Internal Assesment Test –3, January 2022								
INTERNET OF THINGS	**Code:**	**18MCA542**						
25-01-2022	Duration:	90 mins	Max Marks:	50	Sem	V		MCA

1) Illustrate Internet Engineering Task Force architecture fragments in detail (10)

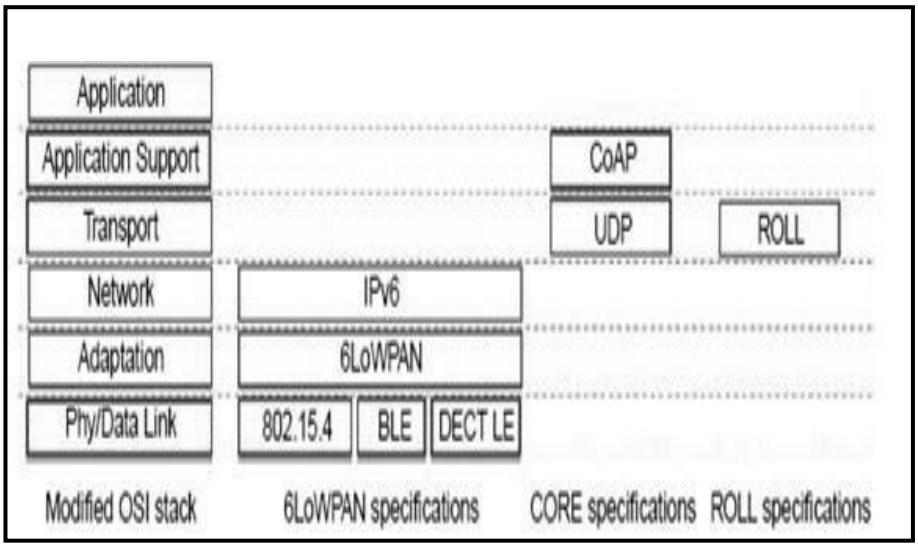

Figure 4.7: IETF Working Groups and Specification Scope

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

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

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

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

The IETF CoRE working group has also produced a draft specification for a Resource Directory. A Resource Directory is a CoAP server resource (/rd) that maintains a list of resources, their corresponding server contact information (e.g. IP addresses or fully qualified

domain name, or FQDN), their type, interface, and other information similar to the information that the CoRE Link Format document specifies (Figure 4.8a).

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

A Mirror Server (Vial 2012) is a rendezvous mechanism for CoAP Server resource presentations. A Mirror Server is a CoAP Server resource (/ms) that maintains a list of resources and their cached representations (Figure 4.8b). A CoAP Server registers its resources to the Mirror Server, and upon registration a new mirror server resource is created on the Mirror Server with a container (mirror representation) for the original server representation. The original CoAP Server updates the mirror representation either periodically or when the representation changes. A CoAP Client that retrieves the mirror representation receives the latest updated representation from the original CoAP Server.
The Mirror Server is useful when the CoAP Server is not always available for direct access.

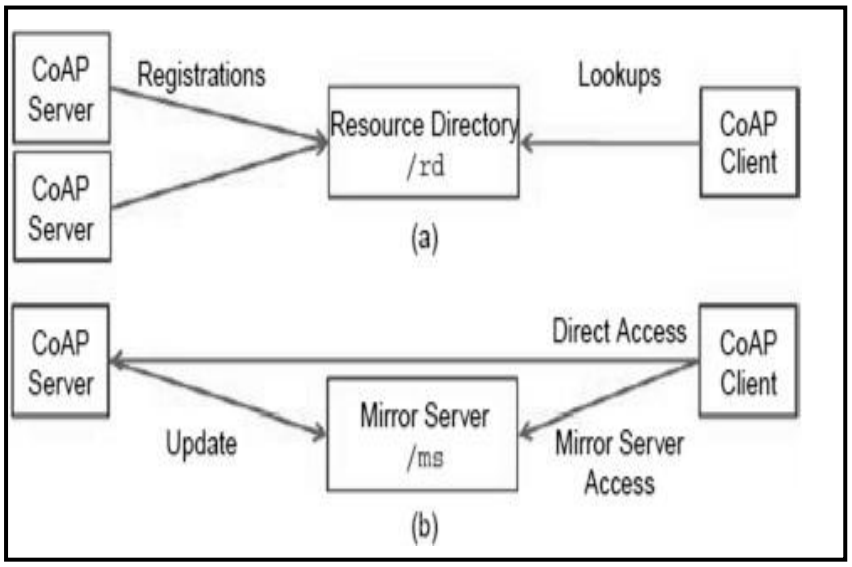

Figure 4.8: IETF CoRE Functional Components a) Resource Directory b) Mirror Server

The figure 4.9 The interworking issues appear when an HTTP Client accesses a CoAP Server through an HTTP-CoAP proxy or when a CoAP Client accesses an HTTP Server through a CoAP-HTTP proxy (Figure 4.9a).
The mapping process is not straightforward for a number of reasons. The main is the different transport protocols used by the HTTP and CoAP: HTTP uses TCP while CoAP uses UDP. The guidelines focus more on the HTTP-to-CoAP proxy and recommend addressing schemes (e.g.

how to map a CoAP resource address to an HTTP address),mapping between HTTP and CoAP response codes, mapping between different media types carried in the HTTP/CoAP payloads, etc.

