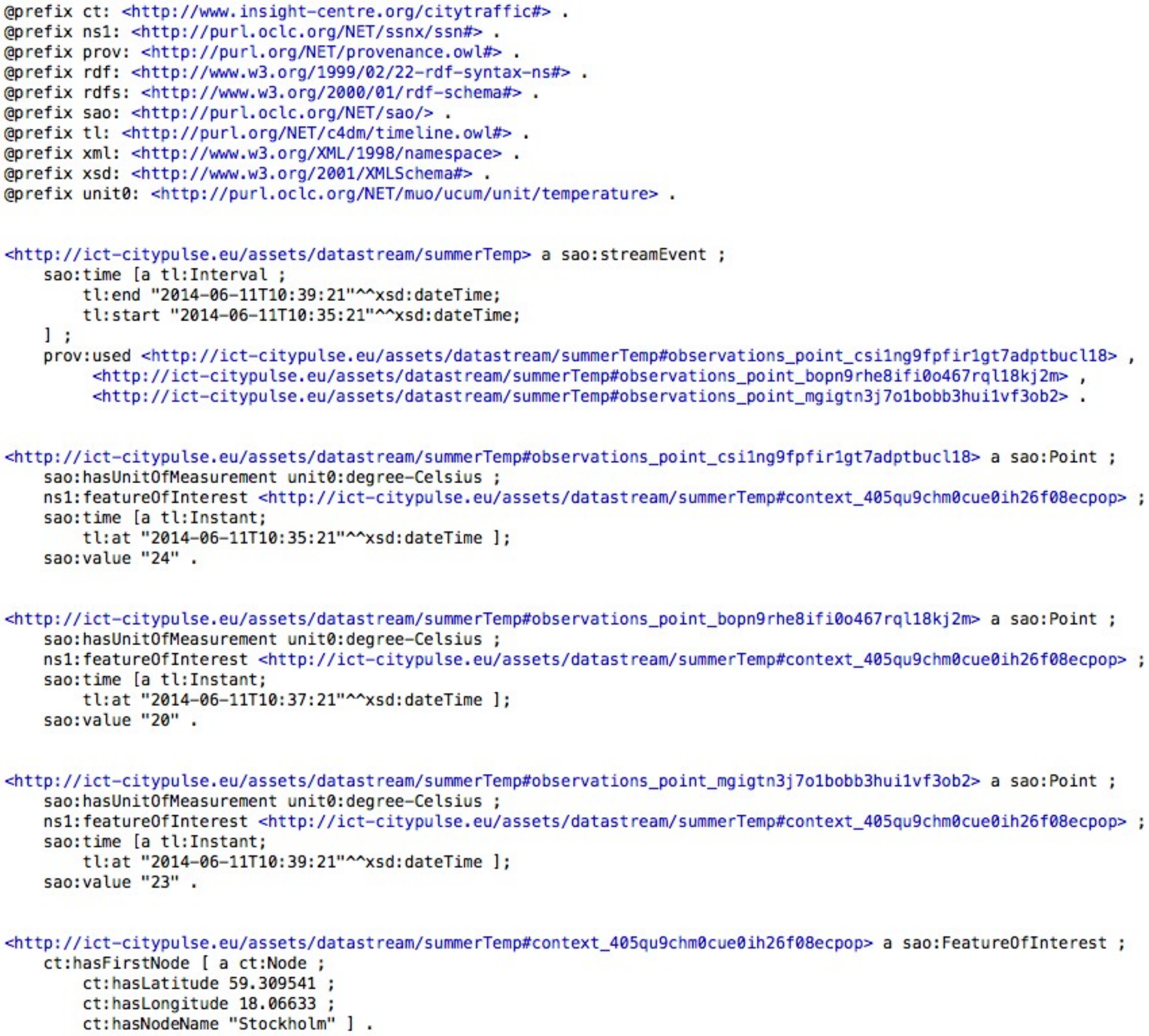
2.1. Semantic Annotation of Smart City Datastreams

Activity 2.3 in Citypulse EU FP7 project captured a number of semantically annotated datasets to be used from subsequent work packages, but also from the community. The annotated datasets are based in the Information Model designed in Work Package 3 [2]. Every dataset is represented in Terse RDF Triple Language (Turtle) format [3], and builds on a number of ontologies including:

- SAO ontology for semantic stream annotation [4]
- MUO ontology [5] and UCUM vocabulary [6] for representing units of measurement. In addition, activity 2.3 extended MUO with it's own "smart city" vocabulary, extending MUO with all measurements from captured datasets that are not found in UCUM [7].
- Timeline ontology used by SAO ontology above for describing time instances [8].
- SSN ontology used by SAO ontology above for describing observations [9].
- City Traffic ontology [10] from Insight Centre [11] for describing contextual information of data streams.

The structure of a semantically annotated dataset is illustrated in figure 1. In this example, the dataset contains one data stream, with 3 temperature readings in Celsius degrees from a hypothetical sensor measuring temperature in Stockholm, Sweden.

