Ref. Ares(2015)554522 - 10/02/2015

SEVENTH FRAMEWORK PROGRAME

Specific Targeted Research Project

Project Number:	**FP7–SMARTCITIES–2013(ICT)**
Project Acronym:	**VITAL**
Project Title:	**Virtualized programmable InTerfAces for innovative cost-effective IoT depLoyments in smart cities**

D2.3 Virtualization Architecture and Technical Specifications

Document Id:	VITAL-D23-300115-Draft
File Name:	VITAL-D23-300115-Draft.docx
Document reference:	Deliverable 2.3
Version :	Draft
Editor :	John Soldatos, Roukounaki Aikaterini, John Kaldis,
Organisation :	AIT
Date :	30 / 01 / 2015
Document type:	Deliverable
Security:	PU (Public)

Copyright © 2015 VITAL Consortium

PROPRIETARY RIGHTS STATEMENT

This document contains information, which is proprietary to the OpenIoT Consortium.
Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the consortium

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	John Kaldis, John Soldatos, Christos Georgoulis	AIT	21/02/2014	Initial Table of Contents Proposal and List of Contributors
V015	John Soldatos	AIT	28/04/2014	Updated Table of Contents following two consortium tele-conferences about the VITAL architecture; Updates to Section1
V016 V017	John Soldatos	AIT	28/04/2014	Inputs to Section 2
V018	John Soldatos	AIT	28/04/2014	Inputs to Section 3; Glossary of VITAL (architecture-related) terms
V019	Riccardo Petrolo	INRIA	05/05/2014	Inputs to Sections 4 and 5 (on INRIA's components)
V020	Christos Georgoulis	AIT	05/05/2014	Initial Inputs on Development Tools
V021	Gregor Schiele	NUIG	06/05/2014	Inputs to Sections 4 and 5 on Data Management Services
V022	Andrea Martelli	REPLY	06/05/2014	Inputs to Section 4 (Hi Reply Platform) and Section 5 (Security Modules)
V023	Fotis Stamatelopoulos	SiLO	06/05/2014	Inputs to Section 5 (BPM, Monitoring Tools, Governance Tools)
V024 V025	John Soldatos	AIT	06/05/2014	Inputs to Section 3 (overview of VITAL architecture); Various edits
V027	John Soldatos	AIT	06/05/2014	Inputs & Revisions to Section 3 (overview of VITAL architecture)
V028 V029	John Soldatos	AIT	09/05/2014	Inputs on Platform Provider Interfaces
V30	John Soldatos	AIT	14/05/2014	Sequence Diagrams in Section 3
V31	John Soldatos	AIT	19/05/2014	Sequence Diagrams in Section 4
V32	Gregor Schiele	NUIG	20/05/2014	Additional text and sequence diagram for VITAL platform access
V33	Riccardo Petrolo	INRIA	20/05/2014	Updated Descriptions of Information Filtering and Discovery modules (Section 5)

Copyright © 2015 VITAL Consortium

V34	John Soldatos John Kaldis	AIT	20/05/2014	Updates to the Introduction
V35 V36	Miguel Angel Mateo	ATOS	21/05/2014	Added chapters 3.3 and 5.4
V36 V37	John Soldatos John Kaldis	AIT	21/05/2014	Revisions to Functional View of the architecture following audio-conference
V41	Fotis Stamatelopoulos	SiLo	21/5/2014	Sequence Diagram and designer use cases descriptions for the BPM/Orchestration module
V42	Riccardo Petrolo	INRIA	22/05/2014	Additions about FIT integration
V43	Andrea Martelli, Giulio Vito de Musso	REPLY	26/05/2014	Sequence diagrams on VITAL to HI Reply interactions
V44	John Soldatos	AIT	26/05/2014	Various Edits and Conclusions
V45	John Kaldis	AIT	26/05/2014	Chapter 6 – Mapping of Integrated Scenarios to the Architecture
V46	John Soldatos	AIT	28/05/2014	Preparation of Version for Quality Control
V47	Nathalie Mitton	INRIA	28/05/2014	Quality Control
V48	Gregor Schiele	NUIG	30/05/2014	Quality Control
V49	John Soldatos	AIT	31/05/2014	Addressing QR Comments; Proof-Reading
V100	Gregor Schiele	NUIG	31/05/2014	Ready for EC submission
Draft	Gregor Schiele	NUIG	12/06/2014	Text duplication fix, minor update

Copyright © 2015 VITAL Consortium

DOCUMENT HISTORY (Second Submission to EC)

Rev.	Author(s)	Organisation(s)	Date	Comments
V1.10	John Soldatos,	AIT	10/01/2015	Revised Structure based on Plan for Addressing the Reviewers'; Recommendations
V1.11	Yildirim Huseyin Umut	ATOS	19/01/2015	Inclusion of revised version of the CEP module (Section 5.7)
V1.12	Angelos Lenis	SiLO	19/01/2015	Inclusion of revised verion of the BPM module (Section 5.8)
V1.13	Katerina Roukounaki, Kasper de Graaf	AIT, IMAGES	20/01/2015	Revised version of the Camden scenario on mobile working
V1.14	John Soldatos	AIT	29/01/2015	Various Edits and Fine-Tuning; Inclusion of Table with changes vs. previous submitted version
V1.15	Martin Serrano	NUIG	29/01/2015	Quality Review & Technical Review
Draft	Martin Serrano	NUIG	30/01/2015	EC submission – Draft version

Copyright © 2015 VITAL Consortium

TABLE OF CONTENTS

Copyright © 2015 VITAL Consortium

Copyright © 2015 VITAL Consortium

Copyright © 2015 VITAL Consortium

LIST OF FIGURES

Copyright © 2015 VITAL Consortium

LIST OF TABLES

Copyright © 2015 VITAL Consortium

TERMS AND ACRONYMS

ARM	Architecture Reference Model
BPEL	Business Process Execution Language
BPM	Business Process Management
CEP	Complex Event Processing
DPWS	Device Profile for Web Services
EC2	Elastic Compute Cloud
ESB	Enterprise Service Bus
EVDS	Electronic Violation Detection System
FCAPS	Fault Configuration Accounting Performance Security
FIT	Future Internet Testbed
FTP	File Transfer Protocol
GSN	Global Sensor Networks
GPS	Global Positioning System
GUI	Graphical User Interface
ICO	Internet-Connected-Objects
IERC	European Research Cluster in the Internet of Things
IoT	Internet-of-Things
ISO	International Organization for Standardization
LCS	Lane Control System
MVC	Model View Controller
PPI	Platform Provider Interface
RDF	Resource Description Format
RTMS	Remote Traffic Microwave Sensor
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SPI	Service Provider Interface
SSL	Secure Sockets Layer
SSN	Semantic Sensor Networks
SSO	Single Sign On
UC	Use Case
UML	Unified Modelling Language
VMS	Variable Message Signs
VUAIs	Virtualized Unified Access Interfaces
X-GSN	eXtended Global Sensor Networks
XML	eXtensible Markup Language

Copyright © 2015 VITAL Consortium

SECTIONS/PARAGRAPHS REVIEWED IN THIS VERSION

Section/Paragaphs	Remarks
3.1	Revised explanations on how ARM has been used
5.4	Updated information and more detailed description of the CEP Module
5.7	Updated information and more detailed description of the BPM/Workflow Module
5.7 5.8	Fixed Typos and Editing Issues
6	Complete rewriting of the Camden Use Case in terms of mapping to VITAL architecture

RECOMMENDATIONS & RELEVANT CONSORTIUM ACTIONS (RESPONSES TO REVIEWERS)

Reviewers Comments and Recommendations (as listed in the official report)	Consortium Responses and Relevant Remedial Actions
This deliverable described the overall virtualization architecture proposed by the project and then proceeds to provide descriptions of its various components, concluding with a discussion of how the different functional elements map to the two application scenarios. Although most functional components are adequately detailed there are several other areas where this is not the cases, notably the description of CEP and the BPM. Both of these functional elements are only described in a rudimentary manner and lack sufficient detail.	In the revised version of the deliverable we have provided more elaborated descriptions of the Complex Event Processing (CEP) and Business Process management (BPM) components (see revised subparagraphs 5.4 and 5.7).
For example the description presented in Section 5.4 on CEP would represent any CEP engines and has no specific reference to particular features and capabilities that match the particular application setting. There is no performance evidence that supports the choice of development language and there is a considerable lack of clarity on	In the revised version of the document Section 5.4 has been revised/updated in a major way in order to provide adequate information about the CEP design and the way it addresses VITAL requirements. More information has also been provided about the design and the implementation of the CEP module, including information on the range of

Copyright © 2015 VITAL Consortium

what components are reused and what are new. Indeed the statements presented here appear to contradict D1.4 on the same matter.	background developments that will be reused, including how these developments will be advanced in the scope of the VITAL project.
Similar for BPM the description is that of a generic system and there is little attempt to show how the specific IoT requirements are reflected on this feature.	The revised version of the deliverable includes major updates and enhancements to Section 5.7 in order to provide a more detailed description of the BPM module. This description includes: (A) A more detailed overview of the functionalities of the BPM module and their alignment to VITAL requirements and use cases; (B) A more detailed description of the designer tool that will accompany the BPM module.
Section 3 shows that IoT-A principles are not fully understood, as for instance the conceptual difference between Reference Model and Reference Architecture does not seem to be reflected in the text.	The revised version of the document includes updates to Section 3 in order to reflect the difference between Reference Model and Reference Architecture.
Section 6 maps system features and modules to scenarios. This Section appears to include several errors that raise concerns about the application of appropriate quality assurance process. Notably Table 37 on page 69 concerning the Camden Town scenario refers to capabilities of the Istanbul setting e.g., in the Platform Access and Data Acquisition Layer. Moreover, reference to Xively and X-GSN do not appear to match the actual technologies implemented in this case. Indeed, reference is made to the fact that access to sensors or other IoT systems is not available in this setting.	The Camden demonstrator scenario in Section 6 has been completed rewritten in order to: (A) Reflect latest decisions of the consortium regarding the functionalities of the demonstrator; (B) Map the updated scenario to modules of the architecture. The updated version comprises more details about the data sources to be used in the scope of the demonstrator. Based on these updates, the description of the demonstrator contained in the first release of the deliverable is no longer relevant. Note that the revised description reflects the functionality and implementation of a mobile worker's use case.
An editorial comment: Section 5.7 and 5.8 are the same (copy pat), just with a wrong title – please fix by removing redundant section 5.8.	In the revised version of the deliverable, this has been corrected.

Copyright © 2015 VITAL Consortium

1 INTRODUCTION

1.1 Scope

VITAL is researching an integrated virtualized paradigm for the development, deployment and operation of smart city applications, which emphasizes the collection and processing of data streams from multiple sensors and IoT platforms across the urban environment. The introduction of this paradigm will be supported by the VITAL middleware platform, which will facilitate access to – as well as processing of – various data streams. It will also provide tools for developing applications and managing VITAL-based deployments. The purpose of this deliverable is to introduce the VITAL virtualization architecture as a set of structuring principles between the various technical modules and building blocks that will comprise the VITAL systems and applications. Along with this virtualization architecture, the deliverable provides a set of technical specifications for the various components and building blocks of VITAL-compliant systems. These technical specifications will drive the detailed design and implementation of the various building blocks as part of later workpackages of the project's workplan.

This deliverable is the third and final deliverable of workpackage WP2 of the project. Earlier deliverables dealt with the documentation of stakeholders' requirements (i.e. deliverable D2.1) and the specification of references scenarios associated with the VITAL paradigm (i.e. deliverable D2.2). These earlier deliverables serve as a basis for producing the architecture and technical specifications in this deliverable, given that these specifications should address the already identified requirements and scenarios. Overall, the present deliverable concludes WP2 of the project, on the basis of a tangible technical output that will drive the development of the main scientific and technological streams of the project as part of workpackages WP3, WP4 and WP5. At the same time, the specified architecture will serve as a basis for integrating the VITAL platform and applications in WP6.

1.2 Audience

The deliverable is primarily targeting VITAL consortium members, notably researchers and engineers that will deal with the design and implementation of the project's technical results in other workpackages (i.e. WP3, WP4, WP5 and WP6). At the same time, the deliverable is likely to stimulate the interest of EU researchers working on IoT and smart cities. For example, selected results of the deliverable could be of great interest to researchers and projects of the IERC cluster.

The target audience for the deliverable includes also researchers working on the IoT Architecture Reference Model (ARM) specified as part of the FP7 IoT-A project and the first activity chain of the IERC. This is because the VITAL architecture intends to adopt concepts from the ARM, thereby becoming an instantiation of this reference model for smart cities. Currently, the ARM model is maintained and enhanced by the IoT Forum, which is therefore another audience that will find the present document interested.

Finally, the VITAL architecture will be instantiated on the basis of real-life IoT platforms, which will be integrated based on the VITAL middleware. The communities of developers and users of these platforms (including open source communities) are therefore among the target audiences of the present deliverable.

Copyright © 2015 VITAL Consortium

1.3 Summary

The VITAL platform is designed as a large scale distributed system, which leverages data and services from a variety of IoT systems in order to ease the development, deployment and operation of integrated cross-silo applications in smart cities. This deliverable is devoted to the introduction of the VITAL architecture, which will drive the developments of the project in several technical workpackages (notably WP3, WP4, WP5 and WP6). The principles driving the design and specification of the VITAL architecture include the need to maximize reuse, the need to virtualize access to IoT systems, the needs to support standards-based modeling of ICOs and IoT services, the need to boost modularity and technological longevity, as well as the need to adopt and exploit open source technologies. At the same time the specification of the VITAL architecture is driven by the batch of requirements and scenarios specified in earlier deliverables of WP2 of the project, namely D2.1 and D2.2. Also, the VITAL architecture has taken into account constructs and principles specified as part of the IoT Forum's ARM (Architecture Reference Model), thereby being aligned to the ARM.

The VITAL architecture is specified as a loosely coupled multi-layer system. It comprises several modules, which map to the main research areas of the project, including an ICO discovery module, a data filtering module, a CEP (Complex Event Processing) module, an IoT services orchestration module for smart cities and more. It also envisages the provision of IoT system/platform agnostic data management services based on the transformation of datasets from IoT systems to datasets compliant to the VITAL smart city semantics and ontologies (which will be based on the W3C SSN ontology). These platform agnostic data management services could be supported on the basis of a Cloud-based ontology management system. Furthermore, the VITAL architecture specifies a range of application development, management/monitoring and governance modules, which are accompanied by respective tools.

As part of the VITAL architecture, the project specifies a number of interfaces that will play a key role in the integration of VITAL based systems and applications. One of these interfaces is the Platform-Provider-Interface (PPI), which specifies the way in which IoT systems will be integrated in the VITAL platform. The four IoT systems/platforms that have been selected in order to demonstrate the platform agnostic functionalities of the VITAL system will implement and expose the PPI as a means of enabling their platform agnostic integration with the VITAL platform. Furthermore, interfaces for virtualized platform agnostic access to the VITAL platform functionalities (such as filtering and CEP) are specified. Also, the architecture specifies push and/or pull interfaces for the exchange of ICO and IoT services data and metadata across the various modules of the architecture.

In addition to supporting a wide range of IoT functionalities for smart cities, the VITAL architecture makes provision for the support of several non-functional requirements. Indeed, several architecture choices support performance, reliability and scalability constraints. Furthermore, the VITAL architecture boosts deployment flexibility, which is a key towards producing deployment configurations that can indeed meet non-functional requirements.

As part of the architecture specification, we also provide insights on the implementation technologies that will be used towards implementing and integrating

Copyright © 2015 VITAL Consortium

the various modules and building blocks of the VITAL architecture. These technologies will be further refined as part of the implementation work in the various workpackages. This implementation work may also lead to refinements to the VITAL architecture, as part of the iterative and evolutionary project's workplan. Such refinements may be introduced as part of the need to address smart city applications requirements. In the scope of this deliverable an initial mapping between the functionalities of the VITAL integrated scenarios (in Camden and Istanbul) and the capabilities of the VITAL architecture has been attempted, as a means of validating that the architecture is on the right track. Overall, the present deliverable concludes successfully WP2 of the project and provides a sound basis for building key research, scientific and technical results in other workpackages.

1.4 Structure

The deliverable is structured as follows:

- Section 2 presents the driving principles of the VITAL architecture. Furthermore it illustrates how the VITAL requirements and use cases (described in earlier deliverables) have been taken into account towards producing the architecture.
- Section 3 provides an overview of the VITAL architecture in terms of its main functionalities and building blocks. In particular, it presents the main modules comprising the VITAL architecture, along with the main structuring principles that will drive their integration.
- Section 4 is devoted to the presentation of the IoT systems (i.e. platforms), which will be integrated based on the VITAL paradigm. The main emphasis is put on how these systems will be integrated in the VITAL platform, rather on describing the main features of the platforms. The later have already been presented as part of earlier deliverables.
- Section 5 provides more details for each one of the modules that comprise the VITAL architecture. Special emphasis is given to the description of their interfaces to other modules. In several cases UML diagrams outlining their interactions with other VITAL modules are presented as well.
- Section 6 provide an initial analysis on the appropriateness of the VITAL platform to support the integrated VITAL scenarios and use cases, notably the scenarios and use cases that have been presented in earlier deliverable D2.2.
- Section 7 is the final and concluding section of this deliverable.

Copyright © 2015 VITAL Consortium

2 VITAL ARCHITECTURE BACKGROUND AND DRIVING FORCES

2.1 Basic Principles

VITAL is designing and implementing a novel paradigm for the virtualized integrated development and deployment of IoT applications for smart cities, notably applications that leverage data and services from multiple IoT systems (i.e. IoT platforms and applications). This paradigm will be empowered by the VITAL platform, which will be integrated and built based on the VITAL architecture. The development of the VITAL architecture is driven by the following principles and characteristics:

- **Maximum reuse**: The VITAL architecture is specified in a way that maximizes reuse of available IoT platforms and components, notably background components developed by the project partners, including the OpenIoT, GSN and FIT platforms. To this end, the implementation technologies that are prescribed for the instantiation of the architecture are in several cases based on existing developments of the partners, while brand new components (i.e. components developed from scratch) will be implemented based on technologies that are compatible to the background components to be used. Overall, the VITAL architecture is influenced by state of the art developments of the partners in the areas of IoT platforms, IoT middleware components and semantic technologies, notably semantic technologies associated with sensor ontologies and linked sensor data.
- **Virtualization**: As the project title indicates, the VITAL platform will boost a virtualization approach to accessing and processing data from multiple IoT platforms and applications. The VITAL virtualization paradigm will emphasize virtualized access to and processing of data streams stemming from various IoT platforms. Processing of data streams will include filtering and CEP-based processing. Furthermore, a virtualized orchestration of the various ICOs and ICO services will be realized.
- **Modularity**: The VITAL architecture emphasizes modularity, since it is based on the composition/integration of a number of different modules, which will be contributed and/or developed by the different partners. This modularity will drive several aspects and activities of the project's work plan, notably the integration of the VITAL platform.
- **Standards-based**: The VITAL architecture is specified and implemented taking into account popular standards for data modeling, data exchange (e.g., XML, RDF) and system integration technologies (e.g., SOA, RESTful technologies). At the same time, it adopts a standards based approach to the modeling of sensors and ICOs based on the W3C SSN ontology. Furthermore, the design of the VITAL architecture takes into account modeling concepts and principles laid out as part of the ARM model for IoT architectures of the IoT Forum. These standards have therefore influenced the design of several VITAL modules, as well as the specification of the relevant interfaces between them.
- **Loosely Coupled**: VITAL will be implemented as a loosely coupled service oriented distributed system. This choice is driven by the fact that VITAL will be a highly distributed and heterogeneous system comprising a wide range of

Copyright © 2015 VITAL Consortium

modules, which will be developed independently by the various consortium partners. Furthermore, several of these modules will be implemented based on diverse technological platforms (e.g., JavaEE, MS .NET, mobile platforms). Therefore, a loosely coupled approach to the integration of these components is necessary in order to minimize technological dependencies and provide flexibility in the project's development, deployment and integration tasks.

