| CMR INSTITUTE OF TECHNOLOGY | USN | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Internal Assesment Test – III, November 2019**

| Sub: | INTERNET OF THINGS | | | | | | | Code: | 17MCA552 |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 19-11-2019 | Duration: | 90 mins | Max Marks: | 50 | Sem: | V | Branch: | MCA |

Answer **ONE FULL QUESTION** from each part

| | | Marks | OBE | |
|---|---|---|---|---|
| | | | CO | RBT |
| | **Part – I** | | | |
| 1a) | Explain in detail about IOT information model | 10 | CO4 | L1 |
| | (OR) | | | |
| 2 | Discuss about IOT Information view | 10 | CO4 | L3 |
| | **Part – II** | | | |
| 3a) | Explain about ETSI M2M high-level architecture | 10 | CO4 | L1 |
| | (OR) | | | |
| 4a) | Describe IoT-A Functional Model. | 10 | CO4 | L3 |
| | **Part – III** | | | |
| 5a) | With a neat diagram, briefly explain the IoT Reference Model. | 5 | CO4 | L2 |
| b) | List and explain ETSI M2M interface | 5 | CO4 | L1 |
| | **(OR)** | | | |
| 6. | Discuss about IMC-AESOP: from the web of things to the cloud of things | 10 | CO5 | L2 |
| | **Part – IV** | | | |
| 7. | Explain about Function View | 10 | CO4 | L1 |
| | **(OR)** | | | |
| 8a) | Explain safety, privacy and trust with respect to IOT | 5 | CO5 | L1 |
| | With neat diagram Identify the relationship between IOT information and domain model | 5 | CO4 | L3 |
| | **Part – V** | | | |
| 9 a) | Explain industrial automation with respect to Service-oriented architecture-based device integration | 10 | CO1 | L1 |
| | **(OR)** | | | |
| 10a) | Summarise IOT technical Design Constraints | 10 | CO4 | L2 |

**Internal Assesment Test – III, November 2019**
**Answer Key**

INTERNET OF THINGS -17MCA552

| | | | |
|---|---|---|---|
| 1a) | Explain in detail about IOT information model. | 10 | CO4 |

IoT Information Model is presented using Unified Modeling Language (UML) diagrams. 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

