# IoT Architecture Reference Model (ARM)

**Module 2**

**Of**

**IoT fundamentals**

*https://www.iot-a.eu/public/public-documents/*

*https://cocoa.ethz.ch/downloads/2014/01/1360_D1%202_Initial_architectural_reference_model_for_IoT.pdf*

## Module 2 - Topics

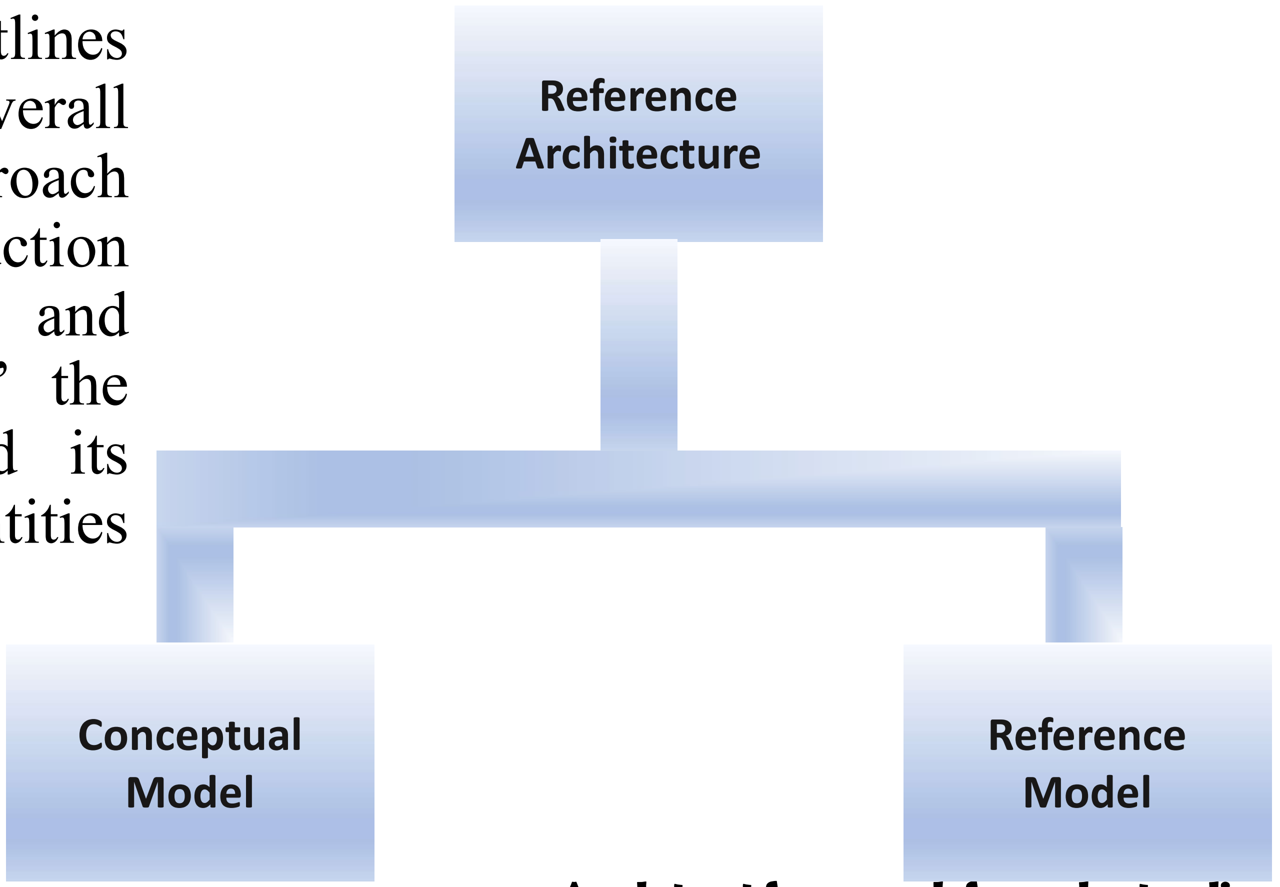
- Domain Model,
- Information Model,
- Functional Model,
- Communication and security model,
- SOA based architecture.

## Reference Architecture For IoT?

- IoT devices are inherently connected – ***A model is needed to specify interactions with the devices.***
- **An architecture** is needed to “***tame***” ***complexity and*** “***achieve***” ***scalability***
- Devices are expect to interact with themselves and the environment, continually – An architecture is need to ***achieve high-availability and support deployment across highly-heterogeneous computational platforms***
- Devices may not be designed for continuous “everyday” usage – An architecture is needed **to support remote, automatic and managed updates of the IoT devices**.
- **IoT devices** are likely **to be used for collecting and analyzing data** – An architecture is need ***for managing the identity and access control for IoT devices to ensure privacy***

## IoT Reference Architecture – Goals and Objectives

- IoT RA outlines "what" the overall structure approach for the construction of IoT systems and indicates "how" the architecture and its domains or entities will operate



It defines **a common structure and definitions describing the concepts and relationships with the IoT** systems

An **abstract framework for understanding relationships among entities of an environment and for developing consistent specifications supporting that environment**

## Vision of Architectural Reference Model (ARM)

- IoT Reference Model to promote **common understanding**
  - **High abstraction level**
  - **Describes the aspects of the IoT domain that do not change**
  - **Enables a general discourse on the IoT domain**

  The description of the reference model includes:

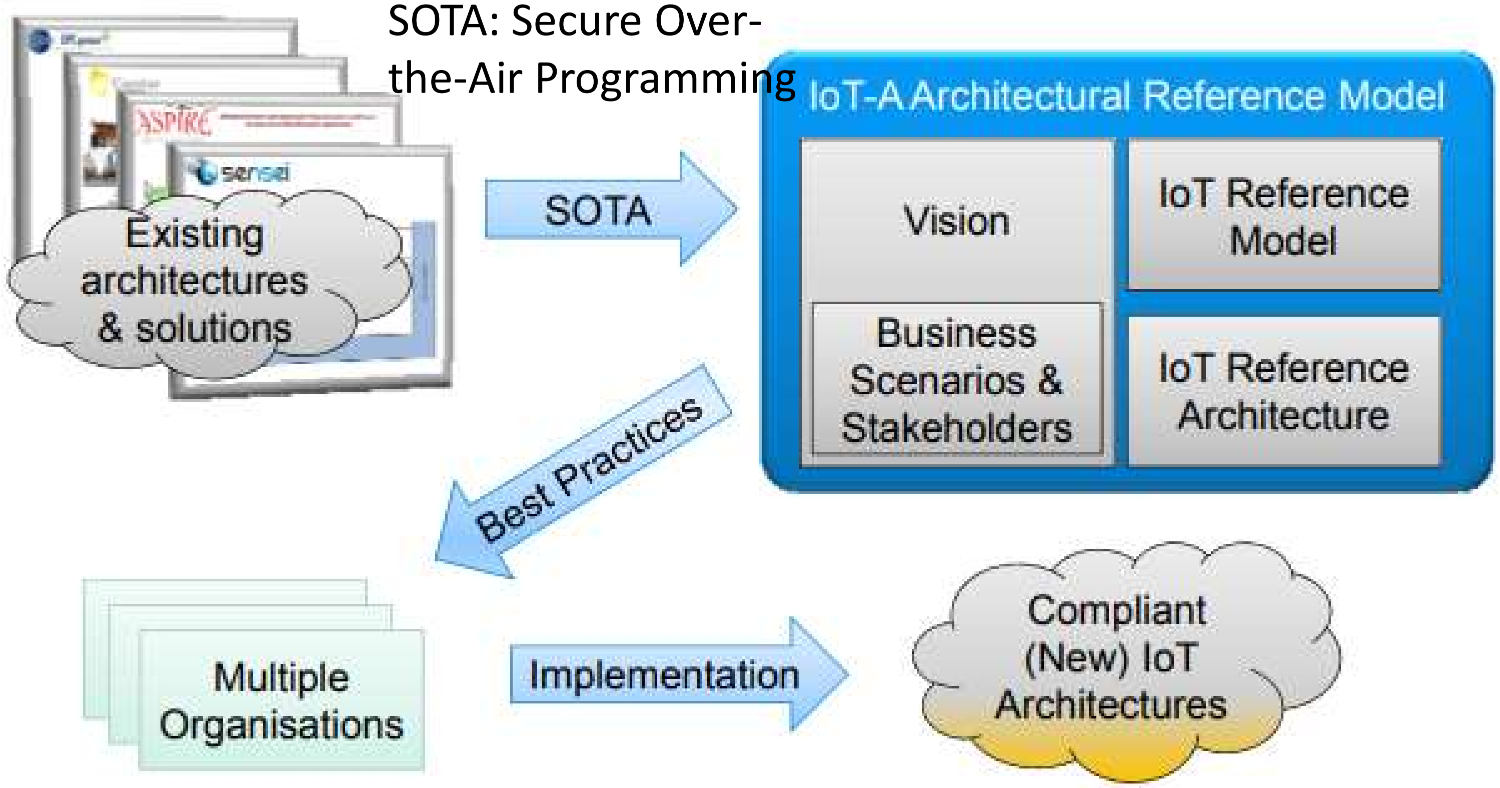
  - **a domain model** as a top-level description,
  - **an information model** explaining how IoT knowledge is going to be modelled, and
  - **a communication model** in order to understand interaction schemes for smart objects

## Vision of Architectural Reference Model (ARM)

- IoT Reference Architecture to describe essential building blocks and identify design choices to deal with conflicting requirements
  - Reference for building compliant IoT architectures
  - Provides views and perspectives on different architectural aspects that are of concern to stakeholders of the IoT
  - Best *Practice / Guidelines* to help in *developing an architecture for a specific system* based on the IoT Reference Architecture
  - Provide guidance for system architects

# IoT-A architectural reference model building blocks

SOTA: Secure Over-the-Air Programming


Existing architectures & solutions
SOTA
IoT-A Architectural Reference Model
Vision
IoT Reference Model
Business Scenarios & Stakeholders
IoT Reference Architecture
Best Practices
Multiple Organisations
Implementation
Compliant (New) IoT Architectures

The **IoT-A architectural reference model** provides **best practices to the organizations so that they can create compliant IoT architectures in different application domains**. Where application **domains are overlapping, the compliance** to the reference architecture **ensures the interoperability of solutions** and allows **the formation of new synergies across those domains**.

## Uses of an architectural reference model

**Generation of architectures**

-use of the reference architecture (together with best practices that are use case specific) for the generations of compliant architectures for specific systems.
-achieved by enabling tool support
-intrinsically provide interoperability of IoT systems

**Identifying differences**

-while using aforementioned system-generation tools based on the architectural reference model, any differences in the derived architectures can be attributed to the particularities of the pertinent use case.
-applying the architectural reference model predictions of system complexity, system cost, etc. are available for the general system parts to be implemented.

## Uses of an architectural reference model

**Benchmarking**

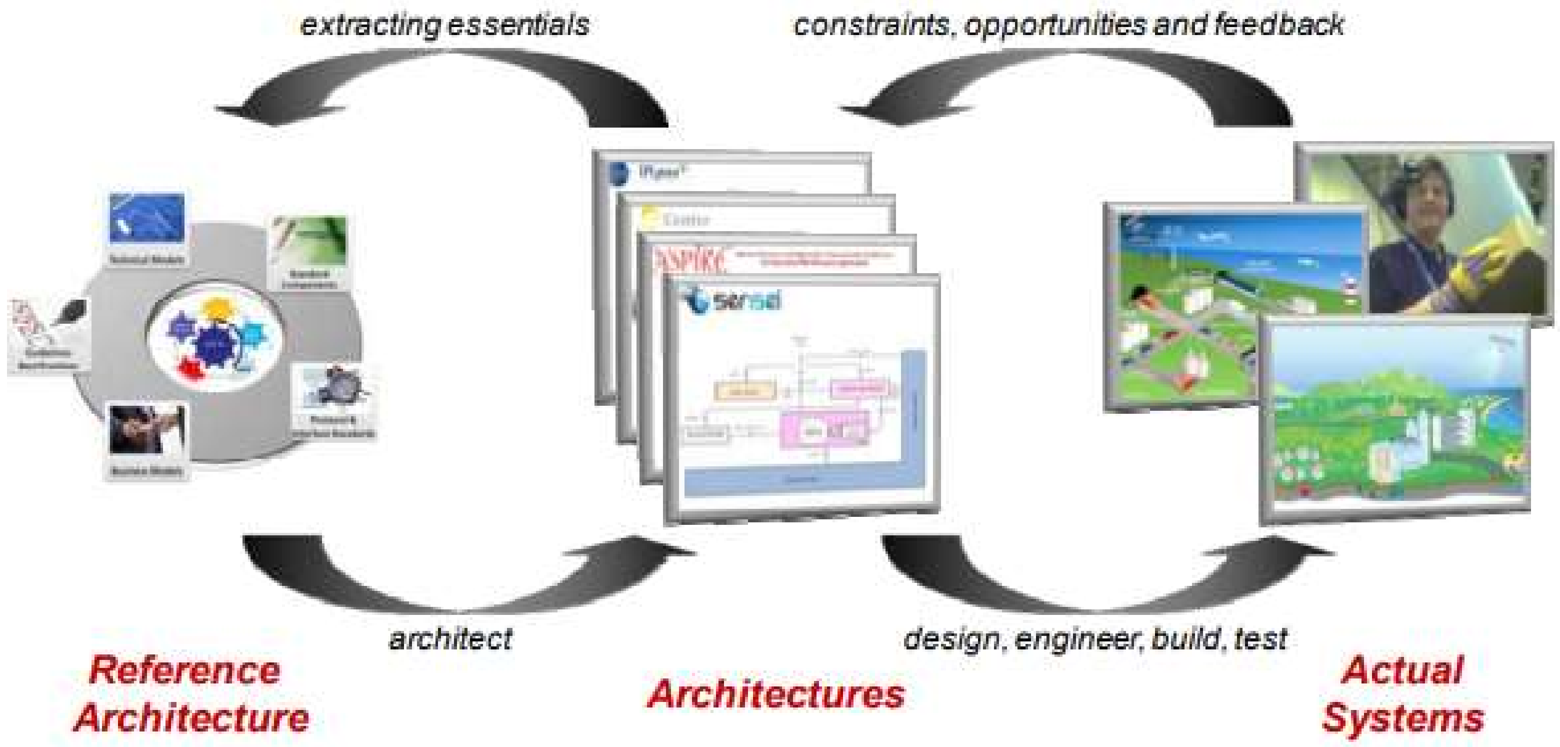
- **For example, NASA used a reference architecture of its new exploration vehicle to better benchmark tenders it was going to receive during a public bidding process.**
- **In other words, while the reference model prescribes the language to be used in the systems/architectures to be assessed, the reference architecture states the minimum (functional) requirement on the systems/architectures**
- **provides a high level of transparency to the benchmarking process.**

## Uses of an architectural reference model

### Cognitive aid

- First, it *aids in guiding discussions*, since it provides a language everyone involved can use, and which is intimately linked to the architecture, the system, the usage domain, etc.
- Second, the high-level view provided in such a model is of *high educational value*, since it provides an abstract but also rich view of the domain.
- Third, the architectural reference model *can assist project leaders* when planning the work ahead and the teams needed. For instance, the *functionality groups identified in the functional view* of the IoT system teams working on an IoT system implementation.
- Fourth, the architectural reference model aids in identifying independent building blocks for IoT systems. This *constitutes very valuable information when dealing with questions like system modularity, third-vendor options, re-use of already developed components*, etc.

## Relationship between a reference architecture, architectures, and actual systems



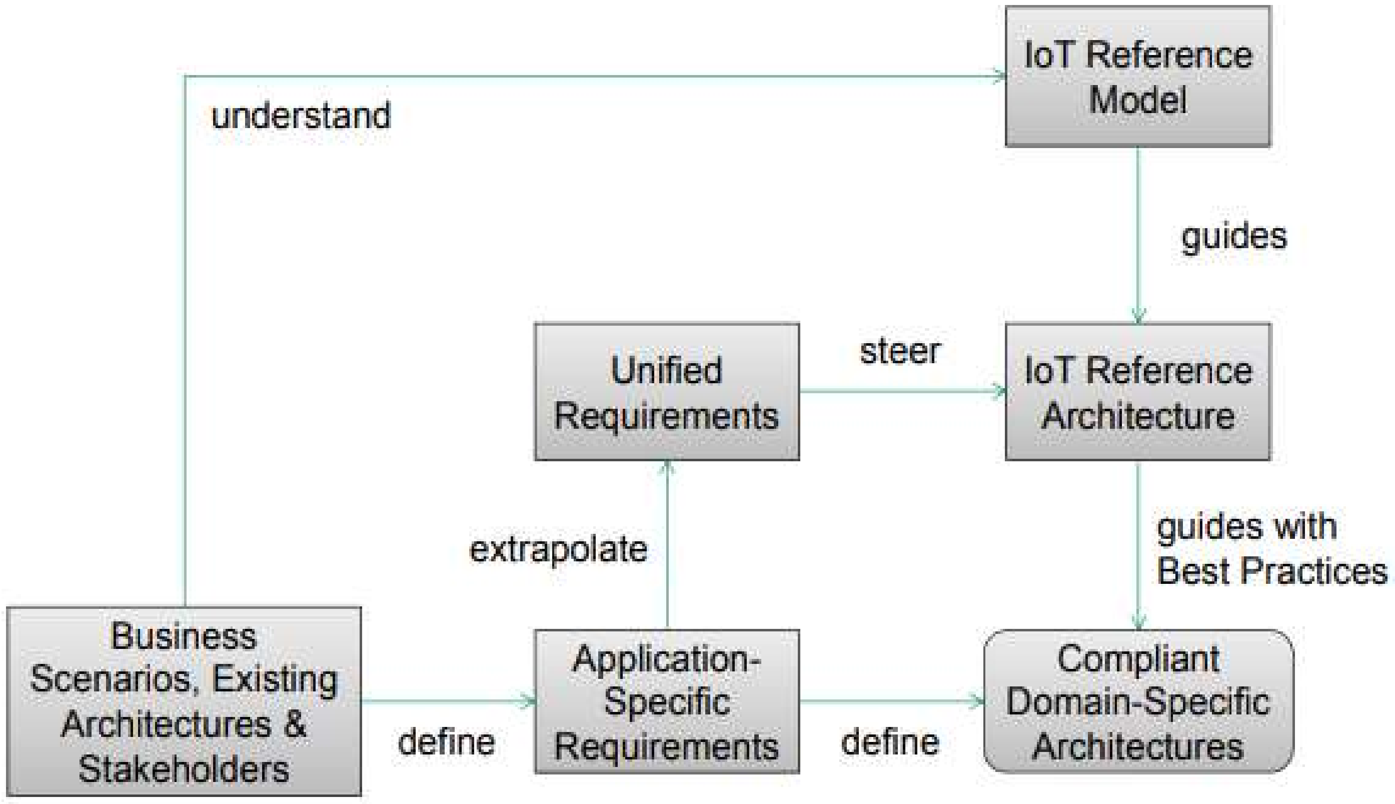
## Relationship between a reference architecture, architectures, and actual systems

**Reference models and reference architectures** provide **a description of greater abstraction than what is inherent to actual systems and applications**.