- **Open Source**: VITAL will be implemented based on a number of open source components and technologies, including the OpenIoT and GSN IoT middleware platforms. Hence, the VITAL architecture makes provisions for the successful licensing and reuse of these technologies, but also for the implementation of new components as open source. Major parts of the architecture will be realized as open source software, as a means of achieving wide adoption and maximizing the potential impact of the project to the IoT and smart cities communities.
- **Driven by VITAL requirements and scenarios**: As illustrated in later sections, the specification of the VITAL architecture is striving to satisfy requirements identified as part of earlier deliverable D2.1, while it is also driven by the reference scenarios identified in deliverable D2.2.

Table 1 maps the above principles and drivers of the VITAL architecture to specific design and implementation choices, which are detailed in the following sections.

Table 1: Mapping Drivers of the VITAL Architecture Design and Implementation Choices

VITAL Architecture Driver or Principle	Design and Implementation Choices
Maximum Reuse	• Reuse of existing IoT systems, including legacy IoT platforms (GSN/X-GSN, Hi Reply, OpenIoT) and IoT applications. • Reuse of ontologies and related data management services (e.g., W3C SSN management via Virtuoso).
Virtualization	• Unification of the semantics of data streams stemming from different platforms. • Specification and Implementation of virtualized functions (e.g., CEP, filtering, discovery) for accessing and processing data streams in a platform-agnostic way. These functions will become accessible via the so-called VUAIs (Virtualized Universal Access Interfaces).
Modularity	• Specification of the VITAL system as a set of distributed interconnected modules. • Assignment of owner / responsible for each one of the different modules.
Standards-based	• Exploitation of XML standards for data modeling and exchange. • Adoption of W3 SSN ontology for modeling sensors and

Copyright © 2015 VITAL Consortium

	ICOs.
Loosely Coupled	• Service Oriented approach to designing and integrating the VITAL platform. • Use of RESTful interfaces (on different technology platforms) for the integration of the VITAL components.
Open Source	• Exploitation of open source IoT components and platforms (e.g., GSN/X-GSN). • Reuse of the open source OpenIoT infrastructure for semantic interoperability of IoT data streams.
Driven by VITAL Requirements and Specifications	• Support for functional requirements listed in Deliverable D2.1 (i.e. filtering, eventing, ICO discovery, CEP, management, monitoring and more). • Support for the main reference scenarios listed in Deliverable D2.2.

2.2 Compliance to Standards and IoT-A ARM

A main outcome of the FP7 IOTA project and the first activity chain of the IERC cluster has been the specification of a general-purpose architecture reference model for IoT systems. This ARM is nowadays taken up, sustained and evolved by the IoT Forum. VITAL has taken into account the concepts and modeling principles of the ARM as illustrated later in this document.

2.3 Alignment to VITAL Functional Requirements

The following tables illustrate how the various requirements (listed in deliverable D2.1) have been taken into account in the development of the VITAL architecture. The various requirements are clustered into groups as in deliverable D2.1. Note that only the Requirements Groups (RG) with relevance to the VITAL architecture are listed. However, there are other RGs, which will be addressed as part of the design of VITAL modules in other workpackages, rather than at the level of the VITAL architecture. For example the requirements of the RG on tools will be addressed in WP5, while requirements of the RG on the VITAL applications will be addressed in WP6 of the project.

Table 2: Impact of Requirements Group (RG1) on VITAL architecture specifications

Requirements Group 1 (RG1): IoT Platforms Virtualization Requirements		
No.	**Requirement**	**Impact on VITAL Architecture**
R1.1	The VITAL platform should act as a hub enabling the orchestration and coordination of IoT interactions with various IoT platforms and applications available in a smart city	The VITAL Architecture should include a module enabling the orchestration of IoT services on various IoT platforms.

Copyright © 2015 VITAL Consortium

R1.2	The VITAL platform should (in the context of the project) support the federation of 2-4 legacy (silo) IoT platforms and applications (as listed in the DoW)	The VITAL architecture should specify uniform interfaces for accessing diverse IoT systems through a single API.
R1.3	The VITAL platform should enable access to the data of sensors / ICOs of the underlying silo platforms and applications.	The VITAL architecture should provide platform-independent access to ICO data. To this end, data should be virtualized in a platform agnostic layer.
R1.4	The VITAL platform should enable access to the metadata and properties of sensors / ICOs of underlying silo platforms and applications.	The VITAL architecture should provide platform-independent access to ICO metadata. To this end, metadata should be virtualized in a platform agnostic layer.
R1.5	The VITAL virtualization platform should support a general model for sensors and ICOs, which should be mapped to underlying silo platforms and applications	The VITAL architecture should prescribe data management services on the basis of a common ontology for representing ICO data and services.
R1.6	The VITAL platform should provide platform agnostic functionalities for data access, data collection, information filtering and event generation.	The VITAL architecture should prescribe interfaces from platform-agnostic access to functionalities such as filtering and CEP.
R1.7	The VITAL platform should allow to present and visualize information streams stemming from any of the underlying silo IoT applications and platforms.	The architecture should allow VITAL applications to access information streams from any IoT system through the virtualized interfaces.
R1.8	The VITAL platform should provide the means for querying information from the underlying platforms and applications.	The architecture should allow VITAL applications to construct and execute queries over information streams from multiple IoT systems through the virtualized interfaces.
R1.9	The VITAL platform should provide tools that facilitate development and deployment of applications that leverage information and data sources from underlying silo plat-forms and smart city applications.	The architecture should prescribe a set of development and deployment tools operating over the virtualized interfaces.
R1.10	VITAL should provide facilities/tools for monitoring data flows through the VITAL platform.	The architecture should prescribe a set of management and monitoring tools operating on the virtualized interfaces.

Copyright © 2015 VITAL Consortium

Table 3: Impact of Requirements Group (RG2) on VITAL architecture specifications

Requirements Group 2 (RG2): Information Fusion and Event Generation Requirements		
No.	**Requirement**	**Impact on VITAL Architecture**
R2.1	VITAL should provide the means for combining/fusing data streams and information from multiple heterogeneous IoT platforms	The VITAL architecture should define modules for information fusion operating over the platform agnostic data management services of the VITAL platform.
R2.2	VITAL should provide the means for combining/fusing heterogeneous data streams from various data sources, including physical sensors, virtual sensors, social media, databases and user devices	The VITAL data handling/management services of the VITAL architecture should make provisions for accessing and processing data streams from both physical and virtual sensors.
R2.3	VITAL should support rule-based combination/fusion of heterogeneous data streams	The VITAL architecture should include data processing modules that enable the application of rules (such as CEP).
R2.4	VITAL should support the fusion and combination of data streams with different velocities including static data streams and live data feeds.	The VITAL architecture should allow the processing of information streams in the type (live, static) and IoT platform agnostic way.

Table 4: Impact of Requirements Group (RG3) on VITAL architecture specifications

Requirements Group 3 (RG3): Sensor Modeling and Virtualization Requirements		
No.	**Requirement**	**Impact on VITAL Architecture**
R3.1	VITAL should support the modeling and persistence of data streams stemming from both physical devices and sensor processing components that can deliver observations.	The VITAL architecture should prescribe ways and interfaces for access information from both physical devices and sensor processing components, without limitations in terms of the IoT system that provides access to them
R3.2	VITAL should support generic representations of sensors and data streams, which shall be mapped to data streams representations in other IoT platforms and systems.	The VITAL architecture should cater for the transformation of data stemming from any IoT system to the generic representations used in the VITAL platform. An appropriate ontology should be used to support the needed generic representations.

Copyright © 2015 VITAL Consortium

R3.3	VITAL should model and manage both data and metadata about sensors, ICOs and data streams.	The data models used in the VITAL architecture should be able to deal with both sensor data and metadata.
R3.4	The VITAL sensor models should be generic, abstract and expose high-level semantics.	The data management services of the VITAL architecture should be based on a semantic models appropriate for IoT services and applications, such as the W3C SSN ontology.

Table 5: Impact of Requirements Group (RG4) on VITAL architecture specifications

Requirements Group 4 (RG4): Sensors and ICOs discovery requirements		
No.	**Requirement**	**Impact on VITAL Architecture**
R4.1	VITAL should support the dynamic discovery of sensors and ICOs across all different platforms attached/connected to the VITAL platform	The VITAL architecture shall comprise a module dedicated to the discovery of sensors and ICOs. This module should operate at the platform agnostic layer in order to enable the discovery of ICOs from any of the underlying IoT systems.
R4.2	VITAL should support the dynamic discovery / querying of data from sensors and ICOs that are attached to different IoT platforms and applications whatever their mobility or connectivity pattern.	The VITAL architecture shall comprise modules for querying sensor and ICO data from different sensors/ICOs. These modules should take into account the mobility characteristics/patterns of the ICOs/devices.
R4.3	VITAL should support the discovery and use of sensors/ICOs that comply to different identification technologies	The ICO discovery module shall operate at the VITAL platform agnostic data management layer, where the details of the IoT and IoT identification technologies are not visible.
R4.4	VITAL should support discovery/querying of data that are commonly used in smart city applications	The VITAL architecture and tools shall support and facilitate the execution of common queries associated with smart city applications (i.e. frequently asked queries).

Table 6: Impact of Requirements Group (RG5) on VITAL architecture specifications

Requirements Group 5 (RG5): Requirements for Standards Support and Open Source Development		
No.	**Requirement**	**Impact on VITAL Architecture**

Copyright © 2015 VITAL Consortium

R5.1	VITAL should reuse open source components (notably from the open source projects that have been built by the project partners)	The VITAL architecture implementation makes provisions for reusing components from OpenIoT, GSN and FIT platforms.
R5.2	VITAL should release results as open source as a means to achieving wide adoption by solution providers for smart cities	N/A
R5.3	VITAL should support interfaces/bindings to standards-based platforms (such as OGC and W3C SSN based platforms)	Relevant IoT standards should be taken into account in specifying the interfaces of the VITAL platform to underlying IoT systems as part of the VITAL architecture.
R5.4	VITAL should support standards-based technologies and techniques for sensor modeling	The data models and ontologies prescribed as part of the VITAL architecture should adhere to standard sensor modeling effort. The W3C SSN ontology is one such standard.
R5.5	VITAL should support standards-based technologies for accessing smart city data flowing through the VITAL platform	The virtualized access interfaces of the VITAL platform should leverage standards-based technologies (e.g., REST) and representations.

Table 7: Impact of Requirements Group (RG6) on VITAL architecture specifications

Requirements Group 6 (RG6): Requirements regarding supported applications		
No.	**Requirement**	**Impact on VITAL Architecture**
R6.1	VITAL should provide application agnostic middleware components and libraries in the areas of sensor/ ICO modeling, sensor/ICO discovery, sensor/ICO data collection, information/data fusion, CEP and event generation	The VITAL architecture should prescribe modules and building blocks for ICO discovery, filtering, data fusion and CEP.
R6.2	VITAL should provide and prioritize the support of IoT-enabled smart transport, smart energy and smart security applications	The data models used in the VITAL architecture should be suitable to support smart transport, smart energy and smart security applications.
R6.3	VITAL should provide and prioritize support for sensors, devices and data streams used in smart transport, smart energy and smart security applications	(Same as above) The data models used in the scope of the VITAL architecture should be suitable to support smart transport, smart energy and smart security applications.

Copyright © 2015 VITAL Consortium

R6.4	VITAL should provide and prioritize support for queries on sensors and ICOs data streams used in smart transport, smart energy and smart security applications	(Same as above) The data models used in the scope of the VITAL architecture should be suitable to support smart transport, smart energy and smart security applications.
R6.5	VITAL should provide and prioritize support for sensor/ICO driven processes associated with smart transport, smart energy and smart security applications	The orchestration and workflow management modules of the VITAL architecture should support specific workflows compliant to the listed applications.
R6.6	VITAL should provide and prioritize support for the sensors and data streams used by IMM and Camden (i.e. GPS, cameras, CCTV streams, road sensors etc.)	These sensors and data streams should be taken into account the design and specification of the VITAL ontology and related data management techniques.

Table 8: Impact of Requirements Group (RG9) on VITAL architecture specifications

Requirements Group 9 (RG9): Security and Privacy Management Requirements		
No.	**Requirement**	**Impact on VITAL Architecture**
R9.1	VITAL should identify manage-ment functionalities ensuring single-sign-on over all the IoT systems connected/interfacing to VITAL	The interfaces of the VITAL platforms to the underlying IoT systems should make provisions for authenticated and authorized access to the data and services of these systems. The VITAL platform should manage the authentications and authorizations of the VITAL applications to all of the underlying IoT systems.
R9.2	VITAL should manage authorizations for accessing data and devices attached to different IoT systems	Relevant management of authentication and authorization credentials should be performed at the modules and interfaces that will enable access to the underlying IoT systems.
R9.3	VITAL should support authorizations at platform, application and sensor/ICO levels	The VITAL platform access mechanisms should support authorizations at all these levels, as soon as the underlying IoT systems support them. Relevant pro-visions should be made at the design of uniform interfaces between VITAL

Copyright © 2015 VITAL Consortium

		and underlying IoT systems.
R9.4	The VITAL platform should respect privacy management rules and constraints of all the underlying IoT systems/ platform interfacing to VITAL	The VITAL platform access mechanisms should support any privacy management features/capabilities of the integrated IoT Systems.
R9.5	The VITAL platform should support integrators in the identification of laws and regulation that are applicable to their developments (including both EU and national/local regulation).	Relevant provisions should be taken into account in the scope of the detailed design of the VITAL management, development and governance tools in WP5.
R9.6	The VITAL platform should support encryption for private data (e.g., personal data)	Data encryption capabilities should be prescribed at the VITAL data access interfaces.
R9.7	The VITAL should provide support for data anonymization processes	Data anonymization capabilities should be prescribed at the VITAL data access interfaces. These should be applicable to specific classes/types of VITAL applications.

The tables provide some initial architecture considerations towards addressing the VITAL requirements. In most cases, these requirements will also be addressed based on choices/decisions that will be made during the detailed design of several VITAL modules as part of other technical workpackages. Therefore, the above listed considerations should be seen as a starting point for designing the VITAL architecture and producing relevant technical specifications. However, they are not completely addressed based on the architecture design, since they impact the detailed design as outlined above.

2.4 Alignment to VITAL Non-Functional Requirements

The following table illustrates how several of the non-functional requirements listed in deliverable D2.2 have been taken into account towards driving the VITAL architecture specification.

Table 9: VITAL Architecture Considerations towards addressing identified non-functional requirements

Code	VITAL Non-Functional Requirements	Architecture Design & Implementation Choices
VITAL-NFR-1	Performance and Scalability	• Deployment of the VITAL data management and storage services within a cloud infrastructure. • Ability for direct low-latency interactions of the

Copyright © 2015 VITAL Consortium

		VITAL applications and tool, instead of imposing that all interactions will be implemented through the VITAL virtualization layer. • Implementation of abstract interfaces to IoT platform (PPIs) for the flexible integration of additional IoT platforms and data streams.
VITAL-NFR-2	Development and Deployment Flexibility	• Abstraction of the interfaces to the various underlying IoT platforms and data streams. • Provision of visual tools for development and deployment tasks.
VITAL-NFR-3	Management and Configuration	• Design and implementation of a management and monitoring module over the VITAL virtualization layer. • Management and monitoring of ICOs, data streams, applications and services.
VITAL-NFR-4	Extensibility and Expandability	• Design of abstract interfaces for the integration of new IoT platforms (PPIs). • Design of adapter modules for the transformation of the semantics of any data stream to the semantics of the VITAL ontology and related data management services.
VITAL-NFR-5	Compliance with standards	• Modeling of the VITAL ontology as an enhanced version of the W3C SSN ontology. • Adoption of the IERC ARM principles for the VITAL IoT architecture.
VITAL-NFR-6	Authentication and Authorization	• Design of a security module for managing authentication and authorizations. • Mapping of VITAL credentials to authentication/ authorization mechanisms of the underlying IoT platforms.
VITAL-NFR-7	Single Sign On	• Incorporation of SSO techniques to the VITAL security module.

2.5 VITAL Scenarios and Use Cases

As already outlined, the VITAL architecture is specified / designed to fulfill all the reference scenarios and use cases that have been specified as part of deliverable D2.2. The following table illustrates specific design and implementation choices concerning the VITAL architecture, which were made following the consideration of each one of the reference scenarios listed in D2.2.

Copyright © 2015 VITAL Consortium

Table 10: Architecture Design and Implementation Choices driven by the VITAL reference scenarios

Code	VITAL Reference Scenario and/or Use Case	Architecture Design & Implementation Choices
UC01	Developer Reflection Interface	• Design of a module for accessing-collecting data streams from multiple platforms. • Design of a module for unifying the semantics of the data streams of different platforms. • Design of a common ontology for platform-agnostic modeling of the ICOs data and metadata.
UC02	Mobile Density Map	• Design of a common ontology for platform-agnostic modeling of the ICOs data and metadata. • Design of interface for platform-agnostic access to the IoT data streams. • Design of lightweight interfaces for providing data from the VITAL system to mobile applications.
UC03	Reprogrammable Sensor Discovery Use Case	• Provision and design of module for discovering sensors and ICOs according to their state. • Implementation of semantic querying module over the VITAL ontology, towards selecting ICOs based on their type, location and more.
UC04	Smart Lighting Use Case	• Design of (event-driven) actuating interfaces over the VITAL ontology and related data management services. • Identification of events / CEP.
UC05	Security Policy Use Case	• Incorporation of encryption mechanisms for information exchanged through the VITAL architecture (for selecting use cases).
UC06	Authentication and Authorization Use Case	• Design of SSO (Single-Sign-On) mechanisms for the VITAL platform to access information from any of the underlying IoT platforms. • Implementation of role-based authentication at the level of the VITAL architecture for accessing any of the underlying IoT platforms.
UC07	Smart Meeting and Mobility Use Case	• Design of a module for processing / filtering data streams from multiple platforms. • Design of modules supporting production of complex events (CEP modules).
UC08	Smart Cities BPM	• Design of a module that can orchestrate services

Copyright © 2015 VITAL Consortium

	Use Case	(including data access and actuation services) provided over ICOs. • Orchestration of the service interactions regardless of the source platforms of the ICOs.
UC09	VITAL Health Map Use Case	• Design and implementation of a monitoring module on top of the VITAL semantically unified data and metadata (i.e. ICT data). • Persistence and management of metadata about the various IoT platforms, IoT services and IoT applications.
UC10	Platform Accounting Service Use Case	• Need to keep track of usage data for ICOs and applications as part of the VITAL data management services. • Design of modules for management and monitoring of QoS and accounting parameters.
UC11	City Incident Reporting Use Case	• Provide the means for the identification of ICOs by type and location. • Enable access, homogenization and visualization of data streams from multiple platforms.
UC12	Smart City Application Development Use Case	• Design and implementation of a visual editor enabling specification of processing functionalities over data streams from multiple platforms. • Reuse and extension of OpenIoT IDE in the above-listed direction.
SC01	Camden Integrated Scenario	Combination of filtering, CEP, ICO discovery and orchestration / workflow management functionalities in a single scenario.
SC02	Intanbul Integrated Scenario	Combination of filtering, CEP, ICO discovery and orchestration / workflow management functionalities in a single scenario.

3 HIGH LEVEL OVERVIEW OF VITAL ARCHITECTURE

3.1 Terminology – Glossary

In order to present the VITAL architecture, we will commonly use a number of technical terms, which are listed in the following table. Note that some of these terms are already defined as part of IoT literature. In the context of the following table, they are refined according to requirements and functionalities pertaining to the VITAL project.

