IoT-A and FIWARE: bridging the barriers between the Cloud and IoT systems design and implementation

Kostas Stravoskoufos[1], Stelios Sotiriadis[2], Euripides G.M. Petrakis[1]
[1]*Intelligent Systems Laboratory Department of Electronic and Computer Engineering,*
Technical University of Crete (TUC), Chania, Greece, GR-73100
[2]*The Edward Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Canada*
kgstravo@intelligence.tuc.gr, s.sotiriadis@utoronto.ca, perakis@ intelligence.tuc.gr

Keywords: Cloud Computing, Internet of Things, IoT-A, FIWARE, Cloud service interoperability

Abstract: IoT systems are designed and implemented to address specific challenges based on domain specific requirements, thus not taking into consideration issues of openness, scalability, interoperability and use-case independence. As a result, they are less principled, vendor oriented and hardly replicable since the same IoT architecture cannot be used in more than one use-cases. To address the fragmentation of existing IoT solutions, the IoT-A project proposes an architecture reference model that defines the principles and standards for generating IoT architectures and promoting the interoperation of IoT solutions. However, IoT-A addresses the architecture design problem, and does not focus on whether existing cloud platforms can offer the tools and services to support the implementation of IoT-A compliant IoT systems. In this work we attempt to fill this gap and we propose an architectural approach based on IoT-A that focuses (as a use case) on the FIWARE open cloud platform that in turn provides the building blocks of Future Internet (FI) applications and services. We further correlate FIWARE and IoT-A approaches to identify the key features for FIWARE to support IoT-A compliant system implementations.

1 INTRODUCTION

Over the recent years, Cloud Computing and Internet of Things (IoT) have been rapidly advancing as the two fundamental technologies of the Future Internet (FI) concept as in [Galis and Gavras, 2013]. Focusing on different domains, IoT and Cloud computing have been evolved independently of each other. However, certain correlations between the two domains can be identified:

- Cloud computing offer the ideal environment for hosting IoT applications. The cloud's virtually unlimited resource pool can be paired with the exponentially growing demands of IoT applications. As enormous amounts of data are generated, cloud offers the requiring facility to store, processes as well as to access this data.
- The cloud offers an elastic environment that can scale-up well on demand and according to the needs of an increasing number of users, thus allowing the creation of more flexible IoT systems that can effectively adapt to their changing requirements.
- High availability is crucial for any IoT system and can be currently guaranteed by modern cloud providers.
- Cloud computing can bridge the gap between devices and applications by abstracting IoT management and composition services, acting as an intermediate layer and by hiding the complexities and the peculiarities of the IoT systems.

The lack of standardization in the IoT domain has resulted in the fragmentation of the approaches in IoT systems design and implementation. Existing IoT architectures have been evolved to address specific challenges with specific requirements, not taking into consideration issues of openness, scalability, interoperability and use-case independence. To address this problem, the IoT-A project of the EU [Bassi et al. 2013] proposes a Architecture Reference Model (ARM) defining the principles and guidelines for generating IoT architectures, providing the means to connect vertically closed systems in the communication (how devices interact with the system) and service (how services are integrated). The adoption of the IoT-A promotes the interoperation of IoT solutions by enabling interoperability as follows:

- At the communication layer to support the co-existence of various communication technologies (existing or emerging).
- At the service layer, thus ensuring smooth integration into the service layer of Future Internet applications.

IoT-A compliant architectures assure that generated knowledge will be modular and re-usable across domain or use-case specific boundaries. Leveraging on results from EU funded research, we propose to design IoT systems deriving from IoT-A based on FIWARE [FIWARE], the EU initiative for cloud services provision and one the candidate reference cloud implementations that can be exploited in the Future Internet. FIWARE platform comprises a set of Generic Enablers (GEs) that are considered general purpose and common to several usage areas. We compare the functionality of FIWARE GEs against IoT-A requirements and guidelines and point out weaknesses and missing points along with suggestions.