[1]: The reader should note the distinction between the term *dataset* and *datastream* in the context of this report. A dataset can be comprised of more than one datastreams. For example, the published traffic dataset contains 449 datastreams (see section 2.2).



```
@prefix ct: <http://www.insight-centre.org/citytraffic#> .
@prefix ns1: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix prov: <http://purl.org/NET/provenance.owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sao: <http://purl.oclc.org/NET/sao/> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix unit0: <http://purl.oclc.org/NET/muo/ucum/unit/temperature> .


<http://ict-citypulse.eu/assets/datastream/summerTemp> a sao:streamEvent ;
    sao:time [a tl:Interval ;
        tl:end "2014-06-11T10:39:21"^^xsd:dateTime;
        tl:start "2014-06-11T10:35:21"^^xsd:dateTime;
    ] ;
    prov:used <http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_csi1ng9fpfir1gt7adptbucl18> ,
         <http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_bopn9rhe8ifi0o467rql18kj2m> ,
         <http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_mgigtn3j7o1bobb3hui1vf3ob2> .


<http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_csi1ng9fpfir1gt7adptbucl18> a sao:Point ;
    sao:hasUnitOfMeasurement unit0:degree-Celsius ;
    ns1:featureOfInterest <http://ict-citypulse.eu/assets/datastream/summerTemp#context_405qu9chm0cue0ih26f08ecpop> ;
    sao:time [a tl:Instant;
        tl:at "2014-06-11T10:35:21"^^xsd:dateTime ];
    sao:value "24" .


<http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_bopn9rhe8ifi0o467rql18kj2m> a sao:Point ;
    sao:hasUnitOfMeasurement unit0:degree-Celsius ;
    ns1:featureOfInterest <http://ict-citypulse.eu/assets/datastream/summerTemp#context_405qu9chm0cue0ih26f08ecpop> ;
    sao:time [a tl:Instant;
        tl:at "2014-06-11T10:37:21"^^xsd:dateTime ];
    sao:value "20" .


<http://ict-citypulse.eu/assets/datastream/summerTemp#observations_point_mgigtn3j7o1bobb3hui1vf3ob2> a sao:Point ;
    sao:hasUnitOfMeasurement unit0:degree-Celsius ;
    ns1:featureOfInterest <http://ict-citypulse.eu/assets/datastream/summerTemp#context_405qu9chm0cue0ih26f08ecpop> ;
    sao:time [a tl:Instant;
        tl:at "2014-06-11T10:39:21"^^xsd:dateTime ];
    sao:value "23" .


<http://ict-citypulse.eu/assets/datastream/summerTemp#context_405qu9chm0cue0ih26f08ecpop> a sao:FeatureOfInterest ;
    ct:hasFirstNode [ a ct:Node ;
        ct:hasLatitude 59.309541 ;
        ct:hasLongitude 18.06633 ;
        ct:hasNodeName "Stockholm" ] .
```

FIGURE 1: Smart City Datastream Model

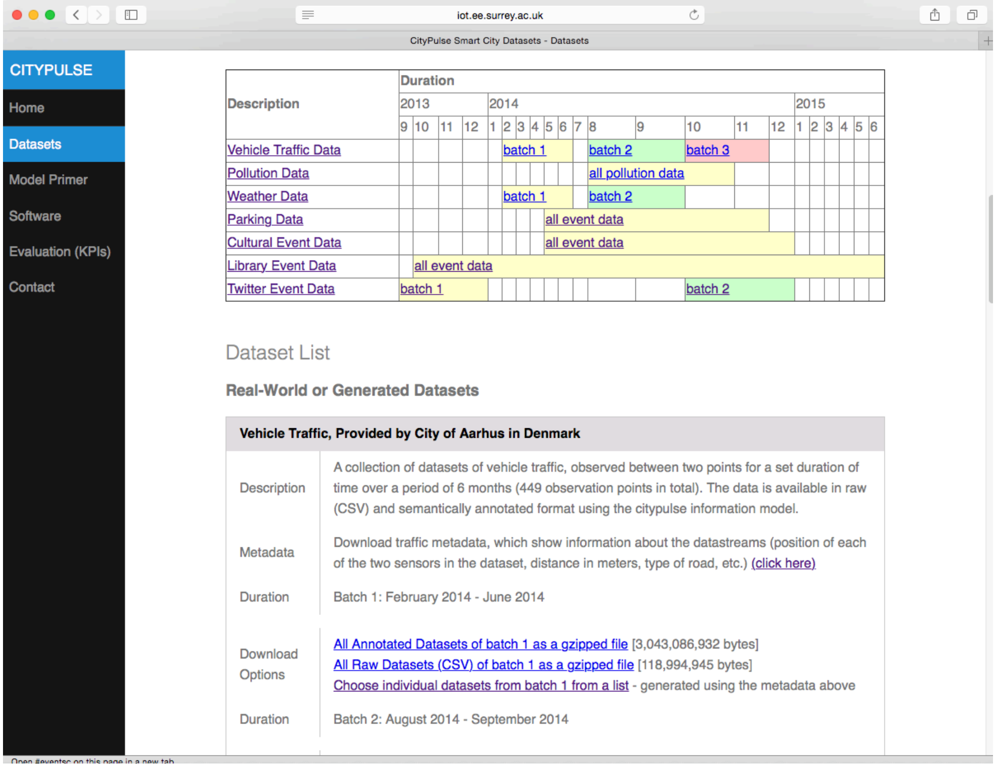
There are three areas to note:

- The *sao:streamEvent* object, which describes some general properties of the data stream: the time of the first observation belonging to this data stream as well as the time of the last observation belonging to the same stream. Additionally, it references all the data points, i.e. the individual observations. Note that every resource in the data stream is identified by a general property prefix (stream event for the streamEvent object, point for the individual observations as well as stream for the feature of interest, (which we will cover later in this section), as well as an appended, randomly-generated hash.
- A series of *sao:Point* objects, each accounting for an individual observation. Every such object has a unit of measurement attached to it (the unit of measurement in this case is degrees Celsius and is described in the UCUM repository), and an optional *featureOfInterest* association in case such an object exists, as well as the time of observation and the observed (could be an integer, string, etc., according to the unit of measurement description).

- The *sao:featureOfInterest* instance is an optional instance, with a purpose to provide a general description for the data stream. In this case it describes the location where the measurements have taken place, both using geographical coordinates as well as a general description of the location. Note the one to many association between the *sao:Point* instances and the *sao:featureOfInterest*. This is not always the case, as individual Point instances can reference different featureOfInterest instances (see for example the traffic ontology above). There can be cases where a sensor is mobile, or sensors form different groups in different locations, etc.

2.2. Published Datasets

The datasets were semantically annotated and published in the **CityPulse datasets website** [12] under the "Creative Commons Attribution 4.0 International License", which practically means that everybody can use them as long as they reference Citypulse work [13] (see figure 2).



