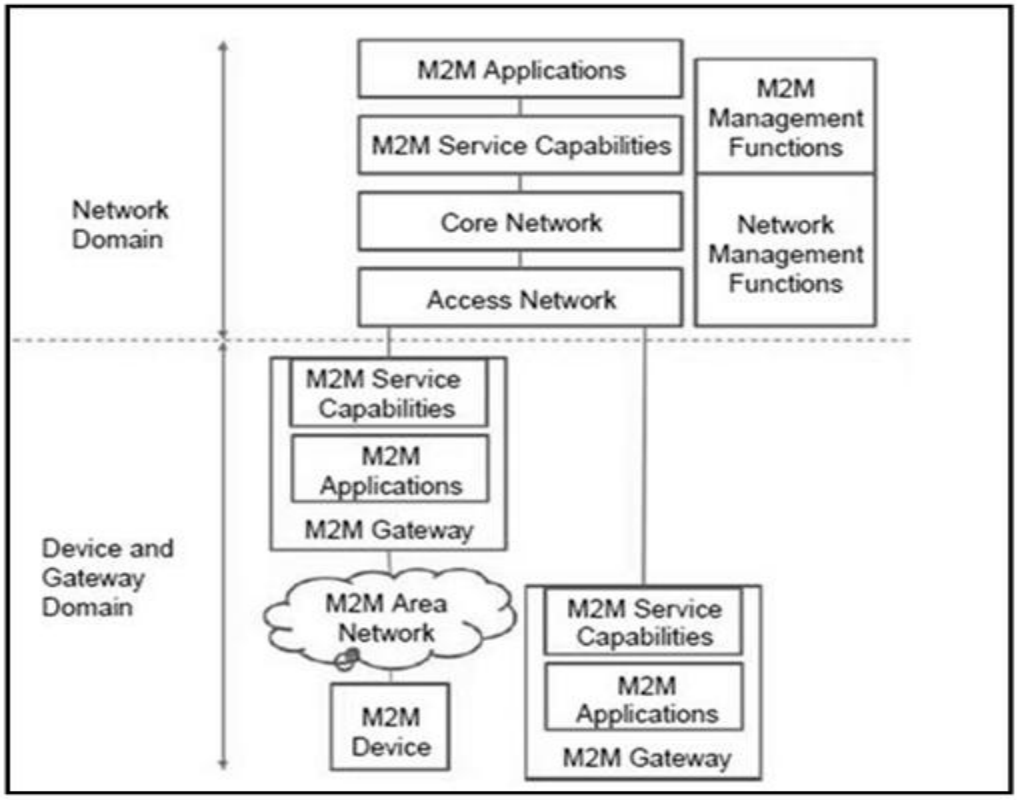
Answer Key

Internal Assessment Test II – December 2021

Sub:	INTERNET OF THINGS							Code:	18MCA542
Date:	17-12-2021	Duration:	90 mins	Date:	17-12-2021	Duration:	90 mins	Date:	17-12-2021

1 Explain about ETSI M2M high-level architecture



• **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.

2. IOT Domain model with example

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.

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.

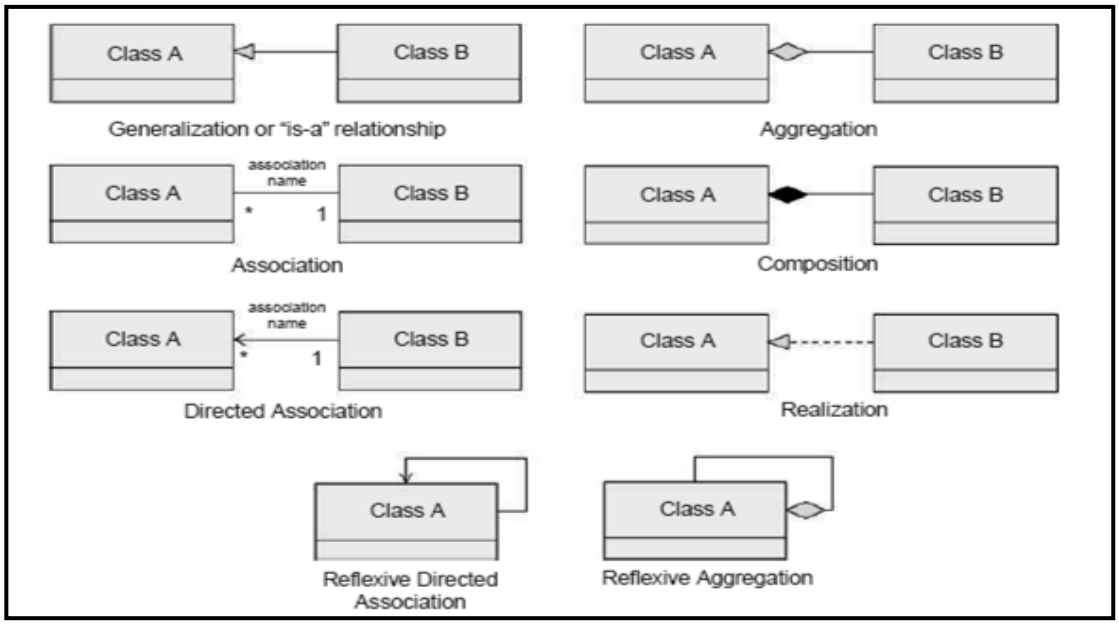


Figure 4.14: UML Class diagram main modeling concepts.

- 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.

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.