2 BACKGROUND WORK

IoT-A's Architecture Reference Model (ARM) is an abstract framework that comprises of a minimal set of unifying concepts, axioms and relationships for understanding significant relationships between entities of the IoT domain. It consists of several sub-models that set the scope for the IoT design space:

- *IoT domain model*: It is a top-level description (a UML diagram or ontology) of the IoT domain that describes the main concepts of the IoT like Devices, IoT Services and *Virtual Entities (VEs)* that is, anything that has a distinct existence, and also relations between these concepts.
- *IoT Information model*: An abstract description (UML diagram or ontology) for explaining information about elements or concepts defined in the *IoT Domain Model* (e.g., applicability of concepts).
- *IoT Functional model*: It identifies *Functional Groups (FGs)* that is, groups of functionalities, grounded in key concepts of the *IoT Domain Model.*
- *IoT Communication model*: Introduces concepts for handling the complexity of communication in an IoT environment. It is one FG in the *IoT Functional model.*
- *Trust, Security and Privacy (TSP) model*: Introduces functionality related to Trust, Security and Privacy. TSP is also one FG in the *IoT functional model.*

2.2 Reference Architecture

The IoT-A Reference Architecture (RA) shows how qualitative requirements are mapped to design choices. It is designed to allow the generation of many, potentially different, compliant IoT architectures that can be tailored to specific use cases. The Reference Architecture is based on the concepts of architectural views and architectural perspectives. Architectural views derive from the concerns of stakeholders (i.e., people, groups, or entities with an interest in the realization of the architecture). Views are representations of one or more structural aspects of an architecture that illustrate how the architecture addresses the concerns set by its stakeholders [Rozanski and Woods 2011]. Views define the main building blocks of the IoT-A RA description with each view addressing one aspect of the architectural structure:

- *Physical Entity View*: It describes all physical entities and their relations (e.g., sensors, actuators, environment measurements) in an IoT system. This view is not covered by IoT-A because it is use-case independent.
- *IoT context View*: It provides context information about physical entities such as the *Physical Entity Vie*w, this view is also not covered by IoT-A as it is use-case independent.
- *Functional View*: It describes the system's runtime Functional Components, their responsibilities, default functions, interfaces and primary interactions. The *Functional View* derives from the *Functional Model* and reflects the developer's perspectives on the system.
- *Information View*: Provides an overview on how (a) static information (i.e., VEs by means of hierarchies, semantics) and (b) dynamic information (i.e., information processing, storage, flow) is represented.
- *Deployment View*: It explains the operational behaviour of the functional components and the interplay of them.

Figure 1 demonstrates the relationship between IoT-A architectural views and model in the process of designing the actual system architecture. IoT-A also introduces architectural perspectives in IoT system architectures [Rozanski and Woods 2011] that are collections of activities, checklists, tactics and guidelines to guide the process of ensuring that a system exhibits a particular set of quality properties.

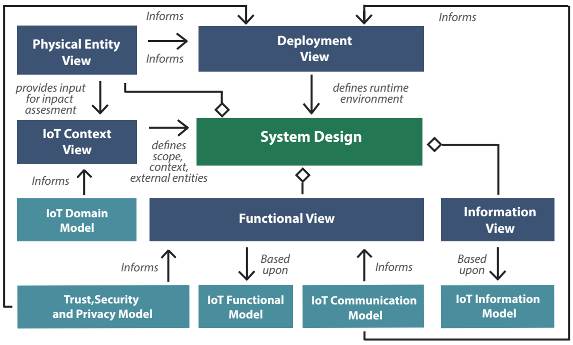


Figure 1: Relationship of IoT-A architectural views and models

The quality properties that have been identified as the most important in the IoT domain are (a) Evolution and interoperability to enable communication between devices and services, (b) Availability and resilience as the ability of the system to stay operational and handle failures, (c) Trust, security and privacy to handle such parameters, (d) Performance and scalability related to the monitoring of system's state and configuration of performance thresholds.

