D2.2 1st iKaaS Semantic Data Model, Knowledge Acquisition and Service Provision Toolbox

<table>
<thead>
<tr>
<th>Deliverable Id</th>
<th>D2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable Name</td>
<td>1st iKaaS Semantic Data Model, Knowledge Acquisition and Service Provision Toolbox</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Dissemination Level</td>
<td>PU</td>
</tr>
<tr>
<td>Due date of deliverable</td>
<td>M18</td>
</tr>
<tr>
<td>Actual submission date</td>
<td>M20</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP2</td>
</tr>
<tr>
<td>Organization name of lead contractor for this deliverable</td>
<td>UNIS</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Suparna De, Stylianos Georgoulas</td>
</tr>
<tr>
<td>Partner(s) contributing</td>
<td>All</td>
</tr>
</tbody>
</table>

**Abstract:** 1st report on the specification, implementation and validation of the Universal Data model and mechanisms for knowledge acquisition and service provision.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme, call: EU-Japan Research and Development Cooperation in Net Futures programme, under grant agreement number: 643262
# Contents

1  Introduction .................................................................................................................. 4  
   1.1  Motivation and Scope of the document ................................................................. 4  
   1.2  Objectives .............................................................................................................. 6  
   1.3  Structure of the document ..................................................................................... 6  
2  iKaaS Semantic Data Model ....................................................................................... 8  
   2.1  Requirements for data model components ........................................................... 8  
      2.1.1  User data model requirements ........................................................................ 8  
      2.1.2  Service data model requirements .................................................................... 8  
      2.1.3  Platform data model requirements .................................................................. 8  
   2.2  Data Model Components ...................................................................................... 9  
      2.2.1  City Data Model ............................................................................................. 9  
      2.2.2  Health Data Model ....................................................................................... 11  
      2.2.3  Access Rights Data Model ............................................................................. 12  
      2.2.4  Virtual Entity Data Model ............................................................................ 14  
      2.2.5  Platform Data model .................................................................................... 19  
      2.2.6  Service Data Model ..................................................................................... 22  
      2.2.7  Knowledge Data Model ............................................................................... 27  
      2.2.8  User Data Model .......................................................................................... 30  
      2.2.9  Mapping of data model components to iKaaS architecture and use cases ....... 32  
3  Data Processing and Knowledge Acquisition for Facilitating Service Provision ........... 35  
   3.1  Toolbox for Data Processing and Knowledge Acquisition Mechanisms .................. 35  
      3.1.1  Complex Event Processing ............................................................................ 39  
      3.1.2  Data Collection and Annotation from Mobile Devices ..................................... 42  
         3.1.2.1  Orientation calibration ............................................................................ 44  
         3.1.2.2  Detection ................................................................................................. 44  
         3.1.2.3  Communication ..................................................................................... 46
3.1.2.4 Pre-processing .................................................. 47
3.1.2.5 Inference....................................................... 52
3.1.3 Data Processing and Knowledge Generation for
Environmental Data .................................................. 52
   3.1.3.1 Pollen and Pollutants Density and Flow ............ 52
   3.1.3.2 Rainfall Prediction........................................... 55
   3.1.3.3 Anomaly Detection ........................................ 57
3.1.4 Data Processing and Knowledge generation on User Data
   67
   3.1.4.1 Health Monitoring ............................................ 67
   3.1.4.2 Prediction of user preferences on device configuration 71
   3.1.4.3 Home automation ............................................. 72
   3.1.4.4 Personalized recommendation on optimal route to
             reach a certain destination .................................. 73
3.2 Mapping of data processing tools to iKaaS architecture and
data model.......................................................... 74
3.3 iKaaS Advances and Innovations .................................... 75
4 Conclusions .......................................................... 79
5 List of Figures .......................................................... 80
6 List of Tables .......................................................... 81
7 List of Abbreviations .................................................. 82
8 List of References ...................................................... 84
1 Introduction

1.1 Motivation and Scope of the document

Semantic modelling as the foundation of a unified data model provides a basis for interoperability among different data sources and applications across different cloud platforms by providing structured, homogeneous, machine understandable and processable annotations. Current developments in semantic data models and data standards have been analysed in the previous deliverable D2.1 [1] (section 2.1), leading to an elicitation of relevant requirements for the iKaaS data model components (D2.1, section 3.1) as well as a first design for the components of the data model (D2.1, section 4). The iKaaS data model has been designed as a set of interacting concepts, each with its own definition of relevant properties and corresponding to the application and service domains supported by the iKaaS project. The supported service domains include the environment sensing systems (CESP (Community Environment Sensing Platform), Madrid system), healthcare support systems (healthcare support system in Tago-nishi, ambient assisted living) and the Tago-nishi town management system. Privacy concerns in each domain are also addressed.

This deliverable provides an update of the iKaaS data model components and their corresponding requirements. The components’ features have been updated to reflect the progress on standardisation as well as identification of new requirements. The standardisation aspect relates to the developments in the HL7 Fast Healthcare Interoperability Resources (FHIR) draft [2], which has a larger variety of potential value types for observations for the health domain and the recognition that it would serve as a better foundation for the health data model than the IEEE11073PHD DIM. The data models with additional identified requirements include the user data model which takes into account the representation of trust relationships between users and the service data model that now features the deployment configuration support in terms of its platform requirements. Analogously, a Platform data model concept has been added to the iKaaS unified data model to capture descriptions of available Platforms for service deployments in multi-cloud environments. Where appropriate, the individual data model components include properties to link to other components within the
unified model. For example, the Service data model allows linking a service instance to the relevant User model instance (where the User is more fully described) through the `<iKaaS:isUsedBy>` property to denote the user of a service. This avoids duplication of information, enables reusability of ontologies and fosters modularity. Figure 1 shows the relationships between the various identified data model components.

Figure 1: Component links in the iKaaS data model

The second main focus of this deliverable is the data processing and knowledge acquisition mechanisms that can facilitate provisioning of services in the iKaaS application domains. The identified data processing mechanisms being developed range from those that allow acquisition of real world Internet of Things (IoT) data to functionality for data analytics, learning, knowledge inference and reasoning to support autonomous behaviour and self-adaptation of applications. Further to the identification of the requirements for the data processing mechanisms and the specification of the targets and scope of these in D2.1 (sections 3.2 and 5, respectively), this deliverable
delves into the details of the various mechanisms, including specification of the underlying algorithms and the involved data flows.

1.2 Objectives

The iKaaS project aims to develop a platform that integrates IoT, big data analysis and privacy management in multi-cloud environments. Fundamental to the development of technologies that integrate big data and IoT in cloud environments is the data perspective. This in turn requires a structured, homogeneous and machine-interpretable view of the data that can be collected from heterogeneous data sources in the IoT and stored in cloud platforms. The iKaaS project seeks to address these aspects with a unified data model represented through Semantic Web technologies. The developed Semantic Web-based data model components form the basis for the data analysis components that enable the notion of knowledge-as-a-service. Some of the major objectives of the semantic data models and data processing mechanisms in the project are summarised as follows.

- To create a semantic data model based on the existing efforts on using semantic technologies in describing various conceptual IoT entities.
- To develop a comprehensive and lightweight semantic model, maximising compatibility and reuse and enabling unambiguous description of relevant information.
- To provide a model that is independent of any particular service technologies.
- To combine data management, information discovery and knowledge acquisition mechanisms for context-aware service provision.
- To support the knowledge acquisition lifecycle, from aggregation of heterogeneous data to the derivation of knowledge.
- To support automated and appropriate deployment of services, taking into account application requirements and cloud resources.

1.3 Structure of the document

The rest of the report is organised as follows: section 2 presents the details of the iKaaS data model components, with the updated requirements from the previous deliverable D2.1 first enumerated in section 2.1, followed by the descriptions of the individual data model
components in sections 2.2.1 through to 2.2.8. The mapping of the presented data model components to the iKaaS architecture elements and their use in the iKaaS use cases is presented in section 2.2.9.

Section 3 presents the different data processing and knowledge acquisition mechanisms being addressed in this project to enable service provision in the identified smart city domains. It begins by outlining the need for distributed processing in the context of the IoT (distributed data sources and collection) to meet the data processing goals while leveraging on cloud computing infrastructure tools. The following sub-sections detail the various mechanisms developed: complex event processing (CEP) (section 3.1.1), data collection mechanisms from mobile nodes (3.1.2) and data processing and knowledge generation for environmental data (3.1.3) and user data (3.1.4). The mapping of the developed data processing mechanisms to the iKaaS architecture components and to the data model components is presented in section 3.2. Innovations proposed and developed in the various data processing mechanisms are outlined in section 3.3. Section 4 concludes the report and discusses the future work.
2  iKaaS Semantic Data Model

2.1 Requirements for data model components

The requirements for the different components of the iKaaS data model have been listed in D2.1, along with a mapping to the project objectives. This section provides an update of the requirements, where these have been identified, for instance, addition of new requirements to the user and service models and a set of requirements for the Platform data model component that has been added to the iKaaS data model. The requirements listed in D2.1 remain valid.

2.1.1 User data model requirements

Table 1: User data model requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Description</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-11</td>
<td>Functional</td>
<td>The user must be associated with particular preferences/policies with respect to other users</td>
<td>Obj_2</td>
</tr>
</tbody>
</table>

2.1.2 Service data model requirements

Table 2: Service data model requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Description</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-16</td>
<td>Functional</td>
<td>A service must be able to specify the minimum requirements for its deployment in terms of the needed platform configuration (e.g. OS, storage, memory etc.)</td>
<td>Obj_1, Obj_3</td>
</tr>
</tbody>
</table>

2.1.3 Platform data model requirements

Table 3: Platform data model requirements
### ID | Type | Description | Objective
--- | --- | --- | ---
P-1 | Functional | Platform must offer the possibility of isolating services based on privacy and regulations affecting to data. A service to process private information can be only deployed in locations allowed by applicable regulation. | Obj_2, Obj_3
P-2 | Functional | The platform should support deployment of services based on CPU architectures and operating systems | Obj_1, Obj_3
P-3 | Functional | The platform should support management and allocation of runtime resources | Obj_1, Obj_3
P-4 | Functional | The platform should provide a mechanism to monitor performance of each component. | Obj_1, Obj_3
P-5 | Functional | The platform description semantic model must be independent from run-time technology, to be future proof and address as many platforms as possible. | Obj_1

### 2.2 Data Model Components

#### 2.2.1 City Data Model

The City data model defines knowledge related to geospatial features, geometry of features for UI of graphical representation for applications, and mechanisms of geospatial referencing for other data models. Overall instances or part of model concepts can be used so as to describe data related information for the VE Model that includes the location feature.

The structure of the data model is based on the well-defined XML based geospatial data model called CityGML published by Open Geospatial Consortium (OGC). The data model has been modified for
application with necessary objects, and defined extended CityGML specification which is called Application Domain Extensions (ADE). An overview of the city data model structure is shown in Figure 2.

A Mechanism for geospatial referencing is based on existing ISO standard spatial referencing by coordinates, spatial referencing by geographic identifier, and other concrete standards. City data model provides geospatial attributes to referencing data models through the mechanisms. Other data models shall define linking attributes to root class of the city data model.

Figure 2: Overview of the city data model structure
City data model as a part of the iKaaS data model shall be converted from CityGML to RDF to be enhanced with additional semantic information.

Geometry of the city data model can be used for representation of 3-dimensional geospatial objects for applications such as a virtual reality application package called Unity.

### 2.2.2 Health Data Model

The health data model is largely based on HL7 FHIR resources. Key resources identified for the use cases are Observation, Questionnaire, Device, and Patient. An “Alarm” resource is being planned in FHIR but is not publicly available at time of writing, and it is likely that one will be modelled within the project for iKaaS purposes. As FHIR is still a draft standard, other changes may also be implemented before deployment. Although a draft standard, it has been well-received, already includes some resources for structured interpretation of the data, and although a communication standard, the resources are well thought-out.

The data model is linked to the city data model for locations, and to the user data model for user information. At least the following additions to the resources will need to be made for use in the iKaaS platform:

<table>
<thead>
<tr>
<th>Observation (DomainResource)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.hl7.org/fhir/observation.html">https://www.hl7.org/fhir/observation.html</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ikaas_code : CodeableConcept [1..1] &lt;&lt;internal code &gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>location[x] : point</td>
</tr>
</tbody>
</table>

Observation ikaas_code: LOINC codes\(^1\) as described in the FHIR standard do not include all iKaaS use cases. Mappings between iKaaS, LOINC, and in IEEE11073PHD nomenclature should be established and documented when possible for interface purposes.

---

\(^1\) LOINC is a commonly used code system for physiological measurements. It is primarily intended for observations in clinical practice or laboratories (e.g. blood pressure, blood glucose), and covers only part of the interesting observations from a home environment (e.g. activity/inactivity are not covered). [http://loinc.org/](http://loinc.org/)
Observation location: Each observation may be linked to zero, one (point), or multiple locations (route). Locations can be linked to the city data model.

The iKaaS health data model presented in D2.1 was based on IEEE11073PHD, with minor expansions. FHIR is also built for use with this standard. As such, this transition should not cause delay. The requirements presented in D2.1 section 3.1.2 remain intact.

2.2.3 Access Rights Data Model

Figure 3 depicts the Access Rights data model that can be used in the context of iKaaS platform so as to model data with respect to the access rights. There are two core concepts that are associated with the access rights root node, a) the “Privacy Certificate” and the ”User Role”.

![Access Right data model](image-url)

Each local cloud has the security gateway at the connection point with the global cloud. The queries from applications and the data on the local cloud are all exchanged through the security gateway. Before asking the local cloud for data, the application is required to be issued a privacy certificate by the privacy Certificate Authority (CA) of the same country, and then must obtain a token from the security gateway.

Table 4 includes the description of the corresponding parameters that are listed into the Privacy Certificate, as well as the description of the
iKaaS user association with the access rights model, through the user role.