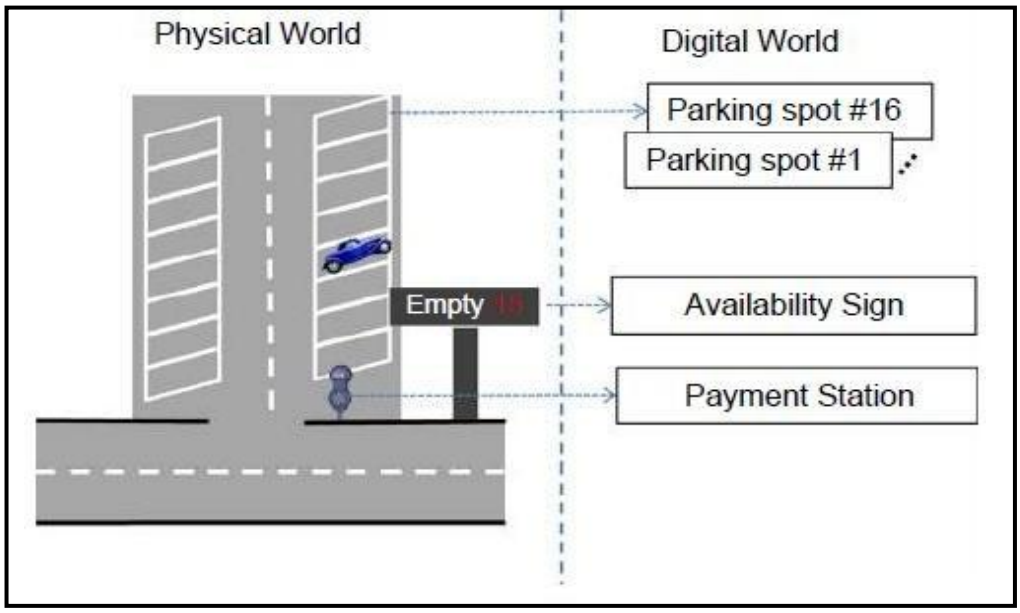


Figure 4.15: Physical vs. Virtual World.

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.

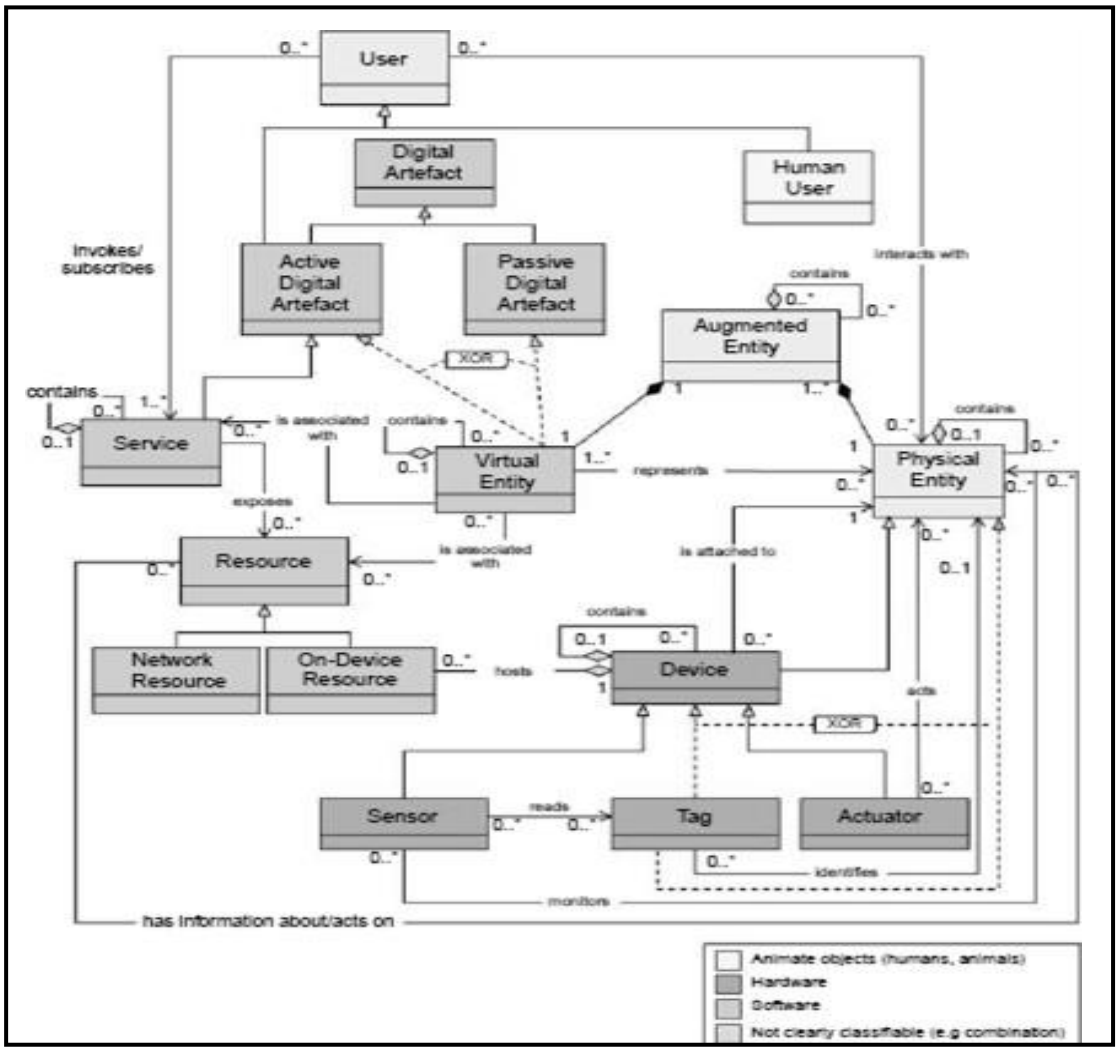


Figure 4.16: IoT Domain Model.

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.