2.3 FIWARE Platform

FIWARE [FIWARE] develops cloud services to build novel FI applications that use remotely accessible modules (GEs) as APIs on a pay on demand model. FIWARE is the leading cloud vendor of the EU that offers open specification for services that could be spread at different geographically locations (and hosted in various nodes-namely XIFI nodes), and are available for utilization over the Internet. In FIWARE, the cloud model defines three main roles namely as the service consumer (the developer), the service provider (the FIWARE open specification) and the service creator (the GE implementer). Traditionally, the service creator generates a service that is hosted by FIWARE and represents the user requirements.

FIWARE is an innovative, open cloud-based infrastructure for cost-effective creation and delivery of Future Internet applications and services named GEs. GEs are considered as software modules that offer various functionalities along with protocols and interfaces for operation and communication. These include the cloud management of the infrastructure, the utilization of various IoT devices for data collection and the provision of APIs (e.g., tools for data analytics) and communication interfaces (e.g., gateways, messaging etc.). GEs are implementations of open specifications of the most common functionalities that are provided by WARE and are stored in a public catalogue, thus developers can easily browse and select appropriate APIs to use.

3 CORRELATION BETWEEN FIWARE AND IOT-A

The role of reference architectures such as IoT-A in the process of creating actual systems is to provide the key building blocks for the generation of the system's architecture. The generated architecture can then be used to guide the process of the actual system implementation. Reference architectures are application independent being more abstract than architecture which are designed with specific constraints and requirements in mind. On the other hand, architectures, which are generated by extracting essentials (parts of existing architectures, mechanisms, standards) from references architectures, should be application specific but platform independent allowing various implementations across different platforms

In the following we propose an architecture deriving from IoT-A and decide whether this architecture can be implemented on FIWARE. Although architectures should be composed by several views, reflecting the concerns of the different stakeholders on the system, in our study we focus on the Functional View of IoT-A adopting the developers' perspective, as our main concern on the system is the implementation.

Figure 2 illustrates the *Functional View* diagram of IoT-A Reference Architecture along with the nine Functionality Groups (FGs) of the Functional Model, which are discussed below. Each FG consists of Functional Components (FC). Each FCs describes a unique functionality and is able to communicate with the others in a distributed environment. Figure 2 shows also existing FIWARE GE mapped to FCs.

Process Management FG: Provides the functional concepts necessary to conceptually integrate the IoT world into traditional (business) processes. The *Process Modeling FC* which provides the tools required for modeling IoT-aware business processes that will be serialized and executed in the *Process Execution FC*, which is responsible for deploying process models to the execution environments.