Table 4: Access Rights model description

<table>
<thead>
<tr>
<th>Access Rights concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy Certificate</td>
<td>The privacy certificate makes it possible for the security gateway to interpret the rules in the country where the application exists. The privacy CA is built for each country and creates the privacy certificate on the basis of the national regulations that prevail in that country and information concerning the application. The privacy certificate has associations with the CA Country, with one or more LC Countries as well as with a particular application. In addition, it includes the expiration date property that refers to the expiry date of the privacy certificate and the signature that is generated with the private key of the privacy CA. The public key is distributed to security gateways.</td>
</tr>
<tr>
<td>CA Country</td>
<td>The name of the country where the privacy CA is established. It also refers to the name of the country where the application exists.</td>
</tr>
<tr>
<td>LC Countries</td>
<td>The names of the countries that the application is permitted to access.</td>
</tr>
<tr>
<td>LC Data IDs</td>
<td>The identifiers indicate the types of data that the application is permitted to access. The values are nested in each value of LC Countries.</td>
</tr>
<tr>
<td>Application</td>
<td>This concept includes all appropriate properties that refer to the application, by including: a) Application IP that refers to the IP address of the application, b) Application ID that define the type of service that the application provides and c) the Application PK that is the public key of the application.</td>
</tr>
<tr>
<td>User Role</td>
<td>This entity defines the role of the user in the system. In particular, the user can have different roles into the iKaaS system that are associated with particular access rights. Indicatively, some roles for the users that could be</td>
</tr>
</tbody>
</table>
Further to the above information that are associated with the Access Rights model, the role of the Issuance of Token function should be highlighted, which actually exploits the above information through an appropriate API. In particular, this function is called when the application obtains the token to access local cloud DBs. The application is then required to specify the data IDs that it wants to have access to and to send the privacy certificate. When issuing a token, the security gateway refers to the privacy certificate in order to comply with the rules of the country where the application exists. A token is transmitted to the application along with its expiry date after being encrypted with the value of the parameter Application PK listed on the privacy certificate. The application is required to decrypt the token with its own private key. When the application transmits the query to the local cloud DBs through the security gateway, the token is treated as a common key of a message authentication code.

2.2.4 Virtual Entity Data Model

The Virtual Entity (VE) model is presented in Figure 4 including all appropriate concepts and properties for the description of virtualized entity that belongs to the real world. The aim of the VE model is to describe the virtual entity that arises as the result of the semantic based software virtualization of an ICT device (e.g. sensor, actuator) that in its turn may or may not be associated to a non-ICT entity (e.g. people, building, plant, etc.). The identified concepts are associated with each other through particular properties that in terms of semantics are called “predicates”, while the participated entities are called “subject” and “object” by following the forward direction of the arrow from the one concept to the other, respectively.
Table 5 presents the details of the concepts and the properties that are included in the VE Model.

Table 5: VE Model description

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Entity (VE)</td>
<td>The root node that introduces appropriate information with respect to the real world entity that is virtualized so as to be part of the IT platform (namely the iKaaS platform in our case). It includes various properties that indicate a unique identifier, a name, the short description and some additional features such as the mobility capabilities for the entity.</td>
</tr>
<tr>
<td>ICT Object</td>
<td>This concept describes the real world device that supports information and communication capabilities (ICT) and is being</td>
</tr>
<tr>
<td><strong>Non-ICT Object</strong></td>
<td>This concept may refer to a real world entity that does not embed ICT capabilities, such as a person, a building, plant, etc. The ICT object is associated to the non-ICT object bridging implicitly the last with the virtual/digital world (e.g. people do not embed ICT capabilities, but through wearable device -smart watch- an IT platform may have particular information on people’s health status).</td>
</tr>
<tr>
<td><strong>Physical Location</strong></td>
<td>The physical location includes particular information in terms of geographical coordinates for the real location where an ICT object is installed and a non-ICT object is located. In the case of a mobile ICT and consequently a non-ICT object, the data of the physical location coordinates may be updated periodically including the real-time geographical coordinated of the entity. It should be clarified that the physical location of the ICT and the non-ICT object may not be always the same (e.g. person who wears a smart watch on his/her hand and the watch (ICT object) follows the person (non ICT object), but in some situation the location may differs. Indicative example is the one that refers to a Security Camera (ICT) that monitors the outdoor place of a smart home (non-ICT object) and detects the presence of persons/animals/cars/etc. In this</td>
</tr>
<tr>
<td>Parameter</td>
<td>A more generic concept that describes information that may correspond to external ontological concepts, covering a variety of properties that correspond to the ICT devices. An indicative use of this concept would be its use for the description of particular hardware features and capabilities that are being virtualized through the VE. For instance this concept would be able to include information from an ontology that describes a temperature sensor from the vendor ‘X’, as well as information from an ontology that describes the smart phone or a smart watch.</td>
</tr>
<tr>
<td>VE Function</td>
<td>The VE Function concept describes the functional capabilities of the virtualized device. Essentially, it focuses on the description of the interaction among the VE and the Device. It describes how the VE (namely the software) interacts with the Hardware device in terms of the inputs and the outputs that is coming from the device to the VE and the outputs that are forwarded form the VE to the device, respectively. There are two main concepts that complement this concept and describe the Input and the Output</td>
</tr>
<tr>
<td>VE Function Feature</td>
<td>This concept enables the semantic-based cognitive capabilities for the iKaaS platform mechanisms, in terms of the dynamic decision making for the selection of the most appropriate devices/VE for addressing a particular Service composition request. More specifically, VE features facilitate the VEs evaluation and are related to aspects such as VE performance, security level, energy consumption, induced network latency, etc.</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input</td>
<td>This concept describes the appropriate information that is required for the end-user of the VE function so as to get access on a logic location (URL, IP:PORT, whatever appropriate) so as to fetch data as input from the real device to the VE. Indicatively in case we consider that as real device we have a temperature sensors that is installed on a WSN gateway (e.g. Arduino) then this concept will provide information about the endpoint (e.g. URL) where some end-user or functional component can get data from this device. Additionally, in case we have a homogeneous data storage system such as the Local Cloud DB in iKaaS, this endpoint URL can correspond to the LocalCloudDB URL for the data fetching of the specific record</td>
</tr>
</tbody>
</table>
that corresponds to the temperature sensor.

**Output**

This concept describes the interaction from the VE to the **ICT Object**, including the data that is fetched from the VE by a sensor and is being stored in a database, or by referring to the endpoint that an actuator device listens and waits to receive some actuation command (e.g. a URL that pass a ON/OFF command to a lamp).

**Access Right**

This concept corresponds to the instantiation of the access rights model, that is described in detail in section 2.2.3.

**Billing Cost**

The billing cost concept may refer to accounting information with respect to the charging of the VE use, in case of future industrial use. The billing cost can refer to external ontologies that address billing/accounting concepts, and it does not embed any particular property in the scope of the VE model.

**User**

This concept points to a particular instantiation of the user model that is described in the section 2.2.8.

---

### 2.2.5 Platform Data model

The Platform Data Model is an abstraction to describe platform capabilities and requirements of services in order to match both and perform automatic deployment of a service among the different platforms in a multi-cloud environment. In the context of the iKaaS data model, the Platform concept is used to describe the platform capabilities of a cloud. In other words, one instance of the Platform model describes one cloud platform.
iKaaS concept defines an architecture based on Local Clouds and Global Cloud. Local cloud comprises the sufficient or appropriate capabilities in terms of computing, storage, virtualization and networking to fulfil the needs of users in a specific geographic area. Whereas Global Cloud is seen in the traditional sense of a construct with on-demand processing power and storage capability (iKaaS D4.1 [6]).

There have been previous projects working in this topic, such as the EU FP7 IoT.est project [3], which built the basis of an environment for service creation and testing. IoT.est developed a semantic data model for IoT services which aims to assist automatic service creation and adaptation based on Service Level Agreements, service requirements and platform capabilities. The idea there is that once an IoT service is described using the model; it needs to be deployed in runtime environments. In order to optimize this process by choosing the more appropriate platform the system has to be able to match the requirements of the IoT-based Service with the capabilities of the available platforms. Once the service has been deployed, it has to be monitored based on Service Level Agreements to check if performance is within admissible thresholds.

In IoT.est a platform is described as collection of platform resources, each of them owning a set of characteristics describing its capabilities. Also, the individual links in the network are modelled by defining the Link class to represent the concept of individual connections between platforms and hiding the complexity of the network. For more details, please refer to IoT.est documents [4] and [5].

As described before, in iKaaS each cloud works for a specific geographical area, this means that each cloud can be affected by different laws regarding data and knowledge privacy and sharing.

Also, each platform can be owned by different legal entities, which introduce other variables regarding privacy laws. This brings the need of introducing in the model those essential characteristics that have to be analysed in the procedure of automatic service deployment, such as where it is physically located, to which area it is providing service and the legal owner.

Finally, a local cloud can move from a place to other and affect the serving location, or not. An example of this nature can be a Linux
server on a bus; if this Linux server is intended to offer services to passengers it can be said that the serving location does change while the regulation applicable does not change, another case is when the bus moves to a place under different regulation. Although this case is out of the scope of the project, our intention is to provide the basis for people interested in continuing this work, in this sense a new resource class is introduced to characterize platform resources by Type that can be used to deal with this kind of special cases.

The Platform data model with its constituent properties is depicted in the following figure:

**Figure 5: Platform Data Model**

Table 6 presents the details of the concepts and the properties that are included in the Platform Model.

**Table 6: Platform Model description**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>The available amount of memory in a platform</td>
</tr>
<tr>
<td>Processing Unit (CPU)</td>
<td>The computing resources in a</td>
</tr>
<tr>
<td>Description</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Storage</td>
<td>defines the available storage capacity available</td>
</tr>
<tr>
<td>Operational System</td>
<td>The operational system of the platform</td>
</tr>
<tr>
<td>Software</td>
<td>The software installed in the platform that can be used by services</td>
</tr>
<tr>
<td>Execution environment</td>
<td>The available execution environments in the platform, like application servers, running machines (i.e. java), etc.</td>
</tr>
<tr>
<td>Deployment Technology</td>
<td>The technology used in the platform to deploy new services</td>
</tr>
<tr>
<td>Processor Architecture</td>
<td>The architecture of computing resources, for instance 64BITS architecture</td>
</tr>
<tr>
<td>hasResource</td>
<td>Platform property to describe the relation between a platform and a resource</td>
</tr>
<tr>
<td>hasPart</td>
<td>Platform property to describe the relation between two platforms</td>
</tr>
<tr>
<td>isPartOf</td>
<td>Platform property, it is the inverse of hasPart</td>
</tr>
<tr>
<td>Link</td>
<td>Defines the individual links of a platform in a network</td>
</tr>
<tr>
<td>HostingLocation</td>
<td>The location where the resource is hosted or deployed</td>
</tr>
<tr>
<td>ServingLocation</td>
<td>Defines the area that a platform serves</td>
</tr>
<tr>
<td>Owner</td>
<td>The owner of a platform resource</td>
</tr>
<tr>
<td>Type</td>
<td>Defines a new characterization of platform resources that can be introduced in the future</td>
</tr>
</tbody>
</table>

### 2.2.6 Service Data Model

The iKaaS Service Model, shown in Figure 6, constitutes an abstract Semantic Model that can cover the modelling requirements for heterogeneous types of services in the context of the iKaaS platform.
As described in deliverable D2.1, the model comprises of one super-class named ‘Service’, complemented with its first level concepts and properties, as well as by two sub-classes the ‘Simple Service’ and the ‘Complex Service’ that inherit the features of the super-class ‘Service’ and they have some additional specific features. The description of the service model concepts is presented in Table 7.

At this point it should be highlighted that some improvements have been performed on the model in comparison to the model that was presented in D2.1. The most important improvement relates to the association among the service with the platform and resource concepts that are described in the platform model (section 2.2.5). Specifically, the “cloudDeploymentRequirements” property has been added to the Service model to denote the relation between a Service class and its cloud platform requirements. The Platform class introduced in the preceding section describes a cloud platform in terms of its Resources, i.e. processor architecture, operating system, supported execution technology, memory, storage etc.), together with scalar values and units of measurement for Resource classes such as Memory, Storage and ProcessingUnit. The same concepts can be used to describe the deployment requirements of a Service on a cloud platform, expressed in terms of a Platform class instance. It denotes what a service needs to be available on a cloud platform for
its deployment and results in an instantiation of the Platform concept (in terms of the required Resource values). The basis for reusing the Platform concept to denote service cloud deployment requirements is that it enables semantic matchmaking of service requirements against available platform descriptions at the time of deploying a service. To determine the most appropriate platform able to support service execution, platform resources that do not match the service requirements can be automatically filtered out using semantic matchmaking and the remaining platforms whose descriptions match the service requirements can be determined to be candidates for service execution.

The service requirements will be specified by the service developer. It is envisaged that the three most common resource requirements, i.e. memory, CPU and storage, need to be mandatorily instantiated through the appropriate Resource classes of the Platform model, in order to denote service deployment requirements. The rest of the platform resource specifications are optional in the data model.