Copyright © 2015 VITAL Consortium

Table 11: Glossary of Terms relating to the VITAL architecture

Term	Meaning
Data Stream	Sequence of data units (e.g., packets) associated with the transmission/reception of information from a source ICT module to a destination module. In VITAL data stream originate from physical or virtual sensor.
(IoT) Discoverer	A software module that enables the discovery of Internet-Connected-Objects that possesses certain properties (e.g., in terms of their type and location).
Physical Sensor	A physical device that measures a physical quantity and provides a corresponding signal to an observer ICT module (e.g., a software component or algorithm).
Platform Provider Interface (PPI)	The boundary between the VITAL platform and a third-party IoT platform. The implementation and programming of the interface is realized according to the Platform Provider API.
Platform Provider Application Programming Interface	Specifies the (software-based) interaction of the VITAL platform with a third-party IoT platform. It provides an abstraction of the functionalities that should be provided by IoT platforms, in order to enable their integration to the VITAL platform.
Virtual Sensor	An ICT software module that provides observations about the surrounding environment in the form of a data stream.
VITAL Architecture	The modules comprising the VITAL platform, along with the structuring principles and the interfaces between them.
VITAL Cloud Infrastructure	A public or private cloud infrastructure that hosts the VITAL ontology/data management services, as well as other VITAL modules (e.g., ICO Discoverer, CEP, filtering).
VITAL Data Management Services	A set of services enabling management on datasets (graphs) complying to the VITAL Ontology.
VITAL Ontology	A formal ICT-based representation of knowledge about IoT platforms and applications for the smart cities domain. The VITAL ontology includes a shared library, which denotes the types, properties and interrelationships of those concepts.
VITAL Platform	A middleware platform (implemented as a distributed system), which facilitates the development, deployment and operation of IoT-based Smart Cities applications based on virtualized functionalities.
IoT System	A system that provides services based on ICOs. In the scope of VITAL both IoT platforms and IoT applications are classified as IoT systems.

Copyright © 2015 VITAL Consortium

3.2 Influences from the IERC Architecture-Reference-Model (ARM)

VITAL aims to provide a mechanism to integrate and use Internet Connected Objects (ICOs) and their services across multiple IoT architectures, platforms, ecosystems and business contexts, through the convergence and federation of different IoT eco-systems and applications.

For this purpose, VITAL will offer Virtual Interfaces, IoT-architecture agnostic techniques, and cross-context tools enabling access, management and orchestration of large number of distributed and heterogeneous ICOs and their services.

Due to the complexity of the goal, and avoiding to reinvent the wheel, VITAL has taken into account the work performed in the scope of the FP7 IoT-A project and their IoT-A Reference Model (http://www.iot-a.eu/public) (ARM), as a guidance to design the VITAL architecture. The ARM has been used as a source of building blocks (e.g., protocols, interfaces, components), which can be used in order to assemble a concrete architecture. VITAL integrates several of these building blocks, with particular emphasis on blocks that deal with services creation, orchestration and protocols, and less emphasis on low-level networking concepts and protocols. The ARM has also been used as means of deciding the structuring of these building blocks into a concrete architecture.

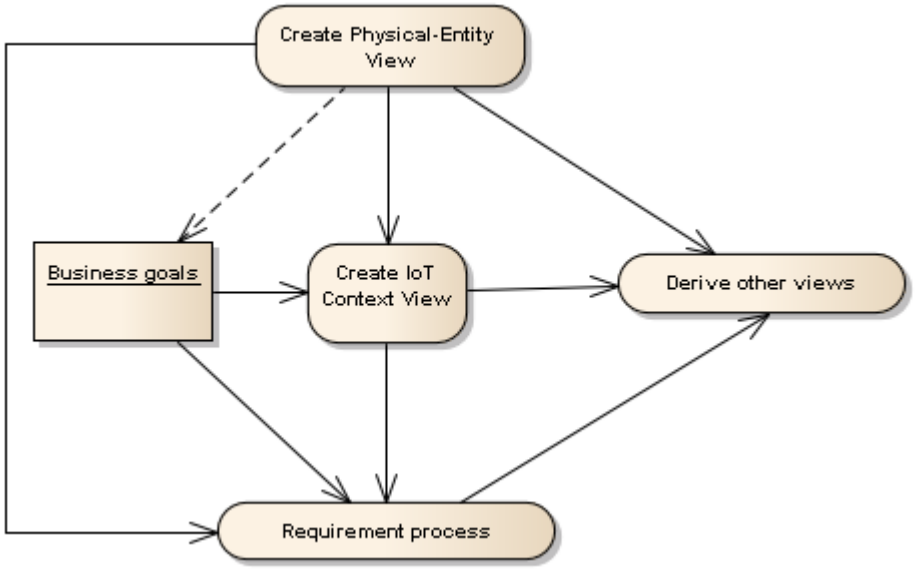

Figure 1: UML activity diagram of the IoT architecture-generation process

The following paragraphs of this sub-section do not perform a complete design of the architecture for VITAL, but rather depict the basic lines of the ARM architectural model that have been taken into account in the development of the VITAL system architecture.

Figure 1 outlines the activities involved in the generation of an architecture according to the IoT-A Reference Model, these activities in priority order are to:

- create the Physical-Entity view;
- create the IoT Context view;

Copyright © 2015 VITAL Consortium

- derive other views;

as discussed in the following paragraphs.

3.2.1 Physical-Entity view

Due to the nature of VITAL, which aims to integrate/federate many different IoT platforms of diverse nature, it is not possible to focus on only one set of physical objects or environments. VITAL should be able to work with any kind of physical entity.

With this premise, and although the architecture could be devised based on the Use Cases, it is preferable not to make any restriction at this point to design a platform as open as possible. Despite this line of no restrictions, there are certain considerations to keep in mind regarding security and privacy. Since the objects' services and data may lead to information that is subject to legal privacy restrictions (e.g., mobile phones), accessing, security and orchestration functionalities must provide the means to meet these restrictions according to the rights of each role defined in VITAL.

3.2.2 IoT Context view

The evolution of the IoT during the last years has been led by different actors like corporations, universities, public organizations, etc. This has resulted in multiple architectures, standards and platforms and a highly fragmented IoT landscape with systems that have been developed and deployed independently, therefore leading to several technological silos.

In smart cities (i.e. VITAL's focus), the above-mentioned fragmentation is also present, thereby leading to IoT application silos within modern smart cities. Note that technological silos are in most cases caused by organizational silos in local government (e.g., the separation of law enforcement from the department of transportation and from the public works department results in different IoT deployments and associated technical silos). Indeed, smart cities are nowadays characterized by the presence of multiple fragmented IoT deployments spanning different applications areas including smart homes, smart industrial automation, smart transport, smart buildings and more.

From the previously mentioned IoT landscape where VITAL aims to serve as a horizontal integrator layer, the following points are derived:

- VITAL aims to develop, deploy and operate across multiple IoT architectures, platforms and business contexts, providing platform and business context agnostic access to IoT.
- Due to this heterogeneous variety of IoT silos, there is no clear vision about the identity of external entities, services and data. Going beyond the specifications of selected IoT platforms for the use cases, VITAL aims to specify and implement Virtualized Unified Access Interfaces (VUAIs) as a mean for accessing heterogeneous ICOs, which reside in different IoT middleware platforms and are managed by different administrative entities.
- External IoT platforms should provide an interface to let VITAL access their data, real time info, and also their services.

Copyright © 2015 VITAL Consortium

- Once VITAL is operating, it will bring an increasingly number of opportunities for smart city applications due to the opportunity of analysing a lot of different data streams from IoT platforms of diverse nature and purpose. This will likely include generating information that is subject to privacy restrictions. In order to avoid this issue, VITAL should enforce data access policies.
- VITAL will implement a set of tools enabling service developers to build applications leveraging virtualized interfaces and federating mechanisms in order to access and combine the capabilities of the ICOs in al compatible and interoperable way.
- VITAL will focus on the design and implementation of a paradigm for managing ICOs, ICO data and services in a platform agnostic manner. VITAL will also specify higher-level management entities (i.e. composite ICOs, services comprising multiple ICOs, applications comprising multiple IoT services etc.), which will be managed regardless of the underlying architectures and platforms.
- VITAL will leverage the great amount of data collected by underlying IoT platforms to build more complex and meaningful applications with the combination of different techniques, such as data analysis, real time processing or data mining, processing information steaming from different and heterogeneous IoT platforms which have different aims.

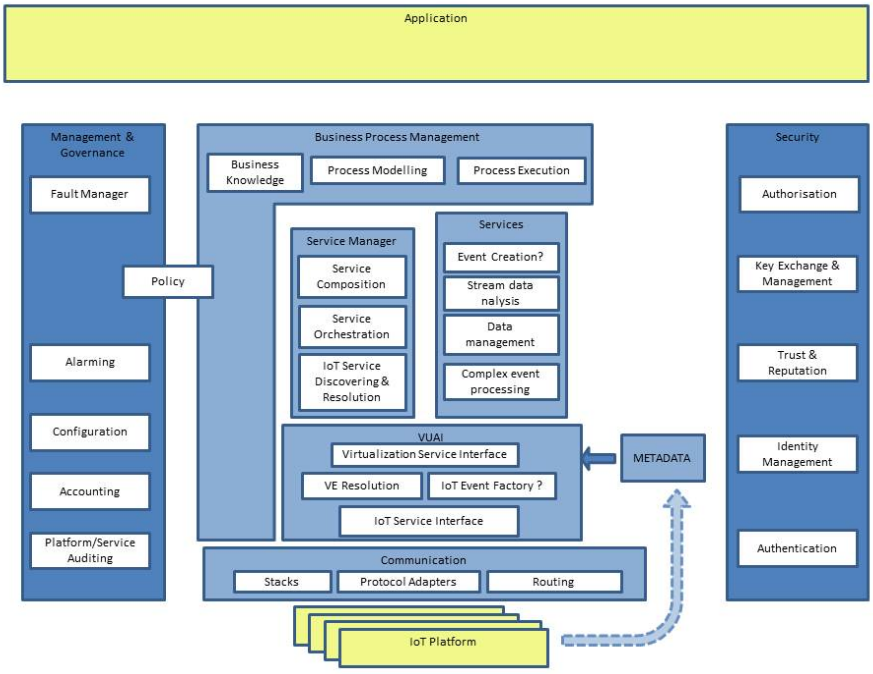

Figure 2: Mapping of VITAL Modules to the ARM

3.2.3 VITAL Functional view

After presenOnce these basic lines have been derived, and starting from IoT-A functional model, the VITAL functional view can be outlined. In order to simplify the process, VITAL functional view will fulfil IoT-A model grouping several related functionalities.

Copyright © 2015 VITAL Consortium

Figure 2 illustrates a mapping of envisaged VITAL modules to the ARM functional modules. Furthermore, the list of functional modules of VITAL and how they cover the different functional groups of IoT-A are outlined in the following paragraphs.

3.2.3.1 Communciation Modules

In order to facilitate communication across IoT platforms, the following modules are envisaged:

- Protocol Adapters, which:
    - Enable communication between VITAL and IoT platforms. The adaptation involves information transformation to the data models that will be used in VITAL i.e. the VITAL ontology).
    - Adapt and translate protocols and data formats to/from the ones agreed for VITAL and the ones used by IoT platforms. In this way, VITAL will be able to handle information from different formats stemming from the underlying platforms including for example json, xml, cdr, csv, zigbee, rest and more.
- Stacks, which:
    - Ensure VITAL platforms ability to use several stacks (tcp, udp,…) and manage relevant QoS parameters (e.g., queue sizes, reliability, encryption parameters, timing).
- Routing, which focuses on:
    - Deal with different network interfaces.
    - Ensuring that when a message has to be routed it is able to find the next hop in a network.

Note that the above-listed modules will be part of the adaptation of existing IoT systems / platforms to the VITAL architecture and paradigm. VITAL will specify a PPI (Platform Provider Interfaces) enabling VITAL systems to access other IoT platforms/systems through the very same interfaces. The above-listed modules will be part of the work for adapting IoT systems/platform to the requirements of the PPI, which are described in following paragraphs.

3.2.3.2 VUAI (Virtualized Unified Access Interfaces)

In order to facilitate virtualized access to IoT data and services, the following ARM modules are envisaged:

- Virtualization Resolution: It provides the functionality to retrieve associations between Virtualization layer and IoT services and offer the required API. It also manages associations between the Virtualization layer and IoT services.
- Virtualization Service interface: This is an interface offered by the Virutalization layer to the VITAL applications and tools.
- IoT Service Interface: This serves as interface sending and receiving info to/from resources (i.e. services, ICOs) respectively.

VITAL envisaged the design and implementation of the directory of (virtualized) IoT resources (i.e. services, ICOs, data), as a means of offering the above listed functionalities and interfaces.

Copyright © 2015 VITAL Consortium

3.2.3.3 Service Manager

Service management functionalities envisaged in ARM, could be used in VITAL in terms of:

- Service Composition, focusing on the resolution of relationships among IoT services, enhanced services and users or appplications requests.
- Service Orchestration, focusing on the resolution of relationships among IoT services, enhanced services and user/app requests. Service Orchestration deals also with the orchrestration of relationships between IoT services, enhanced services and apps. In this context an IoTservice can work for several enhanced service or applications, which an enhanced service could also work for several applications.

3.2.3.4 Business Process Management

In the scope of the VITAL DoW document, BPM functionalities are also envisaged. This can be mapped to the following ARM building blocks:

- Proccess Modeling, which provides an environment for the modelling of IoT-aware business processes that will be serialised and executed. Note that Process Modelling provides also tools for modelling processes.
- Proccess Execution, which executes processes that have been modelled as described above. This execution is achieved by utilising services and IoT-services through the Service Organization Functional group. Process Execution: (a) Ensures the alignment of application requirements with service capabilities and (b) Provides the means for executing applications after resolving services and IoT-services.

3.2.3.5 Management and Governance

The scope of the VITAL platform includes also several management and governance functionalities. VITAL is planning to support FCAPS functionalities. Relevant functional modules of the ARM include:

- Fault Manager, which monitors and distils information provided by other blocks to detect faults. This module helps isolating the source of the fault/issue.
- Alarming Manager, which can be considered as a sub-function of Fault Manager
- Configuration Manager, which retrieves system configuration and provides the means for managing system configuration parameters.
- Accounting Manager, which provides the means for calculating Key Performance Indicators (KPIs).

These ARM functional modules will be part of VITAL's management modules and applications. The relevant modules are reflected in the following paragraph, which provides an overview of the VITAL architecture.

Copyright © 2015 VITAL Consortium

3.3 Overview of VITAL Architecture

3.3.1 Functional View

An overview of the VITAL architecture is provided in Figure 3, which illustrates the main building blocks of the VITAL platform. The architecture is structured according to the following layers:

- **IoT Systems Layer**: This is the lowest layer of the architecture, where various IoT systems (including IoT platforms and IoT applications) deployed in urban environments reside. VITAL is about integrating and processing information from multiple IoT systems (i.e. IoT platforms and applications). Therefore, this layer comprises the various IoT systems that are virtualized and integrated as part of the VITAL platform. In order to validate the VITAL paradigm, the project will attempt to integrate and process data from the X-GSN, FIT, Hi Reply and Xively.com platforms (as explained in deliverable D2.2). However, the VITAL architecture will be flexible in the integration of additional platforms and/or applications, as soon as these platforms/applications expose the VITAL PPI (Platform Provider Interface). The concept of the VITAL PPI is illustrated in the following sections. Note that the integration of data and services from virtually any IoT system in a smart city is a key to alleviating information fragmentation in the urban environment.

- **Platforms Access and Data Acquisition Layer**: This layer enables access to data and services of the underlying platforms in a secure and authenticated way. Its role is to access the low-level capabilities of the IoT systems (through the VITAL PPI) and accordingly to transform the acquired data and metadata into a common data model (i.e. expressed based on the VITAL ontology). Hence, the functionality of the layer involves translating the low-level and poor semantics of the individual platforms to the high-level and richer semantics of the VITAL platform. At this layer we also specify the notion of the VITAL adapter module, which undertakes the transformation of the lower-level PPI semantics to the high-level VITAL ontology semantics. As already outlined, the whole process involves the authenticated and authorized access to the capabilities of the individual IoT platforms, which implies the implementation of an SSO mechanism when it comes to accessing multiple IoT systems (i.e. platforms and applications) from higher layers of the VITAL platform.

- **Platform Agnostic Data Management Layer**: This layer provides cloud-based functionalities for managing data and metadata that comply with the VITAL ontology. Hence, components residing at this layer are deployed within a public (e.g., Amazon EC2) or private cloud infrastructure. At this layer, the various data streams (and their metadata) are modeled and formatted according to the VITAL ontology. The offered services include data and metadata persistence, creation of new data, discovery of data subject to various criteria and more. Access to the data of the platform will be based on semantic web technologies and techniques, given the modeling of the data according to the VITAL ontology. The platform agnostic data management layer offers a wide range of services to higher level applications, which reside in the added-value functionalities layer. These services

Copyright © 2015 VITAL Consortium

will be accessible in a virtualized platform and location agnostic manner, through the VUAIs. The platform agnostic data management layer includes a Service Discovery module, which enables the look-up and resolution of IoT resources. In particular:

- **Service Discovery Module**: This module will provide a directory service for the virtualized IoT resources that will be accessible through the added value functionalities layer (e.g., the CEP and Filtering modules). IoT resources include IoT services, ICOs and IoT data. The functionality of the Service Discovery Module will be extensively used by other modules of the architecture, which will operate on the basis of available IoT services and functionalities. For example, the Orchestrator module will access the Service Discovery Module in order to dynamically access the list of services that could be combined and orchestrated in the scope of VITAL workflows. A key element of the service discovery module will be the ICO Discoverer. The latter will provide the means for discovering ICOs and data streams (in terms of both their data and metadata) on the basis of various criteria, such as the time, the location and the type of the various data streams. The ICO Discoverer module is important towards restricting the data sets to be accessed, processed and visualized to those that are relevant to the application at hand.

- **Added Value Functionalities Layer**: This layer comprises a set of complete services and tools, which leverage data and services from the platform agnostic management layer. The virtualized services of this layer will be provided by the following modules (depicted in Figure 3):
  - **Filtering Module**: The filtering module of the VITAL architecture provides the means for reducing the information associated with individual data streams persisted in the platform agnostic data management layer. It therefore reduces unwanted information, thereby optimizing processing performance and economizing on network bandwidth.
  - **Complex Event Processing (CEP) Module:** This module enables the processing of data streams for multiple sources in order to identify patterns and/or infer events. It may leverage the capabilities of the ICO Discoverer module in order to select data sources (data streams), as well as the capabilities of the filtering modules towards filtering individual data streams.
  - **Orchestration / BPM (Business Process Management) Module:** This module combines and manages multiple services from the above-listed modules, in order to deliver new added-value services. The combination of the various services is based on a workflow of service oriented components and interactions, which may be specified on the basis of rules. Apart from providing conventional BPM functionalities, this module will also deal with the uncertainty of sensor derived data and related sensor-triggered services. In providing conventional BPM functionalities, this module could leverage the capabilities of existing mainstream BPM framework, which could be appropriate enhanced towards supporting scenarios that involve uncertainty.

Copyright © 2015 VITAL Consortium

- **Smart City Applications Layer**: The main objective of the VITAL platform is to support the development, integration, deployment and operation of smart city applications, notably applications leveraging data from multiple IoT platforms and applications (including legacy IoT applications in urban environments). Smart city applications can be developed using the development tools of the project, while their operation can be managed by the management tools. However, all smart city applications are (during their operation) accessing data and services of the platform agnostic data management layer, including the discovery, filtering, CEP and BPM/orchestration services. In particular, this layer includes management and governance tools that enable the monitoring and configuration of IoT platforms, applications and services at various granularities. It also includes development tools enabling the development of smart city applications and services. The various applications and tools will be able to access data from the platform agnostic data management layer, as well as services offered from this layer.

A typical information flow through the layered VITAL system involves access to data and services of legacy IoT systems and their subsequent transformation to the platform agnostic VITAL data semantics in-line with the VITAL ontology. The added-value functionalities of the VITAL platform (including CEP, filtering, Orchestration) can then be enacted over the virtualized platform agnostic ICO data and services. To this end, these services can use the Service Discovery functionalities of the VITAL platform in order to locate and select the IoT data and services that they would like to use. Likewise, the development, management and governance tools of the project will be able to access virtualized platform agnostic data and services from the VITAL platform.