- both **more abstract than system architectures that have been designed for a particular application**.

- **Architectures** do help in **designing, engineering, building, and testing actual systems**. At the same time, ***understanding systems constraints*** *better* can provide input to the architecture design, and in turn this ***allows identifying future opportunities***.
- **By extracting essentials** of **existing architectures**, like **mechanisms or usage of standards**, a **reference architecture can be defined**.
- **A reference architecture** can **provide guidance in form of best practices**. Such guidance can, for instance, **make new architectures and systems compliant to each other**.

# High-level taxonomy of the IoT reference model and IoT reference architecture


IoT Reference Model
understand
guides
Unified Requirements
steer
IoT Reference Architecture
extrapolate
guides with Best Practices
Business Scenarios, Existing Architectures & Stakeholders
define
Application-Specific Requirements
define
Compliant Domain-Specific Architectures

## High-level taxonomy of the IoT reference model and IoT reference architecture

- **IoT Reference Model** provides guidance for the description of the IoT reference architecture.
- **IoT reference architecture** in turn guides the definition of compliant domain specific architectures.
- **Essential inputs** for the definition of the IoT reference model are stakeholder concerns, business scenarios, and existing architectures.
- Important here is to create a common understanding of **the IoT domain** from the different inputs. This is mainly ***a modelling exercise, during which experts have to work together and extract the main concepts and their relations of the IoT domain from available knowledge***.
- **Unified requirements**
    - Furthermore, **business scenarios, existing architectures, and stakeholder concerns can be transformed into application specific requirements**. When extrapolated, these requirements lead to a set of unified requirements.
    - Unified requirements ***in turn steer the definition of the IoT Reference Architecture***.

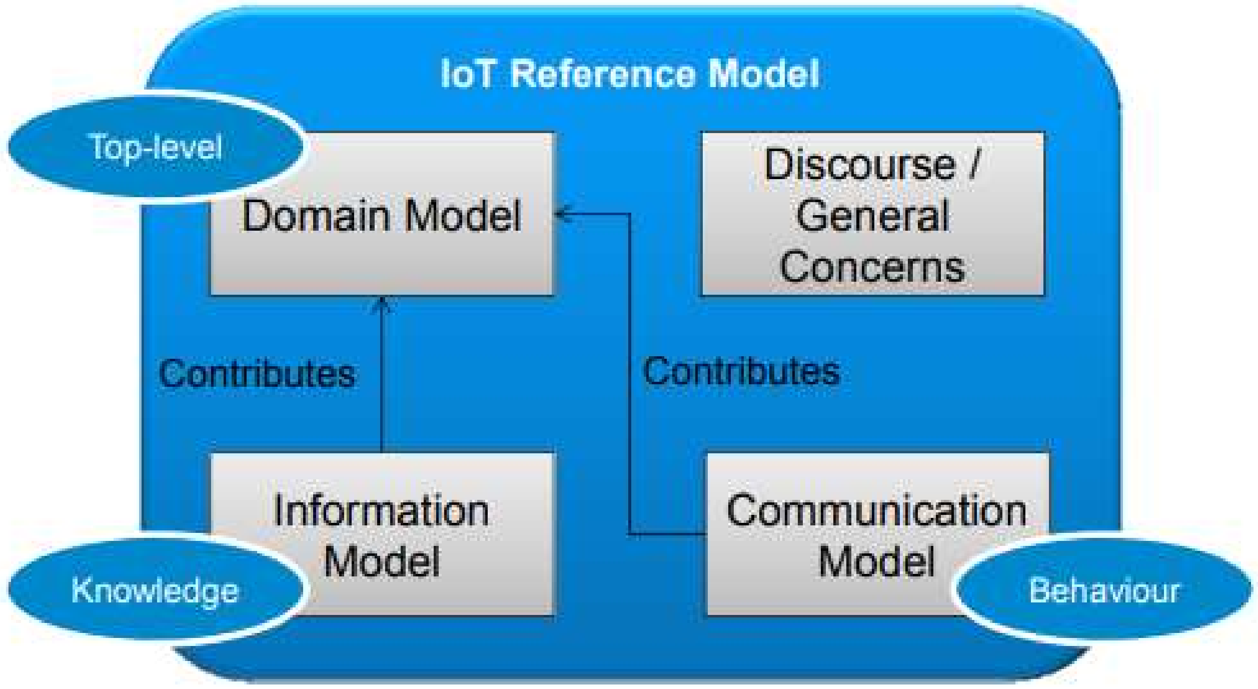
## IoT Reference Architecture structuring approach

- Requirements analysis ***identified several views and perspectives that serve as the basis for the IoT Reference Architecture structuring approach***. The views and their definition are provided in the following list:
  - **Functional** - Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
  - **Information** - Describes the way that the architecture stores, manipulates, manages, and distributes data and information.
  - **Deployment** - Describes the environment into which the system will be deployed, including the dependencies the system has on its runtime environment
  - **Operational** - Describes how the system will be operated, administered, and supported when it is running in its production environment

## Quality aspects of the IoT reference architecture

- **Security and privacy** – Provides and analysis of the *security threads in the functionality groups of the architecture and gives an explanation how security and privacy concerns should be addressed*.
- **Performance and scalability** – Provides the *ability of the system to handle a large number of devices, services and processes in an efficient way*. Further more, the fluctuation of requests towards those has to be handled in a scalable way.
- **Availability and resilience** – The *ability of the system to be fully or partly operational* as and when required, and *to effectively handle failures that could affect system availability*
- **Evolution and interoperability** – The *ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment*; the ability of two or more systems or components to *exchange and use information*.

## Component of IoT ARM



## Components of IoT ARM

**Discourse:**

- ***starts*** with a discourse about IoT.
- ***identify abstract quality concepts*** that have to be taken into account for the realization of IoT systems (**for instance, heterogeneity and interoperability**).
- discourse is ***followed by the definition of the domain model***.

**Domain Model:**

- ***concepts and entities that represent particular aspects*** of the IoT domain are summarised in **a top-level domain description and their relations** are defined.
- **serves as a common lexicon for and taxonomy of the IoT**.
- **entities in a domain model are either responsible for keeping track of certain information and for doing certain things**. This refers to **knowledge and behaviour**, respectively.

**Information Model:**

- ***knowledge is represented*** through an information model
- Its purpose is ***to specify the data semantics of the domain model***.

**Communication Model:**

- ***communication model addresses high-level communication paradigms*** pertinent to the IoT domain.
- ***describes how communication has to be managed in order to achieve the features required in the IoT***

## Components of IoT ARM - Discourse:

**Discourse:**

- ***starts*** with a discourse about IoT.
- ***identify abstract quality concepts*** that have to be taken into account for the realization of IoT systems (**for instance, heterogeneity and interoperability**).
  - **Heterogeneity** - of **technologies in the IoT** is significantly higher than in traditional computing systems.
    - They ***include RFID, sensor networks, embedded systems and mobile technologies***, and also a large variety of existing as well as ***emerging communication technologies***.
  - **Interoperability** - has to be **supported in all functionality groups** (and communication layers).
    - For **communication, the co-existence of technologies, as well as bridges between different technologies**, needs to be supported.
    - At **a minimum, a common way of describing the services and their interfaces** has to be achieved.
    - Interoperability not only has to be **achieved across a single domain, but across different administrative and application domains**.

## Components of IoT ARM - Discourse:

- Scalability - is an important challenge when the *IoT leaves the confinement of small, isolated vertical islands and becomes part of everyday life*.
  - The number of devices that *need to be managed and that communicate with each other*.
  - Even *more critical is the management of the data generated* and their interpretation for application purposes. This *relates to semantics of data, as well as the efficient handling of the resulting data stream.*

- Manageability - of such *large numbers of devices*, especially in environments that cannot be centrally controlled, can *only be addressed through autonomous behaviour including*

  *- self management,*
  *- self-configuration,*
  *-self-healing,*
  *-self-optimisation and*
  *-self-protection*.

## Components of IoT ARM - Discourse:

- Reliability - is a major factor for the acceptance of any system.
  - For example, when looking at the *nature of wireless sensor netw*orks, it becomes *apparent that the availability of information might* vary over time, yet *an end-user service that depends on this information still needs to respond in an appropriate way according to its initial purpose*.

## Components of IoT ARM - Discourse:

- Reliability - is a major factor for the acceptance of any system.
    - For example, when looking at the *nature of wireless sensor netw*orks, it becomes *apparent that the availability of information might* vary over time, yet *an end-user service that depends on this information still needs to respond in an appropriate way according to its initial purpose*.

## Components of IoT ARM – Domain Model:

- The ***IoT-A project defines a domain model as a description of objects belonging to a particular area of interest***.
- The domain model also ***defines attributes of these objects, such as name and identifier.***
- The domain model ***defines relationships between objects, for instance "instruments produce data sets."***
- Domain models also help to ***facilitate correlative use and exchange of data between domains.***
- Main purpose of a ***domain model is to generate a common understanding of the problem domain in question.***
- *Types Entity*
    - *Physical Entity*
    - *Virtual Entity*
    - *Augmented Entity*

## Components of IoT ARM – Domain Model:

- *Physical Entity*
  - **Physical entities** can be almost **any object or environment; from humans or animals to cars; from store or logistic chain items to computers; from electronic appliances to closed or open environments**.
  - The **user is a human person or some kind of active digital entity** (e.g., a Service, an application, or a software agent) that has a goal.
  - The **physical entity is a discrete, identifiable part of the physical environment which is of interest to the user for the completion of his goal**.
- *Virtual Entity*
  - **Digital entities** that are associated to **a single physical entity that they represent. While ideally there is** *only one physical entity for each virtual entity***.**
  - **Each *virtual entity must have one and only one ID* that identifies the represented object.**
  - **Digital entities *can be either active elements (e.g., software code) or passive elements (e.g., a data-base entry)***
- *Augmented entity*
  - as the ***composition of a physical entity and its associated virtual entity***. Any **changes in the properties** of an augmented entity have **to be represented in both the physical and digital world.**

## Components of IoT ARM – Domain Model:

Three types of device types for the IOT Domain Model:

***Sensors***

sensors are devices that detect external information replacing it with a signals that human and machines can distinguish.

IOT ***Sensors used to detect and measure various physical phenomena such as heat and pressure as well as 5 human senses sight, hearing, touch, taste and smell***.

***Actuators***

An actuator is ***a machine component or system that moves or controls the mechanism or the system***. Sensors in the device ***sense the environment, then control signals are generated for the actuators according to the actions needed to perform.***

***Tags***

***-identify the physical Entity that they are attached to.***

***- devices or physical entities but no both as the domain model*** shows. Example: **tag as a device Radio Frequency ID. Tag as a physical entity – paper printed immutable barcode or Quick Response (QR) code.**

## Components of IoT ARM – Domain Model:

**Definition of Resources in Domain Model**

- **Resources** are software components that **provide information about physical entities or enable the controlling of devices**.
    - a distinction between ***on-device resources and network resources***.

- **On-device resources** are ***hosted on devices, viz. software that is deployed locally on a device***. They include executable code for accessing, processing, and storing sensor information, as well as code for controlling actuators.

- **Network resources** are ***resources available somewhere in the network***, e.g., back-end or cloud-based data bases.

- A **virtual entity** can also be associated with ***one or more resources that enable interaction with the physical entity*** that the virtual entity represents. This association is important in look-up and discovery processes.

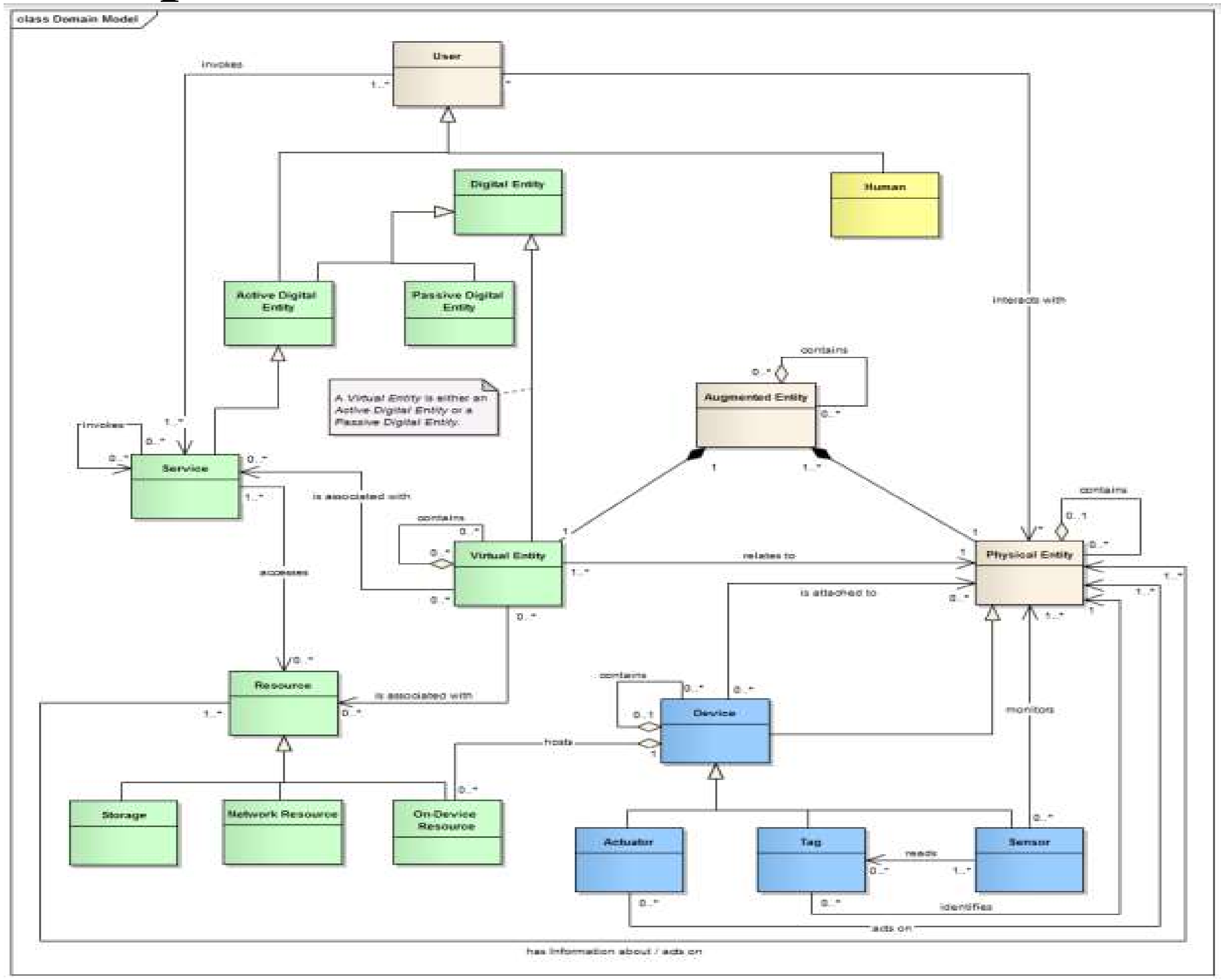
## Components of IoT ARM – Domain Model:

**Definition of Services in Domain Model**

- a ***service provides a well-defined and standardized interface***, offering all ***necessary functionalities for interacting with physical entities and related processes***. All this is done ***via the network***.

- On the lowest level –**closest to the actual device hardware**– , ***services expose the functionality of a device by accessing its hosted resources (access)***.