Table 7: Service Model description

<table>
<thead>
<tr>
<th>Service Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service</strong></td>
<td>This concept indicates the root node of the service model and includes a unique identifier as a URI for the corresponding service. It has various common concepts for the subsets of the services that inherit the service features, such as the logical location, the platform, etc. Each service can be classified either as simple or as complex service and depending of its category, embeds particular features that are presented as concepts.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>This concept describes all appropriate data that refer to the user that consumes the service. As meta-data, container can include additional external features from existing ontologies so as to allow the extension of specific features for the user class. The latest is allowed by introducing the key-pair sets type-value for the introduction of external references to existing ontologies.</td>
</tr>
<tr>
<td>Concept</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Platform</td>
<td>This concept describes the service deployment requirements, further described in terms of constituent Resources, which encompass Memory, Processing Unit (CPU), Storage, Operating System, Software, Execution environment, Deployment Technology and Processor Architecture. Some of these classes, such as Memory, Storage and Processing Unit, are further described by specifying their scalar values and units of measurement. For scalar values a “hasValue” property and their sub-properties (i.e., ‘hasAverageValue’, ‘hasMaxValue’, ‘hasMinValue’, and ‘hasAvailableValue’) are defined.</td>
</tr>
<tr>
<td>Logical Location</td>
<td>This concept refers to the logical location in terms of a URL or any digital access indicator on the software implementation of the service, such as some endpoint, complemented by a particular set of properties provided by some API for the service.</td>
</tr>
<tr>
<td>Simple Service</td>
<td>This concept essentially constitutes the instantiation of the root node for the simple service that is complemented by a unique identifier, expressed as URI. Each simple service is associated with one or more VE functions that actually support the service functionality.</td>
</tr>
<tr>
<td>VE Function</td>
<td>This concept includes a set of different URIs that link to different VE Functions that compose the Simple Service. Each URI leads indirectly, through the reverse association of ‘offersFunction’ named ‘isOfferedBy’ to a particular instance of a VE that is represented by the VE Model.</td>
</tr>
<tr>
<td>Cloud Deployment</td>
<td>The information/data provided by this feature enables the dynamic Decision Making regarding the deployment of Services component on the Global and distributed Local Clouds, taking into account particular functional and system features described by the ‘Type’ and ‘Value’ properties. A service may have one-to-many such features.</td>
</tr>
<tr>
<td>Domain</td>
<td>The ‘Domain’ concept presents the domain, for which the Service has been built so as to support</td>
</tr>
<tr>
<td>Concept</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Physical Location Area</td>
<td>This concept presents in detail the physical area in terms of geographical coordinates that is considered as the area of coverage for a specific domain. It includes different points that in their combination produce a particular shaped location in the physical world, such as a square, a triangle, etc. An indicative example can be presented in terms of a smart home domain that as physical location area can be considered the overall place that is covered by the home building.</td>
</tr>
<tr>
<td>Parameter</td>
<td>This concept works as the meta-data container for the description of extra features either for the service or for the Domain, so as to allow the introduction of additional properties and features that are not included as first level properties into the service model as well as in order to allow the introduction of external, existing ontologies features. Each parameter can be described by a particular name in textual format as well as can include many feature sets as key-pair values for the type – value properties that are included into the concept.</td>
</tr>
<tr>
<td>Complex Service</td>
<td>The ‘Complex Service’ concept essentially constitutes the instantiation of the root node for the complex service that is complemented by a unique identifier, expressed as URI. Each complex service is associated with more than one Simple Services that actually compose its features to provide more complex capabilities.</td>
</tr>
<tr>
<td>Service Logic</td>
<td>The ‘Service Logic’ concept refers to the workflow of the complex service. The service logic is associated with complex service by a specific property that is referred as ‘executesServiceLogic’. The service logic is combined by different compound conditions and</td>
</tr>
</tbody>
</table>
### Compound Conditions

This concept describes the sets of the conditions that are included in the service logic and are associated one-to-one with a particular action.

The ‘Compound Actions’ concept includes the sets of the different Actions that can be included in the service logic and can be triggered by one and only one particular condition.

### Condition

The ‘Condition’ concept is uniquely identified by a URI and includes the key-pair feature for the type-value association of particular properties that can be used for the description of the condition. In general terms, the types of the condition could refer to a particular class (e.g. VE_Function), to a particular logical expression and to a particular condition filter, such as a threshold. Indicatively, through the combination of the condition type-value properties could arise an expression for the condition as the following: `<<VE_Function>> : Sensor_Measurement> Threshold`

### Action

The ‘Action’ concept is uniquely identified by a URI and includes the key-pair feature for the type-value association of particular properties that can be used for the description of the action. In general terms, the types of the action could refer to a particular class (e.g. VE_Function), to a particular class property and to a particular action filter, such as a state action. Indicatively, through the combination of the action type-value properties could arise an expression for the action as the following: `<<VE_Function>> : State = OFF`

## 2.2.7 Knowledge Data Model

This section presents and describes the iKaaS Knowledge data model, and gives an indicative example of the instantiation of the Knowledge data model. In particular, Figure 7 presents the high-level overview of the graph-based representation of the Knowledge data model that can be used for the development of the corresponding Ontology for
the description of its concepts and their relations, as well as it can be used as the base for the deployment of RDF Graph representations for the Knowledge representation.

Figure 7: iKaaS Knowledge data model

The structured data as RDF Graphs that actually correspond to knowledge representations, can be stored in RDF Stores. In the iKaaS platform, based on the first implementation approach, the RDF Store that will store the RDF Knowledge representation data, will be the Global Knowledge DB and the Local Cloud DB. Table 8 presents the description of the corresponding parts that are involved in the Knowledge data model.

Table 8: Knowledge data model description

<table>
<thead>
<tr>
<th>Knowledge Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Root node</td>
<td>The ‘Knowledge Root Node’ presents the root node of the iKaaS Knowledge data model that points to a domain specific Knowledge Model, such as the Smart Home or the Smart City Knowledge data model</td>
</tr>
<tr>
<td>Domain</td>
<td>The ‘Domain’ node presents the domain to which the Knowledge data model refers and describes its concepts. Each domain can be described in terms of its type (e.g.: Smart Home, Smart City, etc.), a name and a textual</td>
</tr>
</tbody>
</table>

<ikaas :hasURI> :URI
<ikaas :hasName> :xsd^^String
<ikaas :hasType> :URI
<ikaas :hasFeatureType> :URI
<ikaas :hasFeatureValue> :Literal
<ikaas :hasParameter>
<ikaas :hasObjectFeature>
<ikaas :indicates>
<ikaas :supports>
<ikaas :hasConceptualObject>
<ikaas :isGeneratedBy>
<ikaas :isUsedBy>

<table>
<thead>
<tr>
<th><strong>Domain Parameter</strong></th>
<th>description in natural language. Each domain may include one or more parameters, the ‘Domain Parameters’, which may describe domain specific concepts required for specific situations, such Date &amp; Time, Indoor Location, Status of a process and/or situation, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Knowledge Conceptual Object</strong></td>
<td>The ‘Knowledge Conceptual Object’ concept can be used so as to present the potential Knowledge concepts that can be involved in a specific domain. This concept can be considered as an abstract meta-data container that can include any data-type/concept of external / third-party ontologies and/or data model. In a general approach, it could be considered as the node that can describe the components, which are included in a specific domain (e.g. Smart Home: [Sensors &amp; Actuators, Persons, Places, etc.], Smart City: [Places, Mobile Devices, QR Codes, etc.])</td>
</tr>
<tr>
<td><strong>Object Feature</strong></td>
<td>The ‘Object Feature’ concept works as assistant node to the ‘Knowledge Conceptual Object’ concept and it is considered as part of the Knowledge data model, in order to support the capability for further description of a Knowledge concept, where it is required. Indicatively, the object feature could include information about the available devices that are represented by particular VEs that are indirectly associated with a simple service through the association between the simple service and the VE Functions of the VEs that belong in the corresponding domain that the knowledge data is referred.</td>
</tr>
<tr>
<td><strong>Dataset</strong></td>
<td>This concept refers to the dataset that has been used as part of data for the knowledge generation. The dataset can refer to various different datasets instances, such as data measurements, actuator workflows, application data (e.g. weather data, traffic data, etc.).</td>
</tr>
<tr>
<td><strong>Instance of Service model</strong></td>
<td>This concept refers to the instantiation of a particular service as it is considered in iKaaS and instantiated based on the service model. Actually, it can describe either a</td>
</tr>
</tbody>
</table>
2.2.8 User Data Model

The User Model is a general abstract representation of users that includes a set of different concepts/entities. These concepts/entities are defined in a way that gives the ability for an abstract representation of user data. These are defined so as to cover the need for modelling of ‘static information’ as well as ‘dynamic and not specific information’ that can be related with the user and can refer to different concepts. Thus, the User Model has to provide an abstraction of user data and it works as a high-level abstraction model.

The users in general can have different sets of data that represent their properties. In particular, static information that is denoted as **User Characteristics** can present some standard characteristics of the user such name, surname, etc. Concepts from Friend-of-a-Friend (FOAF) ontology could be very useful for the description of User Characteristics. FOAF provides a full updated vocabulary that allows the representation of machine-readable data about people, SW agents, Social groups, etc., in the Web. Moreover, it is assumed that the user may have one or more different **User Profiles** that in turn are linked with different **User Preferences** and **User Policies**. Also **User Relationships** with other users are represented. In addition, as is described above, a user can have various and diverse **User Roles** such the owner of a virtual entity, the consumer of a service, etc.
Depending on the User Role, the associations between the entities, in the User Model, can be differentiated because each role has different properties and associations. A user in role of an ‘Observed Entity’ cannot have an association with Service entity as the consumer of a Service. On the other hand the User Policies and User Preferences that are defined by a ‘Service Request’ can be totally different from those that are defined by an ‘Observed User’. For instance a ‘Service Requester’ can provide a set of preferences and policies for the customization of a requested application, whilst an ‘Observed User’ can define preferences and policies that will be used to customize its observations. However, it should be noted that there may be cases in which a user has both roles at the same time. Figure 8 depicts the representation of the User Model diagram, comprising entities associations and their meta-data. In addition, Table 9 presents some relevant information with respect to the main entities that can be included in the User Model.

Figure 8: User Model Entities diagram

Table 9: User Model concepts description

<table>
<thead>
<tr>
<th>User concept</th>
<th>Model description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td></td>
<td>The root node of the User Model that refers to the User that can be either a Human or a Software Agent.</td>
</tr>
<tr>
<td>User Role</td>
<td></td>
<td>This entity defines the role of the user in the system. In particular, the user can have different roles into the iKaaS system that are associated with particular access rights instances that are</td>
</tr>
<tr>
<td>User Relationship</td>
<td>This entity defines the relationships, in terms of levels of trust, that a user has with other users of the iKaaS platform.</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>User Characteristic</td>
<td>An entity that comprises static information regarding the User. Such types of information can refer to the name, surname, age, gender, etc., which constitute standard and stable characteristics for the user independently of his/her profile. (e.g.: [Name = ‘Sample_Name’, Surname = ‘Sample_Surname’, Age=’30’, Gender = ‘Female’, ....]&amp; ([Profile_ID = ‘ID_x’, Profile_Type = ‘Smart Office’] or [Profile_ID = ‘ID_y’, Profile_Type = ‘Smart Home’]).</td>
<td></td>
</tr>
<tr>
<td>User Profile</td>
<td>This entity defines different instantiations of user profiles that can be associated with a specific user. Each distinct user profile is unique for a user and is linked with that user through a unique identifier.</td>
<td></td>
</tr>
</tbody>
</table>
| User Preference | This entity is used so as to allow the expression of user preferences that relate with user activity in the iKaaS system. In particular, the user preferences in case of:  
  - a ‘Service Requester’ can define specific properties for the customization of Services (e.g.: ‘Send me notifications about the traffic status in the Area_X’ or ‘GUI Language = English’)  
  - an ‘Observed User’ can express specific properties for the customization of the observation, (e.g.: ‘change my profile if a set of properties are satisfied’). |

### 2.2.9 Mapping of data model components to iKaaS architecture and use cases

Deliverable D2.1 described the mapping of the data models to the iKaaS architecture components using the models and this is outlined in Table 10 below:
Table 10: iKaaS data model - architecture mapping

<table>
<thead>
<tr>
<th>Component data Model</th>
<th>iKaaS component using the data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>City (Geospatial) data model</td>
<td>Stored in “Local Cloud DB” in RDF format, used by Local and Global data processing. For the exchange of geospatial information, iKaaS extension of CityGML is used.</td>
</tr>
<tr>
<td>Health data model</td>
<td>Local Cloud DB, Global Knowledge DB, Local and Global data processing.</td>
</tr>
<tr>
<td>Access rights data model</td>
<td>Security gateway. Internal data model is not accessible. Only a set of APIs are open for other iKaaS components.</td>
</tr>
<tr>
<td>Virtual entity data model</td>
<td>Global Service Catalogue, Local Service Catalogue, Global Service Manager, Local Service Manager.</td>
</tr>
<tr>
<td>Service data model</td>
<td>Global Service Catalogue, Local Service Catalogue, Global Service Manager, Local Service Manager, Local and Global Data Processing.</td>
</tr>
<tr>
<td>Knowledge data model</td>
<td>Global Knowledge DB, Local DB, Global and Local Data Processing.</td>
</tr>
<tr>
<td>User data model</td>
<td>Global Knowledge DB, Local DB, Global and Local Data Processing.</td>
</tr>
</tbody>
</table>

It is also worth noting that a Platform Catalogue component will be incorporated in the iKaaS architecture to enable the storing of platform capabilities and characteristics representations; the platform data model naturally will be used for these representations.

The data model components also provide the basis of a formal, structured representation of relevant concepts in the iKaaS use cases. The table below maps the use of the different data models in the use cases:

Table 11: iKaaS data model - use case mapping

<table>
<thead>
<tr>
<th>Component data Model</th>
<th>iKaaS use case using the data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>City (Geospatial) data model</td>
<td>Tago-nishi - Town management, Ambient Assisted Living - Smart mobility</td>
</tr>
<tr>
<td>Health data model</td>
<td>Ambient Assisted Living – health monitoring, Tago-nishi – health</td>
</tr>
<tr>
<td>Data Model</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Access rights data model</td>
<td>Tago-nishi - Town management, Tago-nishi - Health support, Ambient Assisted Living</td>
</tr>
<tr>
<td>Virtual entity data model</td>
<td>Ambient Assisted Living, Madrid environmental health</td>
</tr>
<tr>
<td>Service data model</td>
<td>Ambient Assisted Living</td>
</tr>
<tr>
<td>Knowledge data model</td>
<td>Ambient Assisted Living, Madrid environmental health, Tago-nishi - Town management</td>
</tr>
<tr>
<td>User data model</td>
<td>Ambient Assisted Living</td>
</tr>
</tbody>
</table>
3 Data Processing and Knowledge Acquisition for Facilitating Service Provision

3.1 Toolbox for Data Processing and Knowledge Acquisition Mechanisms

In this section we focus on the generics of distributed real time processing contextualising it for the cases in which the data is collected via IoT heterogeneous devices and illustrating how we leverage on cloud computing infrastructure tools for efficient use of available resources while minimising the amount of software coding expected to achieve those data processing goals.

Distributed Real-Time operation stems from the need to accommodate functionality split which relaxes the requirement of having computing-powerful centralised nodes and adds flexibility needed to address latency and privacy problems, but also scalability when delivering IoT data to applications.