Note that the architecture of Figure 3 specifies a layering of the VITAL components. In principle this layering implies that each layer provides data and services to its upper layer and uses/leverages data and services from its lower (underlying layer). However, certain exceptions to this rule might occur in order to address specific requirements and use cases, such as use cases that involve stringent latency constraints. For example, real-time applications might need to access IoT services and data from one or more of the legacy IoT systems/platforms rather than from the VITAL layer of (platform agnostic) data management services. Therefore, there are cases where the layering of the VITAL architecture needs to be bypassed. In general those cases will be treated as legitimate (application specific) exceptions. The conditions and circumstances under which such exceptions occur will be specified in the scope of the implementation of VITAL integrated scenarios.

Copyright © 2015 VITAL Consortium

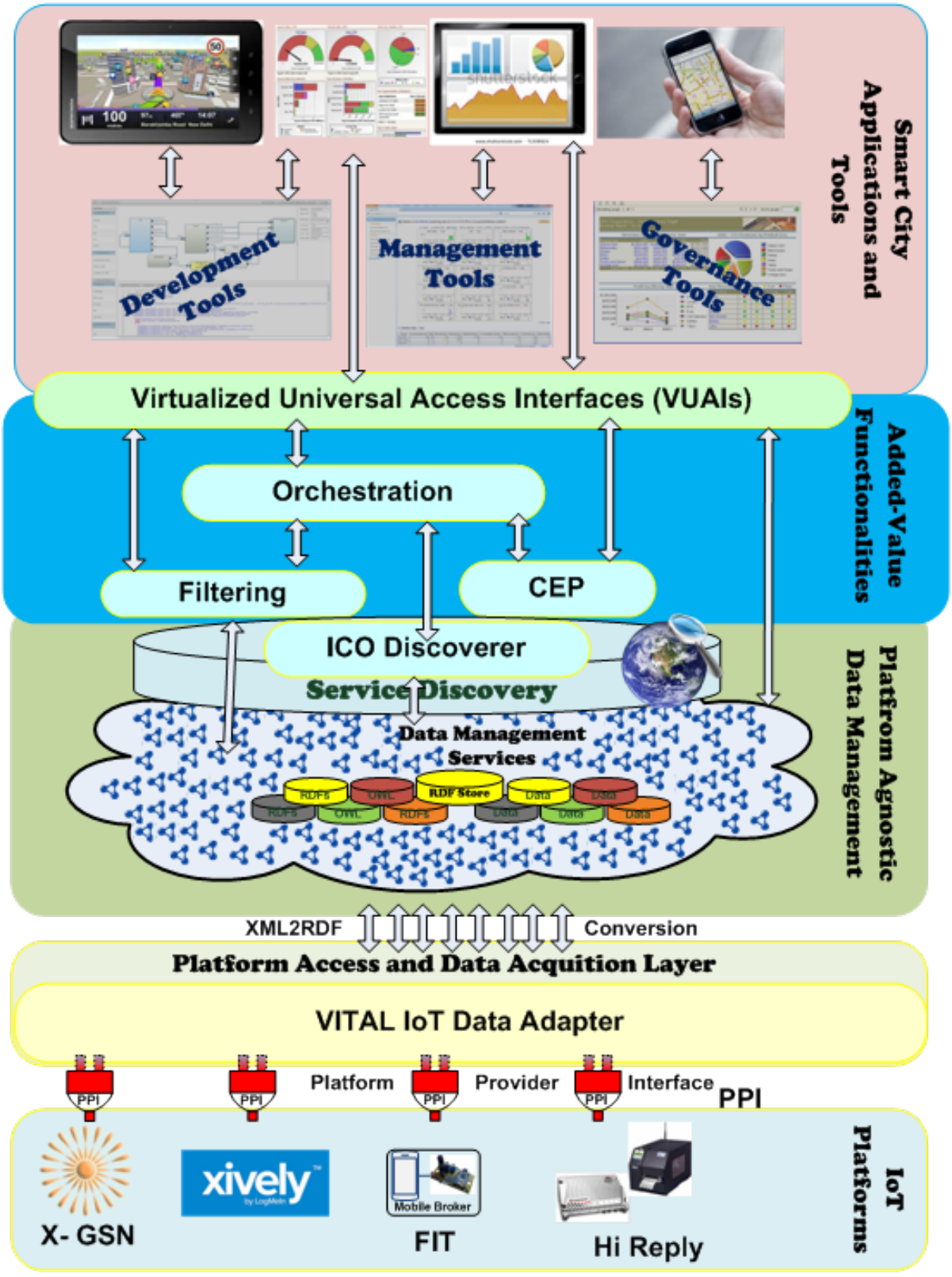

Figure 3: Functional Overview of the VITAL Architecture

3.3.2 Non-Functional Issues

The VITAL architecture makes provisions for the proper handling of several non-functional issues, including performance, scalability, extensibility, reliability, privacy and security. In principles the design of VITAL as a large scale loosely coupled distributed system, allows for the effective handling of several of the above-listed issues. In particular:

- **Scalability**: The scalability of the VITAL platform hinges on several of its features and design choices. Indeed, its cloud-based deployment can serve as a scalability vehicle at multiple levels, namely scalability in handling large numbers of service oriented interactions from the smart city applications, scalability in terms of persisting and managing millions of data streams, as well as scalability in terms of the number of underlying IoT platforms and applications (i.e. platforms/applications contributing data to the VITAL platform). Note that the various modules of the VITAL platform (e.g., data adapters, CEP, ICO Discover) can be all deployed within a cloud infrastructure in order to boost scalability. Note that the loosely coupled nature of the VITAL platform enables its scalable distributed deployment within cloud infrastructures. Nevertheless, we should also note that the exploitation of semantic technologies might hinder the system's scalability in terms of the number of data streams that could be handled. Relevant performance benchmarking will be planned as part of the validation tasks of the VITAL project.
- **Extensibility**: The architecture specifies vendor agnostic interfaces (PPIs) for accessing IoT platforms, as well as virtualized interfaces (VUAIs) for accessing data and services in a platform agnostic way. These interfaces boost the extensibility of the platform in terms of the deployment and integration of additional IoT platforms, as well as in terms of the deployment of additional data access and processing components (such as filtering and CEP algorithms). Combined with the scalability characteristics outlined above, these interfaces facilitate the extensibility of the VITAL platform. At the same time, the VITAL ontology can be also extended in order to accommodate specialized requirements of smart city applications (such as applications tailored to the needs of specific smart city domains (e.g., smart energy, urban mobility)).
- **Performance**: The existence of multiple middleware layers in the VITAL platform could be considered as a performance bottleneck, especially when it comes to assessing the latency of operations that involve real-time access to data feeds and functionalities of the underlying platforms. It is therefore envisaged that VITAL might have performance limitations for real-time applications. These limitations will be assessed as part of the project's validation and evaluation tasks. At this stage, the VITAL architecture specification should consider the possibility of bypassing certain layers of the architecture (e.g., through direct access to the PPIs from higher layers) as a measure of supporting real-time applications and interactions.
- **Reliability**: At the hardware/physical level, all the VITAL elements can be cluster and/or cloud deployed for enhanced reliability. At the software level, the VITAL architecture (as part of its data modeling and BPM) will take into account the uncertainty of sensor-based measurements and observations. This will boost the reliability and resilience of the VITAL applications.

Copyright © 2015 VITAL Consortium

- **Security**: The VITAL architecture includes authentication and authorization mechanisms for accessing the data and services of the platform. These mechanisms will be propagated to the security mechanisms of the individual IoT platforms with a view to ensuring authorized access to data streams and IoT services. To this end, authentication and authorization capabilities will be part of the PPI. In this way, smart city applications will be able to securely access data and services of the VITAL platform.

Despite the above-listed measures, the effective handling of non-functional requirements depends heavily on the proper deployment of the VITAL platform. The following paragraphs discussed such deployment issues, thereby providing the deployment view of the architecture.

3.3.3 Deployment Issues and Roles

From a deployment perspective, the various modules of the VITAL architecture will be deployed and operated by different business roles and stakeholders of the VITAL ecosystem. In the sequel, we illustrate the role of each one of the stakeholders in the VITAL architecture:

- **End-Users**: End-users of the VITAL applications access the VITAL architecture through the top-level smart city applications layer of the architecture. They will be typically using web-based and/or cloud-based smart applications via their desktop or (usually) mobile terminal device (i.e. Tablet PC, smart phone). Note that different types of end-users will be accessing different types of applications in particular: (A) Citizens and Businesses are likely to use operational-level applications that visualize data and information flows from the VITAL platform agnostic data management layer; (B) City authorities are likely to use management and strategic level applications boosting their decision making. The latter are likely to be customized based on the project's management and governance tools.
- **VITAL Smart City Services Providers**: Smart city services providers can take advantage of the VITAL (virtualized) model in order to offer (to end-users) access to integrated services i.e. services utilizing data and capabilities from multiple underlying IoT systems. To this end, smart city services providers are likely to deploy and operate a cloud infrastructure, which shall be hosting the platform agnostic data of the VITAL platform. At the same time, service providers will provide the added-value functionalities of the VITAL platform (e.g., discovery, CEP) over this cloud infrastructure. Moreover, they will use the management and governance tools of VITAL as a means of monitoring the IoT services that they offer to end-users. Note that VITAL Smart City services providers establish SLAs (Service Level Agreements) with providers of legacy IoT platforms and/or providers of legacy IoT applications in order to use their data.
- **Smart City Solution Providers and Integrators**: Smart city solution integrators are likely to deploy and operate the VITAL platform agnostic data management layer, as part of the implementation/integration of an integrated multi-platform IoT solution for smart cities. Furthermore, solution providers and integrators are the prominent users of the development tools of the project, as part of their effort to reduce the time and resources needed to develop VITAL-based applications.

Copyright © 2015 VITAL Consortium

- **Smart City IoT Platform and Applications Providers**: IoT platform providers are contributing one or more platforms (and/or applications) that reside at the lowest IoT platforms layer. At the technical level, IoT platforms and applications providers are likely to implement a PPI over their platform/application in order to make it compliant to the VITAL architecture. At the business level, IoT platform and/or application providers are likely to establish SLAs with the providers of the VITAL infrastructures and services. These SLAs will reflect their incentives for integrating their platforms within the VITAL platform.

The following table associates the above-listed stakeholders and roles with the various modules of the VITAL platform.

Deployment Roles	Relevant VITAL architecture modules
End-Users (Citizens, Businesses, City Authorities)	• Use VITAL Applications
VITAL Smart City Services Providers	• Deploy and operate VITAL Applications. • Deploy and operate the services of the «Platform Agnostic Data Management Layer» in the cloud. • Monitor the operation of the
Smart City Solution Providers and Integrators	• Develop VITAL Applications using the VITAL development tools. • Develop, deploy and operate the platform agnostic data management layer of the VITAL platform, which interfaces to the IoT systems.
Smart City IoT Platform and Applications Providers	• Establish SLAs with VITAL services providers. • Provide access to the data and services of their IoT systems.

Figure 4: VITAL architecture modules and their association with the various stakeholders

4 IOT PLATFORMS – IOT SYSTEMS

In the sequel we provide an overview of the integration of the four IoT platforms (which have been selected in the scope of D2.2) with the VITAL architecture. Note that these platforms provide the means for integrating IoT data and services to the VITAL platform. In the case of the project's applications at Camden and Istanbul, these platforms could act as the sensor/ICO middleware that will facilitate the integration of sensors and ICOs to the VITAL platform. Overall, the cities could either integrate their data directly to the VITAL platform or via the IoT platforms outlined in the following sections.

Copyright © 2015 VITAL Consortium

4.1 X-GSN Platform

4.1.1 Functional overview

GSN is a sensor/IoT middleware platform designed to facilitate the deployment and programming of sensor networks. It enables the collection and processing of data streams from any sensor or sensor networks and its subsequent persistence within a database. To this end, a simple XML format with the main data and metadata of a sensor network data streams has been defined and used to deploy/integrate data streams with the GSN platform. Thanks to the compliance of all data streams to this format, GSN based applications can be hardware-independent, thereby making the sensor network changes invisible to the application. As part of the OpenIoT project, an enhanced version of GSN (namely X-GSN) has been produced. X-GSN enhancements over GSN are focused on two main directions:

- The transformation of the data and metadata of the collected data streams to a semantically richer RDF format (complying to the W3C SSN ontology).
- The streaming of the data from the X-GSN platform to a cloud infrastructure, where data management services over W3C SSN RDF data models are deployed.

In the scope of VITAL X-GSN will be used as one of the underlying IoT platforms that will be integrated into the project's virtualization paradigm. Given the fact that X-GSN performs transforms already data streams to a W3C SSN RDF formats, the platform is partly realizing the data adapter concept of the VITAL architecture. In this way X-GSN provides a sound basis for validating the VITAL data transformation concept, yet additional customizations will be required in order to make X-GSN work with the VITAL data models and data management concepts. Overall, X-GSN will serve as one of the middleware platforms that will be used (during the project) in order to stream data to the VITAL platform-agnostic management layer. To this end, X-GSN will be adapted to the requirements of the project's PPI interface. At the same time, existing data transformation capabilities of the X-GSN platform will be taken into account and/or reused in order to implement the VITAL middleware for converting data derived from the PPI to data compliant to the VITAL ontology.

The following table reviews the main functionalities of the X-GSN platform in the scope of the VITAL architecture.

Table 12: Summary of X-GSN Platform Functionalities in VITAL

No.	Functionality
1	Integration of data streams based on the X-GSN mechanisms and respective XML data formats
2	Secure/authenticated access to X-GSN data streams
3	Secure/authenticated access to the GSN management and configuration capabilities in terms of available data streams, their metadata, data windows where data and provided etc.
4	Transformation of GSN compliant data to W3C SSN data (with appropriate enhancements as required by the VITAL ontology)

Copyright © 2015 VITAL Consortium

4.1.2 Implementation Technologies

X-GSN is a Java based software (requires Java JDK 1.6) and is deployed in a JavaEE container. X-GSN has an embedded web server based on Jetty. The Java language will be used in order to adapt the data access capabilities of the X-GSN to RESTful services compliant to the PPI.

4.1.3 Interfaces to other VITAL modules

As already outlined, X-GSN will interface to the platform access and data acquisition layer of the VITAL platform. The interface to be implemented with be based on the VITAL PPI.

4.2 FIT Platform

4.2.1 Functional overview

FIT IoT-LAB provides a very large-scale infrastructure facility suitable for testing small wireless sensor devices and heterogeneous communicating objects over large scale. IoT-LAB offers full support for embedded software development, ranging from direct access to node hardware to OS-level features. Developers can leverage the different APIs to build applications.

Table 13: Summary of FIT IoT-LAB Platform Functionalities in VITAL

No.	Functionality
1	Full support for embedded software development, ranging from direct access to node hardware to OS-level features.

4.2.2 Implementation Technologies

The platform uses JAVA, JSON and REST technologies.

4.2.3 Interfaces to other VITAL modules

For security/privacy requirements, it is not allowed to access resources from outside the FIT IoT-LAB platform. Therefore, data from the FIT sensors are sent via FTP to a repository, which will be accessible from the VITAL Platform, as shown in the diagram below.

Copyright © 2015 VITAL Consortium

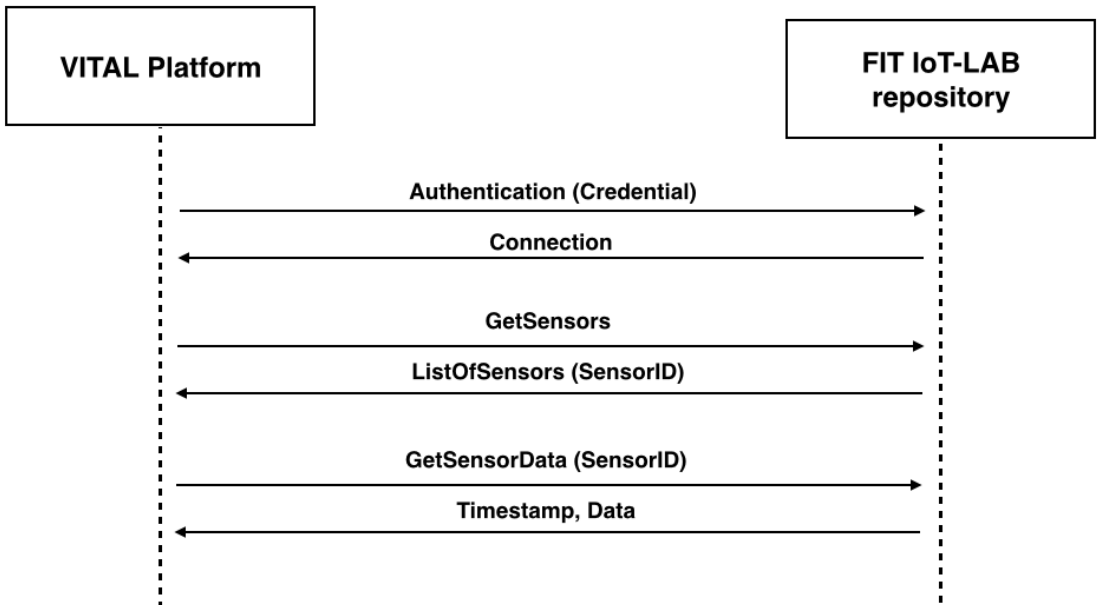

Figure 5: Sample Sequence of Interactions between FIT and the VITAL Platform

4.3 HI Reply Platform

4.3.1 Functional overview

The driving principles of the HI Reply Platform are pervasiveness, transparency, portability, flexibility, context sensitivity and a capacity to self-adapt and self-configure. This kind of flexibility is provided by a layered approach: an application gains data without the need to know and manage communication peculiarities of each device.

The layered approach is made of the following:

- *Hardware Adaptation Layer*: is in charge of establishing the connection with the devices by the given technologies and provides packet transmissions. Supported technologies are: WiFi, Ethernet, Zigbee, USB, Serial link RS232, GSM/GPRS/UMTS, RFID, GPS.
- *Network Layer*: asks and manages requests to open connections through the Hardware Adaptation Layer and provides suitable decoding of received packets. It is also in charge to communicate with the upper Web Service layer to instantiate the services that represent the capability device. In this way the HI Reply Platform hides to the Application layer the heterogeneity of devices and their communication protocols.
- *Web Services*: Network Layer deals with the communication with the upper Web Service Layer to instantiate the services that represent the capability device.
- *Application Layer*: manages the information obtained from the connected devices. The diversity of devices and their communication protocols are hidden from the Application Layer.

Copyright © 2015 VITAL Consortium

Table 14: Summary of Hi Reply Functionalities in VITAL

No.	Functionality
Hi1	Data gathering from several types of devices through 'simple' and 'intelligent' nodes using different communication protocols and technologies (WiFi, Bluetooth, ZigBee)
Hi2	Virtualization: sensors, devices and other kind of entities as services. The exposed services are described through homogeneous XML logical models, derived from a unique extendable XSD hierarchical data model which provides a classification of the IoT entities and related offered services (from the single functionalities of the devices to more complex context-aware services).
Hi3	Web Service based interfaces to sensors and virtualizers
Hi4	Service discovery
Hi5	Data persistency configurable on service basis to make available historical data for post-processing purposes
Hi6	Context awareness services based on semantic reasoning. Data coming from the nodes (heterogeneous sources) can be used by the Context Awareness Framework to recognize the current context and enable a reasoning process for decision-making. Decision-making will result in events to provide recommendations or alert or directed to node associated with the service to modify the actuators state. The logic of the service must be provided in form of inference rules.

