MODULE - IV

IoT Architecture-State of the Art

Introduction, State of the art, **Architecture Reference Model-** Introduction, Reference Model and architecture, IoT reference Model.

IoT Reference Architecture: Introduction, Functional View, Information View, Deployment and Operational View, Other Relevant architectural views.

4.1 Introduction

An ARM is useful as a tool that establishes a common language across all the possible stakeholders of an M2M or IoT system. It can also serve as a starting point for creating concrete architectures of real systems when the relevant boundary conditions have been applied.

4.2 State of the art

Several Reference Architectures and Models exist both for M2M and IoT systems. The four most popular ones from which the specific Reference Architecture is discussed below

4.2.1 European Telecommunications Standards Institute M2M/oneM2M

The European Telecommunications Standards Institute (ETSI) in 2009 formed a Technical Committee (TC) on M2M topics aimed at producing a set of standards for communication among machines from an end to- end viewpoint.

- The ETSI M2M specifications are based on specifications from ETSI as well as other standardization bodies such as the IETF (Internet Engineering Task Force), 3GPP (3rd Generation Partnership Project), OMA (Open Mobile Alliance), and BBF (Broadband Forum).
- ETSI M2M produced the first release of the M2M standards in early 2012, while in the middle of 2012 seven of the leading Information and Communications Technology (ICT) standards organizations (ARIB, TTC, ATIS, TIA, CCSA, ETSI, TTA) formed a global organization called oneM2M Partnership Project (oneM2M) in order to develop M2M specifications, promote the M2M business, and ensure the global functionality of M2M systems.

4.2.1.1 ETSI M2M high-level architecture

The figure 4.1 shows the high-level ETSI M2M architecture.This high-level architecture is a combination of both a functional and topological view showing some functional groups (FG)

clearly associated with pieces of physical infrastructure (e.g. M2M Devices, Gateways) while other functional groups lack specific topological placement.

There are two main domains, a network domain and a device and gateway domain. The boundary between these conceptually separated domains is the topological border between the physical devices and gateways and the physical communication infrastructure (Access network).

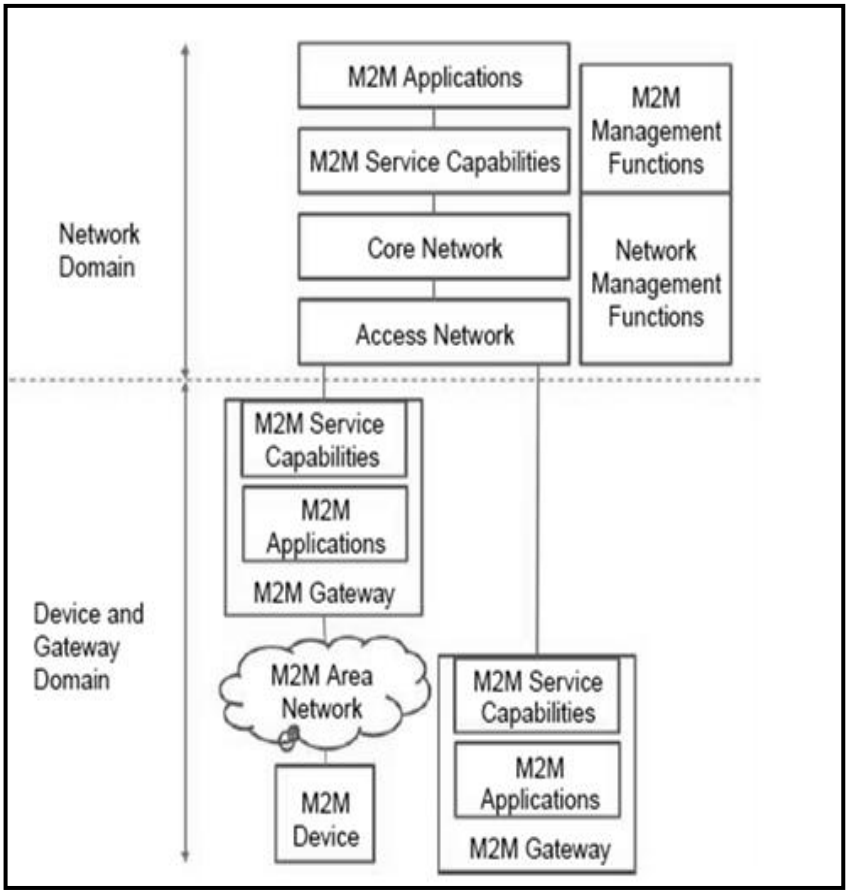


Figure 4.1: ETSI M2M High Level Architecture

The Device and Gateway Domain contains the following functional/topological entities:

• **M2M Device:** This is the device of interest for an M2M scenario, for example, a device with a temperature sensor. An M2M Device contains M2M Applications and M2M Service Capabilities. An M2M device connects to the Network Domain either directly or through an M2M Gateway:

- Direct connection: The M2M Device is capable of performing registration, authentication, authorization, management, and provisioning to the Network Domain. Direct connection also means that the M2M device contains the appropriate physical layer to be able to communicate with the Access Network.

Through one or more M2M Gateway: This is the case when the M2M device does not have the appropriate physical layer, compatible with the Access Network technology, and therefore it needs a network domain proxy. Moreover, a number of M2M devices may form their own local M2M Area Network that typically employs a different networking technology from the Access Network. The M2M Gateway acts as a proxy for the Network Domain and performs the procedures of authentication, authorization, management, and provisioning. An M2M Device could connect through multiple M2M Gateways.

• **M2M Area Network:** This is typically a local area network (LAN) or a Personal Area Network (PAN) and provides connectivity between M2M Devices and M2M Gateways. Typical networking technologies are IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee, IETF 6LoWPAN/ROLL/CoRE), MBUS, KNX (wired or wireless) PLC, etc.

• **M2M Gateway:** The device that provides connectivity for M2M Devices in an M2M Area Network towards the Network Domain. The M2M Gateway contains M2M Applications and M2M Service Capabilities. The M2M Gateway may also provide services to other legacy devices that are not visible to the Network Domain. The Network Domain contains the following functional/topological entities:

• **Access Network**: this is the network that allows the devices in the Device and Gateway Domain to communicate with the Core Network. Example Access Network Technologies are fixed (xDSL, HFC) and wireless (Satellite, GERAN, UTRAN, E-UTRAN W-LAN, WiMAX).

• **Core Network:** Examples of Core Networks are 3GPP Core Network and ETSI TISPAN Core Network. It provides the following functions:

- IP connectivity.
- Service and Network control.
- Interconnection with other networks.
- Roaming.

• **M2M Service Capabilities:** These are functions exposed to different M2M Applications through a set of open interfaces. These functions use underlying Core Network functions, and their objective is to abstract the network functions for the sake of simpler applications. More details about the specific service capabilities are provided later in the chapter.

• **M2M Applications:** These are the specific M2M applications (e.g. smart metering) that utilize the M2M Service Capabilities through the open interfaces.

• **Network Management Functions:** These are all the necessary functions to manage the Access and Core Network (e.g. Provisioning, Fault Management, etc.).

• **M2M Management Functions:** These are the necessary functions required to manage the M2M Service Capabilities on the Network Domain while the management of an M2M Device or Gateway is performed by specific M2M Service Capabilities. There are two M2M Management functions:

• **M2M Service Bootstrap Function (MSBF):** The MSBF facilitates the bootstrapping of permanent M2M service layer security credentials in the M2M Device or Gateway and the M2M Service Capabilities in the Network Domain. In the Network Service Capabilities Layer, the Bootstrap procedures perform, among other procedures, provisioning of an M2M Root Key (secret key) to the M2M Device or Gateway and the M2M Authentication Server (MAS).

• **M2M Authentication Server (MAS):** This is the safe execution environment where permanent security credentials such as the M2M Root Key are stored. Any security credentials established on the M2M Device or Gateway are stored in a secure environment such as a trusted platform module.

ETSI M2M functional architecture is that it focuses on the high-level specification of functionalities within the M2M Service Capabilities functional groups and the open interfaces between the most relevant entities, while avoiding specifying in detail the internals of M2M Service Capabilities.

The interfaces are specified in different levels of detail, from abstract to a specific mapping of an interface to a specific protocol (e.g. HTTP (Fielding 2000),IETF CoAP The most relevant entities in the ETSI M2M architecture are the M2M Nodes and M2M Applications. An M2M Node can be a Device M2M, Gateway M2M, or Network M2M Node Figure 4.2

An M2M Node is a logical representation of the functions on an M2M Device, Gateway, and Network that should at least include a Service Capability Layer (SCL) functional group.

An M2M Application is the main application logic that uses the Service Capabilities to achieve the M2M system requirements. The application logic can be deployed on a Device (Device Application, DA), Gateway (Gateway Application, GA) or Network (Network Application, NA).

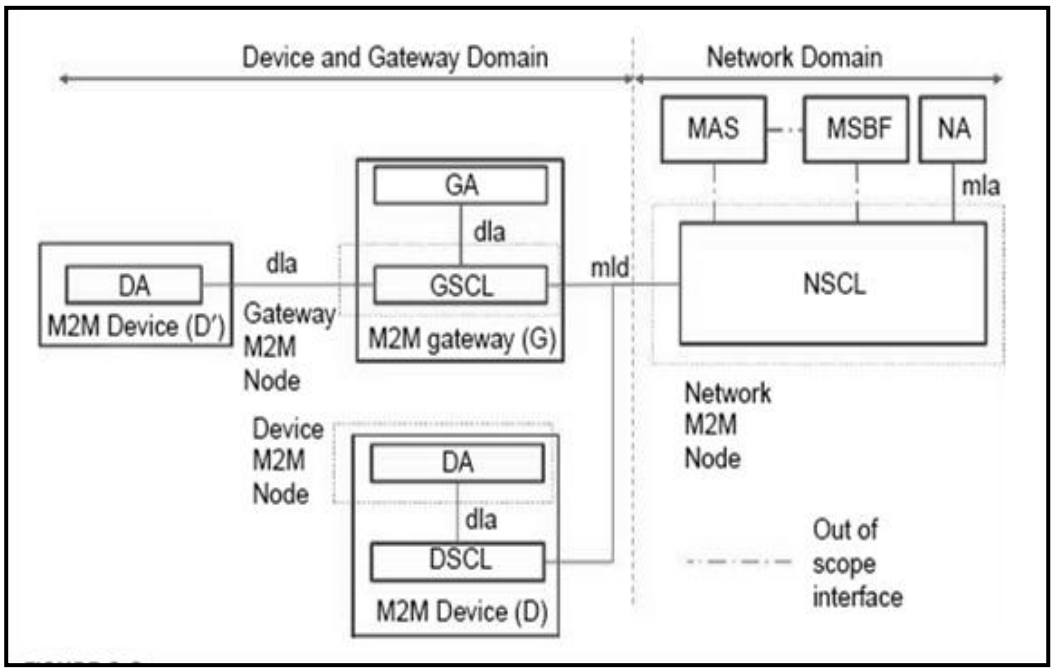


Figure 4.2: M2M service Capabilities, M2M Nodes and Open Interfaces

The SCL is a collection of functions that are exposed through the open interfaces or reference points mIa, dIa, and mId (ETSI M2M TC 2013b).Because the main topological entities that SCL can deploy are the Device,Gateway, and Network Domain, there are three types of SCL: DSCL (Device Service Capabilities Layer), GSCL (Gateway Service Capabilities Layer), and NSCL (Network Service Capabilities Layer).

SCL functions utilize underlying networking capabilities through technology-specific interfaces. For example, an NSCL using a 3GPP type of access network uses 3GPP communication services interfaces.

The ETSI M2M Service Capabilities are recommendations of functional groups for building SCLs,but their implementation is not mandatory, while the implementation of the interfaces mIa, dIa, and mId is mandatory for a compliant system.

4.2.1.2 ETSI M2M service capabilities

All the possible Service Capabilities (where "x" is Network, Gateway, and Device are shown in Figure 4.3:

1. Application Enablement (xAE). The xAE service capability is an application facing functionality and typically provides the implementation of the respective interface: NAE implements the mIa interface and the GAE and DAE implement the dIa interface.The xAE includes registration of applications (xA) to the respective xSCL; for example, a Network Application towards the NSCL.

2. **Generic Communication (xGC).** The NGC is the single point of contact for communication towards the GSCL and DSCL. It provides transport session establishment and negotiation of security mechanisms, potentially secure transmission of messages, and reporting of errors such as transmission errors. The GSC/DSC is the single point of contact for communication with the NSCL, and they both perform similar operations to the NGC (e.g. secure message transmissions to NSCL). The GSC performs a few more functions such as relaying of messages to/from NSCL from/to other SCs in the GSCL, and handles name resolution for the requests within the M2M Area Network.
3. **Reachability, Addressing, and Repository (xRAR).**

   This is one of the main service capabilities of the ETSI M2M architecture. The NRAR hosts mappings of M2M Device and Gateway names to reachability information, and scheduling information relating to reachability, such as whether an M2M Device is reachable between 10 and 11 o'clock.

   - It provides group management (creation/update/deletion) for groups of M2M Devices and Gateways,stores application (DA, GA, NA) data, and manages subscriptions to these data, stores registration information for NA, GSCL, and DSCL,and manages events (subscription notifications).
   - The GRAR provides similar functionality to the NRAR, such as maintaining mappings of the names of M2M Devices or groups to reachability information (routable addresses, reachability status, and reachability scheduling),storing DA, GA, NSCL registration information, storing DA, GA, NA,GSCL, NSCL data and managing subscriptions about them, managing groups of M2M Devices, and managing events. Similar to NRAR and GRAR, the DRAR stores DA, GA, NA, DSCL, and NSCL data and manages subscriptions about these data, stores DA registration and NSCL information, provides group management for groups of M2M Devices and event management.
4. **Communication Selection (xCS):** This capability allows each xSCL to select the best possible communication network when there is more than one choice or when the current choice becomes unavailable due to communication errors. The NCS provides

such a selection mechanism based on policies for reaching an M2M Device or Gateway, while the GCS/DCS provides a similar selection mechanism for reaching the NSCL.

5. **Remote Entity Management (xREM)**
   - The NREM provides management capabilities such as Configuration Management (CM) for M2M Devices and Gateways (e.g. installs management objects in device and gateways), collects performance management (PM) and Fault Management (FM) data and provides them to NAs or M2M Management Functions, performs device management to M2M Devices and Gateways such as firmware and software (application,SCL software) updates, device configuration, and M2M Area Network configuration.
   - The GREM acts as a management client for performing management operations to devices using the DREM and a remote proxy for NREM to perform management operations to M2M Devices in the M2M Area Network. Examples of proxy operations are mediation of NREM-initiated software updates, and handling management data flows from NREM to sleeping M2M Devices.The DREM provides the CM, PM, and FM counterpart on the device (e.g. start collecting radio link performance data) and provides the device-side software and firmware update support.

6. SECurity (xSEC). These capabilities provide security mechanisms such as M2M Service Bootstrap, key management, mutual authentication,and key agreement (NSEC performs mutual authentication and key agreement while the GSEC and DESC initiate the procedures), and potential platform integrity mechanisms.

7. History and Data Retention (xHDR). The xHDR capabilities are optional capabilities, in other words, they are deployed when required by operator policies. These capabilities provide data retention support to other xSCL capabilities (which data to retain) as well as messages exchanged over the respective reference points.

8. Transaction Management (xTM). This set of capabilities is optional and provides support for atomic transactions of multiple operations.

An atomic transaction involves three steps:

(a) propagation of a request to a number of recipients

(b) collection of responses, and

(c) commitment or roll back whether all the transactions successfully completed or not.

9. Compensation Broker (xCB). This capability is optional and provides support for brokering M2M-related requests and compensation between a Customer and a Service Provider. In this context a Customer and a Service Provider is an M2M Application.

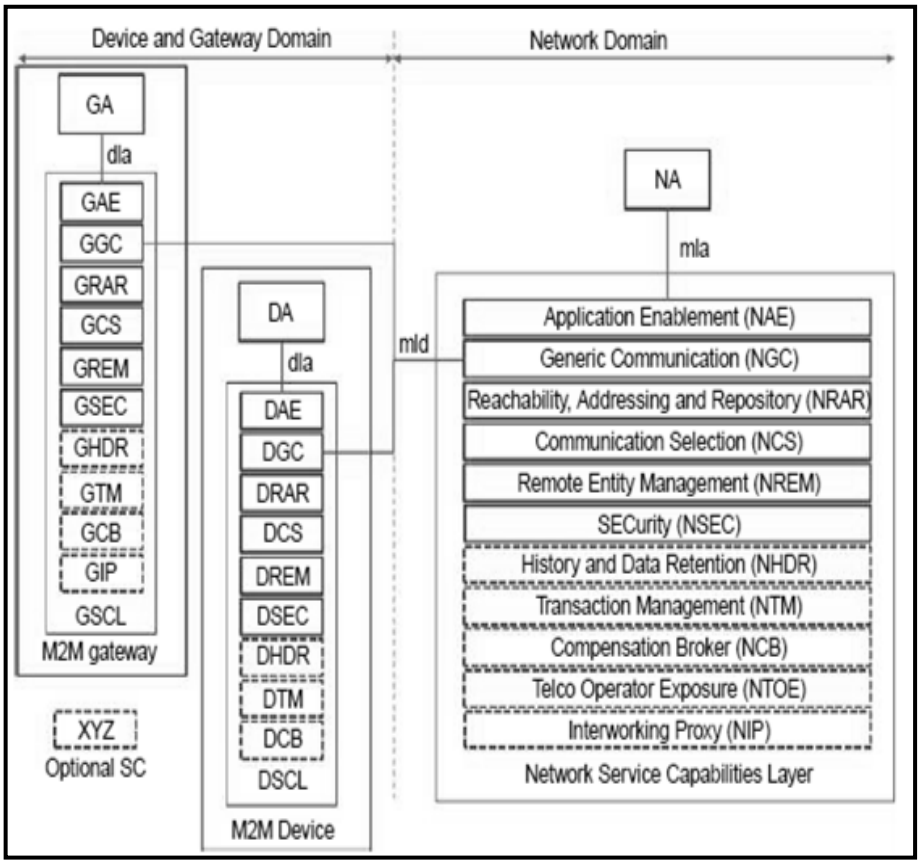


Figure 4.3: M2M Capabilities for different M2M Nodes.