**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

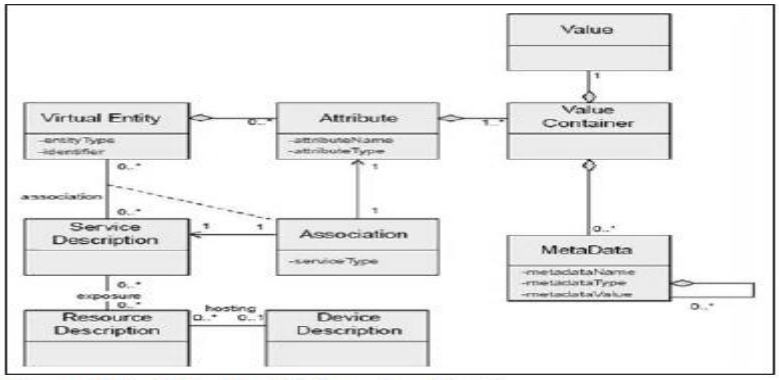



**Figure 4.18:** High –level Information Model

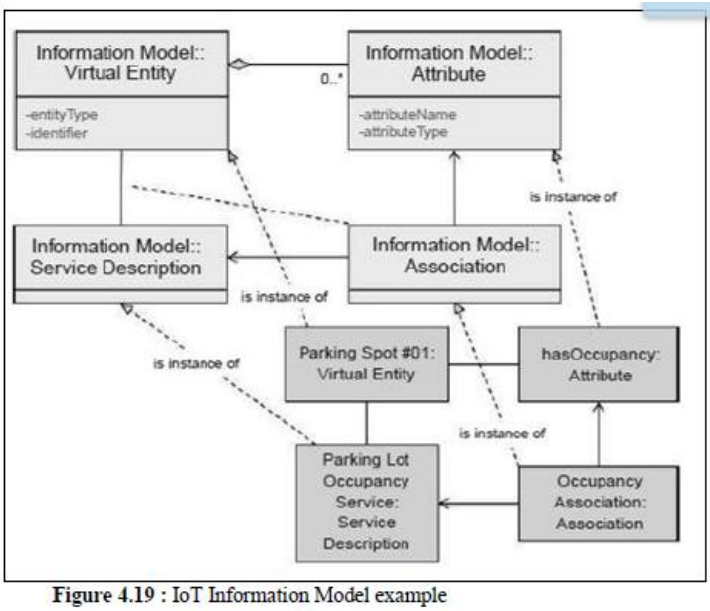



**Figure 4.19 :** IoT Information Model example

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

2. Discuss about IOT Information view. 10 CO4

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,

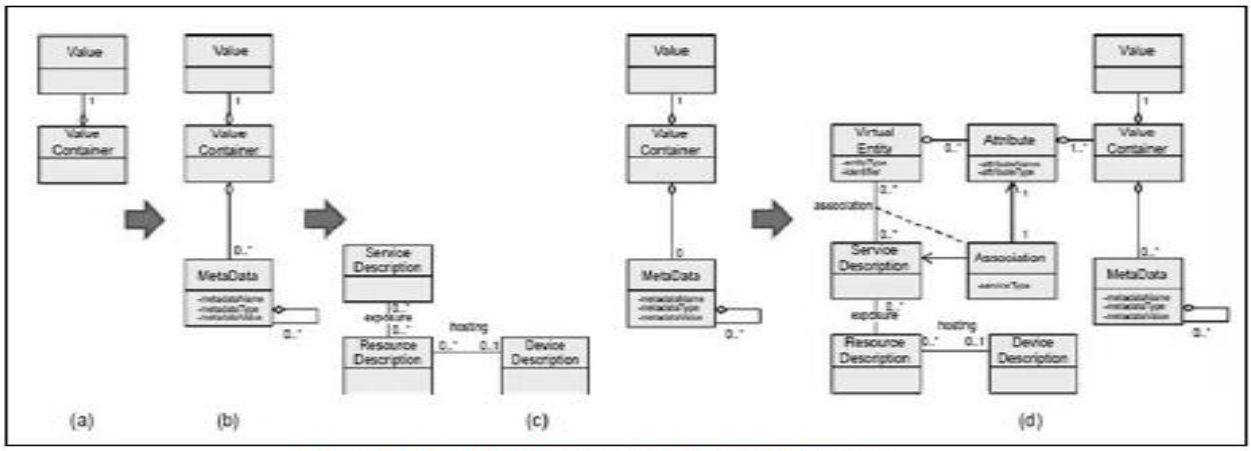



**Figure 4.24: Information Exchange Patterns**

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).
 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.

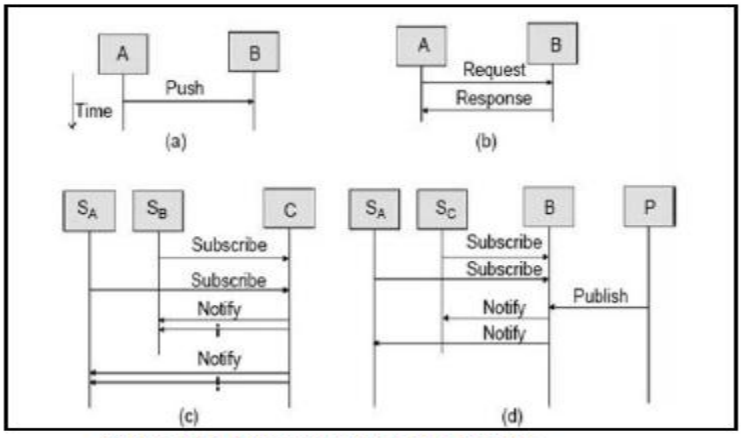



**Figure 4.25: Information Exchange Patterns**

 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
 **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).

| 3a) | Explain about ETSI M2M high-level architecture. | 10 | CO4 |
|---|---|---|---|

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).

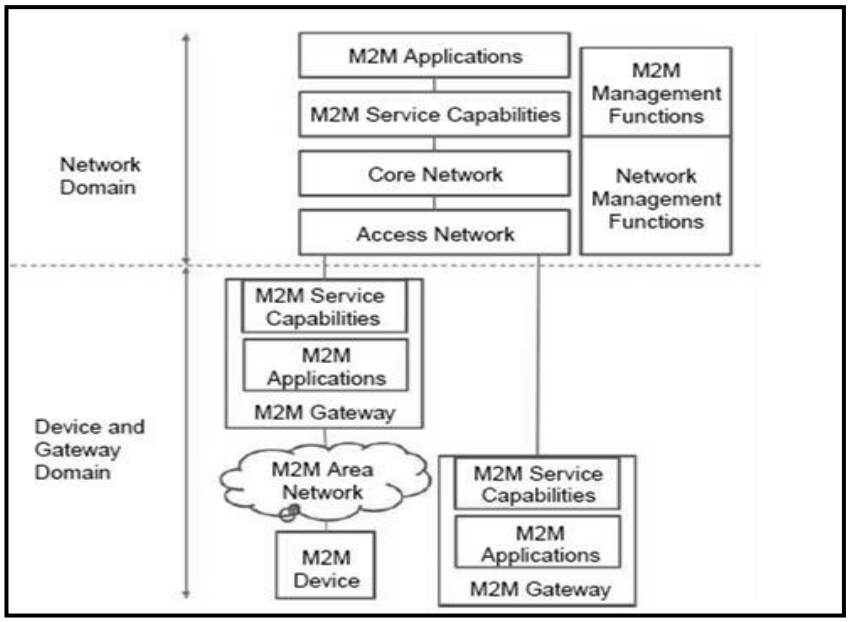



**Figure 4.1:** ETSI M2M High Level Architecture

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).

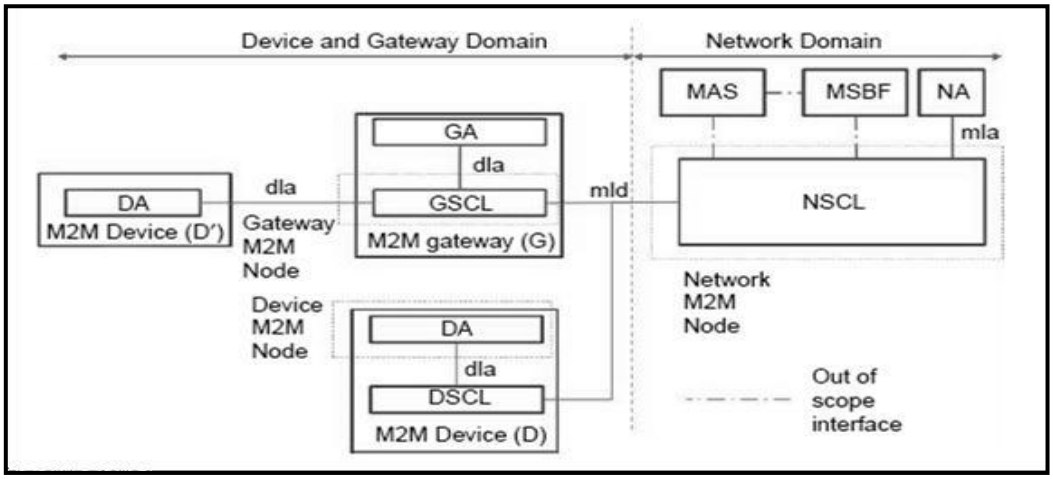



**Figure 4.2:** M2M service Capabilities, M2M Nodes and Open Interfaces

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.