Description	Duration																					
	2013				2014												2015					
	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6
Vehicle Traffic Data						batch 1						batch 2		batch 3								
Pollution Data												all pollution data										
Weather Data						batch 1						batch 2										
Parking Data									all event data													
Cultural Event Data									all event data													
Library Event Data		all event data																				
Twitter Event Data	batch 1													batch 2								

Dataset List

Real-World or Generated Datasets

Vehicle Traffic, Provided by City of Aarhus in Denmark

Description	A collection of datasets of vehicle traffic, observed between two points for a set duration of time over a period of 6 months (449 observation points in total). The data is available in raw (CSV) and semantically annotated format using the citypulse information model.
Metadata	Download traffic metadata, which show information about the datastreams (position of each of the two sensors in the dataset, distance in meters, type of road, etc.) (click here)
Duration	Batch 1: February 2014 - June 2014
Download Options	All Annotated Datasets of batch 1 as a gzipped file [3,043,086,932 bytes] All Raw Datasets (CSV) of batch 1 as a gzipped file [118,994,945 bytes] Choose individual datasets from batch 1 from a list - generated using the metadata above
Duration	Batch 2: August 2014 - September 2014

FIGURE 2: Website of Published Datasets, KPIs and related Software

On the main page of the aforementioned website, users are encouraged to cite relevant CityPulse papers [12]. The datasets were captured from the city of Aarhus in Denmark as well as the city of Guildford in Surrey state of the UK and Ericsson Research in Sweden. Table 1 shows the captured datasets (note that the pollution dataset that was published was actually generated).

Table 1: List of datasets captured at the time the report is drafted.

Description	Duration	Location (Provider)	Datastream number	Type
Vehicle Traffic Data	2/2014 - 6/2014 8/2014 - 9/2014 10/2014 - 11/2014	Aarhus, Denmark (Open Data Aarhus)	449	Real
Pollution Data	8/2014 - 10/2014	Aarhus, Denmark (Open Data Aarhus)	2245	Generated
Weather Data	2/2014 - 6/2014 8/2014 - 9/2014	Aarhus, Denmark (Open Data Aarhus)	6	Real
Cultural Event Data	5/2014 - 1/2015	Aarhus, Denmark (Open Data Aarhus)	1	Real
Twitter Data	9/2013 - 12/2013 10/2014 - 12/2014	Aarhus, Denmark (Twitter)	1	Real
Social Event Data	6/2012 - 6/2014	Surrey, UK (Municipality RSS)	1	Real
Library Event Data	10/2013 - 6/2015	Aarhus, Denmark	1	Real
Parking Data	5/2014 - 11/2014	Aarhus, Denmark	1	Real
Meeting Room Data	Ongoing (live)	Ericsson Research, Kista	4	Real

The datasets are published in "raw" format (most of them in CSV [14]), as well as an annotated format based on the CityPulse information model presented in section 2.1. For those datasets captured in the City of Aarhus, there is spatiotemporal correlation, which means that some of the datasets can be used in combination for evaluating performance of Smart City Frameworks supporting multi-modal scenarios with multiple data streams.

Table 2 shows, we observe significant spatiotemporal correlation of heterogeneous data streams between the months of May to December 2013 (note that only the datasets of Aarhus are shown).

Table 2: Temporal Correlation for the Published Datasets from the city of Aarhus, coloured cells show available data.

Description	Duration (years and months)																					
	2013				2014												2015					
	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6
Vehicle Traffic Data						🟧	🟧	🟧	🟧	🟧		🟧	🟧	🟧	🟧	🟧						
Pollution Data												🟩	🟩	🟩	🟩							

Weather Data						■	■	■	■	■		■	■	■								
Parking Data									■	■	■	■	■	■	■							
Cultural Event Data									■	■	■	■	■	■	■	■						
Library Event Data		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Twitter Event Data	■	■	■	■										■	■	■						

2.3 Datastream Playback and Generation Tool

In addition to published datasets, Activity 2.3 has also implemented and made available a tool for generating and playing back semantically annotated data streams [15]. The semantic annotation supported is based on the Citypulse information model, which practically means that any of the published datasets can be played back using this tool in real-time or accelerated time. Datastream playback of published datasets can benefit both new IoT applications that use data of type similar to those listed in tables 1 and 2 above, but also the evaluation of performance of smart city frameworks (see section 3.2.4).

Additionally, this tool can be used to generate data and supports a variety of distributions in order to approximate realism. In more detail, the feature set of this tool includes:

- Simple to use, command-line user interface.

  – Data Generation using different distributions (Poisson, Exponential, Geometric, Pareto, Gaussian, Uniform/Random and Constant).

  – Ability to specify starting date of data collection, periodicity (amount of time between subsequent measurements), prefix of URI of generated datastream.

  – Uses the MUO ontology and the UCUM vocabulary for representing units of measurement in datasets, as well as the SSN ontology for representing individual observations.

  – Ability to specify stream metadata in the same information model (e.g. description and geographical location)

  – Uses Turtle (.ttl) notation to export the generated files.

- Data Playback support with built-in validation of .ttl using Apache Jena.

  – Customizable speed of playback (real-time, accelerated, decelerated).

    – Customizable mode of data transmission (JSON/UDP sockets).

3. Key Performance Indicators

3.1 Introduction

The vision of a Smart City, where services to improve everyday life are offered to citizens, is realized through a number of use cases or Smart City scenarios, incrementally contributing towards this vision.

Smart City scenarios exhibit great variance in a number of areas. The 101 scenarios defined for the CityPulse EU project exemplify this variance [16]. Number of users and data sources, spatiotemporal scenario coverage, security, and network and data processing capabilities are few of the factors that vary across different scenarios. In order to be able to support Smart City scenarios, designers have to take under account the distinct requirements of every scenario when architecting the Smart City Framework (SCF), i.e. the hardware and software infrastructure to support the aforementioned scenarios.

On the other hand, existing SCFs would require a set of quantifiable metrics for evaluation against the requirements of future Smart City scenarios. This page introduces a primer for evaluating SCF support for Smart City scenarios. Figure 1 below illustrates our approach.