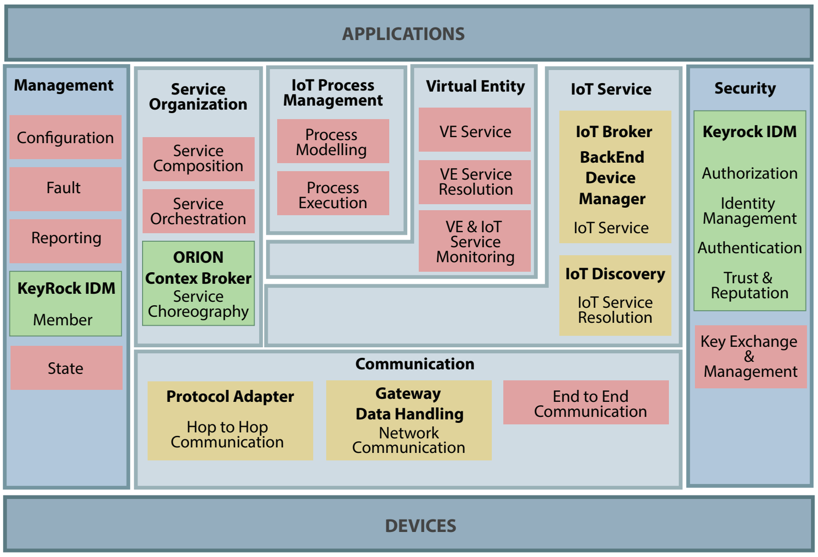


Figure 2: IoT-A Reference Architecture and FIWARE GEs mapped to FCs

Service Organization FG: Acts as a communication hub between several other Functional Groups by composing and orchestrating Services of different levels of abstraction. The *Service Orchestration FC* resolves the IoT Services that are suitable to fulfill service requests coming from the *Process Execution FC* or from Users while the *Service Composition FC* is responsible for creating services with extended functionality by composing IoT services with other services. Finally, the *Service Choreography FC* offers a broker that handles Publish/Subscribe communication between services.

Virtual Entity FG: Provides functionality for the interaction of VEs with the IoT system, for VE look-up and discovery and for providing information concerning VEs. The *VE Resolution FC* provides discovery services for associations between VEs and IoT. *VE & IoT Service Monitoring FC* is responsible for automatically finding new associations based on service descriptions and information about VE's. Finally, the *VE Service FC* handles entity services (e.g., access to an entity for reading and/or updating the value(s) of the entity's attributes).

Service FG: Provides IoT services as well as functionalities for discovery, look-up, and name resolution of IoT Services. The *IoT Service FC* exposes IoT resources (e.g., information retrieved from sensors) making them accessible to other parts of the IoT system. This FC can also be used for delivering information to a resource in order to control actuator devices or to configure the resource (e.g., manage access control and permissions on the resource). IoT Services can be invoked either in a synchronous way by responding to service requests or in an asynchronous way by sending notifications according to subscriptions. Finally, the *IoT Service Resolution FC* provides service discovery and resolution functionalities along with service description management capabilities.

Communication FG: Abstracts the interaction schemes derived from the variety of communication technologies in IoT systems in order to provide a common interface to the *IoT Service FG*. The *Hop To Hop Communication FC* provides the first layer of abstraction from the device's physical communication technology, the *Network Communication FC* enables communication between networks and, finally, the *End to End Communication FC* offers reliable transfer, transport and, translation functionalities, proxy/gateway support and setting configuration parameters when the communication crosses different networking environments.

Security FG: It is responsible for security and privacy matters in IoT-A-compliant IoT systems. The *Authorization FC* is used to apply access control and access policy management (i.e., to determine whether an action is authorized or not, and to manage access policies) while, the *Authentication FC* is used for user and service authentication. The *Identity Management FC* addresses privacy by issuing and managing pseudonyms and accessory information to trusted subjects so that they can operate anonymously. The *Key Exchange and Management (KEM) FC* enables secure communications ensuring integrity and confidentiality by distributing keys upon request in a secure way. Finally, *Trust and Reputation FC* collects user reputation scores and calculates service trust levels.

Management FG: It is responsible for the composition and tracking of actions that involve the other FGs. The *Configuration FC* is responsible for initialising the system's configuration (e.g., gathering applying configurations from FC's and Devices). It is also responsible for tracking configuration changes and planning for future extensions of the system. The *Fault FC* is used to identify, isolate, correct and log faults that occur in the IoT system. The *Member FC* is responsible for the management of the membership of any relevant entity (FG, FC, VE, IoT Service, Device, Application, User) to an IoT system working in cooperation with the *Authorisation* and *Identity Management FCs* of the *Security FG. The Reporting FC* generates reports about the system and, finally, the *State FC* can change or enforce a particular state on the system by issuing a sequence of commands to the other FCs. Moreover, it is constantly monitoring the state of the system notifying subscribers about changes.

3.1 FIWARE IMPLEMENTATION

In the following we show how the IoT-A Reference Architecture can be realized and implemented in FIWARE using FIWARE GEs. We go through each FG which GEs can supply the functionality of FCs pointing out missing points and weaknesses and making suggestions.