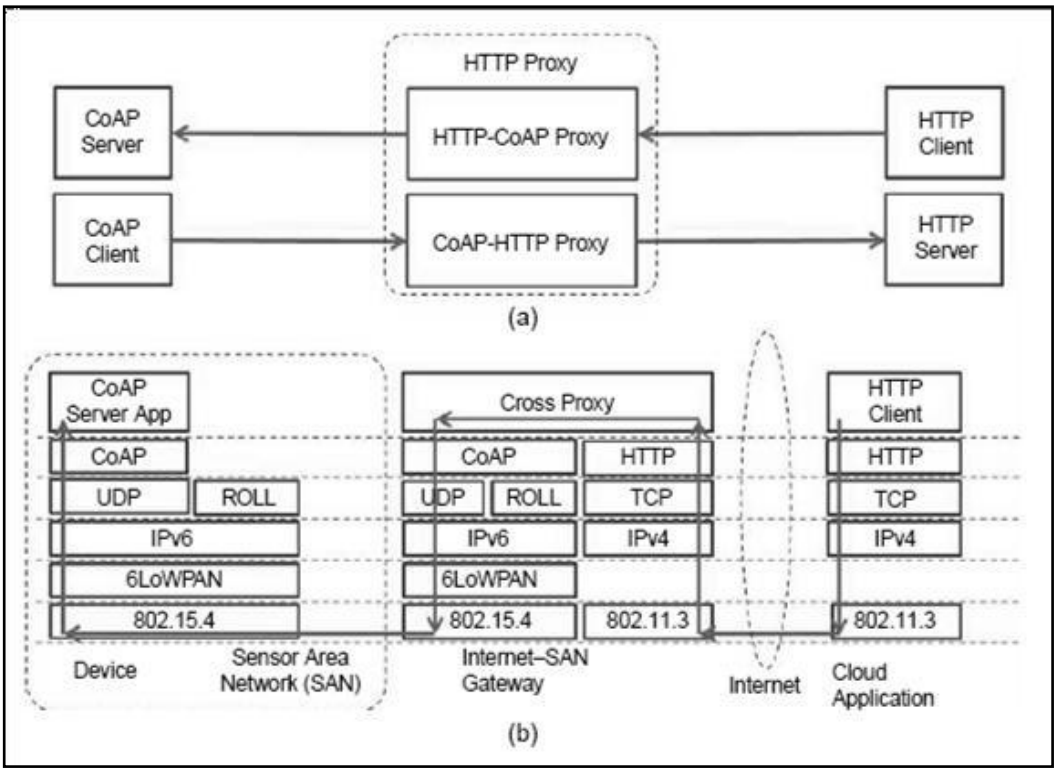

Figure 4.9: IETF CoRE HTTP Proxy

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

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

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

2) Explain safety, privacy and trust module with respect to IOT.

4.1.2 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.

4.1.3 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.

4.1.4 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.

3) Explain Data representation and visualization with example

Each IoT application has an optimal visual representation of the data and the system. Data that is generated from heterogeneous systems has heterogeneous visualization requirements. There are currently no satisfactory standard data representation and storage methods that satisfy all of the potential IoT applications.

Data-derivative products will have further ad hoc visualization requirements. A derivative in these terms exists once a function has been performed on an initial data set _ which may or may not be raw data. These can be further integrated at various levels of abstraction, depending on the logic of the integrator. New information sources, such as those derived from integrated data streams from various logically correlated IoT applications, will present interesting representation and visualization challenges.

4)Explain the IoT Reference Model and IOT reference architecture with suitable diagram

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.

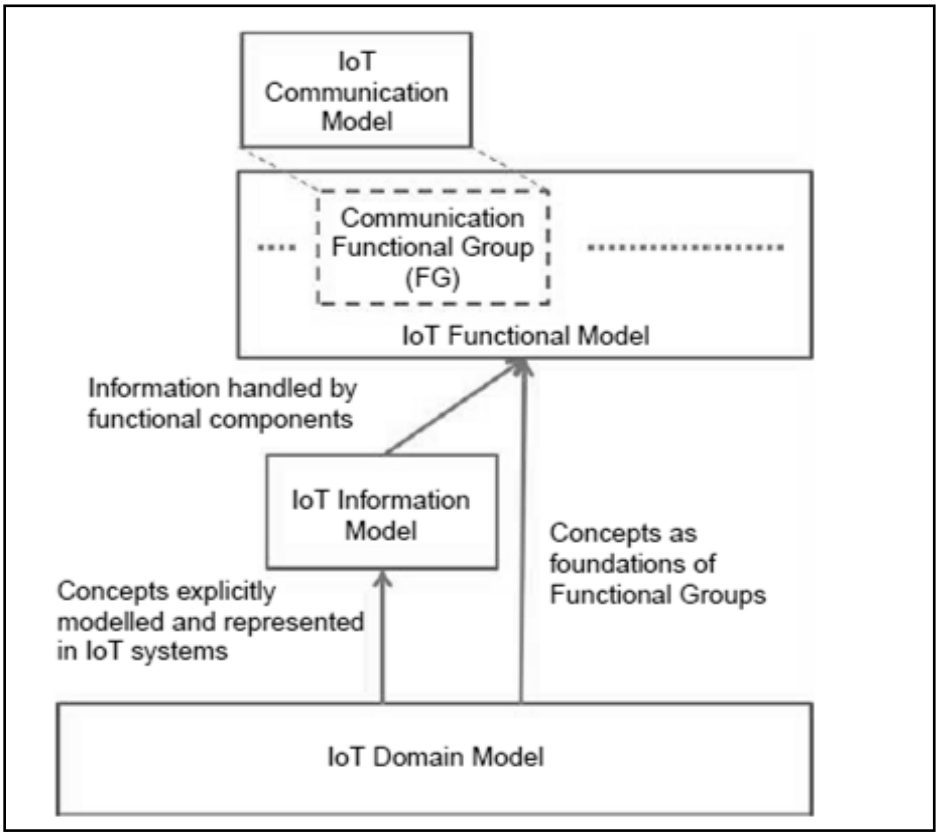

Figure 4.11: IoT Reference Model.