3 Explain Basic and Advanced device types along with Gateways Basic Devices

Basic Devices

Devices that only provide the basic services of sensor readings and/or actuation tasks, and in some cases limited support for user interaction. LAN communication is supported via wired or wireless technology, thus a gateway is needed to provide the WAN connection.

- Intended for a single purpose - measuring air pressure or closing a valve. Some cases several functions are deployed on the same device- monitoring humidity, temperature, and light level.
- The requirements on hardware are low, both in terms of processing power and memory.
- The main focus is on keeping the bill of materials (BOM) as low as possible by using inexpensive microcontrollers with built-in memory and storage, often on an SoC integrated circuit with all main components on one single chip.
- The microcontroller hosts a number of ports that allow integration with sensors and actuators, such as General Purpose I/O (GPIO) and an analog-to-digital converter (ADC) for supporting analog input.
- For certain actuators, such as motors, pulse-width modulation (PWM) can be used.

Advanced devices

In this case the devices also host the application logic and a WAN connection. They may also feature device management and an execution environment for hosting multiple applications. Gateway devices are most likely to fall into this category.

- The distinction between basic devices, gateways, and advanced devices is not cut in stone, but some features that can characterize an advanced device are the following:
- A powerful CPU or microcontroller with enough memory and storage to host advancedapplications, such as a printer offering functions for copying, faxing, printing, and remote management.
- A more advanced user interface with, for example, display and advanced user input in the form of a keypad or touch screen.
- Video or other high bandwidth functions.
- A gateway serves as a translator between different protocols, e.g. between IEEE 802.15.4 or IEEE 802.11, to Ethernet or cellular.
- • There are many different types of gateways, which can work on different levels in the protocol layers. ost often a gateway refers to a device that performs translation of the physical and link layer, but application layer gateways (ALGs) are also common. The latter is preferably avoided because it adds complexity and is a common source of error in deployments.
- • Some examples of ALGs include the ZigBee Gateway Device (ZigBee Alliance 2011), which translates from ZigBee to SOAP and IP, or gateways that translate from Constrained Application Protocol (CoAP) to Hyper Text Transfer Protocol/Representational State Transfer (HTTP/REST).
- • For some LAN technologies, such as 802.11 and Z-Wave, the gateway is used for inclusion and exclusion of devices.
- • This typically works by activating the gateway into inclusion or exclusion mode and by pressing a button on the device to be added or removed from the network.
- • For very basic gateways, the hardware is typically focused on simplicity and low cost, but frequently the gateway device is also used for many other tasks, such as data management, device management, and local applications. In these cases, more powerful hardware with GNU/Linux is commonly used.

- • The following sections describe these additional tasks in more detail.
- 3.1.3.1 Data Management
- • Typical functions for data management include performing sensor readings and caching this data, as well as filtering, concentrating, and aggregating the data before transmitting it to back-end servers.
- 3.1.3.2 Local applications
- • Examples of local applications that can be hosted on a gateway include closed loops, home alarm logic, and ventilation control, or the data management.
- • The benefit of hosting this logic on the gateway instead of in the network is to avoid downtime in case of WAN connection failure, minimize usage of costly cellular data, and reduce latency. The execution environment is responsible for the lifecycle management of the applications, including installation, pausing, stopping, configuration, and uninstallation of the applications.
- • A common example of an execution environment for embedded environments is OSGi, which is based on java applications are built as one or more Bundles, which are packaged as Java JAR files and installed using a so-called Management Agent. The Management Agent can be controlled from, for example, a terminal shell or via a protocol such as CPE WAN Management Protocol (CWMP).
- • Bundle packages can be retrieved from the local file system or over HTTP.
- • The benefit of versioning and the lifecycle management functions is that the OSGi environment never needs to be shut down when upgrading, thus avoiding downtime in the system.
- 3.1.3.3 Device Management
- • Device management (DM) is an essential part of the IoT and provides efficient means to perform many of the management tasks for devices:
- o Provisioning: Initialization (or activation) of devices in regards to configuration and features to be enabled.
- o Device Configuration: Management of device settings and parameters.
- o Software Upgrades: Installation of firmware, system software, and applications on the device.
- o Fault Management: Enables error reporting and access to device status
- • In the simplest deployment, the devices communicate directly with the DM server. This is, however, not always optimal or even possible due to network or protocol constraints, e.g. due to a firewall or mismatching protocols.
- • In these cases, the gateway functions as mediator between the server and the devices, and can operate in three different ways:
- o If the devices are visible to the DM server, the gateway can simply forward the messages between the device and the server and is not a visible participant in the session.
- o In case the devices are not visible but understand the DM protocol in use, the gateway can act as a proxy, essentially acting as a DM server towards the device and a DM client towards the server.
- o For deployments where the devices use a different DM protocol from the server, the gateway can represent the devices and translate between the different protocols (e.g. TR-069, OMA-DM, or CoAP). The devices can be represented either as virtual devices or as part of the gateway

4 Explain information model and function model with example

4.2 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).