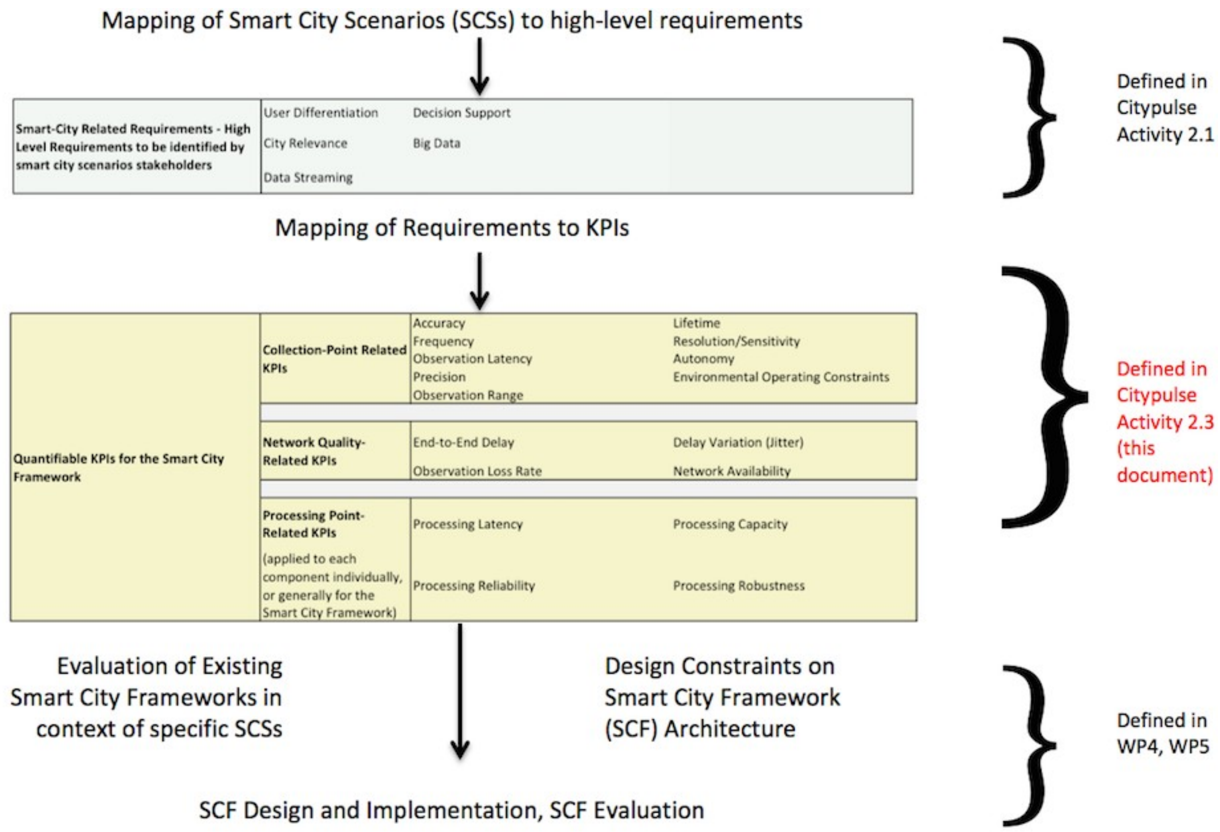

Mapping of Smart City Scenarios (SCSs) to high-level requirements
Smart-City Related Requirements - High Level Requirements to be identified by smart city scenarios stakeholders
User Differentiation
Decision Support
City Relevance
Big Data
Data Streaming
Defined in Citypulse Activity 2.1
Mapping of Requirements to KPIs
Quantifiable KPIs for the Smart City Framework
Collection-Point Related KPIs
Accuracy
Frequency
Observation Latency
Precision
Observation Range
Lifetime
Resolution/Sensitivity
Autonomy
Environmental Operating Constraints
Network Quality-Related KPIs
End-to-End Delay
Observation Loss Rate
Delay Variation (Jitter)
Network Availability
Processing Point-Related KPIs
(applied to each component individually, or generally for the Smart City Framework)
Processing Latency
Processing Reliability
Processing Capacity
Processing Robustness
Defined in Citypulse Activity 2.3 (this document)
Evaluation of Existing Smart City Frameworks in context of specific SCSs
Design Constraints on Smart City Framework (SCF) Architecture
SCF Design and Implementation, SCF Evaluation
Defined in WP4, WP5

FIGURE 3: Cross-Work Package Approach for Evaluating and/or Designing SCFs to support Smart City Scenarios

The high-level requirements for a Smart City scenario were developed in Activity 2.1 of Citypulse. The report established a number of functional and non-functional requirements of Smart City scenarios that can affect planning and design of supporting infrastructure. The non-functional requirements are mostly about relevancy of the scenario to a municipality, individual citizen, cultural, regulatory/government level. They are scenario-specific and require qualitative analysis of context outside of the technical capabilities of a SCF. There are however functional requirements, which can be mapped to measurable/formally definable metrics (Key Performance Indicators or KPIs).

The rest of this page describes the high-level requirements of a Smart City scenario, the measurable KPIs at an SCF level, and:

- How high-level scenario requirements can be mapped to KPIs for designing new SCFs to implement these scenarios.
- How the KPIs for measuring the performance of an SCF can be mapped to high- level requirements to evaluate applicability of current SCFs to smart city scenarios.