4a) Describe IoT-A Functional Model. 10 CO4

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. 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.

### 4.1.1 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.

**4.1.2 : 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.

**4.1.3 IoT Service functional group**

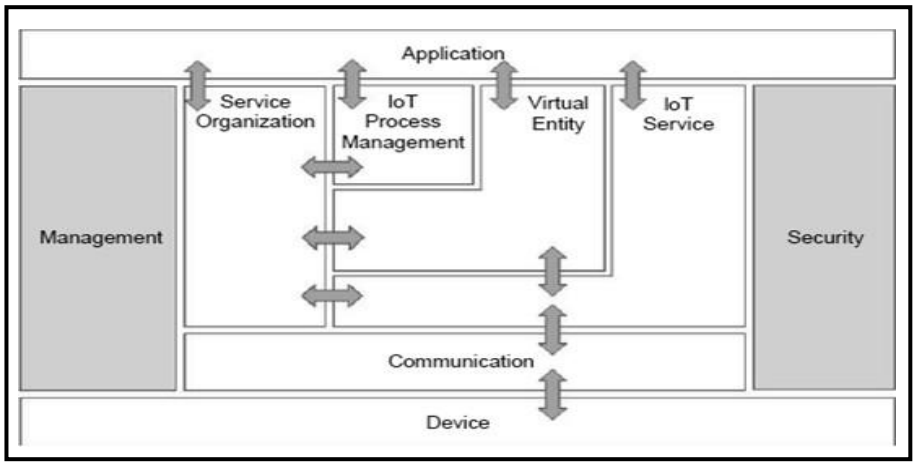
The IoT Service FG corresponds mainly to the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).Support functions such as directory services, which allow discovery of Services and resolution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

**4.1.4 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.




**4.1.5 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.

**4.1.6 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.

**4.1.7 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.

**4.1.8 Security functional group**

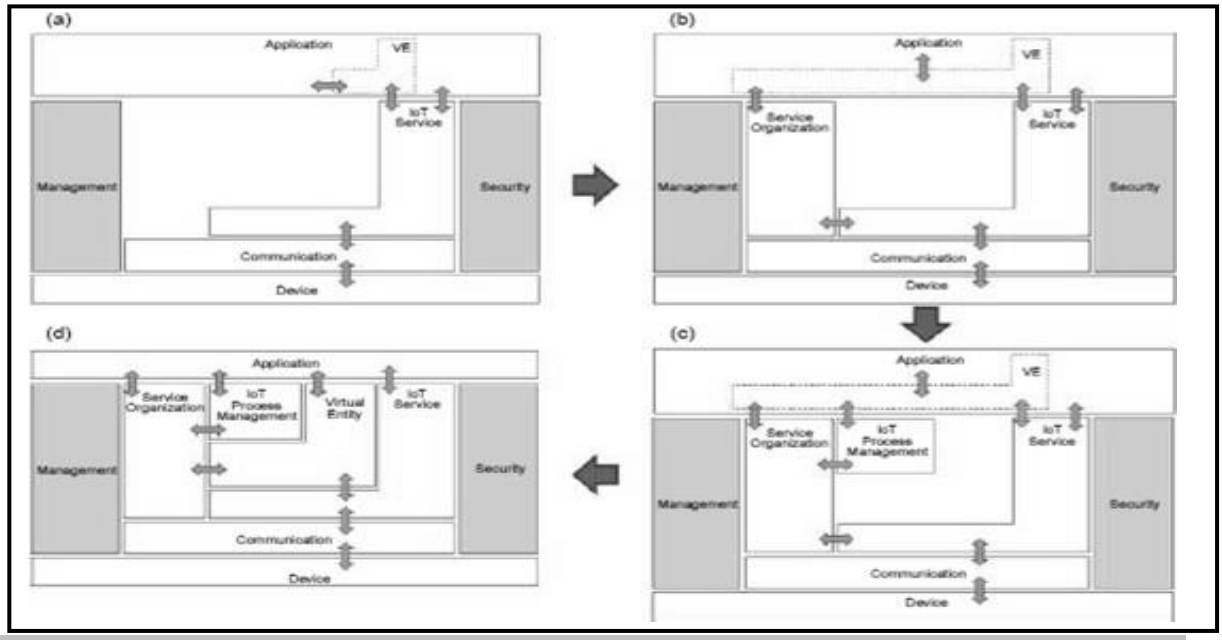
The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy. The Security FG contains components for Authentication of Users (Applications, Humans), Authorization of access to Services by Users, secure communication (ensuring 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.

### 4.1.9 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.

### 4.1.10 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.




5a) With a neat diagram, briefly explain the IoT Reference Model. 5 CO4

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.

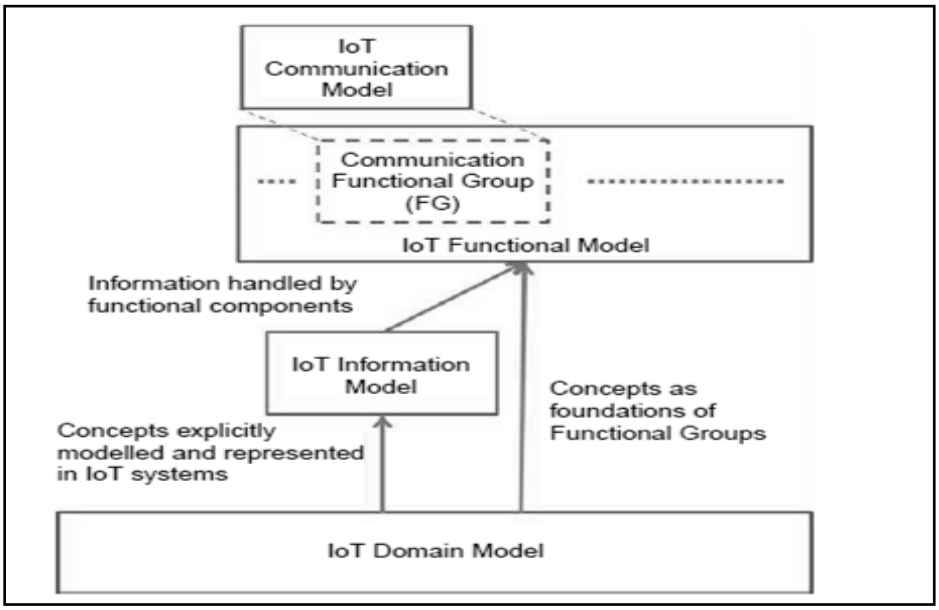



**Figure 4.11:** IoT Reference Model.

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

- 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.