10.Telco Operator Exposure (NTOE). This is also an optional capability and provides exposure of the Core Network service offered by a Telecom Network Operator.

11. Interworking Proxy (xIP). This capability is an optional capability and provides mechanisms for connecting non-ETSI M2M Devices and Gateways to ETSI SCLs. NIP provides mechanisms for non-ETSI M2M Devices and Gateways to connect to NSCL while GIP provides the functionality for non-compliant M2M Devices to connect to GSCL via the reference point dIa, and the DIP provides the necessary mechanisms to connect non-compliant devices to DSCL via the dIa reference point.

4.2.1.3 ETSI M2M interfaces

The main interfaces mIa, dIa, and mId (ETSI M2M TC 2013b) can be briefly described as follows:

• **mIa:** This is the interface between a Network Application and the Network Service Capabilities Layer (NSCL). The procedures supported by this interface are (among others) registration of a Network Application to the NSCL, request to read/write information to NSCL, GSCL, or DSCL, request for device management actions (e.g. software updates), subscription and notification of specific events.

• **dIa:** This is the interface between a Device Application and (D/G)SCL or a Gateway Application and the GSCL. The procedures supported by this interface are (among others) registration of a Device/Gateway Application to the GSCL, registration of a Device Application to the DSCL, request to read/write information to NSCL, GSCL, or DSCL, subscription and notification of specific events.

• **mId:** This is the interface between the Network Service Capabilities Layer (NSCL) and the GSCL or the DSCL. The procedures supported by this interface are (among others) registration of a Device/Gateway SCL to the NSCL, request to read/write information to NSCL, GSCL,or DSCL, subscription and notification of specific events.

4.2.1.4 ETSI M2M resource management

The ETSI M2M architecture assumes that applications (DA, GA, NA) exchange information with SCLs by performing CRUD (Create, Read,Update, Delete) operations on a number of Resources following the RESTful (Representational State Transfer) architecture paradigm. One of the principles of this paradigm is that representations of uniquely addressed resources are transferred from the entity that hosts these resources to the requesting entity. In the ETSI M2M architecture, all the state information maintained in the SCLs is modeled as a resource structure that architectural entities operate on. A very simplified view of the resource structure is a collection of containers of information structured in hierarchical manner following a corresponding hierarchy of a unique naming structure.

In addition to the CRUD operations, ETSI M2M defines two more operations: NOTIFY and EXECUTE. The NOTIFY operation is triggered upon a change in the representation of a resource, and results in a notification sent to the entity that originally subscribed to monitor changes to the resource in question. This operation is not an orthogonal operation to the CRUD set, but can be implemented by an UPDATE operation from the resource host towards the requesting entity.

The EXECUTE operation is not orthogonal as well, but can be implemented by an UPDATE operation with no parameters from the requesting entity to a specific resource.

When a requesting entity issues an EXECUTE operation towards a specific resource, the specific resource executes a specific task.

The following example in Figure 4.4 demonstrates how an ETSI M2M entity communicates with another entity using the CRUD and NOTIFIES operations. Assume that a device application (DA) is programmed to send a sensor measurement to a network application (NA). The DA using the DSCL updates the representation of a specific resource (Ra) residing on the NSCL (steps 1 and 2 in Figure 4.4). The NA has configured the NSCL to be notified when the specific resource is updated, in which case the NA reads the updated representation (steps 4 and 5, Figure 4.4).

The root of the hierarchical resource tree is the ,sclBase. resource, and contains all the other resources hosted by the SCL. The root has a unique identifier. In case the RESTful architecture is implemented in a real system by using web resources, the ,sclBase. has an absolute URI (Universal Resource Identifier), for example, "http://m2m.operator1.com/ some/path/to/base". The top-level structure of the ,sclBase. resource is shown in Figure 4.5. The different fonts used in this figure 4.5 denote

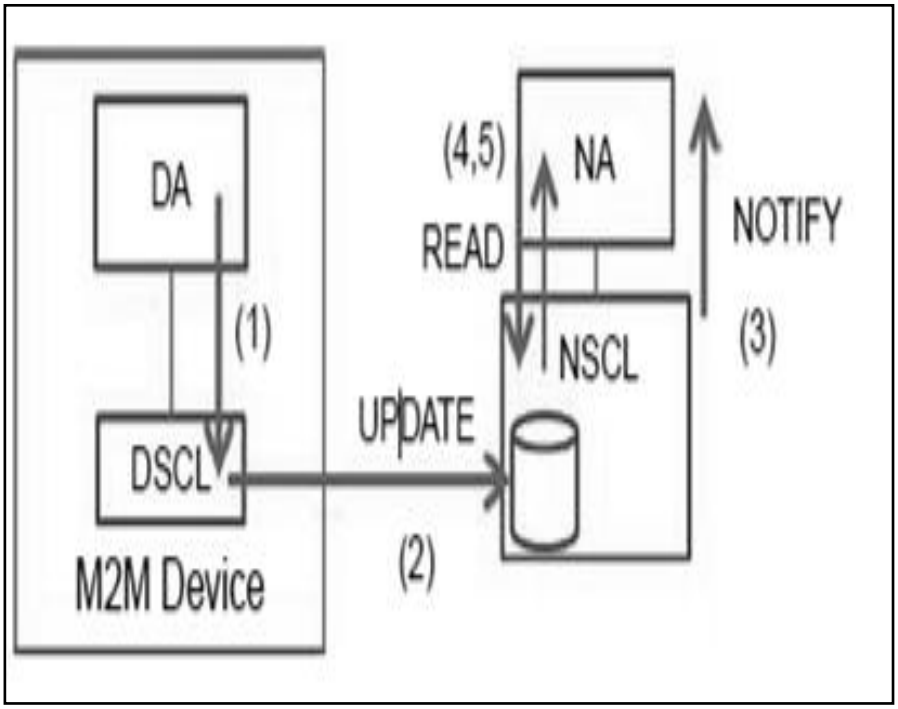


Figure 4.4 : Communication Between DA and NA Using the SCLs.

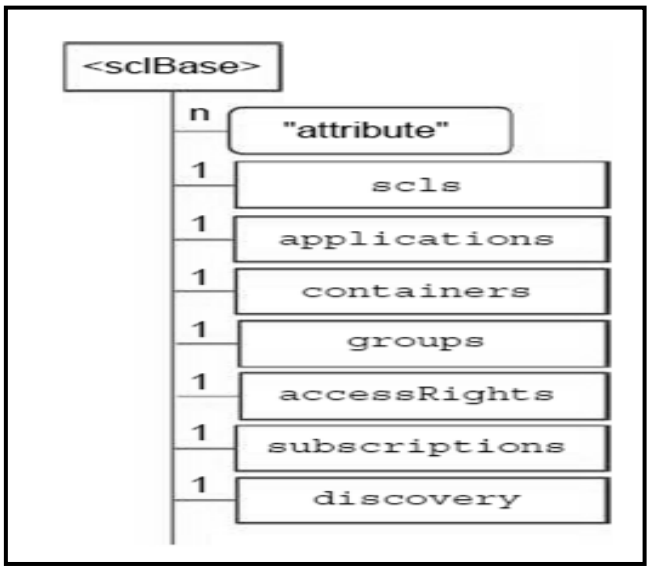


Figure 4.5: The Top-Level Structure of the, sclBase Resource.

different information semantics. A term between the symbols "," and "." denotes an arbitrary resource name; for example, ,sclBase. In **Figure 4.5.** A term within quotes ("") denotes a placeholder for one or more fixed names. In this specific case, "attribute" represents a member of a fixed list of attributes for the resource ,sclBase.. A term annotated in Courier New font such as scls denotes a literal resource name used by the specification as is. The ,sclBase. is structured as a tree with different branches, each of which is annotated with a designation of its cardinality. For example, the ,sclBase. contains n attributes, one scls resource, one applications resource, etc.

4.2.2 International Telecommunication Union Telecommunication sector view

The Telecommunication sector of the International Telecommunication Union (ITU-T) has been active on IoT standardization since 2005 with the Joint Coordination Activity on Network Aspects of Identification Systems (JCA-NID), which was renamed to Joint Coordination Activity on IoT (JCA-IoT) in 2011. During the same year apart from this coordination activity on IoT, ITU-T formed the specific IoT Global Standards Initiative (IoT-GSI) activity in order to address specific IoT-related issues. The latest ITU-T Recommendation, Y.2060 (ITU-T 2013) (Vermesan & Friess 2013), provides an overview of the IoT space with respect to ITU-T. This recommendation describes a high-level overview of the IoT domain model and the IoT functional model as a set of Service Capabilities similar to ETSI-M2M.

The ITU-T IoT domain model includes a set of physical devices that connect directly or through gateway devices to a communication network that allows them to exchange information with other devices, services, and applications.

ITU-T formed the specific IoT Global Standards Initiative (IoT-GSI) activity in order to address specific IoT-related issues. The latest ITU-T Recommendation, Y.2060 (ITU-T 2013) (Vermesan & Friess 2013), provides an overview of the IoT space with respect to ITU-T. This recommendation describes a high-level overview of the IoT domain model and the IoT functional model as a set of Service Capabilities similar to ETSI-M2M.

The ITU-T IoT domain model includes a set of physical devices that connect directly or through gateway devices to a communication network that allows them to exchange information with other devices, services, and applications.

If the ITU-T and ETSI M2M models/architectures are compared, the two approaches are very similar in terms of Service Capabilities for M2M.However, the ITU-T IoT domain model with physical and virtual things,and the physical and virtual world model, shows the influences of more modern IoT architectural models and references such as the IoT-A.

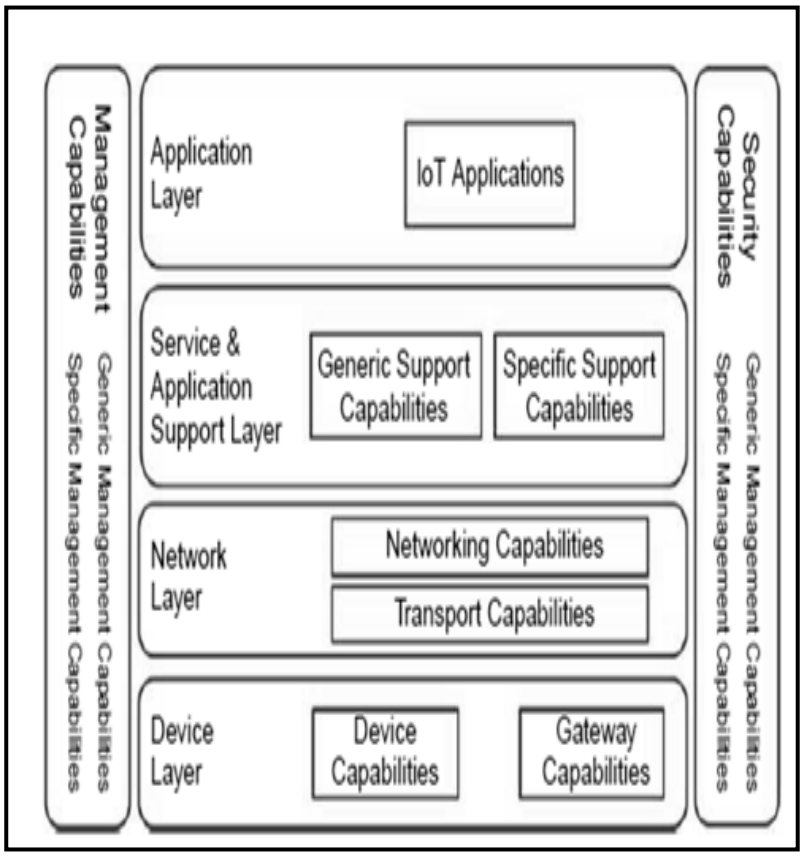


Figure 4.6: ITU-T IoT Reference Model.

4.2.3 Internet Engineering Task Force architecture fragments

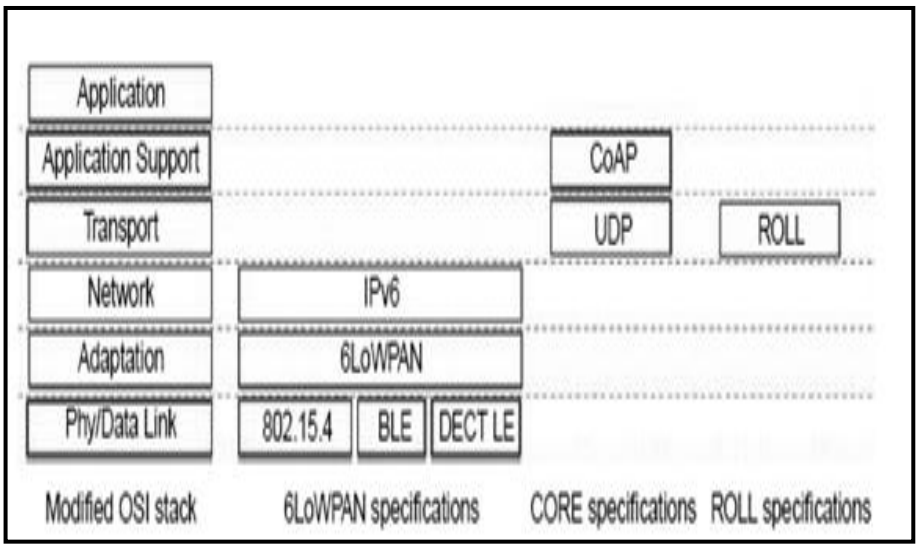


Figure 4.7: IETF Working Groups and Specification Scope

The modified Open Systems Interconnection (OSI) model in which several layers are merged because of the implementation on constrained devices.This is illustrated in Figure 4.7 as one layer called Application Support which includes the Presentation and Session Layers combined.

Moreover, one intermediate layer is introduced: the Adaptation Layer positioned between the Physical/Data Link and the Network Layer and whose main function is to adapt the Network Layer packets to Phy/Link layer packets among others.

An example of an adaptation layer is the 6LoWPAN layer designed to adapt IPv6 packets to IEEE 8021.5.4/Bluetooth Low Energy (BLE)/DECT Low Energy packets. An example of an Application Support Layer is IETF Constrained Application Protocol (CoAP), which provides reliability and RESTful operation support to applications.

Apart from the core of the specifications, the IETF CoRE workgroup includes several other draft specifications that sketch parts of an architecture for IoT. The CoRE Link Format specification describes a discovery method for the CoAP resources of a CoAP server.

The IETF CoRE working group has also produced a draft specification for a Resource Directory. A Resource Directory is a CoAP server resource (/rd) that maintains a list of resources, their corresponding server contact information (e.g. IP addresses or fully qualified domain name, or FQDN), their type, interface, and other information similar to the information that the CoRE Link Format document specifies (Figure 4.8a).

An RD plays the role of a rendezvous mechanism for CoAP Server resource descriptions, in other words, for devices to publish the descriptions of the available resources and for CoAP clients to locate resources that satisfy certain criteria such as specific resource types.

A Mirror Server (Vial 2012) is a rendezvous mechanism for CoAP Server resource presentations. A Mirror Server is a CoAP Server resource (/ms) that maintains a list of resources and their cached representations (Figure 4.8b). A CoAP Server registers its resources to the Mirror Server, and upon registration a new mirror server resource is created on the Mirror Server with a container (mirror representation) for the original server representation. The original CoAP Server updates the mirror representation either periodically or when the representation changes. A CoAP Client that retrieves the mirror representation receives the latest updated representation from the original CoAP Server.

The Mirror Server is useful when the CoAP Server is not always available for direct access.

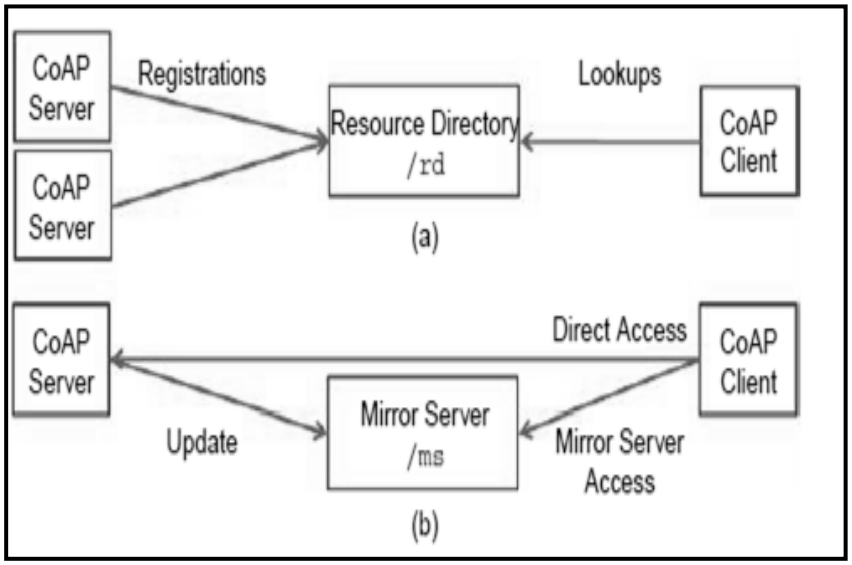


Figure 4.8: IETF CoRE Functional Components a) Resource Directory b) Mirror Server

The figure 4.9 The interworking issues appear when an HTTP Client accesses a CoAP Server through an HTTP-CoAP proxy or when a CoAP Client accesses an HTTP Server through a CoAP-HTTP proxy (Figure 4.9a).