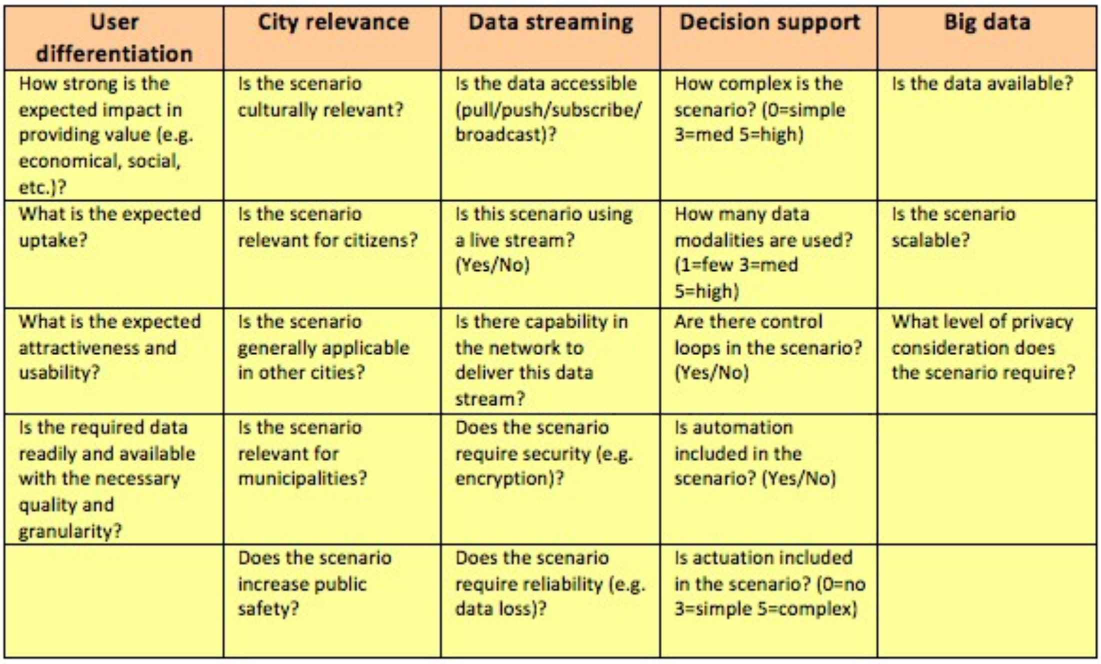
3.2 Presentation of high-level Smart City scenario requirements and SCF-quantifiable KPIs

3.2.1 Introduction

We begin with a description of the high level requirements for Smart City scenarios, and continue with a detailed description of the KPIs on a SCF level. The goal of this section is to offer a complete picture of what aspects should be considered when conceiving a Smart City Scenario and how it's requirements should be documented, as well as to offer a reference of the technical considerations (in the form of KPIs) that have to be taken under account when the scenario is being implemented. The next section discusses how the KPIs can be mapped to some of the high-level requirements.

- Smart City Scenario (application - related) KPIs: These are high-level KPIs in the form of requirements, which were defined as part of Activity 2.1 in Citypulse. The KPIs are illustrated in table 3 below.

Table 3: High-level Smart City scenario requirements. These requirements are rated at a qualitative level (1-5) per scenario.



User differentiation	City relevance	Data streaming	Decision support	Big data
How strong is the expected impact in providing value (e.g. economical, social, etc.)?	Is the scenario culturally relevant?	Is the data accessible (pull/push/subscribe/broadcast)?	How complex is the scenario? (0=simple 3=med 5=high)	Is the data available?
What is the expected uptake?	Is the scenario relevant for citizens?	Is this scenario using a live stream? (Yes/No)	How many data modalities are used? (1=few 3=med 5=high)	Is the scenario scalable?
What is the expected attractiveness and usability?	Is the scenario generally applicable in other cities?	Is there capability in the network to deliver this data stream?	Are there control loops in the scenario? (Yes/No)	What level of privacy consideration does the scenario require?
Is the required data readily and available with the necessary quality and granularity?	Is the scenario relevant for municipalities?	Does the scenario require security (e.g. encryption)?	Is automation included in the scenario? (Yes/No)	
	Does the scenario increase public safety?	Does the scenario require reliability (e.g. data loss)?	Is actuation included in the scenario? (0=no 3=simple 5=complex)	

- Smart City Framework - related (SCF) KPIs: The application-related KPIs presented above drive the design of a SCF, which is itself constrained from functional requirements, i.e. quantifiable, measurable KPIs. We use as primer the functional structure of a generic SCF as illustrated in figure 7, to define KPIs measuring different aspects (e.g. quality, efficiency, reliability, security etc.) of SCF performance upon collection of Smart City data, transmission of the collected data for processing and finally processing of this data. Data is observed (measured) from and transmitted by observation assets to the SCF for further processing. KPIs that evaluate SCFs can be applied to every step of this process, i.e. on the observation assets that collect the data, on the network connection that transports this data from the observation assets to the SCF, as well as internally to the SCF where the data is being processed.

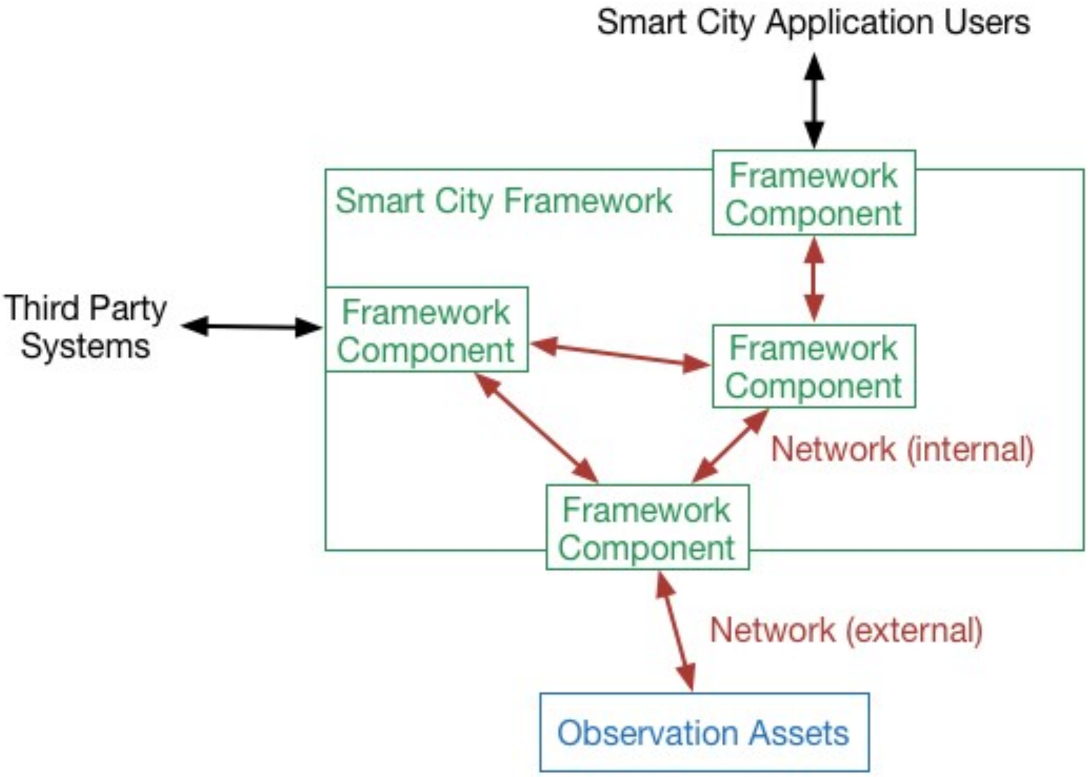


FIGURE 4: Functional view of a generic SCF, which drives categorisation of KPIs on a functional requirement level.