| | | | |
|---|---|---|---|
| b) | List and explain ETSI M2M interface. | 5 | CO4 |

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.

| | | | |
|---|---|---|---|
| 6. | Discuss about IMC-AESOP: from the web of things to the cloud of things. | 10 | CO5 |

Visionary approaches have been further developed in the industry-driven project IMC-AESOP (2013). There the vision of SOCRADES has been pushed forward by considering the rapid advances in hardware and software, as well as IT concepts. Therefore, we go beyond WS-enabled devices towards the cloud in order to harness its benefits, such as resource flexibility, scalability, etc. The result will be a highly dynamic flat information-driven infrastructure (Figure 5.5) that will empower the rapid development of better and more efficient next generation industrial applications, while in parallel satisfying the agility required by modern enterprises. This vision is only realizable due to the distributed, autonomous, intelligent, pro-active, faulttolerant, reusable (intelligent) systems, which expose their capabilities, functionalities, and structural characteristics as services located in a "service cloud."

Although today factories are composed and structured by several systems viewed and interacting in a hierarchical fashion following mainly the specifications of standard enterprise architectures, there is an increasing trend to move towards information-driven interaction that goes beyond traditional hierarchical deployments and can coexist with them. With the empowerment offered by modern SOAs, the functionalities of each system (or even device) can be offered as one or more services of varying complexity, which may be hosted in the cloud and composed by other (potentially cross-layer) services, as depicted in Figure 5.5. This transition marks a paradigm change in the interaction among the different systems, applications, and users. Although the traditional hierarchical view coexists, there is now a flat information-based architecture that depends on a big variety of services exposed by the cyberphysical systems and their composition. Next generation industrial applications can now rapidly be composed by selecting and combining the new information and capabilities offered (as services is depicted in a Figure 5.6 in the cloud) to realize their goals.

The envisioned transition to the future cloud-based industrial systems (Karnouskos & Colombo 2011, Karnouskos et al. 2012)

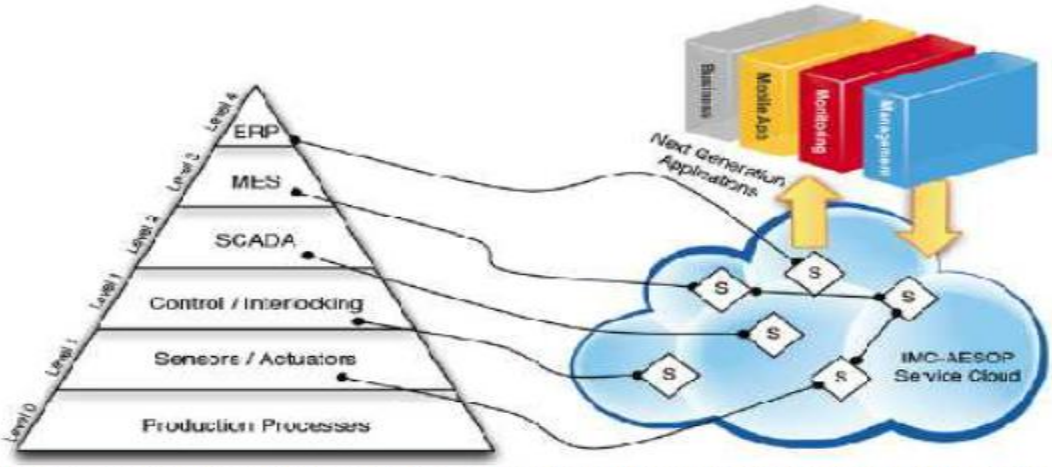



Figure No 5.5 Future industrial system view of cloud-based composition of cyber-physical services.