4.3.2 Implementation Technologies

The HI Reply platform is based on a number of technologies that could be classified in different ways: by device type, by protocol layers, or functionally. Here is an overview:

- ***Operating Systems:*** Windows Server 2008 / Windows 7 ("Aggregator/Virtualizer" node), Windows CE ("Intelligent" node), Contiki ("Simple" node).
- ***Network:*** ZigBee, 6LoWPAN, Bluetooth, IPv4, IPv6
- ***Application framework:*** .NET Framework, .NET Compact Framework, .NET Micro Framework.
- ***Service level:*** SOAP, DPWS, WS-Discovery, WS-Security, REST, XML, HTTP, SSL.
- ***Reasoning:*** RDF, OWL, SPARQL, Jena Framework.

4.3.3 Interfaces to other VITAL modules

Ideally, the only interface between HI Reply and VITAL will be the abstract PPI, which will be realized between the HI Reply platform and the "VITAL IoT Platform

Copyright © 2015 VITAL Consortium

Access" module of the VITAL platform. This interface will enable the following functionalities:

- Query HI Reply for a snapshot of all of the available sensors and services (HTTP REST pull).
- Query HI Reply for specific sensor data/metadata (HTTP REST pull).
- Make sensed data available from HI Reply to VITAL. Mediation mechanisms to allow this to a Java platform shall be defined; they could be based on REST.
- Submit to the HI Reply platform information such as configuration information or user preferences.

Taking into account the services exposed by the HI Reply platform, VITAL requirements and PPI basic features, we outline here the expected interactions between the two systems and some initial ideas on how to integrate them.

First, VITAL has to authenticate to HI Reply in order to use its services. This will be possible through a Web Service interface and a mapping between VITAL users and roles to the HI ones. This mapping will reside into HI's credentials repository and will be used by the Authentication and Authorization Layer to provide SSO capabilities to the PPIs.

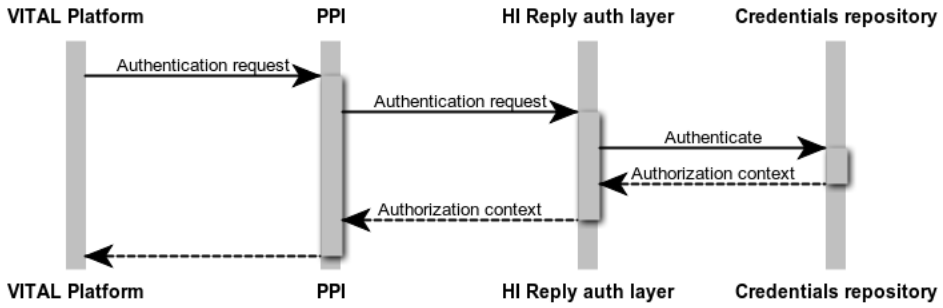

Figure 6: Authentication and authorization between VITAL and HI Reply

Once the authorization request has been dispatched to HI Reply, compared with the credentials and roles repository, and a response has been generated, VITAL should be able to store a temporary "grant" (in the form of a token) and use it in all of the subsequent requests.

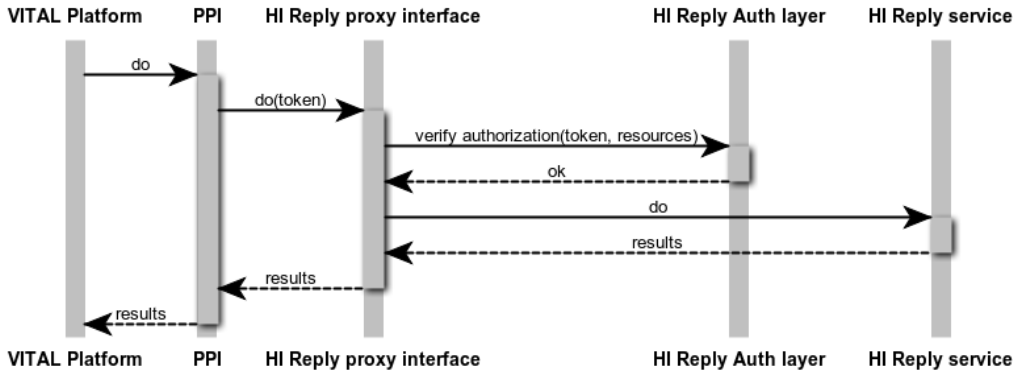

Figure 7: Generic authenticated interaction between VITAL and HI Reply

Copyright © 2015 VITAL Consortium

From that point, all of the requests done to the platform will be handled by a web service interface that will use the HI Reply authentication and authorization backend to check whether they are legitimate or not. The integration architecture will aim at decoupling as much as possible the HI platform and make it impossible for unauthorized users to directly contact platform and sensors services.

The VITAL Platform has to know, whether during a new IoT platform registration or at any further time, which are the metadata of any linked IoT platform. HI Reply allows the VITAL platform to gain those pieces of information just by calling the exposed APIs.

As shown Figure 8, the VITAL Platform will use the HI Reply interface to get HI Reply platform's metadata. VITAL will then insert into the local metadata repository the metadata received.

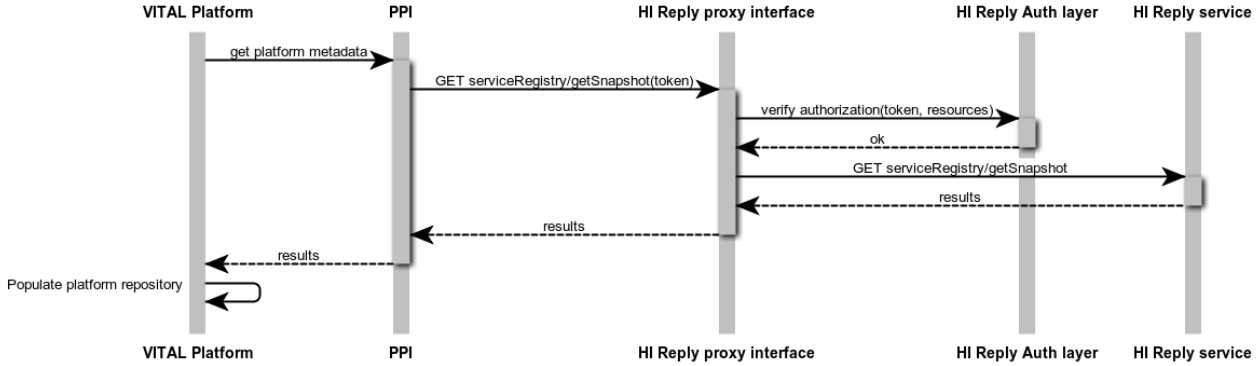

Figure 8 Service discovery workflow with HI Reply

A basic example of integration between VITAL's upper layers and HI Reply is sensor data sharing.

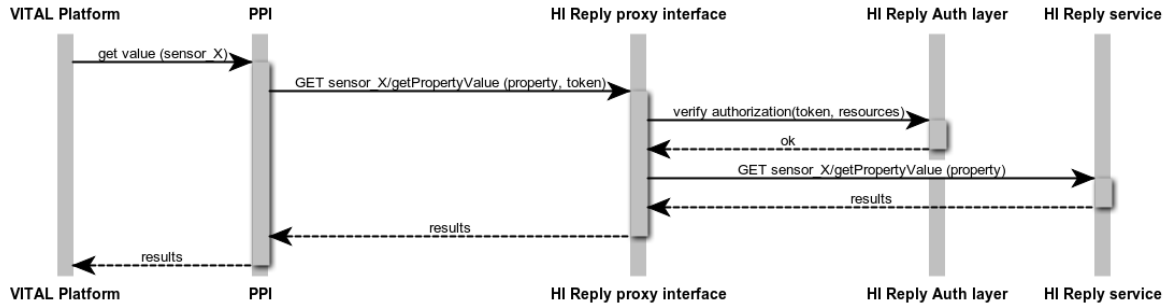

Figure 9: Sensor data query from VITAL to HI Reply

Moreover, the VITAL Platform will need to know the metadata of any of the sensors it will have access to. This way VITAL will know which are the properties of the sensors. In Figure 10 it is shown the process of sensor's metadata retrieving. The process is very similar to the previous one; in fact the PPI calls the same interface, passing a filter for the sensor selection.

Copyright © 2015 VITAL Consortium

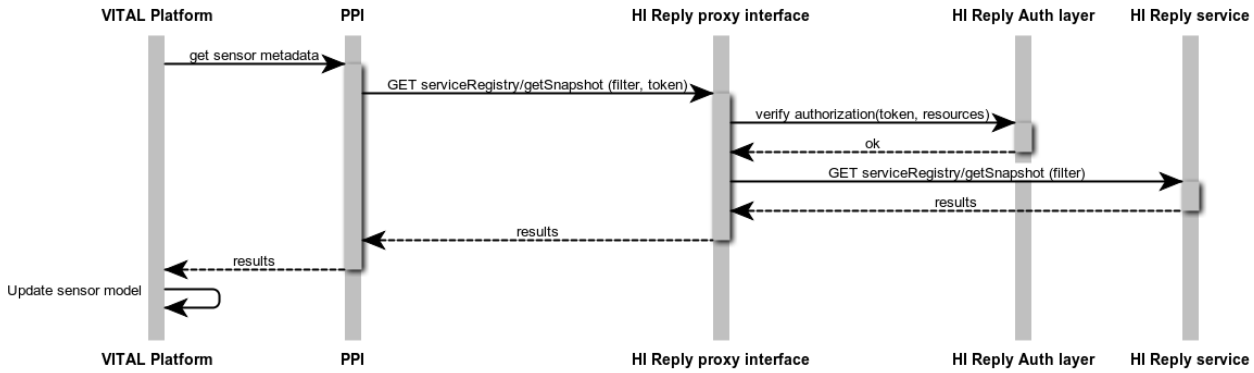

Figure 10 Sensor metadata retrieving from HI Reply

HI Reply platform allows the VITAL platform to retrieve a sensor's values history so that any VITAL reasoning engine can make statistical inferences over the sensors data and enhance user's experience.

As shown in Figure 11, the VITAL Platform can take advantage of the HI Reply proxy interface for the process of authentication/authorization and for the data history retrieving.

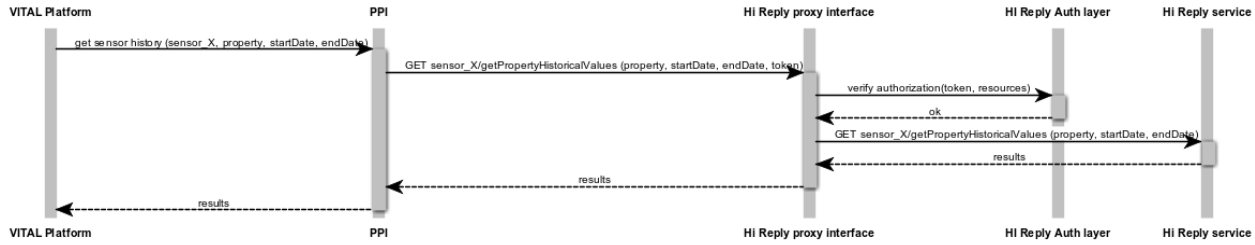

Figure 11: Workflow for the sensor history retrieving

HI Reply exposes all of its resources, being they physical sensors or context awareness features, as entities accessible via web services. This means that once a context awareness service (a set of inference rules to be applied on data) is configured on HI Reply, it can be seen as a sensor by VITAL. So, there could be no need for special developments to share reasoning features between HI and VITAL.

In general, every request will go through the formerly described authorization process. If it passes the security checks, it is forwarded to the specific sensor/service, otherwise it is dropped. A specific sensed property of a sensor is accessible in HI Reply through the `sensor_id/getPropertyValue(property)` service.

4.4 Xively Platform

4.4.1 Functional overview

Xively is a well known public IoT cloud platform, which enables its users to connect data feeds/streams to the cloud and accordingly to build applications based on the

Copyright © 2015 VITAL Consortium

Xively APIs. Xively support a wide range of technology APIs, including APIs based on Java and the Android SDK. On the basis of these APIs, the VITAL PPI will be supported over the Xively platform in order to enable access to Xively.com data feeds, as well as to their metadata.

The following table summarizes the main functionalities that will be implemented over Xively.com, in order to ensure the integration of the platform within VITAL.

Table 15: Summary of Xively Functionalities in VITAL

No.	Functionality
XIVE1	Access to Xively data streams data (pull/pull interfaces)
XIVE2	Access to Xively data streams metadata
XIVE3	Management of Connections/Sessions to Xively Cloud (including authentication/authorization)
XIVE4	Discovery of data streams (according to their data and metadata)

Overall, the above-listed functionalities will be part of the PPI implementation for Xively.

4.4.2 Implementation Technologies

The implementation of the identified functionalities will be primarily based on the Xively Java API. This API is essentially a robust RESTful interface which supports all CRUD (create, retrieve, update delete) functionalities for all its available resources. In the scope of the Xively.com platform these resources are categorized into seven different types, namely Products, Devices, Keys, Feeds, Triggers, Datastreams, and Datapoints. Each of these products is associated with a different set of attributes. The formats that can be obtained from the API are JSON, XML and CSV. These simple formats will be manipulated, processed and converted as required by the need to support the PPI interface.

4.4.3 Interfaces to other VITAL modules

Similar to the case of the other IoT platform, the Xively.com platform will be interfaced to the platform access and data acquisition layer of the VITAL platform, based on the PPI interface. To this end, the above-listed functionalities will be supported via the PPI.

Copyright © 2015 VITAL Consortium

5 VITAL ARCHITECTURE MODULES AND BUILDING BLOCKS

5.1 VITAL Platform Provider Interface

5.1.1 Overview and Role in the VITAL Architecture

The VITAL architecture specifies a vendor agnostic PPI (Platform Provider Interface), which is defined as the interface between the VITAL platform and a third-party IoT system (i.e. IoT platform or IoT application). The purpose of the PPI is to allow the VITAL platform access data, metadata and services of any platform through a unified interface. It therefore provides an abstraction for accessing the underlying IoT systems for the VITAL platform.

IoT systems providers and operators should implement and expose the PPI in order to enable the integration of their systems to the VITAL platform i.e. in order to make their system (i.e. platform or application) VITAL compliant. The VITAL platform will implement a general-purpose platform-agnostic IoT platform access layer, which will be using the PPI in order to access the low-level capabilities of any given IoT system. In the scope of the VITAL architecture, the PPI will boost a «learn once, use across different IoT platforms/systems» discipline, since it will allow VITAL platform developers to leverage a single API for accessing the underlying IoT systems.

As explained in the following paragraph, VITAL will implement a general purpose platform access and data acquisition layer, which will allow data and services from any IoT system to be used in the scope of the platform-agnostic data management layer of the platform. This layer of the VITAL architecture will undertake to convert ICO data derived through any platform (via the PPI) to formats and structures that comply with the VITAL ontology.

5.1.2 Data and Services Accessed via the PPI

The PPI will provide access to the following ICO data and services of an underlying IoT platform/system:

- Sensor/ICO data corresponding to observations provided by the IoT system.
- Sensor/ICO metadata including information about the sensor vendor, capabilities, location, reliability, accuracy and more.
- Other metadata about the streaming of the sensor data, such as the window and the timing of data production from the sensor / ICO data source.
- Metadata about the IoT system, such as the vendor of the system, the area/location it covers, the services that it provides and more.
- Information and metadata about the services provided by the IoT system, including a classification of these services according to their functionalities (i.e. filtering, CEP, discovery and more).
- Credentials for authenticating users against the IoT system and/or providing them with appropriate role-based access to its functionalities.
- Lifecycle information of an IoT system such as information about the status of the platform, the configuration functionalities provided by the platform and more.

Copyright © 2015 VITAL Consortium

- Information about the Service Level Agreement (SLA) between the IoT system provider and the VITAL service provider or integrator.

The above-listed data and information will be provided through services of the PPI. In order to provide opportunities for lightweight PPI implementations, not all of the services will be mandatory. On the contrary several of the services will be optional, in order to allow IoT system/platform providers to become VITAL compliant through only minimal effort for implementing the PPI. Apart from ensuring that PPI can be lightweight, the classification of the services to optional and mandatory will facilitate the integration of IoT systems that lack some of the above functionalities. The following table provides the envisaged classification of PPI data and services into mandatory and optional.

Table 16: Overview of PPI Data & Services and their classification in Mandatory and Optional

PPI Data & Services	Classification	Remarks
Sensor/ICO data	Mandatory	Access to sensor data of an IoT system is a prerequisite for the development of smart city applications, which leverage information from this IoT system.
Sensor/ICO metadata (e.g., vendor, capabilities, location, reliability, accuracy, time windows)	Mandatory	The location and type of a sensor are mandatory attributes. However, some of the specified data for a given sensor/ICO may not be available for given sensor, thereby being optional.
IoT system metadata	Mandatory	Each IoT system should provide a set of reflective properties.
Metadata about the services of an IoT system	Optional	It is likely that an IoT system will provide no services (and associated metadata).
Authentication Credentials	Mandatory	Credentials for authenticating the users should be provided in all cases. These should map to the security subsystems of the IoT platforms and applications. Along with the credentials, additional information about role-based access could be provided.
LifeCycle Information	Optional	The ability to access information about the status and operation of the underlying IoT system is particularly important for the implementation of the management and governance tools. This information could be a dynamic real-time version of IoT system level metadata.
SLA Information	Optional	Several IoT systems may not provide any SLA related parameters or attributes.

Copyright © 2015 VITAL Consortium

5.1.3 VITAL Platform – IoT System Interactions (via the PPI)

PPI functionalities will be accessed by the VITAL platform, which will be in charge of managing all the instances of the integrated IoT systems. To this end, the PPI functionalities will be callable by the VITAL platform. A set of sample interactions between the VITAL platform and any given IoT system (to be integrated) are provided in the following UML sequence diagram.

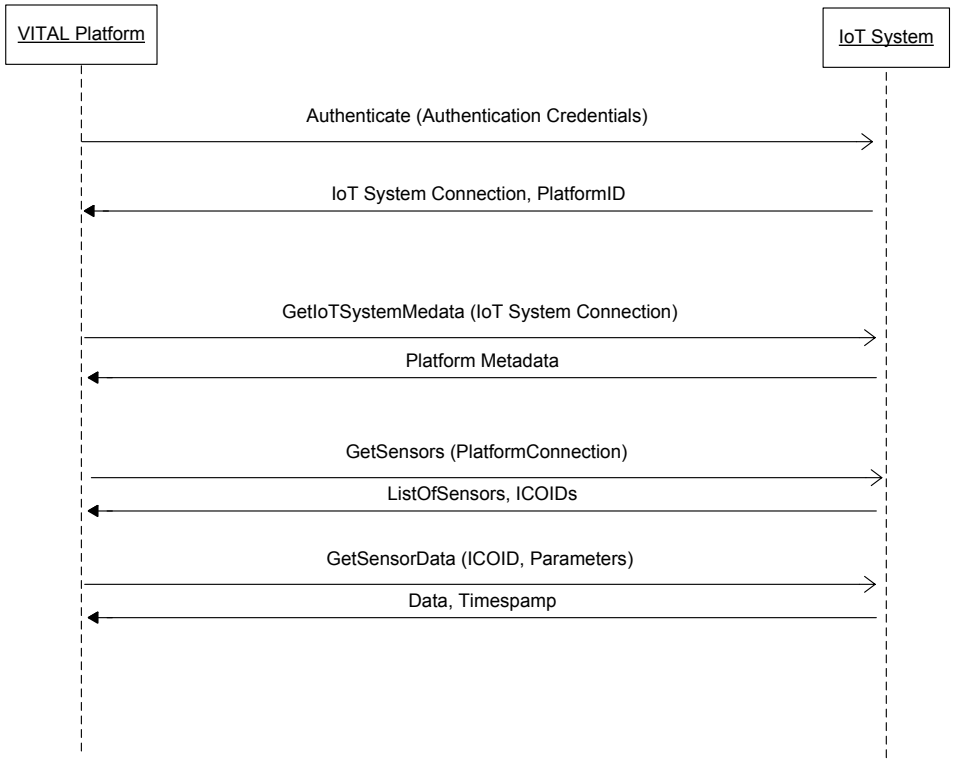

Figure 12: Indicative Interactions between the VITAL Platform and an IoT System (UML Sequence Diagram)

5.1.4 Implementation Technologies

From an implementation perspective the PPI will be realized as a RESTful interface. Each IoT platform/system provider will be therefore expected to implement the PPI and expose its functionality through a REST end-point. The implementation of the functionality of the PPI will be typically based on the technology adopted/used by the IoT system to be adapted (e.g., Java for X-GSN, MS. NET in the case of Hi REPLY). Note however that VITAL will also attempt to integrate systems/platforms without having access to their low-level implementation functionalities and details (e.g., Xively). The integration of these systems will be based on the implementation of loosely coupled middleware adapters, which will harness the APIs provided by the platform in order to access the platform and to expose the functionalities specified by the PPI.