The mapping process is not straightforward for a number of reasons. The main is the different transport protocols used by the HTTP and CoAP: HTTP uses TCP while CoAP uses UDP. The guidelines focus more on the HTTP-to-CoAP proxy and recommend addressing schemes (e.g. how to map a CoAP resource address to an HTTP address),mapping between HTTP and

CoAP response codes, mapping between different media types carried in the HTTP/CoAP payloads, etc.

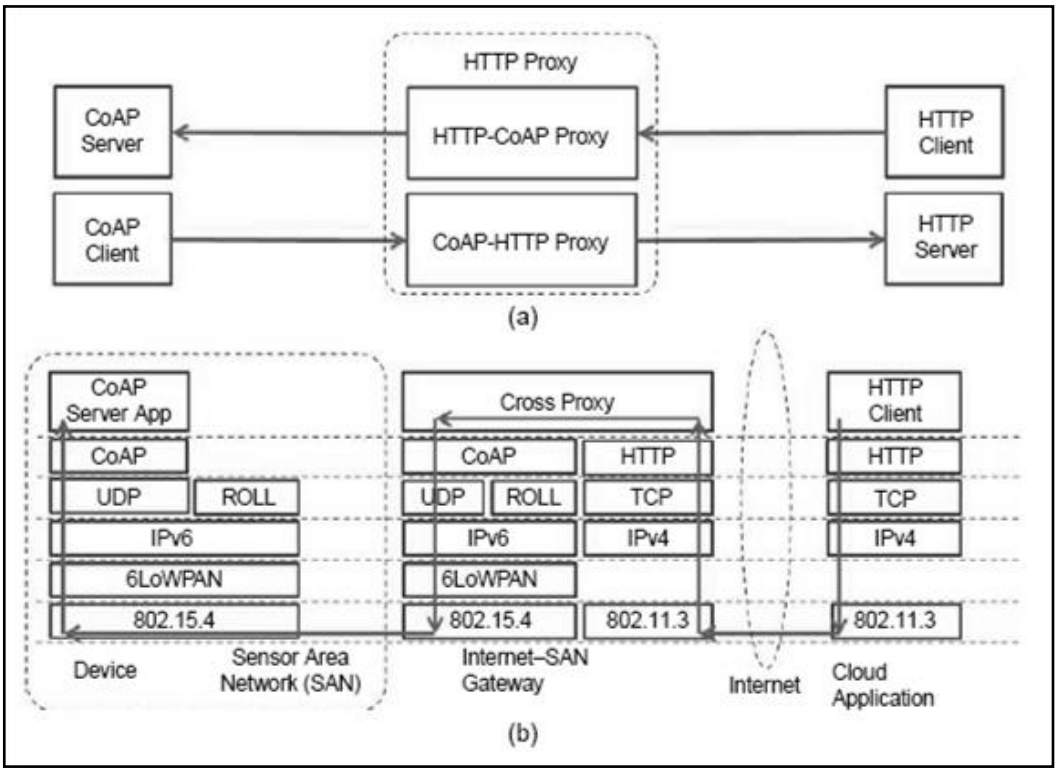


Figure 4.9: IETF CoRE HTTP Proxy

a) Possible configurations b) example layer interaction upon a request from HTTP client to a CoAP server via a HTTP Proxy.

- As an example, consider the case that an HTTP Client sends an HTTP request to a CoAP server (Figure 4.9a) through a Gateway Device hosting an HTTP-CoAP Cross Proxy.
- The Gateway Device connects to the Internet via an Ethernet cable using a LAN, and on the CoAP side the CoAP server resides on a Sensor/Actuator (SAN) based on the IEEE 802.15.4 PHY/MAC.
- The HTTP request needs to include two addresses, one for reaching the Cross Proxy and one for reaching the specific CoAP Server in the SAN. The default recommended address mapping is to append the CoAP resource address (e.g. coap://s.coap.example.com/foo)to the Cross proxy address (e.g. http://p.example.com/.well-known/core/), resulting in http://p.example.com/.well-known/core/coap://s.coap.example.com/foo.

- The request is in plain text format and contains the method (GET). It traverses the IPv4 stack of the client, reaches the gateway, traverses the IPv4 stack of the gateway and reaches the Cross proxy.
- The request is translated to a CoAP request (binary format) with a destination CoAP resource coap://s.coap.example.com/foo, and it is dispatched in the CoAP stack of the gateway, which sends it over the SAN to the end device.A response is sent from the end device and follows the reverse path in the SAN order to reach the gateway.

4.2.4 Open Geospatial Consortium architecture

The Open Geospatial Consortium (OGC 2013) is an international industry consortium of a few hundred companies, government agencies, and universities that develops publicly available standards that provide geographical information support to the Web, and wireless and location-based services.

The functionality that is targeted by OGC SWE includes:

• Discovery of sensor systems and observations that meet an application's criteria.

• Discovery of a sensor's capabilities and quality of measurements.

• Retrieval of real-time or time-series observations in standard encodings.

• Tasking of sensors to acquire observations.

• Subscription to, and publishing of, alerts to be issued by sensors or sensor services based upon certain criteria.

OGC SWE includes the following standards:

- SensorML and Transducer Model Language (TML), which include a model and an XML schema for describing sensor and actuator systems and processes; for example, a system that contains a temperature sensor measuring temperature in Celsius, which also involves a process for converting this measurement to a measurement with Fahrenheit units.
- Observations and Measurements (O&M), which is a model and an XML schema for describing the observations and measurements for a sensor (Observations and Measurements, O&M).
- SWE Common Data model for describing low-level data models (e.g. serialization in XML) in the messages exchanged between OGC SWE functional entities.

- Sensor Observation Service (SOS), which is a service for requesting,filtering, and retrieving observations and sensor system information.This is the intermediary between a client and an observation repository or near real-time sensor channel.
- Sensor Planning Service (SPS), which is a service for applications requesting a user-defined sensor observations and measurements acquisition. This is the intermediary between the application and a sensor collection system.
- PUCK, which defines a protocol for retrieving sensor metadata for serial port (RS232) or Ethernet-enabled sensor devices.
- An example of how these standards relate to each other is shown in Figure 4.10. Because OGC follows the SOA paradigm, there is a registry (CAT) that maintains the descriptions of the existing OGC services ,including the Sensor Observation and Sensor Planning Services. Upon installation the sensor system using the PUCK protocol retrieves the SensorML description of sensors and processes, and registers them with the Catalog so as to enable the discovery of the sensors and processes by client applications. The Sensor System also registers to the SOS and the SOS registers to the Catalog. A client application #1 requests from the Sensor Planning Service that the Sensor System be tasked to sample its

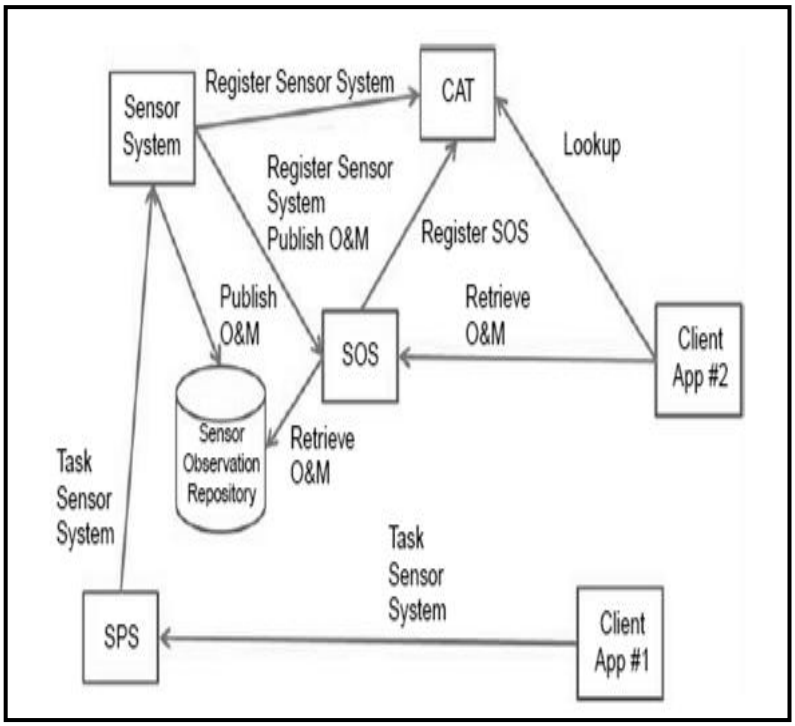


Figure 4.10: OGC functional architecture and interactions.

4.3 Introduction

This chapter provides an overview of the Architecture Reference Model (ARM) for IoT, including descriptions of the domain, information, and functional models. Chapter 8 then outlines the Reference Architecture.

An ARM consists of two main parts: a Reference model and Reference architecture. The foundation of an IoT Reference Architecture description is an IoT reference model. A reference model describes the domain using a number of sub-models (**Figure 4.11**). The domain model of an architecture model captures the main concepts or entities in the domain in question, in this case M2M and IoT. When these common language references are established , the domain model adds descriptions about the relationship between the concepts.

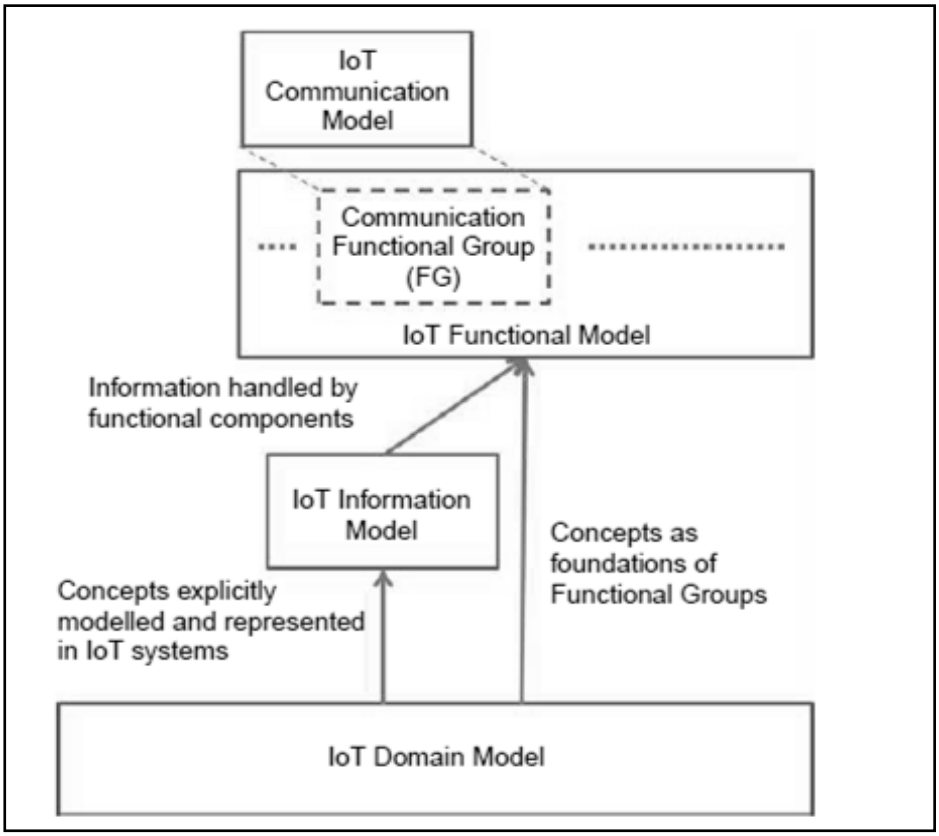


Figure 4.11: IoT Reference Model.

These concepts and relationships serve the basis for the development of an information model because a working system needs to capture and process information about its main entities and their interactions.

A working system that captures and operates on the domain and information model contains concepts and entities of its own, and these needs to be described in a separate model, the functional model. An M2M and IoT system contain communicating entities, and therefore the

corresponding communication model needs to capture the communication interactions of these entities. These are a few examples of sub-models that we use in this chapter for the IoT reference model.

Apart from the reference model, the other main component of an ARM is the Reference Architecture. A System Architecture is a communication tool for different stakeholders of the system. Developers, component and system managers, partners, suppliers, and customers have different views of a single system based on their requirements and their specific interactions with the system.

The task becomes more complex when the architecture to be described is on a higher level of abstraction compared with the architecture of real functioning systems. The high-level abstraction is called Reference Architecture as it serves as a reference for generating concrete architectures and actual systems, as shown in the Figure 4.12

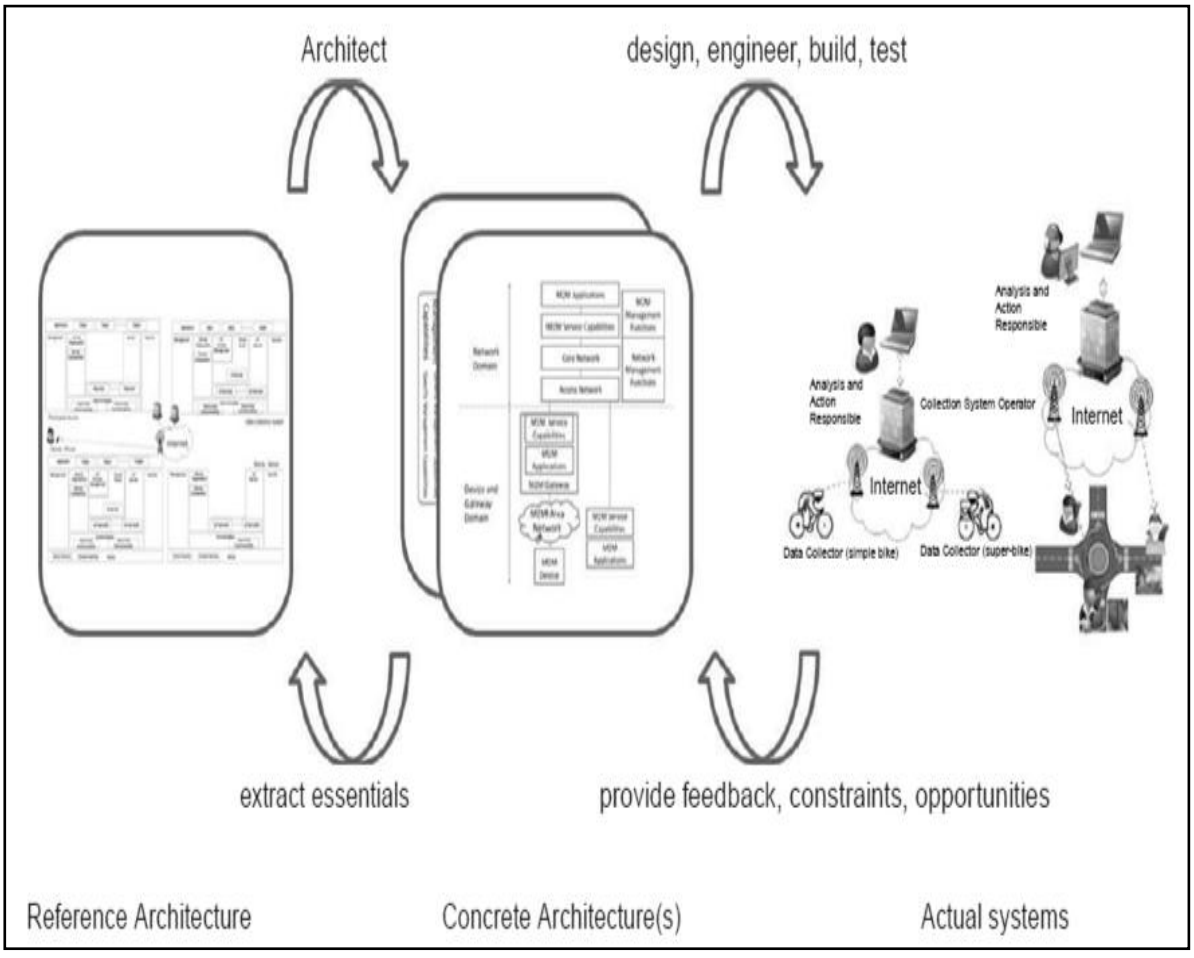


Figure 4.12: From reference to concrete architectures and actual systems.

- Concrete architectures are instantiations of rather abstract and high-level Reference Architectures.
- A Reference Architecture captures the essential parts of an architecture, such as design principles, guidelines, and required parts (such as entities), to monitor and interact with the physical world for the case of an IoT Reference Architecture.
- A concrete architecture can be further elaborated and mapped into real world components by designing, building, engineering, and testing the different components of the actual system. As the figure implies, the whole process is iterative, which means that the actual deployed system in the field provides invaluable feedback with respect to the design and engineering choices, current constraints of the system, and potential future opportunities that are fed back to the concrete architectures. The general essentials out of multiple concrete architectures can then be aggregated, and contribute to the evolution of the Reference Architecture.
- The IoT architecture model is related to the IoT Reference Architecture as shown in Figure 4.13. This figure shows two facets of the IoT ARM: (a)how to actually create an IoT ARM, and (b) how to use it with respect to building actual systems.