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.

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

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

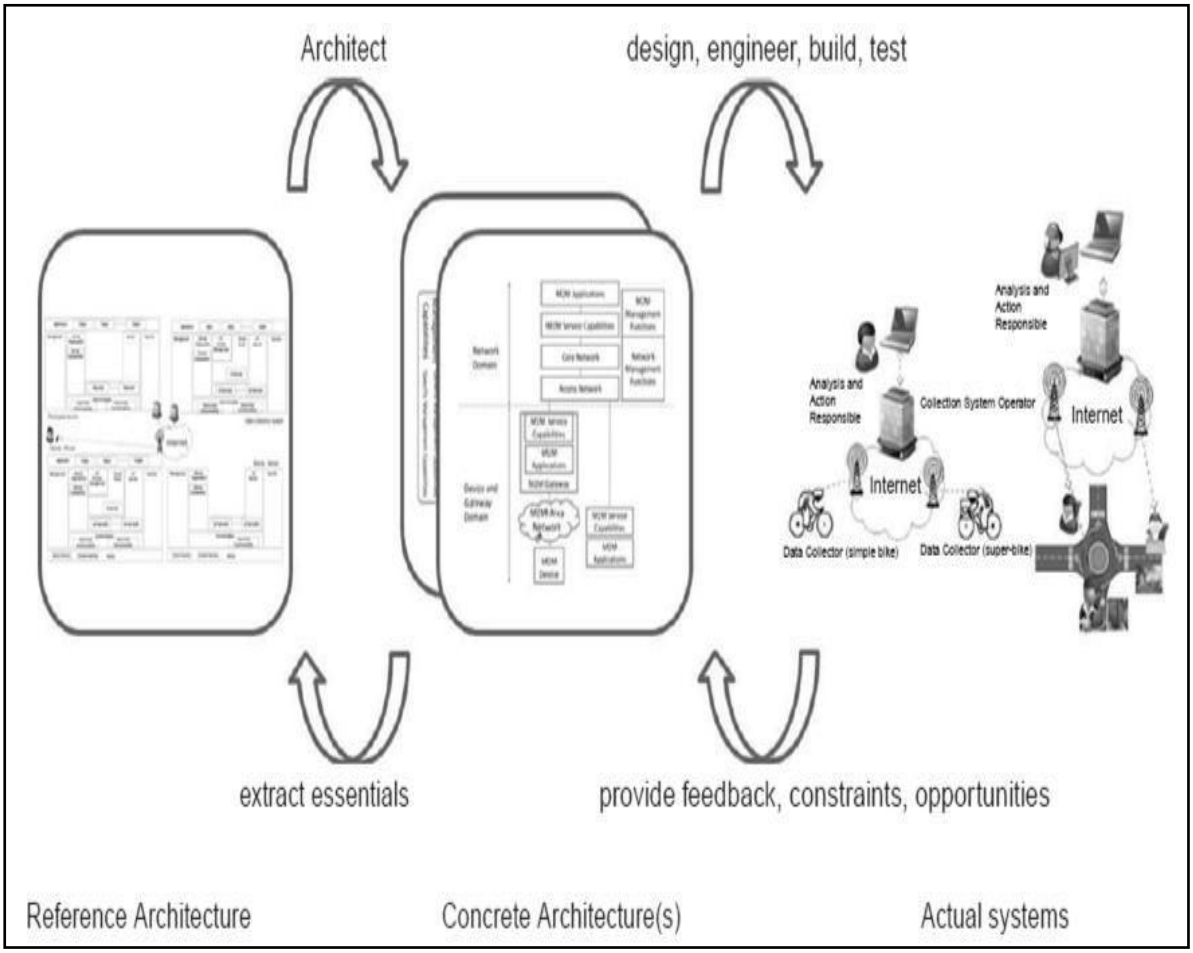

Figure 4.12: From reference to concrete architectures and actual systems.

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

an IoT ARM, and (b) how to use it with respect to building actual systems.