Copyright © 2015 VITAL Consortium

5.2 VITAL IoT Platform Access and Data Acquisition

5.2.1 Functional overview

The VITAL IoT Platform Access module is a combination of software libraries, protocols and APIs to connect existing IoT platforms as well as new VITAL services to the VITAL system. Its main goal is to reduce the complexity of making an IoT platform compliant to the VITAL platform provider interface, e.g. by providing existing solutions to map proprietary sensor data to the VITAL ontology and to register an IoT platform as well as to send state updates from the platform to VITAL. In addition to this it aims at easing the development of VITAL services by enabling them to connect to and use other VITAL services, e.g. the VITAL discovery service. This is depicted in Figure 13. See also

Table 17 for an overview of the provided functionality.

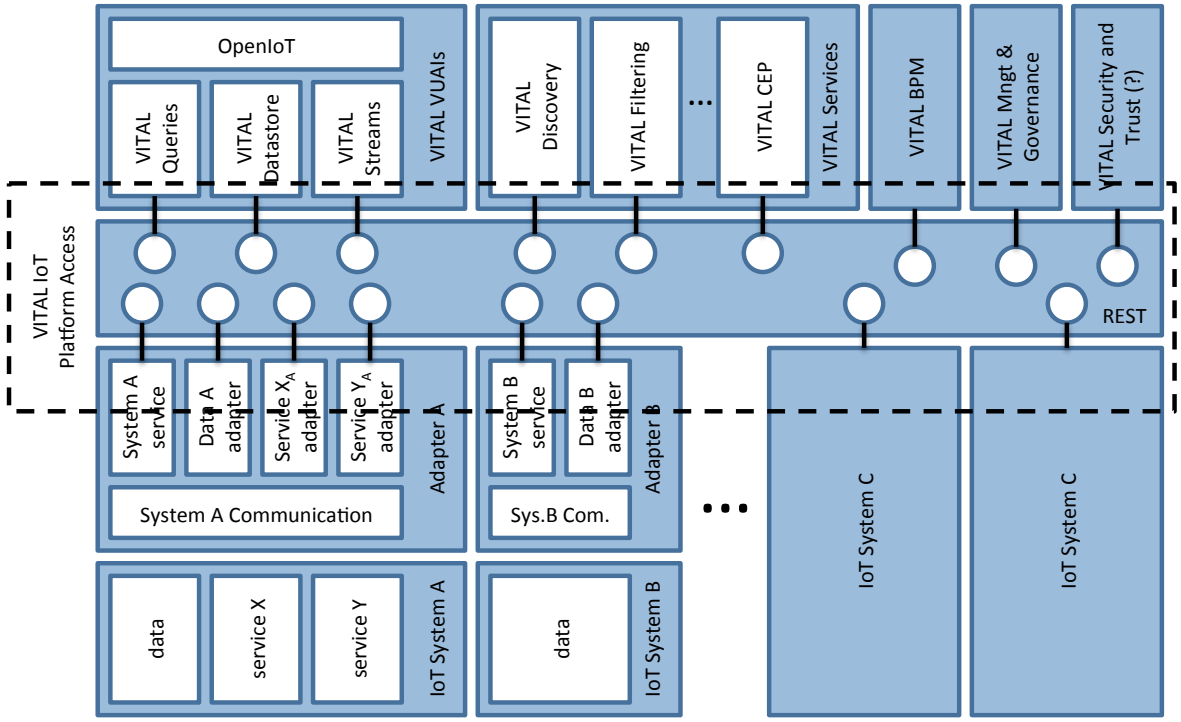

Figure 13: VITAL IoT Platform Access Overview

Table 17: Summary of Platform Access Functionalities

No.	Functionality
PA1	Mapping (raw) data formats to semantic formats and the VITAL ontology
PA2	Support for mapping existing system communication protocols for IoT platforms (e.g. CoAP, MQTT) to REST
PA3	Managing the lifecycle of an IoT system in VITAL (registering, deregistering, metadata export and updating)

Copyright © 2015 VITAL Consortium

PA4	Managing the lifecycle of a VITAL service in VITAL (registering, deregistering, metadata export and updating)

5.2.2 Implementation Technologies

The VITAL platform access subcomponents will be implemented in Java. Communication and access in the VITAL system will be based on RESTful services and a suitable REST framework that is still being selected. To communicate with sensors and actuators in different IoT platforms, different protocols can be used. VITAL will provide helper components to ease mapping important current IoT protocols (CoAP, MQTT) to standard RESTful services over HTTP based on the Eclipse Ponte platform.

5.2.3 Interfaces to other VITAL modules

The following table illustrates the interfaces of the VITAL platform access and data acquisition layer to other modules of the platform.

Table 18: Overview of VITAL Platform Access Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
IoT System (IoT platform, IoT application)	System, sensor, actuator and service data (streams), commands and metadata as specified in the PPI	• Pull & Push for sensor/ICO data retrieval • Push for metadata retrieval and services	Java, REST, Eclipse Ponte
VITAL IoT service (From the Added-Value Functionalities layer)	Service data, commands and metadata	Push & Pull	Java, REST
VITAL data management (from the platform agnostic data management layer)	Metadata and data specified in the VITAL ontology	Pull	REST

To showcase this, Figure 14 shows some example interactions between an integrated IoT system and the VITAL platform via the VITAL platform access component. The figure shows how an IoT system provides continuous updates of its current system state to the VITAL platform access component in a system dependent way using register and update messages. Alternatively the IoT system state could be updated by manually changing data in a web interface. This data would then be forwarded by the platform access to the VITAL data management. In both cases, the

Copyright © 2015 VITAL Consortium

platform access will use this information to implement the VITAL PPI functionality to provide metadata about an IoT system upon request by the VITAL platform. Secondly, the figure shows how the VITAL platform access receives raw data updates via different protocols, e.g. MQTT and CoAP and maps them to RDF data following the VITAL ontology that are provided to the VITAL platform via a RESTful interface. In short, these are two examples showing how the VITAL platform access reduces the complexity of integrating an IoT system into VITAL by providing out of the box functionality for implementing the PPI for systems using a number of well known protocols, data formats, etc.

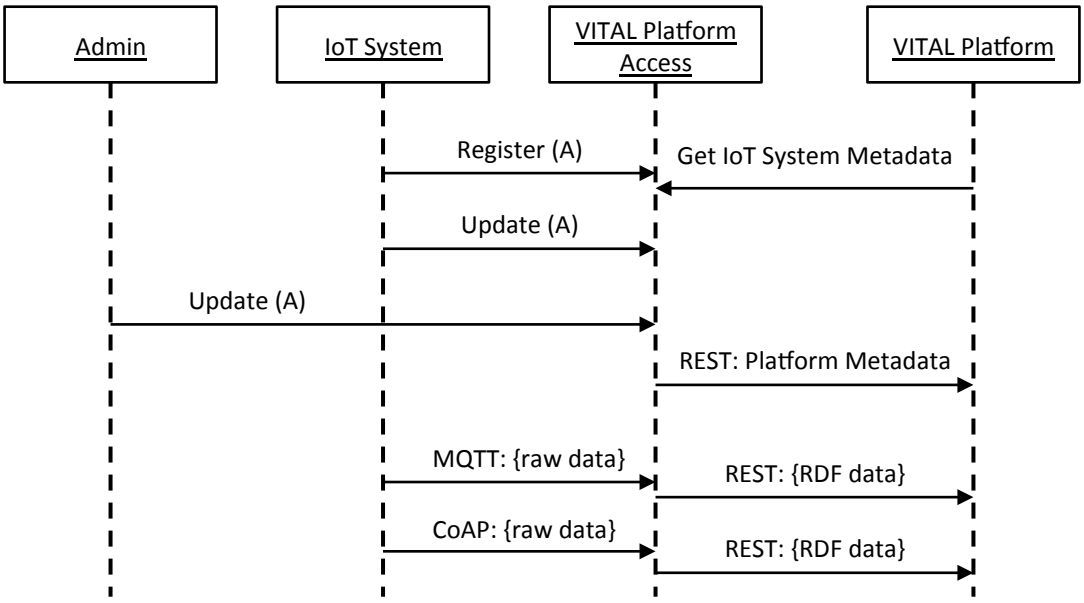

Figure 14: Example Interactions between an IoT System, the VITAL Platform Access and the VITAL Platform (UML Sequence Diagram)

5.3 VITAL Data Management Service

5.3.1 Functional overview

The VITAL data management service provides a central service to store and retrieve data and metadata for VITAL applications as well as VITAL system services. This includes the ability to store historical data, e.g. sensor values over time as well as support for continuous data streams. The data will be modeled using the VITAL ontology. Access will be provided with a suitable query language that allows easy and convenient querying and continuous queries. The language will be based on the SPARQL standard query language for semantic data. The following table reviews the main functionalities of the data management service:

Table 19: Summary of Data Management Services Functionalities

No.	Functionality
DM1	Storing data, including support for data streams

Copyright © 2015 VITAL Consortium

DM2	Retrieving data via a suitable query language based on SPARQL
DM3	(Continuous) query execution

5.3.2 Interfaces to other VITAL modules

The following table illustrates the interfaces of the VITAL data management services (provided within the platform agnostic data management layer).

Table 20: Overview of VITAL Data Management Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
Platform access and data acquisition layer	Data and metadata of sensors, ICOs, IoT systems and IoT services	Push/Pull Interfaces for Data management	REST
VITAL services and applications	Data and metadata of sensors and ICO; Data and Metadata of Services	Pull and Push	REST

The functionalities of the VITAL management services will be available to the modules of the VITAL platform agnostic data management layer (such as the filtering, CEP and discovery services) and will be based on the SPARQL language. SPARQL will enable the execution of queries over the VITAL ontology, notably queries for accessing data and metadata from sensors, ICOs and IoT services. The SPARQL queries will be triggered by the services of the platform agnostic data management layer, as for example shown in Figure 15.

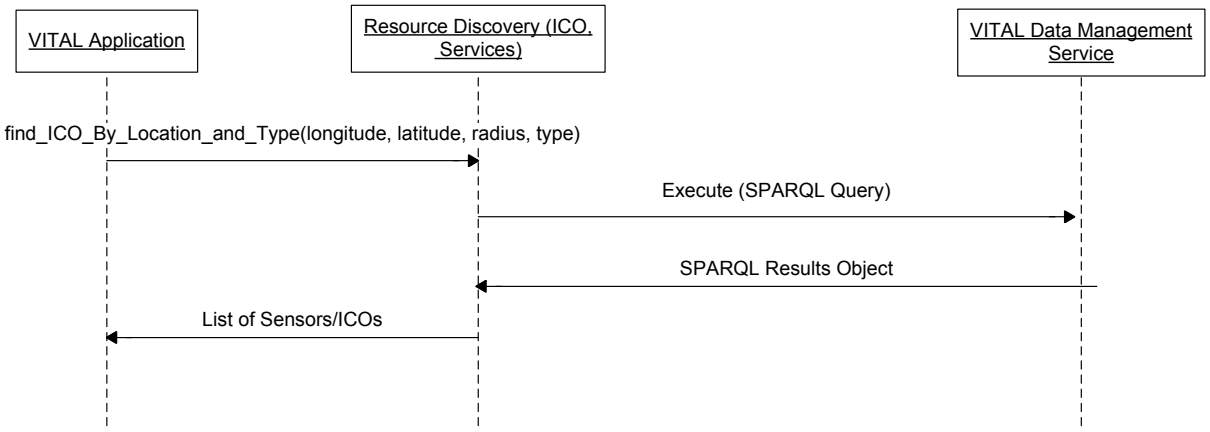

Figure 15: VITAL Data Management Services Interactions

Copyright © 2015 VITAL Consortium

VITAL Data Management Services will be invoked following relevant requests instigated by VITAL applications (i.e. modules/applications using the VITAL added-value functionalities layer)

5.3.3 Implementation Technologies

The VITAL data management will be implemented with the Java language and will be based on the open source OpenIoT platform implementation as well as NUIG's continuous query processor CQELS. This will allow VITAL to leverage OpenIoT's existing support for data management and scalable query execution using Cloud resources, the support for semantic data and data streams as well as CQEL's query language that extends SPARQL with support for temporal query constraints. Note however that the VITAL data management services will be provided over data models compliant to the VITAL ontology, which will be an extended version of the W3C SSN ontology.

5.4 Complex Event Processing (CEP)

5.4.1 Functional overview

- The Complex Event Processing is a component responsible for extraction of valuable information through a real-time analysis of the temporal and structural relations between the various information flows within VITAL. CEP is composed of three elements which are;CEP Engine; responsible for receiving/consuming data streams, run defined rules over data streams and publish detected events.
- Event Persistency; responsible for persisting detected events and make them available and queryable over REST services.
- Event Management; responsible for CRUD operations on event detection rule definitions.

Table 21: Summary of CEP Functionalities

No.	Functionality
CEP1	Pre-processing of the basic events
CEP2	Real time analysis of event streams originating from multiple clients
CEP3	Situational knowledge acquisition based on the streaming data
CEP4	Adaptable event detection based on feedback loop provided by VITAL
CEP5	Detected event persistency, configurable on service basis to make available historical data for post-processing purposes
CEP6	Rule management over RESTful interfaces

5.4.2 Implementation Technologies

The Complex Event Processor will be implemented with the C++ language leveraging the high performance it offers. For the development of the detected event persistency layer Java/JEE technologies will be used. All other communication and integration with other VITAL modules will be based on REST implementation in;

Copyright © 2015 VITAL Consortium

- Java/JEE technologies for the server side components of the module.
- Javascript and MVC frameworks for the GUI application of CEP rule management.

5.4.2.1 Key design Decisions

5.4.2.1.1 Rule-based inference engine

There is no convergence to the type of language or methodology used for specifying what complex events are, due to the fact that complex event processing is a recent discipline. Although the multitude of languages to perform this task, there are two main styles in which they can be grouped:

- Data stream query languages (SQL-like).
- Specification of rules using declarative language or Domain Specific Languages.

Rule specification languages are more convenient for implementing event queries than a general-purpose language, since it hides the complexity of these systems behind a more natural interface.

5.4.2.1.2 Knowledge inference as a service

One of the main goals of VITAL is to integrate different IoT platforms leveraging the vast amount of data, events and services they provide to develop new and more valuable services. CEP offers a dedicated REST API interface to inject new rules or modify existing detection rules for new services or based on the result of analysis performed on the data by other modules of VITAL.

5.4.2.1.3 Architectural Decomposition

Figure 16 shows an overview of all CEP subsystems. In addition, to achieve a better understanding of the overall functionality and configuration and communication possibilities of the Complex Event Processor, main functional components are described individually in Table 22.

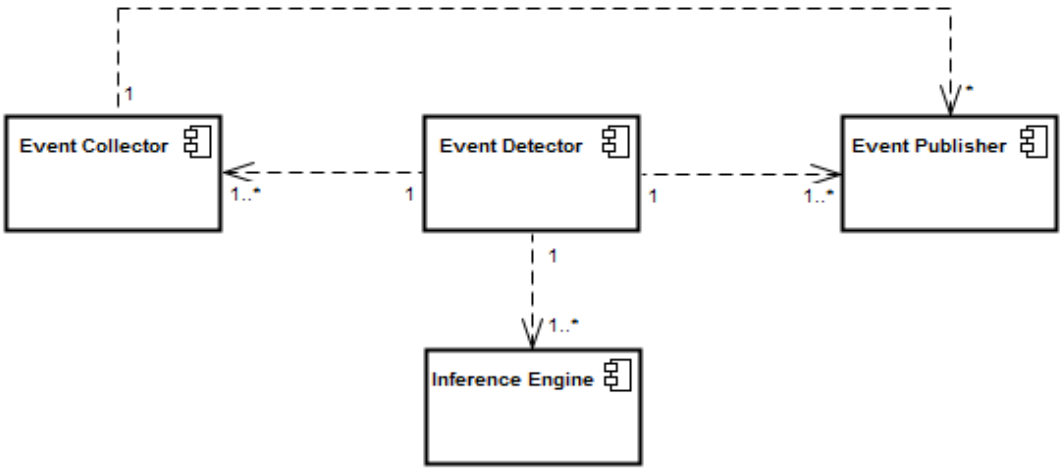

Figure 16: CEP Subsystems

The communication between Event Collector, Event Detector and Event Publisher components is performed using a Publish/Subscribe communication stack on top of TCP/IP network protocol. Therefore these components can be distributed on different physical processing nodes when appropriate. For communication between Event Detector and Inference Engine an inter-process communication mechanism is used.

Component Name	Description
Event Collector	Reads all information coming from different sources using different communication protocols and data formats. It also acts as de-multiplexer, receiving events from multiple sources and sends them in proper format to the next component.
Event Detector	Controls event detection and production of expected results by delegation of particular rules to inference engine(s). Event Detector is also responsible for maintaining internal knowledge base and ensuring safety/fallback against over aggressive conditions.
Inference Engine	Is responsible for detection of particular situation by using temporal persistence of volatile events until constraints of a rule(s) are entirely satisfied.
Event Publisher	Is in charge of delivering the detected information to the expected external listeners by providing support for various communication protocols and data formats.

Table 22: CEP Components

5.4.2.2 Implementation Detail and Planning for Reuse

SOL CEP, which is developed by Atos, will be reused as a basis for VITAL complex event processing engine. In order to be able to fulfill the smart cities requirements, SOL CEP will be extended with the following features;

- Sensor/Source separation; with this feature CEP will be able to run the same rule over a data stream separately for each sensor/source, defined by an id. So, the user will be able to run the same rule for all the sensors/sources in a data stream without a need to define a rule for each one.
- Rule management, the user will be able to apply CRUD operations on rule definitions over REST interfaces. These changes will be on the fly, meaning they will not require a restart, as the case in SOL CEP, since there will be no internal data loss
- Detected event persistency, the detected events and their payloads will be persisted on the database. These persisted events will be made available for querying over services.
- Event Collector and Event Publisher will be adapted to work with VITAL ontology.

Copyright © 2015 VITAL Consortium

- New Mathematical/Statistical functions; new mathematical & statistical functions, that will be needed through the implementation of use cases (for example: standard variation), will be implemented in CEP engine.

5.4.3 Interfaces to other VITAL modules

The following table provides an overview of the interfaces of the CEP modules to other modules of the VITAL platform:

Table 23: Overview of VITAL CEP Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
Discovery	«data» about new discovered ICOs, services, and their status change	Push	REST
VITAL IoT Platform Access	CEP will receive system, sensor, actuator and service data, and steam info (complex events) built from received data.	Push	REST
Workflow Manager - Orchestrator	definition for new or modified complex event detection rules	Push	REST

5.4.4 Relevance / Customization to VITAL Scenarios

As stated in above sections, CEP is responsible for real-time analysis of data streams with defined rules and extraction of valuable information. CEP can be used in different purposes with respect to the use case's needs.

For Istanbul integrated use case, CEP can be used as an "Event Detector". For instance, in a Traffic alarm management use case, CEP processes IMM's traffic data streams, detects the "sudden decrease in average speed" and publishes the detected event and its "payload" to the alert system end point.