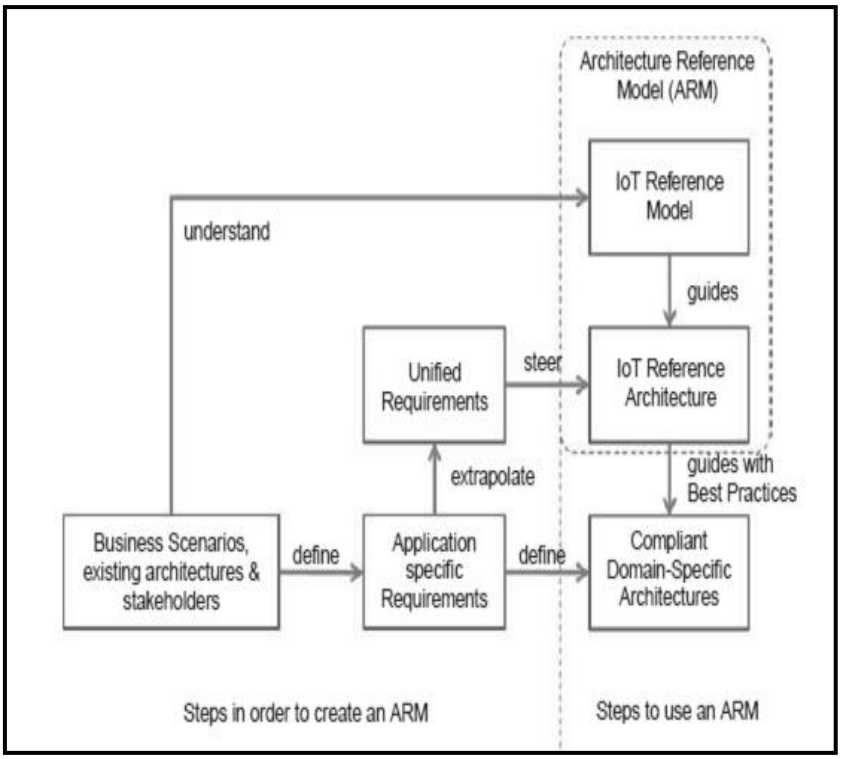


Figure 4.13: IoT Reference Model and Reference Architecture dependencies.

4.4 IOT REFERENCE MODEL

4.4.1 IOT DOMAIN MODEL

- The domain model captures the basic attributes of the main concepts and the relationship between these concepts.
- A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.
- A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.

4.4.1.1 Model notation and semantics

- For the purposes of the description of the domain model, we use the Unified Modeling Language (UML).Class diagrams in order to present the relationships between the main concepts of the IoT domain model.
- The Class diagrams consist of boxes that represent the different classes of the model connected with each other through typically continuous lines or arrows, which represent relationships between the respective classes.
- Each class is a descriptor of a set of objects that have similar structure, behavior, and relationships.
- A class contains a name (e.g. Class A in Figure 4.14) and a set of attributes and operations.

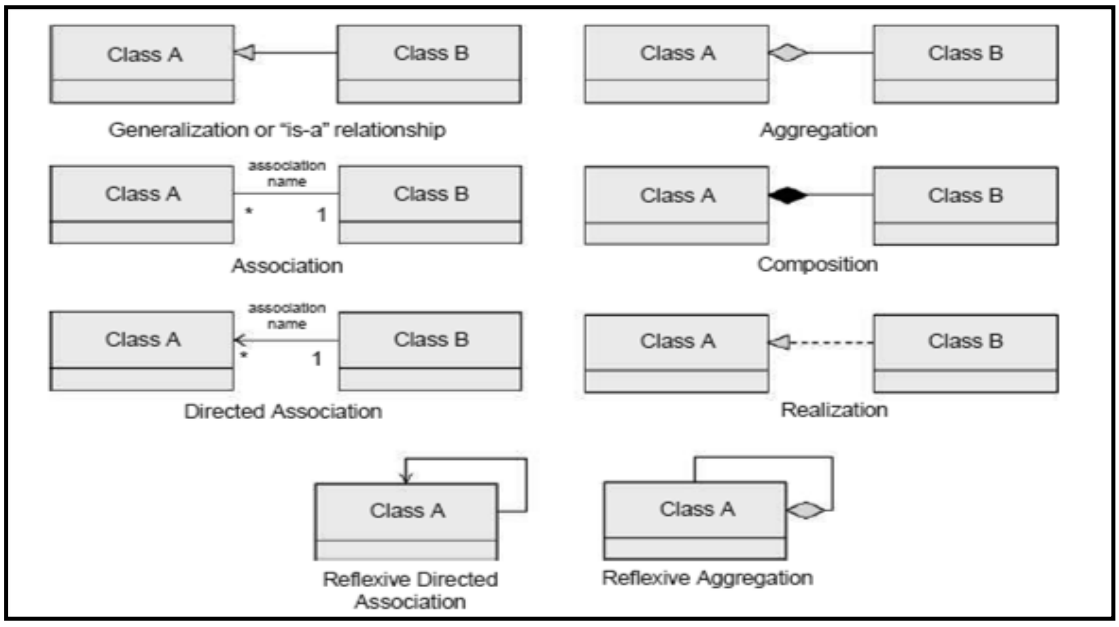


Figure 4.14: UML Class diagram main modeling concepts.

- For the description of the IoT domain model, we will use only the class name and the class attributes, and omit the class operations.
- Notation-wise this is represented as a box with two compartments, one containing the class name and the other containing the attributes. However, for the IoT domain model description, the attribute compartment will be empty in order not to clutter the complete domain model.
- The following modeling relationships between classes (Figure 4.14) are needed for the description of the IoT Domain Model: Generalization/ Specialization, Aggregation and Reflexive Aggregation, Composition, Directed Association and Reflexive Directed Association, and Realization.
- The Generalization/Specialization relationship is represented by an arrow with a solid line and a hollow triangle head. Depending on the starting point of the arrow, the relationship can be viewed as a generalization or specialization.
- The Aggregation relationship is represented by a line with a hollow diamond in one end and represents a whole-part relationship or a containment relationship and is often called a "has-a" relationship. The class that touches the hollow diamond is the whole class while the other class is the part class.
- The Composition relationship is represented by a line with a solid black diamond in one end, and also represents a whole-part relationship or a containment relationship. The class that touches the solid black diamond is the whole class while the other class is the part class.

4.4.1.2 Main concepts

The IoT is a support infrastructure for enabling objects and places in the physical world to have a corresponding representation in the digital world.The reason why we would like to represent the physical world in the digital world is to remotely monitor and interact with the physical world using software.

Let's illustrate this concept with an example (Figure 4.15).Imagine that we are interested in monitoring a parking lot with 16 parking spots. The parking lot includes a payment station for drivers to pay for the parking spot after they park their cars. The parking lot also includes an electronic road sign on the side of the street that shows in real-time the number of empty spots.

Frequent customers also download a smart phone application that informs them about the availability of a parking spot before they even drive on the street where the parking lot is

located. In order to realize such a service, the relevant physical objects as well as their properties need to be captured and translated to digital objects such as variables, counters, or database objects so that software can operate on these objects and achieve the desired effect, i.e. detecting when someone parks without paying, informing drivers about the availability of parking spots, producing statistics about the average occupancy levels of the parking lot, etc.

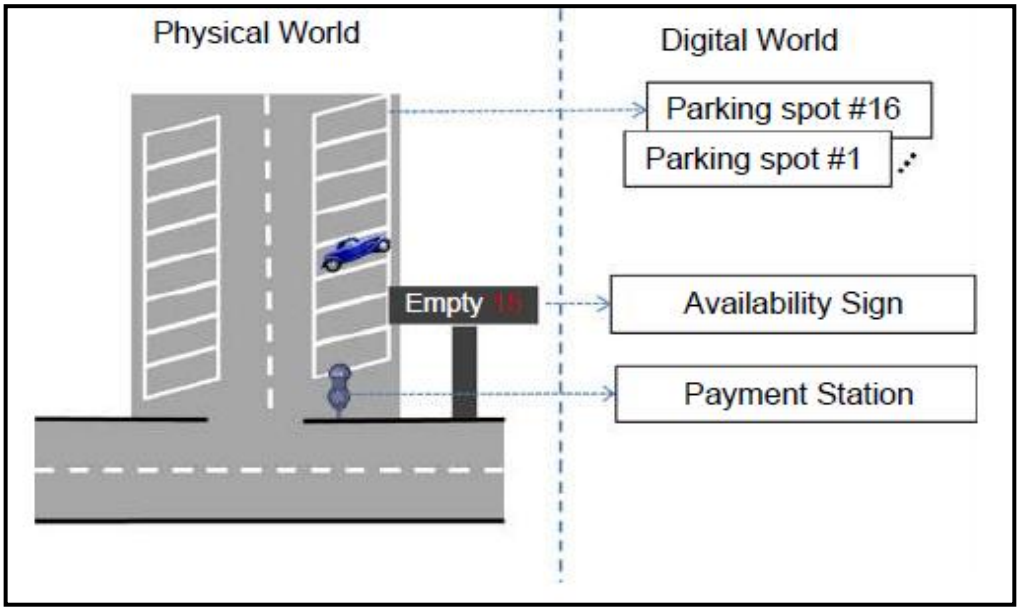


Figure 4.15: Physical vs. Virtual World.

For these purposes, the parking lot as a place is instrumented with parking spot sensors (e.g. loops), and for each sensor, a digital representation is created (Parking spot #1_#16). In the digital world, a parking spot is a variable with a binary value ("available" or "occupied"). The parking lot payment station also needs to be represented in the digital world in order to check if a recently parked car owner actually paid the parking fee. Finally, the availability sign is represented to the digital world in order to allow notification to drivers that an empty lot is full for maintenance purposes, or even to allow maintenance personnel to detect when the sign is malfunctioning.

As interaction with the physical world is the key for the IoT; it needs to be captured in the domain model (Figure 4.16). The first most fundamental interaction is between a human or an application with the physical A User can be a Human User, and the interaction can be physical (e.g. parking the car in the parking lot).

The physical interaction is the result of the intention of the human to achieve a certain goal (e.g. park the car). In other occasions, a Human world object or place. Therefore, a User and a Physical Entity are two concepts that belong to the domain model.

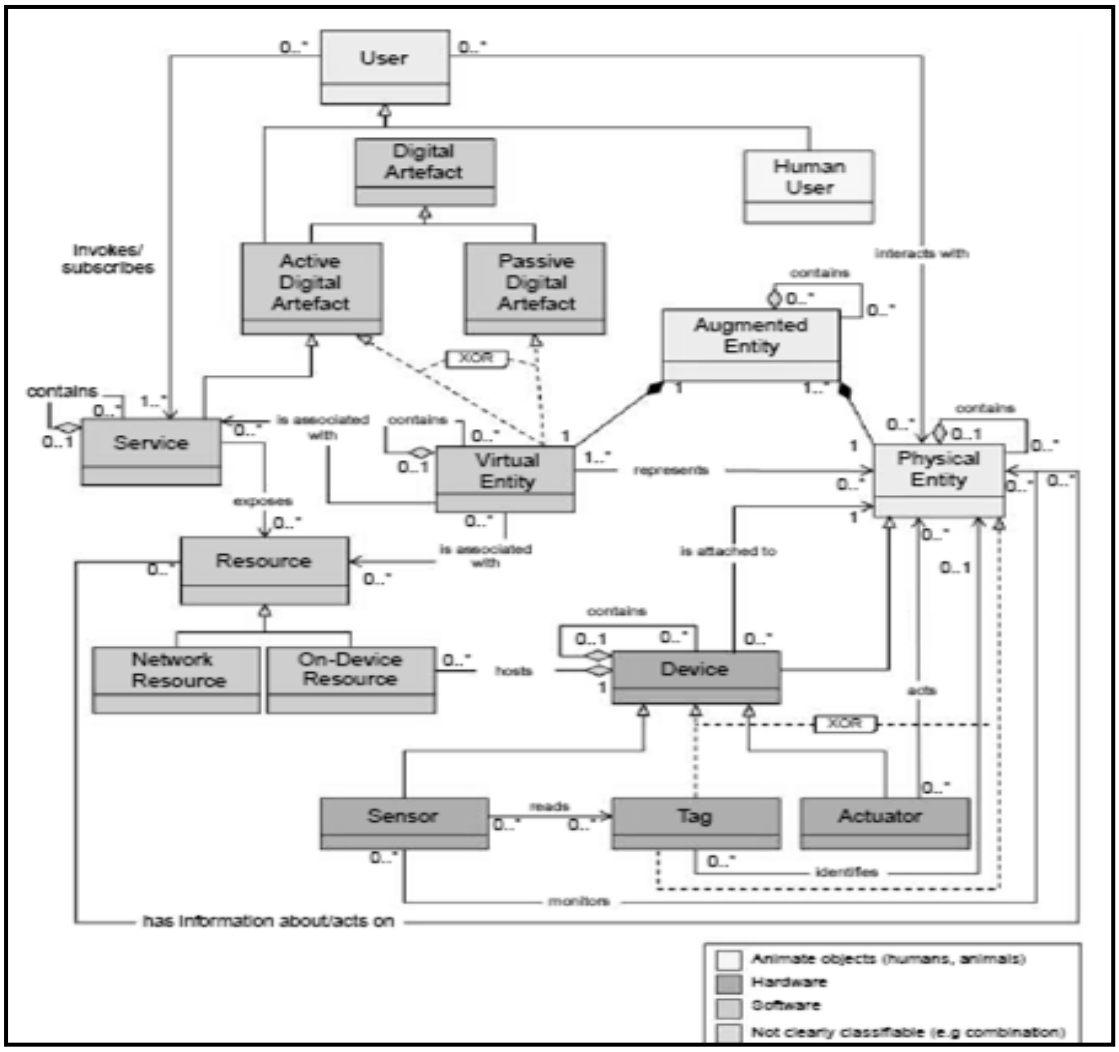


Figure 4.16: IoT Domain Model.

A Physical Entity, as the model shows, can potentially contain other physical entities; for example, a building is made up of several floors, and each floor has several rooms.

The objects, places, and things represented as Physical Entities are the same as Assets mentioned earlier in the book. According to the Oxford Dictionary, an Asset .is an item or property that is regarded as having value"; therefore, the term Asset is more related to the business aspects of IoT. Because the domain model is a technical tool, we use the term Physical Entity instead of Asset.

- A Physical Entity is represented in the digital world as a Virtual Entity.
- A Virtual Entity can be a database entry, a geographical model (mainly for places), an image or avatar, or any other Digital Artifact.
- One Physical Entity can be represented by multiple Virtual Entities, each serving a different purpose, e.g. a database entry of a parking spot denoting the spot

availability,and an (empty/full) image of a parking spot on the monitor of the parking lot management system.

- Each Virtual Entity also has a unique identifier for making it addressable among other Digital Artifacts. A Virtual Entity representation contains several attributes that correspond to the Physical Entity current state.
- The Virtual Entity representation and the Physical Entity actual state should be synchronized whenever a User operates on one or the other, if of course that is physically possible.

For the IoT Domain Model, three kinds of Device types are the most important:

1. Sensors: These are simple or complex Devices that typically involve a transducer that converts physical properties such as temperature into electrical signals. These Devices include the necessary conversion of analog electrical signals into digital signals, e.g. a voltage level to a 16-bit number, processing for simple calculations, potential storage for intermediate results, and potentially communication capabilities to transmit the digital representation of the physical property as well receive commands.

2. Actuators: These are also simple or complex Devices that involve a transducer that converts electrical signals to a change in a physical property (e.g. turn on a switch or move a motor). These Devices also include potential communication capabilities, storage of intermediate commands, processing, and conversion of digital signals to analog electrical signals.

3. Tags: Tags in general identify the Physical Entity that they are attached to. In reality, tags can be Devices or Physical Entities but not both, as the domain model shows. An example of a Tag as a Device is a Radio Frequency Identification (RFID) tag, while a tag as a Physical Entity is a paper-printed immutable barcode or Quick Response (QR) code.

- As shown in the model, Devices can be aggregation of other Devices e.g. a sensor node contains a temperature sensor, a Light Emitting Diode (LED, actuator), and a buzzer (actuator). Any type of IoT Device needs to (a) have energy reserves (e.g. a battery), or (b) be connected to the power grid, or (c) perform energy scavenging (e.g. converting solar radiation to energy).
- The Device communication, processing and storage, and energy reserve capabilities determine several design decisions such as if the resources should be on-Device or not.

- Resources are software components that provide data for, or are endpoints for, controlling Physical Entities. Resources can be of two types,on-Device resources and Network Resources. An on-Device Resource is typically hosted on the Device itself and provides information, or is the control point for the Physical Entities that the Device itself is attached to. The Network Resources are software components hosted somewhere in the network or cloud.
- A Virtual Entity is associated with potentially several Resources that provide information or control of the Physical Entity represented by this Virtual Entity.
- The Virtual Entities that are associated with Physical Entities instrumented with Devices that expose Resources are also associated with the corresponding resource Services.