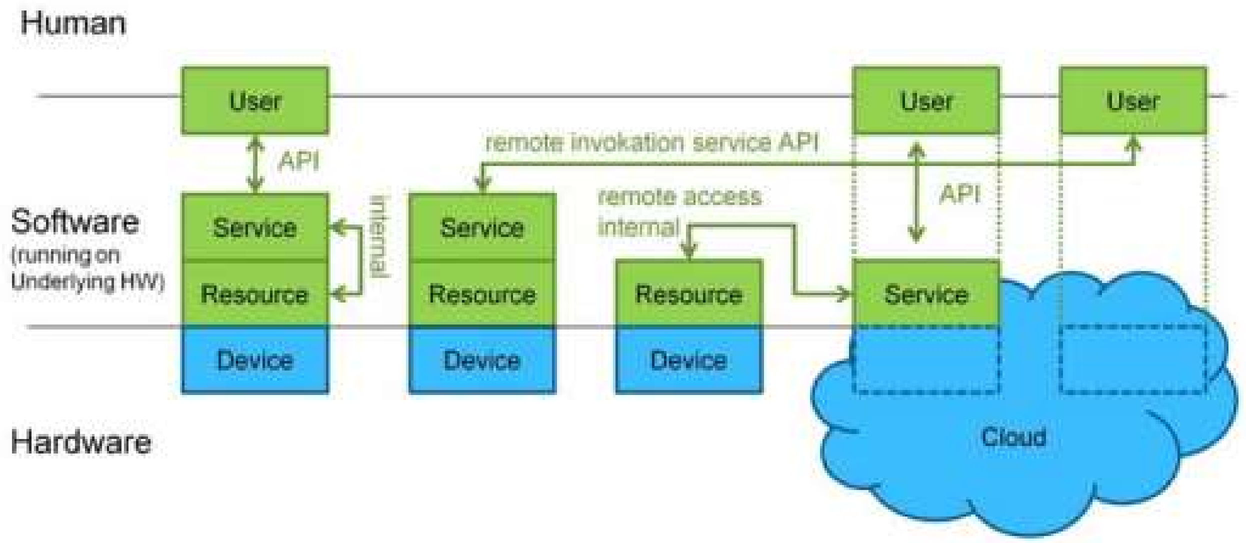
# Components of IoT ARM – Domain Model:



IoT Domain Model. All object names are explained in the main text.

- Hardware concepts are shown in **blue**,
- Software in green, animated objects in yellow, and
- Concepts that fit into either multiple or neither categories in brown.

# Components of IoT ARM – Domain Model:

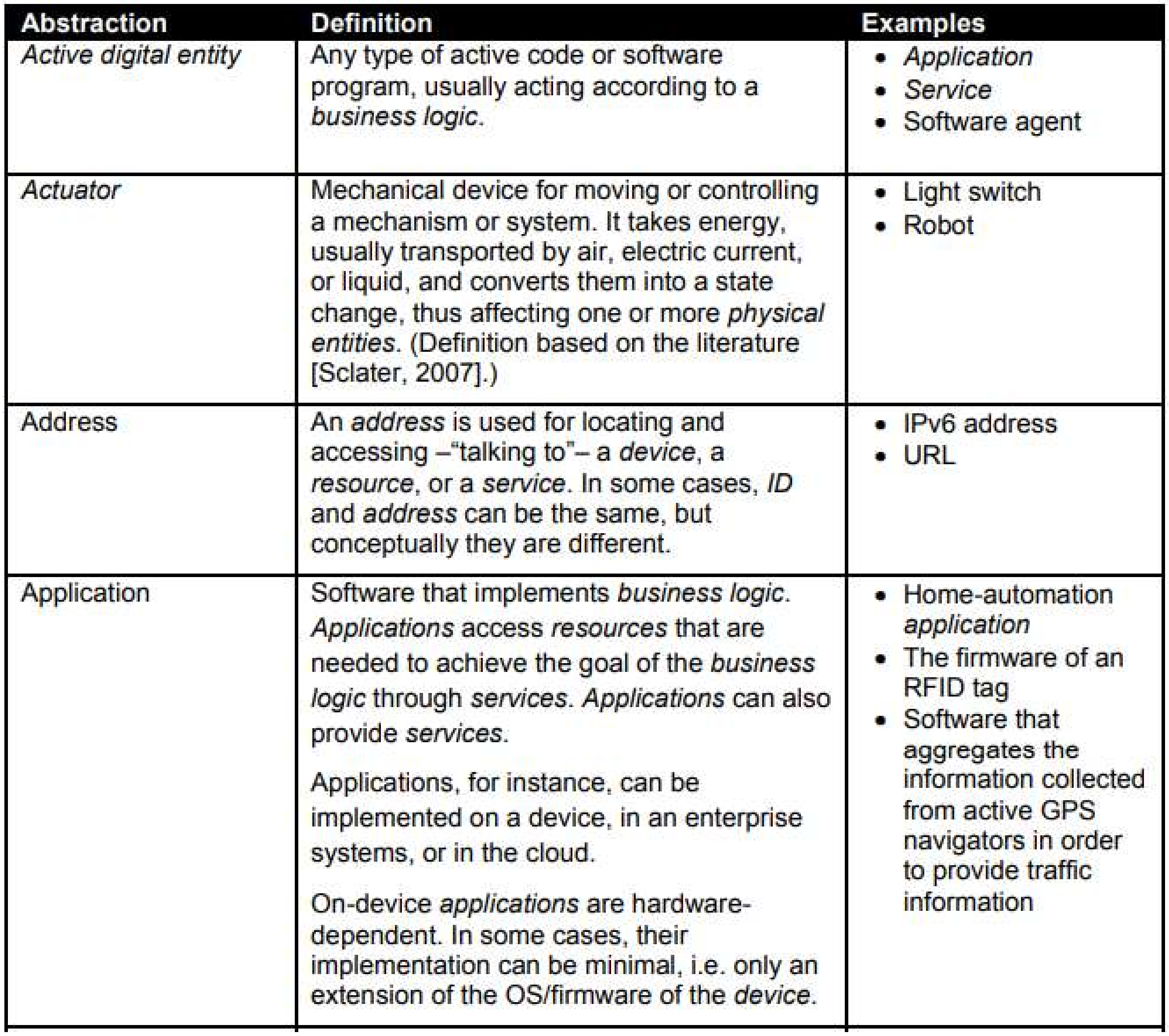


**Devices, resources and services**

IoT Domain Model. All object names are explained in the main text.

- Hardware concepts are shown in blue,
- Software in green, animated objects in yellow,

## Domain Model: Terms and Definition



| Abstraction | Definition | Examples |
|---|---|---|
| *Active digital entity* | Any type of active code or software program, usually acting according to a *business logic*. | • *Application*<br>• *Service*<br>• Software agent |
| *Actuator* | Mechanical device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid, and converts them into a state change, thus affecting one or more *physical entities*. (Definition based on the literature [Sclater, 2007].) | • Light switch<br>• Robot |
| Address | An *address* is used for locating and accessing –"talking to"– a *device*, a *resource*, or a *service*. In some cases, *ID* and *address* can be the same, but conceptually they are different. | • IPv6 address<br>• URL |
| Application | Software that implements *business logic*. *Applications* access *resources* that are needed to achieve the goal of the *business logic* through *services*. *Applications* can also provide *services*.<br><br>Applications, for instance, can be implemented on a device, in an enterprise systems, or in the cloud.<br><br>On-device *applications* are hardware-dependent. In some cases, their implementation can be minimal, i.e. only an extension of the OS/firmware of the *device*. | • Home-automation *application*<br>• The firmware of an RFID tag<br>• Software that aggregates the information collected from active GPS navigators in order to provide traffic information |

## Domain Model: Terms and Definition

| Abstraction | Definition | Examples |
|---|---|---|
| *Augmented entity* | The composition of a *physical entity* and its associated *virtual entity*. | See<br>• *Physical entity*<br>• *Virtual entity* |
| *Business logic* | Goal or behaviour of a system involving *things*. *Business logic* serves a particular business purpose. *Business logic* can also define the behaviour of a single or multiple physical entities, or a complete business process. | • Regulate the temperature of an environment<br>• Check that only authorised personnel can access a building |
| *Device* | Technical physical component (hardware) with communication capabilities to other IT systems. A device can be either attached to, or embedded inside a *physical entity*, or monitor a *physical entity* in its vicinity. | • Mobile Phone<br>• Embedded system or sensor node with multiple *sensors* and/or *actuators*<br>• Any *sensor*, *actuator*, or *gateway* |

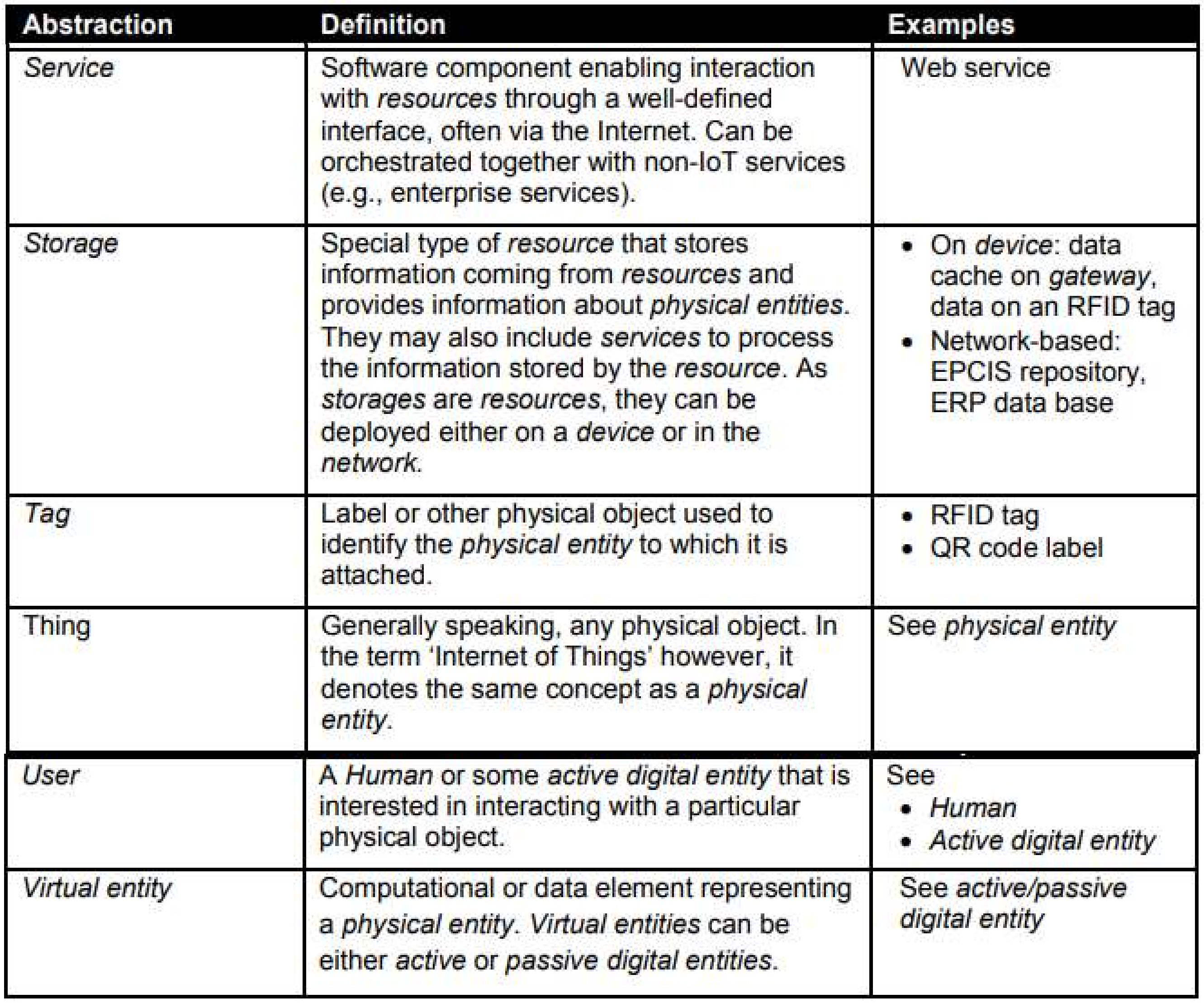
## Domain Model: Terms and Definition

| Abstraction | Definition | Examples |
|---|---|---|
| *Digital entity* | Any computational or data element of an IT-based system. | See • *Active/passive digital entity* • *Virtual entity* |
| Discovery | Discovery is a service for finding unknown resources/services, based on a (rough) specification of the desired result. It may be utilised by a human or another service. Credentials for authorisation are considered when executing the discovery. | • Bluetooth, *device* and *service* discovery • UDDI • Google |
| Gateway | *Device* that provides protocol translation between peripheral trunks of the IoT that are provided with lower parts of the communication stacks (see Section 2.4). For efficiency purposes, *gateways* can act at different layers, depending on which is the lowest layer in a common protocol implementation. *Gateways* can also provide support for security, scalability, service discovery, geo-localisation, billing, etc. | WSN *gateway*, connecting local sensor-node *devices* to the Internet |
| *Human* | A *human* that either physically interacts with *physical entities* or records information about them, or both. | Any person |
| Identity | Properties of an entity that makes it definable and recognizable. | Who am I? |

## Domain Model: Terms and Definition

| Abstraction | Definition | Examples |
|---|---|---|
| *Passive digital entity* | A digital representation of something stored in an IT-based system. | • Data-base entry<br>• File |
| *Physical entity* | Any physical object that is relevant from a user or application perspective. | Pallets, boxes containing consumer goods, cars, machines, fridges etc., as well as animate objects like animals and humans. |
| Resolution | Query response process by which a given ID is associated with a set of *addresses* of information and interaction *services*. Information *services* allow querying, changing and adding information about the thing in question, while interaction services enable direct interaction with the thing by accessing the *resources* of the associated *devices*. Resolution is based on a-priori knowledge. | • EPC ONS<br>• DNS |
| *Resource* | Computational element that gives access to information about, or actuation capabilities on a *physical entity*. | See<br>• *On-device resource*<br>• *Network resource* |
| *Sensor* | A *device* identifying or recording features of a given *physical entity*. | • Temperature sensor<br>• RFID reader<br>• Camera |

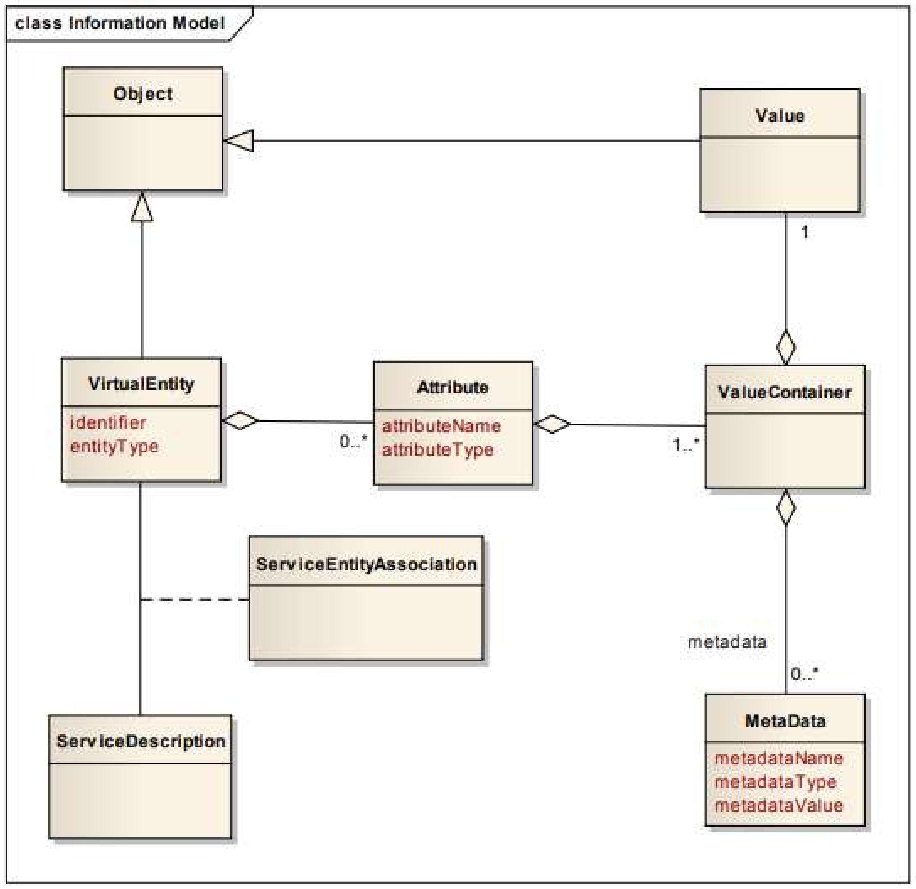
## Domain Model: Terms and Definition



| Abstraction | Definition | Examples |
|---|---|---|
| *Service* | Software component enabling interaction with *resources* through a well-defined interface, often via the Internet. Can be orchestrated together with non-IoT services (e.g., enterprise services). | Web service |
| *Storage* | Special type of *resource* that stores information coming from *resources* and provides information about *physical entities*. They may also include *services* to process the information stored by the *resource*. As *storages* are *resources*, they can be deployed either on a *device* or in the *network*. | • On *device*: data cache on *gateway*, data on an RFID tag<br>• Network-based: EPCIS repository, ERP data base |
| *Tag* | Label or other physical object used to identify the *physical entity* to which it is attached. | • RFID tag<br>• QR code label |
| Thing | Generally speaking, any physical object. In the term 'Internet of Things' however, it denotes the same concept as a *physical entity*. | See *physical entity* |
| *User* | A *Human* or some *active digital entity* that is interested in interacting with a particular physical object. | See<br>• *Human*<br>• *Active digital entity* |
| *Virtual entity* | Computational or data element representing a *physical entity*. *Virtual entities* can be either *active* or *passive digital entities*. | See *active/passive digital entity* |

# Components of IoT ARM – Information model

**Virtual entity** in the IOT Domain Model is the thing in the IOT, the IOT Domain model is the thing in the IOT, the ***IOT information model captures the details of a virtual entity centric model, the IoT Information model is presented using Unified modelling language(UML) diagrams***.



1. By keeping a history of the associations, e.g. by adding metadata to the has *Temperature* values that link to the device entity (data provenance);
2. By recording the current association, e.g. by introducing an attribute *metadata object* that links the entity to the resources that are currently able to contributed to the values of this attribute.

## Components of IoT ARM – Information model

**Information model will expanded to two additional usage areas:**

- **Service description:** Services **provide access to resources and are used to access information or to control physical entities**.
  - A service description describes a service, using, for instance, a user service description language such as USDL.
  - The information a service provides is associated to a VirtualEntity.
  - The association also captures whether the service is used for accessing information or controlling information, or both.
- **Actuation Control:** The information model will be further reviewed **to capture how entities can be controlled**, for instance via actuators.