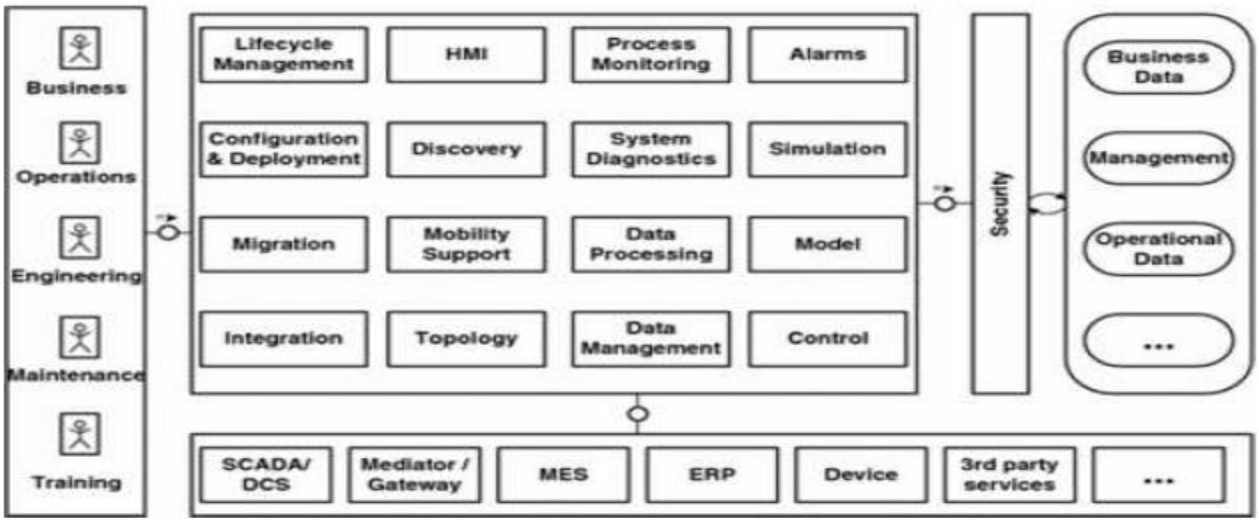



Figure No 5.6 IMC-AESOP cloud-based architecture vision

Several "user roles" will interact with the envisioned architecture (Figure 5.6), either directly or indirectly as part of their participation in a process plant. The roles define actions performed by staff and management, and simplifies grouping of tasks into categories such as business, operations, engineering, maintenance, engineering training, etc.

As depicted in Figure 5.6, it is possible to distinguish several service groups for which there have also been defined some initial services. All of the services are considered essential, with varying degrees of importance for next generation, cloud-based, collaborative automation systems. The services are to provide key enabling functionalities to all stakeholders (i.e. other services, as well) as cyber-physical systems populating the infrastructure. As such, all these systems can be seen as entities that may have a physical part realized in on-premise hardware, as well as a virtual part realized in software potentially on-device and in-cloud. This emerging "*Cloud of Things*" has the potential to transform the way we design, deploy, and use applications and cyber-physical systems. Typical functionalities include alarms, configuration and deployment, some control, data management, data processing, discovery, lifecycle management, HMI, integration, simulation, mobility support, monitoring, security, etc. It is clear that this is a proposal that will need to be further refined in real-world scenarios, however, it clearly depicts a step towards a highly flexible M2M infrastructure for the automation domain that abstracts from devices and focuses on functionalities that can reside on-device, in-network, and harness the power of the cloud.

| 7. | Explain about Function View . | 10 | CO4 |
|---|---|---|---|

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

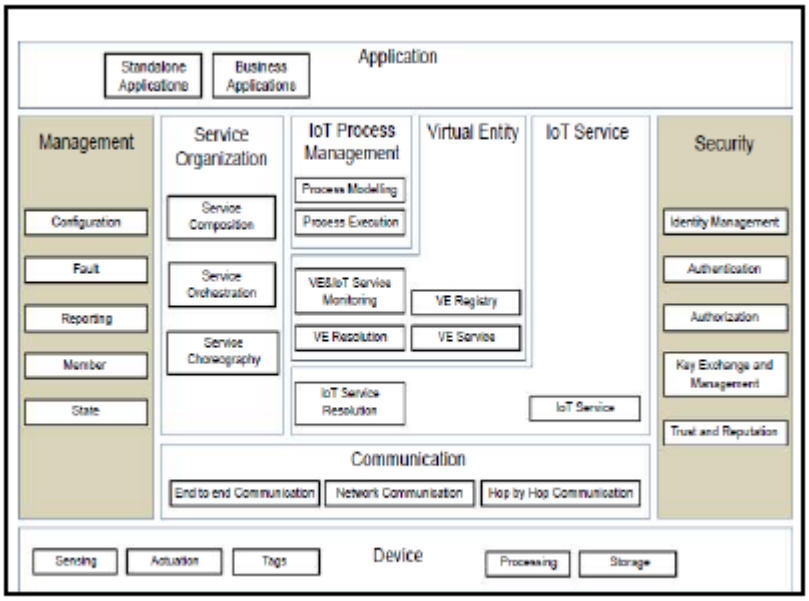



**Figure 4.23**: IoT Functional View

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.

| | | | |
|---|---|---|---|
| 8a) | Explain safety, privacy and trust with respect to IOT. | 5 | CO5 |

**4.2 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.

**4.2.1 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.

**4.2.2 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.

### 4.2.3 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.

b) With neat diagram Identify the relationship between IOT information and domain model. 5 CO4

(a) 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.

(b) 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.

(c) 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.

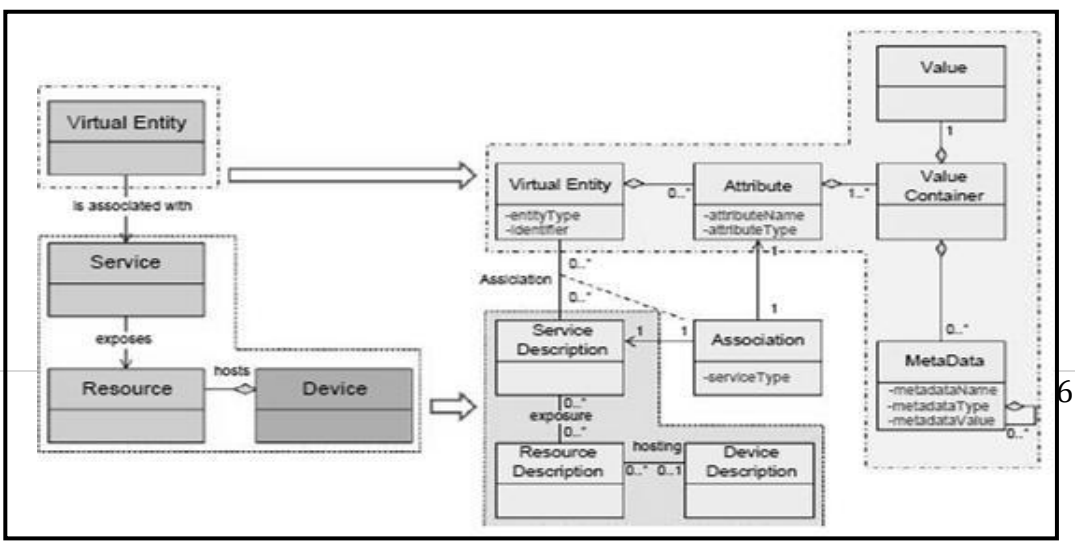


**Figure 4.20:** Relationship between core concepts of IoT Domain Model and IoT information Model

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
- 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.