It is important to note that IoT Services can be classified into three main classes according to their level of abstraction:

1. Resource-Level Services typically expose the functionality of a Device by exposing the on-Device Resources. In addition, these services typically handle quality aspects such as security, availability, and performance issues.
2. Virtual Entity-Level Services provide information or interaction capabilities about Virtual Entities, and as a result the Service interfaces typically include an identity of the Virtual Entity.
3. Integrated Services are the compositions of Resource-Level and Virtual Entity-Level services, or any combination of both service classes.

Instantiation of the IoT Domain Model

An example of an instantiation of the IoT Domain Model is as shown in figure 4.17.

For this instantiation, we use the example of a simple parking lot management system presented earlier, and we model only part of the real system.

For example the part of the model that captures the Loop Sensor #21_#28 and the associated Physical and Virtual Entities is similar to the corresponding part of the model for the Loop Sensor #11_#18 and therefore omitted. We assume that each parking spot is instrumented with a metal sensing loop (Sensor), and half of the loops are physically wired to one sensor node (Device, Sensor Node #1) while the rest are wired to another sensor node (Device, Sensor Node #2). The sensor nodes may have different identifiers (e.g. Sensor #11_Sensor #18 for the first group, and sensor #21-sensor #28 for the second group.

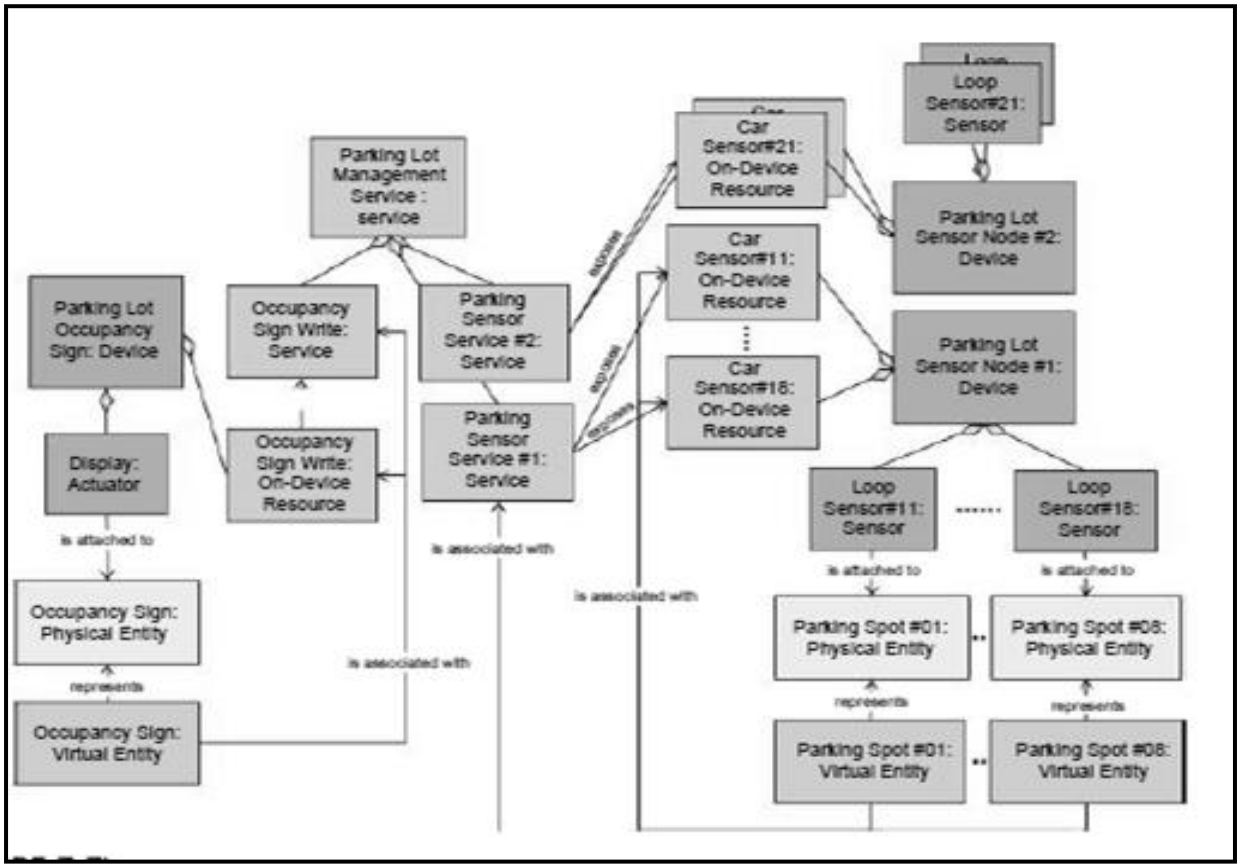


Figure 4.17: IoT Domain Instantiation

The loop sensors can output different impedance based on the existence or the absence of a steel object. These impedances are translated by the sensor nodes into binary "0" and "1" readings.

- Each Parking Lot Sensor Node hosts as many Car Sensor Resources as assigned parking spots. There are also two Parking Sensor Services each running on a Sensor Node. The Parking Sensor Service #1
- provides the reading of the Loop Sensor #11_#18, while the Parking Sensor Service #2 provides the readings of the Loop Sensor #21_#28.
- The Parking Lot Management Service has the necessary logic to map the sensor node readings to the appropriate occupancy indicator (e.g. "0" - "free","1"-"occupied") and maps the parking spot sensor identifier to the corresponding parking spot identifier (e.g. Sensor #11_Sensor #18 – Spot #01_Spot #08, and Sensor #21_Sensor #28 - Spot #09_Spot #16).
- The Virtual Entities that represent the Parking Spot Physical Entities are database entries with the following attributes: (a) identity (e.g. ID #1_#16), (b) physical The occupancy sign consists of a Device that contains a Display (Actuator) that is attached

to a Physical Entity (the actual steel sign). The Device exposes one Resource with one Service that allows writing a value to the sign display. The actual steel sign (Physical Entity) is represented in the digital world with a Virtual Entity, which is a database entry with the following attributes: (a) Location of the sign (e.g. GPS location),(b) status (on/off), and (c) display value (e.g. 15 free spaces). The Parking Lot Management System is a composed Service that contains the parking spot occupancy services and the occupancy sign write service.Internally given the occupancy status of all the parking spots, it produces the total number of free spots and uses this attribute to update the actuator attached to the occupancy sign. dimensions (e.g. 3m32m), (c) the location of the center of the rectangular spot with respect to the parking lot entrance (e.g. 3 meters to the west and 2 meter to the north), and (d) its occupancy level (e.g.“occupied” or “free”).

4.5 Information Model

Similar to the IoT Domain Model, the IoT Information Model is presented using Unified Modeling Language (UML) diagrams. As mentioned earlier, each class in a UML diagram contains zero or more attributes.These attributes are typically of simple types such as integers or text strings, and are represented with red text under the name of the class (e.g.entityType in the Virtual Entity class in Figure 4.18).

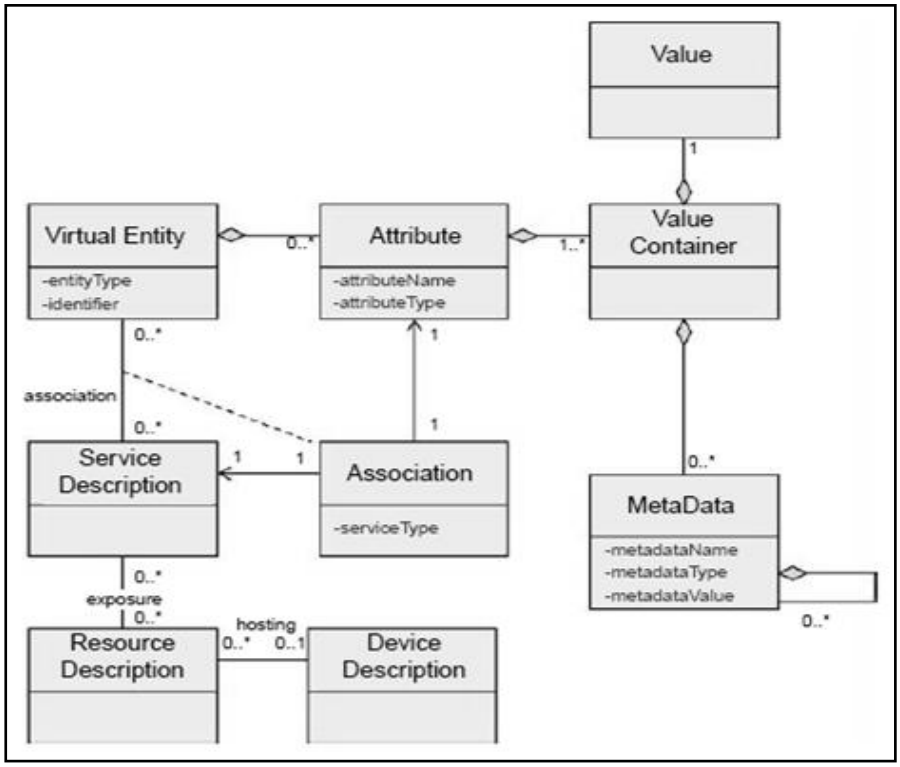


Figure 4.18: High –level Information Model

High level information model

- A more complex attribute for a specific class A is represented as a class B, which is contained
- in class A with an aggregation relationship between class A and class B.Moreover, the UML diagram for describing the IoT Information Model contains additional notation not presented earlier.
- On a high-level, the IoT Information Model maintains the necessary information about Virtual Entities and their properties or attributes. These properties/attributes can be static or dynamic and enter into the system in various forms, e.g. by manual data entry or reading a sensor attached to the Virtual Entity.
- Virtual Entity attributes can also be digital synchronized copies of the state of an actuator as mentioned earlier: by updating the value of an Virtual Entity attribute, an action takes place in the physical world.
- In the presentation of the high-level IoT information model, we omit the attributes that are not updated by an IoT Device (sensor, tag) or the attributes that do not affect any IoT Device.

IoT information Model example

The IoT Information Model describes Virtual Entities and their attributes that have one or more values annotated with meta-information or metadata. The attribute values are updated as a result of the associated services to a Virtual Entity. The associated services, in turn, are related to Resources and Devices as seen from the IoT Domain Model.

A Virtual Entity object contains simple attributes/properties:

(a)entityType to denote the type of entity, such as a human, car, or room (the entity type can be a reference to concepts of a domain ontology, e.g. a car ontology);

(b) a unique identifier; and

(c) zero or more complex attributes of the class Attributes.

The class Attributes should not be confused with the simple attributes of each class. This class Attributes is used as a grouping mechanism for complex attributes of the Virtual Entity. Objects of the class Attributes, in turn, contain the simple attributes with the self-descriptive names attributeName and attributeType.

As seen from the IoT Domain Model, a Virtual Entity is associated with Resources that expose Services about the specific Virtual Entity. This association between a Virtual Entity

and its Services is captured in the Information Model with the explicit class called Association.

Because the class Association describes the relationship between a Virtual Entity and Service Description through the Attribute class, there is a dashed line between Association class and the line between the Virtual Entity and Service Description classes.

The attribute serviceType can take two values:

(a) "INFORMATION," if the associated service is a sensor service (i.e.allows reading of the sensor), or (b) "ACTUATION," if the associated service is an actuation service

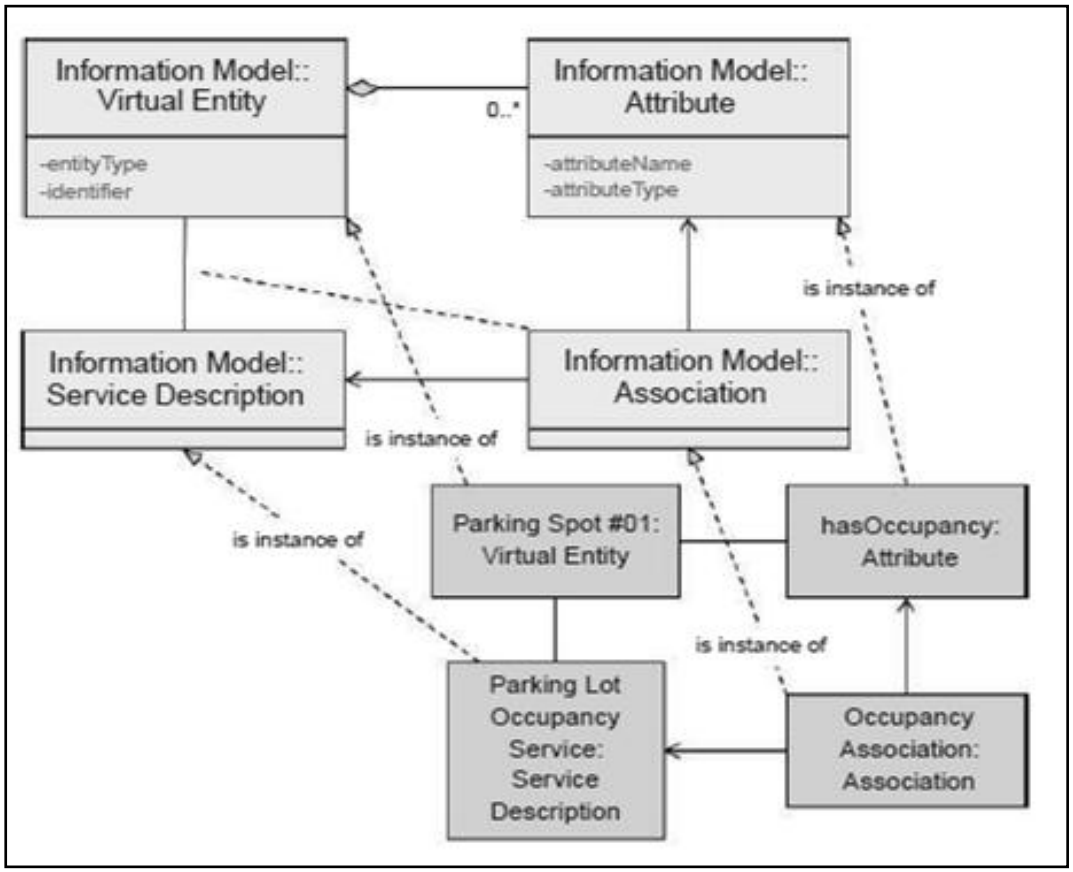


Figure 4.19 : IoT Information Model example

An example of an instantiation of the high-level information model is shown in **Figure 4.19** following the parking lot example presented earlier.

Here we don't show all the possible Virtual Entities, but only one corresponding to one parking spot. This Virtual Entity is described with one Attribute (among others) called hasOccupancy. This Attribute is associated with the Parking Lot Occupancy Service Description through the Occupancy Association. The Occupancy Association is the explicit expression of the association (line) between the Parking Spot #1 Virtual Entity and the Parking Lot Occupancy Service. model, as opposed to the Realization relationship for the IoT Domain Model.

Relationship between IoT Information and IoT Domain Model

- The Figure 4.20 represents the relationship between the core concepts of the IoT Domain Model and the IoT Information Model. The Information Model captures the Virtual Entity in the Domain Model being the "Thing" in the Internet of Things as several associated classes (Virtual Entity, Attribute, Value, MetaData, Value Container) that basically capture the description of a Virtual Entity and its context.
- The Device, Resource, and Service in the IoT Domain Model are also captured in the IoT Information Model because they are used as representations of the instruments and the digital interfaces for interaction with the Physical Entity associated with the Virtual Entity.
- The Information Model is a very high-level model,and omits certain details that could potentially be required in a concrete architecture and an actual system.

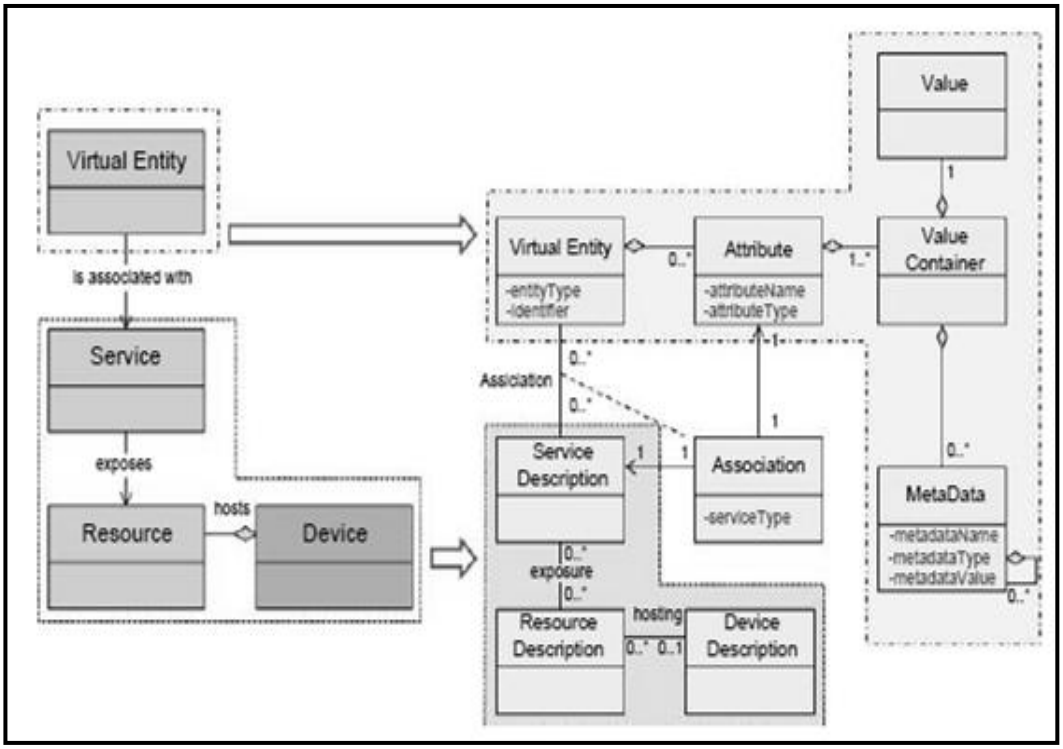


Figure 4.20: Relationship between core concepts of IoT Domain Model and IoT information Model

As mentioned earlier, there are several other attributes or properties that could exist in a Virtual Entity description:

- Location and its temporal information are important because Physical Entities represented by Virtual Entities exist in space and time. These properties are extremely important when the interested Physical Entities are mobile (e.g. a moving car).