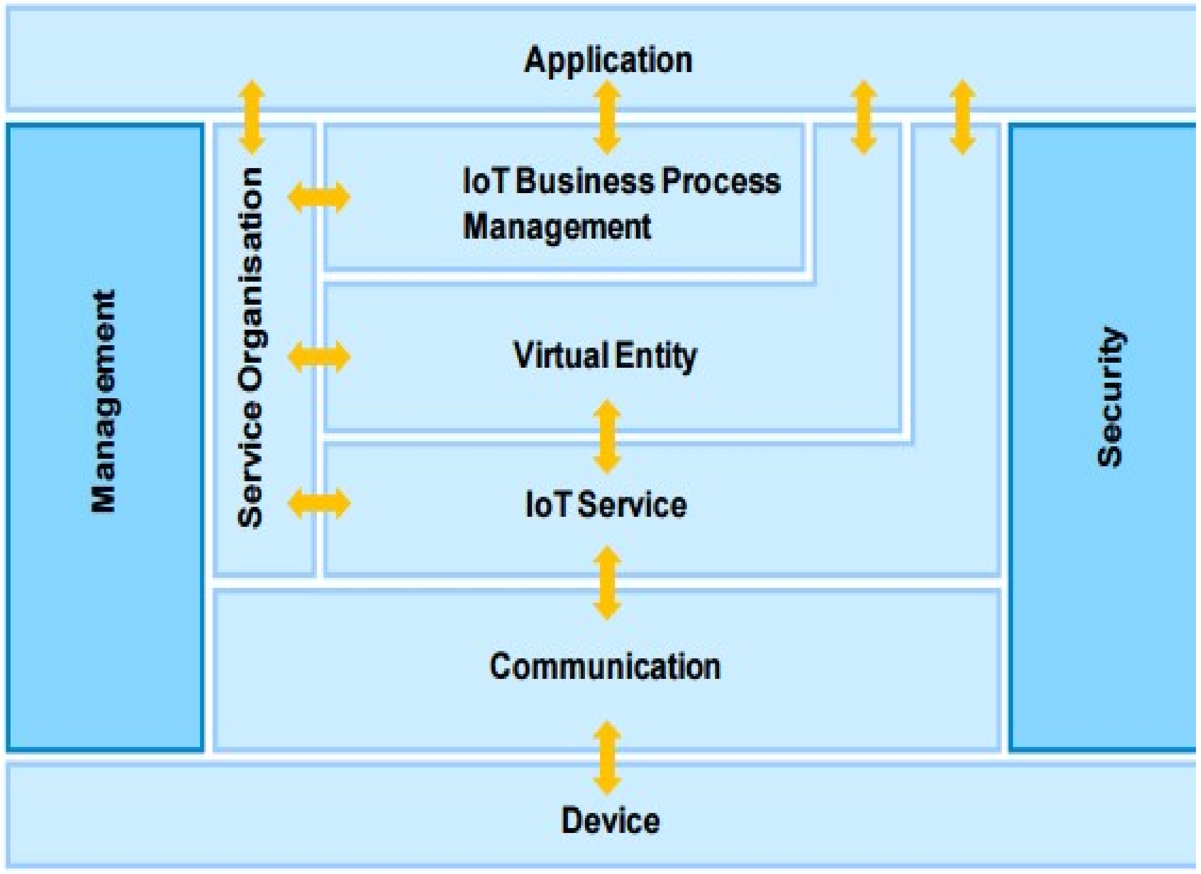
## Functional Model

- It 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 a FG interfaces and interactions between the components**.

- The functional view is typically derived from the functional model in conjunction with **high-level requirement**

## Functional Model – Major Components

The FM is derived from internal and external requirements
- In closely related to the Reference Architecture (see Functional Views)
- 7 Functional Groups strongly related to:
- Domain model
- Communication Model
- Security Model



# Functional Model – Major Components

- **Applications:** This group describes the **functionalities provided by applications** that are built on top of an implementation of the IoT-A architecture.
- 
- **Process execution and service orchestration:** This **functionality group organises and exposes IoT resources** so that they become **available to external entities and services**. Through this set of functionalities, and the APIs that expose them, IoT Services become available to external entities and can be composed by them.

- **Virtual entity (VE) and information:** This **group maintains and organizes information related to physical entities**, **enabling search for services exposing resources associated to physical entities**. It also enables the search for services based on the physical entity they are associated to. When queried about a particular physical entity, this functionality group will return addresses of the service related to this particular physical entity.

- **IoT service & resource: When queried about a specific service, this group will return its description, providing links to the exposed resources**. This group also provides the functionalities required by services for processing information and for notifying application software and services about events related to resources and corresponding physical entities.

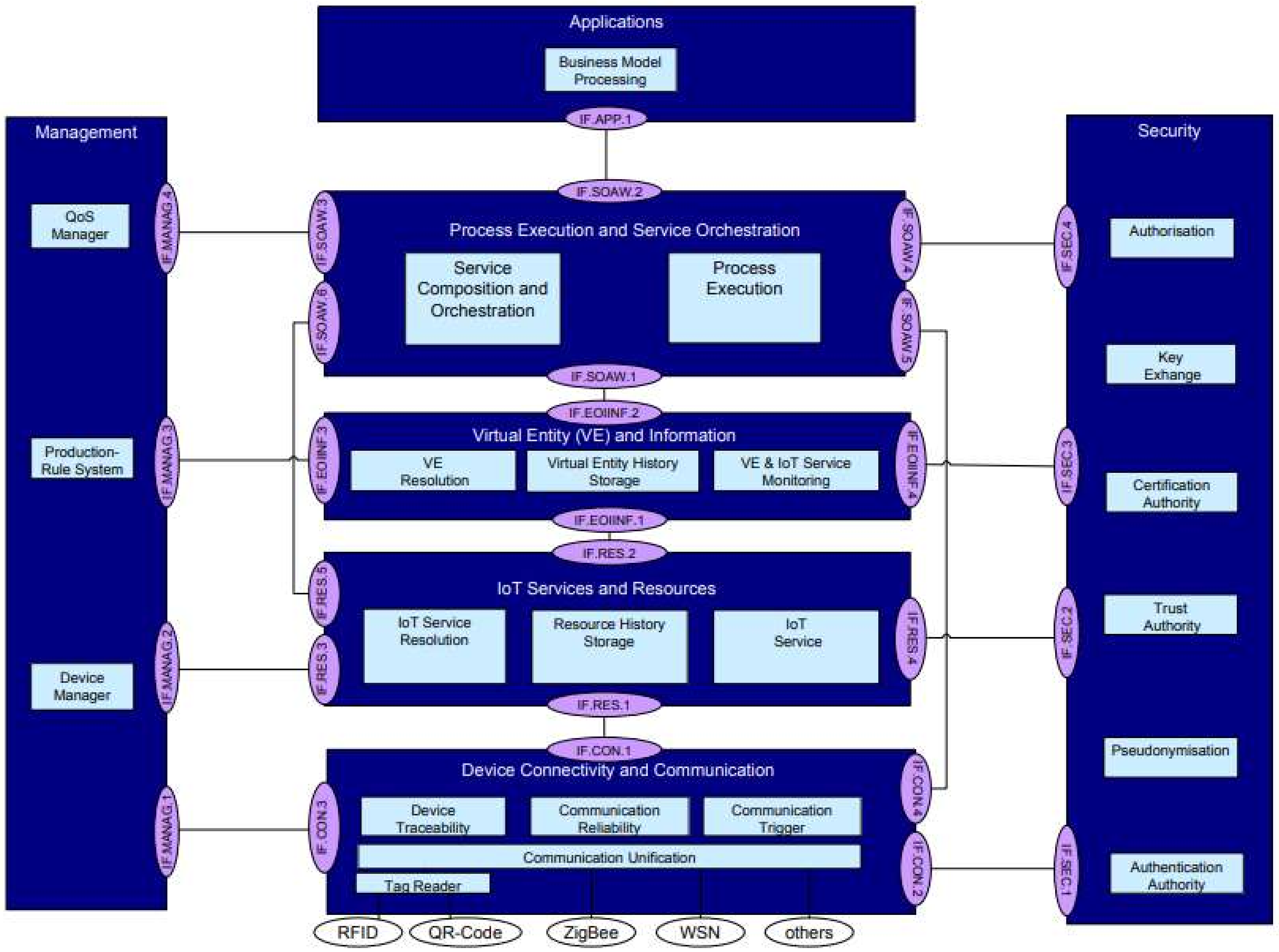
## Functional Model – Major Components

- **Device connectivity and communication:** This functional block **provides the set of methods and primitives for device connectivity and communication** (the first referring to the possibility for a device to be part of a network, the second to the possibility for this device to be source or destination of messages). Also, **this group contains methods for content-based routing**.

- **Management:** In order to **manage computational resources efficiently, management has to be handled by a single group of functionalities**.

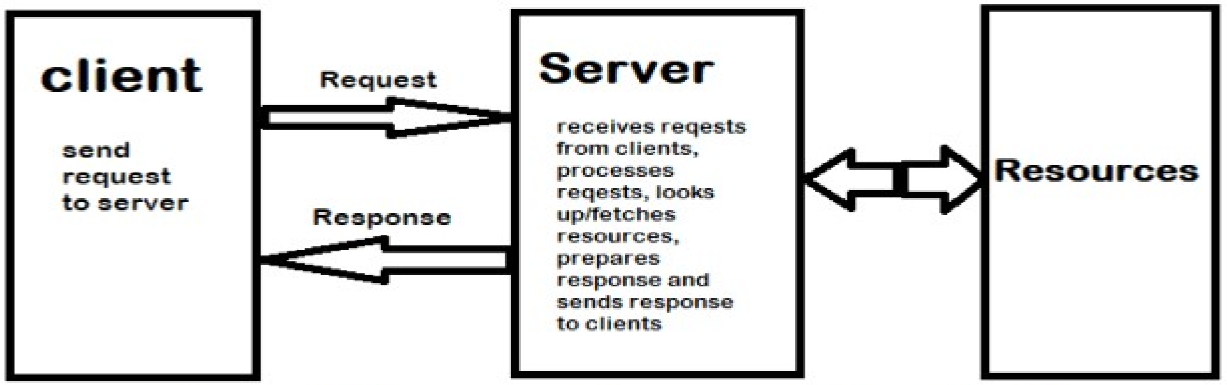
- **Security:** Security functions have **to be consistently applied by the different groups of functionalities**. Specifically, access-control policies shall consistently be applied in order to prevent unauthorised applications from obtaining access to sensitive resources. Privacy will also be enforced through pseudonymity, i.e. different (generic) identities can be used by a user when accessing IoT services.

## Functional Model – Sub Components



# Communication and security model

- **Request-Response** is a communication model in which the *client sends requests to the server and the server responds to the requests*.
- When the **server receives a request**, it decides how to *respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client*.
- **Request-Response model** is a *stateless communication model and each request-response pair is independent of others*.


client
send request to server
Request
Response
Server
receives reqests from clients, processes reqests, looks up/fetches resources, prepares response and sends response to clients
Resources

Request-Response Communication Model

## Components of IoT ARM –Communication model

The **communication model** aims at defining the main communication paradigms for connecting entities, as defined in the domain model.

- a **reference communication stack**, together with insight about the main interactions among the actors in the domain model.
- a **communication stack** similar to the ISO OSI 7-layer model for networks,
- **mapping the needed features of the domain model unto communication paradigms**

## Components of IoT ARM –Communication model

**Physical layer:**

The physical layer remains unchanged from the OSI definition.

**Link layer:**

- to address the heterogeneousness of networking technologies represented in the IoT universe, but customized communication schemes and security solutions,
- to achieve full interoperability, as well as the support of heterogeneous technologies and a comprehensive security framework, this layer must allow for diversity.

**Network layer:**

- provides the same functionalities as the correspondent OSI stack. However, in order to support global manageability, interoperability, and scalability

**ID layer:**

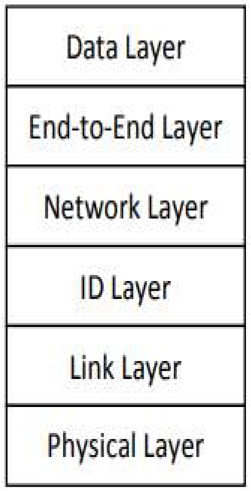
- The virtual-entity identifier (VE-ID), split from the locator, is the centre of the first convergence point in the communication stack, i.e. the ID layer. Also, security, authentication, and high-end services will exploit this layer for providing uniform addressing to the many different devices and technologies in IoT networks.

**End-to-end layer:**

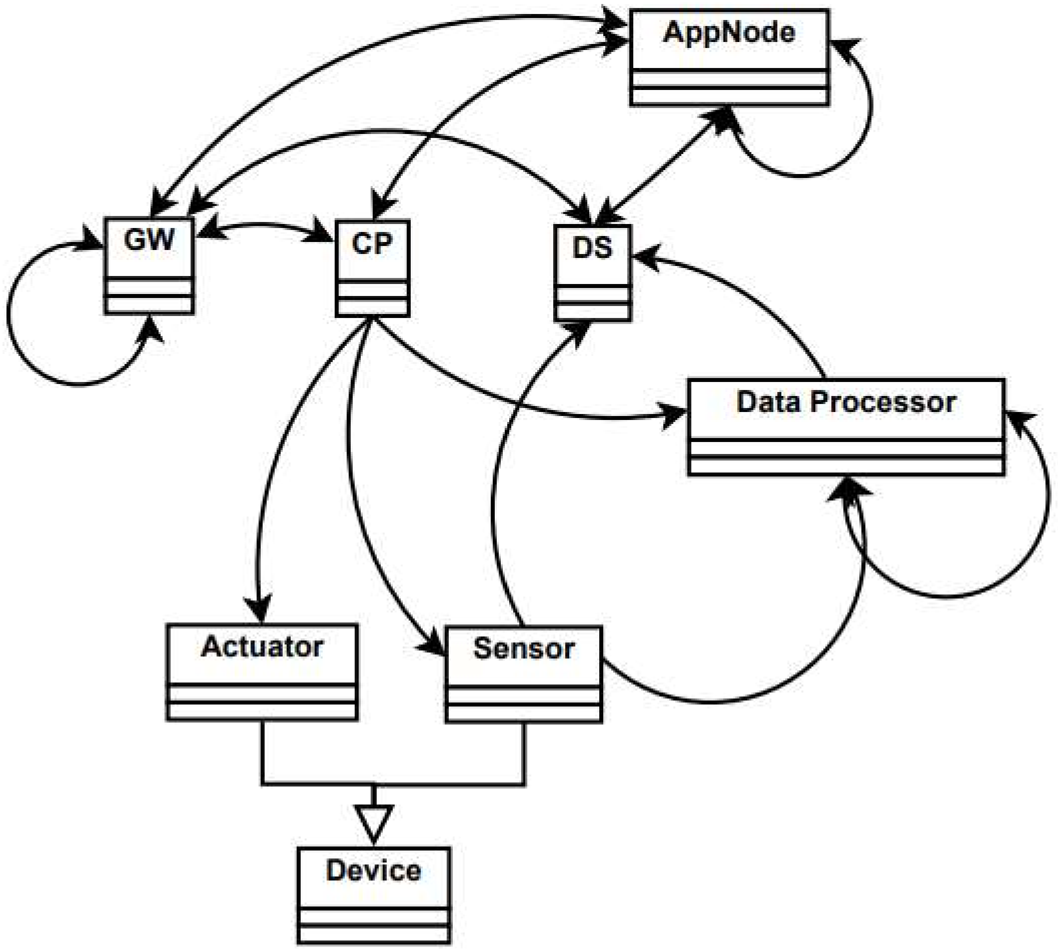
- This layer takes care of translation functionalities, proxies/gateways support and of tuning configuration parameters when the communication crosses different networking environments. By building on top of the ID and the network layers, the end-to-end layer provides the final building block for achieving a global M2M communication model.

**Data layer:**

- at the top of the communication stack, the data layer interfaces with the data layer.
- A high-level description of the data pertinent to IoT is provided by the information model



## IoT Communication model as seen from the application level



Communication layer of the IoT domain model from an application point of view. AppNode: application node; GW: gateway; CP: control point; DS: data sink.

## IoT Communication model as seen from the application level

We can imagine a digital entity to be formed by a group of sensors and actuators. a ***digital entity*** to consist of ***a group of data processors, data sinks, and control points, with at least an*** *AppNode* ***implementing the behaviour of the digital entity***.

**Application node (AppNode):**

- a software agent implementing an application or part of it.
- orchestrate different digital entities.
- The application doesn't deal directly with sensors and actuators but it requires communication with control points and data sinks.
- obviously communicate among themselves, and in this way create a distributed application.

**Control point (CP):**

*A control point is a software agent that controls actuators and sensors, and sends related messages to sensors and actuators.*

- A CP will communicate with sensors, actuators, and data processors, sending them configuration and control messages.
- A CP can handle bidirectional communication with an AppNode. The CP is usually called by AppNodes, but it is also enabled to call AppNodes after certain events, for instance an error.

## IoT Communication model as seen from the application level

We can imagine a digital entity to be formed by a group of sensors and actuators.
a ***digital entity*** to consist of ***a group of data processors, data sinks, and control points, with at least an*** *AppNode* ***implementing the behaviour of the digital entity***.

**Application node (AppNode):**

- a software agent implementing an application or part of it.
- orchestrate different digital entities.
- The application doesn't deal directly with sensors and actuators but it requires communication with control points and data sinks.
- obviously communicate among themselves, and in this way create a distributed application.

**Control point (CP):**

*A control point is a software agent that controls actuators and sensors, and sends related messages to sensors and actuators.*

- A CP will communicate with sensors, actuators, and data processors, sending them configuration and control messages.
- A CP can handle bidirectional communication with an AppNode. The CP is usually called by AppNodes, but it is also enabled to call AppNodes after certain events, for instance an error.