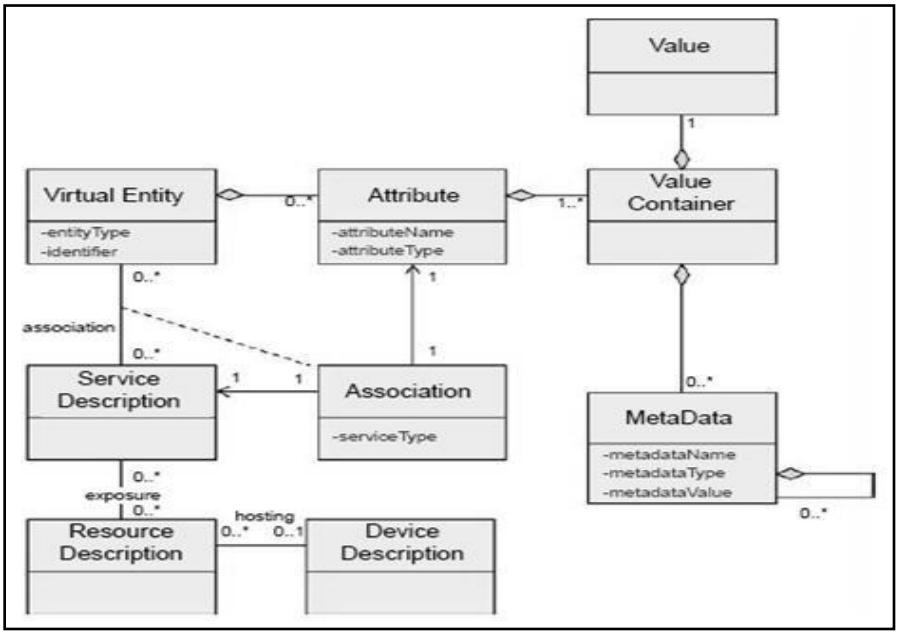


Figure 4.18: High –level Information Model

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.

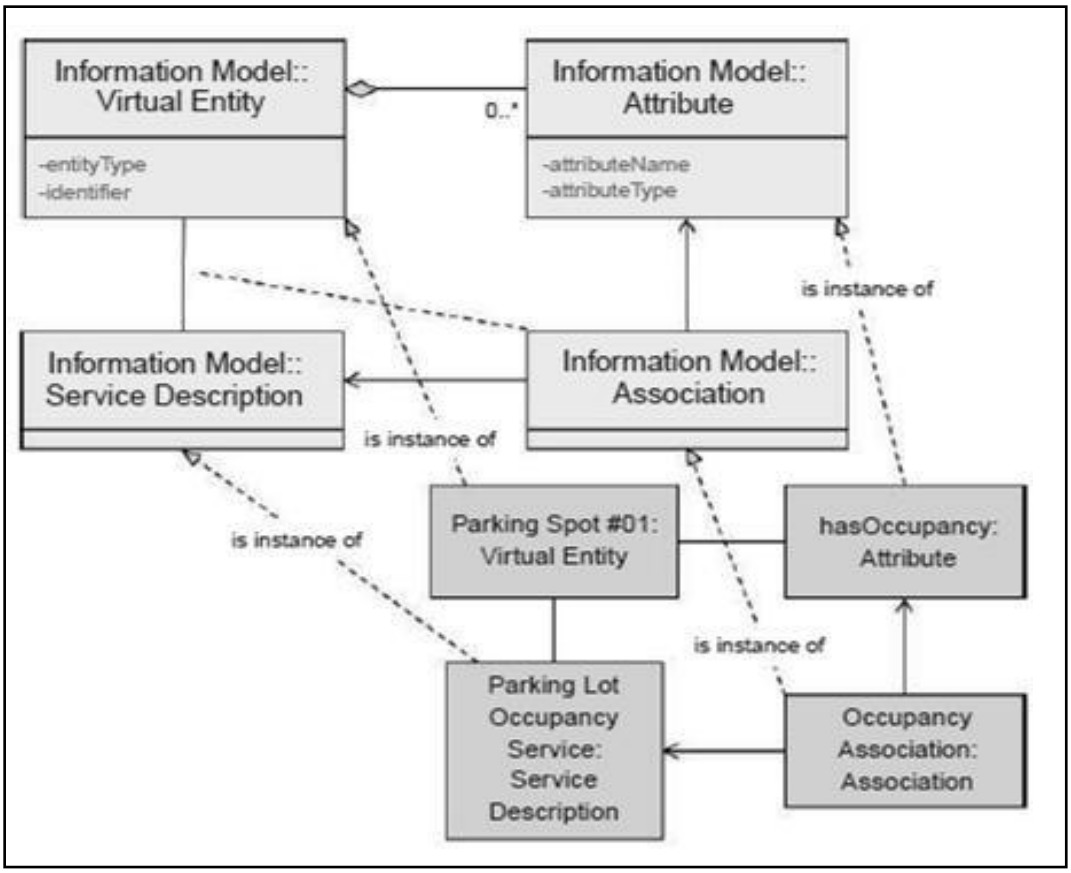
As seen from the IoT Domain Model, a Virtual Entity is associated with Resources that expose Services about the specific Virtual Entity. This association between a Virtual Entityand its Services is captured in the Information Model with the explicit class called Association.

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.

4.3 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

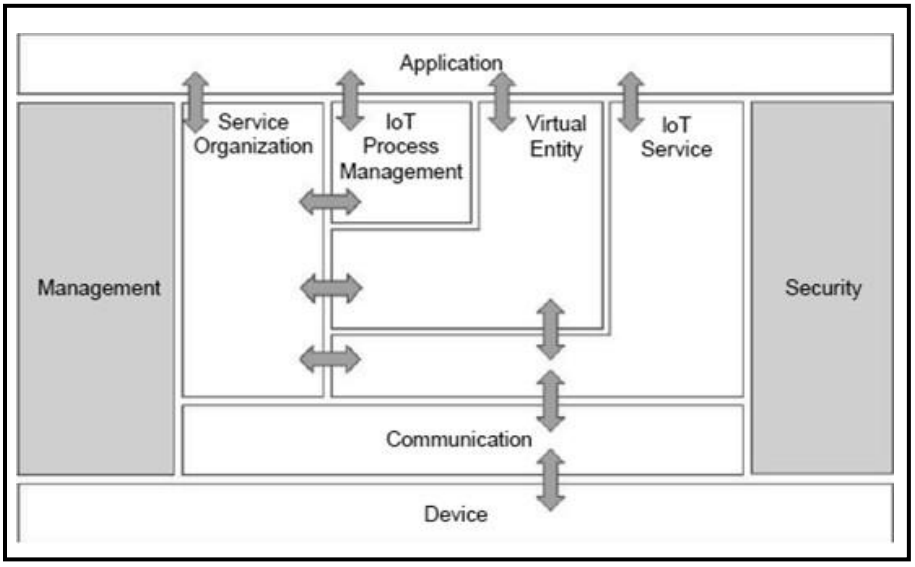


Figure 4.21 : IoT-A Functional Model

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.