- Even non-moving Virtual Entities contain properties that are dynamic with time, and therefore their temporal variations need to be modelled and captured by an information model.
- Information such as ownership is also important in commercial settings because it may determine access control rules or liability issues.

The Services in the IoT Domain Model are mapped to the Service Description in the IoT Information Model.

- Service type, which denotes the type of service, such as Big Web Service or RESTful Web Service. The interfaces of a service are described based on the description language for each service type, for example, Web Application Description Language (WADL) for RESTful Web Services, Web Services Description Language (WSDL).
- Service area and Service schedule are properties of Services used for specifying the geographical area of interest for a Service and the potential temporal availability of a Service, respectively.
- Associated resources that the Service exposes.
- Metadata or semantic information used mainly for service composition. This is information such as the indicator of which resource property is exposed as input or output, whether the execution of the service needs any conditions satisfied before invocation, and whether there are any effects of the service after invocation.

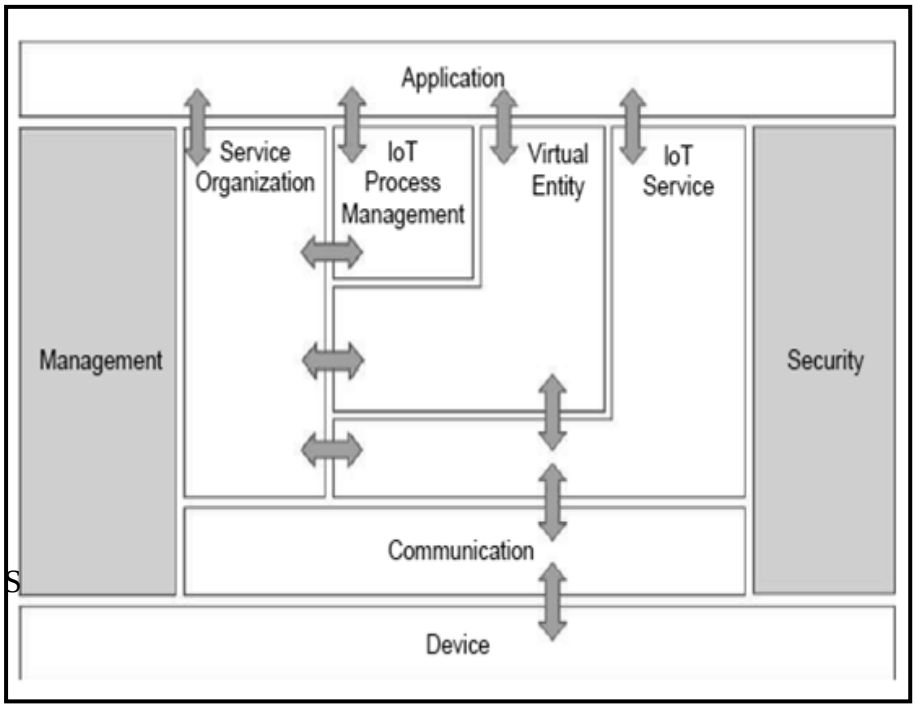
The IoT Information Model also contains Resource descriptions because Resources are associated with Services and Devices in the IoT Domain model. A Resource description contains the following information:

1. Resource name and identifier for facilitating resource discovery.
2. Resource type, which specifies if the resource is (a) a sensor resource,which provides sensor readings; (b) an actuator resource, which provides actuation capabilities (to affect the physical world) and actuator state; (c)a processor resource, which provides processing of sensor data and output of processed data; (d) a storage resource, which provides storage of data about a Physical Entity; and (e) a tag resource, which provides identification data for Physical Entities.
3. Free text attributes or tags used for capturing typical manual input such as “fire alarm, ceiling.”
4. Indicator of whether the resource is an on-Device resource or network resource.

5. Location information about the Device that hosts this resource in case of an on-Device resource.
6. Associated Service information.
7. Associated Device description information.

4.6 Functional model

The IoT Functional Model aims at describing mainly the Functional Groups (FG) and their interaction with the ARM, while the Functional View of a Reference Architecture describes the functional components of an FG,interfaces, and interactions between the components. The Functional View is typically derived from the Functional Model in conjunction with high-level requirements. The IoT-A Functional Model is as shown in figure 4.21



S

Figure 4.21 : IoT-A Functional Model

The Application, Virtual Entity, IoT Service, and Device FGs are generated by starting from the User, Virtual Entity, Resource, Service, and Device classes from the IoT Domain Model. The need for communicating Devices and digital artifacts was the motivation for the Communication FG.

- The need to compose simple IoT services in order to create more complex ones, as well as the need to integrate IoT services (simple or complex)with existing Information and Communications Technology (ICT) infrastructure,is the main driver

behind the introduction of the Service Organization and IoT Process Management FGs respectively.

- The figure shows the flow of information between FGs apart from the cases of the Management and Security FGs that have information flowing from/to all other FGs, but these flows are omitted for clarity purposes.

4.6.1: Device management group

The Device FG contains all the possible functionality hosted by the physical Devices that are used for instrumenting the Physical Entities. This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities.

4.6.2: Communication functional group

The Communication FG abstracts all the possible communication mechanisms used by the relevant Devices in an actual system in order to transfer information to the digital world components or other Devices. Examples of such functions include wired bus or wireless mesh technologies through which sensor Devices are connected to Internet Gateway Devices.

4.6.3 IoT Service functional group

The IoT Service FG corresponds mainly to the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).Support functions such as directory services, which allow discovery of Services and resolution to Resources, are also part of this FG.

4.6.4 Virtual Entity functional group

The Virtual Entity FG corresponds to the Virtual Entity class in the IoT Domain Model, and contains the necessary functionality to manage associations between Virtual Entities with themselves as well as associations between Virtual Entities and related IoT Services, i.e. the Association objects for the IoT Information Model.

Associations between Virtual Entities can be static or dynamic depending on the mobility of the Physical Entities related to the corresponding Virtual Entities.

A major difference between IoT Services and Virtual Entity Services is the semantics of the requests and responses to/from these services.

4.6.5 IoT Service Organization functional group

The purpose of the IoT Service Organization FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services.

Moreover, this FG acts as a service hub between several other functional groups such as the IoT Process Management FG when, for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.

Simple IoT or Virtual Entity Services can be composed to create more complex services,e.g. a control loop with one Sensor Service and one Actuator service with the objective to control the temperature in a building.

4.6.6 IoT Process Management functional group

The IoT Process Management FG is a collection of functionalities that allows smooth integration of IoT-related services (IoT Services, Virtual Entity Services, Composed Services) with the Enterprise (Business) Processes.

4.6.7 Management Functional group

The Management FG includes the necessary functions for enabling fault and performance monitoring of the system, configuration for enabling the system to be flexible to changing User demands, and accounting for enabling subsequent billing for the usage of the system.

4.6.8 Security functional group

The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy. The Security FG contains components for Authentication of Users (Applications, Humans), Authorization of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, and last but not least, assurance of privacy of sensitive information relating to Human Users.

4.6.9 Application functional group

The Application FG is just a placeholder that represents all the needed logic for creating an IoT application. The applications typically contain custom logic tailored to a specific domain such as a Smart Grid.

4.6.10 Modular IoT functions

The Functional Model, as well as the Functional View of the Reference Architecture, contains a complete map of the potential functionalities for a system realization. The functionalities that will eventually be used in an actual system are dependent on the actual system requirements The bare minimum functionalities are Device, Communication, IoT

Services, Management, and Security (Figure 4.22a). With these functionalities, an actual system can provide access to sensors, actuators and tag services for an application or backend system of a larger Enterprise. The application or larger system parts have to build the Virtual Entity functions for capturing the information about the Virtual Entities or the "Things" in the IoT architecture.

Often the Virtual Entity concept is not captured in the application or a larger system with a dedicated FG, but functions for handling Virtual Entities are embedded in the application or larger system logic; therefore, in Figures 4.22a_c, the Virtual Entity is represented with dashed lines.

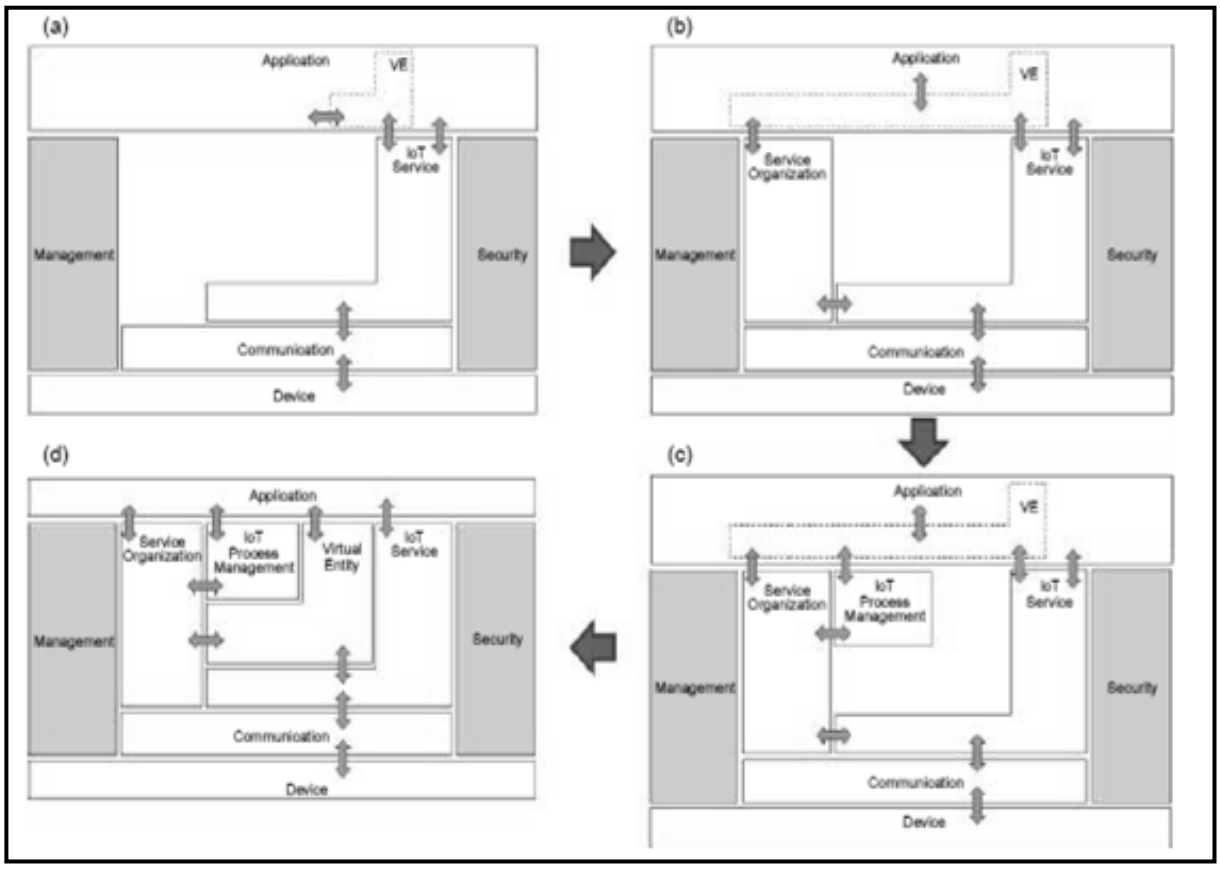


Figure 4.22: Building Progressively complex IoT Systems

4.7 Communication Model

The communication model for an IoT Reference Model consists of the identification of the endpoints of interactions, traffic patterns (e.g. unicast vs. multicast), and general properties of the underlying technologies used for enabling such interactions.

The potential communicating endpoints or entities are the Users,Resources, and Devices from the IoT Domain Model. Users include Human Users and Active Digital Artifacts (Services, internal system components,external applications).

Devices with a Human_Machine Interface mediate the interactions between a Human User and the physical world (e.g. keyboards, mice, pens, touch screens, buttons, microphones, cameras,eye tracking, and brain wave interfaces, etc.), and therefore the Human User is not a communication model endpoint.

The User (Active Digital Artifact, Service)-to-Service interactions include the User-to-Service and Service-to-Service interactions (in the case of an enterprise service/application accessing another service, or in the case of IoT service composition)as well as the Service_Resource_Device interactions. The Userto-Service and Service-to-Service communication is typically based on Internet protocols

4.8 Safety, privacy, trust, security model

An IoT system enables interactions between Human Users and Active Digital Artifacts (Machine Users) with the physical environment. The fact that Human Users are part of the system that could potentially harm humans if malfunctioning, or expose private information, motivates the Safety and Privacy needs for the IoT Reference Model and Architecture. The Trust and Security Model are needed in every ICT system with the objective to protect the digital world.

4.8.1 Safety

System safety is highly application- or application domain-specific, and is typically closely related to an IoT system that includes actuators that could potentially harm animate objects (humans, animals).

IoT-related guidelines for ensuring a safe system to the extent possible and controllable by a system designer. A system designer of such critical systems typically follows an iterative process with two steps: (a) identification of hazards followed by, (b) the mitigation plan.

This process is very similar to the threat model and mitigation plan that a security designer performs for an ICT system.

4.8.2 Privacy

The IoT-A Privacy Model depends on the following functional components: Identity Management, Authentication, Authorization, and Trust & Reputation

- Identity Management offers the derivation of several identities of different types for the same architectural entity with the objective to protect the original User identity for anonymization purposes.
- Authentication is a function that allows the verification of the identity of a User whether this is the original or some derived identity.

- Authorization is the function that asserts and enforces access rights when Users (Services, Human Users) interact with Services, Resources, and Devices.
- The Trust and Reputation functional component maintain the static or dynamic trust relationships between interacting entities.

4.8.3 Trust

A trust model is often coupled with the notion of trust in an ICT system, and represents the model of dependencies and expectations of interacting entities.

The necessary aspects of a trust model according to IoT-A as follows

- **Trust Model Domains:** Because ICT and IoT systems may include a large number of interacting entities with different properties, maintaining trust relationships for every pair of interacting entities may be prohibitive.Therefore, groups of entities with similar trust properties can define different trust domains.
- **Trust Evaluation Mechanisms:** These are well-defined mechanisms that describe how a trust score could be computed for a specific entity.The evaluation mechanism needs to take into account the source of information used for computing the trust level/score of an entity; two related aspects are the federated trust and trust anchor. A related concept is the IoT support for evaluation of the trust level of a Device, Resource,and Service.
- **Trust Behavior Policies:** These are policies that govern the behaviour between interacting entities based on the trust level of these interacting entities; for example, how a User could use sensor measurements retrieved by a Sensor Service with a low trust level.
- **Trust Anchor:** This is an entity trusted by default by all other entities belonging to the same trust model, and is typically used for the evaluation of the trust level of a third entity.
- **Federation of Trust:** A federation between two or more Trust Models includes a set of rules that specify the handling of trust relationships between entities with different Trust Models. Federation becomes important in large-scale systems.

4.9 IoT Reference Architecture

The Reference Architecture is a starting point for generating concrete architectures and actual systems.A Reference Architecture, on the other hand, serves as a guide for one or more concrete system architects. However, the concept of views for the presentation of an architecture is also useful for the IoT Reference Architecture.we have chosen to present the Reference Architecture as a set of architectural views.

- **Functional View:** Description of what the system does, and its main functions.
- **Information View:** Description of the data and information that the system handles.
- **Deployment and Operational View:** Description of the main real world components of the system such as devices, network routers,servers, etc.