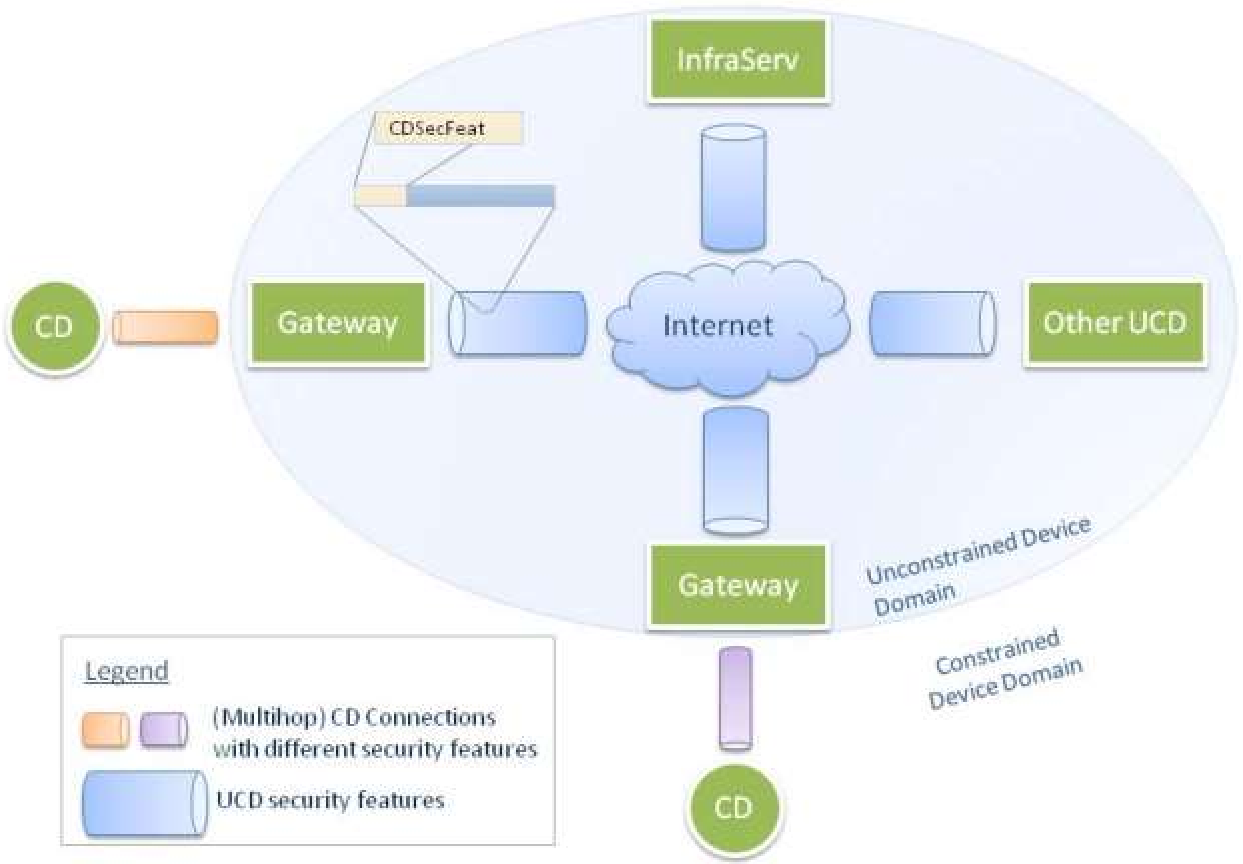
## IoT Communication model as seen from the application level

**Data sink (DS):** A data sink is *a software agent that receives data -which it will consume or store- directly from a sensor or a data processor*. This communication is *event-driven and initiated by either the sensor or the data processor*. Data sinks are controlled by AppNodes, but they also can initiate communications to the AppNodes on given events, like crossing a threshold .

**Data processor (DP):** A data processor is ***a software agent receiving data directly from sensors or from other data processors, performing operations like filtering or aggregation, before sending data to a data sink.***

**Gateway (GW):** ***A Gateway is a forwarding element, enabling various local networks to be connected.*** In this model, sensors and actuators cannot communicate directly with a gateway. Therefore, a control point, a data processor, or a data sink need to be hosted in the same network. A gateway can obviously communicate with other gateways and forward traffic from control points, data sinks, data processors, and AppNodes.

# IoT Communication model


InfraServ
CDSecFeat
CD
Gateway
Internet
Other UCD
Gateway
Unconstrained Device Domain
Constrained Device Domain
CD
Legend
(Multihop) CD Connections with different security features
UCD security features

Providing the best security features for the lower layers in each IoT domain by introducing **Gateways with active functions. CD: constrained device; UCD: unconstrained device. CDSecFeat: security feature for constrained device.**

## IoT Communication model - Gateways

Gateways have to provide the following functionalities in order to hide underlying heterogeneity:

- **Protocol adaptation** between different networks (by definition).
- **Tunnelling** between itself and other nodes of the UCD domain. (Optional; impacts on trust assessment.)
- **Management of security features** belonging to the peripheral network. (Optional)
- **Description of security options** related to outgoing traffic. (Authentication of source node, cryptographic strength, ...)
- **Filtering of incoming traffic** according to user-defined policies, which take into account security options of incoming traffic, destination-node preferences and so on. (Optional)

# IoT Communication model

## *Infrastructure services for enabling security and privacy*

- ***Trust of IoT infrastructure components*** (e,g. resolution/lookup, authorisation, or certification authority)
- ***Trust of IoT actors*** (IoT-service invokers and providers)
- ***Accountability of actions performed*** through the IoT
- ***Privacy for data*** handled by the infrastructure
- Privacy related ***to sensitive resources*** when they are provided to users
- ***Access control for resolution/lookup services***.
- Access control for ***IoT services, providing access to resources. Pseudonymisation of humans as well as IoT services*** (client and provider side); this also applies to the related virtual entity
- **Authentication of users.**
- **Confidentiality, integrity, and freshness of exchanged messages**. Freshness is a communication-security concept, implying that replay attacks cannot be performed.
- This has **implications for the authentication** of the endpoints of a communication path and the **encryption** used for communication.

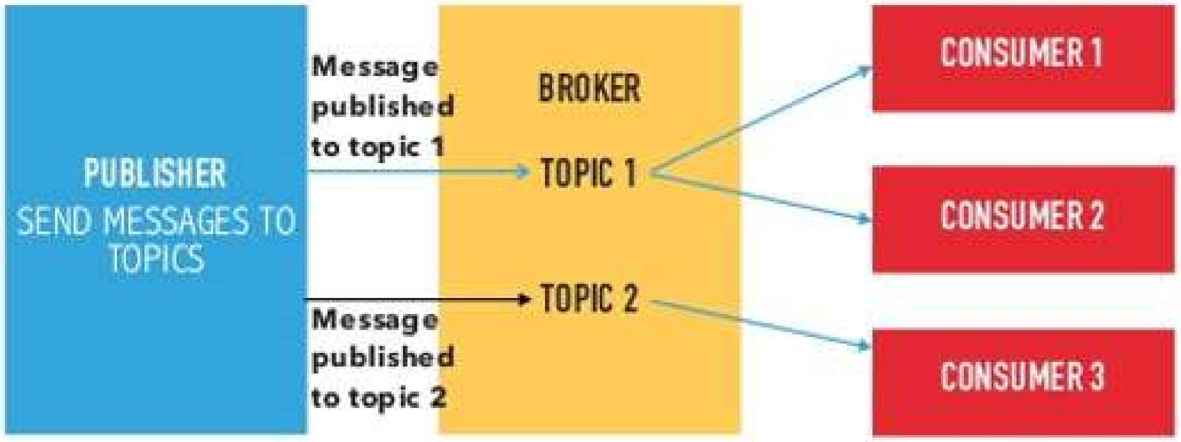
## IoT Communication model

Components of the security domain are:

- Authorisation (AuthS)
- Trust and reputation (TRA)
- Authentication (AuthN)
- Pseudonymisation (PN)
- Key exchange and key management (KEM)
- Certification authority (CA)

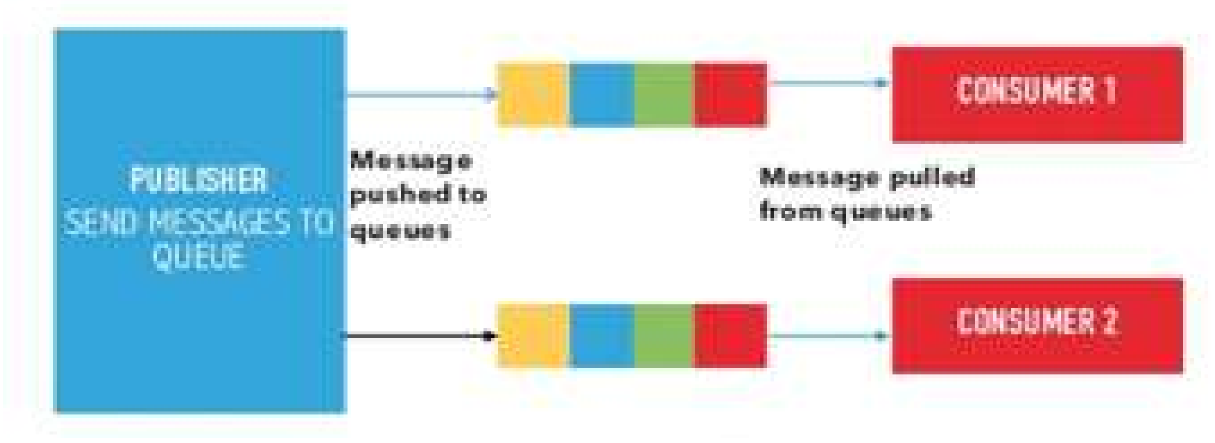
## Communication and security model

- **Publish-Subscribe** is a communication model that involves **publishers, brokers and consumers**.
- Publishers **are the source of data**. Publishers **send the data to the topics which are managed by the broker**.
- Publishers are **not aware of the consumers**. Consumers subscribe to the topics which are managed by the broker.
- When the **broker receives data for a topic from the publisher**, it sends the data to all the subscribed consumers.



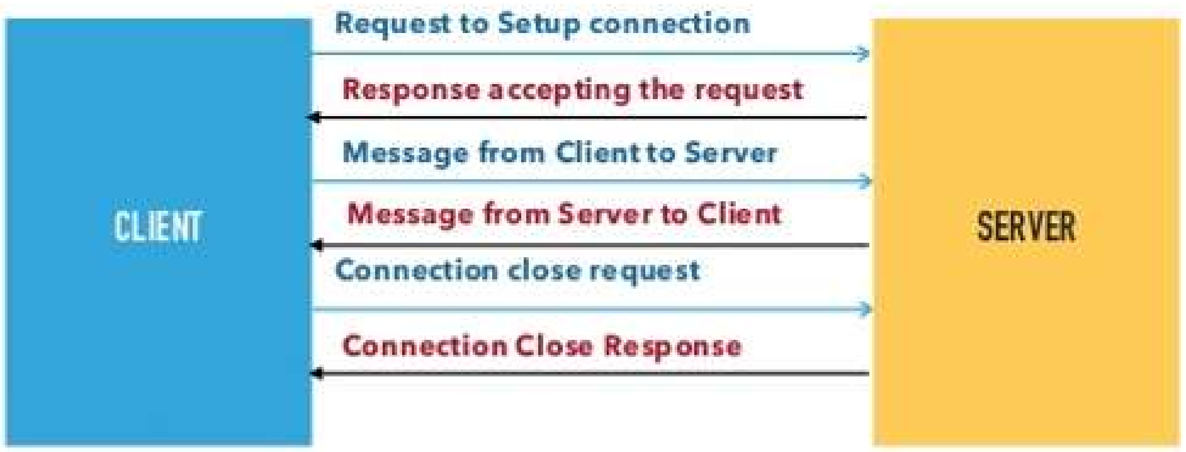
## Communication and security model

- **Push-Pull** : Push-Pull is a communication model in which **the data producers push the data to queues and the consumers pull the data from the queues**.
- **Producers do not need to be aware of the consumers**. **Queues help in decoupling the messaging between the producers and consumers**.
- Queues also act as **a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate rate at which the consumers pull data**.



# Communication and security model

- **Exclusive Pair** : Exclusive Pair is **a bi-directional, fully duplex communication model that uses a persistent connection between the client and server**.
- Once the **connection is setup it remains open until the client sends a request to close the connection**.
- **Client and server can send messages to each other after connection setup.**
- Exclusive pair is **a stateful communication model and the server is aware of all the open connections**.

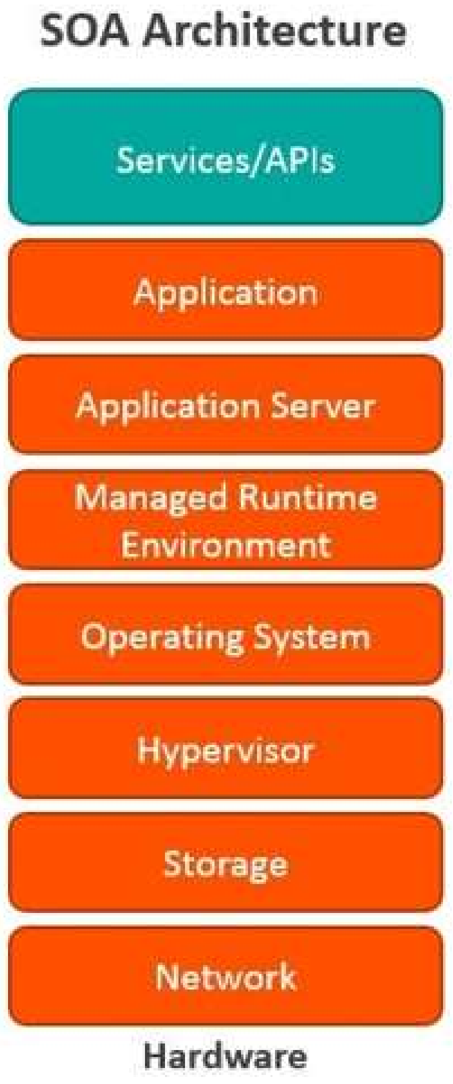

CLIENT
Request to Setup connection
Response accepting the request
Message from Client to Server
Message from Server to Client
Connection close request
Connection Close Response
SERVER

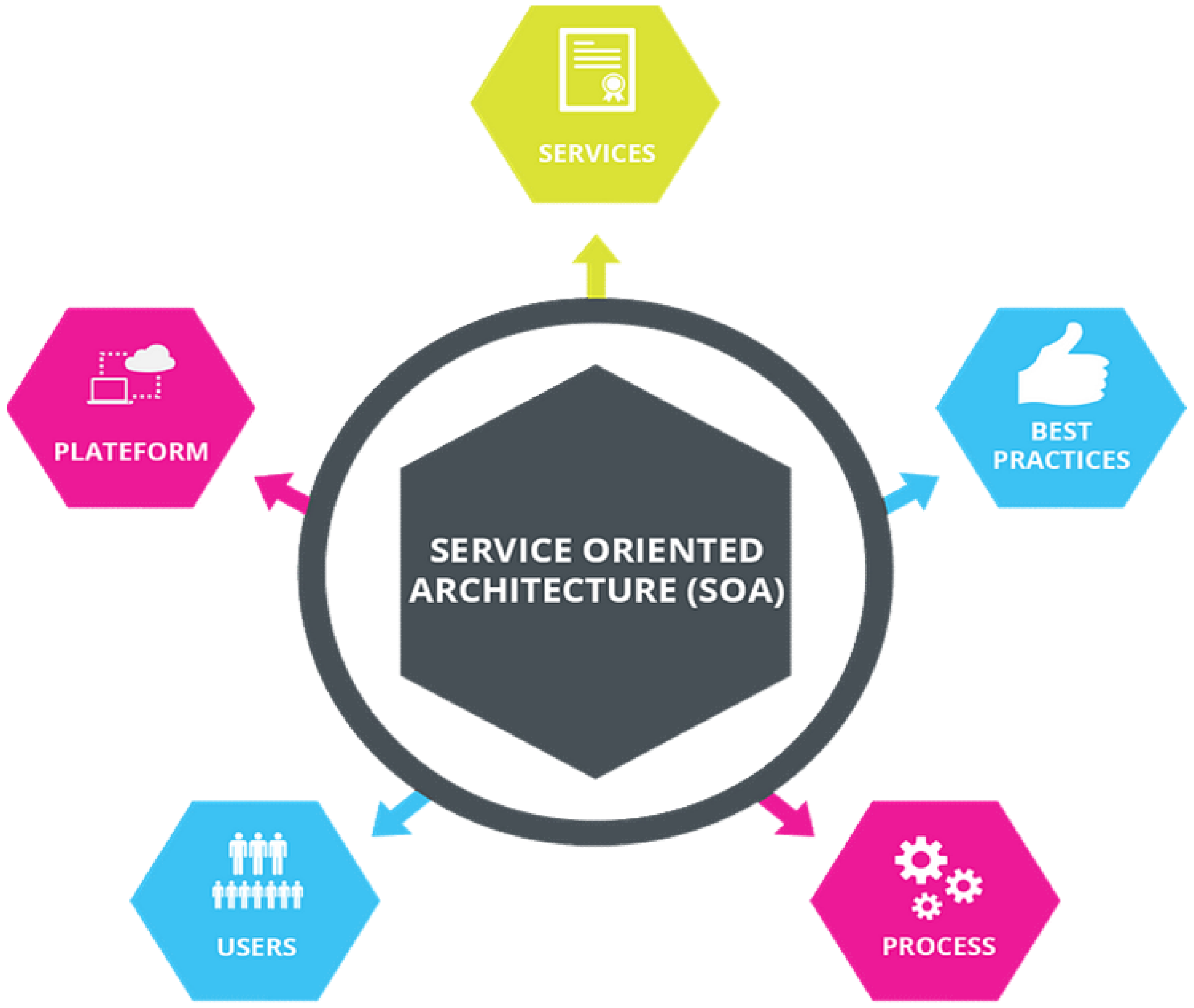
## Service-Oriented Architecture (SOA) based architecture

- SOA is *a style of software design where services are provided to the other components by application components, through a communication protocol over a network*. *Its principles are independent of vendors and other technologies*.
- In SOA, *a number of services communicate with each other, in one of two ways: through passing data or through two or more services coordinating an activity. This is just one definition of Service-Oriented Architecture.*

## SOA based architecture

Typically, Service-Oriented Architecture is implemented with web services, which makes the "functional building blocks accessible over standard internet protocols."