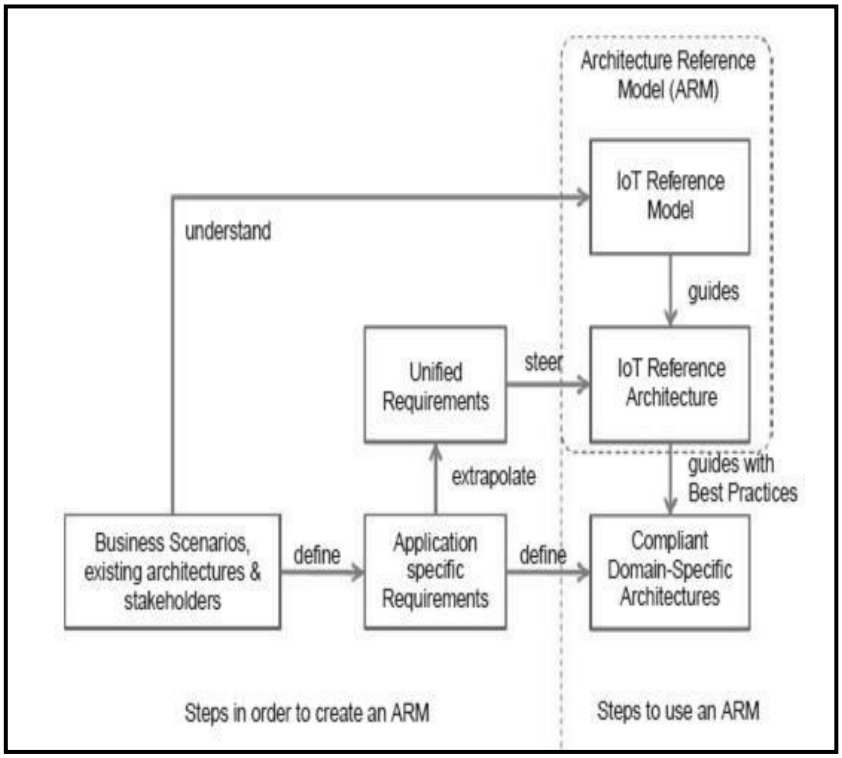

Figure 4.13: IoT Reference Model and Reference Architecture dependencies

5) Explain IMC-AESOP web of things to cloud of things

The IMC-AESOP project is a European research and development project

- The vision of SOCRADES – WS enabled devices
- 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)
- Empower the rapid development of better and more efficient next generation industrial applications
- Satisfying the agility required by modern enterprises.
- This vision is only realizable due to the distributed, autonomous, intelligent, pro-active, fault-tolerant, reusable (intelligent) systems, which expose their capabilities, functionalities, and structural characteristics as services located in a "service cloud."

- Today factories composed and structured by several systems viewed and interacting in a hierarchical fashion
- 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,

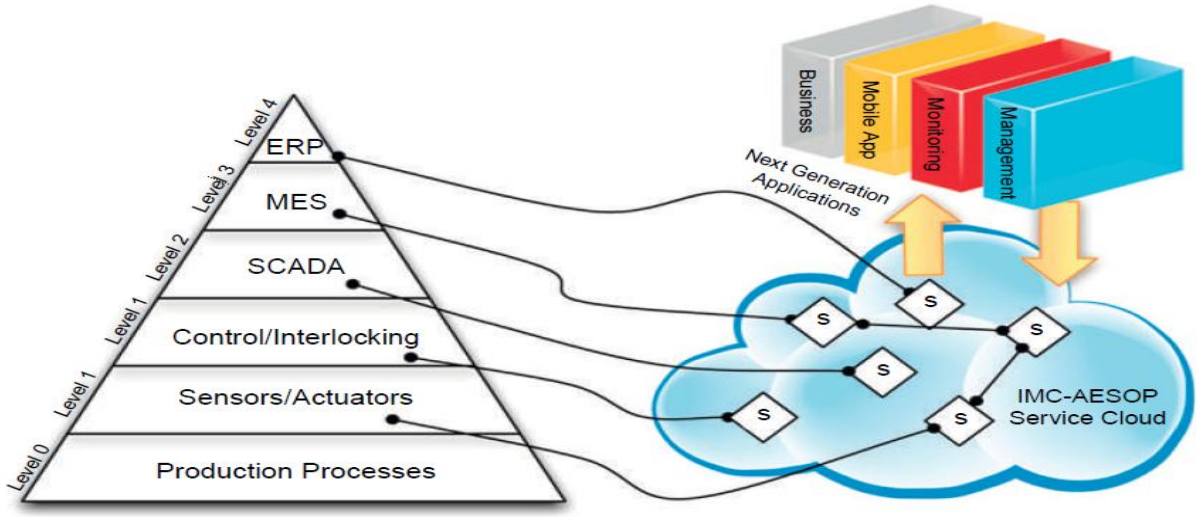

FIGURE 11.5

Future industrial system view of cloud-based composition of cyber-physical services.

- Information-based architecture that depends on a big variety of services exposed by the cyber- physical 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 in the cloud) to realize their goals.