4.10 Functional View

The functional view for the IoT Reference Architecture is presented in Figure 4.23,

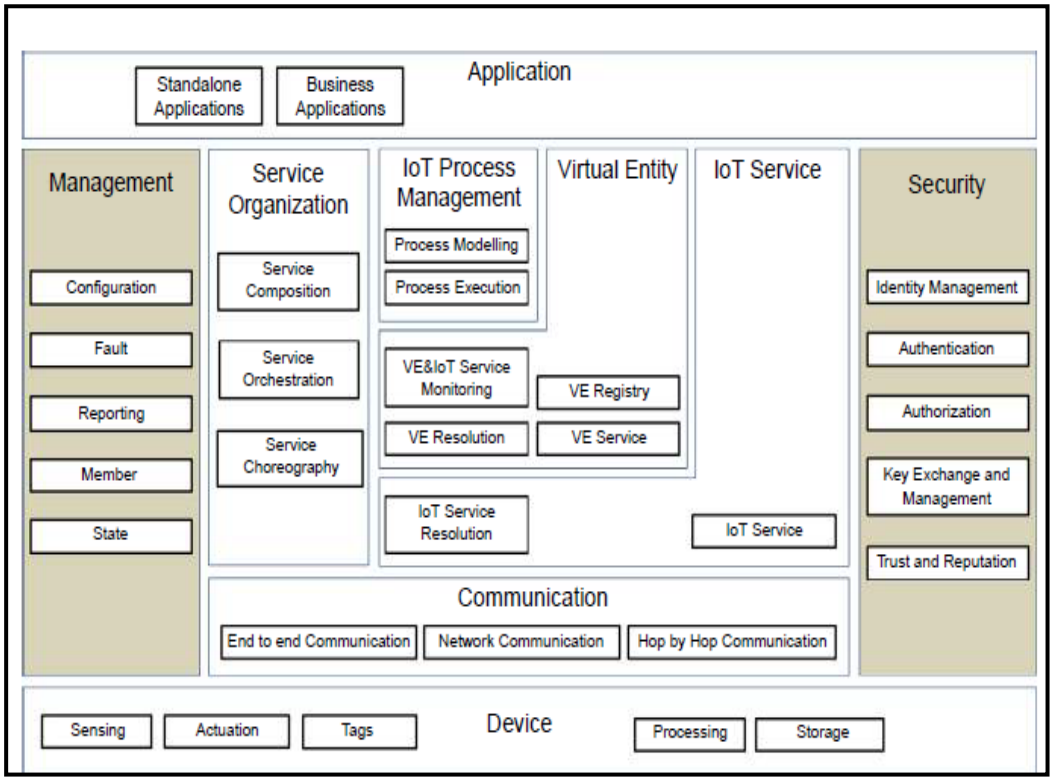


Figure 4.23: IoT Functional View

4.10.1 Device and Application functional group

Device FG contains the Sensing, Actuation, Tag, Processing, Storage FCs, or simply components. These components represent the resources of the device attached to the Physical Entities of interest. The Application FG contains either standalone applications (e.g.for iOS, Android, Windows phone), or Business Applications that connect the IoT system to an Enterprise system.

4.10.2 Communication Functional group

The Communication FG contains the End-to-End Communication, Network Communication, and Hop-by-Hop communication components:

- **The Hop-by-Hop Communication** is applicable in the case that devices are equipped with mesh radio networking technologies such as IEEE 802.15.4 for which messages have to traverse the mesh from node-to-node (hop-by-hop) until they reach a gateway

node which forwards the message (if needed) further to the Internet.The hop-by-hop FC is responsible for transmission and reception of physical and MAC layer frames to/from other devices. This FC has two main interfaces: (a) one "southbound" to/from the actual radio on the device, and (b) one "northbound" to/from the Network FC in the Communication FG.

- **The Network FC** is responsible for message routing & forwarding and the necessary translations of various identifiers and addresses.The translations can be (a) between network layer identifiers to MAC and/or physical network identifiers, (b) between high-level human readable host/node identifiers to network layer addresses (e.g. Fully Qualified Domain Names (FQDN) to IP addresses, a function implemented by a Domain Name System (DNS) server), and (c)translation between node/service identifiers and network locators in case the higher layers above the networking layer use node or service identifiers that are decoupled from the node addresses in the network. Finally, the Network FC is responsible for handling messages that cross different networking or MAC/PHY layer technologies, a function that is typically implemented on a network gateway type of device. The Network FC interfaces the End-to-End Communication FC on the "northbound" direction, and the Hop-by-Hop Communication FC on the "southbound" direction.
- **The End-to-End Communication FC** is responsible for end-to-end transport of application layer messages through diverse network and MAC/PHY layers. In turn, this means that it may be responsible for end-to-end retransmissions of missing frames depending on the configuration of the FC. Finally,this FC is responsible for hosting any necessary proxy/cache and any protocol translation between networks with different transport/application layer technologies.

4.10.3 IoT Service functional group

The IoT Service FG consists of two FCs: The IoT Service FC and the IoT Service Resolution FC:

- **The IoT Service FC** is a collection of service implementations, which interface the related and associated Resources. For a Sensor type of a Resource, the IoT Service FC includes Services that receive requests from a User and returns the Sensor Resource value in synchronous or asynchronous (e.g. subscription/notification) fashion. The services corresponding to Actuator Resources receive User requests for actuation,control the Actuator Resource, and may return the status of the Actuator after the action.

- **The IoT Service Resolution FC** contains the necessary functions to realize a directory of IoT Services that allows dynamic management of IoT Service descriptions and discovery/lookup/resolution of IoT Services by other Active Digital Artifacts.

  The Service descriptions of IoT Services contain a number of attributes as seen earlier in the IoT Functional Model section. Dynamic management includes methods such as creation/update/deletion (CUD) of Service description, and can be invoked by both the IoT Services themselves, or functions from the Management FG.

  - The discovery/lookup and resolution functions allow other Services or Active Digital Artifacts to locate IoT Services by providing different types of information to the IoT Service Resolution FC.
  - Service identifier (attribute of the Service description) a lookup method invocation to the IoT Service Resolution returns the Service description, while the resolution method invocation returns the contact information.

4.10.4 Virtual Entity Functional group

The Virtual Entity FG contains functions that support the interactions between Users and Physical Things through Virtual Entity services. An example of such an interaction is the query to an IoT system of the form, "What is the temperature in the conference room Titan?" The Virtual Entity is the conference room "Titan," and the conference room attribute of interest is "temperature."

The Following FCs are defined for realizing these functionalities:

- The Virtual Entity Service FC enables the interaction between Users and Virtual Entities by means of reading and writing the Virtual Entity attributes (simple or complex), which can be read or written, of course.Some attributes (e.g. the GPS coordinates of a room) are static and non-writable by nature, and some other attributes are non-writable by access control rules.In general attributes that are associated with IoT Services, which in turn represent Sensor Resources, can only be read.There can be, of course, special Virtual Entities associated with the same Sensor Resource through another IoT Service that allow write operations. Virtual Entity represents the Sensor device itself which in turn represent Actuator Resources, can be read and written. A read operation returns the actuator status, while a write operation results in a command sent to the actuator.
- The Virtual Entity Registry FC maintains the Virtual Entities of interest for the specific IoT system and their associations. The component offers services such as creating/reading/updating/deleting Virtual Entity descriptions and associations. Certain

associations can be static; for example, the entity "Room #123" is contained in the entity "Floor #7" by construction, while other associations are dynamic, e.g. entity "Dog" and entity "Living Room" due to at least Entity mobility. Update and Deletion operations take the Virtual Entity identifier as a parameter.

- The Virtual Entity Resolution FC maintains the associations between Virtual Entities and IoT Services, and offers services such as creating/reading/updating/deleting associations as well as lookup and discovery of associations. The Virtual Entity Resolution FC also provides notification to Users about the status of the dynamic associations between a Virtual Entity and an IoT Service, and finally allows the discovery of IoT Services provided the certain Virtual Entity attributes.
- The Virtual Entity and IoT Service Monitoring FC includes: (a) functionality to assert static Virtual Entity_IoT Service associations, (b)functionality to discover new associations based on existing associations or Virtual Entity attributes such as location or proximity, and (c) continuous monitoring of the dynamic associations between Virtual Entities and IoT Services and updates of their status in case existing associations are not valid any more.

4.10.5 IoT Process Management functional group

The IoT Process Management FG aims at supporting the integration of business processes with IoT-related services. It consists of two FCs:

- The Process Modeling FC provides that right tools for modeling a business process that utilizes IoT-related services.
- The Process Execution FC contains the execution environment of the process models created by the Process Modelling FC and executes the created processes by utilizing the Service Organization FG in order to resolve high-level application requirements to specific IoT services.

4.10.6 Service Organization Functional group

The Service Organization FG acts as a coordinator between different Services offered by the system. It consists of the following FCs:

- The Service Composition FC manages the descriptions and execution environment of complex services consisting of simpler dependent services. An example of a complex composed service is a service offering the average of the values coming from a number of simple Sensor Services. The complex composed service descriptions can

be wellspecified or dynamic/flexible depending on whether the constituent services are well-defined and known at the execution time or discovered on-demand.

- The Service Orchestration FC resolves the requests coming from IoT Process Execution FC or User into the concrete IoT services that fulfil the requirements.
- The Service Choreography FC is a broker for facilitating communication among Services using the Publish/Subscribe pattern. Users and Services interested in specific IoT-related services subscribe to the Choreography FC, providing the desirable service attributes even if the desired services do not exist.

4.10.7 Security Functional Group

- The Security FG contains the necessary functions for ensuring the security and privacy of an IoT system. It consists of the following FCs:
- The Identity Management FC manages the different identities of the involved Services or Users in an IoT system in order to achieve anonymity by the use of multiple pseudonyms.
- The Authentication FC verifies the identity of a User and creates an assertion upon successful verification. It also verifies the validity of a given assertion.
- The Authorization FC manages and enforces access control policies.It provides services to manage policies (CUD), as well as taking decisions and enforcing them regarding access rights of restricted resources.
- The Key Exchange & Management is used for setting up the necessary security keys between two communicating entities in an IoT system.This involves a secure key distribution function between communicating entities.
- The Trust & Reputation FC manages reputation scores of different interacting entities in an IoT system and calculates the service trust levels. A more detailed description of this FC is contained in the Safety,Privacy, Trust, Security Model.

4.10.8 Management Functional Group

The Management FG contains system-wide management functions that may use individual FC management interfaces. It is not responsible for the management of each component, rather for the management of the system as a whole.

It consists of the following FCs:

- The Configuration FC maintains the configuration of the FCs and the Devices in an IoT system (a subset of the ones included in the Functional View). The component

collects the current configuration of all the FCs and devices, stores it in a historical database, and compares current and historical configurations.

- The component can also set the system-wide configuration (e.g. upon initialization), which in turn translates to configuration changes to individual FCs and devices.
- The Fault FC detects, logs, isolates, and corrects system-wide faults if possible. This means that individual component fault reporting triggers fault diagnosis and fault recovery procedures in the Fault FC.
- The Member FC manages membership information about the relevant entities in an IoT system. Example relevant entities are the FGs, FCs, Services, Resources, Devices, Users and Applications. Services, Resources, Devices, Users, and Applications.
- The State FC is similar to the Configuration FC, and collects and logs state information from the current FCs, which can be used for fault diagnosis, performance analysis and prediction, as well as billing purposes. This component can also set the state of the other FCs based on system-wise state information.
- The Reporting FC is responsible for producing compressed reports about the system state based on input from FCs.

4.11 Information View

The information view consists of (a) the description of the information handled in the IoT System, and (b) the way this information is handled in the system; in other words, the information lifecycle and flow (how information is created, processed, and deleted), and the information handling components

4.11.1 Information Description

The pieces of information handled by an IoT system complying to an ARM such as the IoT-A.

- Virtual Entity context information, i.e. the attributes (simple or complex)as represented by parts of the IoT Information model (attributes that have values and metadata such as the temperature of a room). This is one of the most important pieces of information that should be captured by an IoT system, and represents the properties of the associated Physical Entities or Things.
- IoT Service output itself is another important part of information generated by an IoT system. For example, this is the information generated by interrogating a Sensor or a Tag Service.

- Virtual Entity descriptions in general, which contain not only the attributes coming from IoT Devices (e.g. ownership information).Associations between Virtual Entities and related IoT Services.
- Virtual Entity Associations with other Virtual Entities (e.g. Room #123 is on Floor #7).
- IoT Service Descriptions, which contain associated Resources, interface descriptions, etc.
- Resource Descriptions, which contain the type of resource (e.g. sensor),identity, associated Services, and Devices.
- Device Descriptions such as device capabilities (e.g. sensors, radios).
- Descriptions of Composed Services, which contain the model of how a complex service is composed of simpler services.
- IoT Business Process Model, which describes the steps of a business process utilizing other IoT-related services (IoT, Virtual Entity,Composed Services).
- Security information such as keys, identity pools, policies, trust models, reputation scores, etc.
- Management information such as state information from operational FCs used for fault/performance purposes, configuration snapshots, reports, membership information Etc.

4.11.2 Information flow and lifecycle

- From devices that produce information such as sensors and tags, information follows a context-enrichment process until it reaches the consumer application or part of the larger system, and from the application or part of larger system information it follows a context-reduction process until it reaches the consumer types of devices (e.g. actuators). The enrichment process is shown in Figure 4.24.
- Devices equipped with sensors transform changes in the physical properties of the Physical Entities of Interest into electrical signals.
- These electrical signals are transformed in one or multiple values (Figure 4.24a) on the device level. These values are then enriched with metadata information such as units of measurement, timestamp, and possibly location information (Figure 4.24b). These enriched values are offered by a software component (Resource) either on the device or the network. The Resource exposes certain IoT Services to formalize access to this enriched information (Figure 4.24c).
- At this point,the information is annotated with simple attributes such as location and time,

and often this type of metadata is sufficient for certain IoT applications or for the use in certain larger systems.This enriched information becomes context information as soon as it is further associated with certain Physical Entities in the form of Virtual Entity attributes (simple or complex, static or dynamic).

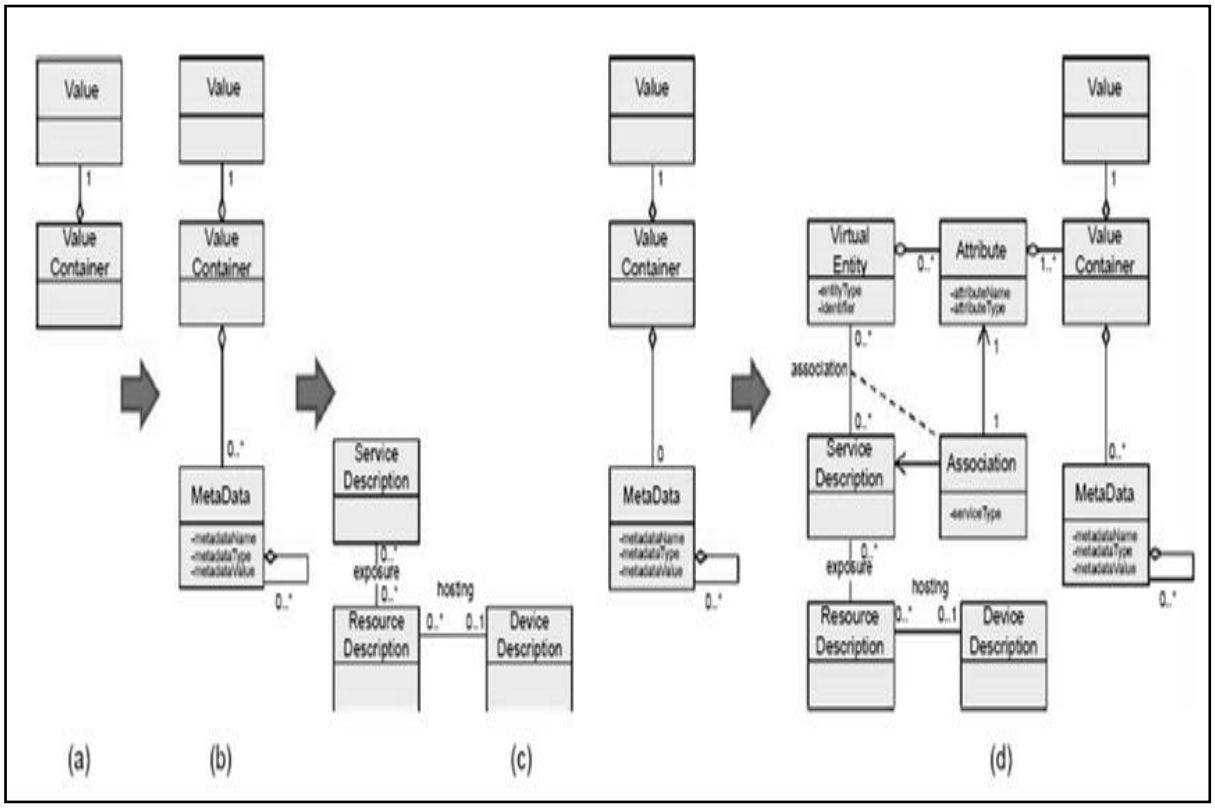


Figure 4.24: Information Exchange Patterns

- Actionable information flows into business processes that implement an action plan. Action plans push context information about Virtual Entities to associated IoT Services, to corresponding Actuation Resources, and finally to the real actuators that perform the changes in the physical world .
- Virtual Entity context information is typically generated by data-producing devices such as sensor devices and consumed either by data-consumption devices such as actuators or services.
- Raw or enriched information and/or actionable information may be stored in caches or historical databases for later usage or processing, traceability, or accounting purposes.

4.11.3 Information Handling

- An IoT system is typically deployed to monitor and control Physical Entities. Monitoring and controlling Physical Entities is in turn performed by mainly the

Devices, Communication, IoT Services, and Virtual Entity FGs in the functional view.