For Camden integrated use case, CEP can be used as a "Data Aggregator". For instance, CEP can correlate GPS position of a person who has to attend to the meeting with the GPS position of the place, as a result we can have "the real time distance and travel time to meeting place". If the distance or travel time is greater

Copyright © 2015 VITAL Consortium

than a defined value, CEP can publish data to an end point that has the ability to take action for “late attendee”.

5.5 Information Filtering

5.5.1 Functional overview

This module interacts with the “Discovery Module” in order to filter data suitable for a certain context. The data discovered by the “Discover Module” will be indeed processed in function of specific parameters given by users (e.g. ICOs temperature > X, and so on).

Table 24: Summary of Information Filtering Functionalities

No.	Functionality
1	Filter data suitable for a context

5.5.2 Implementation Technologies

For the implementation will be used JAVA, SPARQL and REST technologies.

5.5.3 Interfaces to other VITAL modules

The following table provides an overview of the interfaces of the filtering module to other modules of the VITAL platform.

Table 25: Overview of VITAL Information Filtering Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
Discovery	Returns «data» suitable for a context	Pull / Push	JAVA EE REST API
VITAL IoT Platform Access	Returns «data» suitable for a context	Pull / Push	JAVA EE REST API
VITAL Data Management Service	Returns «data» suitable for a context	Pull / Push	JAVA EE REST API

Copyright © 2015 VITAL Consortium

5.6 Discoverer of ICOs, Data Streams and Services

5.6.1 Functional overview

This module uses the "*VITAL IoT Platform Access*" and "*VITAL Data Management Service*". This module returns ICOs in function "Location" and/or "Type". Moreover, the discovery process will be applied also for Services and Data Streaming.

Table 26: Summary of Discovery Functionalities

No.	Functionality
1	Get ICOs with a given location and/or a given type
3	Returns services with a given location and/or and a given type
4	Returns data streams with a given location and/or a given type

5.6.2 Implementation Technologies

The implementation will use JAVA, SPARQL and REST technologies.

5.6.3 Interfaces to other VITAL modules

The following table reviews the interfaces of the VITAL discoverer module to other modules.

Table 27: Overview of VITAL Discoverer's Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
VITAL IoT Platform Access	Get «data» in function of parameters	PULL / Push	JAVA EE, REST API
VITAL Data Management Service	Get «data» in function of parameters	PULL / Push	JAVA EE, REST API

5.7 Workflow Manager – Orchestrator

5.7.1 Functional Overview – General Description

The Workflow Manager – Orchestrator module of the VITAL architecture aims at achieving cross-platform & cross-business-context process integration. The high level goal is to provide an abstract digital layer over the application silos of the modern smart city. More specific goals include the provision of tools and techniques

Copyright © 2015 VITAL Consortium

for producing composite VITAL services comprising selection and combination of data sources/streams, as well as conditional execution of actuating services when specific conditions are met. In the sequel we provide the high-level functionalities of the VITAL Workflow Manager module, while following paragraphs illustrates how these functionalities will be materialized in a way that fulfills key requirements of the VITAL platform.

At a conceptual level the Workflow Manager module of the VITAL platform will:

- Provide the BPM capabilities of the VITAL framework. In the context of VITAL process management functionality will focus on the development and deployment of composite services based on the combination of simpler data access services. Services composition will be based on a set of templates that will address key VITAL functionalities and requirements.
- Orchestrate and generate / offer new services to the solution providers, on top of the VITAL platform. These services will be based on composite VITAL services and will allow solution providers to rapidly develop platform-independent cross-context applications over the VITAL platform.
- Take into account uncertainty / confidence levels of the underlying services & processes during the orchestration and BPM definition. To this end, VITAL components need to associate data streams and/or events with confidence probabilities.

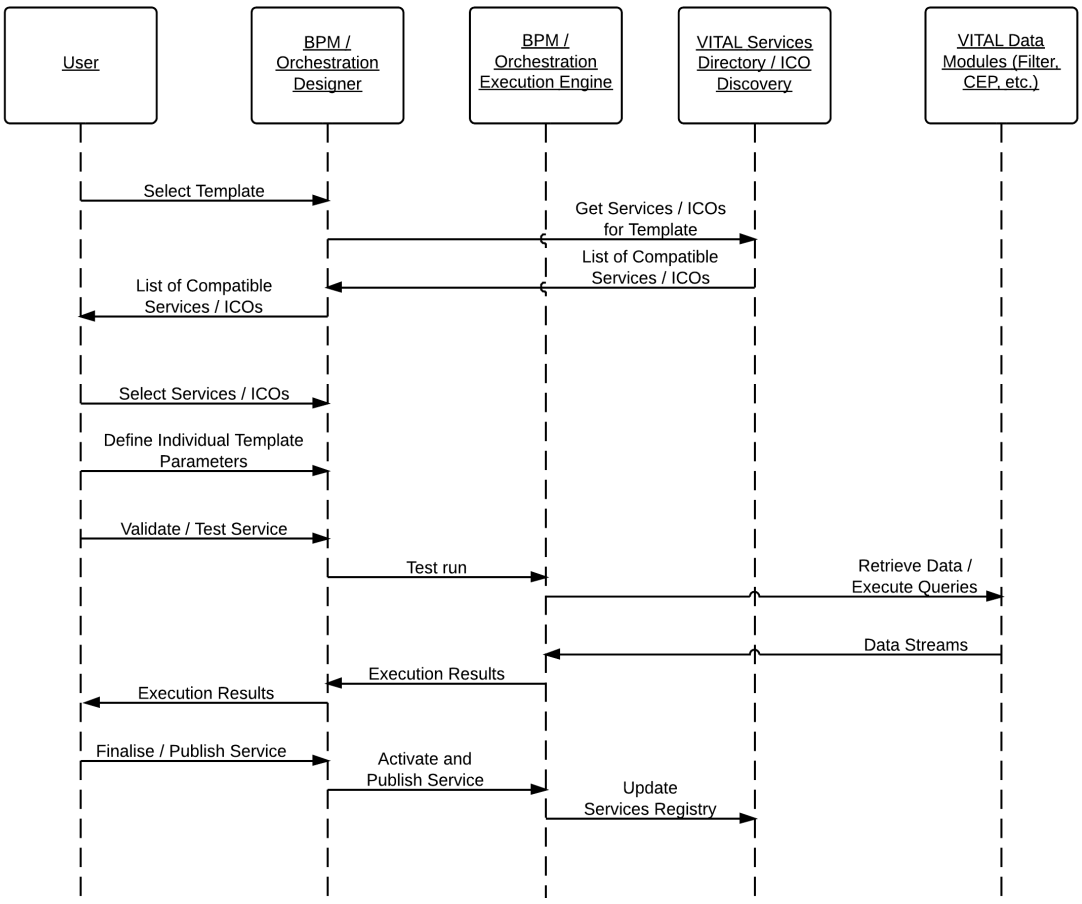

Figure 17: Orchestrator: Designing and publishing a service

Copyright © 2015 VITAL Consortium

The VITAL BPM/Orchestration module will support “smart city service templates”, i.e. predefined workflows that follow typical smart city scenarios for services that can be derived by the orchestration of multiple individual data streams and sensor services. These templates will allow the easy creation of families of services through parameterization and by selecting sensor data streams and services from the homogenized pool of the VITAL platform.

Figure 17 is a sequence diagram that provides an overview of the interaction of the user with the BPM/Orchestration Designer. The user selects one of the services templates and defines specific parameters (conditions and operations for business process definition, filters and other processing info). All the available options are defined by the selected template options but the Designer will also have to contact the VITAL Services Directory for providing available ICOs and Services for use in this definition process. Note that the information provided by the directory includes a fully homogenized set of services offered by the heterogeneous underlying systems as well as the VITAL platform itself (e.g. Filtering, CEP, etc).

The user may request validation & testing of the newly designed service and the Designer will use the BPM/Orchestration Engine to test/execute (without deploying) the service, in which case the appropriate VITAL modules will be used for accessing the appropriate services / data streams. Note that this is a sand-boxed version of the actual execution flow, if and when the service is published and called by an external application using the VITAL platform.

Eventually the user will publish the service via the Designer, which will result in publishing / deploying the service at the BPM/Orchestration Engine and consequently registering the service at the Services Registry, thus making discoverable and available to all connected applications.

As explained earlier the diagram depicts the definition and publishing phase. The flow for updating an already published service is similar, and the execution of a service is almost identical (in interaction terms) to the validation / test run flow presented above. The Designer will support the de-activation / activation of these services by a user with the appropriate permissions. Concerning permissions, the BPM/Orchestration module will rely on the VITAL Security Layer.

5.7.2 VITAL Templates and Workflows

Instead of offering a general-purpose BPM engine, the VITAL Workflow Manager module will focus on supporting smart cities specific templates for composing services i.e. templates tailored to the nature and goals of the VITAL platform. In particular, the VITAL templates will support the combination of the following functionalities:

- Data, data sources and data streams selection on the basis of parameters such as:
    - Location of data source or stream (e.g., data sources located to a certain urban area).
    - Type of data source (e.g., type of sensor).
    - Accuracy and confidence of the data source.
    - Condition on the data streams (e.g., thresholds on the values of particular data attributes).
    - Time Window i.e. start time and end time of the data stream.

Copyright © 2015 VITAL Consortium

- Filtering, aggregation and combination of more than one data sources/streams into a composite data source or stream on the basis of:
    - Encapsulation of the data of the streams in a single stream.
    - Creation of data streams based on the conditional selection of one or more data streams (e.g., based on thresholds on the stream values).
- Execution of actuating functionalities both within and outside the VITAL platform e.g., supporting the configuration of VITAL platforms, tuning the frequency of data provision (from the VUAIs) and invoking functions that send e-mails or SMS messages.
- Each VITAL template will provide a parametric RESTful service with the following characteristics: It will result in a RESTful VITAL service deployable within the VITAL platform. The resulting service will be offered to solution provider/developers with a view to easing and accelerating their application development tasks.
- It will support composition of other VITAL RESTful services including:
    - VITAL platform data and services accessible via VUAIs and/or PPI.
    - VITAL service discovery functionalities (provided by the ICO discoverer module).
    - VITAL CEP functionalities (also accessible via VUAIs).
- It will support storage and caching of data associated with composite data streams, thereby enabling solution providers to have fast high-performance access to the data of the composite services.
- It will be configurable in terms of the functionalities listed above i.e.
    - Parameters/criteria for selecting data sources/streams.
    - Parameters/criteria for creating composite data streams.
    - Parameters/criteria for invoking actuating functions upon the fulfillment of certain conditions.
- It will be able to manage invocations of the template, while also deploying and caching recently used templates

The table below summarizes the functionalities of the module.

Table 28: Summary of Workflow Management Functionalities

No.	**Functionality**
1	Combine services & processes in a BPM fashion
2	Orchestrate generated services and produce new VITAL services offered to the solution provider, for consumption in applications build on top on VITAL
3	Provide a User Interface for supporting the above functions
4	Support selection of data sources and data streams
5	Support the production of composite data streams encapsulating one or more elementary streams
6	Support execution of VITAL services and third-party actuating services upon the fulfillment of specific conditions
7	Support services composition on the basis of parametric configurable

Copyright © 2015 VITAL Consortium

	templates for smart city processes

5.7.3 Templates and Workflow Examples

The following tables provide a set of indicative workflows (templates) that will be used by the Workflow module.

Table 29: Workflow Template 1- GetDataStreamsByLocationAndType()

Template 1:

```
GetDataStreamsByLocationAndType (
    Location location,
    SensorType types[],
    TimeStamp startTime,
    TimeStamp endTime,
    int refreshFrequency
);
```

where:

- Location location: denotes an urban area (specified on the basis of longitude, latitude and radius)
- SensorType types[]: denotes the list of sensor types for which data should be fetched and combined in the same LSON-LD document.
- TimeStamp startTime: denotes the lower bound of the time window for which data will be fetched.
- TimeStamp endTime: denotes the upper bound of the time window for which data will be fetched.
- int refreshFrequency: denotes the frequency (in milli-seconds) of the data update/refresh.

The invocation of this workflow will ultimately return a JSON-LD document containing the selected data. During the execution, the workflow manager will include a local cache of measurements, where JSON-LD documents will be stored at the given frequency.

Table 30: Workflow Template 2- InvokeService()

Template 2:

```
InvokeService (
    Location location,
    SensorType type,
    String property,
    double threshold,
    String operation,
    Service service,
```

Copyright © 2015 VITAL Consortium

boolean poll);

where:

- Location location: denotes an urban area (specified on the basis of longitude, latitude and radius)
- SensorType type: denotes a type from a list of available sensor types in the VITAL ontology
- String property: specifies a (numeric) property of the sensor's output.
- double threshold: is a specified threshold for
- String operator: is an operator (i.e. >, >=, <, ,=, =, != used to compare the specified sensor property against the threshold.
- Service service: is a service to be invoked in case the condition is met.
- boolean poll: specifies whether the service should poll for the fulfillment of the condition (in which case poll=true) or run the threshold test only once.

The execution of this workflow will result on a loop that will poll for measurements and if the condition is met it will call the specified operation. The workflow engine will be able to monitor the process, and control it's execution cycle (start, stop).

Table 31: Workflow Template 3-GetDataStreamsByLocationTypeAndAccuracy ()

Template 3:

```
GetDataStreamsByLocationTypeAndAccuracy (
    Location location,
    SensorType types[],
    TimeStamp startTime,
    TimeStamp endTime,
    int refreshFrequency,
    int accuracy
);
```

All parameters are similar to Template 1, but only streams with higher accuracy than the specified one qualify for the result. The result includes the accuracy of the individual selected streams and computes a weighted average of the composite stream as well.

Table 32: Workflow Template 4- GetDataStreamsByLocationTypeAndAccuracy()

Template 4:

```
GetDataStreamsByLocationTypeAndAccuracy (
    Location location,
    SensorType types[],
    TimeStamp startTime,
    TimeStamp endTime,
```

Copyright © 2015 VITAL Consortium

```
    int refreshFrequency,
    int accuracy,
    double frequency[]
);
Similar to the above, but sampling streams according to the sampling
frequencies contained in table double frequency[] and producing the
resulting JSON-LD.
```

5.7.4 Implementation Technologies

Implementation will be in Java / JEE for the server side components of the module. We will try to re-use whatever off-the-self FOSS components and libraries are appropriate, especially in implementing:

- The BPM capabilities.
- The orchestration of services.

Orchestration of SOAP services is supported by a plethora of FOSS products and specifications (e.g. BPEL), however in VITAL we rely on RESTful resources. We need to do extensive research on available workflow engines, focusing on their level of extensibility/customization and experiment with potential implementations.

A list of available platforms that are evaluated contain the following:

- Activiti BPM Platform. Activiti is a light-weight workflow and Business Process Management (BPM) Platform targeted at business people, developers and system admins. Its core is a super-fast and rock-solid BPMN 2 process engine for Java. It's open-source and distributed under the Apache license. Activiti runs in any Java application, on a server, on a cluster or in the cloud. It integrates perfectly with Spring, it is extremely lightweight and based on simple concepts. (source: http://activiti.org)
- jBPM. jBPM is a flexible Business Process Management (BPM) Suite. It makes the bridge between business analysts and developers. Traditional BPM engines have a focus that is limited to non-technical people only. jBPM has a dual focus: it offers process management features in a way that both business users and developers like it. (source: http://www.jbpm.org)
- Copper Engine. Copper is an open-source, powerful, light-weight, and easily configurable workflow engine. The power of COPPER is that it usesJava as a description language for workflows (source: http://copper-engine.org).
- Scripting language inside the JVM. In this case the workflows are defined as scripts executed dynamically in the JVM with different parameters. Possible candidates for the definition of workflow scripts are Groovy, JRuby, Jython). This is the most flexible solution but most demanding in development time.

Copyright © 2015 VITAL Consortium

•

Figure 18 below sketches a high level architectural diagram of the BPM / Orchestrator.

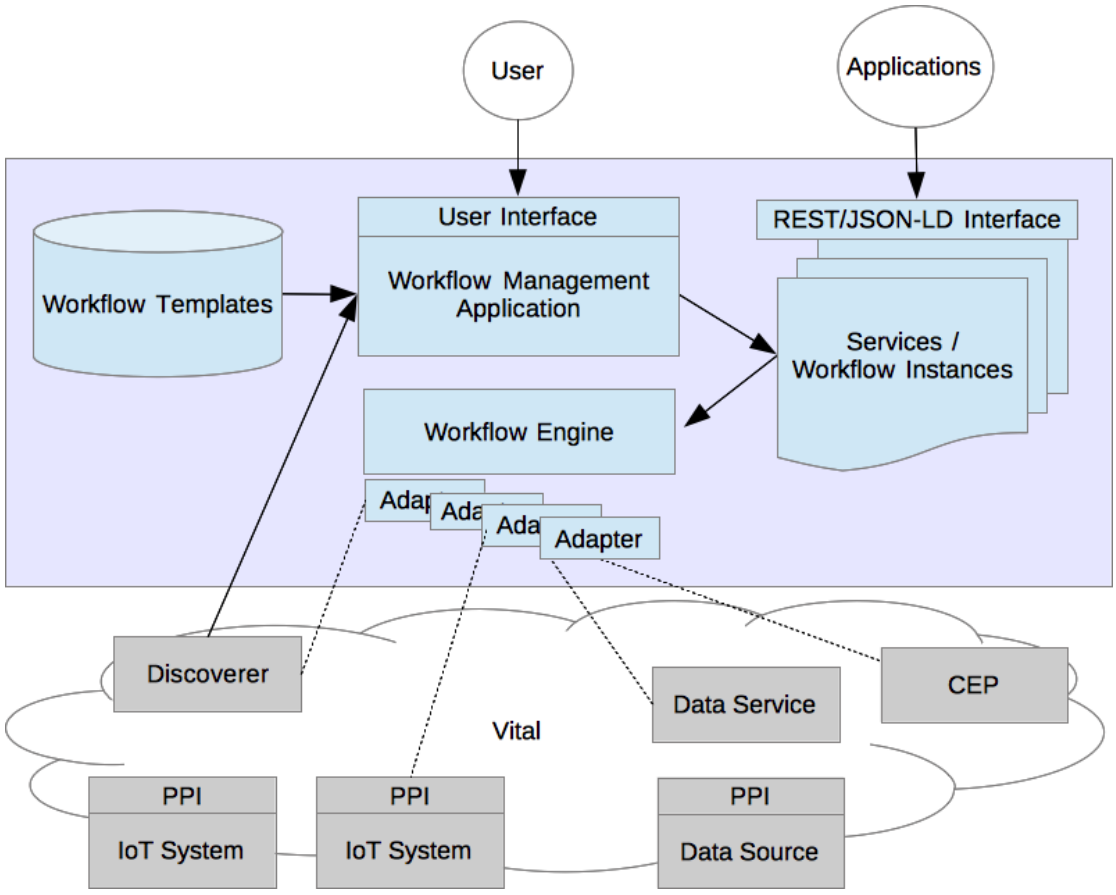

Figure 18: High level BPM/Orchestration module architecture

5.7.5 Interfaces to other VITAL modules

Table 35 provides an overview of the envisaged interfaces of the BPM/Orchestration module to other modules of the VITAL platform.

Table 33: Overview of VITAL Workflow Manager's Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
VITAL IoT Platform Access	ICO Data, ICO Data Management Services (exception)	PULL	REST
VITAL Data Management	ICO Data, ICO Data Management	PULL	REST

Copyright © 2015 VITAL Consortium

Service	Services		
Complex Event Processing	Events, ICOs data and metadata	PULL	REST
ICO Discovery and Data stream & services discovery	Topology and services description metadata	PULL	REST
Filtering	Events, ICOs data and metadata	PULL	REST
Development Tools	RESTFul Functionalities; Metadata of the Configurable Parameters	PULL/PUSH	REST

5.8 Management and Governance Tools

5.8.1 Functional overview

The Management services will implement functionality from the FCAPS model/framework (introduced by ISO) that applies to the VITAL platform use cases. It implements a management layer that enables the efficient monitoring and management of the VITAL platform. This is required for supporting uninterrupted and efficient operation of the services and applications built on top of VITAL.