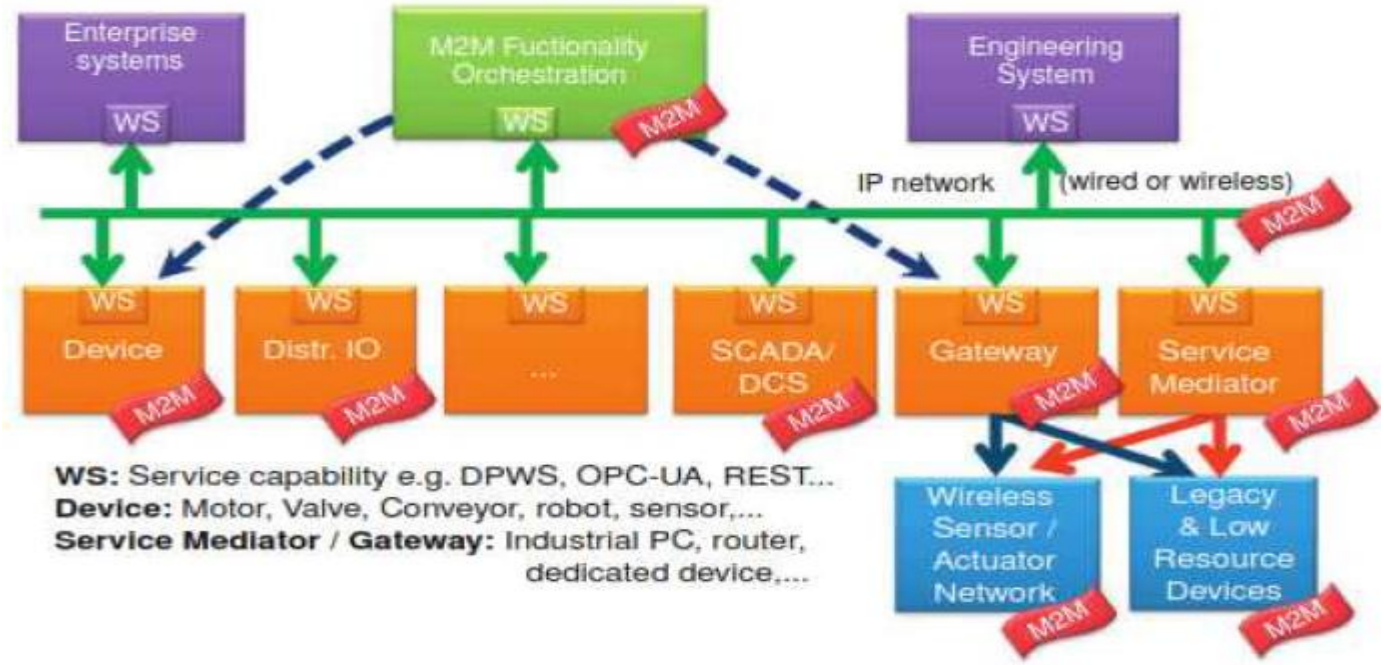
The IoT Information Model also contains Resource descriptions because Resources are associated with Services and Devices in the IoT Domain model. A Resource description contains the following information:

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.

## 9. Explain industrial automation with respect to Service-oriented architecture-based device integration.

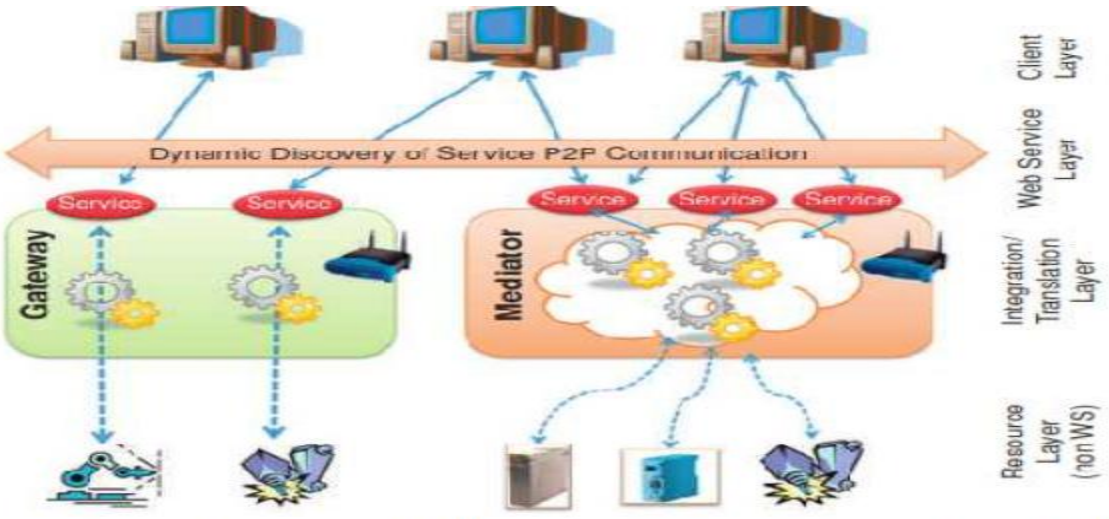
The emerging approach in industrial environments is to create system intelligence by a large population of intelligent, small, networked, embedded devices at a high level of granularity, as opposed to the traditional approach of focusing intelligence on a few large and monolithic applications. This increased granularity of intelligence distributed among loosely coupled intelligent physical objects facilitates the adaptability and re-configurability of the system, allowing it to meet business demands not foreseen at the time of design, and providing real business benefits.