Based on the functional component these KPIs are measured in, we split them into three categories:

- *Observation Point-Related KPIs*: These are KPIs that relate to the function of observation points (as observation points we define assets such as virtual and physical sensors that report their observations of parameters of the virtual, physical environment to the SCF) as well as the environmental, temporal constraints these observation points operate in.
- *Network Connection-Related KPIs*: These are KPIs that relate to the quality of the network connection between the observation points and the SCF, but also internally to the SCF, between its components.
- *Data Processing-Related KPIs*: These are KPIs that relate to the computational capabilities of the SCF and can be applied to different levels of abstraction, i.e. to the SCF as a whole, or on individual components of the SCF. If applied to a component-level, these KPIs are further split into two subcategories, internal KPIs (which concern the performance of data processing inside a component), and external KPIs (which treat the components as ”black boxes”, only measuring different parameters of data input and output).

We describe the specific KPIs as well as ways they can be measured in the subsections below. Note that the measurement methods supplied are merely suggestions, and that KPIs can also be measured in other ways - consistent of course with the original definition of the respective KPI.

3.2.2 Observation Point KPIs

As *Observation Point*, we define an entity that monitors its environment and measuring one or more values. An Observation Point entity can be thought of as a sensor, or groups of sensors that measure qualities (e.g. temperature, humidity, amount of traffic, etc.) And send the observed data to a Processing Entity (e.g. IoT middleware software) using a Network Connection.

Table 4 shows the KPIs for the Observation Points. Note that in many cases the unit of measurement in the metric column is contextual to the smart city application implemented.

Table 4 Observation Point Key Performance Indicators (KPIs)

KPI Name	Description - What this KPI is About	Metric - What is Measured in the KPI	Method - Practical Way to Measure KPI
Accuracy	How close are observed values to real values.	Measured in absolute terms (units of measured quality) or as a percentage.	Observed values can be compared against a trustworthy source.
Frequency	Number of observations recorded in a given time span.	Number of observations per unit of time.	Can be measured as a mean of . Standard deviation can be used for indicating whether the frequency is a synchronous quality or not.
Observation Latency	Time between a value was observed and the value was reported by the observing observation point.	Can be measured in a unit of time, indicating the time span (e.g. milliseconds)	Can be measured by profiling observation point software. Can also be retrieved from the specifications of the observation points.
Precision	Can the observed value be consistently reproduced and can the number of significant digits of the observed value be reliably measured.	Measured in absolute terms (difference in units of measured quality) or as a percentage.	Can be checked by validating the observation point's mean value against the true value, compared against a trustworthy resource. Can also be retrieved by the observation point manufacturer specifications.
Observation Range	The range of values that can be observed from the specific observation point. Values out of range cannot be observed/reported properly.	Expressed as a bounded set of [a, b], where a is the minimum observed value, and b is the maximum observed value	Can be retrieved from the specifications provided form the observation point manufacturer or through experimentation.
Lifetime	The time the observation point can function properly/reliably	Measured in a unit of time (e.g. months)	Can be retrieved from the specifications provided form the observation point manufacturer or through experimentation.
Resolution/Sensitivity	Smallest change in the observed value that can be detected from the observation point.	Measured in absolute terms (units of measured quality)	Can be retrieved from the specifications provided form the observation point manufacturer or through experimentation.
Autonomy	How much time can the observation point continue to operate without need for maintenance (e.g. replacement of batteries).	Measured in a unit of time (e.g. years)	Can be retrieved from the specifications provided form the observation point manufacturer or through experimentation.

Security and Confidentiality	Can the observations be retrieved and relayed in a secure way? Can the observation be delivered to the proper requestor?	Requestor authorization (in case of a pull-based observation request mechanism), data encryption.	Supported confidentiality and security mechanisms can be retrieved from the specifications and/or implemented using a combination of security protocols and access control lists.
Environmental Operating Constraints	The ranges of environmental qualities within which the observation point operates properly.	Measured in absolute terms, or units of time, expressed as bounded sets of [a, b] where ‘a’ is the minimum observed value, ‘b’ is the maximum value.	Can be retrieved from the specifications provided form the observation point manufacturer or through experimentation.

3.2.3 Network Connection KPIs

The Network Connection KPIs measure the quality properties of the communication medium through which observed data are transferred from the Observation Points to the Data Processing Point. Table 5 shows the KPIs for the network connection. Note that these KPIs can also be used to measure network performance internally to the SCF between components.

Table 4 Network Connection Key Performance Indicators (KPIs)

KPI Name	Description - What this KPI is About	Metric - What is Measured in the KPI	Method - Practical Way to Measure KPI
End-to-End Delay	The delay between the time the observation was send from an Observation Point and the time it was received from a Processing Point.	Measured in a unit of time (e.g. milliseconds)	Make sure clocks of Observation Point and Processing Point are synchronized (e.g. by use of an NTP server). A: Ensure all observations are annotated using a timestamp of time of transmission (see CityPulse information model - Model Primer section). Compare the timestamp of the observation with the clock of the processor at the time of reception. B: Use ICMP "ping" protocol periodically to take measurements.
Observation Loss Rate	Observations transmitted from the Observation Point but never received from from the Processing Point due to a network issue.	Number of observation per unit of time ("rate of lost data").	Can be measured by storing information on data sent from Observation Points and data received from Collection Point and comparing the two sets, e.g. using a simple comparison algorithm and a database.
Delay Variation (Jitter)	Variation of time of arrival of Observations at the Processing Point, relative to time of transmission from the Observation Point.	Measured in a unit of time (e.g. milliseconds)	Can be checked by validating the observation point's mean value against the true value, compared against a trustworthy resource. Can also be retrieved by the observation point manufacturer specifications.