Following Section 2.3.2 on state of the art analysis for data processing from D2.1, in this section we delve deeper into a complete toolbox for data processing, selecting few of those technologies with the aim of showing how these different open-source technologies can be used for implementing the iKaaS architectural building blocks. The selection has been done paying attention to whether or not these can leverage on recent cloud advances (i.e. make use of micro-services running over containers for flexible resource allocation and migration) and whether or not real-time operation is supported.

Distributed Real-Time processing systems are made of a number of computing engines connected through communication networks and distributed in a way that minimises the impact of latency, the influence of the various connections bandwidth and accounts for the physical location of data sources. In addition to this "performance-driven" distribution, other requirements concern privacy, leading to functional components having to migrate from a global to a local cloud.

In IoT context, distributed processing provides a strong asset to leverage upon when aiming for wide-scope sensing implemented through a variable number of distributed data-sources. The most
common real-time processing will happen on a stream of data, and for a variety of reasons, which a Complex Event Processing engine might fulfil only some of. Distributed processing extends the role of CEPs (illustrated in more detail further down in the section) as it also gives the option of handling batch processing when real-time requirements are not so stringent.

Here we frame distributed real-time processing into the more specific context of iKaaS functional architecture (see Figure 9 below) illustrating how the iKaaS toolbox can be implemented using existing technologies that can be deployed over cloud infrastructure.

Starting from the data sources on the left hand side of the iKaaS architectural picture, there is scope for Virtual Entities and associated simple edge data processing tasks (like semantic enrichment, anonymization, data format translation / adaptation, data aggregation) to be performed over cloud infrastructure. Such an approach makes iKaaS platform able to deal with a variety of data sources and provides a first checkpoint on the source to destination delivery path, for addressing interoperability problems (such as adaptation of data formats from proprietary and vendor-specific ones to iKaaS compatible data formats).

**Figure 9: iKaaS Functional Architecture**

All services for this first step “close to source” data processing are
registered with the Service Manager and available as part of the Service Catalogue.

It must be noted that different requirements will drive different implementations choices as to what data processing takes place and where this will be executed between Local and Global Cloud.

We address most of these requirements exploiting tools that have already a level of resilience in ensuring proper configuration of nodes and distribution of workload amongst these. Besides the internal reconfiguration capabilities of the used engines, we must also pay attention to what is outside the scope of how a particular tool manages its own resources.

Figure 10: Mapping toolbox for real-time distributed data processing over iKaaS functional architecture
Moving towards the Global Cloud data manipulation requires more complete and powerful engines and sets of interfaces; examples of the latter are filtering on aggregated data streams (stream processing) and big-data analytics algorithms (for batch processing).

Here we see a clear conceptual separation between the use of specific tools for generic data processing (like Apache Flink and, to some extent, Hadoop and Spark) and what we need to ensure if we enlarge the scope of our analysis to include also resources which cannot be managed directly by those specific tools (but rather managed with external orchestrators such as Zookeeper), reported here for completeness but outside the scope of iKaaS.

Figure 10 illustrates in more detail how iKaaS functional architecture is mapped onto a Software Architecture for the purpose of achieving real-time data processing objectives (i.e. scalability, interoperability, robustness etc.) over cloud infrastructure.

For this IoT edge processing there are several iKaaS partner internal assets that we plan to make use of (i.e. enabling the creation of cloud-supported Virtual Entities, the semantic enrichment prior to RDF storage, the mapping to iKaaS supported data models etc.).

Moving towards the right-hand side of the picture, we illustrate open-source cloud infrastructure tools (such as Apache Flink and Hadoop), and partner assets (such as CEP) that enable us to run iKaaS data-processing algorithms on (i.e. for Big Data and knowledge acquisition), trusting the fact that the engines themselves will ensure an adequate use of the underlying infrastructure resources.

Besides the flexibility of stated technologies for automatically (and transparently to the developer) managing own resources, in Figure 10 we also plan to support the scenario where some functionality can be isolated and installed to run on containers (i.e. Docker, Wardens etc.) for added flexibility and scalability. Similar to inner resource management rules of above-referenced technologies, here we also transparently rely on the fact that new containers can be spawned to support increasing processing demand, or to address mobility through which components can be optimally placed or moved around to meet given “fitness functions” (i.e. minimise latency, or use of network resources etc.).

The algorithms that implement the work assignments (usually orchestrators implemented as rules or placement engines) throughout
such a system must take into account intrinsic infrastructure aspects as well as higher-level requirements and produce a design blueprint that, besides achieving the main objective (be it performance optimisation or support for privacy), also accounts for synchronisation and system nodes potential failure.

Having introduced the generics of cloud-supported data processing toolbox and the potential mapping of the associated iKaaS functional architecture components onto software infrastructure tools, we introduce in the next few paragraphs specific examples of tasks accomplished with this framework.

3.1.1 Complex Event Processing

Complex Event Processing is a particular example of real-time data processing, normally used for extracting knowledge from raw data streams, based on simple rules or more complex machine learning algorithms executed using IoT data streams as input.

As introduced in [1], CEP techniques have been used for years as tools for environment awareness and response generation. The combination of CEP with Big Data and Machine learning technologies brings the opportunity to cope with a more wide range of issues but complementary functionality needs to be added to CEP.

As a first step, the selected CEP tool can be included in more different scenarios, and it will be used for different type of users, human (experts and no experts), other services, etc. To solve this issue an interface should be provided which can be used by other services and facilitate the implementation of human interfaces.

This interface should, as a minimum, fulfil with the requirements of CRUD model (create, retrieve, update and delete) in order to manage instances of CEP engines.

As a second step, the CEP system evolves from a simple tool to a Service as far it can be used to fulfil the requirements of simple scenarios or can be combined with other services to afford more complex scenarios. It is needed to semantically model the CEP in order to be exposed and be used by other services. This model has to describe capabilities, functions and every parameter needed to include the CEP into the service catalogue.

These first two steps have been explored in the VITAL FP7 project [7], where a CEP engine has been modelled as a Service and
managed through a REST API interface. The concepts developed in this project will be used to develop the same work in iKaaS.

The following step is to prepare the CEP engine tool to work in a cloud environment, in which it is expected to be widely used and so it has to provide a mean to perform automatic deployment based on performance aspects. The CEP engine will be adapted to the technology chosen to implement the requirements for Cloud in iKaaS.

The steps described above are an abstract of requirements to build a CEP tool as a service, able to work in a cloud environment. But some more functionality is needed in order to combine CEP with Big Data technologies.

To improve the process of knowledge generation at the CEP it is important to add more sources of data beyond those a priori defined. For instance, the values measured by a UV sensor are useful by themselves, but if CEP has access to some metadata of the sensor such as working output thresholds it will be possible to take extra actions when values out of thresholds are received, like discard these measures, raise alarms or launch some off line processes. There are two ways to feed the CEP with metadata of data sources, giving access to CEP to the metadata repository (Service Catalogue) or aggregating the metadata to the input stream of CEP. The first option looks like the more logical way but it has a cost on the CEP performance, and can bring other problems like blocking CEP threads while accessing an external repository. The second option lets to isolate CEP from external systems and does not affect its performance.

The knowledge generation process when combining CEP and Big Data technologies is a way of two directions. The first one is clear, the output of CEP can be stored and processed off line, or it can be used to launch processes or modify behaviours of these processes. This is easy as the output of a CEP is not defined a priori; it depends on the scenario.

In the reverse direction, to use the output from off line processes (Big Data analysis processes, machine learning systems, etc.) depends on the format of the output of these processes and on the means used by the CEP to define the patterns it has to detect.

For instance, a CEP engine deciding when the output of a sensor is valid or not, using the function provided by the vendor:
\[ F(x) = a_0 + a_1 x + a_2 x^2 \]  \hspace{1cm} (1)

Where:

\[ F(x) > 0 \text{ value OK} \]
\[ F(x) \leq 0 \text{ value KO} \]

It is easy to manually define a pattern in the CEP to calculate the formula and evaluate the result.

Applying statistical analysis, it is discovered that the function defining the result in the error is a polynomial function of order \( n \), where \( n \) depends on time in the following way:

- from 00:00h to 06:00h \( \rightarrow n=1 \)
- from 06:00h to 12:00h \( \rightarrow n=3 \)
- from 12:00h to 17:00h \( \rightarrow n=4 \)
- from 17:00h to 00:00h \( \rightarrow n=2 \)

In this situation it is still possible to manually define four patterns, one per time window. But after a second pass of the statistical analysis it is discovered that factors \( (a_0, a_1, a_2, \ldots) \) depend on temperature, therefore it is not possible to define the patterns manually.

A module is required to translate the model provided by data analysis processes into patterns.

At a first sight it is not possible to build a function able to translate any model into patterns, but hopefully the models generated depend on the implementation of processes, and they are known.

This means that it is possible to build a repository of models and their patterns, in order to automatically modify them on the CEP based on changes of the models or aspects of influence, like time and temperature in our example.

At this stage the CEP engine tool chosen to implement this system is the ATOS engine tool (described in [9-10], as it is being used in other projects, in order to leverage the studies and developments done.)
3.1.2 Data Collection and Annotation from Mobile Devices

As discussed in [1], user social relationships and the eventual social trust calculation based on them [8] can prove very useful in a multi-cloud environment because they can determine suitable end-points in a service chain as well as suitable local clouds as a whole for use in a multi-cloud based service chain. For example if two people are deemed as being in a “trusted relationship”, then in the case of a fall or accident of the first person, an alert can be sent only to the other “trusted” person to go assist (endpoint of service chain). Once “trusted” people are identified in a service provisioning chain, then their devices and local cloud resources (if owned) can then be leveraged for service provisioning. E.g. if a person goes to a trusted person’s home, then these home “local cloud” resources can be utilized and have the notion of the “follow me as I move” service depending on the access levels that entities deploying local clouds are willing to grant to other parties based on social trust levels.

As, also argued, social interaction detection comprises two major and important components: a) interpersonal distance estimation and b) relative orientation computation.

Inline with these identified factors, for the context of iKaaS we have been working on developing a social interaction detection enabler as a mobile phone application (VE software intended to run at mobile phone device level) which provides the capability of detecting real-world social interactions [11] (for simplicity reasons we will be referring to it as face-to-face (F2F) enabler). The system utilizes commercial off-the-shelf mobile phones in an opportunistic and collaborative manner to sense contextual information and perform the appropriate inference. The social interaction inference is based on the spatial arrangement among people [12]. To infer the relative spatial arrangement among people the interpersonal distance and relative orientation of the users are estimated. The interpersonal distance of the users is estimated through the Bluetooth Received Signal Strength Indicator (RSSI), where a machine-learning model is trained that is performing the inference. The interpersonal distance estimation is inferring if the users are in proximity and also the social relation of the people based on the estimated distance. The relative orientation is computed by considering the users’ facing direction. The users’ facing direction is computed through uDirect [13], an algorithm that computes users’ facing direction independent of the wearing position. A collaborative scheme that leverages the native
capability of Bluetooth for ad-hoc communication, is also developed that allows the ad-hoc communication among the devices in order for them to exchange the users’ facing direction to compute the relative orientation and the mutual Bluetooth RSSIs to speed up the process of sensing and inference. Figure 11 depicts the information flow in a F2F enabler application while also includes the communication part with nearby devices.

The core components of the F2F enabler application are:

- **Orientation calibration.** In this part the device calibrates its orientation with respect to the earth’s coordinates and then it detects the orientation of the device with respect to the user.
- **Detection.** The device performs Bluetooth scanning to detect nearby devices that are related to the F2F enabler and also tracks the user’s facing direction.
- **Communication.** The device performs ad-hoc communication with the nearby devices that belong to the network of the F2F enabler. During this communication the devices exchange the facing directions of their users and the mutual Bluetooth RSSIs.
- **Pre-processing.** The device estimates the interpersonal distance with the users in vicinity and also computes the relative orientation with each of the users.
- **Inference.** This is the final stage of the application, where based on the interpersonal distance and the relative orientation the device infers if users are socially interacting or not.
In the following subsections a more detailed analysis is provided regarding the design and implementation of the social interaction detection inference and the components it is comprised by.

3.1.2.1 Orientation calibration

The orientation calibration state constitutes the initial stage of the inference process. During this process the device calibrates its orientation with respect to the earth’s coordinates. This constitutes an initial reference that will be used to transform the device orientation with respect to the user, into the users’ orientation with respect to the earth’s coordinates. This step is important as otherwise the users’ orientation that will be computed in the following step will be shifted by a certain offset. This offset is eliminated by the orientation calibration module. This step is performed once when the mobile app is initiated. It should be mentioned that during this process the user is standing.

Having performed the initial calibration of the device orientation with respect to the earth’s coordinates, the application estimates the user’s orientation with respect to the device’s coordinates. During this process the user starts to walk without any restriction, except from the assumption that the user moves forward. Leveraging a human person’s walking locomotion during which different acceleration patterns are applied, the uDirect [13] algorithm estimates the user’s orientation with respect to the device coordinate system. Only the first step of the user needs to be exactly forward, apart from this the user could make manoeuvres, stop walking and then continue again. After a few steps the algorithm converges on a particular orientation of the user with respect to the device.

3.1.2.2 Detection

In this step the device detects the user’s facing direction and the existence of other devices/users in vicinity. This module is one of the core components that operates in a periodic manner, with different periods of time for the user’s facing direction detection and the detection of nearby devices. In this context when we refer to the detection of nearby devices, we assume that each device is associated with one user. For that reason, the detection of a nearby device corresponds to the detection a person in vicinity with which the user may be interacting with.
3.1.2.2.1 Nearby devices

The enabler was developed to facilitate energy efficiency without compromising the user experience. This means that instead of performing continuous sensing and inference, this enabler allows to detect when other devices that are running also the enabler, are in the vicinity. Only when nearby devices are detected, the inference process of interpersonal distance and relative orientation are initiated, in order to finally estimate the existence of an on-going social interaction.