## Services

- A mechanism *enabling the provisioning of access to one or more capabilities described in service description*
- An Interface for the service providing the access to capabilities
- Applications or enterprise can subscribe on selection among number of services
- A service level agreement (SLA) binds the enterprise application and service.
- The access to each capability consistent with the constraints and policies
- *A collection of self contained, distinct and reusable components*
- Providing the *logically grouped and encapsulated functionalities*. Example: Traffic lights synchronizing service

## Types of Services

- There are four basic types of services:
  - **Functional service:** these define core business operations.
  - **Enterprise service:** these implement the functionality defined by the functional services
  - **Application service:** these are confined to specific application content
  - **Infrastructure service:** implements non-functional tasks such as authentication, auditing, security, and logging

## Characteristics Of Service-Oriented Architecture

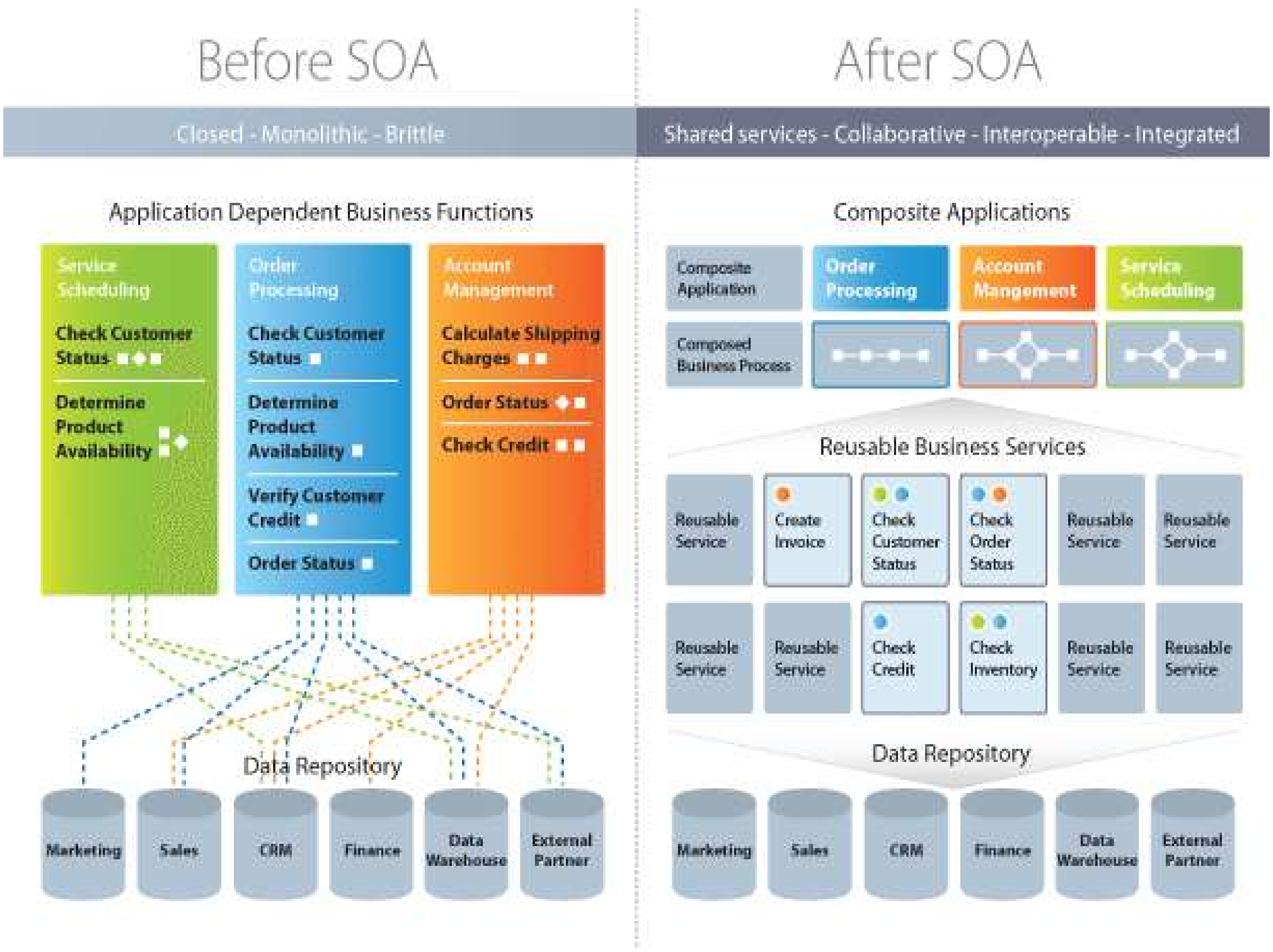
- While the defining concepts of ***Service-Oriented Architecture vary from company to company***. These core values include:
  - **Business value**
  - **Strategic goals**
  - **Intrinsic inter-operability**
  - **Shared services**
  - **Flexibility**
  - **Evolutionary refinement**

## Benefits Of Service-Oriented Architecture

- **Use Service-Oriented Architecture to create reusable code:** Not only does this cut down on time spent on the development process, but there's no *reason to reinvent the coding wheel every time you need to create a new service or process*. Service-Oriented Architecture also ***allows for using multiple coding languages because everything runs through a central interface.***
- **Use Service-Oriented Architecture to promote interaction:** With Service-Oriented Architecture, a standard form of communication is put in place, *allowing the various systems and platforms to function independent of each other*. With this interaction, Service-Oriented Architecture is also ***able to work around firewalls, allowing "companies to share services that are vital to operations."***

## Benefits Of Service-Oriented Architecture

- **Use Service-Oriented Architecture for scalability:** It’s important to be *able to scale a business to meet the needs of the client*, however ***certain dependencies*** can get in the way of that scalability. Using Service-Oriented Architecture ***cuts back on the client-service interaction, which allows for greater scalability***.
- **Use Service-Oriented Architecture to reduce costs:** With Service-Oriented Architecture, ***it’s possible to reduce costs while still “maintaining a desired level of output.”*** Using Service-Oriented Architecture ***allows businesses to limit the amount of analysis required when developing custom solutions.***


Before SOA
After SOA
Closed - Monolithic - Brittle
Shared services - Collaborative - Interoperable - Integrated
Application Dependent Business Functions
Service Scheduling
Check Customer Status
Determine Product Availability
Order Processing
Check Customer Status
Determine Product Availability
Verify Customer Credit
Order Status
Account Management
Calculate Shipping Charges
Order Status
Check Credit
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner
Composite Applications
Composite Application
Order Processing
Account Mangement
Service Scheduling
Composed Business Process
Reusable Business Services
Reusable Service
Create Invoice
Check Customer Status
Check Order Status
Reusable Service
Reusable Service
Reusable Service
Reusable Service
Check Credit
Check Inventory
Reusable Service
Reusable Service
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner

## Case Study
## IoT Design Methodology (Home Automation)
## based on
## IoT Architecture Reference Model (ARM)

***Text Book: "Chapter 5" - Arshdeep Bahga, Vijay Madisetti - Internet of Things_ A Hands-On Approach-Universities Press (India) Private Limited (2015)***

## IoT Design Methodology based on IoT Architecture Reference Model that includes:

- Purpose & Requirements Specification
- Process Model Specification
- Domain Model Specification
- Information Model Specification
- Service Specifications
- IoT Level Specification
- Functional View Specification
- Operational View Specification
- Device & Component Integration
- Application Development

## Introduction

- Designing IoT systems can be **a complex and challenging task** as these systems involve interactions between various components such as **IoT devices and network resources, web services, analytics components, application and database servers**.
- IoT system designers - often tend **to design IoT systems keeping specific products/services in mind**.
- IoT System design are **tied to specific product/service choices made**.
- IoT System designed in such away in **updating the system design to add new features** or **replacing a particular product/service choice for a component becomes very complex**, and in many cases may **require complete re- design of the system.**

## Introduction

- IoT system design are:

    **- independent of specific product, and**

    **- service or programming language.**

- IoT systems designed with the proposed methodology –

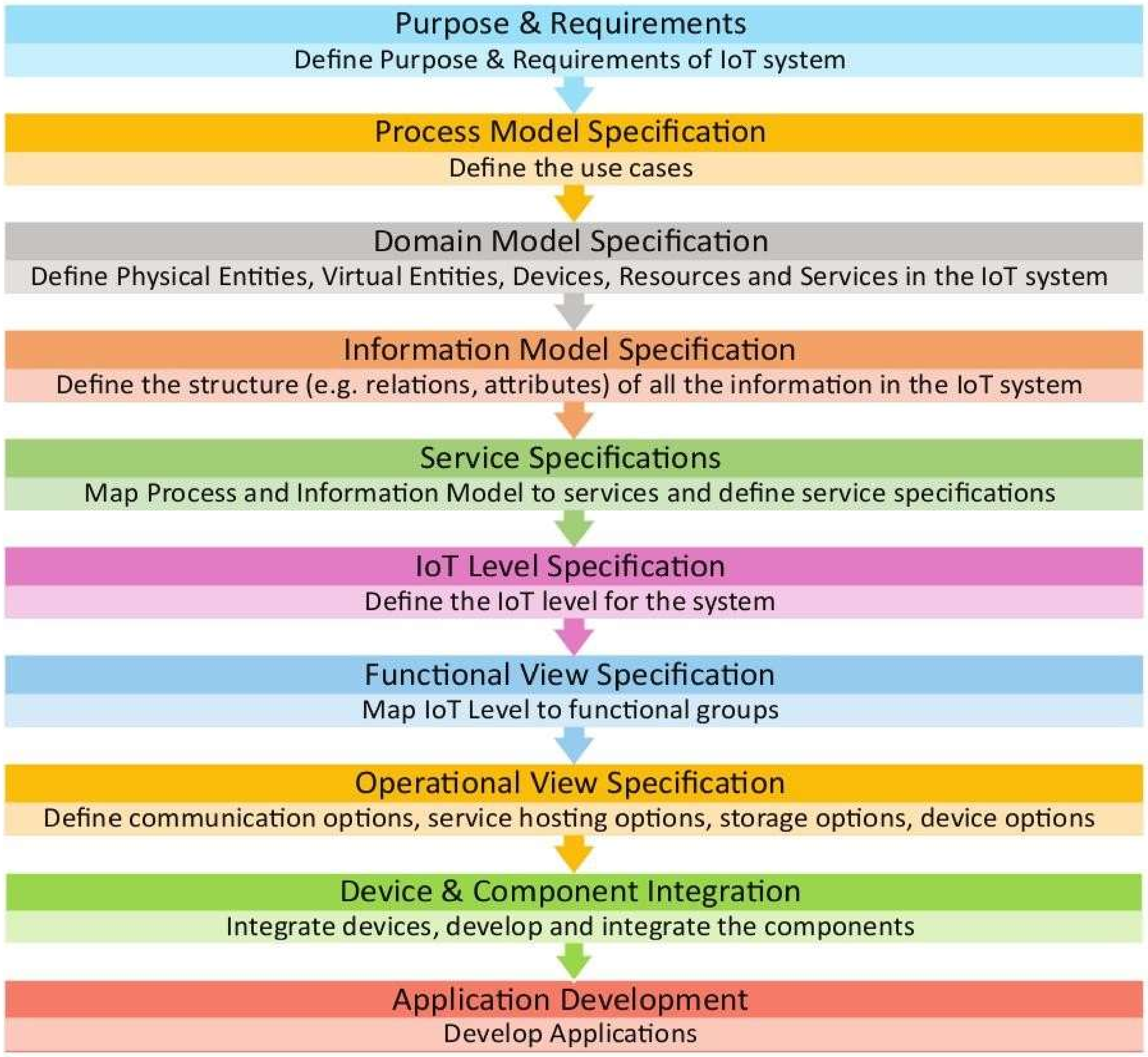
    **- reduced design,**

    **- testing and maintenance time,**

    **- better interoperability, and**

    **- reduced complexity.**

## IoT Design Methodology Steps


Purpose & Requirements
Define Purpose & Requirements of IoT system
Process Model Specification
Define the use cases
Domain Model Specification
Define Physical Entities, Virtual Entities, Devices, Resources and Services in the IoT system
Information Model Specification
Define the structure (e.g. relations, attributes) of all the information in the IoT system
Service Specifications
Map Process and Information Model to services and define service specifications
IoT Level Specification
Define the IoT level for the system
Functional View Specification
Map IoT Level to functional groups
Operational View Specification
Define communication options, service hosting options, storage options, device options
Device & Component Integration
Integrate devices, develop and integrate the components
Application Development
Develop Applications

## Purpose & Requirements Specification

**The first step in IoT system design methodology is to define the purpose and requirements of the system.**

**In this step, involves the following steps such as**

- **the system purpose,**
- **behavior and**
- **requirements such as** *(such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements, …)* **are captured.**

## Purpose & Requirements Specification

- **Purpose :** A **home automation system** that allows **controlling of the lights** in a home remotely using a **web application**.
- **Behavior :** The home automation system should have **auto and manual modes**. In **auto mode**, the system **measures the light level in the room and switches on the light when it gets dark**. In manual mode, the system provides **the option of manually and remotely switching on/off the light**.
- **System Management Requirement :** The system should provide **remote monitoring and control functions.**

## Process Specifications

- The second step in the IoT design methodology is **to define the process specification**.
- In this step, the use cases of the IoT system are **formally described based on and derived from the purpose and requirement specifications**.

## Process Specifications

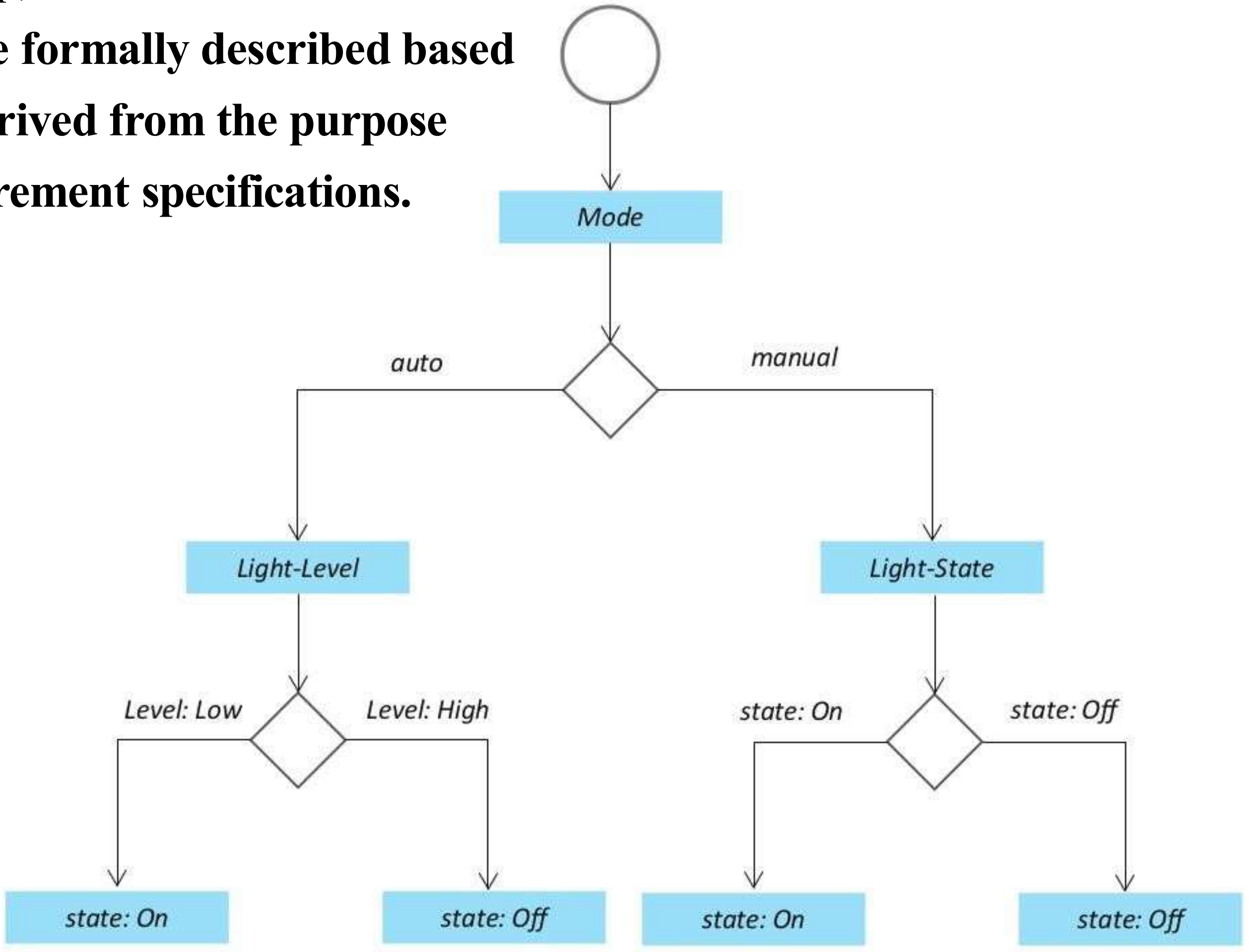
- **Purpose :** A **home automation system** that allows controlling of the lights in **a home remotely using a web application**.
- **Behavior :** The home automation system should have **auto and manual modes**. In auto mode, the system **measures the light level in the room and switches on the light when it gets dark**. In manual mode, the system provides the **option of manually and remotely switching on/off the light**.
- **System Management Requirement :** The system should provide **remote monitoring and control functions.**

## Process Specifications

- **Data Analysis Requirement :** The system should **perform local analysis of the data**.
- **Application Deployment Requirement :** The application should be **deployed locally on the device, but should be accessible remotely**
- **Security Requirement :** The **system should have basic user authentication capability**.