- Certain FCs of these FGs, as well as the rest of the FGs (Service Organization, IoT Process Management, Management, Security FGs), play a supporting role for the main FGs in the Reference Architecture, and therefore in the flow of information.
- Information handling of an IoT system depends largely on the specific problem at hand.
- The presentation of information handling in an IoT system assumes that FCs exchange and process information in figure 4.25

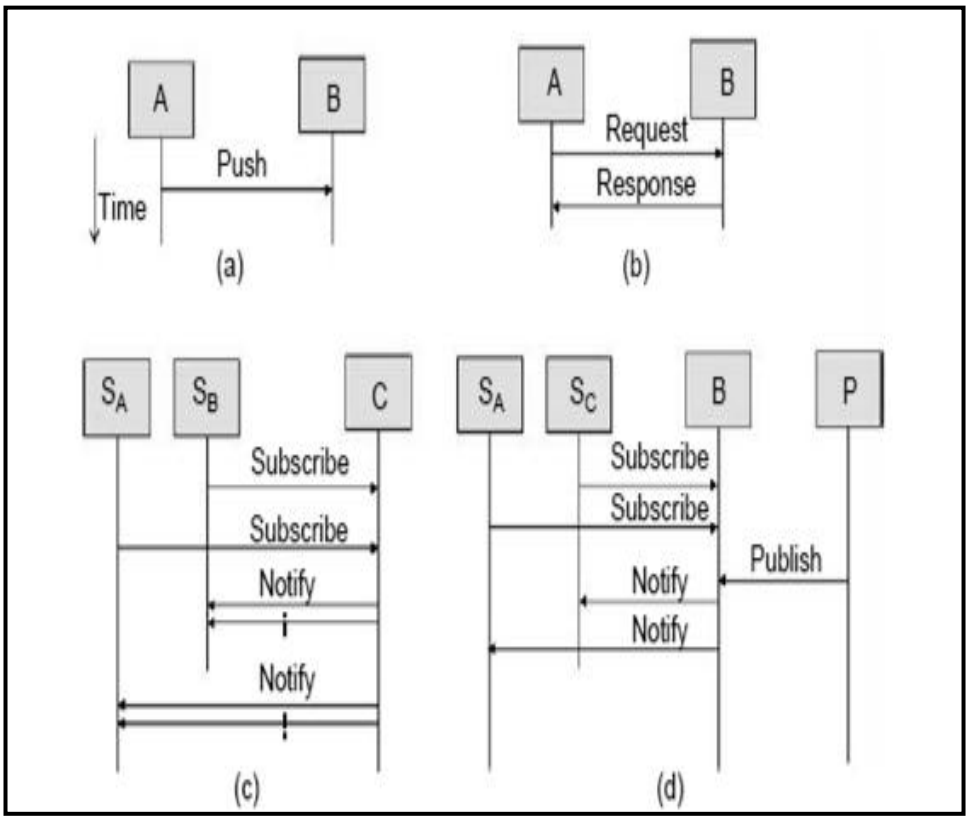


Figure 4.25: Information Exchange Patterns

- **Push:** An FC A pushes the information to another FC B provided that the contact information of the component B is already configured in component A, and component B listens for such information pushes.
- **Request/Response:** An FC A sends a request to another FC B and receives a response from B after A serves the request. Typically the interaction is synchronous in the sense that A must wait for a response from B before proceeding to other tasks, but in practice this limitation can be realized with parts of component A waiting, and other parts performing other tasks. Component B may need to handle concurrent requests

and responses from multiple components, which imposes certain requirements on the capabilities for the device or the network that hosts the FC.

- **Subscribe/Notify**: Multiple subscriber components (SA, SB) can subscribe for information to a component C, and C will notify the relevant subscribers when the requested information is ready. This is typically an asynchronous information request after which each subscriber can perform other tasks. The Subscribe/Notify pattern is applicable when typically one component is the host of the information needed by multiple other components. Then the subscribers need only establish a Subscribe/Notify relationship with one component. If multiple components can be information producers or information hosts, the Publish/Subscribe pattern is a more scalable solution from the point of view of the subscribers.
- **Publish/Subscribe**: In the Publish/Subscribe (also known as a Pub/Sub pattern), there is a third component called the broker B, which mediates subscription and publications between subscribers (information consumers) and publishers (or information producers).

Device , IoT Service, and Virtual Entity Service Interactions

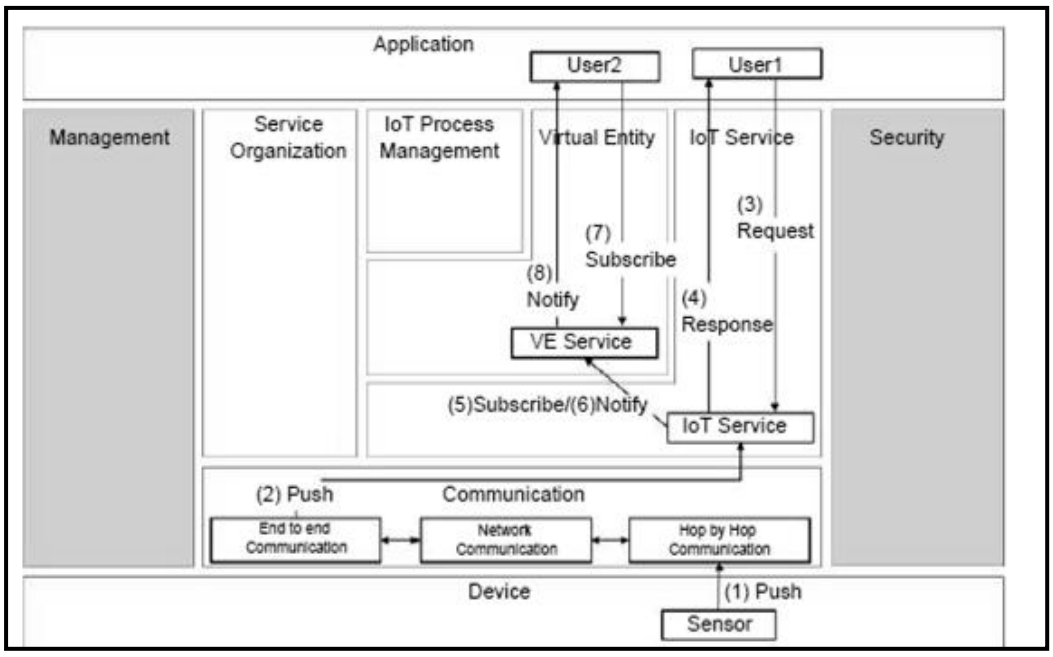


Figure 4.26: Device, IoT Service and Virtual Entity

In Figure 4.26 we assume that the generated sensed data is pushed by a sensor device that is part of a multi-hop mesh network such as IEEE 802.15.4 through the Hop-by-Hop, Network, and End-to-End communication FCs towards the Sensor Resource hosted in the network the

Sensor Resource is not shown in the figure, only the associated IoT Service. A cached version of the sensor reading on the Device is maintained on the IoT Service.

- When User1 (Step 3) requests the sensor reading value from the specific Sensor Device (assuming User1 provides the Sensor resource identifier), the IoT Service provides the cached copy of the sensor reading back to the User1 annotated with the appropriate metadata information about the sensor measurement, for example, timestamp of the last known reading of the sensor, units, and location of the Sensor Device.
- Also assume that that the Virtual Entity Service associated with the Physical Entity (e.g. a room in a building) where the specific Sensor Device has been deployed already contains the IoT Service as a provider of the "hasTemperature" attribute of its description. The Virtual Entity Service subscribes to the IoT Service for updates of the sensor readings pushed by the Sensor Device (Step 5).
- Every time the Sensor Device pushes sensor readings to the IoT Service, the IoT Service notifies (Step 6) the Virtual Entity Service, which updates the value of the attribute "hasTemperature" with the sensor reading of the Sensor Device.
- At a later stage, a User2 subscribing to changes on the Virtual Entity attribute "hasTemperature" is notified every time the attribute changes value.

IoT Service Resolution

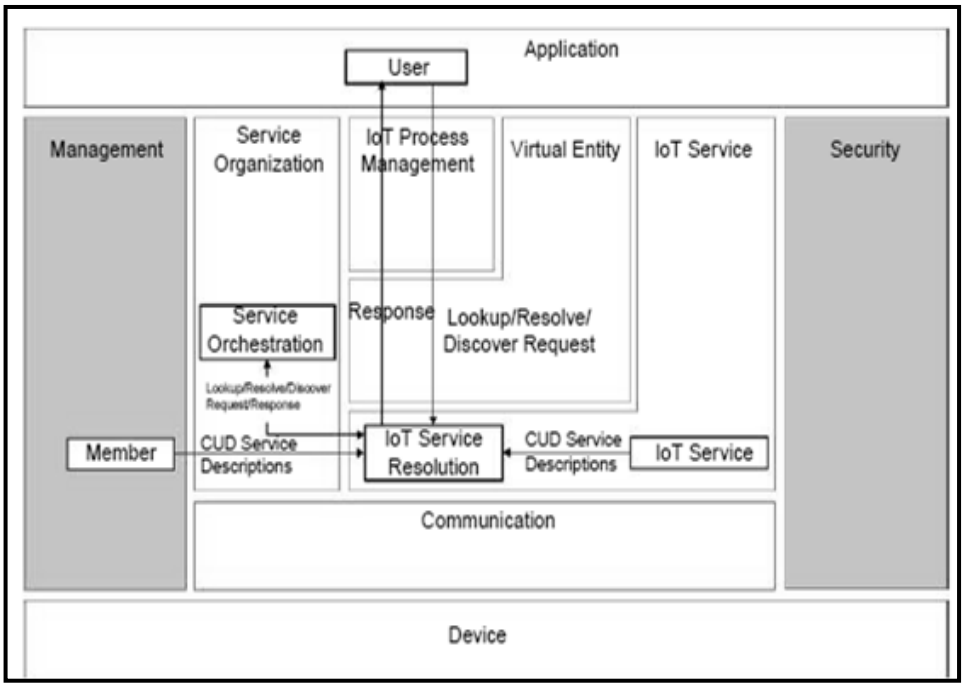


Figure 4.27: IoT Service Resolution

- In figure 4.27 depicts the information flow when utilizing the IoT Service Resolution FC. The IoT Service Resolution implements two main interfaces,one for the CUD of Service Description objects in the IoT Service Resolution database/store, and one for lookup/resolution/discovery of IoT Services.
- As a reminder, the lookup and resolution operations provide the Service Description and the Service locator, respectively, given the Service identifier and the discovery operation returns a (set of) Service Description(s) given a list of desirable attributes that matching Service Descriptions should contain.
- The CUD operations can be performed by the IoT Service logic itself or by a management component (e.g. Member FC in Figure 4.27).
- The lookup/resolution and discovery operation can be performed by a User as a standalone query or the Service Orchestration as a part of a Composed Service or an IoT Process. If a discovery operation returns multiple matching Service Descriptions, it is upon the User or the Service Orchestration component to select the most appropriate IoT Service for the specific task. Although the interactions in Figure 4.27 follow the Request/Response patterns, the lookup/resolution/discovery operations can follow the Subscribe/Notify pattern in the sense that a User or the Service Orchestration FC subscribe to changes of existing IoT Services for lookup/resolution and for the discovery of new Service Descriptions in the case of a discovery operation.

Virtual Entity Service Resolution

- The Figure 8.6 describes the information flow when the Virtual Entity Service Resolution FC is utilized. The Virtual Entity Resolution FC allows the CUD of Virtual Entity Descriptions, and the lookup and discovery of Virtual Entity Descriptions.
- A lookup operation by a User or the Service Orchestration FC returns the Virtual Entity Description given the Virtual Entity identity, while the discovery operation returns the Virtual Entity Description(s) given a set of Virtual Entity attributes (simple or complex) that matching Virtual Entities should contain.
- The Virtual Entity Resolution FC mediates the equests/responses/subscriptions/notifications between Users and the Virtual Entity Registry,which has a simple create/read/update/delete (CRUD) interface given the Virtual Entity identity.

- The FCs that could perform CUD operations on the Virtual Entity Resolution FC are the IoT Services themselves due to internal configuration, the Member Management FC that maintains the associations as part of the system setup, and the Virtual Entity and IoT Service Monitoring component whose purpose is to discover dynamic associations between Virtual Entities and IoT Services.

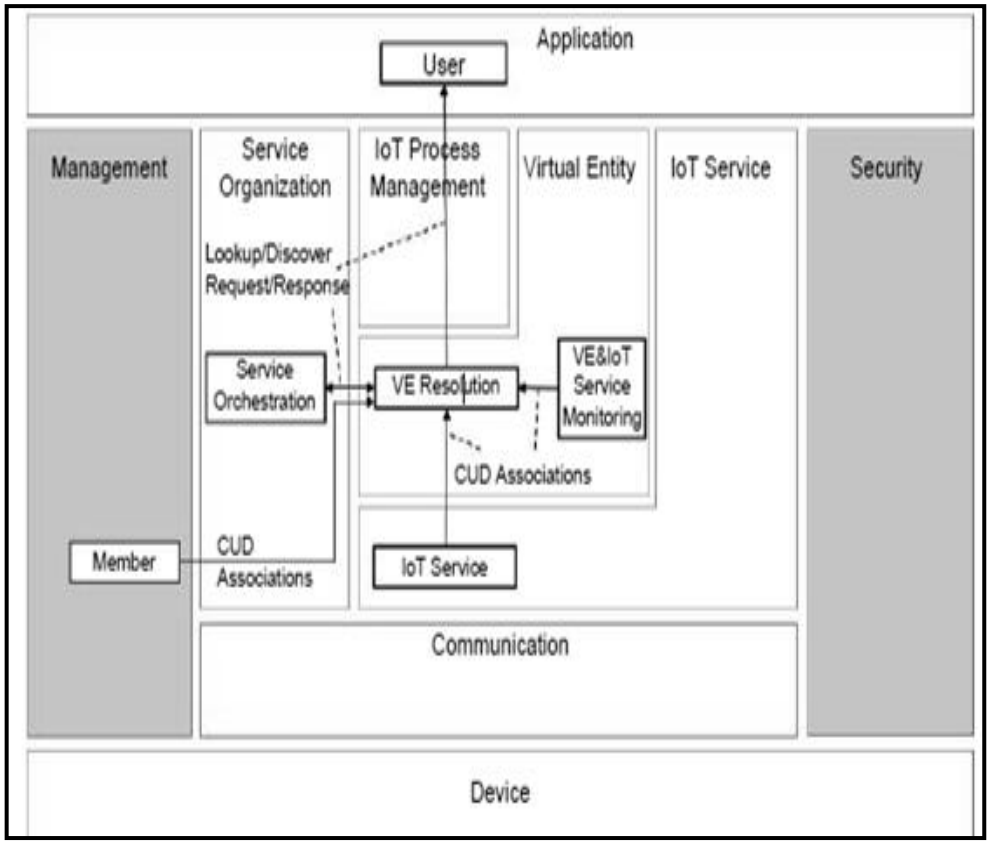


Figure 4.28: Virtual Entity Resolution

- It is important to know that Subscribe/Notify interaction patterns can also be applicable to the lookup/discovery operations, the same as the Request/Response patterns provided the involved FCs implement Subscribe/Notify interfaces.

Service Choreography and Processing IoT Services

- The CEP Service needs the information from two IoT services, e.g. IoT Services corresponding to two Sensor Resources hosted on two Sensor Devices,and produces one output.
- The CEP IoT Service expects the inputs to be published/pushed to its interfaces, while the output interface conforms to a Subscribe/Notify interaction pattern.

- The individual IoT Services A and B expose interfaces that also comply with the Publish/Subscribe interaction pattern. The FC that can connect these three components is the Service Choreography FC that realizes a Publish/Subscribe interaction pattern.
- As a first step, the IoT Service C subscribes to the Service Choreography FC that it requires IoT Services A and B as inputs In the meantime, a User subscribes to the Service Choreography FC that it needs the output of the CEP IoT Service C. When the individual IoT Services A and B publish their output to the Service Choreography FC, these outputs are published/forwarded to the IoT Service C, which needs them to produce information of type C. After performing CEP filtering, the IoT Service C publishes the output of type C to the Service Choreography FC, which ublishes/forward it to the User.

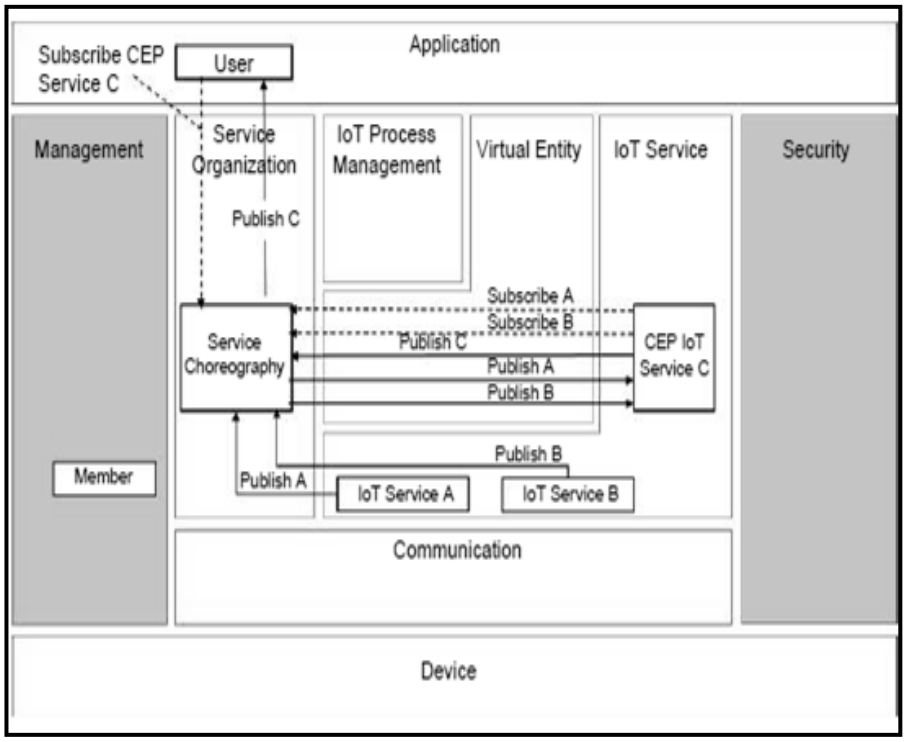


Figure 4.29: Service Choreography and Processing IoT Services

****