4.3.1 : 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.

4.3.2 : 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.

4.3.3 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.

4.3.4 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.

4.3.5 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.

4.3.6 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.

4.3.7 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.

4.3.8 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.

4.3.9 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.

4.3.10 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.

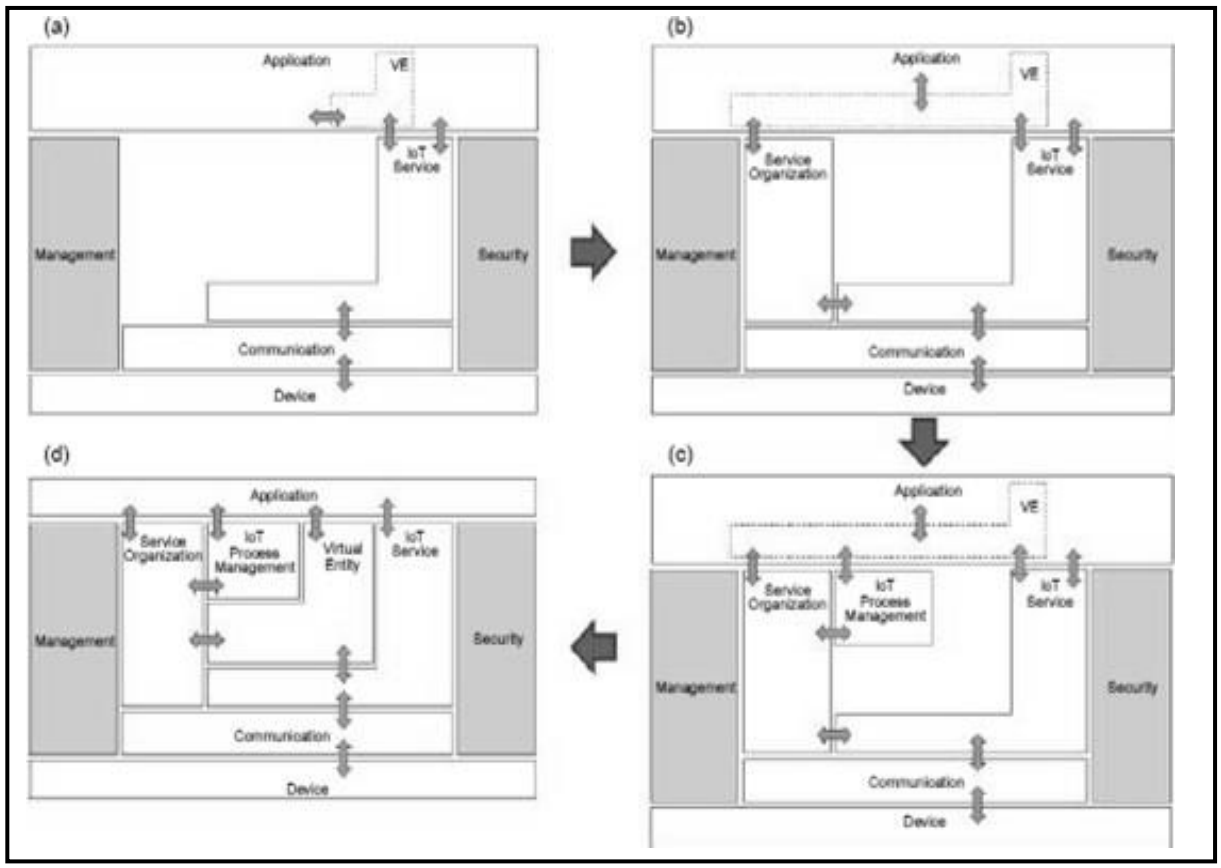


Figure 4.22: Building Progressively complex IoT Systems

5 With a neat diagram, Explain the IoT Reference Model.

Explain Basic and Advanced device types along with Gateways Basic Devices

Basic Devices

Devices that only provide the basic services of sensor readings and/or actuation tasks, and in some cases limited support for user interaction. LAN communication is supported via wired or wireless technology, thus a gateway is needed to provide the WAN connection.

- Intended for a single purpose - measuring air pressure or closing a valve. Some cases several functions are deployed on the same device- monitoring humidity, temperature, and light level.
- The requirements on hardware are low, both in terms of processing power and memory.
- The main focus is on keeping the bill of materials (BOM) as low as possible by using inexpensive microcontrollers with built-in memory and storage, often on an SoC integrated circuit with all main components on one single chip.
- The microcontroller hosts a number of ports that allow integration with sensors and actuators, such as General Purpose I/O (GPIO) and an analog-to-digital converter (ADC) for supporting analog input.
- For certain actuators, such as motors, pulse-width modulation (PWM) can be used.

Advanced devices

In this case the devices also host the application logic and a WAN connection. They may also feature device management and an execution environment for hosting multiple applications. Gateway devices are most likely to fall into this category.

- The distinction between basic devices, gateways, and advanced devices is not cut in stone, but some features that can characterize an advanced device are the following:
- A powerful CPU or microcontroller with enough memory and storage to host

3 advancedapplications, such as a printer offering functions for copying, faxing, printing, and remote management.