# Process Specifications

**In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications.**



## Domain Model Specification

- The third step in the IoT design methodology is to define the **Domain Model**.
- The domain model **describes the main concepts, entities and objects in the domain of IoT system to be designed.** Domain model **defines the attributes of the objects and relationships between objects**.
- Domain model provides **an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform**.

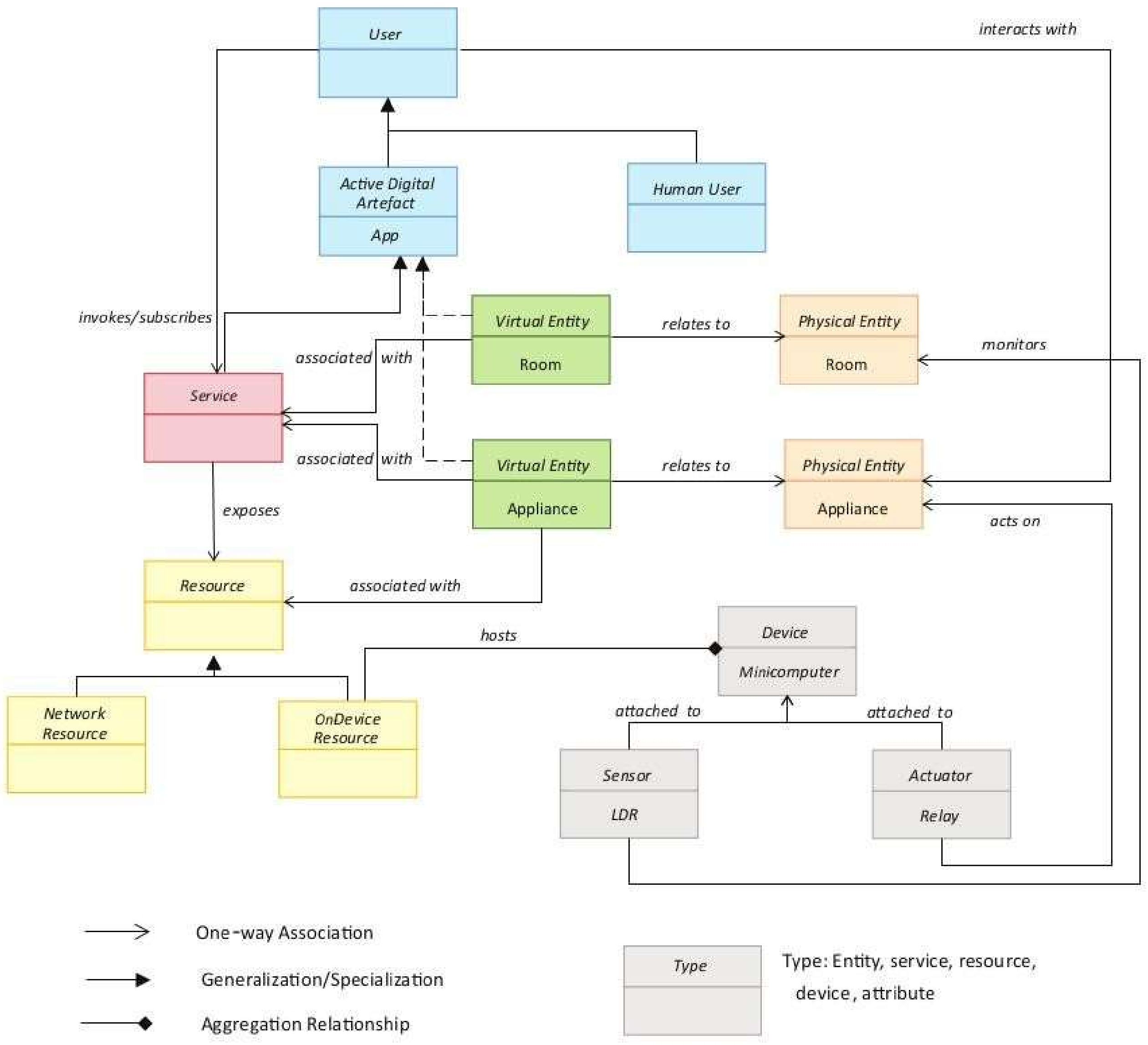
## Domain Model Specification

- The **entities, objects and concepts** defined in the domain model include:
- **Physical Entity :** Physical Entity is **a discrete and identifiable entity in the physical environment** (e.g. a room, a light, an appliance, a car, etc.).
- **Virtual Entity :** Virtual Entity is **a representation of the Physical Entity in the digital world**.
- **Device** provides **a medium for interactions between Physical Entities and Virtual Entities**. Devices are either attached to Physical Entities or placed near Physical Entities.

## Domain Model Specification

- **Resource :- Resources are software components which can be either "on-device" or "network-resources". On-device resources are hosted on the device and include software components that either provide information on or enable actuation upon the Physical Entity to which the device is attached.**
- **Service :** Services provide **an interface for interacting with the Physical Entity**. Services access the resources hosted on the device or the network resources to **obtain information about the Physical Entity or perform actuation upon the Physical Entity**.

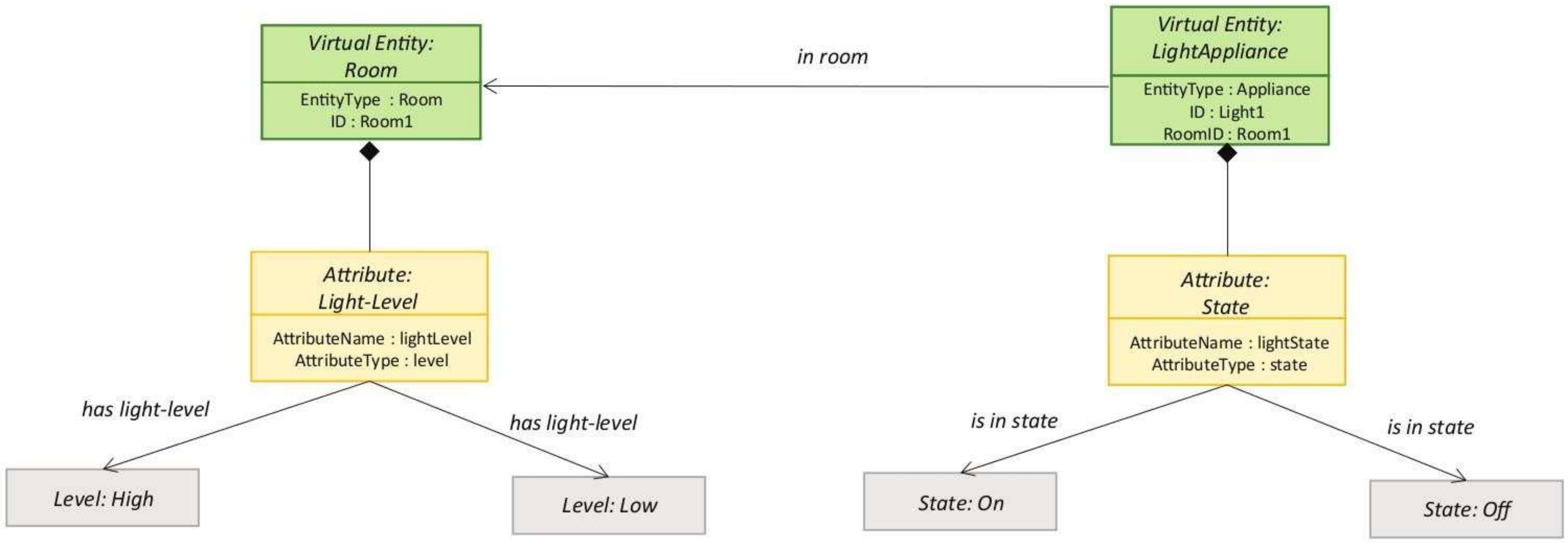
# Domain Model Specification



## Information Model Specification

- The fourth step in **the IoT design methodology is to define the Information Model**.
- Information Model **defines the structure of all the information in the IoT system**, for example, attributes of Virtual Entities, relations, etc.
- Information model does not **describe the specifics of how the information is represented or stored**.
- To define the information model, we first **list the Virtual Entities defined in the Domain Model**.
- Information model **adds more details to the Virtual Entities by defining their attributes and relations**

# Information Model Specification

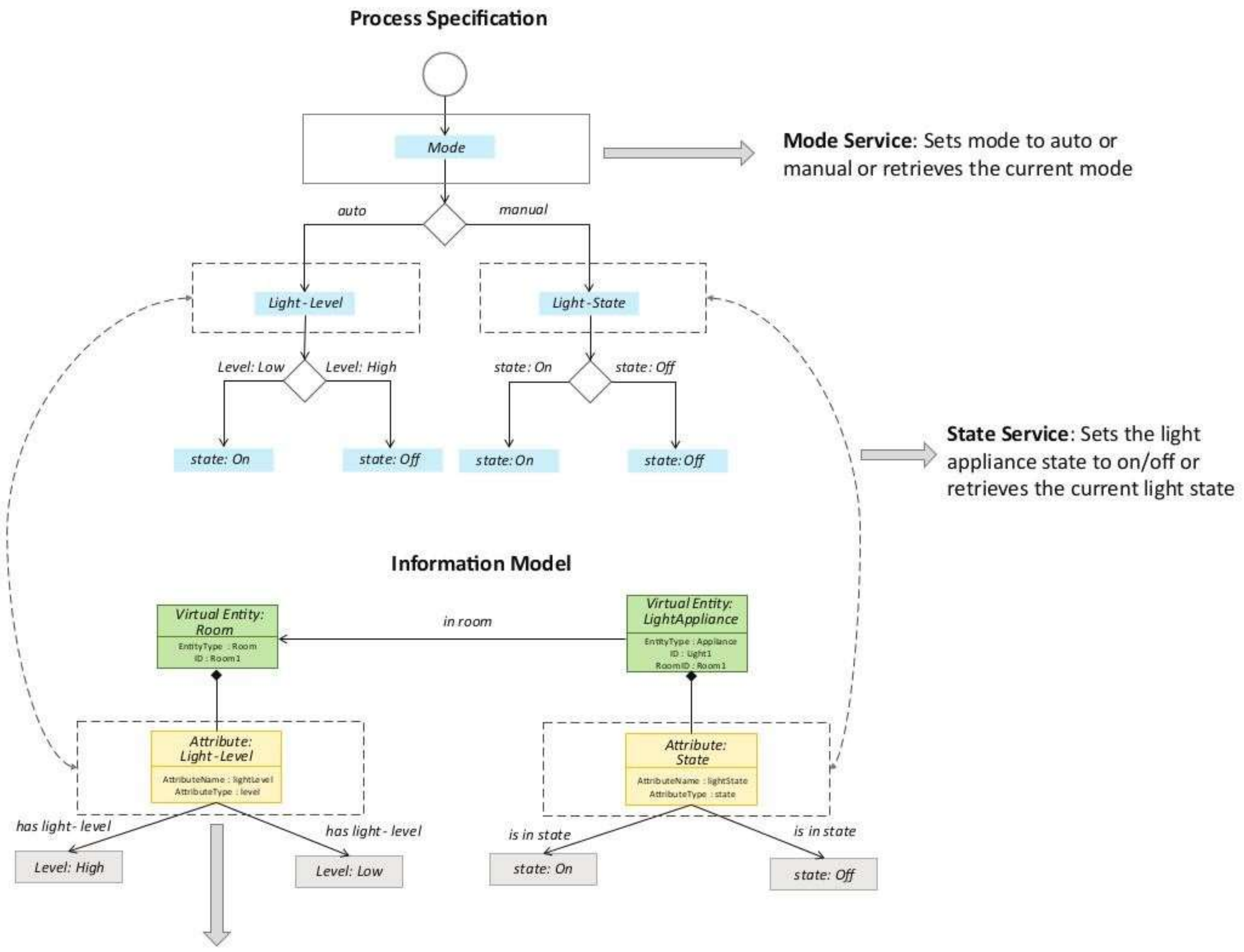


## Service Specifications

- The fifth step in the IoT design methodology is to **define the service specifications**.
- **Service specifications -define the services in the IoT system,**
  - **service types,**
  - **service inputs/output,**
  - **service endpoints,**
  - **service schedules,**
  - **service preconditions and**
  - **service effects.**

# Service Specifications

- From the process specification and information model, we identify the states and attributes.
- For each state and attribute we define a service.
- These services either change the state or attribute values or retrieve the current values.

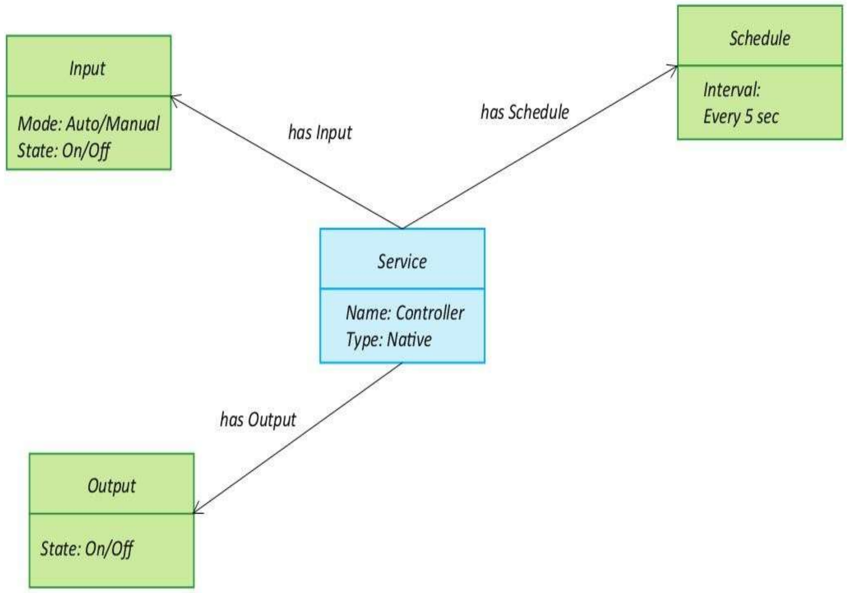


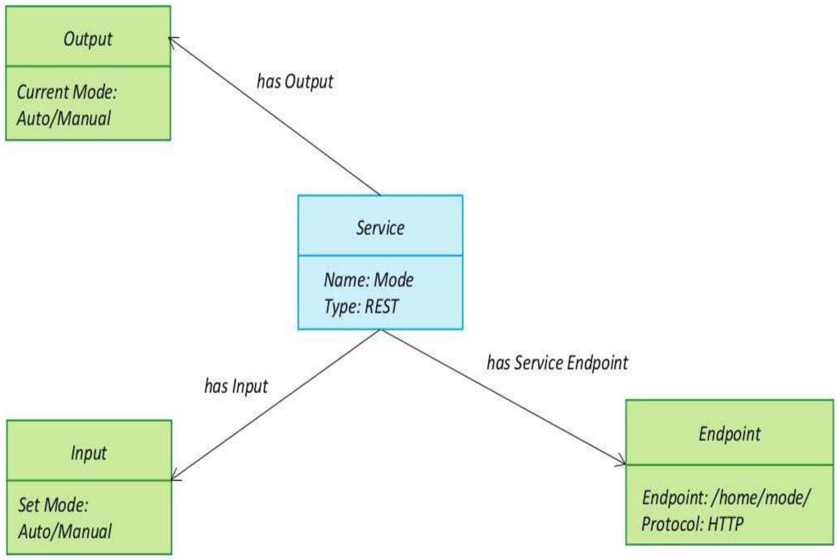
**Mode Service**: Sets mode to auto or manual or retrieves the current mode

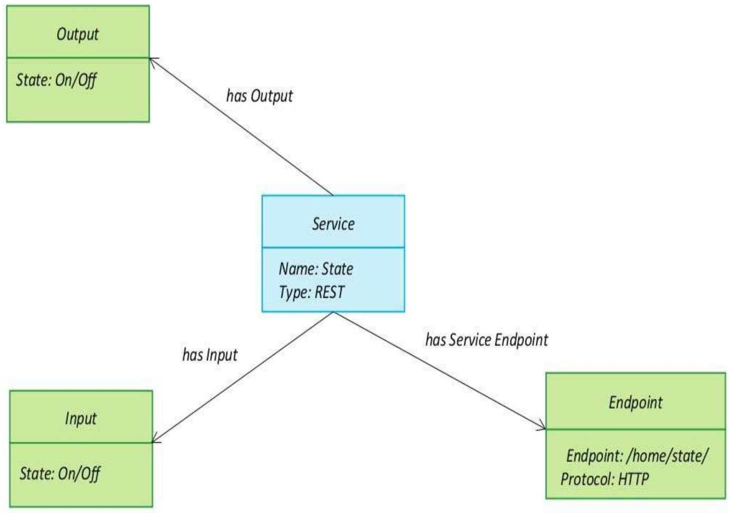
**State Service**: Sets the light appliance state to on/off or retrieves the current light state

**Controller Service**: In auto mode, the controller service monitors the light level and switches the light on/off and updates the status in the status database.
In manual mode, the controller service, retrieves the current state from the database and switches the light on/off.