The absolute or the relative position of the devices in vicinity to detect nearby devices is required. Absolute positioning requires continuous monitoring of the user with positioning sensors such as GPS but are very energy consuming and do not operate indoors. Other absolute positioning systems exist but require additional hardware. In order to estimate the relative position of users in vicinity audio, WiFi or Bluetooth signals could be utilized. The audio signals require firmware modifications, time synchronization and additional hardware in order to allow the detection of interpersonal distance. WiFi interface requires also firmware modification in order to adjust the transmission power and to enable the ad-hoc discovery and information exchange among devices. Finally, Bluetooth has the native capability to perform ad-hoc discovery and information exchange without the need to modify the firmware of the device. Also, it provides the ability to retrieve the RSSI which allows the estimation of the interpersonal distance among the devices.

3.1.2.2.2 User facing direction tracking

In the orientation calibration phase, the enabler initially detects the device orientation with respect to the earth’s coordinates. Then it computes the user’s orientation with respect to the device orientation. The sole purpose is to estimate the user’s facing direction with respect to the earth’s coordinates. This is performed by utilising the information extracted in the orientation calibration phase. Having acquired the orientation of the device orientation with respect to the earth’s coordinates and the user’s orientation with respect to the device orientation, the approach maps the user’s facing direction to the earth’s coordinates and then keeps track of the any changes in the user’s facing direction. In that way this module is able to provide an estimation of the user’s facing direction every 4 seconds, by considering only a very simple computation of scalars. This is performed because there is no need to keep track of all the three
dimensions in an Euler angles orientation representation, only the azimuth with respect to the earth’s coordinates is required. As the azimuth, we consider the y axis vertical to the earth. By taking into account the changes in the rotations around the y axis, while considering the device orientation with respect to earth’s coordinate and the user orientation with respect to the device coordinate system, the module keeps track of user’s current facing direction.

3.1.2.3 Communication

The sensing process should be independent of any external hardware in order to allow pervasive observation of social interactions. The sensing and inference period needs to be as short as possible to detect short-time social interactions occurring frequently in daily life situations. To preserve the privacy of individual users, the inference should be performed out locally on the device, without the need of any centralised server or system.

The communication module addresses these challenges by inducing the collaborative sensing module. The devices exchange users’ facing directions and mutual Bluetooth RSSI measurements of users through ad-hoc communication. A discovery process is performed followed by a dynamic ID-based master-slave role assignment for the connection initiation, which will lead to information exchange. As smartphones operate in an ad-hoc mode, every pair of devices detected performs the above process. Each device acquires user's direction in vicinity and the mutual Bluetooth RSSIs via collaborative sensing.

The communication follows the steps below:

1. Discovery. The application performs an inquiry scan for the detection of the devices in proximity. The discovery process has duration of approximately 12 sec.
2. Pick Over. After the end of the discovery process followed by the retrieval of the nearby devices, a selection is performed, where only the devices that belong to the system are kept, the rest are discarded.
3. Role Assignment. For each pair of devices, a role assignment protocol defines the role (master/slave) of the local and each nearby device. The role assignment is based on the ID of the device, where the device with the highest ID is nominated as master and the others as slave.
4. Exchange Information. A device initiates the connection with a nearby device if the assigned role in this relationship is master. When two devices are connected, they exchange their directions and the RSSI from each other. In case of failure while
initiating the connection multiple attempts are performed, mediated by random delay.

### 3.1.2.4 Pre-processing

After retrieving the information from the devices in vicinity, the application goes into the pre-processing step. This step includes the estimation of the interpersonal distance and the relative orientation of the users. Each component is detailed in the following subsections.

#### 3.1.2.4.1 Interpersonal distance

Hall [13] initially identified the importance of interpersonal distance in the human behaviour by monitoring animals’ behaviour and following these observations he focused on humans. Through his experiments he was able to understand at which interpersonal distance humans feel comfortable to socially interact and depending on the interpersonal distance what type of social relationship do the interlocutors have. Following this research, we focused on developing a mechanism that allows the coarse and fine-grain estimation of users’ interpersonal distance. Literature have utilised various techniques on smartphones to estimate interpersonal distance such as Time of Arrival, Time Difference of Arrival, Angle of Arrival for Radio Frequency and audio signals. However, these techniques require firmware modifications in order to operate. Thus, we focused on leveraging the Bluetooth RSSI signal that is provided by the native API of commercial off-the-shelf mobile phone operating systems. Current techniques for interpersonal distance estimation lack of accuracy due to the large fluctuation observed on RSSI and also require a large number of Bluetooth RSSI samples e.g. 20-samples [15].

In order to tackle the aforementioned issues of state-of-the-art techniques a machine-learning model was developed based on Bluetooth RSSI. The technique requires only 6 RSSI samples in order to infer if the users are in proximity and also to detect what type of social relation users have based on [13]. For the social relation detection, a hierarchical approach is introduced that trains a specific model for each social relation and then an overall model infers about the social relation of two persons. Both models are based on machine-learning and in particular on a boosting technique called MultiBoostAB [16]. The models are trained based on an extensive dataset i.e. 48000 Bluetooth RSSI samples and a large bank of extracted features. The models are considering the fluctuation of the
Bluetooth RSSI in the indoor environment and also take into account the human body absorption. It is worth noting, that both models infer about the interpersonal distance with only 6 Bluetooth RSSI, allowing the interpersonal distance inference in short-term contacts.

3.1.2.4.2 Training the models

The interpersonal distance estimation technique relies on initially generating a generic training set and then extracting a bank of features from this training set. A feature selection process is followed in order to choose the most informative and less redundant feature set. Each classifier in each level of the hierarchy is evaluated in order to find the most appropriate choice.

A data collection campaign was performed in an indoor office environment through HTC One S smartphones, in order to construct the training set for the classifier. The Bluetooth interface on one of the smartphones was configured as discoverable and the other device was performing the discovery process. After the end of the experiment, Bluetooth RSSI data were collected from eight different distances, three different device relative orientations. These device relative orientation were a) Screen-to-screen b) Screen-to-Back c) Back-to-Back. Empirical evaluation showed that these vertical relative orientations constitute representative for the effect of the facing direction variation. A large number of Bluetooth RSSI samples was collected i.e. 2000 samples for each different distance and orientation combination, resulting in a dataset of 48000 Bluetooth RSSI samples, for reasons of statistical significance. As the data collection process was extremely lengthy, the humans were replaced with water-filled cylinders to which devices were attached, in order to simulate to human body absorption [17]. The devices with the bottles were placed at 0.8m height from the floor to simulate the most common wearing position (i.e. trousers pocket) [18].
A large feature set of 3050 features including several statistics was extracted from this dataset, considering a maximum window of 6 Bluetooth RSSI samples. The basic features that were extracted refer to various statistics. The relative features which are produced by combining basic features with various statistics of the target class. The combined features combine the basic and relative features through a statistical metric such as deviation and z-score. Such large number of features was generated in order to be confident that the feature reduction techniques will produce the most informative features. Given the level of consistency [19] of each feature based on the target class, a subset of features is chosen. A wrapper subset evaluation [20] followed to retrieve an optimised feature set for the given classifiers. Various evaluations showed that MultiBoostAB [16] with decision tree J48 [21] performed best.

3.1.2.4.3 Relative orientation
Relative orientation is a substantive component in the process of understanding social interactions in a real-world environment [22]. Figure 13 depicts a social situation in which even though participants (A), (B) and (C) are in proximity, only participants (A) and (B) are socially interacting. The participant (C) does not have the appropriate relative orientation in order to interact with participants (A) and (B). The relative orientation estimation is extracted from the facing direction of each participant and is defined as the angle required for a user to turn in order for both users are facing each other directly [24]. An implicit assumption is made, that the participants' direction is the same as their facing direction [22]. One of the drawbacks of a prior work [24] was the requirement for fixed on body position, thus there is a need to remove the dependency of the device on-body...
position to limit the intrusiveness of the system and make it suitable for real-world situations.

Figure 13: Importance of including relative orientation in the social interaction detection process.

To tackle the aforementioned drawback, a state-of-the-art technique for direction estimation was advanced through developing an outlier removal algorithm that considers the error distribution of the approach [13]. The estimation algorithm does not depend on the wearing position of the device. The algorithm is based on two phases. During the first phases, considered as calibration phase, the algorithm computes the relative orientation of the device with respect to the earth's coordinates. In the second phase, considered as direction estimation, the algorithm leveraging on the walking locomotion of the user, estimates the relative orientation of the device with respect to the user. Then, given the calibration and the direction estimation phase, the device orientation is transformed to the user's orientation with respect to the earth's coordinates. This allows the orientation tracking of the user's facing direction. To smooth the results of direction estimations, a low pass filter with smoothing factor $a = 0.5$ was introduced, which increased the robustness of the facing direction estimations.

Given the Gaussian error distribution of uDirect [13], a novel outlier removal technique was introduced to discard erroneous facing direction estimations. The algorithm is a categorisation technique that clusters the direction estimations and selects the most popular cluster. Through a voting approach the most popular cluster is selected based on the number of estimation contained in each
cluster. The values in the most popular cluster are averaged based on the Equation (2). The computed average value of the most popular cluster is considered as the direction estimation. The average of n facing direction estimations is calculated through Equation (2).

$$\text{Mean}(\theta) = \arctan \left( \frac{\sum_{i=0}^{n} \sin \theta_i}{\sum_{i=0}^{n} \cos \theta_i} \right)$$ (2)

The decision about averaging the most popular cluster was taken due to the Gaussian error distribution that characterises uDirect [13]. The above process is performed iteratively until the algorithm converges to a satisfactory facing direction. The satisfactory clause is defined by two sequential facing direction estimations that do not differ by more than 10° i.e. the distance between the two estimations. When then the criteria are satisfied, the two final facing directions are averaged and produce the final facing direction. The 10° distance between two following sequential facing directions was chosen to reduce the error of the magnetometer sensor.

![Figure 14: Detection of the relative orientation of two users.](image)

After the above process, the device orientation with respect to the users’ coordinates has been estimated. Given the initial calibration phase and the above process, the users' facing direction is being tracked. When a user in proximity is detected, then the current facing directions are exchange among the devices. This allows the computation of the relative orientation among the users. At the front part of the users' torsos an imaginary cone of 90° is considered as the appropriate relative orientation in order to perform a social interaction. For two users to participate in a social interaction these two imaginary cones need to overlap (see Figure 14) and also that the two users are in proximity to interact.
3.1.2.5 Inference

The final inference includes the combination of the interpersonal distance and relative orientation result in order to infer the existence of a social interaction with a person in vicinity. A small decision tree has been developed that considers the output of the interpersonal distance and the relative orientation. In this decision mechanism, in order to infer that there is an ongoing social interaction with a particular person, the person needs to be in proximity to interact and to have a relative orientation that allows the participants to take part in a social interaction. If both expressions are true then the mechanism infers that there is an on-going social interaction with the particular person, otherwise the mechanism infers that the two people are not taking part in a social interaction.

3.1.3 Data Processing and Knowledge Generation for Environmental Data

3.1.3.1 Pollen and Pollutants Density and Flow

In this section we describe two tools under development for prediction of pollen and pollutants density and flow.

3.1.3.1.1 Pollen density and flow

Predictive Models are very accurate tools for giving more complete information to health care professionals and population, so we keep on working on them for improving the public health service that we have been offering until now [25]. Pollen Levels Information is really necessary for the best knowledge and management of asthma and allergic diseases by patients, in order that symptoms are depending of individual sensibility and susceptibility.

Different Forecasting models have been elaborated with Autoregressive Integrated Moving Average Models (ARIMA-SPSS), using pollen data and meteorological parameters temporal series years [26]. Forecasting of pollen types levels have been made successfully since 2002, giving an individual bulletin every day with forecasting for 72 hours, with different “ARIMA” models of temporal series of each pollen type.

All the aerobiological information is made with real data, so our principal effort has been to give some predictive data every day in a short term way. Pollen levels are directly influenced by meteorological parameters, so forecasting has been elaborated with the daily meteorological predictive data, and the information is changed every
day, by inputting meteorological and pollen data to each Arima model.

Daily and weekly bulletins with data and charts have been elaborated and distributed to Population, Sanitary professionals, Hospitals, Healthy centres, Scientific societies and several press media such a radio, television and newspapers, carrying out the Public Health Service of the Palinocam Network [25, 27]. Figure 15 illustrates the whole process of the pollen information system which is driven by real-time and predicted pollen levels.

![Figure 15: Information flow of pollen information system.](image)

3.1.3.1.2 Pollutants density and flow

3.1.3.1.2.1 Air quality data

Each of the 24 air pollution monitoring stations sends data from the analysers to the Control center hourly. These hourly data are published after 20 minutes approximately and, once imported, they become part of Madrid city Council’s Air Quality Protection Service.

In addition to show data in terms of absolute values so as to make information more easily understood by citizens, data is also provided qualitatively through an air quality index available on Madrid city Council’s website and on the “Aire the Madrid” App.
Besides real-time information every hour, 24-hour air quality prediction is also updated.

3.1.3.1.2.2 Predictive model

The SERENA Model (statistical air quality prediction system based on neural networks) is based on a statistical or physical model that enables us to estimate the evolution of gas pollutants in remote stations of the Surveillance System.

The model obtains every hour from the Air Quality Surveillance System both pollutant concentration data and the meteorological data included in the calculations, thus modelling the hourly evolution of the concentration of various parameters such as nitrogen dioxide, ozone and particulate matter (PM10).

Based on the model results, air quality prediction for the next day is generated using the air quality index for each of the five zones in which the city has been divided. The prediction will be displayed based on the worst quality index expected for each day and each zone. Each zone will be represented in the colour appropriate for each quality index. Figure 16 illustrates the process of deriving the air quality index from air quality monitoring information.

![Diagram](image-url)

Figure 16: Information flow of air quality index system.
3.1.3.1.2.3 Health information

The obtained environmental data will be able to generate health care recommendations in high pollution situations providing healthy routes for the system’s users.

3.1.3.2 Rainfall Prediction

In this section, we describe how to use the Knowledge for town management domain which is one of the use cases, and how the iKaaS platform supports to process the knowledge effectively.

For one of the functions of the town management use case, we provide a weather prediction function to understand the situation of a very geographically limited and specific area in the town for the near future.