The IT Governance services provide a layer for parameterizing the VITAL platform on every specific deployment. The Governance tools support dashboard-like graphical user interfaces (web-based) that allow the editing of various parameters and configurable options that affect various operational parameters. For example this GUI will allow the definition of the sensor types supported in the specific deployment, configuration of the adapters for the connected IoT Systems, and any other operational parameter of the VITAL platform.

5.8.2 Implementation Technologies

All communication and integration with other modules of the VITAL ecosystem ideally should be based on REST. In addition, implementation will be based on:

- Java/JEE technologies for the server side components of the module.
- Javascript and MVC frameworks for the GUI applications of the management and the governance layers, which require graphical UIs for visualizing time sensitive information and time series.

Table 34: Summary of Management and Governance Tools Functionalities

Copyright © 2015 VITAL Consortium

No.	Functionality
MGMT1	Performance and fault monitoring. This includes performance and health KPIs as well as information and alerts on fault and high utilization conditions
MGMT2	Accounting. The accounting functionality enables detailed monitoring of the usage of the platform by the various solution providers on top of VITAL. It will ensure fairness between hosted applications/solutions and provide a mechanism for safeguarding the quality of service delivered to the end-user. Note that accounting is a pre-requisite for supporting usage-based charging models that would allow a wider range of high quality connected data streams that may follow such models.
MGMT3	Security monitoring. It will support the means for monitoring security events and generate security alerts.
GOVR1	Dashboard GUI for parameterizing and configuring the VITAL platform deployment.

5.8.3 Interfaces to other VITAL modules

The module will have RESTful interfaces to all other VITAL modules, as shown in Table 35. These will be used for monitoring and management as well as accounting.

Table 35: Overview of VITAL Management and Governance Tools Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
All VITAL modules	Monitoring data	Pull & Push	REST

5.9 Development Tools

5.9.1 Functional overview

The VITAL development tools will primarily enable the visual definition of information acquisition queries over the smart city infrastructures. In order to enable the definition of such queries, the tools will provide the means for discovering ICOs and filtering information based on the above-listed modules of the VITAL platform. The design of the development tools of the project will be based on the respective visual tools of the OpenIoT platform, which will be appropriately extended in order to meet the VITAL requirements. Note that these OpenIoT visual development tools (part of the

Copyright © 2015 VITAL Consortium

OpenIoT IDE) include modules, which provide the functionality for visual definition and representation of data feeds/sensor streams, in a way that is usable by non-advanced users. They also include discovery of deployed data streams and implementation of applications or services based on these data streams. In the context of the VITAL framework, these tools will be enhanced with concepts and services that are commonly associated with Smart-City applications (such as applications relating to smart buildings, smart energy, smart transport and more).

Table 36: Summary of Development Tools Functionalities

No.	Functionality
1	The definition of existing data feeds (host, connection protocol, data model) that represent Smart-City services within a city (e.g. traffic congestion, citizen reports, resource utilization etc.). This functionality should be available in a way that is friendly to non-experts.
2	REST API that exposes currently streamed data and manages the functionality of the defined data feeds (starting/stopping streaming of data, setting data streaming intervals) and interfaces the feed data model with the Vital Ontology.
3	Service that discovers available data feeds and presents them through a WEB GUI visually on a map.
4	A WEB GUI that supports visual creation of services that utilize the available data feeds. For example (current traffic congestion in a specific area, average population density per hour in specific area, consumption of energy resources in a temporal range).
5	REST API that exposes data from the created applications. This API will be used as a basis for the development of custom WEB and Mobile applications that utilize the developed Smart-City data-feed services.

5.9.2 Implementation Technologies

The technologies that will be used for the implementation of the Development Tools are the following:

- **Java Standard Edition 8,** which is the most popular programming language currently used. Version 8 which was recently announced introduced revolutionary additions to the language that provided new possibilities for programmers, namely: Lambda support (Closures); Streaming API for manipulation of collections; New Date/Time API; Inclusion of JavaFX in the Standard Java SE library.
- **Java Enterprise Edition 7** is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment, for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Standard Edition platform, providing an API for object – relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely

Copyright © 2015 VITAL Consortium

on modular components running on an application server. Additionally, the platform emphasizes convention over configuration and annotations for configuration. Optionally XML can be used to override annotations or to deviate from the platform defaults.

- **PrimeFaces** is a JSF Component library that features a rich set of 117 components (core components + variants). This suite uses behind the scenes, jQuery with its widgets, plugins, themes and Ajax interactions. This suite has exhibited significantly increased adoption by the community, widely surpassing other once popular frameworks such as RichFaces and IceFaces.
- **PrimeFaces Extensions** is a community driven open source project which has an aim to be a lightweight and fast JSF 2 component library in additional to PrimeFaces. This project is an extended component set with useful components missing in other JSF 2 libraries or with improved components which already exist somewhere but don't work there smoothly.
- **OmniFaces** is a utility library for JSF 2 that focuses on utilities that ease everyday tasks with the standard JSF API. The library does not contain much if any of the visually oriented components that those other libraries have already. Rather, it is more geared toward "utilities" that solve every day practical problems and workarounds for small shortcomings in the JSF API.
- **Wildfly Application Server** is an application server fully compliant to the Java EE 7 stack and fully available as open source. It is a direct continuation of the JBoss Application Server project and is developed by Red Hat.

5.9.3 Interfaces to other VITAL modules

The following table provides an overview of the interfaces of the development tools to other modules of the VITAL platform.

Table 37: Overview of VITAL Development Tools Interfaces to Other Modules

VITAL Module	Type of Information Exchange	Type of Interface (Pull/Push)	Implementation Technology
VITAL IoT Platform Access	JSON/XML	PULL	Java EE7 Restful API
Complex Event Processing	JSON/XML	PULL	Java EE7 Restful API
VITAL Data Management Service	JSON/XML	PULL	Java EE7 Restful API
Filtering	JSON/XML	PULL	Java EE7 Restful API

Copyright © 2015 VITAL Consortium

5.10 Virtualized Unified Access Interfaces

5.10.1 Overview and Role in the VITAL Architecture

According to the layering of the VITAL architecture, its added-value functionalities (i.e. service discovery, filtering, CEP) will be platform agnostic and accessible via virtualized interfaces. The VITAL architecture prescribes a set of Virtualized Universal Access Interfaces (VUAIs), which will be used to access the above-listed added-value functionalities over data and IoT services stemming from any of the underlying IoT systems. The detailed specification, design and implementation of the VUAIs is part of WP3 of the project.

5.10.2 Implementation Technologies

VUAIs will be implemented as loosely coupled RESTful interfaces over VITAL modules, notably modules of the added-value functionalities layer and of the platform agnostic data management layer of the VITAL platform.

6 VITAL ARCHITECTURE SUPPORT FOR PROOF-OF-CONCEPT INTEGRATED SCENARIOS

In the current section, we wish to verify that the architecture proposed above is in line with the integrated scenarios of Camden and Istanbul as they were presented in earlier deliverable D2.2. This will validate the appropriateness of the VITAL platform to support these scenarios, as well as the relevant use cases analyzed in D2.2. We should note here that the validation use cases presented in D2.2 were specifically chosen to accurately address specific individual functionalities and features of the VITAL platform, and hence their matching with most of the modules of the VITAL architecture is straightforward. On the other hand, the two integrated scenarios being inherently complicated and diverse, present a multitude of interactions and overlaps with the architecture modules and building blocks as these scenarios are designed and deployed. We will attempt to cross-tabulate overlaps in order to make sure that as many of the modules as possible are being involved in each scenario, (ideally all of them will be properly addressed within the two integrated scenarios) and hence we will be able to deem the architecture proposed as suitable to conform to all the needs of the scenarios. It is also reasonable to assume that if such complicated city scenarios can be supported properly by the current architecture, then several other similar smart city applications will also be properly addressed without any major modifications in architecture modules.

6.1 Camden Mobile Working

6.1.1 Synopsis of the scenario

Based on the description of the Camden Mobile Working scenario described in the revised (V2.0) of VITAL deliverable D2.1., we present a short synopsis of the integrated scenario.

Copyright © 2015 VITAL Consortium

Table 38: Synopsis of the Integrated Scenario at Camden

Primary Sector of Application based on Questionnaires	**Mobile Working**
Secondary Sectors of Application based on Questionnaires	• Real Estate
Basic Data Feeds	• Footfall data feed (provided by Spring-Board) • Weather data feeds (provided by Met Office) • Air quality data feed (provided by Air Quality) • Transport-related data feeds (provided by Transport for London)
Basic Targets / Needs Addressed	• Spatio-temporal data analysis (e.g. for footfall data) • Data estimation (e.g. for footfall data)
Functions / Applications	• Layered data filtering • Data estimation based on historical values • Data correlation • Data visualisation
Added Value	Data provision for location comparison
Intended End-User	Mobile workers seeking a suitable short-term working environment
Which data feeds present in the city will be used in this use case	• Footfall data • Estimated temperature • Estimated humidity • Air quality • Bus stop locations and timetables
Trigger	User Driven

6.1.2 Indicative application

Mobile workers require suitable, short-term working environments, whereas at the same time office owners may have free space and desks in their offices that they can rent. The application we intend to build enables both office owners to publish live data about their available working spots and mobile workers to select among the available working spots the one that better suits their needs. Information published about available working spaces might include location, price and amenities (e.g. printers).

A mobile worker can use the application and specify the time period, for which he/she wants to find a working location, as well as any amenities that the working location should have (e.g. printer, scanner, Wi-Fi). The application searches for any available working spots in Camden that meet the specified criteria and shows them on a map.

Copyright © 2015 VITAL Consortium

Each working spot is accompanied with information about the estimated air quality, temperature, humidity and footfall in the area, as well as with information about how he/she can move to and from that working spot. The user can go through all this information in order to ultimately select among the different working spots the one that he/she prefers.

6.1.3 Modules and Building Blocks Involved

With reference to the architecture schema we derive the following table to illustrate which modules and building blocks of the proposed architecture are collaborating during this indicative instantiation of the Camden integrated scenario.

With reference to the architecture schema we derive the following table to illustrate which modules and building blocks of the proposed architecture are collaborating during this indicative instantiation of the Camden integrated scenario.

Table 39: Mapping of the Integrated Scenario at Camden to the VITAL Architecture Modules

Architecture Layer	**Architecture Module**	**Involved**	**Short Explanation**
Smart City Applications and Tools	Governance Tools	☑	Dashboard-like graphical user inter-faces (web-based) for city authorities
	Management Tools	☑	Performance and fault monitoring as well as security alerts are typically needed
	Development Tools	☑	Mobile application for the general public where all available working spots are enhanced with weather, footfall and air quality information
Virtualized Universal Access Interfaces		☑	Platform-agnostic way to access services from the added-value functionalities layer, but also data available at the VITAL data management layer
Added Value Functionalities	CEP (Complex Event Processing)	☑	Temperature and humidity data combination in order to infer more information about the quality of a working location
	Filtering	☑	Spatial (geo-tile-based) filtering for all data fields Temporal filtering for transport data feeds

Copyright © 2015 VITAL Consortium

	Orchestration	☑	Combination of services and processes is required in a typical smart city scenario
Platform Agnostic Data Management	ICO Discoverer	☑	Data acquisition only from geographically related data feeds (i.e. ICOs)
	Service Discovery	☑	Data acquisition from relevant services
Platform Access and Data Acquisition Layer		☑	X-GSN, Xively and the footfall system access in order to acquire all necessary data
IoT Platforms	X-GSN	☑	Transport and air-quality data feed integration into VITAL through X-GSN
	Xively	☑	Weather data feed integration into VITAL through Xively

6.2 Traffic Management at Istanbul - Integrated Scenario SC02

6.2.1 Synopsis of the scenario

Based on the analysis of the previous deliverable D2.2, section 6, we present a short synopsis of the integrated scenario envisaged for implementation/integration at Istanbul.

Table 40: Synopsis of the Integrated Scenario at Istanbul

Primary Sector of Application based on Questionnaires	**Traffic and Transport management**
Secondary Sectors of Application based on Questionnaires	Smart Security & Disaster Prevention
Basic Datafeeds	Basic Live: • 500+ traffic cameras • 500+ road sensors; • 250 Bluetooth sensors; • 35 weather sensors. Traffic-related Systems: • Variable message boards • Remote Traffic Microwave Sensor (RTMS) • Bluetooth Sensor • TERRA • TERRA Tunnel

Copyright © 2015 VITAL Consortium

	• Smart Sensor • Wireless Magnetic Sensors • Loop Detectors • Traffic Cameras • Electronic Violation Detection System (EVDS) • Variable Message Signs (VMS) • Lane Control System (LCS) • Online Signalized Intersections • Weather Observation Sensors
Basic Targets / Needs Adressed	Analysis of spatio-temporal variations of traffic Meaningful Analysis of massive data and complex events concerning traffic Traffic estimation & re-routing Urban-planning queries
Functions / Applications	Layered Filtering – Transport Management Queries Layered Filtering of Announcement Data Fusion of Fleet Management Sensor Data Weather and traffic data correlation Data Mining, and Analysis of Massive sensor datafeeds
Added Value	Analysis and prediction results to end-users which could be used for scientific research. Provision of real-time travel times on road segments (even in a mobile app)
Intended End-User	Local citizens (may plan their travels based on predictions/suggestions of VITAL platform) IMM traffic department (with statistics obtained some conclusions can be reached and congestion reducing precautions can be taken)
Which datafeeds present in the city will be used in this use case	Provided by IMM Traffic data obtained from radar/image processing sensors, fleet management systems. Travel time data obtained from Bluetooth sensors, weather data obtained from weather observation sensors and announcements data
Trigger	User Driven (Authority request for analysis of a specific large set of data in a given spatiotemporal window) to extract meaningful conclusions.

Copyright © 2015 VITAL Consortium

6.2.2 Indicative application

Given the identification of specific regions (geo-tiles) that present the largest bursts of traffic per time-window, (e.g., during the last X months or during massive events), as well as by drawing comparisons before/after a certain urban change was introduced, the city authorities decide (user-driven) to request an analysis of a specific large set of data in a given spatiotemporal window for the specific geo-tiles and their neighbouring geo-tiles only. In order to extract meaningful conclusions, extensive complex event processing and massive layered filtering of the data is employed. Fusion of sensor data, combination of announcement data, segmentation per geo-tile and even combination with weather info for the specific temporal window of interest are also performed. Finally this dispersed, large dataset that was initially extremely difficult to handle and for which real-time calculations were very intensive, is reduced through Data Mining, and Analysis of the datafeeds into a more manageable subset which is then used for statistical conclusions. It also forms a “baseline” of comparison for the future and urban planning decisions. The same baseline is used for comparison versus live data every day (what is happening now). A subset of these results is presented to the public and the whole picture of the analysis is given to the IMM servers for more “operational” queries and further analysis.

6.2.3 Modules and Building Blocks Involved

With reference to the architecture schema we derive the following table that illustrates which constituent parts and modules of the proposed architecture are collaborating during this indicative instantiation of the Istanbul integrated scenario.

Table 41: Mapping of the Integrated Scenario at Istanbul to the VITAL Architecture Modules

Architecture Layer	Architecture Module	Involved	Short Explanation
Smart City Applications and Tools	Governance Tools	☑	Dashboard-like graphical user interfaces (web-based) for IMM and other interested authorities
	Management Tools	☑	Performance and fault monitoring as well as security alerts are typically needed for such “big data” handling
	Development Tools	☑	Presentation of data in a Dashboard at IMM headquarters. An enriched version of the current mobile app can be launched for the general public.
Virtualized Universal Access Interfaces		☑	Provide an IoT system/platform agnostic way to access services from the added-value functionalities layer, but also traffic management data available at the VITAL data management layer.

Copyright © 2015 VITAL Consortium

Added Value Functionalities	CEP (Complex Event Processing)	☑	Analysis of streams originating from multiple clients (not necessarily real time). Perform processing of data streams from multiple sources to identify patterns and/or infer events. Assist ICO Discoverer module in order to select appropriate data sources
	Filtering	☑	Filter Specific Geo-Tile and routes (streets) within the tile and its neighboring ones. Filter relevant Transport Datafeeds only Filter according to specific authority needs
	Orchestration	☑	This is a typical smart city scenario that requires to combine services & processes
Platform Agnostic Data Management	ICO Discoverer	☑	Request specific sensors in specific geo-tile only. Agnostic about diverse systems present in Istanbul. Select relevant only according to CEP instructions.
	Service Discovery	☑	Request data from Transport Data Services
Platform Access and Data Acquisition Layer		☑	X-GSN and/or Xively are accessed in order to provide data stemming from the ICOs and sensors at Istanbul.
IOT Platforms	Present systems in Istanbul (available via X-GSN, Xively)	☑	REST interfaces can be used to access data from systems within Istanbul. These data can be then made available to VITAL via the X-GSN and/or Xyvely platform.

Copyright © 2015 VITAL Consortium

7 CONCLUSIONS

This deliverable has introduced the VITAL architecture as a set of functional modules and building blocks and the main structuring principles and interfaces between them. The introduction of the VITAL architecture has taken into account requirements, scenarios and use cases, which have been identified in earlier deliverables of WP2 of the project, namely deliverables D2.1 and D2.2. Furthermore, it has been based on a range of important principles that are driving the development of the VITAL project, such as the principles of maximizing reuse and adhering to standards, but also the principles of modularity and virtualization. The VITAL architecture has been also influenced by the ARM of the IoT Forum. VITAL adopts several ideas, modules and principles of the ARM, while also being interested in contributing to the IoT Forum ideas about the smart cities flavour of the architecture. Such contributions are part of the IERC cluster activities of the project in WP7.

The VITAL architecture specifies modules and interfaces for accessing IoT systems in a uniform way, based on an appropriate PPI. It also prescribes functionalities for transforming platform/system dependent data format to the semantically rich and platform independent VITAL semantic model (ontology). Furthermore, functionalities for managing ICO/sensor data in a platform agnostic way are specified. Key to operation of the VITAL systems will be a range of added-value functionalities of the VITAL platform agnostic data management functionalities. These include ICO and IoT services discovery, filtering of ICO data streams, CEP and IoT services orchestration requirements. On top of these functionalities the VITAL architecture foresees the operation of several visual tools for management, governance and development of IoT applications in smart cities.

Note that the VITAL architecture does not cover in-depth all the requirements and expectations from the VITAL systems and applications. Rather it provides a basis for fulfilling the requirements identified in earlier deliverables. We expect the full coverage of VITAL requirements to be achieved at later stages, during the detailed design and implementation of all the modules and building blocks that comprise the VITAL architecture. Nevertheless, the architecture provides the «big picture» through defining the structuring principles of the fully fledged VITAL platform. In this sense, the present deliverable is a major step in the evolution of the technical developments of the project, actually being a milestone that ensures the graceful transition to more detailed developments in the individual technical workpackages. Therefore, the VITAL architecture is a key outcome (of WP2) for the graceful continuation of the project's technical developments in other workpackages. Hence, the deliverable successfully concludes WP2.

Copyright © 2015 VITAL Consortium

REFERENCES

[Haller12] Stephan Haller et. al. «IoT Reference Model White Paper», IOT-I (257565), Deliverable D1.5, September 2012.

[Lê Tuán12] Anh Lê Tuán, Hoan Nguyen Mau Quoc, Martin Serrano, Manfred Hauswirth, John Soldatos, Thanasis G. Papaioannou, Karl Aberer: Global Sensor Modeling and Constrained Application Methods Enabling Cloud-Based Open Space Smart Services. UIC/ATC 2012: 196-203

[Serrano13] Martin Serrano, Manfred Hauswirth, Nikos Kefalakis, John Soldatos: A Self-Organizing Architecture for Cloud by Means of Infrastructure Performance and Event Data. CloudCom (1) 2013: 481-486

[Soldatos12] John Soldatos, Martin Serrano, Manfred Hauswirth: Convergence of Utility Computing with the Internet-of-Things. IMIS 2012: 874-879

Copyright © 2015 VITAL Consortium