# Service Specifications



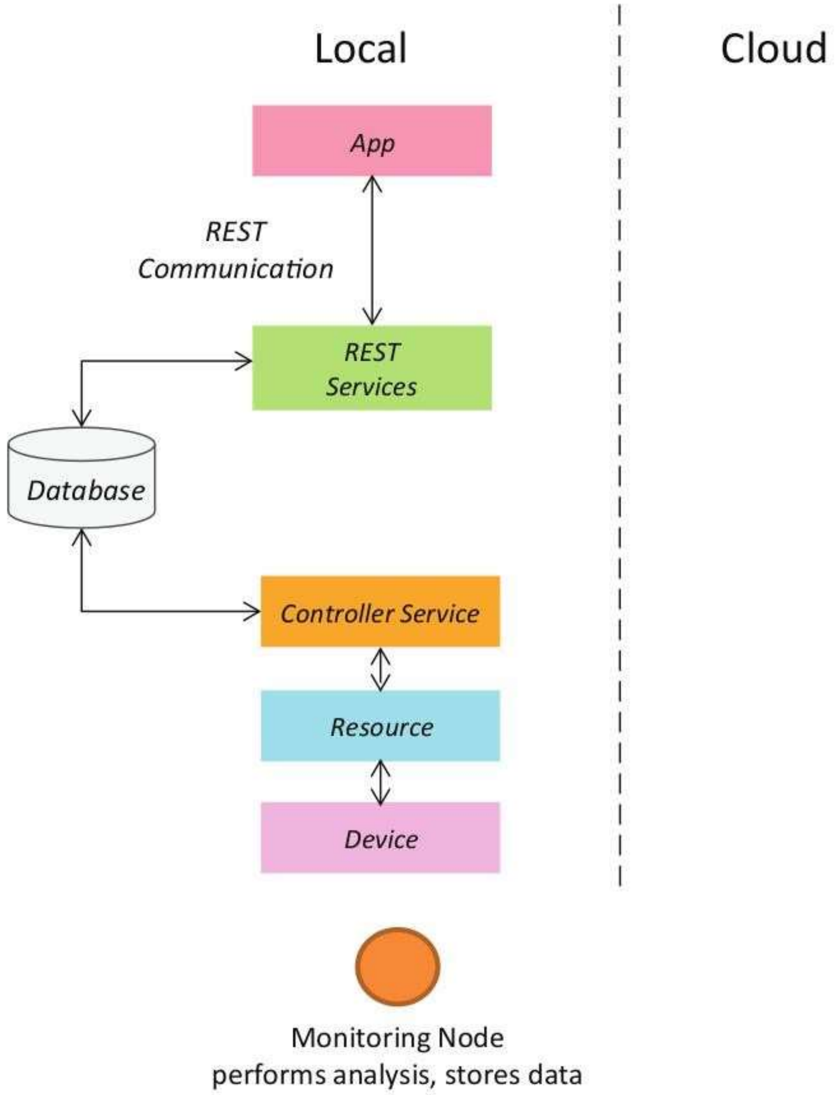




# IoT Level Specifications

- The sixth step in the IoT design methodology is to **define the at least one of the IoT level for the system**.

**Representational State Transfer (REST)** is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. **REST APIs follow the request-response communication model described in previous section**. The REST architectural **constraints**
**apply to the components, connectors, and data elements, within a distributed hypermedia system**.



## Functional View Specification

The **Functional View (FV)** defines **the functions of the IoT systems grouped into various Functional Groups (FGs).**

Each **Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts**.

## Functional View Specification

The Functional Groups (FG) included in a Functional View include:

**Device :** The device **FG contains devices for monitoring and control. In the home automation example**. the device FG includes a single board mini-computer, a light sensor and relay switch(actuator).

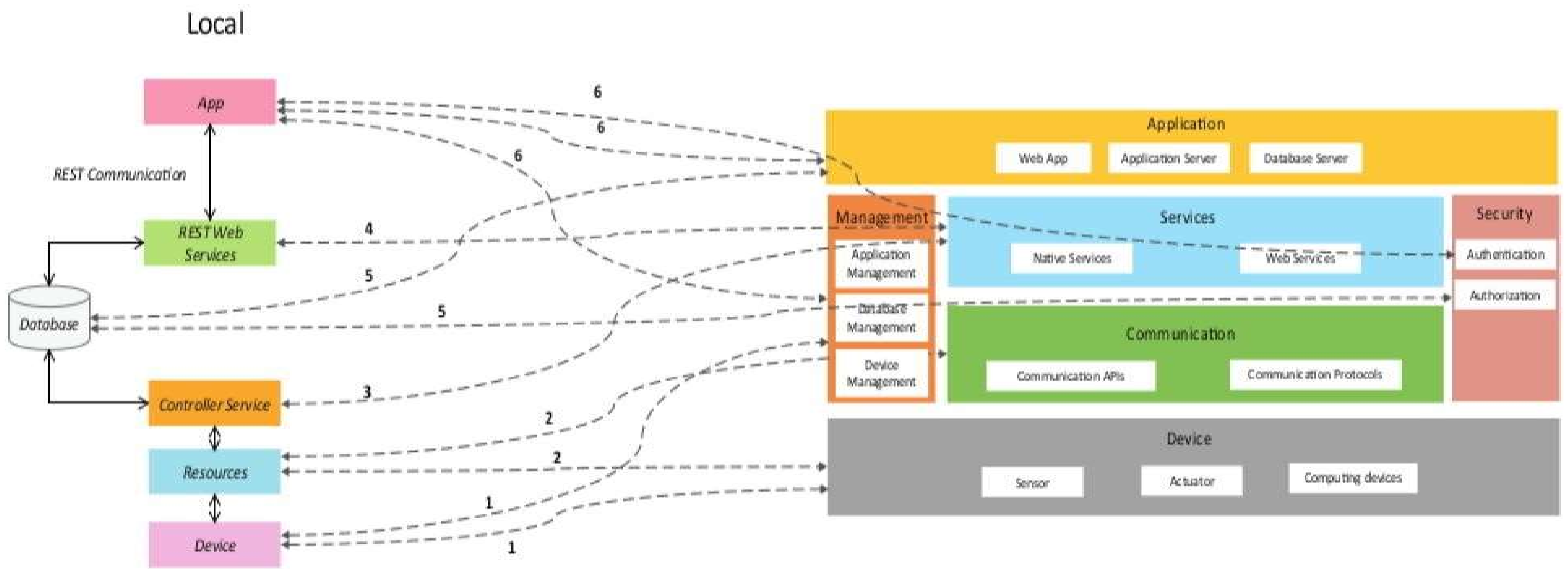
**Communication** : The communication **FG handles the communication for the IoT system**. The communication FG includes the **communication protocols that form the backbone of IoT systems and enable network connectivity**.

The communication FG also **includes the communication APIs (such as REST and WebSocket) that are used by the services and applications to exchange data over the network**.

## Functional View Specification

- **Services:** The service **FG includes various services** involved in the IoT system such as **services for device monitoring , device control services, data publishing services and services for device discovery**.
- **Management:** The management **FG includes all functionalities that are needed to configure and manage the loT system** .
- **Security :** The security **FG includes security mechanisms for the loT system such as authentication, authorization, data security, etc**.
- **Application :** The application FG includes **applications that provide an interface to the users to control and monitor various aspects of the IoT system**. Applications also allow users to **view the system status and the processed data**.

# Functional View Specification



1. IoT device maps to the Device FG (sensors, actuators devices, computing devices) and the Management FG (device management)

2. Resources map to the Device FG (on-device resource) and Communication FG (communication APIs and protocols)

3. Controller service maps to the Services FG (native service). Web Services map to Services FG (web services)

4. Web Services map to Services FG (web services)

5. Database maps to the Management FG (database management) and Security FG (database security)

6. Application maps to the Application FG (web application, application and database servers), Management FG (app management) and Security FG (app security)

## Functional View Specification – Major Functions

1. loT device maps to the Device FG (sensors, actuators devices, computing devices) and the Management FG {device management)
2. Resources map to the Device FG (on-device resource) and Communication FG (communication APls and protocols)
3. Controller service maps to the Services FG (native service).
4. Web Services map to Services FG (web services) Web Services map to Services FG (web services)
5. Database maps to the Management FG (database management) and Security FG {database security)
6. Application maps to the Application FG (web application, application and database servers), Management FG (app management) and Security FG (app security)

## Operational View specifications

- In this step, various options pertaining to the loT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc.
- Operational View specifications for the home automation example are s follows:
- **Devices:** Computing device (Raspberry Pi), light dependent resistor (sensor), relay switch (actuator).
- **Communication APIs:** REST APIs
- **Communication Protocols:** Link Layer - 802.11, Network Layer - 1Pv4/1Pv6, Transport TCP, Application - HTTP.

## Operational View specifications

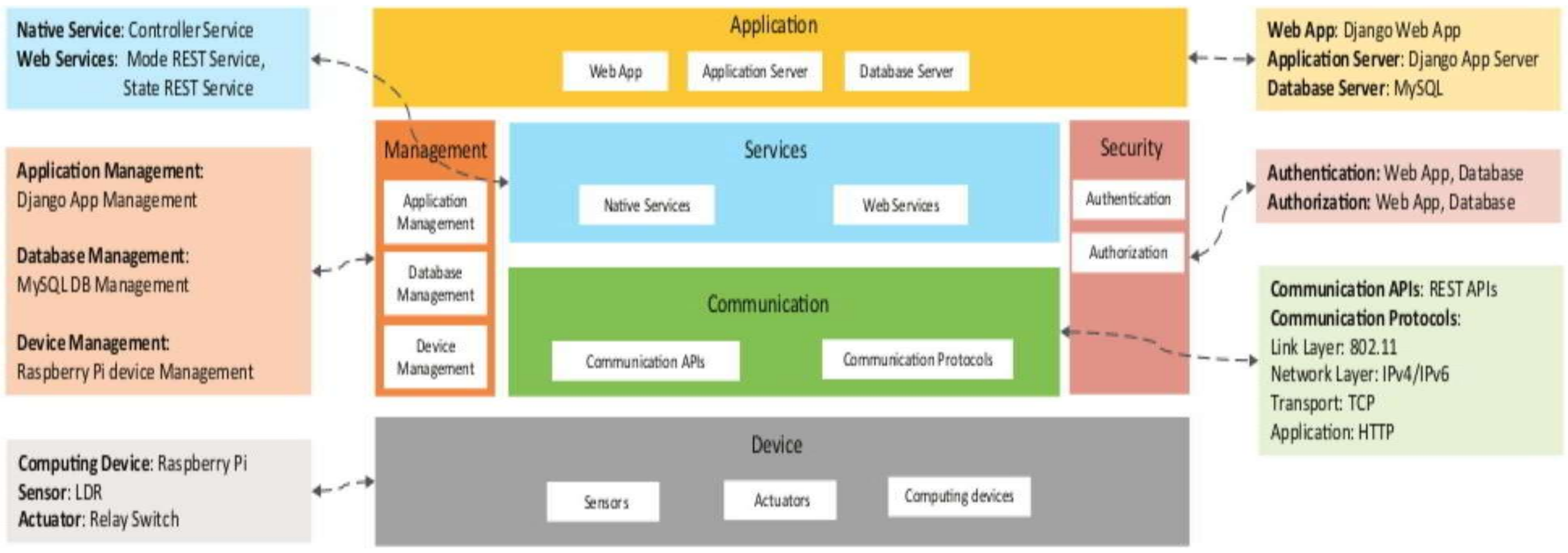
- Operational View specifications for the home automation example are as follows:
- **Services:**
  - **Controller Service** - Hosted on device, implemented in Python and run as a native service.
  - **Mode service** - REST-ful web service, hosted on device, implemented with Django-REST Framework.
  - **State service** - REST-ful web service, hosted on device, implemented with Django-REST Framework.
- **Application:**
  - Web Application - Django Web Application, Application Server - Django App Server, Database Server - MySQL.

## Operational View specifications

Operational View specifications for the home automation example are as follows:

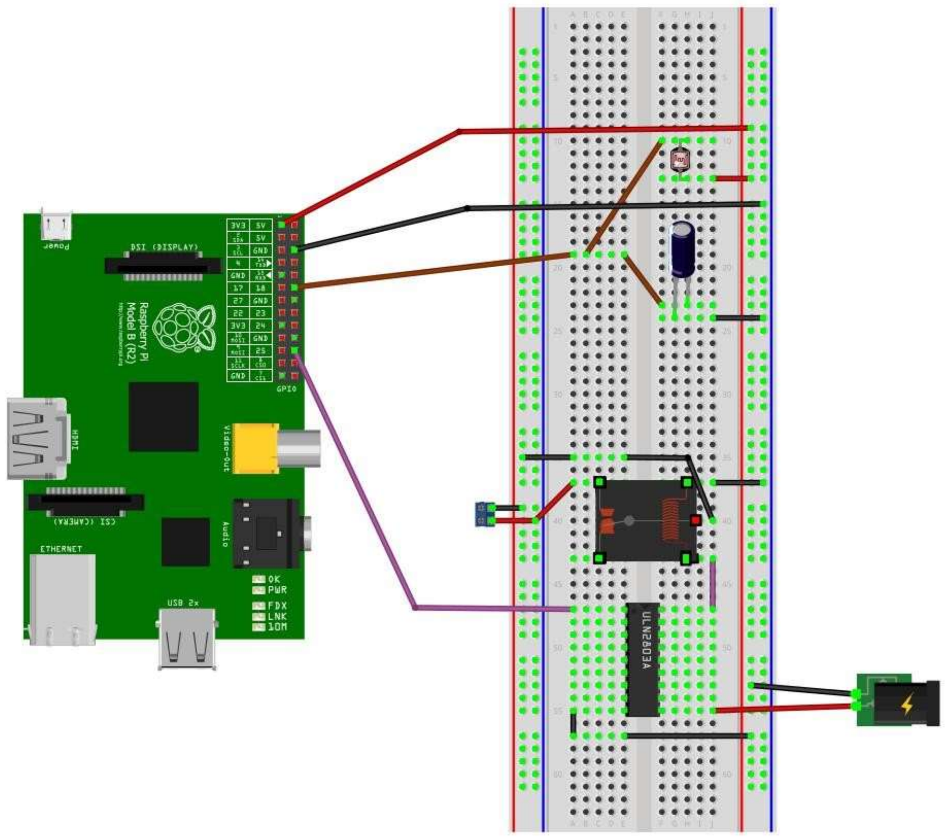
- **Security:**
  - **Authentication:** Web App, Database Authorization: Web App, Database
- **Management:**
  - Application Management - Django App Management Database Management - MySQL DB Management, Device Management - Raspberry Pi device Management.

## Operational View specifications


**Native Service**: Controller Service
**Web Services**: Mode REST Service, State REST Service
Application
Web App
Application Server
Database Server
**Web App**: Django Web App
**Application Server**: Django App Server
**Database Server**: MySQL
**Application Management**: Django App Management
**Database Management**: MySQL DB Management
**Device Management**: Raspberry Pi device Management
Management
Application Management
Database Management
Device Management
Services
Native Services
Web Services
Security
Authentication
Authorization
**Authentication**: Web App, Database
**Authorization**: Web App, Database
Communication
Communication APIs
Communication Protocols
**Communication APIs**: REST APIs
**Communication Protocols**:
Link Layer: 802.11
Network Layer: IPv4/IPv6
Transport: TCP
Application: HTTP
Device
Sensors
Actuators
Computing devices
**Computing Device**: Raspberry Pi
**Sensor**: LDR
**Actuator**: Relay Switch

## Device and Component Integrations

**The ninth step in the IoT design methodology is the integration of the devices and components.**

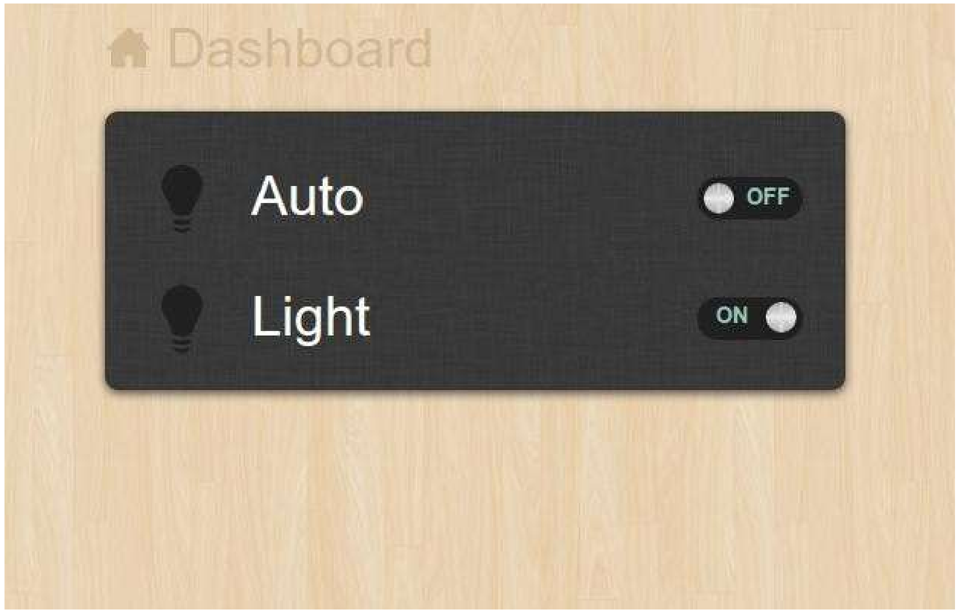


## Application Development

The application has controls for the **mode (auto on or auto off) and the light (on or off)**.

In the auto mode, the IoT system **controls the light appliance automatically based on the lighting conditions in the room.**

- When auto mode is enabled the light control in the application is disabled and it reflects the current state of the light.
- When the auto mode is disabled, the light control is enabled and it is used for manually controlling the light.


Dashboard
Auto
OFF
Light
ON