Security and Confidentiality	Relates to whether encryption and/or data access permission protocols are used to avoid malicious use of monitored data.	Relates to whether the transmission of data from the observation point to the processing point is done using an encrypted set of protocols and/or permissions on data access.	An example for secure data transmission would be an SSL-enabled HTTP session (e.g. using REST protocol). Low-power protocols such as 6LowPan and Xbee allow for security using Access Control Lists (ACLs).
Network Availability	Measurement of network availability as a percentage of total time	Measured in a percentage (% available of total time)	Periodically using the ICMP ping protocol to check if the connection is still up

3.2.4 Data Processing Related KPIs

As illustrated in figure 4, we consider a SCF as a collection of components which interoperate and perform individual functions (e.g. data aggregation, federation/transformation, reasoning, etc.). Based on their function, some of these components may interact with external entities, e.g. other systems or users.

It is possible that the output of one component may be required as part of an input to another. As *Processing Point*, we define an entity that is part of the SCF and performs some processing on incoming data. This entity may be a combination of hardware and software resources, and can consist of one component, several components, or even the complete system, depending on the desired level of abstraction. For example, it may be desired to measure the performance of the data collection and storage components of the SCF together, or individually.

We define two categories of KPIs for a *Processing Point*, namely ”Generic Processing Point Performance” and ”Internal Processing Point Performance”.

- Generic Processing Point Performance KPIs

    These KPIs are generic and consider the Processing Point to be a ”black box”, i.e. they only measure aspects of performance in terms of quantitative parameters for data input, and processed data output. They cannot evaluate the specific implementation of a Processing Point, and may be more suitable when the Processing Point implementation details are not known or are too complex to measure.

    In this level of abstraction, a Processing Point performs a *process* operation on an incoming Processing Unit. Depending on the nature of the Processing Point being evaluated, a Processing Unit can be defined as an individual observation in a data stream, multiple observations of one data stream, multiple observations of many data streams, or other data. The result of the *process* operation is a Processed Unit, which can be valid (if the process operation completed successfully) or invalid (if the process operation completed but the Processing Unit was not the one expected). If the Processing Unit was discarded or the process operation did not complete, a unit is nominated as Unprocessed Unit. Table 5 shows the Generic Processing Point Performance KPIs defined in this activity.

Table 5: Generic Processing Point Performance KPIs

KPI Name	Description - What this KPI is About	Metric - What is Measured in the KPI	Method - Practical Way to Measure KPI
Processing Latency	The average delay between receiving a Processing Unit of a datastream in a Processing Point's input interface and returning the result of the Processed Unit as output	Measured in a unit of time (e.g. seconds)	Can be measured by using a profiler if processing component is a function. Alternatively, logging of time of input of the Processed Unit and output of the Processed Unit can be used to calculate an average.
Processing Reliability	The ratio of incorrectly-processed (invalid) Processed Units of a datastream versus the total number of Processed Units	Measured in a normalized scale [0,1], values closer to 0 indicating a more reliable Processing Point	The validation process depends on the operation of a Processing Point. For example, if the goal of processing is to do data transformation, Processed Units can be validated against an ontology.
Processing Capacity	The number of Processed Units a Processing Point can output for a set duration	Numbers of Processed Units over a unit of time (e.g. seconds)	Can be measured by using a profiler if processing component is a function, i.e. logging of number of convergences of the processing function over time can be used to calculate the processing rate.
Processing Robustness	The total number of Processed Units versus the number of Processed and Unprocessed units	Measured in a normalized scale [0,1], values closer to 1 indicating a more robust Processing Point	Can be measured by logging the number of incoming Processing Units and the number of Processed Units as well as the number of discarded (unprocessed) units.

- Internal Processing Pont Performance KPIs

These KPIs measure aspects of performance of data processing algorithms inside a Processing Point, and can be used to compare performance of different implementations of a Processing Point with the same function but different data processing algorithms. In addition to the aforementioned Smart City specific KPIs, more conventional measures of algorithm complexity are also applicable using the Big-O notation. From this perspective it is interesting to evaluate proposed algorithms in terms of time (how much it takes for the algorithm to complete it's task in relation to the input) and in terms of memory (how much memory an algorithm consumes in relation to it's input.) In case of well-known algorithms (e.g. graph search algorithms such as backtracking, beam search, or sorting algorithms such as quick sort and bubble sort), complexity is known in advance, whereas in other cases, complexity has to be found e.g. by reduction using a solver. The classification of an algorithm will indicate whether suitable in terms of execution time and resources an algorithm is for the task it was chosen to perform. On a practical level, the algorithm's performance can be quantified in computational resources and execution time using a reference system. Such measurements can be good for benchmarking different implementations of an algorithm (e.g.

in different programming languages), or different algorithms that produce the same output given the same input. This helps to choose one algorithm implementation over another, even if the algorithms compared are the same or have the same complexity class (see table 6).

Table 6: Processing Point Key Performance Indicators (KPIs)

KPI Subcategory	KPI Name	Description - What this KPI is About	Metric - What is Measured in the KPI	Method - Practical Way to Measure KPI
Complexity - Compare different algorithms	***Temporal Complexity***	The class of time complexity of an algorithm, or the classes of complexity of a group of algorithms in the processing point.	Big-O Notation	Can be measured using a solver or by reference.
	Spatial Complexity	The class of space complexity of an algorithm, or the classes of complexity of a group of algorithms in the processing point.	Big-O Notation	Can be measured using a solver or by reference.
Benchmarking - Compare implementations of the same algorithm	***RAM Utilization***	How much system memory (RAM) the algorithm uses in average, measured on a reference system, using specific input.	Utilization in bytes, or multiple (kilobytes, megabytes, etc.)	Algorithm can be implemented using a programming language and can be profiled using a code profiling tool.
	CPU Utilization	How much CPU time the algorithm uses in average, measured on a reference system, using specific input.	Utilization in percentage of execution time (see below)	Algorithm can be implemented using a programming language and can be profiled using a code profiling tool.
	HDD Utilization	How much system memory (RAM) the algorithm uses in average, measured on a reference system, using specific input.	Utilization in bytes or multiple (kilobytes, megabytes, etc.)	Algorithm can be implemented using a programming language and can be profiled using a code profiling tool.
	Execution Time	How long does it take for the algorithm to converge, measured on a reference system, using specific input.	Measured in a unit of time (e.g. seconds)	Algorithm can be implemented using a programming language and can be profiled using a code profiling tool.