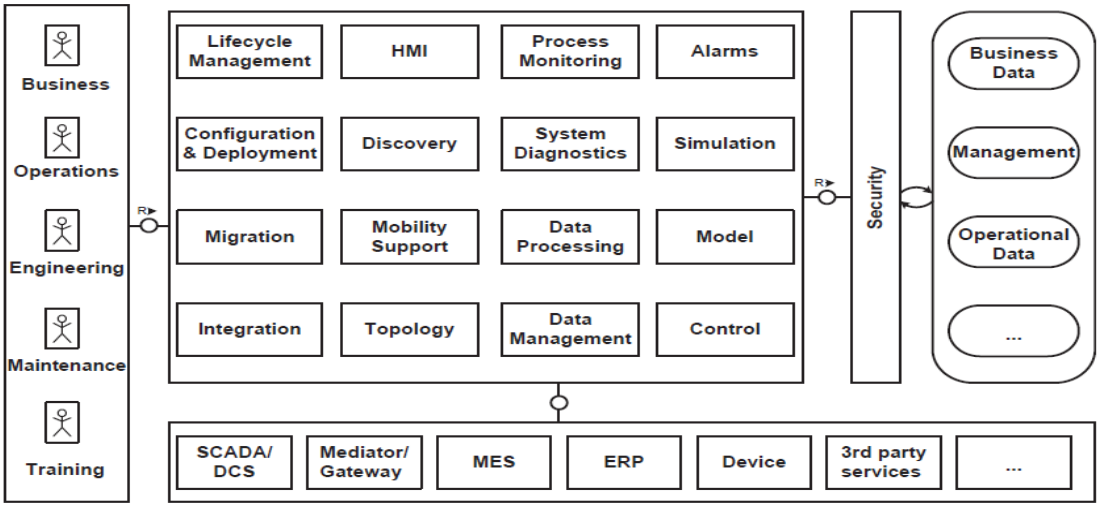

FIGURE 11.6

IMC-AESOP cloud-based architecture vision.

- Several "user roles" will interact with the envisioned architecture 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.
- 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.
- 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.
- illustrate 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.

- Outsourcing key functionalities to the cloud is challenging, and further research needs to be done towards the applicability of it for several scenarios.
- For instance, monitoring scenarios have been successfully realized demonstrating the benefits of the IT concept prevalence to traditional industrial system design and operation.
- However, several aspects with relation to real-time interactions, reliability, and resilience, as well as control (especially closed-loop control), are still at an early stage.
- Constructing such complex systems additionally bears challenges with respect to safety, maintainability, and security.

6) With neat diagram Illustrate the relationship between IOT information and domain model

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

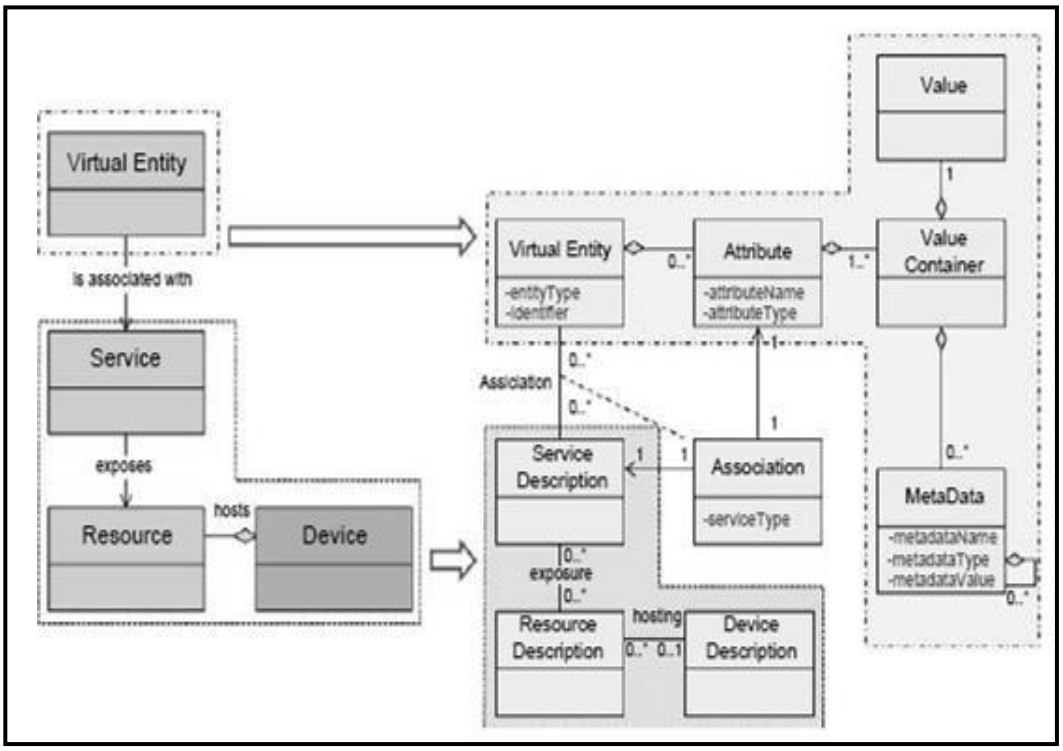

Figure 4.20: Relationship between core concepts of IoT Domain Model and IoT information Model

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

- Location and its temporal information are important because Physical Entities represented by Virtual Entities exist in space and time. These properties are extremely important when the interested Physical Entities are mobile (e.g. a moving car).
- Even non-moving Virtual Entities contain properties that are dynamic with time, and therefore their temporal variations need to be modelled and captured by an information model.
- Information such as ownership is also important in commercial settings because it may determine access control rules or liability issues.

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

- Service type, which denotes the type of service, such as Big Web Service or RESTful Web Service. The interfaces of a service are described based on the description language for each service type, for example, Web Application Description Language (WADL) for RESTful Web Services, Web Services Description Language (WSDL).
- Service area and Service schedule are properties of Services used for specifying the

geographical area of interest for a Service and the potential temporal availability of a Service, respectively.

- Associated resources that the Service exposes.
- Metadata or semantic information used mainly for service composition. This is information such as the indicator of which resource property is exposed as input or output, whether the execution of the service needs any conditions satisfied before invocation, and whether there are any effects of the service after invocation.