The Service-Oriented Architecture (SOA) paradigm can act as a unifying technology that spans several layers, from sensors and actuators used for monitoring and control at shop-floor level, up to enterprise systems and their processes as envisioned in Figure 5.1.




5.1 M2M SOA-based integration

This common "backbone" means that M2M is not limited to direct (e.g. proximity) device interaction, but includes a wide range of interactions in a cross-layer way with a variety of heterogeneous devices, as well as systems and their services. This yields multiple benefits for all stakeholders involved. Such visions have been proposed and realized, demonstrating the benefits and challenges involved. Internet Protocol (IP)-based, and more specifically web technologies and protocols (e.g. OPC-UA, DPWS, REST, Web Services (WS), etc.), constitute a promising approach towards the fundamental goal of enabling easy integration of device-level services. with enterprise systems overcoming the heterogeneity and specific implementation of hardware and software of the device. Surely industry specific requirements for security, resilience, and availability of near real-time event information needs to be effectively tackled. The latter are also seen as key enablers for the new generation of enterprise system applications such as business activity monitoring, overall equipment effectiveness optimization, maintenance optimization, etc. The SOA-based vision is not expected to be realized overnight, but may take a considerable time depending on the lifecycle processes of the specific industry, and may be impacted by micro- and macro-economic aspects. Hence, it is important that migration capabilities are provided so that we can harvest some of the benefits today and provide a stepwise process towards achieving the vision. The concepts of gateway and service mediator, as depicted in Figure 5.2,




5.2 Non- Service-enabled device integration: Gateway vs. Service Mediator

can help towards this direction. Dynamic device discovery is a key functionality in the future M2M. As an example, Figure 11.3 depicts how Windows 7 can discover dynamically heterogeneous devices that areSOA-ready (i.e. equipped with web services; Devices Profile for Web Services, DPWS). 5.2 Non-Service-enabled device integration: Gateway vs. Service Mediator A Gateway is a device that controls a set of lower-level non-service-enabled devices, each of which is exposed by the Gateway as a service-enabled device. This approach allows gradually replacing limited-resource devices or legacy devices by natively WS-enabled devices without impacting the applications using these devices. This is possible since the same WS interface is offered this time by the WS-enabled device and not by the Gateway. This approach is used when each of the controlled devices needs to be known and addressed individually by higher-level services or applications. Originally meant to aggregate various data sources (e.g. databases, log files, etc.), the Mediator components evolved and are now used to not only aggregate various services, but possibly also compute/process the data they acquire before exposing it as a service.

Service Mediators aggregate, manage, and eventually represent services based on some semantics (e.g. using ontologies). In our case, the Service Mediator could be used to aggregate various non WSenabled devices. This way, higher-level applications could communicate to Service Mediators offering WS instead of communicating to devices with proprietary interfaces. The benefits are clear, as we don't have the hassle of (proprietary) driver integration. Furthermore, now processing of data can be done at Service Mediator level, and more complex behavior can be created which was not possible before from the standalone devices. As we can see in future IoT infrastructures dominated by billions of devices with different capabilities and needs, we have to consider how these integrate with each other and enable the realization of new innovative approaches. This assumes increased integration and collaboration among the various layers existing in industries (i.e. from the shop floor up to enterprise systems). Several concepts and efforts are directed towards abstracting from the device-specific aspects and defining device-agnostic, but functionality-focused, layers of integration.

## 10. Summarise IOT technical Design Constraints.

**Technical Design Constraints- hardware is popular again**

- The IoT will see additional circuitry built into a number of existing products and machines _ from washing machines to meters. Giving these things an identity, and the ability to represent themselves online and communicate with applications and other things, represents a significant, widely recognized opportunity.
- For manufacturers of products that typically contain electronic components, this process will be relatively straightforward. Selection of appropriate communications technologies that can be integrated with legacy designs (e.g. motherboards) will be relatively painless.
- The operational environments and the criticality of the information transmitted to and from these products, however, will present some unconventional challenges and design considerations.
- The IoT will, on the other hand, allow for the development of novel applications in all imaginable scenarios. Emerging applications of M2M and wireless sensor and actuator networks have seen deployment of sensing capabilities in the wild that allow stakeholders to optimize their businesses, glean new insight into relevant physical and environmental processes, and understand and control situations that would have previously been inaccessible.
- The technical design of any M2M or IoT solution requires a fundamental understanding of the specificity of the intended application and business proposition, in addition to heterogeneity of existing solutions.
- Developing an end-to-end instance of an M2M or IoT solution requires the careful selection, and in most cases, development of a number of complementary technologies.
- This can be both a difficult conceptual problem and integration challenge, and requires the involvement of the key stakeholder(s) on a number of conceptual and technological levels.Typically, it can be considered to be a combinatorial optimization problem _ where the optimal solution is the one that satisfies all functional and nonfunctional requirements, whilst simultaneously delivering a satisfactory cost-benefit ratio.
- This is particularly relevant for organizations wishing to compete with existing offerings, or for start-up ventures in novel application areas. Typically, capital costs in terms of "commissioning" and operational costs in "maintenance" must be considered. These may be balanced by resultant optimizations.

**5.2.1 Devices and Networks**

• Devices that form networks in the M2M Area Network domain must be selected, or designed, with certain functionality in mind.
• At a minimum, they must have an energy source (e.g. batteries, increasingly EH), computational capability (e.g. an MCU), appropriate communications interface (e.g. a Radio Frequency Integrated Circuit (RFIC) and front end RF circuitry), memory (program and data), and sensing (and/or actuation) capability.
• These must be integrated in such a way that the functional requirements of the desired application can be satisfied, in addition to a number of nonfunctional requirements that will exist in all cases.

#### 5.2.1.1 Functional requirements

Specific sensing and actuating capabilities are basic functional requirements. In every case with the exception of devices that might be deployed as a routing device in the case of range issues between sensing and/or actuating devices- the device must be capable of sensing or perceiving something interesting from the environment.