- A more advanced user interface with, for example, display and advanced user input in the form of a keypad or touch screen.
- Video or other high bandwidth functions.
- A gateway serves as a translator between different protocols, e.g. between IEEE 802.15.4 or IEEE 802.11, to Ethernet or cellular.
- There are many different types of gateways, which can work on different levels in the protocol layers. ost often a gateway refers to a device that performs translation of the physical and link layer, but application layer gateways (ALGs) are also common. The latter is preferably avoided because it adds complexity and is a common source of error in deployments.
- Some examples of ALGs include the ZigBee Gateway Device (ZigBee Alliance 2011), which translates from ZigBee to SOAP and IP, or gateways that translate from Constrained Application Protocol (CoAP) to Hyper Text Transfer Protocol/Representational State Transfer (HTTP/REST).
- For some LAN technologies, such as 802.11 and Z-Wave, the gateway is used for inclusion and exclusion of devices.
- This typically works by activating the gateway into inclusion or exclusion mode and by pressing a button on the device to be added or removed from the network.
- For very basic gateways, the hardware is typically focused on simplicity and low cost, but frequently the gateway device is also used for many other tasks, such as data management, device management, and local applications. In these cases, more powerful hardware with GNU/Linux is commonly used.
- The following sections describe these additional tasks in more detail.

- 3.1.3.1 Data Management
- Typical functions for data management include performing sensor readings and caching this data, as well as filtering, concentrating, and aggregating the data before transmitting it to back-end servers.
- 3.1.3.2 Local applications
- Examples of local applications that can be hosted on a gateway include closed loops, home alarm logic, and ventilation control, or the data management.
- The benefit of hosting this logic on the gateway instead of in the network is to avoid downtime in case of WAN connection failure, minimize usage of costly cellular data, and reduce latency. The execution environment is responsible for the lifecycle management of the applications, including installation, pausing, stopping, configuration, and uninstallation of the applications.
- A common example of an execution environment for embedded environments is OSGi, which is based on java applications are built as one or more Bundles, which are packaged as Java JAR files and installed using a so-called Management Agent. The Management Agent can be controlled from, for example, a terminal shell or via a protocol such as CPE WAN Management Protocol (CWMP).
- Bundle packages can be retrieved from the local file system or over HTTP.
- The benefit of versioning and the lifecycle management functions is that the OSGi environment never needs to be shut down when upgrading, thus avoiding downtime in the system.
- 3.1.3.3 Device Management
- Device management (DM) is an essential part of the IoT and provides efficient means to perform many of the management tasks for devices:
- o Provisioning: Initialization (or activation) of devices in regards to configuration and features to be enabled.
- o Device Configuration: Management of device settings and parameters.
- o Software Upgrades: Installation of firmware, system software, and applications on the device.
- o Fault Management: Enables error reporting and access to device status
- In the simplest deployment, the devices communicate directly with the DM server. This is, however, not always optimal or even possible due to network or protocol constraints, e.g. due to a firewall or mismatching protocols.
- In these cases, the gateway functions as mediator between the server and the devices, and can operate in three different ways:
- o If the devices are visible to the DM server, the gateway can simply forward the messages between the device and the server and is not a visible participant in the session.
- o In case the devices are not visible but understand the DM protocol in use, the gateway can act as a proxy, essentially acting as a DM server towards the device and a DM client towards the server.
- o For deployments where the devices use a different DM protocol from the server, the gateway can represent the devices and translate between the different protocols (e.g. TR-069, OMA-DM, or CoAP). The devices can be represented either as virtual devices or as part of the gateway

7 With diagram explain CRISP-DM process in detail.

The phases in the CRISP-DM process model are described in Figure 5.15, which is followed by descriptions of each of the phases. These are illustrated using an example from Predictive Maintenance (PdM) for pump stations in a water distribution network. Although the figure indicates a certain order between the phases, analytics is an iterative process, and it's expected that you will have to move back and forth between the phases to a certain extent.

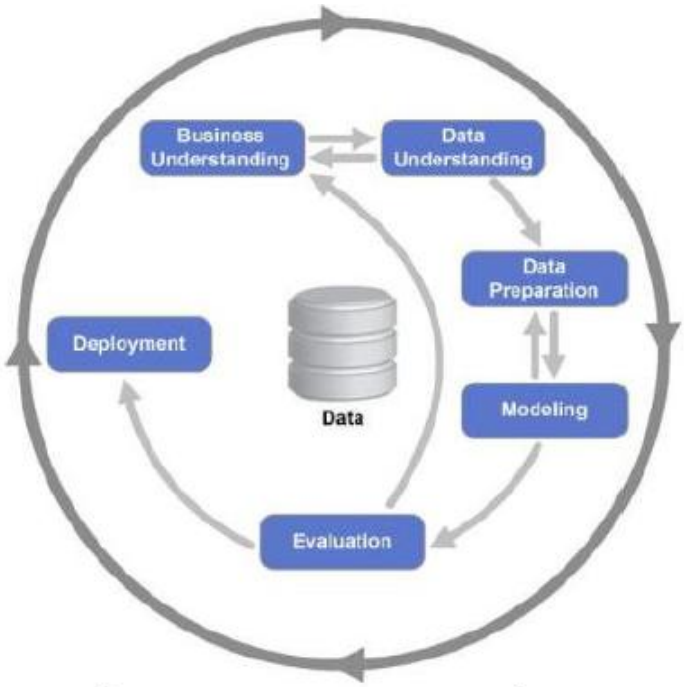


Figure 5. 15 CRISP-DM Process Diagram.