Service Organization FG: The functionality of the Service Choreography FC is encapsulated by the *Orion Context Broker GE*. The broker offers Publish/Subscribe capabilities, providing NGSI9/10 interfaces as in FIWARE Catalogue, allowing clients to do several operations like register context producer applications, update context information and get notifications when context information changes take place or with a given frequency. The broker allows also querying context information and stores context information update so queries are resolved based on that information. The functionalities of the other two FCs of this FG are not covered by any FIWARE GE.

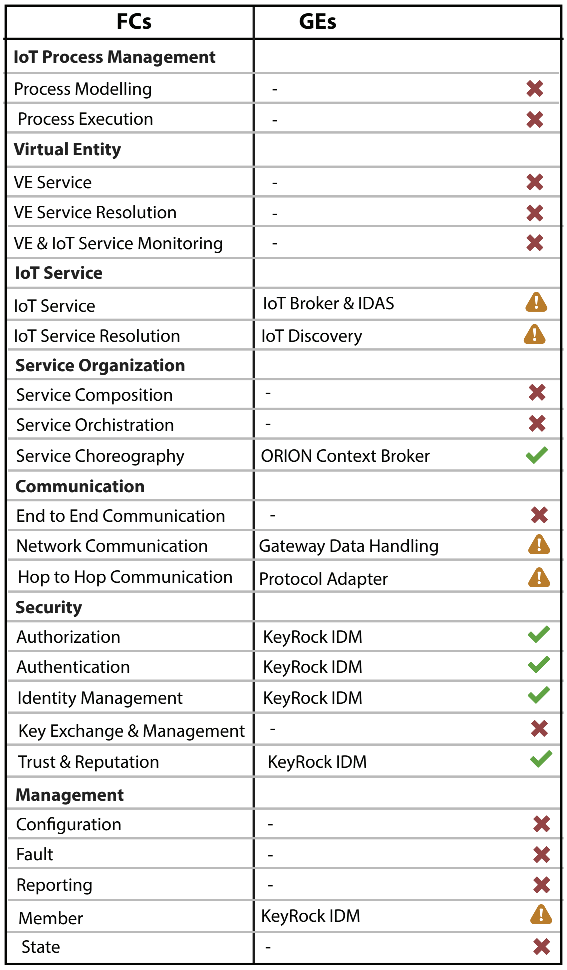
Service FG: The functionality of the two FCs in the IoT Service FG is partially covered by the IoT *Discovery*, the *IoT Broker* and the *Backend Device Management (IDAS) GEs*. The *IoT Discovery GE* allows context producers to register their IoT Objects in linked-data format and, in turn allows context consumers to discover them using search techniques. Although the *IoT Discovery GE* offers a device discovery mechanism, it does not offer a service resolution mechanism thus, covers only partially the functionality of the *IoT Service Resolution FC*. A service discovery and resolution mechanism should be added to this GE.

The *IoT Broker GE* and the *Backend Device Management GE*, each encapsulate part of the functionality of the *IoT Service FC*. The *IoT Broker GE* retrieves and aggregates information from IoT devices acting as a middleware component that separates IoT applications from devices. The GE is based on *NGSI9/10* and closes the gap between information-centric applications and device-centric IoT installations by communicating simultaneously with large quantities of IoT gateways and devices in order to obtain exactly the information that is required by the running IoT applications. By using the *IoT Broker GE*, all IoT devices can be abstracted to be viewed as *NGSI* entities on a higher level hiding the complexity of the IoTs from developers. The *IoT Broker GE* does not currently support delivering information to resources in order to control them or configure them. This is supported by the *Backend Device Management GE* which provides an API for M2M application developers and a device communication API for device (sensor/actuators/gateways) communication which currently implements the *SensorML* and *Lightweight SensorML* protocols.

The *Backend Device Management GE* collects data from devices and translates them into *NGSI* events available at a context broker. Application developers can use data and send commands through the broker. An open source Reference Gateway referred to as *FIGWAY* is also offered for Raspberry PI and Z-wave devices. Although this GE complements the *IoT Broker GE* towards supporting the specifications of the *IoT Service FC* it does not make a clear separation of the layers as they are defined by IoT-A and it's functionality intervenes into the communication layer. Moreover, this GE currently works only with *FIGWAY* meaning that it can collect data and manage only specific devices controlled by the Raspberry PI and Z-wave gateways. It should be extended to support any kind of gateway and support all protocols other than *CoaP* (e.g., XMPP, MQTT, REST).