In normal weather prediction, the weather forecast at the site of the meteorological observatory is taken as the prediction for the average weather conditions citywide and across the region too. Therefore, the accuracy of the prediction would be decreased if the characteristics of the region of interest have particular geographical features and building structures. For example, at the foot of a mountain, near the river, windswept square, shade of the building, etc., the prediction may be incorrect. However, if it can be determined, the accurate weather prediction for a specific point in the town will be very useful for town management. For instance, in the winter season, if we can estimate the low temperature at a specific street corner, we can ensure the safety of walkers and automobiles at that corner by spraying the snow melting water in advance. If snow predictions are available over the next few hours, the schedule for snow removal and the equipment necessary for that can be appropriately planned.

For understanding the weather situation at a very specific point, we will install some environmental sensors at that location to collect weather-related data and store the data in the DB. In addition, we can use the past data of many official weather monitoring stations installed around the target region, which are available as open data. The sensor data of the town and the open data of the official weather monitoring points have some correlations. By using these big data of the past as the "Knowledge", we will predict the weather for the near future.

More specifically, as the official weather observation point of Japan, AMeDAS (Automated Meteorological Data Acquisition System) points
(a total of 1,300 positions) have been installed across the country in about 17km distance. At these locations, the amount of rain, temperature, sunshine duration, wind direction, wind speed, the amount of snow are been measured. The past observations can be used as open data. Thus, with this weather information and from the sensor data for a particular location, we will forecast the weather condition.

As the prediction algorithm we use the Memory Based Reasoning (MBR) [28]. The data processing for the weather prediction based on MBR is shown in Figure 17 below.

![Figure 17: Memory-based Reasoning for weather prediction](image)

As the expert system, rule-based system to make inferences based on the rules is well-known. The rules are normally obtained from domain experts through a knowledge acquisition process, and they are stored in the rule base. In contrast, MBR stores data directly to the database. MBR finds the most similar case among them. Then it presents the facts associated with the most similar case as the answer. Therefore, it is effective for the analysis from the big data, in case a large amount of data are available. On the other hand, Case Based Reasoning (CBR) is a similar method [29]. This method extracts the appropriate case from the data in advance and derives a solution by modifying the conclusions in the cases. To apply CBR, knowledge from domain experts is required. Because of the cost of
case acquisition and limitation of the target area, the MBR method is a better fit.

In other words, we define a “fact” by summarizing data of AMeDAS points that show the current weather situation and a series of data of the local observation point. We search for past cases that are most similar to this fact. In this case, the closeness of each parameter is used to quantify their similarity, which determines the most similar case. Time of the case is a past situation where most weather conditions are found similar to the current one. The weather conditions of a particular point after the specified time are presented as a solution.

Here, there are various methods for calculation of Similarity. However, if there is a point relatively close to the observation point, or a point whose geographical features are similar, by increasing the weight of the parameter of those points, it will be possible to obtain a more accurate estimation of results.

These weights, already stored as historical data, can be investigated by specifying the training cases and test cases. Since it is possible to simulate the prediction at any point in time, it would be possible to automatically optimize this in advance. For example, we can calculate the impact parameters by using the Cross Category Feature (CCF) method [30].

The MBR mechanism as described above, by providing one of the data processing functions of the iKaaS platform, can be reused for inference by utilizing the same kind of knowledge.

### 3.1.3.3 Anomaly Detection

A city which exploits data obtained from sensors installed at various locations around the city typically aims to build ‘city intelligence’ so that citizens and authorities can get a better understanding of their city. As noted in deliverable D2.1 [1], to build intelligence and derive knowledge, external information (e.g. locations, time, events that may influence sensed data) needs to be taken into account to put the collected data into context. For example, incidents or more generally anomalies, represent some events or points of inflection in which the sensed data may show sharp and sudden changes. As is also noted in the literature: “the real life relevance of anomalies is a key feature of anomaly detection [31]”. Thus, it is necessary to correlate the sensed
data from that obtained from other domains, such as social media streams, which can offer near real-time crowd sensing opportunities.

The developed methods for anomaly detection and cross-domain data correlation are being applied to the scenario of urban air pollution in order to validate the concepts. Air pollution is one of the main concerns of a city. With most air pollution sensing sites being situated at the sides of roads, the measured pollution levels are mainly influenced by traffic. The influence of industrial activities is not considered in this work. This leads to the focus on human activities that affect traffic conditions and are classed as real world events in this work. However, as each event will affect air pollution data differently (negative, positive), they should be first measured individually. As one of the main factors influencing air pollution, quantitative methods are needed to estimate the scale and influence of human activities on the real world. Then, the estimation can be used for examining the relationship between human activities and the sensed air quality data. For the human activities, we mean planned real world events and any accidental events that have an effect on human dynamics, that is, on the general human mobility patterns in a city. Assuming that human dynamics have an influence on air pollution data, there will be a correlation between the estimation of events and sensed data. Applying external information to all the sensed data may be time-consuming and inefficient since much of the data just represents normal behaviour, which is not of interest in most instances. Therefore, it is better to first identify anomalous data instances and then apply related external information to that subset.

3.1.3.3.1 Anomaly detection in air pollution data

As environmental situations in a city may vary not only temporally but also spatially, anomalies at different locations may be totally different from one another. Supervised approaches that require a large labelled training set may fit one location but not others. Thus, this work focuses on unsupervised approaches to generic mechanisms that are applicable across various locations.

Several methods have been implemented to detect anomalies in air pollution data and include distance-based approach, Kolmogorov complexity-based approach, and functional data analysis approach.

3.1.3.3.1.1 Distance-based

One approach for anomaly detection is to first define a normal behaviour, followed by determination of measurements about how a
pattern conforms to normal behaviour, and finally to decide on anomalies based on the measurement.

Based on this process, a distance-based approach is proposed. When a daily pattern is considered, a normal pattern is defined by mean values at each hour of a day. Euclidean distance (equation 3) is applied to determine how far a day pattern is from the normal pattern.

$$ \text{Distance} = \sqrt{\sum_i (d_i - n_i)^2} $$  \hspace{1cm} (3)

where \( d \) is a vector of a day pattern, \( n \) is the vector of the normal pattern, \( i \) indicates one dimension of a vector. Moreover, Z-scores (equation 4) are used to evaluate the computed distances.

$$ z = \frac{x - \mu}{\sigma} $$  \hspace{1cm} (4)

where \( x \) represents each distance from (1), \( \mu \) is the mean value of all the measurements, \( \sigma \) is the standard deviation. Thresholds can be set based on Z-score (equation 4) [32]. Those with Z-score over 3 are anomalies, while any measurement with Z-score between 2 and 3 is being regarded as uncertain.

$$ |z| \leq 2 \rightarrow \text{Normal} $$

$$ 2 < |z| < 3 \rightarrow \text{Uncertain} $$  \hspace{1cm} (5)

$$ |z| \geq 3 \rightarrow \text{Abnormal} $$

3.1.3.3.1.2 Kolmogorov Complexity-based

In practice, there will be a lack of information about anomalous behaviour. Therefore, many techniques make an assumption so that anomalies can be detected. Information theoretic techniques assume that “anomalies in data induce irregularities in the information content of the data set [31].” Thus, they assume that anomalies will make the data set more complex. One typical complexity measurement is Kolmogorov Complexity, which can be computed through Kolmogorov-Smirnov test [33].

The approach to compute anomaly score based on Kolmogorov Complexity [34] consists of computing, for one data point, the Euclidean distances to all data points in a random sampled sequence. The distances form a sequence A. Next, a new sequence is sampled
randomly. For the data points in this new sequence, their distances of data points to the first random sampled sequence are computed and form distance sequences $\emptyset_B = \{B_1, B_2, \ldots, B_n\}$, where $n$ is the number of sampled data points. The Kolmogorov-Smirnov test is applied to $A$ and each $B$ in $\emptyset_B$. The mean value of Kolmogorov-Smirnov test values is regarded as the anomaly score. Thresholds can be set on this anomaly scores to decide anomalies.

3.1.3.3.1.3 Functional Data Analysis

Functional data analysis considers the entire function or curve, not the value at a particular data point [35]. Functional data are continuously changing, but observed at discrete points. Thus, functional data analysis will first construct curves from discrete observations. This step uses basis functions with coefficients to form a function to represent the curve. Curves are compared and analysed to get more information.

With respect to anomaly detection, functional depth is introduced to measure how close a curve is to the centre of the rest of curves [36] [37]. Observations that differ much from the rest of data are treated as an anomaly, i.e. the shallowest curves are anomalies. Instead of computing depths on continuous curves, computation of functional depth is adapted on discrete observations directly.

Fraiman and Muniz Depth (FMD) and H-Modal Depth (HMD) are two methods measuring depth of a curve [36] [37]. For FMD, the cumulative empirical distribution function (equation 6) is used.

$$F(x_i(t)) = \frac{1}{n} \sum_{k=1}^{n} I(x_k(t) \leq x_i(t))$$  \hspace{1cm} (6)

where $I(.)$ is an indicator function, $n$ is the number of data points at time point $t$, $i$ indicates the considered curve, $x_i(t)$ is the data value of curve $i$ at time point $t$, and $k$ indicates each data point. The depth of one data point is calculated consequently, as shown in (equation 7).

$$D(x_i(t)) = 1 - \left| \frac{1}{2} - F(x_i(t)) \right|$$  \hspace{1cm} (7)

The depth of a curve (equation 8) is indicated by the integral of depths of data points.

$$FMD(x_i) = \int_{a}^{b} D(x_i(t))dt$$  \hspace{1cm} (8)
where \([a,b]\) is the interval of time points. The discrete version, Sample FMD (SFMD) (equation 9), is obtained through Riemann sum:

\[
SFMD(x_i) = \sum_j \Delta_j D(x_i(t_j))
\] (9)

where \(\Delta_j\) is a time interval between two data points.

HMD (equation 10) is computed by the sum of kernel function.

\[
HMD(x_i) = \sum_{k=1}^{n} K\left(\frac{||x_i - x_k||}{h}\right)
\] (10)

where \(K(\cdot)\) is a kernel function, \(h\) is a bandwidth, \(x_i\) is a curve, \(||\cdot||\) is a norm in the functional space. The kernel function could be the truncated Gaussian kernel function (equation 11). The norm could be Euclidean distance (equation 1).

\[
K(t) = \frac{2}{\sqrt{2\pi}} e\left(\frac{-t^2}{2}\right), \quad t > 0
\] (11)

3.1.3.3.2 Real World Event Detection from Social Networks

Social media (Twitter) is considered as the data source for events information. This work proposes to extract event information from a large volume of tweets. To achieve that, a classification of events is first derived that is based on their direct influence on traffic and air pollution, as well as the amount of people involved.

**Table 12: Classification of expected real world events**

<table>
<thead>
<tr>
<th>Category</th>
<th>Traffic influence</th>
<th>Air pollution influence</th>
<th>People involved</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>High</td>
<td>-</td>
<td>Many</td>
<td>fast/slow traffic, roadwork, road incident, collision</td>
</tr>
<tr>
<td>Culture</td>
<td>High</td>
<td>Depends on scale</td>
<td>Could be many</td>
<td>concert, celebration, performance, exhibit, fair, festival,</td>
</tr>
</tbody>
</table>
market, parade, firework show, and any other real world event that gather a lot of people

<table>
<thead>
<tr>
<th>Sports</th>
<th>Low</th>
<th>Many</th>
<th>Sports match, race, tournament</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air-quality</td>
<td>-</td>
<td>High</td>
<td>description of air pollution</td>
</tr>
<tr>
<td>Weather</td>
<td>-</td>
<td>High</td>
<td>any weather description; includes wind, precipitation, temperature, cloud, sun, etc.</td>
</tr>
<tr>
<td>Disaster</td>
<td>-</td>
<td>High</td>
<td>Many</td>
</tr>
<tr>
<td>Non-event</td>
<td>-</td>
<td>few</td>
<td>description of personal activity, no effect on others</td>
</tr>
</tbody>
</table>

It is difficult to obtain events information due to several reasons:

- There is no uniform list about different types of planned events.
- There is no platform reporting accidental events.
- Traditional newswire may mainly focus on big news and may ignore events information that happen locally but may influence human dynamics.

For these reasons, twitter is chosen as the data source of events. This offers multiple advantages.
- There are a large number of Twitter users in different cities; this allows extracting events information for various places without changing the implementation.

- Twitter users focus on every type of topic. This provides opportunities to get complete information about different events.

- Twitter is relatively more popular than other social networks and thus has a larger user base reporting different types of public events.

However, to get event information from Twitter, simple search and query is not enough. This is because the highly restrictive keywords may lead to only limited events being retrieved and missing most events, especially in the absence of domain knowledge about any given city. On the other hand, general keywords will involve a huge amount of work to create a training set that will also be specific to the known and expected events of a particular city. To design a generic solution and avoid the need of creating a training keyword set for each city, an unsupervised method based on Twitter-LDA (Twitter Latent Dirichlet Allocation) [38] is proposed. The approach consists of the following steps:

3.1.3.3.2.1 **Tweet retrieval**

The tweets are retrieved from the Twitter search website API (using mobile version for lower traffic). The search statement is constrained only by place keywords and date parameters. This is to get a complete set of tweets for a place on a certain day. An example tweet retrieved for the city of London on the 5\textsuperscript{th} of February 2016 is shown below:

Elton John performs impromptu concert in London
rightrelevance.com/search/article pic.twitter.com/RuvZ5uScIJ

3.1.3.3.2.2 **Pre-processing**

Before applying Twitter-LDA, the set of retrieved tweets need to be pre-processed. This step includes tokenising, stop words removal, and noise words removal. Tokenising splits a sentence into tokens, which are basic elements of the sentence, such as words and punctuations. Then, stop words, which are words commonly appearing in any kind of topic in a language, are removed. This is to avoid computation effort on unimportant words. The stop words list is built from Rainball stoplist [39] plus words commonly used in tweets
that have little or no meaning, such as *ha, hah*, etc. Next, URL links and some unreadable codes are considered as noise words that will not contribute to the later steps and are removed. Thus, words are separated from the original sentence and less meaningful words are filtered out. The original tweet is shown below after it is pre-processed:

```
Elton John performs impromptu concert London
```

### 3.1.3.3.2.3 TwitterLDA Analysis

After pre-processing, TwitterLDA can be applied on the cleaned and formatted tweets set. Twitter-LDA is a topic model designed for tweets based on LDA (Latent Dirichlet Allocation) \[40\]. LDA is a generative probabilistic model dealing with discrete data such as text. LDA topic model assumes a text corpus has a fixed number of topics, a document in the corpus has a mixture of topics, which form a Dirichlet distribution, and the order of words or documents does not matter. The generative process \[41\] of LDA model can be described as follows:

1. Randomly choose a Dirichlet distribution over topics.
2. For each word in the document
   a. Randomly choose a topic from the distribution over topics in step #1.
   b. Randomly choose a word from the corresponding distribution over the vocabulary.