Business Understanding

 The first phase in the process is to understand the business objectives and requirements, as well as success criteria. This forms the basis for formulating the goals and plan for the data mining process.

 Many organizations may have a feeling that they are sitting on valuable data, but are unsure how to capitalize on this. In these cases, it's not unusual to bring in the help of an analytics team to identify potential business cases that can benefit from the data.

Data Understanding

 The next phase consists of collecting data and gaining an understanding of the data properties, such as amount of data and quality in terms of inconsistencies, missing data, and measurement errors. The tasks in this phase also include gaining some understanding of actionable insights contained in the data, as well as to form some basic hypotheses.

Data Preparation

 Before it's possible to start modeling the data to achieve our goals, it's necessary to prepare the data in terms of selection, transformation, and cleaning. In this phase, it's frequently the case that new data is necessary to construct, both in terms of entirely new attributes as well as imputing new data into records where data is missing.

 It's quite common for this phase to consume more than half the time of a project.

Modeling

 At the modeling phase, it's finally time to use the data to gain an understanding of the actual business problems that were stated in the beginning of the project. Various modeling techniques are usually applied and evaluated before selecting which ones are best suited for the particular problem at hand. As some modeling techniques require data in a specific form, it's quite common to go back to the data preparation phase at this stage. This is an example of the iterativeness of CRISP-DM and analytics in general.

 After evaluating a number of models, it's time to select a set of candidate models to be methodically assessed. The assessment should estimate the effectiveness of the results in terms of accuracy, as well as ease of use in terms of interpretation of the results. If the assessment shows that we have found models that meet the necessary criteria, it's time for a more thorough evaluation, otherwise the work on finding suitable models has to continue.

Evaluation

 Now the project is nearing its end and it's time to evaluate the models from a business perspective using the success criteria that were defined at the beginning of the project. It is also customary to spend some time reviewing the project and draw conclusions about what was good and bad. This will be valuable input for future projects. At the end of the evaluation phase, a decision whether to deploy the results or not should be made.

Deployment

 At this last phase in the project, the models are deployed and integrated into the organization. This can mean several things, such as writing a report to disseminate the results, or integrating the model into an automated system. This part of the project involves the customer directly, who has to provide the resources needed for an effective deployment. The deployment phase also includes planning for how to monitor the models and evaluate when they have played out their role or need to be maintained. As last steps, a final report and project review should be performed.

8 **A) Write short note on (i) data (ii) information (iii) knowledge**

Data: Data refers to "unstructured facts and figures that have the least impact on the typical manager". data includes both useful and irrelevant or redundant facts, and in order to become meaningful, needs to be processed.

 Information: Within the context of IoT solutions, information is data that has been contextualized, categorized, calculated, and condensed. This is where data has been carefully curated to provide relevance and purpose for the decision- makers in question. The majority of ICT solutions can be viewed as either storing information or processing data to become information.

 Knowledge: Knowledge, meanwhile, relates to the ability to understand the information presented, and using existing experience, the application of it within a certain decision making context.

B) Explain Knowledge Reference Architecture for M2M and IoT with diagram.

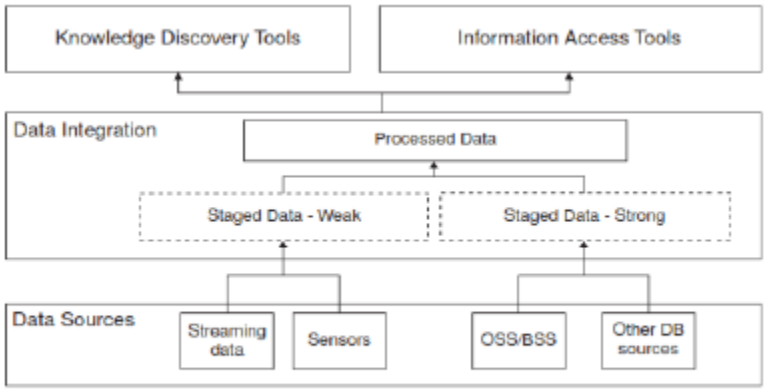


Figure No 5.17 Knowledge Reference Architecture for M2M and IoT.

Figure 5.17 outlines a high-level knowledge management reference architecture that illustrates how data sources from M2M and IoT may be combined with other types of data, for example, from databases or even OSS/ BSS data from MNOs. There are three levels to the diagram: (1) data sources, (2) data integration, and (3) knowledge discovery and information access.

 Data sources

Data sources refer to the broad variety of sources that may now be available to build enterprise

solutions.

 Data integration

The data integration layer allows data from different formats to be put together in a manner that can be used by the information access and knowledge discovery tools.

 Staged Data: Staged data is data that has been abstracted to manage the rate at which it is received by the analysis platform. Essentially,"staged data" allows the correct flow of data to reach information access and knowledge discovery tools to be retrieved at the correct time. Big data and M2M analytics were discussed in detail in here we focus on the data types required for staging the data appropriately for knowledge frameworks. There are two main types of data: weak data and strong data. This definition is in order to differentiate between the manner in which data is encoded and its contents _ for example, the difference between XML and free text.

 Strong Type Data: Strong type data refers to data that is stored in traditional database formats, i.e. it can be extracted into tabular format and can be subjected to traditional database analysis techniques. Strong data types often have the analysis defined beforehand, e.g. by SQL queries written by developers towards a database.

 Weak Type Data: Weak type data is data that is not well structured according to traditional database techniques. Examples are streaming data or data from sensors. Often, this sort of data has a different analysis technique compared to strong type data. In this case, it may be that the data itself defines the nature of the query, rather than being defined by developers and created in advance. This may allow insights to be identified earlier than in strong type data.