3.3 Mapping High-Level Smart City Scenario Requirements to Measurable SCF KPIs

This section describes how high-level smart city scenario requirements can be mapped to measurable SCF KPIs (see table 7). High-level requirements at a Smart City scenario-level affect some of the quantifiable KPIs. The actual values of each of the KPIs (e.g. a desired value range, a higher or lower threshold of operations etc.), are scenario-specific and to be determined from a requirement engineer. For example, consider a fire prevention scenario, which monitors variations in temperatures of buildings, in order to detect a potentially dangerous situation as early as possible. In this case, the live-stream and reliability requirements (R3.3 and R3.5 in figure 3) may pose more strict constraints in the KPIs (e.g. much lower end-to-end delay and data loss), than for example, a scenario, which monitors average rainfall in a city. The mapping can be interpreted in two different ways:

- As design constraints to system architects for creating a new SCF for specific Smart City scenarios or Smart City scenario categories.
- As a benchmark to measure performance of existing SCFs against specific Smart City scenarios or Smart City scenario categories.

Table 7: Mapping of High-Level Smart City Requirements to KPIs

	High-Level Requirement	KPIs Applicable
User Differentiation	*R1.1 How strong is the expected impact in providing value (e.g. economical, social, etc.)?*	Non-Functional Requirement
	R1.2 What is the expected uptake?	Uptake affects Network-Related KPIs (delay, loss rage, jitter, availability)
	R1.3 What is the expected attractiveness and usability?	Non-Functional Requirement
	R1.4 Is the required data readily and available with the necessary quality and granularity?	Observation Point-Related KPIs
City Relevance	*R2.1 Is the scenario culturally relevant?*	Non-Functional Requirement
	R2.2 Is the scenario relevant for citizens?	
	R2.3 Is the scenario generally applicable in other cities?	
	R2.4 Is the scenario relevant for municipalities?	
	R2.5 Does the scenario increase public safety?	
Data Streaming	*R3.1 Is the data accessible (pull/push/subscribe/broadcast)?*	Affects Data Accessibility KPI
	R3.2 Is this scenario using a live stream? (Yes/No)	Live Stream affects parameters of Network-Related KPIs
	R3.3 Is there capability in the network to deliver this data stream?	Network-Related KPIs
	R3.4 Does the scenario require security (e.g. encryption)?	Affects Security KPI on Observation Point and Network categories
	R3.5 Does the scenario require reliability (e.g. data loss)?	Network-Related KPIs

Decision Support	*R4.1 How complex is the scenario?*	Non-Functional Requirement
	R4.2 How many data modalities are used?	Affects Processing KPIs
	R4.3 Are there control loops in the scenario?	
	R4.4 Is automation included in the scenario?	
	R4.5 Is actuation included in the scenario?	
Big Data	*R5.1 Is the data available?*	Affects Observation KPIs
	R5.2 Is the scenario scalable?	Affects Processing, Network and Observation KPIs
	R5.3 What level of privacy consideration does the scenario require?	Affects data confidentiality, security KPI on Observation Point and Network categories

References

[1] CityPulse Project Proposal: “Real-Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications”, https://mobcom.ecs.hs-osnabrueck.de/svn/CityPulse/Proposal/CityPulse_Proposal_final.pdf

[2] S. Kolozali, D. Puschmann, A. Karapantelakis, H. Liang, D. Kümper, T. Iggena, M. I. Ali, F. Gao, Deliverable D.3.1: "Semantic Data Stream Annotation for Automated Processing", September 2014. http://www.ict-citypulse.eu/page/sites/default/_les/citypulse d3.1 v1.3.pdf

[3] David Beckett and Tim Berners-Lee, “Turtle – Terse RDF Triple Language”, online resource, visited at Thursday 26 February 2015, http://www.w3.org/TeamSubmission/turtle/

[4] SAO Ontology, http://iot.ee.surrey.ac.uk:8080/resources/ontologies/sao.ttl

[5] MUO Ontology, http://idi.fundacionctic.org/muo/muo-vocab.owl

[6] UCUM Vocabulary, http://idi.fundacionctic.org/muo/ucum-instances.owl

[7] Athanasios Karapantelakis, “CityPulse MUO Vocabulary Extensions”, http://iot.ee.surrey.ac.uk:8080/resources/ontologies/citypulsemuovocab.rdf

[8] Jerry R. Hobbs, Feng Pan, “Time Ontology in OWL”, http://www.w3.org/TR/owl-time/

[9] W3C, Semantic Sensor Network Ontology, http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[10] Feng Gao, CT Ontology, http://www.insight-centre.org/citytraffic

[11] Insight Centre for Data Analytics, https://www.insight-centre.org/

[12] CityPulse Dataset Collection: A collection of semantically annotated datasets for Smart Cities from the CityPulse EU FP7 Project, http://iot.ee.surrey.ac.uk:8080/index.html

[13] Creative Commons, “Attributions 4.0 International License”, http://creativecommons.org/licenses/by/4.0/

[14] Y. Sharfranovic, “RFC4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files”, https://tools.ietf.org/html/rfc4180

[15] Athanasios Karapantelakis, “CPA tool, used for datastream generation and playback using the CityPulse information model”, http://citypulse.github.io/cpa/

[16] Citypulse 101 Scenarios, http://www.ict-citypulse.eu/scenarios/