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

a) Resource name and identifier for facilitating resource discovery.
b) 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.
c) Free text attributes or tags used for capturing typical manual input such as “fire alarm, ceiling.”
d) Indicator of whether the resource is an on-Device resource or network resource.
e) Location information about the Device that hosts this resource in case of an on-Device resource.
f) Associated Service information.
g) Associated Device description information.

7) Explain about Function View along with it Function groups

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

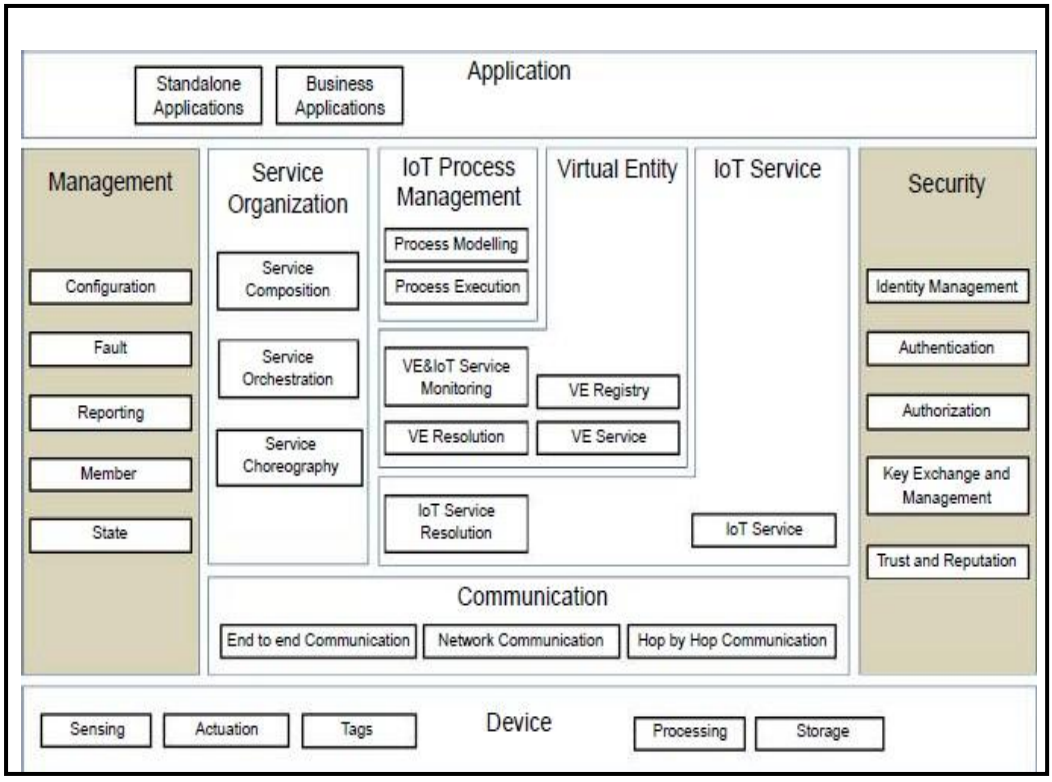

Figure 4.23: IoT Functional View

4.1.5 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.

4.1.6 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.

4.1.7 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.

4.1.8 Virtual Entity Functional group

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

The Following FCs are defined for realizing these functionalities:

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

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

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

4.1.9 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.

4.1.10 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.

4.1.11 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.

4.1.12 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.

8. Explain in detail about Information View

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

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.