Communication FG: The *Protocol Adapter GE* handles low-level communication between devices and the rest of the IoT system encapsulating the functionality of the *Hop to Hop Communication FC*.

Although it does cover the required functionality, the *Protocol Adapter* GE is currently working with specific devices over a specific platform and, thus, needs to be extended to be compatible with all available devices and platforms. The *Network Communication FC* functionality is partially covered by the *Gateway Data Handling GE*, which is designed to provide a common access in real time to all data. Using a simple local XML storage, the GE locally stores relevant processed data.



FCs	GEs	
IoT Process Management		
Process Modelling	-	✖
Process Execution	-	✖
Virtual Entity		
VE Service	-	✖
VE Service Resolution	-	✖
VE & IoT Service Monitoring	-	✖
IoT Service		
IoT Service	IoT Broker & IDAS	⚠
IoT Service Resolution	IoT Discovery	⚠
Service Organization		
Service Composition	-	✖
Service Orchistration	-	✖
Service Choreography	ORION Context Broker	✓
Communication		
End to End Communication	-	✖
Network Communication	Gateway Data Handling	⚠
Hop to Hop Communication	Protocol Adapter	⚠
Security		
Authorization	KeyRock IDM	✓
Authentication	KeyRock IDM	✓
Identity Management	KeyRock IDM	✓
Key Exchange & Management	-	✖
Trust & Reputation	KeyRock IDM	✓
Management		
Configuration	-	✖
Fault	-	✖
Reporting	-	✖
Member	KeyRock IDM	⚠
State	-	✖

Figure 3: IoT-A FIWARE Correlation Table

Security FG: *KeyRock Identity Management (IDM) GE* provides secure and private authentication from users to devices, networks and services, authorization, user profile management and privacy-preserving disposition of personal data, encapsulating the functionality of four out of five FCs. The *Key Exchange* and *Management FC* functionalities are not supported by an FIWARE GE. Also, as the KeyRock architecture encapsulates the functionality of four out of five FCs, the architecture becomes less modular as there is no clear separation of modules.

Management FG: The functionality of the *Member FC* is covered by the *KeyRock Identity Management (IDM) GE* which handles all entities membership in the IoT system. This GE encapsulates functionality of FCs in two different layers of the architecture demoting modularity and task separation. The functionality of the remaining *Management FG FCs* is not offered by any FIWARE GE.

Figure 2 illustrates the IoT-A Reference Architecture substituting FCs with FIWARE GEs. Green color denotes that the GE can effectively providing the specified functionality of the FC, yellow that it does so partially and red denotes there is currently no GE that can provide the functionality of the FC. Finally, we provide an array with all FCs and the corresponding GEs in Figure 3.

4 CONCLUSIONS

In this work, we focused on the relationship between FIWARE and IoT-A. We proposed an architecture based on IoT-A and showed how it can be implemented on FIWARE using GEs. The results show that FIWARE cannot currently support IoT architectures fully compatible with the IoT-A standards and specifications as it is lacking the tools (GEs) to do so. New GEs need to be designed and implemented to provide the required functionality while a clear distinction of the architecture layers specified by IoT-A should be as well.

REFERENCES

Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Kranenburg, R. v., Lange, S. & Meissner, S. (eds.) (2013). *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Heidelberg: Springer.

Galis A. and Gavras, A. (2013) *The Future Internet: Future Internet Assembly 2013 Validated Results and New Horizons*. Springer Publishing Company, Incorporated.

Rozanski, N., Woods, E., (2011) *Software systems architecture – working with stakeholders using viewpoints and perspectives*. Addison Wesley, Boston.

FIWARE: https://www.fiware.org