In order to fit short text like tweets, Twitter-LDA model makes some modifications to normal LDA. It assumes one tweet talks about only one topic and involves a small amount of background words that do not contribute to any topic. The generative process \[42\] is described as follows:

1. Randomly choose a Dirichlet distribution \(b\) over background words and a Bernoulli distribution \(bt\) over decision on background words and topic words.
2. Randomly choose a Dirichlet distribution \(t\) over topical words for each topic.
3. For each user’s tweet collection
   a. Randomly choose a Dirichlet distribution \(tu\) over topic.
   b. For each word
      a. Randomly choose a multinomial distribution governed by \(bt\) over decision on whether it is a background word
      b. Randomly choose a multinomial distribution governed by \(b\) over word, if the word is a background word;
randomly choose a Multi distribution governed by $t$ over word, if the word is a topical word.

Parameters of the model can be inferred through Gibbs Sampling [43], which is a Markov Chain Monte Carlo method to estimate a probability distribution. Output includes topics with keywords explanation, topic distribution for each user, and the number of tweets for each topic.

During implementation, tweets collected for one day are considered as a user’s tweets collection, talking about a mixture of topics. Then topics with keywords and the number of tweets for that day can be inferred through Twitter-LDA.

Assuming a number of tweets talking about the same topic as in the example above (i.e. Elton John performing in London) in different ways, the output of Twitter-LDA is shown below:

```
john elton surprise train station piano plays performance concert watch commuters surprises crowd pancras medley hits-filled st impromptu sir station, play piano, performs pops station: leaves #eltonjohn deliver fans #music
```

### 3.1.3.3.2.4 Topic Event Labelling

Topics with top related keywords are one of outputs of Twitter-LDA. Since Twitter-LDA is an unsupervised approach, output topics are not labelled with any meaningful name. In order to link unlabelled topics to real world events, the topics are labelled based on the event type model specified in Table 12. The model is built on event types with highly related keywords for each type. A topic will be set as an event type which has the most number of keywords in the topic. The topic with no matched keywords in the event type model will be set as a non-event.

### 3.1.3.3.2.5 Event Scale Estimation

At the end of the Twitter-LDA process, each type of event will contain several topics, and each topic will contain a number of tweets. Thus the number of tweets talking about one type of event can be computed as well as the **Event Tweet Frequency**, which is computed by the following equation.
\[
\text{Event Tweet Frequency} = \frac{\text{No of Detected Event Tweets}}{\text{No of Sampled Tweets}} = \frac{\text{Population Involved in Event} \times \text{Tweet Rate} \times \text{Event Influence Factor} \times \text{Sampling Rate}}{\text{Population at the Place} \times \text{Tweet Rate} \times \text{Sampling Rate}}
\]

(12)

where \textit{Tweet Rate} is the percentage of people involved in an event writing a tweet, \textit{Sampling Rate} is the rate of number of tweets sampled from the tweets collection, and \textit{Event Influence Factor} indicates how many multiples of a tweet for an event will be posted than that in a normal situation. By assuming sampling is random, we can treat \textit{Sampling Rate} is the same and divide it out. Thus we have following equation.

\[
\text{Event Tweet Frequency} = \frac{\text{Population Involved in Event} \times \text{Event Influence Factor}}{\text{Population at the Place}}
\]

(13)

Then the estimation of the scale of event based on population can be calculated by the following equation.

\[
\text{Population Involved in Event} = \frac{\text{Population at the Place} \times \text{Event Tweet Frequency}}{\text{Event Influence Factor}}
\]

(14)

where \textit{population at the place} is easy to obtain from openly available data, \textit{event tweet frequency} is computed from the above algorithm, and \textit{event influence factor} is set by experience. The output of the algorithm for one event type (Culture) is shown below:

```
Culture || 8664.774 || john elton surprise train station
```

At the end, the type of events, top keywords for explanation, and population estimation are the output of a certain day. By applying the algorithm on different days, series estimation can form a vector for a particular event and will be used for correlation and analysis with air pollution data.

3.1.3.3.2.6 Event Location Tagging

In order to find the relationships between detected anomalies and events, their location information need to be determined. Although the location information of anomalies can be obtained from sensing sites, determination of location information of the detected events is not straight-forward. This is because social media data are informal. Social media data do not always follow the grammar syntax and may contain mistyped words, special words, or even wrong words. To
overcome these issues, an aggregation and rank-based location entity detection approach is proposed. For each detected event, the approach examines all the related tweets and finds the location entities in the tweets, using location named entity recognition model in OpenNLP [44]. The detected location entities are aggregated and ranked by their occurrences. The top 2 entities are used to represent the location of the event. Also, the precise latitude and longitude can be obtained by sending a query with the top 2 entities to the Google Maps Geocoding API [45]. The resultant output is shown below:

| Culture || 8664.774 || john elton surprise train station || lat:51.5268540 || lon:-0.1245670 || London - St Pancras |

3.1.3.3 Correlation between air pollution anomalies and real world events
The above two subsections describe how to detect anomalies in air pollution sensing data and real world events from social networks. In order to get a thorough understanding of the relationships between them, it is required to do a reasonable correlation analysis. The key points of correlation are spatial and temporal coincidence, i.e. detected anomalies and events should happen within a certain time period overlap and at the same place so that they can be correlated. The temporal axis is taken into account by determining the real-world events that take place on the day of the detected anomalies. Next, the locations of events need to be determined. Although the tweets are retrieved through place keywords, this covers a wide region (e.g. an entire city/town) and we still need to analyse the exact location of one event and get the relationship between events and sensing site. Thus the events that happen at the same location of anomalies can be selected. Since each kind of event will affect air pollution data in different ways (positive or negative), it is necessary to analyse correlations between anomalies and each event as well as aggregation of each kinds of events (e.g. all sports events). In addition, events may affect different areas in different levels (e.g. affect roads more than residential areas). Therefore, the type of area of sensing site needs to be considered as well.

3.1.4 Data Processing and Knowledge generation on User Data

3.1.4.1 Health Monitoring
Data processing in this case addresses the monitoring and assessment of vital signs, activities and patterns of movements (e.g. walking, sleeping patterns, eating patterns, physical and social activity, etc.) for inferring the physical condition of an individual. The
aim is to acquire knowledge on the user health status and potentially behaviour and identify potential irregularities that may call for medical treatment. Notifications/alarms can be raised in case something is not yet abnormal but the recorded values show a trend towards a potential problematic situation (e.g. increasing blood pressure which has still not reached a certain threshold may still be worrying). The acquired knowledge may also be exploited for automated decision making that requires taking into account the user’s health status. For the sake of simplicity, in the following we focus only on the monitoring of vital signs.

In this section a machine learning algorithm is presented, implementing a knowledge acquisition tool for the prediction of future vital signs of elderly/disabled people, so as to prevent unpleasant situations. The algorithm is based on time series forecasting. The input of this algorithm is a specific number of vital sign measurements, which are collected via a wearable device (e.g. a smartwatch). After a predefined number of input measurements, hereafter termed as window size, the algorithm is triggered and reforecasts the future vital signs, with the new appended values to the machine learning training dataset, in order to offer more accuracy. In case the prediction consists of abnormal vital values, then a notification is issued to the user’s smartphone that informs the user for the possible upcoming health situation. Also, it is possible to inform a designated doctor/health care professional about the possible abnormality of the under supervision person. Apart from the prediction part, there is the simple monitoring functionality, which is dedicated to the raising of notifications/alarms when the instantaneous collected vital measurement is critical or generally above the normal levels (i.e. certain predefined thresholds).

In order to analyse the math behind the time series forecasting algorithm we consider an example where the input of the algorithm is the upper blood pressure of an elderly person. Figure 18 depicts the whole training procedure, as extracted from running the algorithm.
More specifically, in Figure 18:

- The first column represents the number of periods at which data is collected. For example, the number 1 may depict the first day, if the collection of data is set to be day per day, or it may depict the first hour if the collection of data is performed at an hourly rate, etc.

- The second column lists all the measurements collected via the wearable device per monitoring period (column 1), which constitute the training dataset (input data) for the time series forecasting algorithm.

- The third column is the Centered Moving Average (CMA), which holds the average between the measurements that have been taken within a specific period of time e.g. within a week. In our case we assume that three measurements have been taken. In this step, if we draw the line which connects the CMA values it would produce a smoothed curve, which consists a “baseline” for the next steps.

- The fourth column shows the Seasonal with the Irregular component, which is calculated by the division of the measurements (input data) with the corresponding CMA. The produced number depicts the divergence from the baseline. To clarify, a value of 0.90 means that the Seasonal with the Irregular component is 10% below the baseline.

- The fifth column holds the Seasonal component without the Irregularity. In our case, where three measurements are taken within a period of time, it means that we are going to have three seasonal components. In this step the logic is, as it is also depicted in the figure, to average all the Seasonal with Irregularity values that were calculated first in every specified period of time, so as to derive the first seasonal component,
then the second and so on. Afterwards, the results are placed with a pattern as depicted in Figure 18.

- The next step is to de-seasonalize the data. This is calculated by dividing the measurements (input data) with the corresponding seasonal component.
- In order to derive the Trend component (Tt) it is necessary to run a Simple Linear Regression with Y (response) variable the de-seasonalized data and X variable the time period t from the first column of Figure 18. The produced result is used to derive the coefficients (intercept, slope). Finally, the trend component is computed by the addition of intercept with the slope times the time period t (Tt = intercept + slope*t).
- The final step is to extract the forecasted values. These are calculated by multiplying the seasonal component with the trend component. These forecasted values can be compared to the real input values (measurements) so as to check the accuracy of the algorithm. To actually forecast future values, it is only needed to extend the first column, so as to calculate the future trend component (the coefficients are still the same) and to continue the filling pattern of the seasonal component.

By running the time series algorithm for the measurements shown in the previous figure, the results depicted in Figure 19 are produced.

![Vital Signs Forecasting Charts](image)

Figure 19: Output of time series forecasting algorithm.
The red points in Figure 19 represent the input – training dataset, which consists of the vital signs collected from the user with the use of a wearable or other device and the blue points are the forecasted values. For the sake of the example the collected measurements are random and the time between them is one second.

### 3.1.4.2 Prediction of user preferences on device configuration

Data processing in this case deals with learning user preferences/desires with respect to home and appliances configuration. The corresponding acquired knowledge can then be exploited for the purpose of home automation by proactively taking actions/offering recommendations. For example if the user keeps adjusting the temperature (turning the heating/air-conditioning up/down, switching the lights on earlier than in the system configuration) these adjustments are recorded along with time, weather, date information to gradually (autonomously) derive knowledge on what the user prefers most. For the sake of simplicity, in the following we focus only on the temperature settings.

The knowledge acquisition tool described in this section is a machine learning algorithm based on **Bayesian Statistics**. In order to describe how this algorithm works we will consider that we use this algorithm to predict the desired temperature in the domain of Smart Home. The aim is to estimate the probability of the temperature preference for a user, given a certain location and time zone. In other words, conditional probabilities for the temperature are calculated. The input of the algorithm is a temperature setting at a given time and location within the home.

In the implementation, for the sake of simplicity we considered that the possible temperature values, which a user may like to set in home, are in the range of 18 – 25 degrees Celsius. This range can however be adjusted. Moreover, the daytime is separated in five time-slots: Morning (5:01 - 10:00), Midday (10:01 - 15:00), Afternoon (15:01 - 20:00), Evening (20:01 - 00:00) and Night (00:01 - 5:00). These time-slots are helpful so as to specify the user preferences in a more generic timeline.

#### 3.1.4.2.1 Bayesian Statistics training

The machine learning algorithm presented in this section is based on supervised learning, which means that in order to build a model to make predictions, it is necessary to have a training dataset. The
algorithm learns from the patterns being observed in the training examples and builds a probabilities table. More specifically, the training dataset is composed by every possible temperature in every room at every time-slot, from which the algorithm learns every possible eventuality. As a result, initially, it assigns an equal probability to each eventuality. Afterwards, according to user preferences and actions, the probabilities are adapted for various combinations of temperature, location and time.

When the algorithm runs for the first time, it reads the training dataset and creates a **counter** that counts the frequency of occurrence of each temperature. It also, creates a **probability table**, in which there are all the possible temperature values in every possible combination with the different locations and time-slots.

### 3.1.4.2.2 Bayesian Statistics Algorithm

To avoid repeating the training dataset to make new forecasts, the probabilities table is used. When the algorithm is triggered, the probability of a specific temperature, at the given location and time is updated. In this way, the training dataset is used only once and it is not increased, which would make it difficult and time consuming to process. Instead, the knowledge is imprinted in the probabilities table, which is easy to manage.

The Bayesian algorithm is triggered when the user performs a temperature setting (increasing/decreasing the temperature) at a certain location within the Smart Home, at a specific time. Based on these observations the instantaneous probabilities for the temperature are estimated. The next step is the calculation of adapted (conditional) probabilities. These calculated probabilities yield the most likely user preferences in terms of temperature per location and time period, which in turn are used as input by the functionality for deciding on the most appropriate device configuration (in this case the heating). A mathematical formulation for the functionality for dynamically learning user preferences can be found in [1].