This is the basis of the application. Sensors, broadly speaking, are difficult to categorize effectively. Selecting a sensor that is capable of detecting a particular phenomenon of interest is essential. The sensor may directly measure the phenomenon of interest (e.g. temperature), or may be used to derive data or information about the phenomenon of interest, based on additional knowledge (e.g. a level of comfort). Sensors may sense a phenomenon that is local (i.e. a meter detecting total electricity consumption of a space) or distributed (e.g. the weather).

In many cases, sensing may be prohibitively expensive or unjustifiable at scale, and thus motivates the derivation of models that can reason over the sensor readings that are available. Air and water quality monitoring systems are typical of this type of problem. Given a particular phenomenon of interest, there are often numerous sensors capable of detecting the same phenomenon (e.g. types of temperature sensors), but have widely varying characteristics.

These characteristics relate to the accuracy of the sensor, its susceptibility to changing environmental conditions, its power requirements, its signal conditioning requirements, and so forth.

Displacement can be sampled intermittently, whereas if an accelerometer is duty-cycled, it is likely that data points of interest (i.e. real-world events) may be missed. Furthermore, the data requirements of the stakeholder must be taken into account. If all data points are required to be transmitted (which is the case in many scenarios, irrespective of the ability to reason locally within an M2M Area Network or WSN), this implies higher network throughput, data loss, energy use, etc. These requirements tend to change on a case-by-case basis.

#### 5.2.1.2 Sensing and communications field

The sensing field is of importance when considering both the phenomenon to be sensed (i.e. Is it local or distributed?) and the distance between sensing points. The physical environment has an implication on the communications technologies selected and the reliability of the system in operation thereafter. Devices must be placed in close enough proximity to communicate. Where the distance is too great, routing devices may be necessary.

Devices may become intermittently disconnected due to the time varying, stochastic nature of the wireless medium. Certain environments may be fundamentally more suited to wireless propagation than others. For example, studies have shown that tunnels are excellent environments for wireless propagation, whereas, where RF shielding can occur (e.g. in a heavy construction environment), communication range of devices can be significantly reduced.

#### 5.2.1.3 Programming and embedded intelligence

Devices in the IoT are fundamentally heterogeneous. There are, and will continue to be, various computational architectures, including MCUs (8-, 16-, 32-bit, ARM, 8051, RISC, Intel, etc.), signal conditioning (e.g. ADC), and memory (ROM, (S/F/D)RAM, etc.), in addition to communications media, peripheral components (sensors, actuators, buttons, screens, LEDs), etc. In some applications, where it

would previously have been typical to have homogeneous devices, a variety of sensors and actuators can actually exist, working collaboratively, but constituting a heterogeneous network in reality. In every case, an application programmer must consider the hardware selected or designed, and its capabilities. Typically, applications may be thought of cyclically and logically.

Application-level logic dictates the sampling rate of the sensor, the local processing performed on sensor readings, the transmission schedule, and the management of the communications protocol stack, among other things.

The ability to reconfigure and reprogram devices is still an unresolved issue for the research community in sensor networks, M2M, and the IoT. It relates both to the physical composition of devices, logical construction of the embedded software, and addressability of individual devices and security, to name a few.

#### 5.2.1.4 Power

Power is essential for any embedded or IoT device. Depending on the application, power may be provided by the mains, batteries, or conversion from energy scavengers (often implemented as hybrid power sources). The power source has a significant implication on the design of the entire system. If a finite power supply is used, such as a battery, then the hardware selected, in addition to the application level logic and communications technology, collectively have a major impact on the longevity of the application. This results in short-lived applications or increased maintenance costs. In most cases, it should be possible to analytically model the power requirements of the application prior to deployment. This allows the designer to estimate the cost of maintenance over time.

#### 5.2.1.5 Gateway

The Gateway, is typically more straightforward to design if it usually acts as a proxy; however, there are very few effective M2M or IoT Gateway devices available on the market today. Depending on the application, power considerations must be taken into account. It is also thought that the Gateway device can be exploited for performing some level of analytics on data transitioning to and from capillary networks.

#### 5.2.1.6 Non-Functional Gateway

There are a number of nonfunctional requirements that need to be satisfied for every application.

These are technical and non-technical:

- Regulations
  - For applications that require placing nodes in public places, planning permission often becomes an issue.
  - Radio Frequency (RF) regulations limit the power with which transmitters can broadcast. This varies by region and frequency band.
- Ease of use, installation, maintenance, accessibility
  - Simplification of installation and configuration of IoT applications is as yet unresolved beyond well-known, off-the-shelf systems. It is difficult to conceive a general solution to this problem. This relates to positioning, placement, site surveying, programming, and physical accessibility of devices for maintenance purposes.
  - Physical constraints (from several perspectives)
  - Can the additional electronics be easily integrated into the existing system?
  - Are there physical size limitations on the device as a result of the deployment scenario?
  - What kind of packaging is most suitable (e.g. IP-rated enclosures for outdoor deployment)?
  - What kind and size of antenna can I use?

#### 5.2.1.7 Financial Cost

Financial cost considerations are as follows:

- Component Selection: Typically, the use of these devices in the M2M Area Network domain is seen to reduce the overall cost burden by using non-leased communications infrastructure. However, there are research and development costs likely to be incurred for each individual application in the IoT that requires device development or integration. Developing devices in small quantities is expensive.
- Integrated Device Design: Once the energy, sensors, actuators, computation, memory, power, connectivity, physical, and other functional and nonfunctional requirements are considered, it is likely that an integrated device must be produced. This is essentially going to be an exercise in Printed Circuit Board (PCB) design, but will in many cases require some consideration to be paid to the RF front-end design. This means that the PCB design will require specific attention to be paid to the reference designs of the RFIC manufacturer during development, or potentially the integration of an additional Integrated Circuit (IC) that deals with the balun and matching network required.