 Processed data

Processed data is combined data from both strong and weak typed data that has been combined within an IoT context to create maximum value for the enterprise in question. There are various means by which to do this processing _ from stripping data separately and creating relational tables from it or pooling relevant data together in one combined database for structured queries. Examples could include combining the data from people as they move around the city from an operator's business support system with sensor data from various buildings in the city. A health service could then be created analyzing the end-users routes through a city and their overall health _ such a system may be used to more deeply assess the role that air pollution may play in health factors of the overall population.

9 Discuss elaborately about the term – XaaS

There is a general trend away from locally managing dedicated hardware toward cloud infrastructures that drives down the overall cost for computational capacity and storage. This is commonly referred to as "cloud computing."

Cloud computing is a model for enabling ubiquitous, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be provisioned, configured, and made available with minimal management effort or service provider interaction.

Cloud computing, however, does not change the fundamentals of software engineering. All applications need access to three things: compute, storage, and data processing capacities. With cloud computing, a fourth element is added _ distribution services _ i.e. the manner in which the data and computational capacity are linked together and coordinated.

A cloud-computing platform may therefore be viewed conceptually (Figure 5.11). Several essential

characteristics of cloud computing have been defined as follows:

o On-Demand Self-Service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed, or automatically, without requiring human interaction with each service provider.
o Broad Network Access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g. mobile phones,tablets, laptops, and workstations).
o Resource Pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g. country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
o Rapid Elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited, and can be appropriated in any quantity at any time.
o Measured Service. Cloud systems automatically control and optimize resource use by leveraging a metering capability, at some level of abstraction, appropriate to the type of service (e.g. storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
Once such infrastructures are available, however, it is easier to deploy applications in software. For M2M and IoT, these infrastructures provide the following:
1. Storage of the massive amounts of data that sensors, tags, and other "things" will produce.
2. Computational capacity in order to analyze data rapidly and cheaply.
3. Over time, cloud infrastructure will allow enterprises and developers to share datasets, allowing for rapid creation of information value chains.
Cloud computing comes in several different service models and deployment options for enterprises wishing to use it. The three main service models may be defined as
o Software as a Service (SaaS): Refers to software that is provided to consumers on demand, typically via a thin client. The end-users do not manage the cloud infrastructure in any way. This is handled by an Application Service Provider (ASP) or Independent Software Vendor (ISV). Examples include office and messaging software, email, or CRM tools housed in the cloud. The end-user has limited ability to change anything beyond user-specific application configuration settings.
o Platform as a Service (PaaS): Refers to cloud solutions that provide both a computing platform and a solution stack as a service via the Internet. The customers themselves develop the necessary software using tools provided by the provider, who also provides the networks, the storage, and the other distribution services required. Again, the provider manages the underlying cloud infrastructure, while the customer has control over the deployed applications and possible settings for the application-hosting environment o Infrastructure as a Service (IaaS): In this model, the provider offers virtual machines and other resources such as hypervisors (e.g. Xen, KVM) to customers. Pools of hypervisors support the virtual machines and allow users to scale resource usage up and down in accordance with their computational requirements. Users install an OS image and application software on the cloud infrastructure. The provider manages the underlying cloud infrastructure, while the customer has control over OS, storage, deployed applications, and possibly some networking components.

o Deployment Models:

Private Cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g. business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
Community Cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g. mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
Public Cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination thereof. It exists on the premises of the cloud provider.
Hybrid Cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g. cloud bursting for load balancing between clouds).

10 Describe key characteristics and management of M2M data.

These are the communications technologies that are considered to be critical to the realization of massively distributed M2M applications and the IoT at large.
o Power Line Communication
6 | P a g e
(PLC) refers to communicating over power (or phone, coax, etc.) lines. This amounts to pulsing, with various degrees of power and frequency, the electrical lines used for power distribution. PLC comes in numerous flavors. At low frequencies (tens to hundreds of Hertz) it is possible to communicate over kilometers with low bit rates (hundreds of bits per second).
Typically, this type of communication was used for remote metering, and was seen as potentially useful for the smart grid. Enhancements to allow higher bit rates have led to the possibility of delivering broadband connectivity over power lines.
o LAN (and WLAN)
Continues to be important technology for M2M and IoT applications.
This is due to the high bandwidth, reliability, and legacy of the technologies. Where power is not a limiting factor, and high bandwidth is required, devices may connect seamlessly to the Internet via Ethernet (IEEE 802.3) or Wi-Fi (IEEE 802.11). The utility of existing (W) LAN infrastructure is evident in a number of early IoT applications targeted at the consumer market, particularly where integration and control with smartphones is required.
o Bluetooth Low Energy
(BLE; “Bluetooth Smart”) is a recent integration of Nokia’s Wibree standard with the main Bluetooth standard. It is designed for short-range (,50 m) applications in healthcare, fitness, security, etc., where high data rates (millions of bits per second) are required to enable application functionality. It is deliberately low cost and energy efficient by design, and has been integrated into the majority of recent smartphones.
o Low-Rate, Low-Power Networks
are another key technology that form the basis of the IoT.
o IPv6 Networking
making the fact that devices are networked, with or without wires, with various capabilities in terms of range and bandwidth, essentially seamless.
It is foreseeable that the only hard requirement for an embedded device will be that it can somehow

connect with a compatible gateway device.

o 6LoWPAN

(IPv6 Over Low Power Wireless Personal Area Networks) was developed initially by the LoWPAN Working Group (WG) of the IETF

The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices", and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things

o RPL

IPv6 Routing Protocol for Low-Power and Lossy Networks. Abstract Low-Power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained. LLN routers typically operate with constraints on processing power, memory, and energy (battery power).

o CoAP

Constrained Application Protocol (CoAP) is a protocol that specifies how low-power computeconstrained devices can operate in the internet of things (IoT).