4.1.13 Information flow and lifecycle

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

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

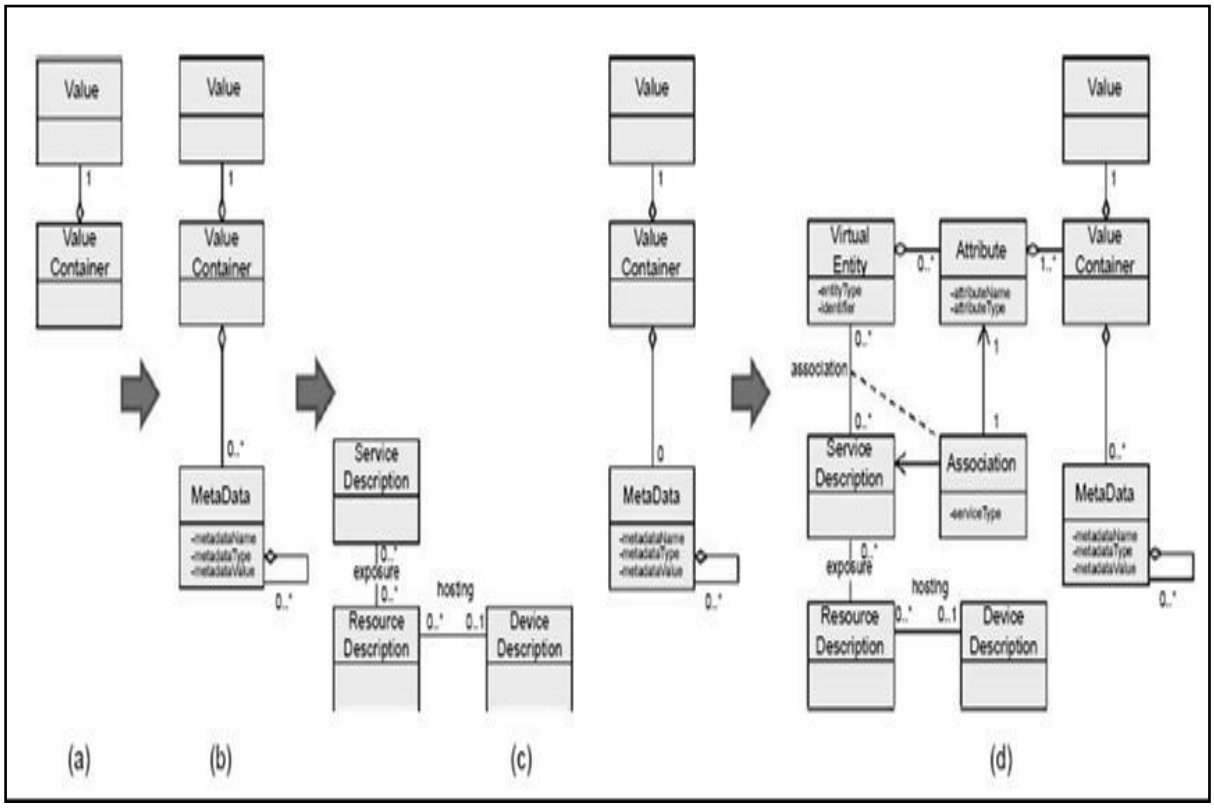

Figure 4.24: Information Exchange Patterns

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

4.1.14 Information Handling

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

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

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

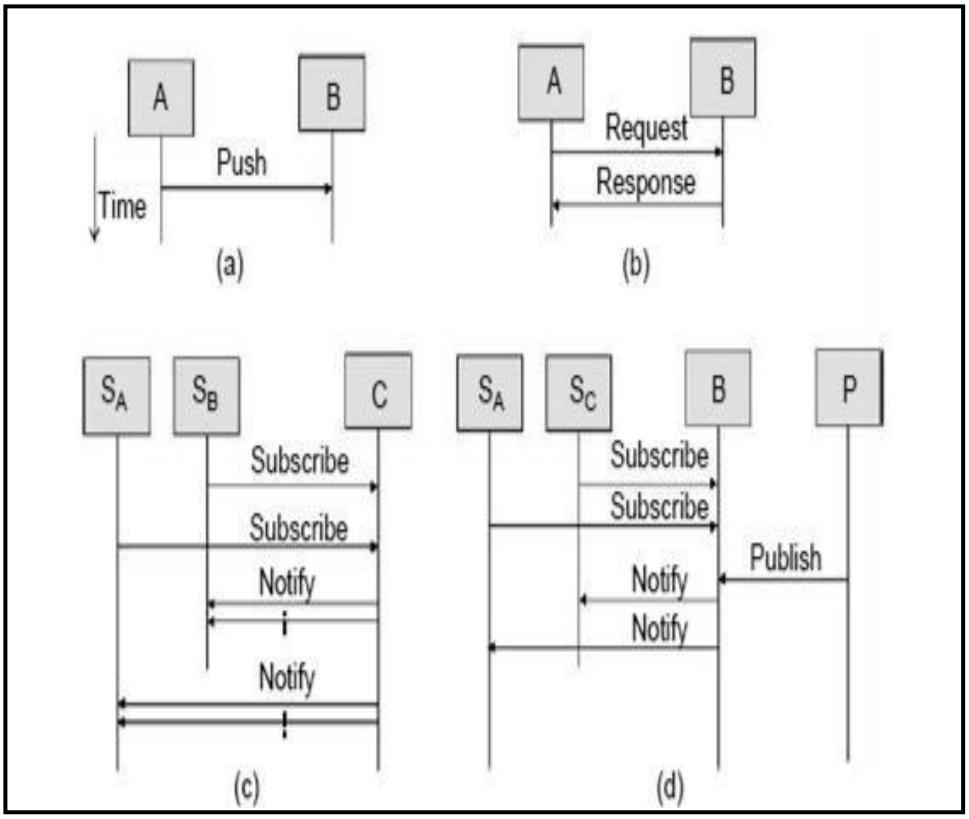

Figure 4.25: Information Exchange Patterns

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

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

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

9. Summarize SOCRADES Integration Architecture (SIA) with neat diagram

Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices

- Agility and flexibility are required from modern factories.
- Rapid advances in Information Technology (IT), both in hardware and software, as well as the increasing level of dependency on cross-factory functionalities, sets new challenging goals for future factories.
- Rely on a large ecosystem of systems where collaboration at large scale will take place.
- Mashing up services has proven to be a key advantage in the Internet application area; and if now the devices can either host web services natively or be represented as such in higher systems, then existing tools and approaches can be used to create mash-up apps that depend on these devices.
- A visionary project that followed this line of thinking was the Industry-driven European Commission funded project SOCRADES.
- Driven by the key need for cross-layer M2M collaboration (i.e. at shop-floor level among various heterogeneous devices as well as among systems and services up to the Enterprise (ERP) level), an architecture had been proposed, prototyped (Colombo et al. 2010),
- SOCRADES proposed and realized SOA-based integration, as shown in Figure 11.2, including migration of existing infrastructure via gateways and service mediators, as shown in Figure 11.3.