### 3.1.4.3 Home automation

This functionality comprises of a decision making process for selecting the optimal configuration for home devices/systems such as heating. This takes into account current context (i.e. measured temperature, user location and time zone) and knowledge on user preferences. The latter is obtained via the mechanism described in section 3.1.4.2.
This process is triggered periodically to ensure that the temperature settings within the users’ home are aligned with the forecasted user preferences. The forecasted temperature preference is achieved by enabling or disabling the corresponding actuators. For example, if the forecasted temperature value is 20 degrees Celsius and the temperature in the room is 17 degrees Celsius, then the heating will be switched on.

In order to ensure correctness of the actions taken by the decision making functionality and the probabilities calculated by the learning mechanism described in section 3.1.4.2, a type of reinforcement learning is added to the procedure. More specifically, a pop up window is shown in the application’s screen that asks the user to give feedback about the forecasted temperature value and corresponding action taken. If the user approves the temperature forecast and corresponding action the algorithm is given a reward. This is used to “enforce” the corresponding probability for the specific temperature, for a certain home location and time zone. If the user prefers another temperature for the specific location and time, it is easy to set it through the pop up window and the corresponding actuators will be enabled or disabled according to the situation. Also, the algorithm will not receive a reward and the probability for the temperature explicitly provided by the user will be enforced.

3.1.4.4 Personalized recommendation on optimal route to reach a certain destination

This functionality exploits knowledge on user health status, user preferences, public transportation means and conditions, city events (e.g., protests, concerts), weather, pollution, pollen, etc. so as to provide a personalized recommendation on the optimal route to a destination desired by the user. For example, depending on the user sensitivity to pollen it may be best to select a "pollen-free" route even if it may not be the shortest one. Essentially this mechanism will rely on decision-making algorithms. This functionality is currently being developed in the scope of the assisted living use case but is generic with respect to the use case, i.e. it can be utilized also in different contexts. The aim is to showcase the exploitation of knowledge being provided by various sources.

Input

- User profile, knowledge on preferences and health status
- Information on city events along potential routes to the destination
- Data on available public transportation means (if applicable, i.e. if the user is interested in using public transport)
- Knowledge on traffic conditions along potential routes to the destination
- Knowledge on weather conditions along potential routes to the destination
- Knowledge on pollution conditions along potential routes to the destination
- Knowledge on pollen conditions along potential routes to the destination

**Process**

First a set of candidate routes to the desired destination is identified. This includes diverse means of transportation (i.e. public transportation, car or walking). Then for each of these routes the estimated time to reach the specified destination is calculated, taking into account information on city events, traffic conditions, public transportation and weather conditions. In addition, for each candidate route an optimality “score” is calculated considering pollution and pollen conditions in conjunction with user preferences and inferred health status. Basically, the aim is to select the optimal route in terms of the minimum time required to reach the destination, while also minimising the exposure to pollen and pollution and other potentially hazardous situations (e.g. avoiding a demonstration or an area where an accident has occurred). Multi-criterial decision making methods will be used for this purpose. This process is triggered by the user when requesting for a route from a specific starting point to a specific destination.

**Output**

Recommendation on the optimal route to a certain destination. A prioritized list of alternative routes may also be displayed.

### 3.2 Mapping of data processing tools to iKaaS architecture and data model

The developed data processing and knowledge acquisition tools map to different components of the Data Processing block (information processing/decision making/learning) in the iKaaS architecture. The
specific mappings of the different tools are listed in Table 13 below along with the different data model components employed by the tools for their functioning.

Table 13: iKaaS data processing tools - architecture and data model mapping

<table>
<thead>
<tr>
<th>Data processing tool</th>
<th>iKaaS component</th>
<th>architecture</th>
<th>iKaaS data model component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Event Processing</td>
<td>Information processing</td>
<td>VE model, service</td>
<td></td>
</tr>
<tr>
<td>Data collection from mobile devices</td>
<td>Virtual entity (Local cloud)</td>
<td>VE model, service</td>
<td></td>
</tr>
<tr>
<td>Data processing for environmental data – Pollutant and Pollen flow</td>
<td>Information processing, learning, decision making</td>
<td>Service model, knowledge model instance as output</td>
<td></td>
</tr>
<tr>
<td>Data processing for environmental data – Rainfall prediction</td>
<td>Information processing, learning, decision making</td>
<td>Service model, knowledge model instance as output</td>
<td></td>
</tr>
<tr>
<td>Data processing for environmental data – Anomaly detection</td>
<td>Information processing, learning</td>
<td>Service model, knowledge model instance as output</td>
<td></td>
</tr>
<tr>
<td>Data processing for User data</td>
<td>Information processing, learning, decision making</td>
<td>User model, service model (input), knowledge model instance as output</td>
<td></td>
</tr>
</tbody>
</table>

3.3 iKaaS Advances and Innovations

Complex Event Processing: we propose a distributed data processing mechanism able to self-adapt to the environment. The developed light complex event processing engine is able to run in small devices and collaborate with data bases and analytics tools. An
important objective of the Madrid Use case is to build real-time maps of pollen and pollution particles distribution, which in turn brings up the need for a tool that is able to fix deviations on measurements on the fly. This self-adapting data processing mechanism will be applied to the filtering and calibration of sensor networks autonomously. In scenarios like the Madrid use case the measurements taken by new deployed sensors will be analysed against measurements from legacy sensor networks to derive patterns on deviations and fix them on the fly.

**Data Collection and Annotation from Mobile Devices:** the novel opportunistic approach for detecting social interactions uses only off-the-shelf mobile phones and does not require user involvement in the inference process. Users’ interpersonal distance is determined in a fine-grained manner based on only 6 Bluetooth RSSI samples. Users’ relative orientation is determined by improving a state-of-the-art technique [13] for facing direction detection, independent to device on-body wearing position and these are enhanced through collaborative sensing for faster, privacy preserving and real-time inference. Additionally, the system requires neither any external hardware nor any firmware modifications because it leverages Bluetooth’s native capability for ad-hoc discoverability and communication, making it suitable for pervasive deployment by simply downloading an app. Each device performs inference online in order to eliminate any privacy issues occurring when transmitting data to third parties. Devices calculate interpersonal distance and relative orientation with respect to each nearby user. Finally, we perform classification on the occurrence of a social interaction, depending on the selected target class (proximity or interaction zone).

**Knowledge generation for Rainfall Prediction:** the developed mechanism uses Memory Based Reasoning (MBR) for the prediction algorithm, which finds the most similar case from past data. The past data considered include weather-related data of the specific point in the town as well as those from official weather monitoring stations installed around the target region. The proposed mechanism can achieve more accurate prediction results by setting higher weights for the parameters for those geographical points that are relatively close to the observation point, or whose geographical features are similar to the observation point. These weights can be optimised in advance by using the Cross Category Feature (CCF) method and also take into
account the season and prediction duration. These optimisations, which are part of the pre-processing step of the MBR method, can reduce the size of the data set based on the weights, leading to improved efficiency as well as accuracy.

**Data Processing for Anomaly Detection:** this mechanism brings together the social world with data obtained from cyber-physical systems to provide a semantic context to anomalies detected in physical data streams. As social media is becoming an important source of information in the smart city domain, an unsupervised approach is proposed to detect real world events from Twitter data. Current event detection approaches from tweets can either detect specific events [46-47] or general open events [48-49]. Specific event detection approaches may not be feasible to give a context to the detected anomalies on sensing data, because the meaning of anomalies is not known beforehand. On the other hand, open event detection approaches are not sufficient due to their lack of distinction between related real world events and other non-related ordinary events. The developed LDA-based a bag of words model can detect any topic being discussed on social media and it is supported by a keyword-based event type model to label detected topics as types of real world events. This allows non-event topics to be filtered out and enrich the explanation of the detected topics. Since location information is important for linking events to detected anomalies, a location detection approach has also been developed which determines the location information of related events. Moreover, few approaches provide impact measurements of detected events. In order to correlate events with sensing data, it is particularly important to provide a quantitative estimation of the impact of the event in the real world. The impact of events is estimated according to event type, number of tweets, etc. These measurements can be aggregated based on event type and reflect the impact on the real world.

**Data Processing and Knowledge generation on User Data:** a key innovation is the forecasting on user preferences and potential health risk, which is exploited for the provision of corresponding personalised recommendations/actions. More specifically, knowledge is derived on user preferences, wellbeing and health for autonomous adaptation of home towards user needs as well as proactive recommendations towards the improvement of personalized health/fitness based on knowledge derived on potential health risks.
The derived knowledge can also be exploited by other applications/services for further enhanced context/situation aware applications. In addition, the tools being developed aim at being independent from specific devices, whereas a very large number of existing solutions/products (e.g. Apple HomeKit, Apple HealthKit, Nest (Google), SmartThings (Samsung), iHealth, Netatmo, Bodytel, etc.) are tied to devices from specific vendors.
4 Conclusions

This deliverable has provided an update on the design of the iKaaS data model components and the mechanisms for data processing and knowledge acquisition to support service provisioning in multi-cloud environments. The addition of the platform concept allows specification of the capabilities of a cloud platform. Analogously, the service model has been updated to reflect the requirements of a service in terms of cloud deployment requirements (e.g. what are the minimum memory, CPU requirements, etc.). These additions contain important concepts and will form the foundation of other activities in the iKaaS project. Specifically, these will enable automating the process of optimal service deployment, as they will facilitate taking into consideration not only application and user preference requirements, but also available cloud resources. The user model has also been updated to capture modelling of trust relationships between users and can form the basis for resource sharing and access determination mechanisms. The links between the data model components and the iKaaS architecture components have been listed. Similarly, the different data processing mechanisms have been mapped to the internals of the Data Processing block of the iKaaS architecture as well as the data model components they use.

Progress on the data processing and knowledge acquisition mechanisms has been reported with the specification of the details of the algorithms employed and initial results. The reported knowledge generation mechanisms map to the focus application areas of the iKaaS project: environmental monitoring and user data processing for health monitoring. The contributions that these mechanisms make to the state-of-the-art have also been reported. The next deliverable D2.3 will report on the implementation and validation of the developed mechanisms, which will enable knowledge derivation as an explicitly modelled data in a knowledge database, to allow the provision of knowledge to any authorized third party entity.
5 List of Figures

Figure 1: Component links in the iKaaS data model .................. 5
Figure 2: Overview of the city data model structure ................. 10
Figure 3: Access Right data model .................................. 12
Figure 4: VE Model .................................................... 15
Figure 5: Platform Data Model ...................................... 21
Figure 6: Service Model ................................................. 23
Figure 7: iKaaS Knowledge data model ............................... 28
Figure 8: User Model Entities diagram ................................. 31
Figure 9: iKaaS Functional Architecture ............................... 36
Figure 10: Mapping toolbox for real-time distributed data processing over iKaaS functional architecture ....................... 37
Figure 11: Information flow of the F2F enabler application including the communication part of it with nearby devices. .............. 43
Figure 12: Detection of interpersonal distance between users. ...... 49
Figure 13: Importance of including relative orientation in the social interaction detection process ........................................... 50
Figure 14: Detection of the relative orientation of two users. ...... 51
Figure 15: Information flow of pollen information system. ........... 53
Figure 16: Information flow of air quality index system ............... 54
Figure 17: Memory-based Reasoning for weather prediction ........ 56
Figure 18: Output of time series forecasting algorithm training. ..... 69
Figure 19: Output of time series forecasting algorithm ................. 70
6 List of Tables

Table 1: User data model requirements .................................................. 8
Table 2: Service data model requirements .................................................. 8
Table 3: Platform data model requirements ................................................. 8
Table 4: Access Rights model description ................................................. 13
Table 5: VE Model description ................................................................. 15
Table 6: Platform Model description ......................................................... 21
Table 7: Service Model description ........................................................... 24
Table 8: Knowledge data model description .............................................. 28
Table 9: User Model concepts description ............................................... 31
Table 10: iKaaS data model - architecture mapping ................................. 33
Table 11: iKaaS data model - use case mapping ....................................... 33
Table 12: Classification of expected real world events .............................. 61
Table 13: iKaaS data processing tools - architecture and data model mapping ................................................................................. 75
# 7 List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADE</td>
<td>Application Domain Extensions</td>
</tr>
<tr>
<td>AMeDAS</td>
<td>Automated Meteorological Data Acquisition System</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average Models</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CCF</td>
<td>Cross Category Feature</td>
</tr>
<tr>
<td>CEP</td>
<td>Complex Event Processor/Processing</td>
</tr>
<tr>
<td>CESP</td>
<td>Community Environment Sensing Platform</td>
</tr>
<tr>
<td>CMA</td>
<td>Centered Moving Average</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>FHIR</td>
<td>Fast Healthcare Interoperability Resources</td>
</tr>
<tr>
<td>GML</td>
<td>Geography Markup Language</td>
</tr>
<tr>
<td>HL7</td>
<td>Health Level 7</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>LDA</td>
<td>Latent Dirichlet Allocation</td>
</tr>
<tr>
<td>MBR</td>
<td>Memory – based Reasoning</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>PK</td>
<td>Public Key</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VE</td>
<td>Virtual Entity</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
8 List of References


3. Internet of Things Environment for Service Creation and Testing (IoT.est), http://ict-iotest.eu/iotest/


7. Virtualized Programmable InTerfAces for Smart, Secure and Cost-Effective IoT depLoyments in Smart Cities (VIrTAL), EU FP7 project. Available online: http://vital-iot.eu/project


11. N. Palaghias, S. A. Hoseinitabatabaie, M. Nati, A. Gluhak and K. Moessner “Modelling Social Interaction Detection with...


47. P. Anantharam, P. Barnaghi, K. Thirunarayan and A. Sheth “Extracting City Traffic Events from Social Streams”, ACM
Transactions on Intelligent Systems and Technology, 9(4) 2014.


49. A. Ritter, O. Etzioni and S. Clark, “Open domain event extraction from twitter”, In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1104-1112), August 2012.
Partners:

- 1st iKaaS Semantic Data Model, Knowledge Acquisition and Service Provision Toolbox

Coordinator's contact:

Prof Klaus Moessner
University of Surrey
k.moessner@surrey.ac.uk

This project has received funding from the European Union’s Horizon 2020 research and innovation programme, call: EU-Japan Research and Development Cooperation in Net Futures programme, under grant agreement number: 643262