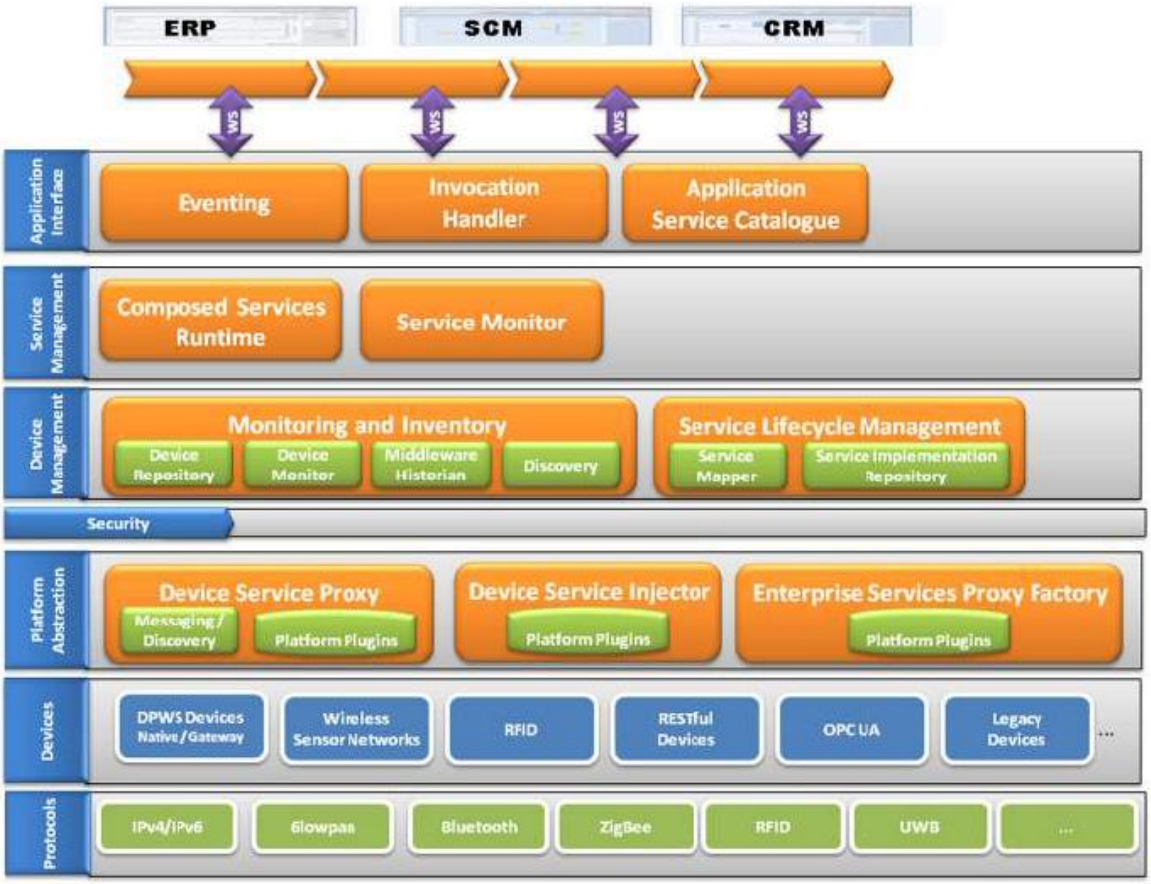

FIGURE 11.4

The SOCRADES Integration Architecture (SIA) enabling the coupling of (industrial) machines at shop-floor and enterprise systems.

- **Application Interface**: This part enables the interaction with traditional enterprise systems and other applications. It acts as the glue for integrating the industrial devices, and their data and functionalities with enterprise repos and traditional information stores.
- **Service Management**: Functionalities offered by the devices are depicted as services here to ease the integration in traditional enterprise landscapes. Tools for their monitoring are provided.
- **Device Management**: Includes monitoring and inventory of devices, including service lifecycle management.

- **Platform Abstraction**: This layer enables the abstraction of all devices independent of whether they natively support WS or not, to be wrapped and represented as services on the higher systems. In addition to service-enabling the communication with devices, this layer also provides a unified view on remotely installing or updating the software that runs on devices.
- **Devices & Protocols**: These layers include the actual devices that connect over multiple protocols to the infrastructure. The respective plugins of course need to be in place so that they can be seamlessly integrated to SIA.

- Integration between a programmable logic controller, a robotic gripper, and SunSPOT wireless sensor nodes, while these are monitored by the SAP Manufacturing Integration and Intelligence software (SAP MII), which is also responsible for the execution of the business logic.
- Event-based interaction between Radio Frequency Identification (RFID) Reader (product ID via the RFID tag), robotic arm (used to demo transportation), a wireless sensor which monitored the usage of an emergency button, an IP-plugged emergency lamp, and a web application monitoring the actual production status and producing analytics.
- Production planning, execution, and monitoring via SAP MII of a test-rig controlled by Siemens Power Line Communication (PLC) and communicating over OPC.
- Passive energy monitoring via the usage of sensors (Ploggs) and gateways.

- Integration between a programmable logic controller, a robotic gripper, and SunSPOT wireless sensor nodes, while these are monitored by the SAPManufacturing Integration and Intelligence software (SAP MII), which is also responsible for the execution of the business logic.

- Event-based interaction between Radio Frequency Identification (RFID) Reader (product ID via the RFID tag), robotic arm (used to demo transportation), a wireless sensor which monitored the usage of an emergency button, an IP-plugged emergency lamp, and a web application monitoring the actual production status and producing analytics.
- Production planning, execution, and monitoring via SAP MII of a test-rig controlled by Siemens Power Line Communication (PLC) and communicating over OPC.
- Passive energy monitoring via the usage of sensors (Ploggs) and gateways.